Adaptive Streaming On Heterogeneous Networks

Massimo Bernaschi
National Research Council
massimo@iac.cnr.it

Filippo Cacace,
Raffaele Clementelli
Università Campus
Bio-Medico
f.cacace@unicampus.it
raffaele@raffasoft.it

Luca Vollero
Consorzio Nazionale CINI
vollero@ieee.org

ABSTRACT

Specific network protocols, like MobileIP, offer seamless connectivity to mobile systems. However, the QoS requirements of streaming applications and video conferencing systems require an approach that spans across different layers of the network stack. In this paper we study how to integrate an efficient method for vertical handoff and adaptation support for multimedia streaming through heterogeneous networks. We also present experimental results obtained with our prototype on IEEE 802.11 and UMTS networks.

Categories and Subject Descriptors: C.2.2 [Network Protocols]:Applications; H.3.4 Systems and Software: Performance evaluation.

General Terms: Design, Performance.

Keywords: Mobility, Adaptivity, Multimedia Streaming

1. INTRODUCTION

The growing demand to support continuous connectivity for mobile devices has led to the definition of mobility protocols (like Mobile IP [1] and Mobile IPv6 [2]) at network level. Even if these protocols are not widely deployed yet, the characterization of their performance and usability is an active field of research. A key point for a successful deployment of mobility protocols is the capability of supporting streaming (such as video-on-demand) and real-time applications (such as VoIP or videoconferencing).

In order to enhance the Quality of Service (QoS) for mobile users, it is necessary to guarantee [3]: (i) seamless connectivity across heterogeneous networks, (ii) application adaptability to optimize the end users' perceived quality, and (iii) flexibility with respect to network features such as bit rate, wireless channel errors and congestion.

When a mobile device roams across heterogeneous networks such as 802.11 WLAN or 3G cellular networks, it may experiment variations in bit rate, delay, *etc.*, significantly wider than in stable operating environments. For example, the bit rate ratio between WLAN and cellular networks is about 10 and this difference may cause streaming applications to freeze, due to network congestion, when the receiving device moves to a network with a lower bit rate. Adaptation capability is thus a key component of streaming applications for mobile devices, since the side effects of the network technology change may be much more disruptive than the short break in data delivery due to the handoff. In order to be suitable to mobile devices, the adaptation mechanism should not require too much computational power or buffering space. Moreover any solution should not rely on special features of the core network.

In this paper we present an approach that integrates mobility on heterogeneous networks and streaming adaptation taking into account the above requirements. We have implemented a cross-layer method to improve vertical handoff latency and packet loss [27]. We study how this mechanism can also be used to support video adaptation. The adaption method is based on the change of the streamed video, a process that we denominate "session handoff". A finer-grained adaptation based on transcoding is used on the server-side to handle possible fluctuations in bit rate, due to concurrent traffic and channel errors, typically found on wireless networks. One advantage of our approach is that it does not require changes to network infrastructures and it is compatible with legacy multimedia applications on the client side.

The main contributions of the paper are a cross-layer approach that spans from data-link to session level, the introduction of a double adaptation mechanism to cope with different adaptation requirements, experimental results collected in a working test-bed. The rest of the paper is organized as follows: in Section 2 we survey adaptation techniques for streaming applications. In Section 3 we present our adaptive architecture for mobile devices. In Section 4 we discuss implementation issues and results of our experiments. These results are compared to other approaches in Section 5. Section 6 concludes the paper.

2. BACKGROUND AND RELATED WORK

2.1 Video Adaptation

A comprehensive framework for video adaptation techniques can be found in [18]. Any approach for streaming adaptation, both in the unicast and in the multicast case, can be classified according to two features: *i) how* the adaptation is performed, and *ii) who* is responsible for the adaptation procedure. As for the first point, the proposed ap-

WMuNeP'05, October 13, 2005, Montreal, Quebec, Canada.

proaches can be labeled as follows:

2.1.1 Single stream adaptation.

In this approach, the rate at which the stream is sent, is dynamically tuned by means of an adaptive protocol, a congestion control mechanism, and/or a buffer control mechanism on the client side. Note that traditional congestion avoidance schemes such as TCP Additive-Increase/ Multiplicative-Decrease (AIMD) cause large oscillations in the transmission rate that degrade the user-perceived quality of the video stream. An analysis of buffering for quality adaptation with non-AIMD congestion control and layered streams is reported in [8]. Other protocols of this class include RAP [4], SCP [6] and VTP [7]. A major limitation of this approach is that the bit rate determined by the congestion mechanism must match the intrinsic transmission rate of the stored video.

2.1.2 Multiple streams adaptation.

In this case the client receives distinct streams when the network conditions change. There are several alternatives in order to change the stream: *i)* to switch between multiple copies pre-encoded at different resolutions [9]; *ii)* to perform transcoding [5, 15], *iii)* to use frame-dropping [10]. The first solution requires less computing power and more storage resources whereas the opposite happens with transcoding. Frame-dropping requires a form of priority-mapping among the elements of the streamed video.

2.1.3 Layered adaptation.

In this case the video content is compressed into some non-overlapping streams, or layers. There is a *base layer*, which contains the most important features of the video, whereas additional *enhancement layers* contain data that progressively refine the reconstructed video quality. The layers can be decoded either independently [11] or cumulatively [13, 12]. A bandwidth penalty is incurred by encoding a video stream in layers.

Layered adaptation has been regarded, in general, as the best mechanism, at least in the multicast case. However, in [14] the authors show that there are situations where multiple streams may be a better solution, since layered adaptation not only consumes more bandwidth but also requires more buffering and synchronization capabilities. Hybrid approaches that combine the basic adaptation techniques have been proposed as well: [19] proposes a combination of transcoding and frame dropping, whereas [22] describes an architecture that uses stream switching and layering.

As for the second point (*who* adapts), the adaptation mechanism can be triggered: *i)* by an adaptive protocol that includes a congestion detection and avoidance mechanism; *ii)* at application level, through a feedback channel; *iii)* by means of a cross layer integration/notification mechanism. As an example of the first approach, the Video Transport Protocol (VTP) [7] is able to perform congestion detection and to reduce both the sending rate and the video encoding rate to a level the network can sustain. In [21] Yu *et al.* propose an application layer QoS control protocol, called QCP, to control packet delay, jitter and packet loss of video traffic. In [22] a video streaming architecture that relies on the standard RTCP protocol as a feedback channel is proposed. The adaptive modules are located on the server side only. This choice allows any existing client application to access the service. A general framework for cross layer video adaptation over wireless networks is presented in [20], where both network and end-to-end based solutions are analyzed. [16] proposes an end-to-end architecture for video adaptation that requires the introduction of three components on the server side: Quality Adaptation (QA), Rate Adaptation (RA) and Error Control (EC). In [19] Lei and Georganas propose a cross layer schema specific for wireless channels. The Automatic Repeat Request (ARQ) error control is used along with a feedback channel to estimate the current channel state. Such link layer information allows to perform video adaptation by means of a combination of transcoding and frame dropping.

2.2 Adaptation and Mobility

Inoyue *et al.* [6] present a cross layer architecture, where no layer attempts to hide aspects of mobility from other layers. The application layer is in charge of adapting a scalable content, whereas the adaptation library determines the size of change. Network changes are detected by the operating system, and since Mobile IP is not used, the network change requires the reconfiguration of the application, leading to poor handoff performance. A similar proposal [17] resorts to Mobile IP for mobility management at network level: multiple paths are maintained while a mobile node transits the overlapping area of two adjacent cells. A source adaptive multi-layer schema is used to perform adaptation: base layer is transmitted to each path, whereas enhancements layers are transmitted only through paths with enough bandwidth.

To the best of our knowledge, our proposal is the first to address the integration of MobileIPv6 and video adaptation in the context of vertical mobility. This requirement poses a specific challenge: the differences in bandwidth are potentially wider than in the horizontal mobility case.

3. ADAPTIVE STREAMING ARCHITECTURE

Our proposal is based on a multilayer integration of the following components that we briefly describe below:

- at network level, Mobile IPv6 [2] to transparently manage mobility;
- at transport level, RTCP (in combination with RTP) [25] to set up a feedback channel in charge of monitoring the delivery of the stream;
- at session level, the Real Time Streaming Protocol (RTSP) [23] and the Session Description Protocol (SDP) [24] to manage streaming sessions.
- on the server side, an efficient transcoding mechanism for fine-grained adaptation [15].

3.1 Mobile IPv6

Mobile IPv6 provides network level mobility for mobile nodes roaming across different subnets without disrupting sessions at transport level. When a mobile node (MN) is away from its home subnet, a router on the home subnet, known as *home agent* (HA), keeps track of the current binding of the mobile node. When an handoff takes place, the mobile node sends a *Binding Update* (BU), indicating its new "care-of-address", both to the home agent and to any node with which it is communicating, usually indicated as

Correspondent Node (CN). The application running on the mobile node always uses the home address to perform network operations and is unaware of any change at network level. Mobile IPv6 MNs use Router Advertisements (RAs) sent by IPv6 routers to detect mobility events. We implemented an improved schema for the handoff triggering phase based on link-layer monitoring.

3.2 RTCP

The Real Time Control Protocol (RTCP) defines the control channel of a Real Time Protocol (RTP) stream. It is based on the periodic transmission of control packets to all participants in a session. In most implementations RTCP packets are sent through a reliable transport protocol like TCP. RTCP feedback on the quality of distribution serves several purposes: *i)* to avoid congestion; *ii)* to control adaptive encoding; *iii)* to diagnose faults in the distribution.

3.3 RTSP and SDP

The Real-Time Streaming Protocol (RTSP) is a client-server presentation protocol that controls single or multiple time-synchronized streams of continuous media such as audio and video. There is no notion of RTSP connection in the protocol. Instead, an RTSP server maintains a session labeled by an identifier to associate groups of media streams and their states. During a session, a client may open and close many reliable transport connections to the server to issue RTSP requests for that session. TCP is the preferred protocol to send RTSP requests and responses. The set of streams to be controlled in an RTSP session is defined by a presentation description, usually specified through the SDP protocol.

A resource specified by an URI may aggregate more than one media object (for instance, it may contain the audio and video part of the same clip). RTSP allows to specify at session setup whether the control on the resource is *aggregated* or not. In the first case, subsequent requests (such as PLAY, PAUSE and TEARDOWN) apply to all the media objects contained in the resource; in the second case individual control is possible.

The Session Description Protocol (SDP) may be used to describe streams or presentations in RTSP. For instance, it is possible to specify via SDP the bit rate of each individual stream that is part of a session.

3.4 Server-side adaptation

The approach we followed for fine-grain adaptation is based on the schema presented in [15], where a method with low computational cost for real-time video transcoding is described. The processing efficiency of the method is crucial to fulfill the real-time constraints of video streaming.

Fine-grain adaptation can be triggered either at server or at client side by the RTCP feedback channel information. We resort to server side triggering since the server must, in any case, reduce the video quality when the available bit rate changes.

3.5 Performing streaming adaptation

In order to provide access to a clip containing audio and video, a coarse-grained adaptation mechanism may use different versions of the video part, encoded, say, at 50, 200, 1000 and 5000 kbps, along with a fixed audio encoding. These resources can be included in a single SDP session descriptor by specifying the type, name and bit rate of each resource. Then, a RTSP/SDP combination offers the possibility of realizing stream switching without resorting to *custom* mechanisms. The SDP information can be used on the client-side to choose the most suitable version of the video part, depending on the kind of network available at session setup time. When a vertical handoff takes place, the client can issue RTSP PAUSE/PLAY requests to stop the sending of the previous video stream and start a new video stream, with the appropriate bit rate. We call this action *session handoff*, since it is the session level response to a network handoff. A fine-grain adaptation mechanism is still needed in a wireless network where the available bit rate fluctuates when the signal weakens or other clients join a cell of the network. Luckily, in this case, the adaptation requirements are limited and the transcoding provided by the server (described in 3.4) is sufficient [7, 15].

3.6 Mobility Management Module

In general roaming across heterogeneous networks entails "up" and "down" vertical handoffs[1].An "up" handoff is a forced change from a faster network to a slower network with wider coverage (for example from a WLAN to a cellular network). It causes packet loss during the handoff detection and execution.Conversely, a "down" handoff is toward a preferred network when more than one network is available (for example from cellular network to a WLAN). Since the mobile device is connected through more than one interface at the same time, there is no packet loss. However, due to different RTT in the access link, packets may arrive out of order [26].

Multimedia streaming to a device moving across heterogeneous networks suffers from *i)* packet loss during up handoffs; *ii)* possible link congestion after a up handoff due to bandwidth disparity. The second point is more relevant since it can possibly block the stream reproduction. As for the first point, vertical handoff performance requires an approach that is different from the horizontal case [27].

Our approach is based on the Mobility Management module (MM), shown in Fig. 1, in charge of: *i)* timely detecting vertical handoff conditions; *ii)* informing the session and application layers about mobility events. The basic idea is to monitor the status of the interface at link-layer in order to trigger the handoff without resorting to RAs (see [27] for more details).

The *Event_Handler* runs in user-space and manages events read from an *Event Queue*, where events are inserted by modules (handlers) in charge of monitoring all the network interfaces. Each handler runs in user-space as a thread of the *Event_Handler* and resorts to specific *ioctl* calls to monitor the status of a single network interface. At this time, the prototype runs in the Linux environment and is able to manage Ethernet, IEEE 802.11b/g, and GPRS/UMTS interfaces. The *Event_Handler* can either trigger a vertical or horizontal handoff (that is, a change of interface or link) or configure an idle interface to prepare a possible handoff. In any case, the resulting command is passed to the MIPL Mobile IPv6 implementation [30] for the execution.

The handoff decision policy is contained within the *Event_Handler* module. A general solution to the network selection problem must take into account several factors [28], like

[1]The terms "up" and "down" are used referring to a hierarchy of overlaid wireless networks.

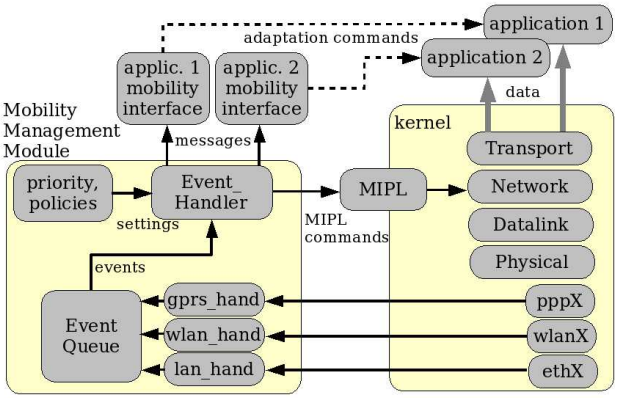

Figure 1: Software architecture of the MM

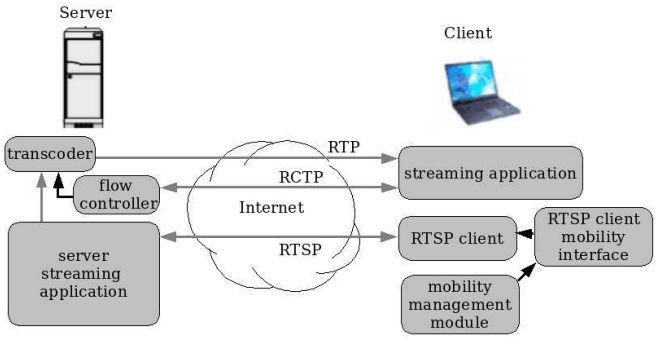

Figure 2: Cross-layer integration mechanism

signal strength, network security and condition, application type, user preferences for QoS, cost, etc. As a result, different algorithms can be implemented in the *Event_Handler* module. Our current implementation uses a variant of the classic algorithm based on 2 thresholds (*warning* and *alarm*) for the Radio Signal Strength (RSS) [29]. The handoff from a preferred interface is triggered when the RSS drops below the alarm threshold, whereas the handoff toward a preferred interface is triggered when the RSS rises above the warning threshold. Two additional parameters are introduced to avoid ping-pong effects and spurious handoffs: a *dwell timer*, that determines the minimal interval in which the RSS must be above the warning threshold before the handoff is triggered, and am *alarm timer*, that has the same meaning when the RSS drops below the alarm threshold. These four parameters determine the handoff execution policy. A "prompt migration" toward better connections improves packet loss and handoff delay at the expenses of throughput and ping-pong risks, whereas the opposite is true for a "delayed migration" approach. Since applications can register their own mobility interface as "plugins" of the MM, the value of the thresholds and timers can be dynamically adjusted by the MM according to the type of service requested by the applications. We leave for future work the issue of determining the best parameters estimation in the case of streaming applications, and assume that the *Event_Handler* executes the "two threshold" algorithm with the values of Table 1. This particular choice determines a "prompt migration" policy.

The mobility interface of each application receives only messages related to the kind of events to which the application is interested, and may instruct the application to change its behavior. This change is application dependent (for example, a streaming application could change the buffer size, or a VoIP application could switch to a different voice codec). In Section 3.7 we propose a possible implementation of a mobility interface for streaming applications.

The use of the MM offers, in our context, two key benefits. First, the handoff delay at network layer can be reduced from 1-2 s to 0.2-0.4 s. The packet loss for the "up" handoff is virtually eliminated by performing the handoff before the wireless connection is lost [27]. Second, the MM can pro-actively trigger the stream adaptation when a handoff occurs. The adaptation request can be sent *before* the handoff execution in nearly all cases, since signal strength monitoring allows to forecast handoff situations. The only exception are up handoffs from cabled interfaces where the handoff cannot be predicted.

3.7 Cross-layer integration mechanism

The integration of the MM in an adaptive streaming architecture is shown in Figure 2. On the server side, the streaming application is enhanced by a flow controller, that issues commands to the transcoder. This component implements the fine-grained adaptation mechanism described in Section 3.5 by performing a downgrade of the stream bit rate before sending the flow by means of the RTP protocol. On the client side, the MM described in Section 3.6 has two tasks: *i)* managing network connectivity and mobility policies through Mobile IPv6; *ii)* sending messages to the RTSP client mobility interface.

The RTSP client mobility interface is the key component of the adaptive streaming architecture. It receives information about the available video resolutions through the SDP description at the beginning of the session. It also acquires information about the available bit rate, handoff decisions, *etc.* from the MM. By using this information it canselect which stream to ask when a stream switching becomes necessary and issue appropriate RTSP commands to synchronize subsession switching with network handoffs. The appropriate timing of these commands is crucial, as shown in Section 4, in order to guarantee video quality reproduction. The video client application receives and displays the stream. Since, in general, the RTSP client is part of a video client application, it is necessary to develop only the support for the RTSP client mobility interface. An existing (unmodified) video client application can still display adaptive video streams if the RTSP client is a distinct application.

4. IMPLEMENTATION AND EXPERIMENTAL RESULTS

We implemented a prototype of the adaptive streaming architecture described in the previous section in order to assess its validity in a real-world setting. The experiments have been conducted on a private IEEE 801.11b WLAN and a 3G UMTS cellular network managed by a TelCo operator. Currently our prototype only supports client-side

Table 1: Network parameters used in the experimental test-bed

	802.11b	UMTS	GPRS
nominal bit rate	11 Mbps	384 kbps	54 kbps
delay	5 ms	150 ms	300 ms
concurrent traffic	no	yes	yes
handoff threshold	warning=-70 dB alarm=-85 dB		
handoff timers	alarm=0.2 s dwell=0.5 s		

coarse-grain adaptation based on RTSP. We employed the open-source LIVE.COM libraries[2] for multimedia streaming to build an RTSP client with a mobility interface. The same library is largely used to add streaming support to several open-source media player applications. The enhancements required to the original library include: *i)* IPv6 support; *ii)* the extension of the RTSP PLAY method to allow random access (as stated in [23]). The development of an RTSP client supporting the mobility interface was quite straightforward. We were able to use unmodified existing applications to receive and display the adapted unicast stream via RTP.

Mobility support was based on the MIPL 1.0 implementation of Mobile IPv6 for a Linux 2.4.24 kernel. The MM module monitored Ethernet, 802.11b and a data card providing UMTS connectivity. The characteristics of the networks used in our test-bed are reported in Table 1.

In the first set of trials 300-500 kbps video streams were received by a MN while moving through WLANs and UMTS networks with no adaptation. The MM module is used to speed up the handoff execution. A sample trial plot for up handoff at 300 kbps is shown in Fig. 3(i) (the vertical dashed lines in the figures indicate the moment of the handoff). Note that the handoff is completed before t=20 s, and the delay is approximately 200 ms, but there are additional delays due to the set-up of a dedicated UMTS radio link for the high traffic rate. At this bit rate, the UMTS link takes approximately 5 s to reach a steady throughput.

The conclusions that can be drawn from this set of experiments are that:

- with the help of the MM module it is possible to perform lossless vertical handoffs. The MN undergoes only a short delay of 0.2-0.4 s when it migrates from the faster 802.11b to the slower UMTS network.
- the UMTS network produces significant delays and packet losses at bit rates exceeding 300 kbps. In the sample plot, there is a pause in the receiving process (at t $\simeq$ 20.5 s), a sequence of lost packets (at t$\simeq$ 23.3 s) and a sequence of out-of-order packets (at t$\simeq$ 22.5 s).
- the down handoff causes a sequence of out-of-order packets that lasts for $\simeq$ 2 s (see Fig 3(ii)). This is a consequence of the different packets' delays and buffering policies on the UMTS network that degrades the quality of the stream reproduction more than the (short) handoff delay.

[2][Online] Available: `www.live.com`

A flexible buffering mechanism should allow to reduce the impact of out-of-order packets at application level. A reduction of the bit rate may reduce the number of out-of-order packets, since when the MN comes to the area covered by the WLAN, there are fewer packets still traveling along the UMTS link.

As expected, at a bit-rate of 500 kbps the stream suffers growing delays and packet losses, as shown in Fig. 3 (iii)-(iv). In these plots, the up handoff is at t$\approx$13 s, and the down handoff at t$\approx$46 s (we omitted the sequence of out-of-order packets).

The strategy implemented in the RTSP mobility interface is to let the network handoff happen on the stream with the lower bit rate. This means thatthe session handoff must be executed *before* the up handoff and the session handoff must be executed *after* or *during* the down handoff. This approach offers three advantages: *i)* packet loss is lower; *ii)* in a down handoff there are fewer packets traveling on the slower link that arrive out of order; *iii)* in a up handoff the slower link does not become congested in the time frame between the network and the session handoff.

The critical issues for the adaptation of the stream are: (i) the choice of the new stream among the available ones, (ii) the timing of the session handoff with respect to the network handoff and (iii) the timing of the PAUSE and PLAY requests in order to start receiving the new stream from the same point where the old one ends and to minimize the number of out-of-order packets.

4.1 Stream Choice

The choice of the stream is made by the RTSP mobile client by comparing the description of the available streams contained in the SDP descriptor. The choice must consider the header overhead, as it may be meaningful due to additional tunnel headers. Each packet contains at least the IPv6, UDP and RTP headers (60 bytes). In the worst case, if we are using a VPN plus a IPv6-in-IPv4 tunnel (to carry IPv6 packets in the current cellular networks that are not IPv6 enabled), and a IPv6-in-IPv6 tunnel from the HA to the MN, the header overhead grows to 140 bytes (2 IPv6 header, 2 IPv4 headers, UDP and RTP).

If H is the header size, f the frame size, and b the target stream rate (bytes/s), the packet rate is $\lambda = \frac{b}{f-H}$ packets/s and the total bitrate B of the stream will be $B = \lambda f = \frac{fb}{f-H}$. Since the stream coding is at variable bit rate, it is appropriate to consider an additional factor k to take into account the bit rate fluctuations. In practice, we assume $k = 2$. If $\{S_i\}$ is the ordered list of available bit rates, and B^* the bit rate of the access link after the handoff, the RTSP client chooses the greatest S_i such that:

$$B^* \geq \frac{kfS_i}{f-H}$$

The header overhead amounts to 5-12% of the total bit rate. The RTSP client can discover B^* as part of the information provided by the MM along with the handoff notification. Note that, due to fine-grain adaptation, the actual bit rate $B^{'}$ will be limited to the range $B^* \geq B^{'} \geq B^*/4$.

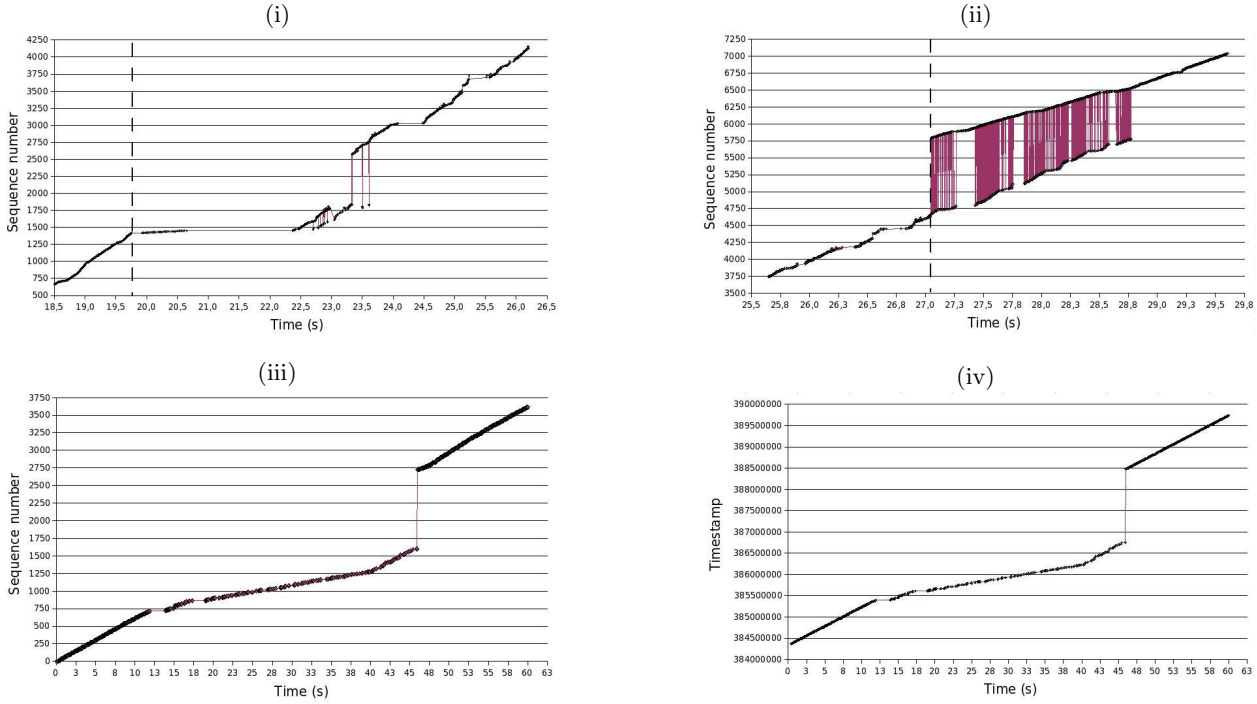

Figure 3: (i) up handoff at 300 kbps; (ii) down handoff at 300 kbps; (iii) time/sequence, and (iv) time/timestamp plots at 500 kbps with two vertical handoffs.

This is not taken into account in the stream choice phase by the MN, because the fine grain adaptation is performed on the server side and the MN has no control on it. The server could not provide this functionality or it might have been disabled.

4.2 Session/Network handoff Timing

In case of an up handoff, the RTSP client must issue the RTSP request to perform the session handoff *before* the network handoff execution. This is possible by means of the *delayed_handoff (t)* function, part of the MM API, that asks the MM for a notification t_{up} ms before the handoff execution. This function entails a t_{up} delay in the beginning of the network handoff.

The minimal value for t_{up} is determined by the duration of the session handoff. The session handoff requires that: *i)* the RTSP PAUSE command arrives to the server; *ii)* the client waits for the RTSP OK; *iii)* the RTSP PLAY arrives to the server. Neglecting the time to elaborate the commands, this yields an estimate for the lower bound: $t_{up} \geq \frac{3}{2}RTT$. Note that the RTT to be used is the one on the faster link. This ensures that t_{up} is not too high to perform the session handoff before the network handoff. In practice, for a WLAN we can expect a t_{up} value in the order of 50-300 ms, that is a reasonable delay for the network handoff execution.

In case of a down handoff there is no such strict timing requirement. The following two approaches are feasible:

1. *Postpone the session handoff.* The RTSP client waits t_{down} s after the down handoff before switching again to a higher quality video session. This delay is required to allow handoff execution and avoid potential ping-pong effects at session level. In practice, a value for t_{down} of 2-3 s is sufficient.

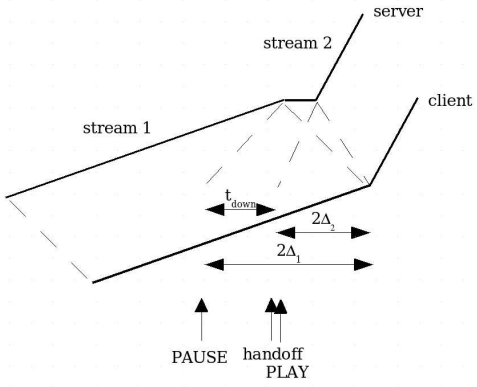

Figure 4: Timing of RTSP commands in a down handoff.

2. *Interleave session and network handoff.* The RTSP client issues a PAUSE t_{down} ms before the down handoff and a PLAY immediately after. In this case, if Δ_1 and Δ_2 are the trip times from the MN to the server along, respectively, the old and new access links ($\Delta_1 > \Delta_2$), packets arrive in the correct order provided that $t_{down} = 2(\Delta_1 - \Delta_2)$ (see Fig. 4).

Our current implementation uses fixed values for bit rate and delay for each network technology. We are planning to develop a probing module running on the MN and its HA to improve the precision of the estimate performed by the MM. A possible, attractive alternative could be to rely on QoS SLAs statically provided by the cellular network operator.

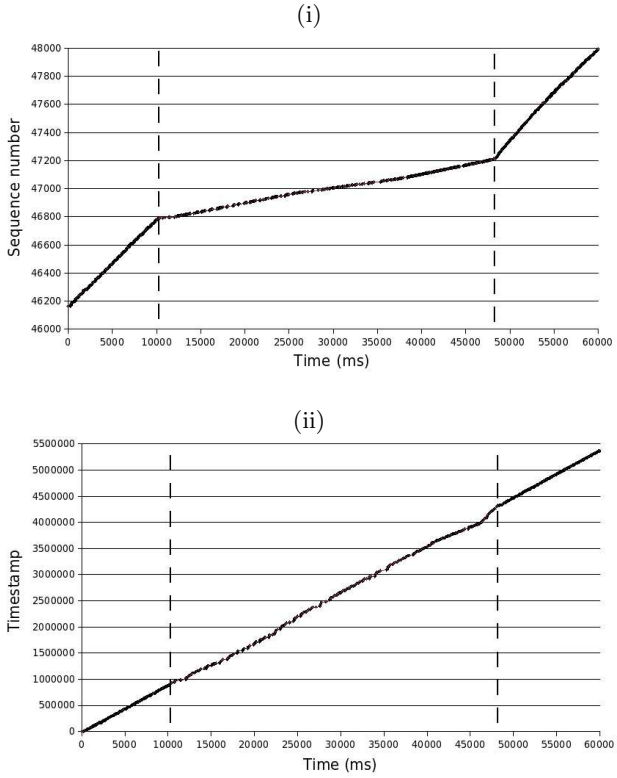

Figure 6: time/sequence (i), time/timestamp (ii) during "up" and "down" handoffs with adaptation (500-100 kbps).

The support of QoS is presently still experimental, but in the near future, and particularly with the introduction of the 3G IP Multimedia Subsystem, it will be a standard feature of UMTS networks.

Figg. 5-6 show a sample trial, with stream adaptation from 500 to 100 kbps, in the "up" and "down" handoff cases. Note from the time/timestamp plot of Fig. 6 how the packet rate slows down after the session switching, but the speed of the stream reproduction remains the same. Table 2 summarizes performance parameters in a set of 5 trials for each configuration. In these trials we used the "postponed session handoff" approach for down handoffs. These results suggest that:

- packet loss in the up handoff is negligible. Thanks to the MM module, the break in the reception of packets lasts less than 0.5 s, and it depends mostly on the higher RTT of the UMTS link.
- adaptation reduces the number of out-of-order packets in the down handoff. Using the "interleaved handoff" approach we expect to further reduce this effect.
- the down handoff delay is not relevant, since this is a soft handoff with no packet loss.
- even a limited buffering of packets on the client side is sufficient for smooth reproduction during the handoffs.

Table 2: Overall performance of adaptation with handoffs

	UMTS 500-100 kbps	UMTS 500-50 kbps	GPRS 500-50 kbps
up delay	460 ms	470 ms	1790 ms
t_{up}	150 ms	150 ms	150 ms
up packet loss	3	0	7
t_{down}	500 ms	300 ms	300 ms
out of order packets	15	3	61
down delay	150 ms	150 ms	200 ms

5. COMPARISON TO RELATED APPROACHES

The use of multiple streams in our approach is motivated by the large differences in bit rate across heterogeneous networks: when the range varies from 50 kb/s (GPRS networks) to 11Mb/s or more (on 802.11 WLANs) neither layered adaptation nor frame-dropping or limited transcoding provide a sufficient range of adaptation [18]. Moreover, adaptive protocols are too slow in their response to the sudden bit rate variation induced by a network handoff.

With respect to layered multi-path approaches (like [17]), our proposal offers two advantages, namely *i)* it is easier to deploy, since it does not require modifications on the server side; *ii)* it does not rely on simultaneous connections, that are not necessarily common for mobile devices and are more difficult to synchronize. A cross-layer mobility/adaptation schema can be also employed for applications different from multimedia streaming (like for instance interactive or VoIP applications) that can benefit from network status notification.

An important issue that we have not considered in our work is the integration of the adaptive mechanism with users' policies on network selection [28]. For example, access costs on different networks may vary considerably and should influence not only the handoff decision but also the adaptation process. For example, the user might prefer a lower resolution version of a clip on an expensive network, or suspend the download when the cost exceeds a fixed threshold.

6. CONCLUSION

We have shown that the problem of adapting video streams on heterogeneous networks can be solved with an architecture that integrates session and network mobility management. The advantages of our approach are its end-to-end nature, the limited computational cost and the combination of existing protocols at different levels. We implemented a testbed to show that this solution works as expected in a realistic scenario. We are extending our work to introduce fine-grained adaptation and performing experiments in different network conditions and with more applications. In particular, we are going to investigate the interaction between the algorithms for session handoff and the size of the application buffer. Finally, we plan to introduce in the MM a probing module for the dynamic detection of network parameters.

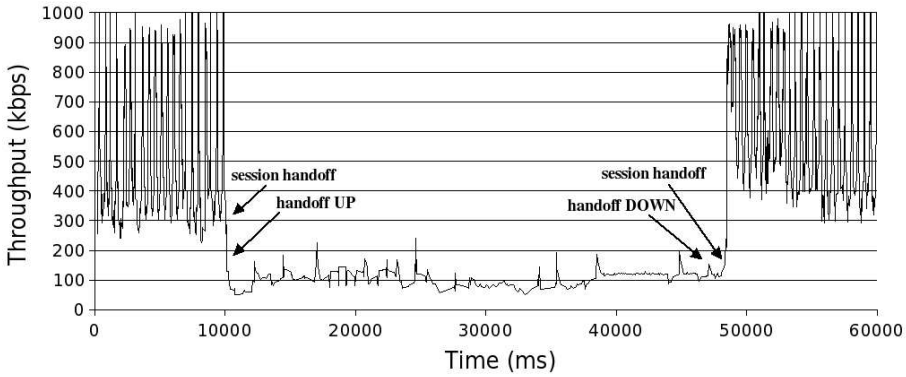

Figure 5: RTP throughput during "up" and "down" handoffs with adaptation (500-100 kbps).

7. ACKNOWLEDGMENTS

This work was funded by the Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) with the FIRB project WEB-MINDS.

8. REFERENCES

[1] C. Perkins, IPv4 Mobility support. *RFC 2002*, IETF, October 1996.

[2] D. Johnson, C. Perkins and J. Arkko, Mobility Support in IPv6. *RFC 3775*, IETF, June 2004.

[3] L. Chen, G. Yang, T. Sun, M. Y. Sanadidi, and M. Gerla, Adaptive Video Streaming in Vertical Handoff: A Case Study. In *Proceedings of IEEE MobiQuitous 04*, August 2004, pp 111-112.

[4] R. Rejaie, M. Handley, and D. Estrin, RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. In *Proceedings of IEEE Infocom 99*, March 1999.

[5] E. Amir, S. McCanne, and H. Zhang. An Application Level Video Gateway. In *Proceedings of ACM Multimedia 95*, San Francisco, November 1995.

[6] J. Inouye, S. Cen, C. Pu, and J. Walpole. System Support for Mobile Multimedia Applications. In *Proceedings of IEEE NOSSDAV97*, St. Louis, May 1997, pp. 143-154.

[7] A. Balk, D. Maggiorini, M. Gerla, and Y. Sanadidi. Adaptive MPEG-4 Video Streaming with Bandwidth Estimation. In *Computer Networks Journal*, vol. 44, issue 4, March 2004, pp. 415-439.

[8] N. Feamster, D. Bansal, and H. Balakrishnan. On the Interactions Between Layered Quality Adaptation and Congestion Control for Streaming Video. In *Proceedings 11th Int. Packet Video Workshop*, Kyongju, May 2001.

[9] S. Y. Cheung, M. Ammar, and X. Li. On the use of destination set grouping to improve fainess in multicast video distribution. In *Proceedings of IEEE Infocom 96*, San Francisco, March 1996.

[10] C. Krasic, J. Walpole, and W. Feng. Quality-Adaptive Media Streaming by Priority Drop. In *Proceedings of NOSSDAV 03*, Monterey, June 2003, pp. 112-121.

[11] J. Byers, M. Luby, and M. Mitzenmacher. Fine-grained layered multicast. In *Proceedings of IEEE Infocom 2001*, Anchorage, April 2001.

[12] X. Li, S. Paul, and M. Ammar. Layered Video Multicast with retransmissions (LVMR): Evaluation of hierarchical rate control. In *Proceedings of IEEE Infocom 98*, San Francisco, March 1998.

[13] S. McCanne, V. Jacobson, and M. Vetterli. Receiver driven layered multicast. In *Proceedings of ACM SIGCOMM 96*, Stanford, August 1996.

[14] T. Kim and M. Ammar. A Comparison of Layering and Stream Replication video Multicast Schemes. In *Proceedings of NOSSDAV 01*, Port Jefferson, June 2001, pp. 63-72.

[15] L. Vollero, G. Iannello, and F. Delfino. An Open Software Architecture for Structured Data Elaboration and Transcoding. In *Proceedings of ITCC 03*, Las Vegas, Aprile 2003, pp. 681-687.

[16] R. Rejaie, M. Handley, and D. Estrin. Architectural Considerations for Playback of Quality Adaptive Video over the Internet. In *Proceedings of IEEE ICON 00*, Singapore, September 2000, pp. 204-209.

[17] Y. Pan, M. Lee, J. B. Kim, and T. Suda. An End-to-End Multi-Path Smooth Handoff Scheme for Stream Media. In *Proceedings of ACM WMASH 03*, San Diego, September 2003, pp. 64-74.

[18] S. Chang and A. Vetro. Video Adaptation: Concepts, Technology, and Open Issues. In *Proceedings of the IEEE*, vol. 93, no. 1, January 2005, pp. 148-157.

[19] Z. Lei and N. Georganas. Adaptive video transcoding and streaming over wireless channels. In *The Journal of Sytems and Software*,vol. 75, 2005, pp. 253-270.

[20] Q. Zhang, W. Zhu, and Y. Zhang. End-to-End QoS for Video Delivery Over Wireless Internet. In *Proceedings of the IEEE*, vol. 93, no. 1, January 2005, pp. 123-134.

[21] X. Yu, D. Hoang, and D. Feng. A QoS COntrol Protocol for Rate-adaptive Video Traffic. In *Proceedings of IEEE ICON 01*, Bangkok, October 2001, pp. 434-438.

[22] C. Bouras, and A. Gkamas. Streaming Multimedia Data with Adaptive QoS Characteristics. In *Proceedings of Protocols for Multimedia Systems*, Cracow, October 2000, pp. 129-139.

[23] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). *RFC 2326*, IETF, April 1998.

[24] M. Handley, V. Jacobson. SDP: Session Description Protocol. *RFC 232*7, IETF, April 1998.

[25] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RTP: A Transport Protocl for Real-Time Applications. *RFC 3550*, IETF, July 2003.

[26] M. Bernaschi, F. Cacace, A. Pescapè, and S. Za. Analysis and Experimentation over Heterogeneous Wireless Networks. In *Proceedings of Tridentcom 2005*, Trento, February 2005, pp. 182-191.

[27] M. Bernaschi, F. Cacace, G. Iannello, S. Za, and A. Pescapè. Seamless Internetworking of WLANs and Cellular Networks: Architecture and Performance Issues in a Mobile IPv6 Scenario. In *IEEE Wireless Communications Magazine*, Vol. 12, No. 3, June 2005, pp. 73-80.

[28] Q. Song, and A. Jamalipour. Network Selection in an Integrated Wireless LAN ans UMTS Environment Using Mathematical Modeling and Computing Techniques. In *IEEE Wireless Communications Magazine*, Vol. 12, No. 3, June 2005, pp. 42-49.

[29] N. D. Tripathi, J. H. Reed, and H. F. Vanlandinghum. Adaptive Handoff Algorithm for Cellular Overlay Systems Using Fuzzy Logic. In *IEEE 49th VTC*, vol. 2, May 1999, pp. 1413-18.

[30] HUT Laboratory for Theoretical Computer Science - GO/Core project. MIPL. Mobile IP for Linux (MIPL). http://www.mobile-ipv6.org.