An Energy-efficient Protocol for Multimedia Streaming in a Mobile Environment

G. Anastasi
University of Pisa - Dept. of Information Engineering
Via Diotisalvi, 2 - 56126 Pisa, Italy
g.anastasi@iet.unipi.it

M. Conti, E. Gregori, and L. Pelusi
IIT Institute - CNR Research Area
Via G. Moruzzi, 1 - 56124 Pisa, Italy
{*marco.conti, enrico.gregori, luciana.pelusi*}*@iit.cnr.it*

A. Passarella
University of Cambridge - Computer Laboratory
15 JJ Thomson Avenue - Cambridge, CB3 0FD, UK
andrea.passarella@cl.cam.ac.uk

Received: January XX 2005; revised: November XX 2005

***Abstract*— Pervasive services and smart environments are becoming more and more popular as an ever-increasing number of people enjoys these services typically by means of portable devices. These devices are battery-fed and, thus, energy efficiency is a critical factor for the deployment of pervasive services.**

In this paper we focus on multimedia streaming services for mobile users. Specifically, we consider a scenario where mobile users with Wi-Fi devices access the Internet to receive audio files from a remote streaming server. We propose a proxy-based architecture and an energy-efficient streaming protocol that minimize the energy consumption of the Wi-Fi interface at the mobile device, while guaranteeing the real-time constraints of the audio streaming. The experimental analysis performed on a prototype implementation shows that our solution allows an energy saving ranging from 76% to 91% of the total consumption due to the network interface. Moreover, it also preserves a good user-level Quality of Service.

***Index Terms*— Energy saving, audio streaming, Wi-Fi, mobile computing, proxy.**

I. INTRODUCTION

The increasing diffusion of portable devices and the growing interest towards pervasive and mobile computing is encouraging the extension of popular Internet services to mobile users. However, due to the differences between mobile devices and desktop computers, a big effort is still needed to adapt the classical Internet applications and services to the mobile wireless world. Energy issues are perhaps the most challenging problems since portable devices typically have a limited energy budget [1].

Among the set of potential mobile Internet services, multimedia *streaming* is expected to become very popular in the near future. Streaming is a distributed application that provides on-demand transmission of audio/video content from a server to a client over the Internet while allowing playback of the arriving stream chunks at the client. The playback process starts a few seconds (*initial delay*) after the client has received the first chunk of the requested stream, and proceeds in parallel with the arrival of subsequent chunks. Before playback, the arriving chunks are temporarily stored at a *client buffer*. The streaming service must guarantee a good playback quality. This imposes stringent *time constraints* to the transmission process because a timely delivery of packets is needed to ensure their availability at the client for playback. Moreover, since constant-quality encoded audio/video content produces *Variable Bit Rate (VBR) streams*, matching the time constraints is a real challenge in the best-effort Internet environment where the network resources are highly variable. Hence, one of the main goals for a streaming server is to use a *transmission schedule* that tunes the outgoing traffic to the desired network and client resource usage, while also meeting the playback time constraints. A good scheduling algorithm should allow an efficient use of the available network resources without overflowing or under-flowing the client buffer. Moreover, it should limit the initial delay before playback, because users do not generally tolerate high initial delays (greater than 5–10 s), especially when waiting for short sequences. Many smoothing algorithms have been proposed in the literature [2]. However, none of them can be considered the best one because the performance metrics to be considered are often conflicting and cannot be optimised all together. Moreover, the same algorithm generally performs differently when applied to different media streams with different characteristics (e.g., *burstiness*). As a result, the most suitable smoothing algorithm may change every time depending on (i) the content of the media stream to be sent, (ii) the specific performance goals of the system components, (iii) the particular resource-usage policy adopted at the server, the client and at the network routers, respectively.

In this paper we focus on streaming services in mobile wireless scenarios where servers are supposed to be fixed hosts in the Internet, whereas clients are supposed to run on mobile devices (e.g., PDAs, laptops or mobile phones with wireless network interfaces). Client devices access the Internet through an intermediate access point (or base station). We will assume the presence of a *proxy* between the client and the server since proxies are commonly used to boost the performance of wireless networks. In this environment we will concentrate on energy efficiency and will propose a proxy-based architecture based on the *Real Time and Power Saving (RT_PS)* protocol. This protocol implements an energy management policy whose main goal is to minimize the energy consumed by the *Wireless Network Interface Card (WNIC)* at the mobile host, while guaranteeing the real-time constraints of the streaming application. In this paper we will refer to the Wi-Fi technology [3]. However, our solution is flexible enough to be used with any type of infrastructure-based wireless LAN. Our approach allows the WNIC to save a significant amount (up to 91%) of the energy typically consumed without energy management. Another challenging issue, in this area, is to minimize the client buffer size, as memory costs for some kinds of mobile devices are still high. We will show that our solution achieves good energy savings even with small buffer sizes (e.g., 50 KB).

The remainder of the paper is organized as follows. Section II describes some related work on energy management with emphasis on those pieces that refer to real-time streaming applications. Section III introduces the proxy-based architecture that we propose, while Section IV describes the *RT_PS* protocol. Section V and Section VI are devoted to the description of some specific aspects of the *RT_PS* protocol. Section VII presents an experimental analysis of our proposal carried out on a prototype implementation, and finally, Section VIII concludes the paper.

II. Related Work

Energy issues in wireless networks have been addressed in many papers. [4] [5] [6] found that one of the most energy-consuming components in mobile devices is the WNIC since networking activities cause up to 50% of the total energy consumption (up to 10% in laptops, up to 50% in hand-held devices).

A first approach to reduce the energy consumption of the WNIC is to improve the throughput at the transport layer. The work in [7] proposed the *Indirect-TCP* architecture as an alternative to the classical TCP/IP architecture for communications between mobile and fixed hosts. The Indirect-TCP *splits* the TCP connection between the mobile host and the fixed host in two parts, one between the mobile host and the Access Point, the other between the Access Point and the fixed host. At the Access Point a running daemon is in charge of relaying data between the two connections. The Indirect-TCP avoids the typical throughput degradation caused by the combined action of packet loss in the wireless link and the TCP congestion control mechanism. As a result, the total transfer delay successfully reduces.

Nevertheless, the reduction in the total transfer delay can only produce a slight energy saving if compared to the energy wastage caused by the inactivity periods of the WNIC during data transfers. In fact, the natural *burstiness* of common Internet-traffic patterns determines long idle periods for the WNIC, when it neither receives nor transmits, but drains a large amount of energy the same, as is shown in [4] [6] [8]. A more effective solution to significantly reduce the energy consumption is to switch the WNIC *off* (or putting it in the *sleep mode*) when there is no data to send/receive.

The work in [6] retains the standard TCP/IP architecture, and either uses the 802.11 Power Saving Mode (PSM), or keeps the WNIC continuously active, based on predictions about the application behavior in the near future. [4] [5] [9] [10] use the Indirect-TCP architecture and also switch off the WNIC during the idle periods. [9] proposes an application-dependent approach tailored to web applications and exploits the knowledge of statistical models for web traffic to *predict the arrival times* of the traffic bursts. [10] proposes an application-independent approach, which is more suitable when several applications are running concurrently on the same portable device. Works in [9] [10] also highlight how traffic burstiness can be *exploited* to save energy. In fact, since switching the WNIC between different operating modes has a cost, having *few long* idle times rather than *several short* idle times is preferable from an energy saving standpoint.

The solutions so far mentioned are not suitable for streaming applications because of the time constraints imposed by such applications to the transmission process. Existing energy-management solutions for real-time applications are based on both the two following general approaches: (i) switching the WNIC to the sleep mode during inactivity periods, and (ii) reducing the total transfer delay of the streaming session. Switching the WNIC completely off is not typically considered because too time-consuming and thus potentially dangerous for the fulfilment of the real-time constraints.

Transcoding techniques have been introduced to shorten the size of audio/video streams so as to reduce the transfer delay [11] [12]. However, while leading to little energy savings (see above), these techniques may determine significant reduction in the *stream quality*.

The IEEE 802.11 PSM is not suitable for streaming applications because, as is stated in [13], it only performs well when the arriving traffic is regular, whereas it is not suitable for popular multimedia streams. All the solutions described below therefore assume that PSM in not active.

Some application layer techniques have been introduced that rely on predictions on the arriving traffic behavior. They switch the WNIC to the active mode when a packet is expected to arrive and to the sleep mode when an idle period is expected to start. [14] [15] propose solutions with client-side predictions of the packet inter-arrival times based on the past history. Wrong predictions result in packet loss and quality degradation. Techniques based on client predictions can lead to energy savings ranging between 50% and 80% [14] however, similarly to PSM, the best performance is achieved when the incoming traffic is regular. Predictions can thus benefit from some traffic shaping to become more effective. However, traffic shaping at the server is useless because transmission over the Internet changes the traffic profile by introducing *jitter* in the delay experienced by the packets. Therefore, traffic shaping must be performed at a proxy close to the mobile host. The solutions proposed in [13] and [15] perform traffic shaping at the proxy and also make use of *client-side predictions* to switch the WNIC to the sleep mode during the idle periods. The work in [15] also proposes a second solution where predictions are *proxy-assisted*. The proxy sends to the client a sequence of consecutive packets in a single burst. Then, it informs the client that the data transmission will be suspended for a given time interval. The client can thus switch the WNIC to the sleep mode for the announced time interval. The total energy saving achieved in [15] ranges from 65% to 85% when using client-side predictions, and from 80% to 98% with proxy-assisted predictions. These results have been obtained by reducing the energy consumption of both the WNIC (which is put in the sleep mode during inactivity times), and the CPU (whose usage is decreased thanks to transcoding).

Our solution relies on a proxy-based architecture where the proxy uses an *ON/OFF schedule* for data transmissions to the clients, i.e., it alternates transmission and non-transmission periods. In addition, the proxy warns the client as soon as a non-transmission period is starting so as the client can accordingly switch the WNIC to the sleep mode. In our proposal, unlike the solution in [15], the streaming policies are decoupled on the wireless and in the wired networks. In addition, the transmission schedule is dynamically adjusted by the proxy based on the current available bandwidth as well as on the client buffer level.

We focus on *Stored Audio Streaming* applications, and mainly refer to MP3 audio files. Moreover, we have decided not to use transcoding techniques to preserve the audio quality since the MP3 compression already reduces the quality of the audio stream. We believe that only video streams can take advantage of transcoding techniques because they allow a significant reduction in the file size without severely affecting the playback quality [15].

III. Proxy-based Architecture

The proxy-based architecture that we refer to in our solution is depicted in Fig. 1. The proxy functionality is supposed to be implemented at the Access Point. Therefore, the data path between the server and the mobile client is split at the border between the wireless and the wired networks (like in the Indirect-TCP model). The target in each part of the data path is different. In the proxy-client communications the major goal is to reduce the energy consumption at the mobile host. On the other hand, for the proxy-server communications a classical, smothness-oriented approach to the streaming is required. Smoothing is a good approach for the wired network since, by reducing the peak transmission rate, it improves the network-resource usage. However, it also increases the total transfer time and reduces the traffic burstiness, thus letting less room for energy management policies.

The streaming server is located at the fixed host and sends streams to the proxy (Access Point). Its scheduling policy is smoothness-driven. Real-time data is encapsulated in *RTP* (*Real Time Protocol*) [16] [17] packets while the *Real Time Streaming Protocol (RTSP)* [18] is used to exchange control information for playback with the other streaming peer.

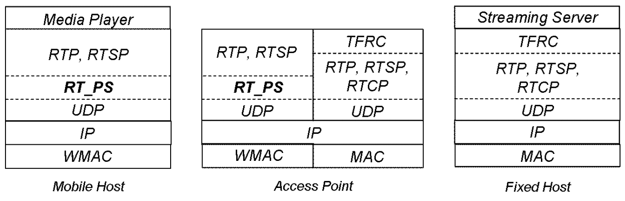

Fig. 1. Streaming architecture for a wireless scenario.

Since RTP uses UDP as underlying transport protocol, and UDP provides no built-in congestion control mechanism, the presence of multimedia traffic in the Internet can potentially lead to congestion. The *TCP-Friendly Rate Control* (*TFRC*) [19] protocol adaptively establishes an upper bound to the server transmission rate, thus leading to congestion avoidance. RTP and *RTCP* (*Real Time Control Protocol*) [20] packets are used in order to carry TFRC information [21].

The *Real Time and Power Saving* (*RT_PS*) protocol, which is the core of this architecture, is located on top of the UDP transport protocol and implements the energy management policy. Since the focus of this paper is on energy efficiency, hereafter we will only focus on the proxy-client communications over the wireless link. The *RT_PS* protocol works as follows. The proxy schedules packet transmissions by following an *ON/OFF scheme*. During the *ON periods* the proxy transmits packets to the mobile host at the highest possible rate allowed by the wireless link. During the *OFF periods* the mobile host switches the WNIC to the sleep mode. The traffic shaping performed by the proxy relies on the *knowledge of the audio/video frame lengths*, the client *buffer size*, and the current *available bandwidth* on the wireless link. During an ON period the proxy decides whether to stop transmitting or not depending on the available bandwidth. When the available bandwidth is high, the proxy continues transmitting until the client buffer fills up. When the available bandwidth is low, it stops transmitting to avoid increasing congestion and transfer delay. The duration of an OFF period is decided by the proxy when it stops transmitting, and depends on the client buffer level. An OFF period ends when the client buffer level falls down a dynamic threshold, named *low water* level, which warns the proxy about the risk of playback starvation. Finally, to avoid bandwidth wastage, only the frames that are expected to arrive in time for their playback are delivered to the client whereas the others are *discarded*. The computation of the frame arrival times relies on the estimates of the available throughput on the wireless link.

In addition to the client buffer, the energy management policy implemented by the *RT_PS* protocol also relies on the *initial playback delay*. When the streaming session is established the playback process does not start immediately, instead, a given amount of data is accumulated in the client buffer before the playback process can start. However, it is worth noticing that the client buffer and the initial playback delay are not special requirements of our protocol. Indeed, they are needed in any streaming system to absorb the delay jitter introduced by the network.

IV. *RT_PS* Protocol Description

The *RT_PS* protocol includes a proxy component running at the proxy and a client component running at the mobile device. The proxy-side component carries out the followings tasks: i) evaluates the playback *start point*, i.e., the initial playback delay, ii) transmits the (audio) frames to the client during the ON periods, iii) keeps track of the level of the client buffer by considering the progressions of both the transmission and the playback processes, iv) decides whether or not to stop transmitting given the current available throughput, and v) calculates the duration of the sleep periods for the WNIC of the mobile device in such way to prevent the playback process from starving due to buffer underflow.

The client-side component performs the following tasks: i) sends the initial request for an (audio) file to the proxy also specifying the size of the client buffer, ii) triggers the playback process when the start point elapses, iii) collects the arriving (audio) frames in a buffer where the media player can take and play them out, iv) estimates the throughput currently available on the wireless channel and notifies the *RT_PS* proxy-side component, v) switches the WNIC to the sleep mode upon receipt of a sleep command from the proxy, and vi) switches the WNIC back to the active mode as soon as the sleep time has elapsed.

A complete description of the *RT_PS* protocol is omitted for the lack of space (the reader can refer to [22]). However, some key tasks are detailed in Section V and Section VI.

V. Start Point and Sleep Time Computations

This section describes how the *RT_PS* protocol derives the start point and the durations of the ON and OFF periods. It makes use of the following analytical model to refer to the streaming process over time.

A. Analytical Model

Fig. 2 represents the playback process of an MP3 audio file by means of a graphical model [2] [23] (symbols are described in Table I). An MP3 audio file is a sequence of frames and its playback begins at the *start point* (SP in the figure), which is the time when the first frame of the file is played out. Each subsequent frame must be played out starting at fixed, equally spaced, times called *playback times*. In Fig. 2 frames are spaced by Δ seconds.

The *playback function* $V(t)$ defines the total number of bytes that *must have been* played out by time t in order to correctly reproduce the stream at the client. Specifically, $V(t)$ is defined as follows:

$$V(t) = \begin{cases} 0 & if \quad t < SP \\ \sum_{i=0}^{k} f_i & if \quad t \in [SP + k\Delta,\ SP + (k+1)\Delta) \end{cases} \qquad (1)$$

$k = 0, \ldots, N-1$; $i = 0, \ldots, k$. f_i is the size in bytes of the i^{th} frame in the stream, and k is the number of the Δ-*sized* inter-frame spaces occurred from the starting point SP up to time t. The playback times can thus be formally defined as $SP + k\Delta$, $k = 0, \ldots, N-1$, and are actually deadlines for frame arrivals at the client, and for frame deliveries at the proxy. During a streaming session, frames usually arrive at

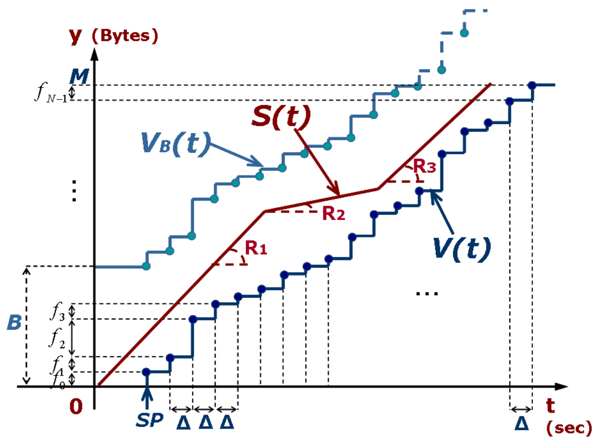

Fig. 2. Streaming model under real-time constraints: graphical model.

TABLE I
Symbols used in the Streaming Model.

Symbol	Meaning
Δ	The time interval between two consecutive frames.
N	The number of frames in the multimedia file.
$f_i,\ 0 \le i < N$	The size of the i^{th} frame in the multimedia file.
$M = \sum_{i=0}^{N-1} f_i$	The size of the multimedia file.
$B \ge \max_{0 \le i < N} f_i$	The size of the client buffer.
$S(t)$	The scheduling function.
R_i	The i^{th} transmission rate.
SP	The Start Point: when playback begins.
$V(t)$	The playback curve.
$V^B(t)$	The upper bound constraint to the scheduling function.

the client ahead of their designated playback times, and are thus temporarily stored in the client buffer, whose size will be hereafter referred to as B. As $V(t)$ represents a lower bound for the number of bytes arrived at the client by time t, it is possible to define an *upper bound* for this number of bytes, denoted as $V^B(t)$. $V^B(t)$ is the maximum number of bytes that the client can host by time t without causing a buffer overflow, and can thus be computed as $V^B(t) = B + V(t)$.

The *scheduling function* $S(t)$ defines the number of bytes sent by the proxy to the client by time t, while the corresponding arrival function $A(t)$ defines the number of bytes received by the client by time t. Clearly, the relation $S(t) \ge A(t)$ always holds, the difference between $S(t)$ and $A(t)$ depending on the network delay and jitter. However, it is generally assumed that $S(t) = A(t)$ and the difference between the two curves is heuristically taken into account by slightly enlarging the size of the client buffer so as it can *absorb* both the jitter and the transfer delay. Therefore, hereafter we will assume $S(t) = A(t)$. Based on this assumption, and recalling the definition of both $V(t)$ and $V^B(t)$, it follows that a scheduling function is any non-decreasing function confined between $V(t)$ and $V^B(t)$, i.e., the following equation holds:

$$V(t) \leq S(t) \leq V^B(t). \tag{2}$$

Finally, it is worth noting that the playback process extracts one single frame at a time from the client buffer at each playback time. Therefore, the level of the client buffer, at the time instant t, is the difference between the total number of bytes arrived at the client by time t, and the total number of bytes already played back, i.e., $S(t)-V(t)$.

The scheduling algorithms define different scheduling curves depending on the particular goal targeted. When traffic smoothing is the target, the scheduling curve is a polyline with line segments of different slopes corresponding to different transmission rates (R_i is the i^{th} transmission rate). In this paper we will use polylines that alternate line segments with zero slope (OFF periods) and line segments with non-zero slope (ON periods).

B. Start Point Computation

In this section we will show how the *RT_PS* protocol derives the start point, i.e., the initial playback delay. As a first step, it is worth focusing on constant-rate ON/OFF schedules, i.e. ON/OFF schedules where data transfers during the ON periods occur at a constant rate R. A deep analysis on constant-rate ON/OFF schedules applied to VBR traffic can be found in [23] and [24]. These pieces of work prove that, given the constrained region delimited by the functions $V(t)$, $V^B(t)$, $t=0$, $y=0$ and $y=M$ (see Fig. 2), it is possible to find an ON/OFF schedule only in case the transmission rate is greater than a minimum value, hereafter referred to as $\bar{R}$. For some combination of $V(t)$, $V^B(t)$, and M, $\bar{R}$ could be 0, meaning that the schedule is always feasible. For any $R \geq \bar{R}$, many ON/OFF schedules can be conceived, which differ in the start point value, and the number, position and duration of the OFF periods. However, a *minimum* ON/OFF schedule exists among these, which is defined as *R-envelope* and denoted as $E_R(t)$. It was formally defined in [23] and is, intuitively, the ON/OFF schedule closest to the playback curve $V(t)$. Therefore, for any possible ON/OFF schedule $S_R(t)$ the following equation holds:

$$S_R(t) \geq E_R(t); \tag{3}$$

$\forall t,\ 0 \leq t < SP+N\Delta,$, where N is the total number of frames composing the file, and Δ is the time the playback process takes to play a single frame. For the sake of semplicity, hereafter we will only refer to the sequence $E_R[k]$ [25] instead of the continuous function $E_R(t)$. $E_R[k]$ is composed by the samples picked up from $E_R(t)$ at regular time intervals. It can easily be obtained starting from the definition of its last sample $E_R[N-1]$ which corresponds to the end of the playout curve (M) at the last playback time. Formally, $E_R[N-1] = V(t^{last}) = M$, where $t^{last} = SP+(N-1)\Delta$. Drawing from there a descending *R-sloped* line segment that ends at the previous playback time (i.e., Δs before, when $t = t^{prev} = SP+(N-2)\Delta$), the new sample $E_R[N-2]$ corresponds to the end-point of the segment, in case it is higher than the value of the playout curve $V(t^{prev})$, or to the corresponding value of the playout curve $V(t^{prev})$ otherwise. Subsequent samples can be obtained, again, starting from the new sample $E_R[N-2]$ and drawing a new *R-sloped* line segment till the previous playback time, and so on. By repeating this procedure until the time $t=0$ is reached, the entire sequence $E_R[k]$ is obtained. More formally, $E_R[k]$ can be expressed as follows:

$$E_R[k] = E_R(SP+k\Delta) =$$
$$\begin{cases} V(SP+(N-1)\Delta) & if\ \ k=N-1 \\ max\{E_R[k+1]-R_\Delta,\ V(SP+k\Delta)\} & if\ \ 0\leq k<N-1 \\ 0 & if\ \ k<0 \end{cases} \tag{4}$$

where $0 \leq k < N$ and R_Δ is the total number of bytes that can be transmitted during the time interval Δ assuming a transmission rate R.

The minimum rate $\bar{R}$ that allows an ON/OFF schedule to be performed is exactly the minimum rate for which an $E_R(t)$ curve exists inside the region defined by $V(t)$, B, and M. Specifically, the following equation holds:

$$\bar{R} = min\{R \mid \exists E_R(t)\}. \tag{5}$$

At the beginning of the streaming session, the *RT_PS* proxy-side component calculates the minimum rate $\bar{R}$ for the audio sequence to be transmitted, and contrasts it with the bandwidth R^0 available on the wireless link. The streaming process can actually start only if $R^0 \geq \bar{R}$.

After this preliminary check, the *RT_PS* proxy component can decide the playback *start point*. Assuming that the proxy starts to transmit the frames at time 0, the start point corresponds to the initial delay. Obviously, a long initial delay grants more time for *pre-buffering* at the client and prevents the playback process from starving due to buffer underflow. However, several reasons suggest keeping the initial delay short. First, the user is not willing to wait for long time before starting to listen to the selected audio file. Moreover, as is shown in [23], an increase in the initial delay causes an increase in the total idle period (i.e., the sum of all the OFF periods) while the total ON period remains unmodified. In our solution this increases the energy consumption due to the energy amount consumed by the WNIC while in the sleep mode.

The start point is calculated as follows. Given the initial transmission rate R^0, the minimum possible initial delay d_{R^0} is the distance, measured on the *X-axis* (t), between R^0-*envelope* $E_{R^0}(t)$ and $V(t)$. To avoid underflowing the client buffer due to possible variations in the available throughput, the initial delay is calculated with reference to $R^0/2$-*envelope* instead of R^0-*envelope*, i.e., $d_{R^0/2}$ is used instead of d_{R^0}. The rationale behind this heuristic comes from the following property [23]. Given two transmission rates R_1 and R_2, such that $0 \leq R_1 \leq R_2$, the following relations hold:

$$E_{R_1}(t) \geq E_{R_2}(t),\ d_{R_1} \geq d_{R_2}; \tag{6}$$

$\forall\ t,\ 0 \leq t < SP + N\Delta$. Using d_{R^0} would lead to buffer underflow even for small reductions in the available throughput. On the other hand, using $d_{R^0/2}$ a buffer underflow does not occur unless the available throughput falls below $R^0/2$.

Finally, for very low values of R^0, the initial delay $d_{R^0/2}$ may be too long, beyond the maximum delay the user is willing to tolerate. In this case the initial delay is taken as the maximum initial delay specified by the user.

C. Sleep Time Computation

The next issue to be addressed is how to compute the duration of an OFF period during which the WNIC can be switched to the sleep mode. An OFF period starts when either the client buffer fills up, or the available throughput becomes too low (half the initial value).

The procedure described above to calculate the initial delay is used to compute the durations of the OFF periods too. Let us assume that the server has just scheduled and sent to the client $S(t^*)$ bytes. Let us also assume that, according to the last estimate provided by the client, the current transmission rate (available throughput) on the wireless link is R^*. Then, the maximum allowed sleep time corresponds to the distance, measured on the $y = S(t^*)$ horizontal axis, between $S(t^*)$ and R^*-*envelope* $E_{R^*}(t)$. Intuitively, assuming that the throughput R^* will also be available in the near future, the mobile host can sleep until the scheduling function intersects $E_{R^*}(t)$. As above, to reduce the risk of playback starvation, $R^*/2$-*envelope* $E_{R^*/2}(t)$ is used instead of R^*-*envelope* $E_{R^*}(t)$. Therefore, a buffer underflow will not occur unless the available throughput falls down $R^*/2$.

Obviously, an OFF period actually starts only if the calculated OFF duration is greater than the time needed to the WNIC to switch from active to sleeping and back from sleeping to active. When this condition does not hold the client buffer level is deemed to be a *low water* level because the scheduling function is very close to the playback curve and a transmission interruption is too dangerous. In this case, the transmission must continue even if the available throughput on the wireless channel is low. Otherwise, the client buffer level is considered safe and an OFF period can start.

VI. Available Throughput Estimate

In this section we will show how the *RT_PS* protocol estimates the throughput available on the wireless link. The transmission rate used by the *RT_PS* protocol to derive the ON/OFF schedule corresponds to the estimated available throughput.

It is yielded with a very simple and fast technique that guarantees good accuracy for our purposes nevertheless. In principle it works as follows. The *RT_PS* proxy sends a train of *back-to-back* packets to the *RT_PS* client. The packet train is preceded by a special packet informing the client that a new packet train is starting. Upon receipt of the special packet the client records the time. Moreover, it starts counting the number of bytes contained in the subsequent packet train. The available throughput experienced by the client is thus estimated as the ratio between the total number of bytes contained in the received packet train and the total time needed to receive those bytes (time interval between receipt of the special packet and receipt of the last packet in the train).

For an accurate estimate two key factors must be taken into account: the *packet train length* and the *size of each single packet*. In our experiments we found that the best accuracy is achieved when the packet size is 1472 bytes. This value corresponds to the maximum UDP-payload size that does not undergo the MAC layer fragmentation (when the packets transmitted during the streaming session are shorter than 1472 bytes, the overhead times introduced for each single transmission at the MAC layer -according to the 802.11b standard- have a greater impact on the total transmission time and the resulting throughput calculated). We also found that, to achieve an accurate estimate the packet train should include at least 50 1472-*byte* packets. Under these conditions, we found that the maximum throughput experienced by the client in a Wi-Fi WLAN without any other competing traffic is approximately 6.65 Mbps when the WNIC data rate is 11 Mbps, 4.04 Mbps when the data rate is 5.5 Mbps, and 1.69 Mbps when the data rate is 2 Mbps.

However, in our implementation, estimates work out from the *RT_PS* traffic during the ON periods. Hence, a packet is an *RT_PS* message that includes an RTP packet. The RTP packet is further composed by couples <*ADU descriptor, ADU frame*> that cannot be fragmented for the sake of loss tolerance [26]. This may lead to a large variability in the packet sizes and to degrading the accuracy in the estimates. To overcome this problem, the *RT_PS* proxy sends an integer value of couples <*ADU descriptor, ADU frame*> in a single RTP packet so as to keep the *RT_PS* packet sizes as close as possible to the ideal value (i.e., 1472 bytes). Moreover, the train length is increased from 50 to 60 packets. This is generally possible in our scenario except when the client buffer is small. In that case the accuracy in estimates slightly decreases. Fig. 3 shows the

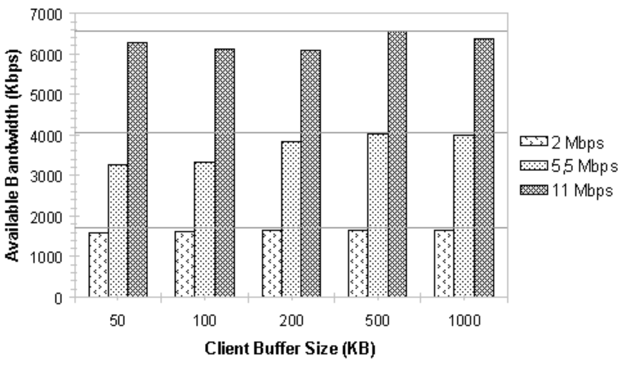

Fig. 3. Available bandwidth estimates for different client buffer size and WNIC data rate.

throughput estimated on a WLAN without competing traffic, for different data rates of the WNIC and different sizes of the client buffer. These results have been obtained by using a prototype implementation of the proposed architecture (see Section 7 for details about the experimental testbed and the measurement methodology). Specifically, the above algorithm was used for estimating the throughput experienced by the client. The results in Fig. 3 show that, in absence of interfering traffic, a streaming session is able to exploit up to 6.54 Mbps (instead of 6.65 Mbps) with 11 Mbps data rate, up to 4.01 Mbps (instead of 4.04 Mbps) with 5.5 Mbps data rate, and up to 1.66 Mbps (instead of 1.69 Mbps) with 2 Mbps data rate. These results show that our estimate methodology, based on real *RT_PS* traffic is accurate.

The client buffer size is another factor that affects the accuracy in the throughput estimates. When the client buffer is large, the *RT_PS* schedule produces a small number of very long OFF periods. This results in a very bursty traffic where transmissions are concentrated in short periods separated by long interruptions. As the client buffer size decreases the number of the OFF periods increases, whereas their lengths decrease. Therefore, more control packets are needed [22], which are shorter than the packets carrying audio data. Clearly, this results in the available throughput be slightly underestimated, as highlighted in Fig. 3. Based on the above experimental results, we can conclude that, for sufficiently large buffers, our algorithm for bandwidth estimate is accurate. Please note that, since each estimate is derived from a long train of packets, short-term variations in the available throughput are filtered out.

We also investigated the effect of some competing traffic on the accuracy of the estimate algorithm. When adding background traffic to the streaming flow the estimates of the available bandwidth decrease accordingly. Fig. 4 focuses on a

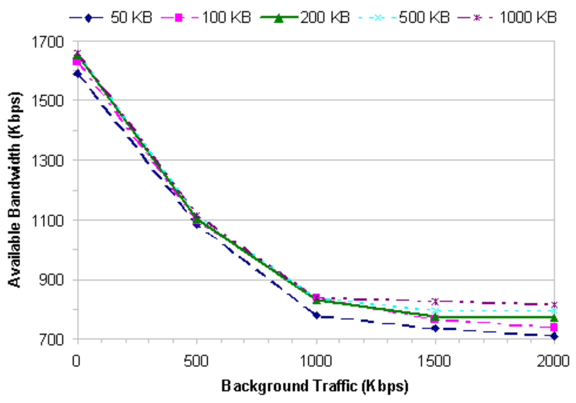

Fig. 4. Available bandwidth estimates for different client buffer size and background traffic.

WNIC data rate of 2 Mbps and shows the estimated throughput for increasing rates of the background traffic and different client buffer sizes. As expected, the available throughput decreases as the background traffic grows up. When the background traffic is so high to saturate the wireless bandwidth both the background and the streaming traffic are expected to achieve half the maximum available bandwidth, i.e. 845 Kbps (as is stated above, the maximum available bandwidth is 1.69 Mbps when the data rate of the WNIC is 2 Mbps). Fig. 4 shows that the throughput estimated is very close to the expected results. Again, the best accuracy is achieved with the largest client buffer size.

VII. Experimental Evaluation

We implemented the *RT_PS* protocol in a prototype version of our architecture and conducted an extensive experimental analysis to evaluate its performance. In our analysis we first considered a scenario with a single streaming session from the proxy to the mobile client. Then, we also considered the more general case with multiple simultaneous streaming sessions.

We evaluated the energy efficiency of our protocol by measuring the power saving index I_ps which is the ratio between the energy consumed during the streaming session by the WNIC of the client device when the *RT_PS* protocol is used, and the energy consumed without the *RT_PS* protocol [9], [10]. More formally, I_ps is defined as follows:

$$I\_ps = \frac{E_{RT\_PS}}{E_{\overline{RT\_PS}}}; \tag{7}$$

where:

$$E_{RT\_PS} = (T_{ON} \cdot W_{ON}) + (T_{SLEEP} \cdot W_{SLEEP}), \tag{8}$$

and

$$E_{\overline{RT\_PS}} = W_{ON} \cdot (T_{SLEEP} + T_{ON}). \tag{9}$$

In Equations 8 and 9 T_{ON} is the sum of all the ON periods, while T_{SLEEP} is the sum of all the OFF periods. The power consumption of the WNIC in the active mode was approximated by a constant value, irrespective of the specific state (rx, tx, idle) the WNIC was operating in. Specifically, we considered $W_{ON} = 750mW$ and $W_{SLEEP} = 50mW$ [27].

Finally, it should be pointed out that each experiment was replicated three times, and the results reported below are the average values over the three replicas.

A. Experiences with a single streaming session

In the first set of experiments we considered a single streaming session from the proxy to the client, and investigated the performance of the *RT_PS* protocol for different data rates of the WNIC (we manually set the WNIC to work at a constant data rate of either 2 Mbps, 5.5 Mbps or 11 Mbps during each single experiment) and different sizes of the client buffer (from 1000 KB down to 50 KB). We also analysed the impact on the performance of some background traffic.

When there is no background traffic the throughput achieved by the streaming session is maximum, and the proxy stops transmitting data to the client only when the client buffer is full. As expected, the *RT_PS* protocol performs better for greater data rates. This is because when the data rate is greater the data transfer is faster, and the WNIC is active for shorter time. We found that, with a WNIC data rate of 11 Mbps, I_ps is almost constant (i.e., it exhibits only a slight dependence on the client buffer size), and ranges from 8.50% (10 MB) to 8.63% (50 KB). Hence, the *RT_PS* protocol allows up to 91.50% saving of the energy consumed when no energy management policy is used. When the WNIC data rate decreases, the I_ps index increases, i.e., the energy saving decreases. With a WNIC data rate of 5.5 Mbps, the I_ps index ranges from 9.66% (10 MB) to 10.33% (50 KB), and with a data rate of 2 Mbps it ranges from 13.90% (10 MB) to 14.23%

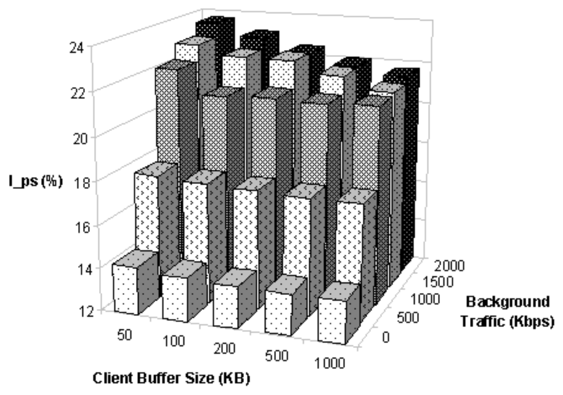

Fig. 5. I_ps experienced by a streaming flow competing with CBR background traffic. Min burstness: 1 packet per single burst.

(50 KB). In the worst case the energy saving is greater than 85%. The above results show that the I_ps index increases as the client buffer size decreases. These variations can be explained as follows. As anticipated in Section VI, when the buffer size decreases the following side effects occur: i) the amount of control traffic increases, ii) the accuracy of the estimator of the available bandwidth decreases, and iii) the schedule is characterized by a *greater* number of *shorter* ON periods. Due to i) the WNIC consumes more energy because is active for longer time to support the increased control traffic. Moreover, since for small buffer sizes the estimator tends to underestimate the available bandwidth, the WNIC remains active for much more time in order to prevent buffer underflows. Finally, a greater number of short ON periods in the transmission schedule contributes to increase the energy consumed in frequently switching the WNIC on and to sleep mode.

TABLE II
I_ps VS. CLIENT BUFFER SIZE. WORST CASE: 2 MBPS DATA RATE, 2 MBPS BACKGROUND TRAFFIC.

Client Buffer Size (Kbytes)	**Average Transmission Rate (Mbps)**	I_ps **(%)**
1000	0.815	21.62
500	0.795	21.99
200	0.773	22.49
100	0.738	23.06
50	0.711	23.65

To investigate the impact of the interfering traffic on the protocol performance we introduced an additional mobile host sending CBR (Constant Bit Rate) traffic, i.e., a periodic flow of UDP packets, to the proxy. We tested our system in the worst case, i.e., with a WNIC data rate of 2 Mbps. Fig. 5 summarizes the results obtained from the experiments with the background CBR traffic. It clearly emerges that I_ps increases when the amount of background traffic increases, thus leading to a higher energy consumption. This is because the streaming session experiences a lower throughput and the WNIC must be active for longer time since the client buffer fills up slowly and the risk of playback starvation is higher. When the rate of the background flow increases beyond 845 Kbps the wireless bandwidth is almost equally shared between the streaming flow and the CBR flow, i.e., the throughput experienced by the streaming session is approximately constant (845 Kbps is half of the maximal bandwidth, 1.69 Mbps, experienced with 2 Mbps data rate). Therefore, for a given buffer size, I_ps tends to flatten. Table II shows the performance of the *RT_PS* protocol in the worst conditions, i.e., when the rate of the background traffic is 2 Mbps and, hence, the interfering mobile host has always a packet ready for transmission. I_ps is, at most, equal to 23.65% which corresponds to an energy saving of 76.35%.

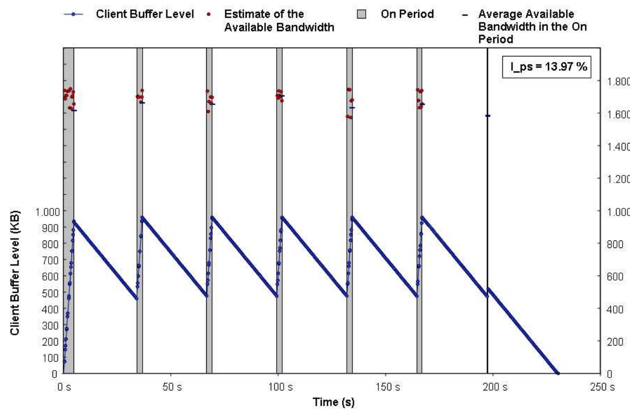

Fig. 6. Variations in the client buffer level during the streaming session. The client buffer size is 1000 KB. No background traffic is present.

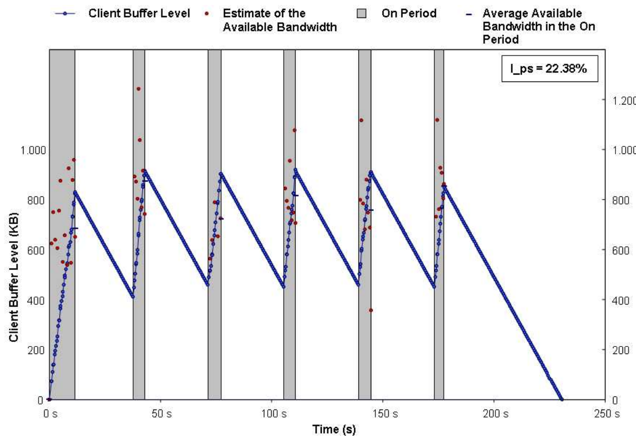

Fig. 7. Variations in the client buffer level during the streaming session. The client buffer size is 1000 KB. Maximal background traffic is present.

To conclude our analysis on the impact of the background traffic, we now show the evolution over time of the following parameters: client buffer level, ON and OFF durations, and estimated bandwidth. Fig. 6 shows the system behavior when there is no background traffic, while Fig. 7 shows the same behavior when the background traffic is maximum. Fig. 6 shows that the estimated bandwidth (the dots inside the ON periods) is high because there is no interfering traffic, and the buffer level grows up until the buffer becomes full. During the OFF periods the playback process consumes data in the client buffer at a constant rate and the buffer level decreases linearly. In Fig. 7 the bandwidth estimated is lower because of the background traffic. Therefore, the buffer level during the ON periods increases at a slower rate than in Fig. 6. Moreover, the ON periods are longer compared with those in Fig. 6. Finally, there is a higher fluctuation in the estimates of the available bandwidth. Nevertheless, the available bandwidth is on average about 845 Kbps, i.e., half the total available bandwidth. This proves that the bandwidth is equally shared by the two competing traffic flows.

TABLE III
I_ps VS. BURST SIZES IN THE BACKGROUND TRAFFIC. 1 MBPS BACKGROUND TRAFFIC, 2 MBPS DATA RATE

Buffer Size(KB)	Burst Size (pkts) 1	10	50	100	200
1000	21.21	20.27	18.76	18.27	16.70
500	21.15	20.65	18.82	19.20	16.51
200	21.23	20.72	18.49	17.69	17.47
100	21.13	20.44	16.76	14.76	18.05
50	22.22	20.65	18.20	18.65	17.71

Finally, we investigated the effects on I_ps of the *burstiness* in the background traffic. To this end we conducted a set of experiments where the CBR traffic generator delivered a *burst* of packets at a time instead of a single packet at a time. We used burst sizes of 10, 50, 100, 200 packets in our experiments. Since the rate of the background traffic is kept constant, an increase in the burst size results in a longer inter-time between two consecutive bursts.

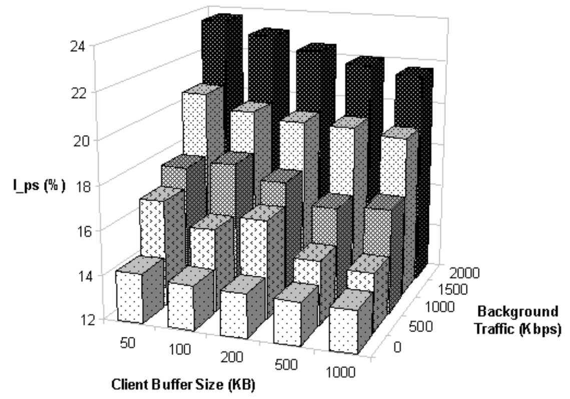

Fig. 8. I_ps experienced by a streaming flow competing with a bursty CBR traffic. Max burstness: 200 packets per single burst.

Table III shows the I_ps values for different burst and client buffer sizes. These results were obtained with a WNIC data rate of 2 Mbps and background traffic of 1 Mbps. They show that as the burst size increases the energy efficiency increases accordingly. This is because the streaming session takes advantage of the non-transmission periods in the background traffic. When the *RT_PS* proxy component accesses the wireless channel, it may be either busy or free depending on whether the generator of the background traffic is producing a burst or not. If the generator is silent the proxy starts transmitting immediately. On the other hand, if a burst is flowing up the *RT_PS* protocol estimates a low available bandwidth and the proxy defers the transmission. Finally, if a new burst starts in the middle of a transmission causing the estimated available bandwidth to go down, the proxy stops transmitting.

Intuitively, for a given background traffic rate, an increase in the burst size leads to longer time intervals between consecutive bursts. During such intervals the proxy can exploit the maximum bandwidth to fill up the client buffer, and hence the performance approaches the case without background traffic. The best condition occurs when the proxy finds the channel free for enough time to fill up the client buffer. The streaming flow and the background traffic may become somewhat synchronized, i.e., the ON periods of the streaming flow fit into the silent periods of the background traffic and vice-versa. In this case, both the streaming and the background traffic flows can thus exploit the maximum available bandwidth.

Fig. 8 summarizes the results obtained from the set of experiments with the bursty background traffic. By contrasting Fig. 8 with Fig. 5, it clearly appears that higher energy

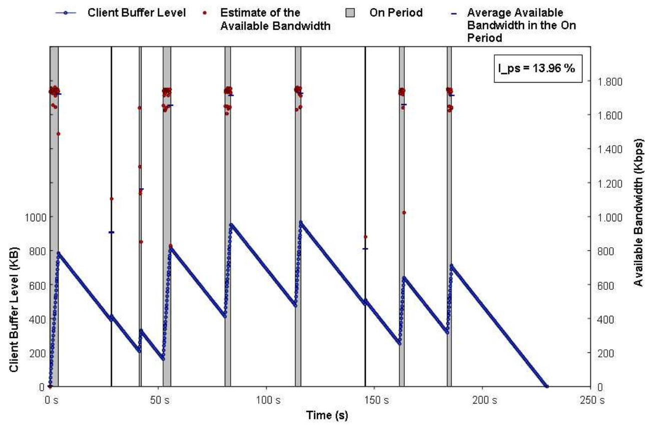

Fig. 9. Variations in the client buffer level during the streaming session. The client buffer size is 1000 KB. The background traffic is 500 Kbps with 200 packets sent per single burst.

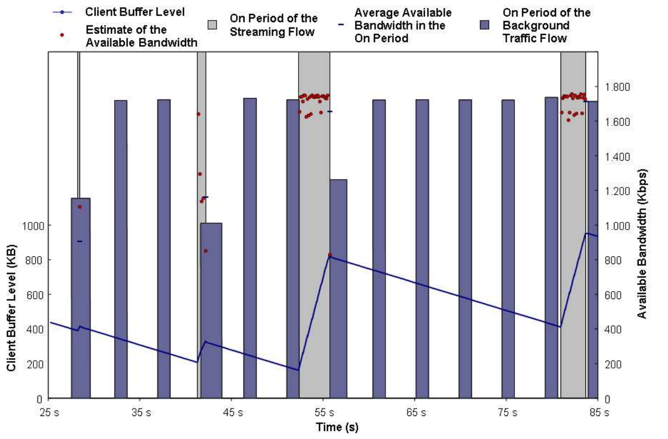

Fig. 10. Variations in the client buffer level during the streaming session from 25 s to 85 s. The client buffer size is 1000 KB. The background traffic is 500 Kbps with 200 packets sent per single burst.

efficiency is achieved when the burstiness in the background traffic increases.

Fig. 9 shows the client buffer level evolution during an entire streaming session in presence of background traffic of 500 Kbps. The background traffic generator generates 200 packets per burst. The client buffer size is 1000 KB. It appears that the *RT_PS* protocol is able to adapt to the background traffic behavior as the ON periods of the streaming session correspond almost always to the idle periods in the background traffic (the available bandwidth estimated during the ON periods is very high). To better highlight this behavior a portion of Fig. 9 is magnified in Fig. 10 where both the ON periods in the streaming session and the idle phases in the background traffic are shown. The first ON period in the streaming flow starts when a burst is in progress. The available bandwidth estimated by the *RT_PS* protocol is thus low and the mobile client enters the sleep mode. The second ON period begins during an idle phase in the background traffic. However, the traffic generator starts sending a new burst soon later. As soon as the estimated bandwidth falls below a threshold the *RT_PS* protocol decides to defer the transmission. Finally, during the third and fourth periods no significant overlap is experienced, and data flows at the maximum speed. In conclusion, the *RT_PS* protocol is able to adapt to the background traffic behavior. The throughput experienced during the ON periods is thus very high, as if there was no competing traffic on the channel, and the energy efficiency is very good.

B. Experiences with two simultaneous streaming sessions

In this section we will analyse the performance of the *RT_PS* protocol when there are more simultaneous streaming sessions. Specifically, in our experiments we will consider two concurrent streaming sessions with clients running on two different mobile hosts. In some experiments we will also consider a third mobile host generating interfering traffic.

Fig. 11 shows the I_ps index achieved by both clients when no background traffic is injected in the network. In this set of experiments the two streaming sessions start at the same time. The data rate of the WNICs is 2 Mbps. As is highlighted in the figure, there is no significant difference in the performance achieved by the two sessions. At a smaller scale, results show a slightly favorable trend towards bigger buffer sizes, as expected. Anyway, we can conclude that the energy saving achieved by both clients is always high. It is worth pointing out that, when there is no background traffic, no streaming session undergoes contention over the wireless LAN, since both the streaming traffic flows flow from the proxy to the client. However, depending on the particular streams and on their starting points, they may or may not achieve the I_ps value experienced in the same conditions by a single streaming session alone. Actually, when the ON periods of both streams overlap, the proxy has to schedule packets for both of them. By assuming a simple round-robin policy, both streams will experience half the maximum bandwidth. As anticipated in the previous section, this leads to a higher energy consumption. On the other hand, both streams achieve the maximum energy saving when the ON periods do not overlap. To conclude our

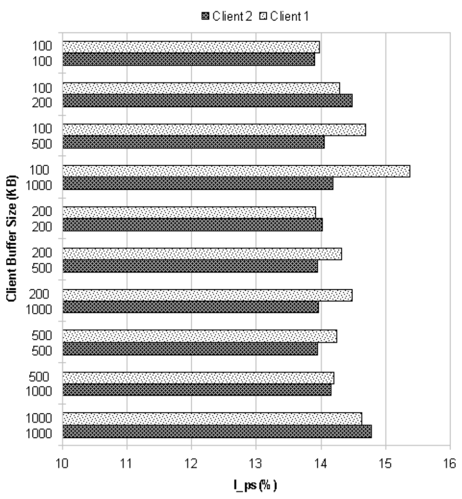

Fig. 11. Two competing streaming flows with no background traffic: I_ps experienced when combining different client buffer sizes for the two flows.

analysis, we considered the impact of some background traffic also in this second scenario. We considered both UDP and TCP interfering traffic flows (see Fig. 12). As expected, when introducing UDP traffic the I_ps index of both the streaming flows increases (i.e., the energy efficiency decreases). The worst case is experienced when the background traffic reaches half the total bandwidth of the wireless channel. However, there are not significant differences in the I_ps values achieved by the two streaming flows. When the interfering traffic is TCP, the I_ps value experienced by the two streaming flows slightly worsens. The additional bandwidth required by the TCP ACKs can be accounted for this behavior. Again, there is no significant difference in the I_ps values achieved by the two streaming flows.

The above analysis was aimed at characterizing the energy efficiency of our system. However, in a multimedia streaming application the frame loss and initial playback delay are two important performance figures that need to be taken into consideration as well. In all our experiments the initial playback delay was always less than $2s$. Moreover, the fraction of frames discarded by the proxy (because expected to arrive late for playback) or lost during the transmission was always below 5% of the total number of frames. This percentage is considered largely acceptable since, in real-time audio streaming, up to 20% loss can be tolerated if loss concealment techniques are implemented at the receiver.

VIII. Conclusions

In this paper we have considered a mobile wireless scenario for multimedia streaming applications, and proposed an architecture for providing energy-efficient streaming services to mobile clients. Specifically, we have focused on MP3 audio streaming in a Wi-Fi environment. Our architecture is based on the server/proxy/client model, where the proxy functionality is implemented at the Access Point (or at a fixed host on the same LAN of the Access Point). The architecture relies on the *RT_PS* (*Real Time and Power Saving*) protocol between the proxy and the mobile client. It implements an energy

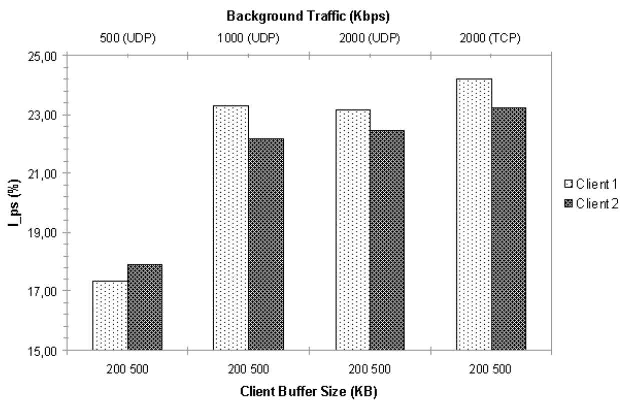

Fig. 12. I_ps experienced in two competing streaming flows when adding background traffic of different types.

management policy whose main goal is to minimize the energy consumed by the wireless network interface at the mobile host, while guaranteeing the real-time constraints of the streaming application. The protocol uses an adaptive ON/OFF schedule for the transmission of audio frames to the mobile client, and switches the wireless network interface to the low-power sleep mode during the OFF periods. The *RT_PS* scheduling policy relies on the availability of a buffer at the client where frames can be stored before their playback times, and is mainly based on the following heuristics. When the available bandwidth on the wireless link is high the proxy transmits data until the client buffer becomes full. When the available bandwidth is low -due to competing traffic on the wireless network- the proxy defers the data transmission to better times. However, if the client buffer level becomes too low the proxy transmits data irrespective of how much bandwidth is available. To evaluate the performance of our solution in a real environment we implemented our *RT_PS* protocol in a real prototype system based on the Wi-Fi technology. The results obtained by means of an extensive experimental analysis have shown that our solution can save up to 91.50% of the energy consumed by the Wi-Fi interface at the mobile host when no energy management policy is used. The energy efficiency of the protocol proposed depends on the data rate of the wireless interface (it increases as the data rate increases) and the client buffer size (it increases as the buffer size increases). However, even with small buffer sizes (e.g., 50 KB) the energy saving is very high. We also investigated the impact of some interfering traffic on the system performance. Even though the competing traffic reduces the energy efficiency, the measured energy saving is never less than 75%. Interestingly, we also found that the energy efficiency increases when the burstiness in the interfering traffic is higher.

Acknowledgments

This work has been carried out partly under the financial support of the *Italian Ministry for Education and Research (MIUR)* in the framework of the *VICOM* project, and partially under the financial support of the *Fondazione Cassa di Risparmio di Pisa* in the framework of the *SMILE* project.

References

[1] G. Anastasi, M. Conti, A. Passarella. Power Management in Mobile and Pervasive Computing Systems. *Algorithms and Protocols for Wireless and Mobile Networks*, Azzedine Boukerche (Editor): CRC_Hall Publisher, 2005.
[2] Feng and J. Rexford, A comparison of bandwidth smoothing techniques for the transmission of prerecorded compressed video, *in Proc. of the IEEE INFOCOM*, Apr.1997, pp. 58-66.
[3] Web site of IEEE 802.11 WLAN: http://grouper.ieee.org/groups/802/11/main.html.
[4] G. Anastasi, M. Conti, W. Lapenna. Power Saving Policies for Wireless Access to TCP/IP Networks. *in Proc. of the* 8^{th} *IFIP Workshop on Performance Modelling and Evaluation of ATM and IP Networks (IFIP ATM&IP2000) - Ilkley (UK)*, July 17-19, 2000.
[5] R. Kravets, P. Krishnan. Power Management Techniques for Mobile Communication. *in Proc. of the* 4^{th} *Annual ACME/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, 1998.
[6] M. Anand, E.B. Nightingale, and J. Flinn. Self-Tuning Wireless Network Power Management. *Wireless Networks*, **11**(4), 2005.
[7] A. Bakre, B. R. Badrinath. Implementation and Performance Evaluation of Indirect TCP. *IEEE Transactions on Computers*, **46**(3), March 1997.
[8] M. Stemm, R. H. Katz. Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices. *in Proc. of the* 3^{rd} *International Workshop on Mobile Multimedia Communication - Princeton, NJ*, September 1996.
[9] G. Anastasi, M. Conti, E. Gregori, A. Passarella. Performance Comparison of Power Saving Strategies for Mobile Web Access. *Performance Evaluation Journal*, **53**(3-4): 273-294, 2003.
[10] G. Anastasi, M. Conti, E. Gregori, A. Passarella. Balancing Energy Saving and QoS in the Mobile Internet: An Application-Independent Approach. *in Proc. of the* 36^{th} *Hawaii International Conference on System Sciences (HICSS-36) - Hawaii*, January 6-9, 2003.
[11] S. Acharya and B. C. Smith. Compressed Domain Transcoding of MPEG. *In Proc. of IEEE Conference on Multimedia Computing Systems (ICMCS) - Austin TX*, 1998.
[12] N. Bjork and C. Christopoulos. Video transcoding for universal multimedia access. *in Proc. of the* 8^{th} *ACM Conference on Multimedia - Los Angeles, CA*, November 2000.
[13] S. Chandra, A. Vahdat. Application-specific Network Management for Energy-Aware Streaming of Popular Multimedia Formats. *in Proc. of the General Track: 2002 USENIX Annual Technical Conference*, 2002.
[14] S. Chandra. Wireless Network Interface Energy Consumption Implications of Popular Streaming Formats. *Multimedia Systems*, **9**(2), August 2003.
[15] P. Shenoy, P. Radkov. Proxy-Assisted Power-Friendly Streaming to Mobile Devices. *Multimedia Computing and Networking*, 2003.
[16] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. *Internet-draft, updates RFC 1889*, http://www.ietf.org/internet-drafts/draft-ietf-avt-rtp-new-12.ps, March 2003.
[17] H. Schulzrinne. RTP site. http://www.cs.columbia.edu/ hgs/rtp/, 1999.
[18] H. Schulzrinne, A. Rao, R. Lanphier. Real Time Streaming Protocol (RTSP). *Internet-draft, updates RFC 2326*, http://www.rtsp.org/2004/drafts/draft06/draft-ietf-mmusic-rfc2326bis-06.txt, February 2004.
[19] M. Handley, S. Floyd, J. Padhye, J. Widmer. TCP-Friendly Rate Control (TFRC): Protocol Specification. *RFC 3448*, http://www.faqs.org/rfcs/rfc3448.html, January 2003.
[20] C. Huitema. Real Time Control Protocol (RTCP): attribute in Session Description Protocol (SDP). *RFC 3605*, http://www.faqs.org/rfcs/rfc3605.html, October 2003.
[21] L. Gharai. RTP Profile for TCP-Friendly Rate Control. *Internet-draft*, http://macc.east.isi.edu//tfrc-profile.txt, August 2004.
[22] G. Anastasi, M. Conti, E. Gregori, A. Passarella, and L. Pelusi. A Power-Aware Multimedia Streaming Protocol for Mobile Users. *Technical report, IIT-CNR*, available on-line at http://bruno1.iit.cnr.it/ bruno/techreport.html, February 2005.
[23] J. Zhang. Optimal buffering algorithms for client-server VBR video retrievals. *PhD thesis - Rutgers University*, 1996.
[24] J. Zhang, J. Y. Hui. Traffic characteristics and smoothness criteria in VBR video traffic smoothing. *in Proc. of the IEEE International Conference on Multimedia Computing and Systems*, June 1997.
[25] L. Huang, F. Hartung, U. Horn, M. Kampmann. A Proxy-based TCP-friendly streaming over mobile networks. *in Proc. of the* 5^{th} *ACM international workshop on Wireless mobile multimedia - Atlanta*, September 2002.

[26] R. Finlayson. A More Loss-Tolerant RTP Payload Format for MP3 Audio. *Internet-draft, updates RFC 3119*, http://www.cs.columbia.edu/ hgs/rtp/drafts/draft-ietf-avt-rfc3119bis-03.txt, October 2004.

[27] R. Krashinsky, H. Balakrishnan. Minimizing Energy for Wireless Web Access with Bounded Slowdown. *in Proc. of the ACM International Conference on Mobile Computing and Networking (Mobicom)*, 2002.

Giuseppe Anastasi is an associate professor of Computer Engineering at the Department of Information Engineering of the University of Pisa, Italy. His research interests include mobile and pervasive computing, ad hoc and sensor networks, and power management. He was a co-editor of the book *Advanced Lectures in Networking* (LNCS 2497, Springer, 2002), and published more than 50 papers in the area of computer networking and pervasive computing, both in international journals and conference proceedings. He is on the editorial board of the *Journal of Ubiquitous Computing and Intelligence (JUCI)*, and is currently serving as Program Chair for the 4^{th} *International Workshop on Mobile Distributed Computing* (MDC 2006). He is also Workshop Chair for *IEEE PerCom 2006* and *IEEE WoWMoM 2006*. He served as General Co-chair for *IEEE WoWMoM 2005*, and program co-chair for the 1^{st} *International Workshop on Sensor Networks and Systems for Pervasive Computing* (PerSeNS 2005). He served in the Technical Program Committee of several international conferences including *IEEE PerCom*, *IEEE ISCC*, and *IEEE AINA*. He has been a member of the IEEE Computer Society since 1994.

Marco Conti is a senior researcher at IIT, an institute of the Italian National Research Council (CNR). His research interests include Internet architecture and protocols, wireless networks, ad hoc networking, mobile and pervasive computing. He co-authored the book *Metropolitan Area Networks* (Springer, London 1997) and is co-editor of the book *Mobile Ad Hoc Networking* (IEEE-Wiley 2004). He published in journals and conference proceedings more than 150 research papers related to design, modeling, and performance evaluation of computer-network architectures and protocols. He served as TPC chair of IFIP-TC6 Conferences *Networking2002* and *PWC2003*, and as TPC co-chair of *ACM WoWMoM 2002*, IFIP-TC6 *WONS 2004*, *WiOpt '04*, and the 6^{th} *IEEE WoWMoM 2005* Symposium. He is the TPC chair of *IEEE PerCom 2006* conference, TPC co-chair of *ACM MobiHoc 2006* Symposium, and general co-chair of the 7^{th} *IEEE WoWMoM 2006* Symposium. He is Associate Editor of *Pervasive and Mobile Computing* Journal, and he is on the editorial board of: *IEEE Transactions on Mobile Computing*, *Ad Hoc Networks* journal and *Wireless Ad Hoc and Sensor Networks: An International Journal*. He served as guest editor, among others, for *IEEE Transactions on Computers*, *ACM MONET*, *ACM WINET*, and *Performance Evaluation*. He is member of ACM, IEEE, and IFIP WGs 6.2, 6.3, and 6.8.

Enrico Gregori received the Laurea in electronic engineering from the University of Pisa in 1980. He joined CNUCE, an institute of the Italian National Research Council (CNR) in 1981. He is currently a CNR research director. In 1986 he held a visiting position in the IBM research center in Zurich working on network software engineering and on heterogeneous networking. He has contributed to several national and international projects on computer networking. He has authored more than 100 papers in the area of computer networks and has published in international journals and conference proceedings and is co-author of the book *Metropolitan Area Networks* (Springer, London 1997). He was the General Chair of the second IFIP-TC6 Networking Conference *Networking2002*. His current research interests include: Wireless access to Internet, Wireless LANs, Quality of service in packet-switching networks, Energy saving protocols, Evolution of TCP/IP protocols. He is on the editorial board of the *Cluster Computing* Journal.

Andrea Passarella is a Research Associate at the Computer Laboratory of the University of Cambridge, UK. He received a PhD and MS Degree in Computer Engineering, both from the University of Pisa, Italy, in 2005 and 2001, respectively. His current research is mostly on ad hoc and sensor networks, specifically on p2p systems, multicasting, transport protocols, and energy-efficient protocols. His research interests also include opportunistic and delay-tolerant networking, and wireless access to the Internet. In 2002 he won the Vodafone Prize for the Best Italian MS Thesis of the year in the field of Technology Innovation. He served and is currently serving in the TPC of several international conferences and workshops, including *PerCom 2006* and *WoWMoM 2006*. He was TPC Vice-Chair for *REALMAN 2005*, and is currently TPC Vice-Chair for both *MDC 2006* and *REALMAN 2006*. He is also an Associate Technical Editor for *IEEE Communications Magazine*.

Luciana Pelusi received the Laurea degree in Computer Engineering from the University of Pisa, Italy, in 2003. She is a Ph.D. student at the Institute for Informatics and Telematics of the Italian National Research Council (IIT-CNR) in Pisa. Her research interests are in the area of pervasive systems and include multimedia networking, energy-efficient protocols for ad hoc and sensor networks, and opportunistic networking.