Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Networks 49 (2005) 4–26
Computer Networks
www.elsevier.com/locate/comnet

For here or to go? Downloading music on the move with an ultra reliable wireless Internet application

V. Ghini [a], G. Pau [b], M. Roccetti [a,*], P. Salomoni [a], M. Gerla [b]

[a] University of Bologna, Department of Computer Science, Mura A. Zamboni 7, Bologna 40127, Italy
[b] UCLA, Computer Science Department, CA 90095, USA

Available online 19 May 2005

Abstract

The maturing distributed file sharing technology implemented by Napster has first enabled the dissemination of musical content in digital form, permitting to customers to retrieve stored music files from around the world. In the post-Napster era, the Apple iTunes online music service has hit a record share of 16.7% in the MP3 player market [J. Mc Hugh, Why Wi-Fi is a good business? Wired (2003) 25–26]. This is only the most prominent example of the success of digital music distribution based on packet network technologies. However, to the best of our knowledge, the most noteworthy aspect of the success of digital music distribution is that little about this music delivery technology is really new. To drastically change the nature of this business, we claim that wireless delivery technology must come into the picture. This way, the digital music delivery model will benefit from the integration of the wired Internet with a broad gamut of wireless access technologies, such as WiFi, WPAN and 3G. In this paper we address the problem of the "customer on the move"; we present a wireless Internet application designed to support the distribution of digital music to handheld devices. The main novelty of our software is the ability to provide a seamless music delivery service even in the presence of handoffs within the same radio medium (horizontal) and across media (vertical). Actual measurements from a deployed application show that our system can deliver a smooth, ultra reliable, low latency music service to mobile users.

Keywords: Internet wireless applications; Inter-technologies roaming; Music delivery technologies; Music consumer networking; Wireless networks for entertainment

1. Introduction

Napster has been probably the most revolutionary and unprecedented (in terms of scale) example of digital music distribution over packet switched

* Corresponding author. Tel.: +39 051 2094503; fax: +39 051 2094510.
E-mail addresses: ghini@cs.unibo.it (V. Ghini), gpau@cs.ucla.edu (G. Pau), roccetti@cs.unibo.it (M. Roccetti), salomoni@cs.unibo.it (P. Salomoni), gerla@cs.ucla.edu (M. Gerla).

1389-1286/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2005.04.002

networks [1,2]. In the post-Napster era, several major record labels and Internet providers launched their own versions of MP3-based music delivery causing a proliferation of content (music, video and games) distribution techniques to wired devices on a very large scale. Perhaps the latest and most famous example of widespread access to a music storehouse is the *Apple iTunes Music Store* [3]. With such a system, each browser-equipped music consumer can search new songs as well as popular hits he/she has not heard in years. Previews may be played for free and the song downloaded in pristine digital quality at a low cost, with just one click.

A new problem and opportunity is now emerging as users equipped with Internet-enabled cellular phones and other handheld devices, want to link to music content on the Web. One may argue that the current trend in business and technology in digital music distribution can be significantly altered only if the music delivery scheme becomes wireless. However, a wireless music delivery model will be successful only if it guarantees affordable access and reliable service [4].

In this context, modern radio technologies will play a key role in providing a large-scale, wireless infrastructure for music delivery to mobile listeners.

From a technology standpoint, the most widely used standard for second generation (2G) mobile radio networks is the so called *Personal Communications Service* (PCS). This term describes a set of digital cellular technologies being deployed in the US, and working over CDMAone (also called IS-95A), *Global System for Mobile communications* (GSM) and North American TDMA (also called IS-136) air interfaces [5]. With the introduction of the *General Packet Radio Service* (GPRS) in mobile GSM networks in Europe, packet data connections have been allowed with rates up to the order of 50 Kb/s (a comparable 2.5G wireless radio technology in US is CDMAone, IS-95B).

Third generation (3G) mobile systems, using either the *Wideband Code Division Multiple Access* (W-CDMA) or the CDMA-2000 radio technologies, will soon offer higher data rates of up to a few Mb/s and an increased capacity on a large scale. Thus, the data rates offered by the (3G) radio technologies, plus appropriate compression techniques, will eventually enable consumers to access video/audio entertainment services even from smart phones and similar smaller devices.

Besides the above mentioned radio cellular infrastructures, WLAN/WPAN technologies, such as, for example, WiFi and Bluetooth, are emerging as robust, *locally* available and low cost means for the deployment of music-on-demand distribution services. Naturally, since WLAN/WPANs are interconnected with the Internet, and since the musical resources are distributed on an ubiquitous, anytime, anywhere basis over the Web, the users under WLAN/WPAN coverage have unlimited access to all those resources.

Since the mobile user will move freely across the above telecommunication infrastructures (e.g., UMTS), continuous local/personal area network connectivity is an opportunity to drastically enhance the range, throughput, availability and performance of a music distribution service.

In particular, stimulating scenarios for music distribution that may take benefit by the integrated use of global/local wireless infrastructures include the following:

1. *Music Delivery over cellular networks*: a wireless entry point to music delivery networks (e.g., wireless music portals) may represent an innovative and value-added service offered by cellular carriers to those mobile customers wishing to use their 2.5/3G-enabled cellular phones for playing songs, as well as other enhanced multimedia objects. In this context, traditional Internet-based music providers (or music producers) may exploit 2.5/3G cellular technologies to extend the reach of their Web-based music services to nomadic music listeners.
2. *WLAN-based music showers*: integrated wired/wireless infrastructures may be built to support the setting up of music showers that distribute digital musical contents to authorized customers. Obviously, those customers must be under the reach of the corresponding WLAN access point. This may be the typical case for music Internet cafés, music kiosks, and cyber music-saloons.

3. *Opportunistic music communities*: a *music community* is a community of several users of which only a few are under the coverage of a music shower. Contents may be forwarded from covered members to all the other members, hopping from user to user, as an extension of the existing wireless network (WLAN/WPAN or cellular). A prominent example of opportunistic music communities are WLAN/WPAN-equipped drivers on roadways with music showers distributed along the way. Another example are *music nomads* who roam in a section of the city, for business or pleasure wandering about nearby hotels, parks, shops and college dorms. Only those exposed to music showers are able to download their desired songs. The other members of the community, constantly probe neighbors to share the downloaded musical content.

Interesting technical problems arise when wireless technologies are integrated with the fixed Internet, with the aim of distributing music contents to mobile devices.

Take TCP, for example,designed to interpret unexpected delays and packet losses as symptoms of network congestion. In a wireless environment packet losses and delays are often caused by mobility, such as handovers, transmission errors and temporary link outages. Hence, if mobility is mistaken for congestion, and aggressive TCP congestion prevention is activated, then further performance degradation is experienced even beyond that caused by the wireless environment [6,7].

If the wireless link is interrupted for a prolonged time (as during a handoff, or driving in a tunnel), the song download may just be aborted, adding to user frustration. To address the latter point, an important requirement is to allow each handheld device to function as any other Internet-connected device. In essence, seamless internetworking must be provided between the wired and the wireless segments of the communication.

Besides the end-to-end wired/wireless IP continuity, mobile clients should be able to benefit from a heterogeneous wireless environment where different mobile network technologies (e.g., 802.11, 3G, satellite, etc.) can alternatively offer wireless access at different quality and price to the available music showers. Simply put, a crucial point for wireless music distribution is that of providing an ''intertech'' roaming scheme that allows mobile clients to enjoy a continuous online music service (at the best price) while passing by different areas that are under the coverage of alternative wireless technologies in the heterogeneous wireless infrastructure [8,9].

To address the needs of the *music nomad* we have designed, developed and field tested a mobile software application [10] that allows users to enjoy an online music-on-demand service on wireless devices.

A *Session Management Layer* (SML) permits online music distribution to mobile devices with the Internet as a backplane. SML supports:

1. end-to-end IP continuity of a download activity in the face of failures, link outages or handovers, and
2. a price/performance-sensitive vertical roaming scheme for switching across alternative wireless technologies (e.g., WiFi, WPAN and cellular).

While TCP adjusts its parameters to match the unstable requirements of the wireless environment, SML aims at ensuring a successful termination of the music download activity even when:

- the underlying link connections are damaged or disrupted due to device mobility (*horizontal handoffs*), and/or
- a mobile user passes through different areas with signal coverage provided by different alternative wireless technologies (*vertical handoffs*).

As explained in the following Section 2, this goal is achieved by freezing the download state activity at the session layer when horizontal and/or vertical handoffs are detected, thus permitting resumption of the music data stream as soon as the causes that caused the interruption of the music download activity have cleared. Needless to say, as typical downloadable songs amount to large MP3 files (3/4 MB, on average), seamless music distribution in the presence of horizontal and vertical handoffs, is of great value. In our

experiments we have assessed the effects that our wireless Internet application has on the download performance of digital music content. The results indicate that our wireless applications can deliver an ultra reliable and responsive music service to mobile users.

The reminder of this paper is organized as follows. In Section 2, we discuss the architecture design. Section 3 presents real-world experimental scenarios and reports on the field trial results. In Section 4, related work is discussed. Section 5 concludes the paper.

2. System architecture

The general architecture of our proposed wireless application (Fig. 1) is comprised of the following three software components [10]:

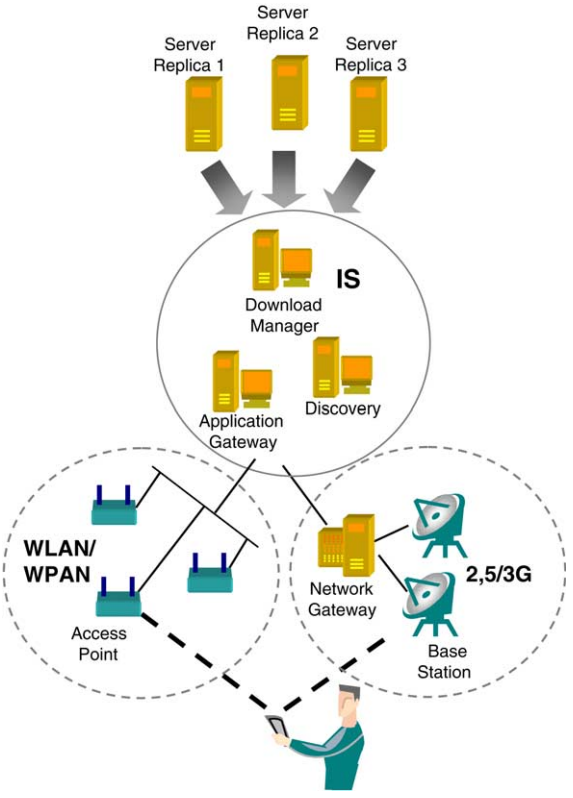


Fig. 1. System architecture.

1. *Mobile Clients*, i.e., music handheld devices connected through wireless network access points. This part of the wireless application supporting mobile clients has the responsibility for the subsequent activities of: (i) searching the MP3 files corresponding to the client's favorite songs over the Internet, (ii) downloading them on the handheld device, and, finally, (iii) playing them as soon as the download has been completed. Currently, several alternative wireless access network technologies are supported, in particular WiFi (802.11b), Bluetooth, CDMA2000 (1xRTT) and GPRS.
2. The (IS) *Intermediate System*, this software component is hosted in an Internet server and represents the core of our application. It is in charge of managing all the communications between the handheld device and the wired Internet infrastructure. It consists of three software subsystems, namely the *Application Gateway*, the *Discovery* system and the *Download Manager*. These components are described below.
   (a) The *Application Gateway* accepts and manages all the requests for songs arriving from the client connected to a given mobile device. The main characteristic of this software component is that of guaranteeing reliable communication between the IS and the mobile client. In particular, the Application Gateway embodies a *Session Management Layer* (SML) that ensures seamless IP continuity in the face of possible link outages and handoffs over the wireless channel, while allowing the client to switch across different mobile technologies in the midst of a music download activity.
   (b) The *Discovery* system discovers the music resources on the Web (i.e., the MP3 files) which correspond to songs requested by the users.
   (c) The *Download Manager* downloads the songs that have been identified by the Discovery system. The Download Manager incorporates a novel mechanism that is able to engage concurrently several identical copies of a given music resource to speed up the download activity from the wired Internet

to the IS. With this particular mechanism, each client's request of a given music resource is fragmented into a number of sub-requests for separate parts of the resource. Each of these sub-requests is issued concurrently to a different available replica server, which possesses that resource. The mechanism periodically monitors the downloading performance of available replica servers and dynamically selects, at runtime, those replicas to which the client sub-requests can be sent, based on both the network congestion status and the replica servers' workload [11].

3. A set of *Web server replicas*. These are Web servers geographically distributed over the Internet which function as replicated music repositories. The decision of replicating each music resource over several servers was motivated by the goal of increasing service responsiveness. Simply put, this replication scenario can be thought of as a *loosely coupled* replication system where different servers may support different sets of songs, and each single song may be simultaneously replicated within a number of geographically dispersed Web servers.

Alongside a detailed description of a search/download/play session performed by our system (Section 2.1), in the following Sections 2.2 and 2.3 we discuss the design and the implementation of the protocol architecture we have devised to support the communication between the wireless client and the IS. Finally, in Section 2.4 we report on the protocol stack on which all the communication between the IS and the Web server replicas is based.

2.1. Search, download and play back of musical resources

A complete search/download/play session for music resources steps through three different phases. In the first phase, a user from his/her handheld device issues to the Application Gateway a request for a given song. (Such a request may refer either to a specific song author or to a given song title.) The Application Gateway passes this request down to the Download Manager. The Download Manager asks the Discovery for the complete list of all the available music resources matching the request issued by the user. Discovery performs the search of the songs required by the client. Such activity steps through two additional different phases and proceeds as follows.

First, Discovery tries to establish a relationship between the requested song titles and the system MP3 description files. (Note that different songs stored in different Web servers may match the request issued by the user.)

Once this activity is completed, Discovery passes to the user (via the Application Gateway) the list of all the songs (and corresponding files) that match the initial request. Upon receiving this list, the user chooses one of the proposed songs.

This choice activates an automatic process to download the correspondent MP3 file. It is the Download Manager, now equipped with all the relevant information needed to locate it that downloads the required MP3 file, and finally delivers it to the software application running on the mobile device.

It goes without saying that during this second phase, it is the responsibility of the Discovery subsystem to individuate the Web locations (URI) of all the copies of the chosen song, while the Download Manger carries out a concurrent fragment-based download activity by engaging all the different replica servers that maintain a copy of the requested MP3 song. As soon as the MP3 file arrives at the mobile client, the software application running on the mobile device starts the third and final phase amounting to the play back of the downloaded MP3 file.

Before all this happens, a preliminary phase has to be carried out where in each music repository announces the list of the songs it wishes to make available for distribution. Each music server which wants to add its own repository to our IS system may do so by running a software application called the Data Collector which is in charge of communicating the list of the clips along with the associated multimedia resources to Discovery.

2.2. Providing connection continuity on the wireless link

Our wireless application has been structured based on the use of an open IP approach where the mobile device functions as any other Internet connected device. End-to-end direct TCP/IP continuity is ensured by exploiting the TCP/IP protocol stack [12]. The main motivation is to provide a seamless internetworking scheme between the wired and the wireless segments.

Further, as one of the most crucial problems for multimedia distribution to wireless devices is that of an unexpected link interruption in the middle of a long download activity, we have augmented our wireless protocol architecture with a session layer developed on top of the transport protocol. The aim of our session layer is to guarantee that the music download is not compromised when long link outages or (horizontal/vertical) handoffs occur. It is very important to notice that here the problem is no longer one of adjusting the transport protocol performance to match the unstable requirements of the wireless environment. Rather, it is one of ensuring a successful completion of the download activity even when the underlining connection is broken at the transport layer due to device mobility.

Fig. 2 shows the protocol stack we have devised to provide support to all the communications between the mobile client and the Application Gateway. As seen from the figure, at the client side, different radio technologies may implement the data link layer at the basis of our protocol stack. On the top of this multi-protocol radio layer, a standard TCP/IP stack is installed that guarantees TCP/IP continuity between the wireless link and the wired segments of the Internet. The application layer built on top of TCP consists of two different sub-layers:

- a *Session Layer*: this protocol layer is devoted to managing a download session which provides users with the possibility of resuming a communication that was previously interrupted due to problems occurring at the lower levels of our communication architecture,
- a *Music Application Layer*: this protocol layer is in charge of supporting the different connections needed to search, download and play songs.

It is well-known that the standard implementation of the TCP protocol interprets unexpected delays and packet losses as symptoms of network congestion.

In such a case, the standard implementation of TCP reacts by triggering an aggressive back off procedure that reduces the congestion window to achieve connection stability. Unfortunately, unexpected delays and packet losses on a wireless link may be caused not by congestion, but by mobility,

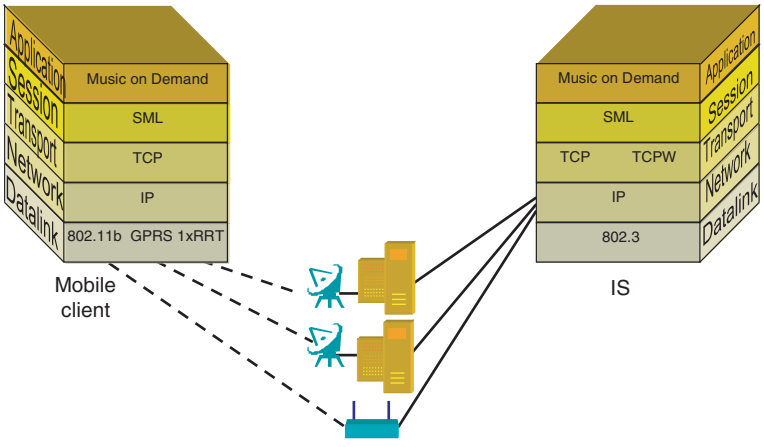


Fig. 2. Mobile client-IS: protocol architecture.

such as handoffs, link outages and severe transmission errors. Hence, if mobility is mistaken for congestion and aggressive congestion control procedures are activated, the overall performance of the wireless connection is drastically reduced with a significant performance degradation of the whole system.

To overcome the performance problems of a traditional TCP stack in a wireless environment, a host of alternative techniques were proposed [7]. However, one of the most prominent (and lesser-known) problems for music distribution to mobile device is that of an unpredicted link interruption in the midst of a long download.

A crucial aspect of TCP-oriented communications, in fact, is represented by the problems that are caused by long link outages and handoffs. In other words, a mobile user may enter an area of no signal coverage for a given period of time. This could happen for several different causes including the following:

- no signal coverage is provided for inaccessible areas, such as tunnels, subways, basements and cellars;
- a mobile user is passing through different areas which are under the coverage of different base stations, working with the same radio technology (horizontal handoffs);
- a mobile user is passing through different areas with signal coverage provided by different alternative wireless technologies (vertical handoffs);
- a user is in an area simultaneously covered by several different radio technologies; he/she may wish to select which radio access to use based on opportunistic motivation (e.g., cost, amount of available bandwidth, etc.).

Typically, if the time period of no coverage exceeds a given threshold (e.g., a few minutes) the network gateway located in between the wired and the wireless segments assumes that a link level interruption has occurred and, consequently, interrupts the data link connection over the wireless link.

We have two possible cases. In one case, the mobile client may have left the area of no signal coverage (say, a tunnel) in time to receive the information that the communication has been destroyed at the link level. In such a case, the TCP software running on the mobile client explicitly shuts down the TCP connection, as no more data will be delivered to the mobile device through that TCP connection.

In the second case, the mobile client may still be in the area of no signal coverage (a "long" tunnel) when the network gateway tries to inform it that the link level communication has been lost.

In this case, there is no way the client TCP software can learn from the gateway about the link level interruption. Thus, it is the operating system that intercepts the no coverage signal coming from the network interface adapter, and autonomously takes the decision to propagate an error that makes the TCP socket no longer available for data delivery at the client side.

A further negative result of this anomalous situation is that the TCP connection remains open forever at the server side, even though no more data may be transmitted over that TCP connection.

Both the above cases lead to the same final result: the music download is interrupted with no possibility to resume the data stream. This is a clear evidence that a session mechanism is needed to manage communication interruptions. With this in view, the aim of our session layer is no longer one of improving TCP performance to cope with the instability of the wireless environments, but one of ensuring a successful termination of the download activity even when the underlying TCP connections are disrupted or damaged due to the device mobility.

Stated simply, the full success of a song download activity may be really guaranteed only by a session mechanism able:

- to *freeze* the download activity in the presence of temporary communication outages, and
- to *resume* the data transfer as soon as signal coverage is available again.

To this aim, we have designed a session management layer (SML) which is able to manage possible interruptions at the lower levels of the communication architecture due to the phenomena of either horizontal or vertical handoffs.

Even though our session mechanism can work on any TCP stack, we have extended our protocol architecture to include, at the transport layer, one of the TCP protocols which was specifically devised for a wireless lossy environment [13,14]. Namely, we have embodied in our protocol architecture the TCP Westwood (TCPW) protocol [15].

The traditional congestion control algorithm of TCP Reno includes three phases: slow-start, congestion avoidance and Fast Retransmission–Fast Recovery that provide an enhanced recovery from sporadic errors. This mechanism is not well suited to wireless lossy links since sporadic losses due to radio channel problems are often misinterpreted as a symptom of congestion [6]. To overcome this problem, modifications to the standard TCP Reno such as Explicit Congestion Notification (ECN) or split-connections have been proposed [16,17].

Another drawback of TCP Reno is the sensitivity to an erroneous (low) initial setting of the slow-start threshold that can lead to premature congestion avoidance and thus under utilization of the channel. In this area again TCPW provides an elegant solution in that it continuously estimates the "eligible bandwidth" and resets the slow-start threshold accordingly [18–20].

More precisely, the TCPW source performs an end-to-end estimate of the *eligible bandwidth* by measuring (and low-pass filtering) the rate of returning ACKs. This estimate is then used to compute new values for both the congestion window and the slow-start threshold to be exercised during the specific TCP connection.

The window and threshold estimates are used at slow-start and after a congestion episode. While TCP Reno blindly cuts the window by half after loss, TCPW sets the congestion window to match the bandwidth available to the connection at this time. An important advantage of TCPW is the sender side only modification of the code (with respect to TCP Reno). This permits leaving the TCP client software unchanged, as illustrated in the protocol architecture reported in Fig. 2.

2.3. Implementing the session management layer

Our SML works at the client side as shown in Fig. 3, and summarized below. At the beginning

```
1   TCP_connection_ID:=connect_to_gateway(IP_AGw, port_number_AGw)    /* select the most
                                    appropriate TCP connection with the Application Gateway */
2   request_new_session(TCP_connection_ID)
3   sessionID:=receiveID()
4   while not(download_completed) do
5     if not(TCP_error or connection_timeout) then  /*read data from the current TCP
                            connection unless a TCP error is intercepted or a timeout occurs */
6       read()
7     else      /*the link level connection is interrupted*/
8       save_session_state()   /*save session state: session_ID and last_byte_received */
9       suspend_session()      /*destroy old TCP connection at the Client side*/
10      while not(session_resumed) do  /*try to resume session */
11        TCP_connection_ID:=connect_to_gateway(IP_AGw, port_number_AGw)    /*select the most
                                    appropriate TCP connection to the Application Gateway */
12        resume_session(TCP_connection_ID)   /*open a new TCP connection using the
                                  session_ID and the pointer to the last_byte_received */
13      od
14    fi
15  od
16  close_session()      /*  download completed at the client, termination of
                         session is communicated to the Application Gateway */
```

Fig. 3. SML: implementing the mobile client.

of each download session, a mechanism has been implemented that selects the most *appropriate* radio interface among all those available on the mobile device. Through this radio interface (and the corresponding IP address) a TCP connection is established with the Application Gateway (line 1, Fig. 3).

When the mobile device opens a download session with the Application Gateway through this TCP connection, the Application Gateway assigns a unique identifier to this new session and sends it to the client (lines 2–3). During the download process (line 4, global variable: download_completed), if the client is informed of an interruption occurred at the wireless link level (line 5, global variable: TCP_error), or if too long a period of time has passed since the instant when the last byte was received (line 5, global variable: connection_timeout), then the download session state is saved. In essence, a pointer to the last byte which was received prior to the interruption is saved at the mobile client along with the session identifier (lines 7–8), while the current TCP connection is closed since it has been (probably) disrupted due to a link outage (line 9).

After an interruption of a download session, an automatic mechanism is started that tries to establish a new TCP connection with the Application Gateway by concurrently exploiting all the radio interfaces which are available on the mobile device (line 11). With the first radio interface responding positively to this attempt, a new TCP connection is opened that permits to resume the interrupted download session with the Application Gateway. In essence, the session is restored as soon as the mobile client is able to send to the Application Gateway a message containing the session state that was previously saved (lines 10–13, global variable: session_resumed).

Eventually, when the client detects that the download is successfully completed, the session is terminated (line 16). At this point, the termination of the session is communicated to the Application Gateway.

For the sake of completeness, we have reported in Fig. 4, the software code implementing the function that selects the most *appropriate* radio interface available to identify the wireless client (*connect_to_gateway*). In particular, first the operating system running on the wireless device is interrogated to identify the IP addresses corresponding to all the available radio interfaces (line 1, Fig. 4). Then, all the IP addresses of the available radio interfaces are simultaneously exploited to try to connect to the Application Gateway (lines 2–7). The identifier of the first TCP connection which is successfully established is finally returned to our SML in order to start (resume) a corresponding download session (line 8). It is important to notice that in the current implementation of the *connect_to_gateway* function a criterion is used to select the most *appropriate* radio interface which is based on the speed with which each radio interface is able to respond to the connection request issued by our SML. Needless to say, future extensions of our SML may be devised that include different

```
1   local_IPs:=get_available_local_IP_interfaces() /* interrogate the OS of the device for the
                                                    IP address of each available local radio interface */
2   CONNECTION_ESTABLISHED:=-1
3   foreach IP in local_IPs do
4       CONNECTION_ESTABLISHED:= TCP_connect(IP, IP_Gw, port_number_Gw) /* try to establish
            a TCP connection from each available radio interface to the Application Gateway */
5   od
6   while not(selection_timeout or CONNECTION_ESTABLISHED > 0) do
7   od
8   return(CONNECTION_ESTABLISHED) /* return false if no TCP connection has been established
        within a predefined selection timeout. Return the TCP ID for the first radio interface
        that has successfully completed the connection with the Application Gateway */
```

Fig. 4. Selecting the most *appropriate* radio interface.

```
1   if (new_session_request) then
2       open_session()
3       sendID(sessionID)    /*send ID identifier to the Client, ready to transmit
                                   data from the first byte */
4   else
5       resume_session()     /*terminate old TCP connection, resume session ready to
                                   transmit data from last_byte_received */
6   fi
7   while not(download_completed or session_timeout) do    /*write data onto the current TCP
                             connection unless the download is completed or a timeout occurs */
8       write()
9   od
10  close_session()    /*session closed */
```

Fig. 5. SML: implementing the Application Gateway.

alternative criteria to be used to perform the choice of the most *appropriate* radio interface. Obviously, changing this selection criterion only entails the replacement of the software code implementing the *connect_to_gateway* function, while the general software organization of our SML may remain unaffected.

The SML implemented at the Application Gateway side works to keep the operations performed by the Application Gateway coordinated with the client. Fig. 5 shows that the Application Gateway exploits an internal mechanism to find out if a given request for a TCP connection coming from the wireless client represents a new session or an attempt to restore an already existing session.

If the Application Gateway receives a new session request (line 1 in Fig. 5, global variable: new_session_request), it assigns an identifier to this new session and, using the current TCP connection, transmits the session identifier to the mobile client along with the first data of the song which was requested (lines 2–3 and 8).

If the request is to restore an already existing session, the Application Gateway first performs a check to detect if an old TCP connection exists which is still open but inactive.

In this case, the Application Gateway closes the old TCP connection because it does not have an active counterpart at the client side. Then, the Application Gateway resumes the session and starts sending data to the client using the most recently established TCP connection.

The restoration of this upload activity is performed by exploiting the pointer to the last byte which was received by the client before the link disruption (lines 5 and 8).

When the client informs the Application Gateway that the download activity has been successfully completed, the current session is terminated at the Gateway side (lines 7 and 10, global variable: download_completed).

Additionally, the SML at the Gateway side timesout a session when a large amount of time has passed without any communication coming from the client (lines 7 and 10, global variable: session_timeout). Currently, the value of this timeout is set equal to 1 h.

It should be clearly understood that SML is able to restore download activities interrupted due to long link outages and handoffs, but it cannot recover from system failures occurring at the wireless client or at the Application Gateway. We consider this type of problem as an issue of future extensions of our system.

2.4. Exploiting the Web as music repository

In the previous Section 2.3 we have discussed the protocol architecture that supports all the communications between the mobile client and the IS-counterpart, that is, the Application Gateway. We are ready now to report on the protocol stack on which all the communications between the Web replica servers and the IS-counterpart

(i.e., the Download Manager) are based. In essence, it is the Download Manager that is the real agent responsible for the download process. It was devised to maximize service availability (i.e., the percentage of successfully served songs requested), as well as to maximize the service responsiveness (i.e., the time after which a requested song is successfully downloaded on the mobile client).

These goals have been fulfilled by exploiting the technique of *replicated Web servers*. According to this technology, a software redundancy is introduced on the Internet side: the songs which compose a music-on-demand service are replicated across a number of Web servers, geographically dispersed over the Internet.

In this context, a typical strategy to guarantee service responsiveness and availability consists of dynamically binding the client to the available server replica with the least congested connection [21].

An approach recently proposed to implement such a download strategy at the Internet side consists in using a software mechanism, called the Client-Centered Load Distribution (C^2LD) mechanism [11].

The main task of this mechanism is to minimize what we term the User Response Time (URT), i.e., the time elapsed between the generation of a request for the retrieval of a given Web resource and the delivery of that resource to the final user over the wireless link.

Simply stated, rather than binding a client to its *most convenient* replica server as proposed in [21], C^2LD captures each request for a music resource, and fragments that request into a number of sub-requests for separate fragments of that resource. Each sub-request is sent to a different available replica server, concurrently. For each sub-request, an internal timeout is set, and if this timeout expires before the requested fragment is received, the fragment is requested from another replica server. Finally, the replies received from the replica servers are reassembled to reconstruct the requested song which is then delivered to the final user.

One of the main advantages of the C^2LD mechanism is that it is able to adapt dynamically to the state changes in both the wired network (e.g., route congestion, link failures) and the replica servers (e.g., replica overload, unavailability). To this end, the C^2LD mechanism monitors periodically the available replica servers and selects, at run-time, those replicas to which the sub-requests can be sent following a dynamical Web server replica selection procedure. In essence, this dynamic selection procedure is based on the idea that those replicas are selected that can provide the requested song fragments within a time interval that permits minimizing the URT [22].

We have implemented the C^2LD mechanism on top of the HTTP 1.1 interface, as shown in Fig. 6. It is also worth mentioning that the use of the C^2LD mechanism does not force music providers to organize Mp3 repositories which are all perfect replicas of the same list of songs. A song, in fact, may be replicated within only some of the available server replicas of our system. The way in which our system is able to determine if a requested song is in existence in the system is managed by the Discovery system. Indeed, in order to obtain a song, a mobile user starts a request by providing the name of the song.

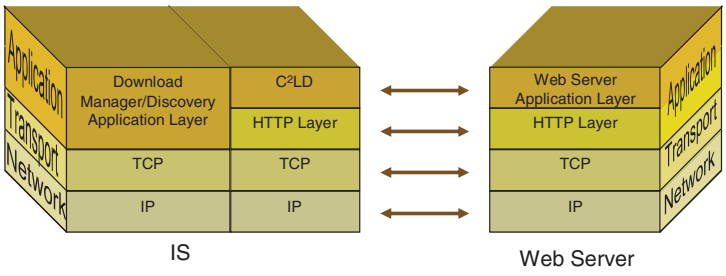


Fig. 6. The Download Manager protocol stack.

In essence, the main responsibility of the Discovery is that of performing a sort of *naming resolution* for musical songs which are requested by clients. In particular, it is in charge of:

- establishing a formal relationship between the requested songs and the correspondent Mp3 files stored in the system,
- identifying the exact Internet locations where Mp3 files are replicated.

It is a known fact that song titles and correspondent authors are useful for file indexing, but are not sufficient for accurate file identification. For example, the same song may be encoded at different sampling rates thus resulting in different Mp3 files, or identical Mp3 files may exist with different names or titles.

To overcome this kind of problem, the Discovery function is able to identify identical Mp3 copies of a given song by calculating a 32 bit-based identifier (called the *checksum*) which is computed on the file content, as discussed in [10].

Specifically, the Discovery system manages two different hash indexes: the former, needed to resolve user's requests, is created on the basis of the song title and author while the latter, is created on the basis of the checksum value mentioned above and is used to localize the requested files.

To minimize the traffic overhead, we took the decision to implement a distributed scheme where each music server wishing to upload its own songs on our system has to locally run a software application, termed the *Data Collector*, which provides the possibility to add or to delete the songs to be referenced by the Discovery system. In this case, it is the responsibility of the Data Collector to locally perform the checksum computation. The Data Collector is implemented as a Java applet to enhance software portability, and also meets standard security constraints, as it can only read from the local file system, but it cannot execute local write operations.

After having computed the checksum of all the files that a given music provider wishes to distribute, the Data Collector opens a TCP connection towards Discovery and uploads the computed checksums to it. It is worth noticing that our signature mechanism has been specifically designed to cope with the fragmentation/reassembly process provided by the C^2LD mechanism which needs to work with identical Mp3 copies of a given song.

Summarizing, with our signature mechanism, it is possible to build a replicated repository for each different version of a given song. Each replicated repository can be built with the contribution of each different owner of an identical copy of a given Mp3 song. The main advantages of this scheme are that:

- the construction of such a replicated repository may be conducted based on a distributed and independent process, where each owner may use the Data Collector and its signature mechanism to add/delete its copy of a given Mp3 song, and
- an expedited song distribution is guaranteed by the C^2LD mechanism which is able to exploit all the available identical copies of a given MP3 song.

3. An experimental assessment

In this section we present an experimental study we have conducted to assess the efficacy of our music on demand application. The main motivation behind our experimental assessment was that of investigating the quality and the performance of Web/mobile client music download sessions supported by our wireless application. During the period August 1–August 30 2003, we conducted 500 experiments consisting in the download of a set of different songs (MP3 files.) In the next two subsections we provide detailed information concerning, respectively, the experimental scenario where our trials were carried out (Section 3.1) and the empirical results we gathered with our on-the-field trials (Section 3.2).

3.1. Experimental scenario

With the aim of providing several different replicas for the songs to be distributed, we exploited

four different Web servers providing the same set of 20 different songs. The four different replica servers were respectively located in USA (Washington), Brazil (San Paolo), Japan (Tokyo), Italy (Cesena), as shown in Fig. 7. They were all based on the Linux Debian operating System, and exploited an Apache HTTP server.

The Intermediate System was running on a Pentium III machine (800 MHz, 128 MB RAM) equipped with the FreeBSD v. 4.7 operating system, and was located in the LAN (100 Mb/s) of the Network Research Laboratory in the Department of Computer Science at UCLA (USA).

Further, the wireless device, on which the client of our application was running, was installed on a HP iPAQ H5450 PDA (400 MHz, 64 MB RAM) equipped with the Windows CE 3.0 operating system, as represented in Fig. 8. In particular, this figure shows how the different wireless cards may be plugged into the HP iPAQ device used in our experiments.

We conducted our experiments using four different radio technologies installed on the wireless device. We exploited the four following radio interfaces: WLAN 802.11b (infrastructure-based), WLAN 802.11b (ad hoc), CDMA-2000 (1xRTT), cellular GPRS (the corresponding cards are depicted in Fig. 9). In particular, we used the wireless network adaptors with the following characteristics:

- Infrastructure-based WLAN: iPAQ embedded interface working at 11 Mb/s.
- Ad hoc WLAN: external interface (PCMCIA), Orinoco Gold produced by Lucent Technologies, working at 1 Mb/s in ad hoc mode.
- 3G CDMA2000 (1xRTT): external interface (PCMCIA), Sierra Wireless AirCard 555, nominal data rate of 144 Kb/s.
- GPRS: external interface (PCMCIA), Sierra Wireless AirCard 750 (triband 1900, 1800, 900 MHz), working at the data rate of 56Kb/s.

It is worth mentioning here that the two different 802.11b configurations (namely, infrastructure-based and ad hoc) corresponded to two different topologies:

- infrastructure-based WLAN (11 Mb/s): the wireless PDA is connected to the WLAN access point and the wired LAN segments of the Network Research Laboratory at UCLA;
- ad hoc WLAN (1 Mb/s): the wireless PDA is directly connected, through its radio interface, to the machine hosting the IS using an ad hoc configuration (1 Mb/s).

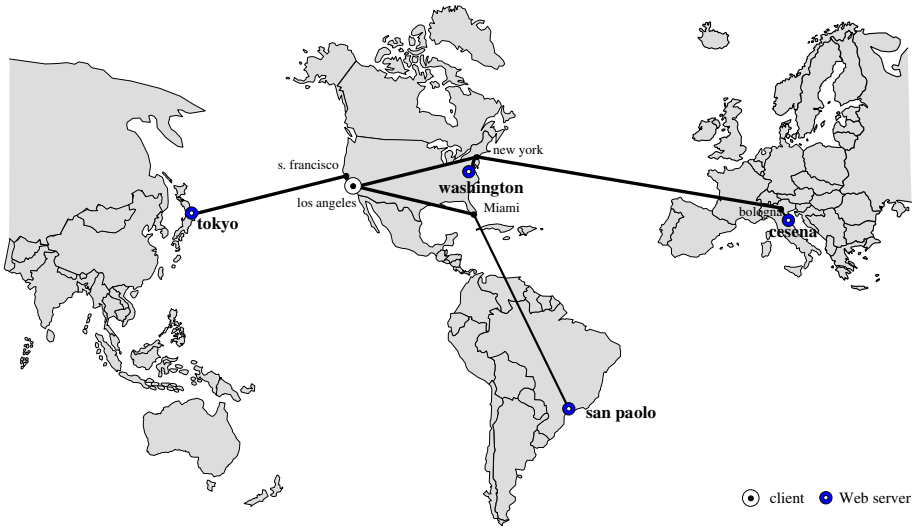


Fig. 7. Distributing the Web servers.

Fig. 8. The HP iPAQ H5450 with three wireless interfaces installed.

To provide the reader with an approximate knowledge of the transmission times experienced on the Internet links in question, we report the average Round Trip Time measurements, obtained with the *ping* routine, between the IS, located at UCLA, and the four different Web servers, i.e., Washington, San Paolo, Tokyo and Cesena. The RTTs were 79, 189, 132 and 157 ms, respectively.

The transmission time and packet loss over the wireless channels between the IS and the mobile client, were also measured. As expected the results varied with the radio technology as follows:

- infrastructure-based WLAN → 3–6 ms, 1–15%;
- ad hoc WLAN → 5–53 ms, 1–20%;
- CDMA2000—1xRTT (144 Kb/s) → 309–930 ms, 1–12%;
- GPRS (56 Kb/s) → 550–610 ms, 1–18%.

The packet loss is due in part to buffer overflow and in part to wireless channel errors.

3.2. Empirical results

This section reports on a large set of results obtained within the experimental scenario we have described above. In particular, we will present the measurements of the download times we obtained for the distribution of MP3 songs from the Web to the mobile clients for all the three following download situations:

- The entire song download process is conducted by using the same wireless network technology, without any kind of interruption.
- The song download process is split into a few different subsequent phases (typically, 2 or 3). The split is due to horizontal handoffs that caused a download interruption. In essence, after each handoff, the download process is resumed by using the same wireless technology that was operational before the interruption caused by the handoff.
- The song download process is split into two different phases which are separated each from other due to vertical handoffs. In this case, after the handoff the music download session is restored using a wireless technology which is different from that supporting the download activity prior to the occurrence of the handoff. E.g., we have a transition from WLAN to CDMA2000 (or 1xRTT).

Fig. 9. Some of the wireless cards exploited in the experiments.

Figs. 10 and 11 show the results where the same kind of radio access technology was used. In both these figures the song download times (expressed in seconds over the *Y*-axis) are recorded for one of the following situations (specified over the *X*-axis):

- The user who downloads songs is stationary and does not suffer from any download interruption (Still User, 0 interr.).
- The user who downloads songs is in motion and does not experience any download interruption (Motion, 0 interr.).
- The user downloading songs is in motion and experiences exactly one download interruption (Motion, 1 interr.).
- The user downloading songs is in motion and experiences exactly two download interruptions (Motion, 2 interr.).

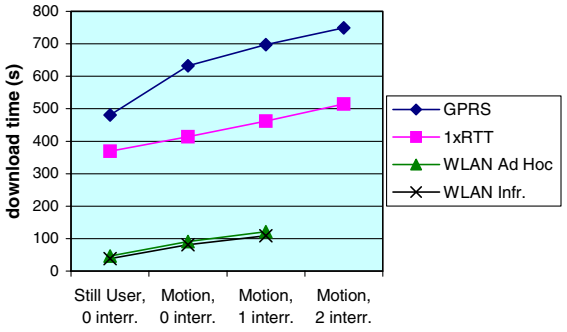


Fig. 10. Download time depending on number of interruptions, when using TCP Reno.

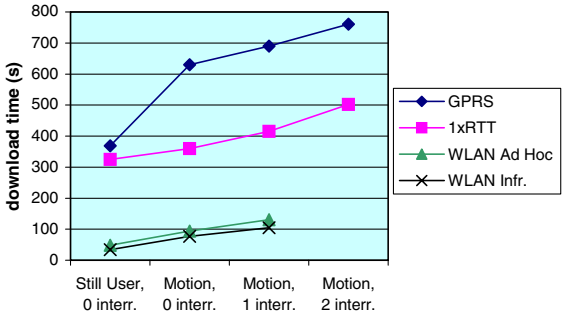


Fig. 11. Download time depending on number of interruptions, when using TCP Westwood.

In Fig. 10 the transport layer was TCP Reno. In Fig. 11 it was TCP Westwood.

The first general (and anticipated) conclusion is that the larger the number of interruptions, the larger the download time. Based on this consideration it is clear that a user in motion typically experiences larger download times w.r.t. the case when the user is static.

A second consideration touches upon the fact that smaller download time may be obtained with radio access technologies able to guarantee a larger bandwidth. This is an expected result which explains the smaller download times obtained with the WiFi technologies w.r.t. the cellular infrastructure (GPRS-1xRTT).

As a third comment, we wish to observe that WiFi technologies are typically more robust in the sense that, within this technology, more than one interruption rarely occurs as evidenced by the plots in Figs. 10 and 11.

The final, and perhaps most important, consideration regards the fact that TCP Westwood (TCPW) consistently outperforms TCP Reno when the radio technology is cellular. However, TCPW does not clearly outperform the traditional TCP in the experiments conducted with the WiFi technology (Fig. 12).

In the latter scenarios the download times obtained with TCP Reno are often smaller than those obtained with TCPW.

The superiority of TCPW in cellular environments is simply explained by the fact that ARQ

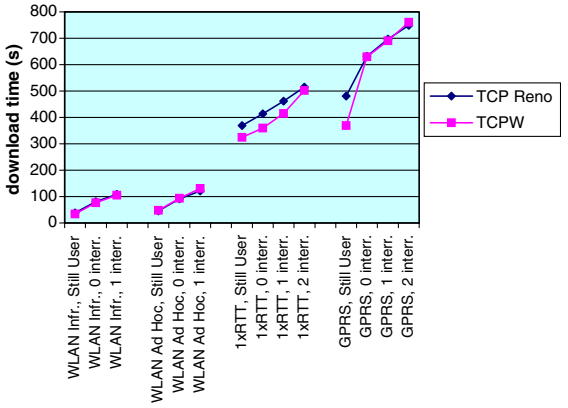


Fig. 12. Comparison between TCP Reno and TCP Westwood.

mechanisms in cellular networks retransmit only a very limited number of times (once in the case of 1xRTT).

Moreover the packet latency is large. All these factors weigh in favor of TCPW.

The reason for the poor behavior of TCP Westwood in 802.11 can be traced to the fact that, in absence of random channel errors, TCP Reno outperforms TCPW when the bottleneck buffer (in our case the buffer on the 802.11 PCMCIA card) is larger than the "pipe size". The pipe size is defined as the number of packets outstanding on the path. It is proportional to the {delay × bandwidth} product.

The experimental scenario based on the WiFi technology, indeed, has quite a low {delay × bandwidth} product, due to the fact that the propagation delay from IS to wireless client is very low.

Thus, the PCMCIA buffer is always much larger than the {delay × bandwidth} product. Moreover, the random loss rate is extremely low since 802.11 retransmits a packet in error up to 7 times! As a consequence, TCP Reno prevails as expected.

To confirm this hypothesis we performed an additional experiment where the IS was placed far from the mobile client, in Cesena. The mobile client downloaded songs while moving around the UCLA Campus. The {delay × bandwidth} product is now much larger than before. The measurements taken with this additional experiment and reported in Fig. 13 confirm our hypothesis: TCPW exhibits lower delay than TCP Reno.

As to the experiments conducted to test our wireless application in the case of vertical handoffs, we may report the fact that in all those experiments (100%) our mechanism was able to guarantee the completion of the download session even in the presence of several radio technology switchovers (and corresponding vertical handoffs).

As significant examples of our experimental trials, we report below in Table 1 the traces of six different download sessions conducted switching

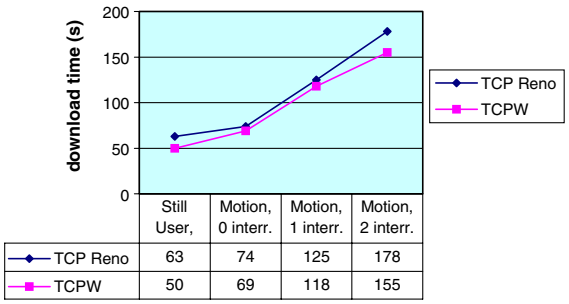


Fig. 13. Comparison between TCP Reno and TCPW within the WiFi scenario.

Table 1
Vertical handoffs

Experiment #1	ADHOC—51 s, 24% Interruption—2 s GPRS—236 s, 30% Interruption—1 s WLAN—36 s, 46% Download time = 326 s
Experiment #2	WLAN—31 s, 37% Interruption—1 s ADHOC—81 s, 33% Interruption—1 s ADHOC—27 s, 0% Interruption—4 s 1xRTT—110 s, 30% Download time = 255 s
Experiment #3	1xRTT—144 s, 34% Interruption—18 s WLAN—56 s, 41% Interruption—18 s ADHOC—26 s, 25% Download time = 262 s
Experiment #4	ADHOC—51 s, 24% Interruption—2 s GPRS—236 s, 30% Interruption—1 s WLAN—36 s, 46% Download time = 326 s
Experiment #5	WLAN—38 s, 52% Interruption—1 s ADHOC—78 s, 35% Interruption—2 s GPRS—99 s, 13% Download time = 218 s
Experiment #6	GPRS—38 s, 3% Interruption—1 s WLAN—72 s, 44% Interruption—2 s GPRS—104 s, 0% Interruption—2 s ADHOC 38 s, 53% Download time = 257 s

through different radio access technologies (e.g., from WiFi to GPRS/1xRTT). Out of those six traces, three are concerned with the experiments conducted switching from WiFi (infrastructure-based and ad hoc) to GPRS, while the other three were conducted switching from WiFi to 1xRTT.

In particular, for each given trace the following information is reported in Table 1:

- Radio access technology used to download the song before interruption; download times (expressed in seconds) and correspondent amount of downloaded song (percentage) before interruption.
- Duration of the interruption (in seconds).
- Total download time (in seconds).

Based on the results we report in Table 1, we wish to point out that the session layer mechanism we have adopted ensures that song download may be carried out even if several link interruptions occur due to vertical handoffs. In simple words, our wireless application is able to maintain the download session active even in the case when long signal interruptions are caused by vertical handoffs. Further, our wireless application is able to: (i) freeze the status of the download session even when a radio access technology selected to bypass an interruption turns out to be not able to download data, and (ii) find out and switch to an alternative available technology to resume the data download activity (see traces of Experiments #2 and #6).

It is very important to mention that without the support of our session layer mechanism in similar cases the download activity should be restarted from scratch with each new interruption.

To conclude our experimental study, we have carried out a deeper analysis on how our mechanism behaves during the execution of the experiments reported in Table 1. In particular, we have examined Experiment #1 as a significant example of the behavior of our mechanism.

Specifically, this experiment was conducted while moving around the UCLA campus as described in Fig. 14. It may be noticed that all the

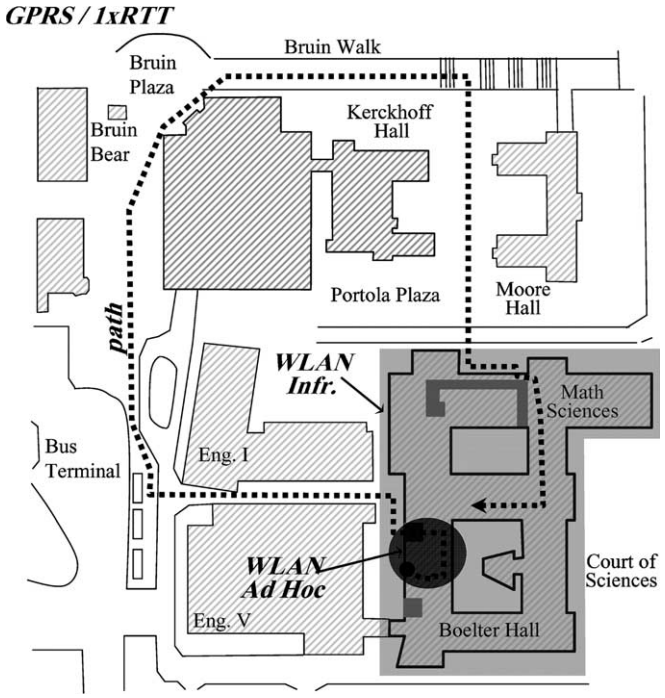


Fig. 14. Experiment #1: moving throughout the UCLA campus.

buildings and living areas represented in the figure were covered by the GPRS/1xRTT wireless access technologies.

Only the Bolter Hall building was covered by an infrastructure-based WLAN, as well as by an ad hoc WLAN set up inside the area of the Networking Research Laboratory (dark circle in the figure). Experiment #1 was conducted moving along the path pointed out by the arrow in Fig. 14.

During this experiment, the time needed to download the requested music was distributed over the three different wireless technologies (based on their availability) with the percentages shown in Fig. 15.

Further, Fig. 16 shows the evolution of the music download process over time expressed as the number of bytes that were downloaded with each different network technology.

Fig. 17, finally, compares the performance (download times) of our mechanism which engages simultaneously different wireless technologies (on the basis of their availability) with alternative approaches where only one access technology is exploited for downloading the entire music resource of Experiment #1.

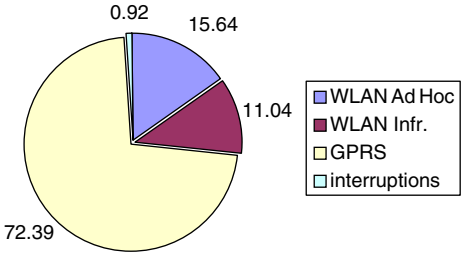


Fig. 15. Experiment #1: percentage of download time for different wireless technologies.

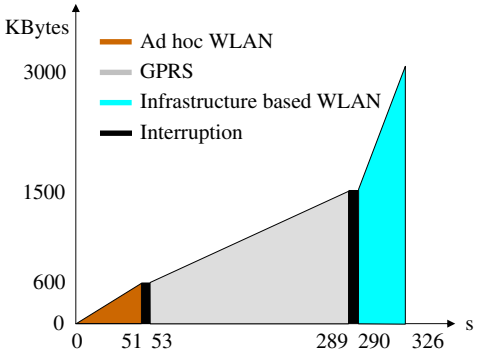


Fig. 16. Experiment #1: integral of downloaded bytes over time with different wireless technologies.

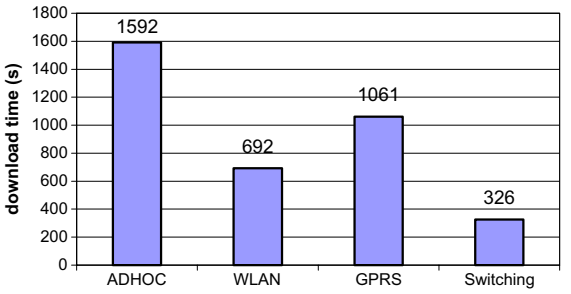


Fig. 17. Comparison of our session-based mechanism with different single-technology approaches.

Obviously, this comparison has been conducted under the hypothesis that each single access technology is available in the experiment according to the percentage of active download time specified in Fig. 15.

It is easy to recognize that, under these hypotheses, our session-based multi-technology mechanism is able to guarantee the completion of the download process with smaller time w.r.t. an approach where alternative wireless technologies are used separately. Based on the above considerations, we may conclude that our developed mechanism is very effective for the download of digital music over wireless devices, and hence may be considered as *ultra reliable*.

4. Related work

The aim of this section is to briefly present some related work and to compare it with the most relevant design choices we have taken to implement our system.

4.1. Wireless multimedia delivery service

An issue of paramount importance for the development of our application is that of multimedia (music) distribution over wireless networks [23]. To this aim, there has been plenty of attention

about the possibility of an effective, secure and reliable access of multimedia (music) information from wireless terminals. For example, a well-known approach is represented by the use of the i-Mode technology.

In this context, a multimedia distribution service is being developed which distributes music to user handsets that have player functionalities [22]. Two kinds of service are offered, the former amounts to the provision of promotional music samples, the latter provides download of entire songs for immediate purchase. The first limitation of this approach is that it does not provide the support for maintaining a download session when the user moves between different devices and access networks. Another limitation of this approach is that it rests upon a specific technology (i-Mode).

In contrast, our approach permits the application to adapt to changes in the communication environment, thus enabling the user to experience the data (music) flow as transparently as possible even in the presence of environmental (network) changes.

Another prominent issue in the design of wireless distribution services is concerned with the fact that typically multimedia (music) information is conceived as a flow of continuous data [24,25]. Following this approach, the main problem turns out to be how to react to latency and packet loss problems caused by handovers. Typical solutions to ameliorate problems due to latency jitter and packet loss amount to *buffering* or *smoothing* techniques [26].

Although these techniques are very effective in reacting to latency and packet loss, they usually introduce buffering delays that try to surmount the interruption of the data flow, but increase the overall end-to-end delay.

It is well-known, however, that a large increase of the end-to-end delay may have a negative effect on human perception, and hence should be avoided. In this sense, attention should be paid to the fact that "low-quality" music distribution services may induce a significant drop in the use of modern wireless technologies for music distribution [3].

In contrast, our application was designed to provide the user with a wireless music distribution service that, in some sense, resembles the iTune system [3]. According to this approach, the main goal is that of ensuring the delivery of a high-quality copy of an entire song to the final user. The challenge, here, is to develop a distribution system that is able to seamlessly extend the reach of wirelined Internet-based distribution services to mobile users. In essence, for the type of applications we have developed, less attention has to be paid to those latency-hiding problems which are typically faced by streaming applications. Instead, a seamless delivery service has to be guaranteed where no byte is lost while moving between different network technologies.

4.2. Always best connected

Being always best connected (ABC) means that a person is not only always connected, but connected through the best available access technology. Defining what the best connection is may depend on a number of different factors including, for example, user preferences, device size and capabilities, available network resources and coverage, application requirements, service policies [27,28]. Perhaps, the simplest form of an ABC service amounts to a situation where the user is provided with capabilities to access services over different types of network technologies, without any automatic mobility support [29–32]. More complex ABC services are currently being devised where the user may transfer the application session from one ABC terminal to another.

Following a full ABC-type approach, we have developed our system with the aim of guaranteeing a sophisticated ABC service where the wireless terminal is allowed to move seamlessly between different access network technologies, while maintaining a (session-level) connection to the music application servers, without losing data or needing to manually restart the application in the case of handoffs.

We have developed our service based on a traditional IPv4 stack without Mobile IP [33]. According to this solution, with each event of mobility, the user must acquire a new private or

public IP address, through the DHCP, from the local network gateway. However, adopting a simple IP service would mean that ongoing transport level sessions are lost when the client moves. To overcome this problem we have developed an intelligent interface selection operating on the client device (Section 2.4) which allows employing at each interface a different IPv4 address depending on the network to which the device is connected. As described in the above mentioned Section 2.4, the state information needed for the dynamic switching of interfaces (IPv4 addresses) is managed through a session layer mechanism built on top of TCP.

A large number of alternative solutions exist where the goal of preserving user sessions, when the user is roaming among heterogeneous networks (or different providers), is achieved through the use of mobile-IP based architectures [34,35]. The use of Mobile IP here allows delivery of packets to a fixed address called a Home Address (HoA). To this aim Mobile IP employs two network elements: a Home Agent (HA), in the home network, and a Foreign Agent (FA), in the foreign network. When moving to a foreign network, a mobile node discovers a local FA, registers the address of FA as a care-of-address (CoA) with its HA, and creates the binding between the HoA and the CoA. The HA is in charge of intercepting packets directed to the mobile node, encapsulating them and tunneling them to the FA. The FA decapsulates the original packets and delivers them to the mobile node. Following this approach, all the transport layer sessions are preserved as the mobile node maintains its HoA. The motivations behind our choice of using IPv4 without Mobile IP are the following:

1. A IPv4 service without Mobile IP does not need a specialized client software for service access, nor requires modifications at TCP-IP stack in the network.
2. A IPv4 service without Mobile IP does not need to perform tunneling activities for the delivery of packets to the client (typically, tunneling activities may be the cause of service performance degradation).
3. A Mobile IP based service needs an overlapped coverage between alternative wireless technologies to avoid service disruption and packet loss during service handoffs. Instead, our choice to resort to a session level mechanism ensures that no packet is lost during service handoffs even in the absence of overlapping coverage.

Further, to conclude this comparison, it is worth recalling that a problem of very practical relevance for wireless music distribution is that of unexpected link interruptions in the midst of a long song download activity. The aim of our session level is to ensure that the download activity is not destroyed nor a single byte lost due to very long link outages or handoffs. In essence, our session has been developed to guarantee a successful termination of the download process even when the underlying TCP connection is interrupted because of device mobility. Obviously the disadvantage of our approach is that our session level mechanism should be embedded into each new multimedia distribution application to be developed. This problem could be surmounted by devising a middleware-based approach including our session layer mechanism [36].

5. Conclusions

We have developed an Internet ‘‘wireless’’ application that can efficiently distribute music files from Web Servers to mobile clients across different types of wireless media.

The main characteristics of a ‘‘wireless’’ application are: (i) ability to provide support to mobile clients connected to the Web through different radio access technologies (e.g., WiFi, GPRS, 1xRTT), and (ii) ability to provide song download continuity even in the presence of horizontal and vertical handoffs. The wireless application we designed does satisfy the above requirements.

Moreover, it incorporates sophisticated software mechanisms that always guarantee to the mobile user the best download alternative. Music download measurements confirm the efficacy of our approach.

Future extensions of this work include the design of QoS negotiation strategies for music distribution, and the implementation of Authentication, Authorization, and Accounting (AAA) mechanisms [37].

Acknowledgements

This work was supported in part by the UC Discovery Grant Micro #03-032/ST-Microelectronics, and in part by the Italian MIUR under the Interlink Project Grant. Marco Roccetti and Paola Salomoni wish to extend their gratitude to Diego Gardini, Matilde Gardini and Edvige Manghi for their emotional support during their stay at UCLA.

References

[1] J. Mc Hugh, Why Wi-Fi is a good business? Wired (September) (2003) 25–26.
[2] Napster Home Page. Available from: http://www.napster.com/.
[3] Apple iTunes. Available from: http://www.apple.com/itunes/.
[4] J. De Vriendt, P. Laine, C. Lerouge, X. Xu, Mobile network evolution: a revolution on the move, IEEE Communications Magazine 40 (4) (2002) 104–111.
[5] The 3rd Generation Mobile Wireless, CDMA Development Group, 2001. Available from: http://www.3gnewsroom.com/html/whitepapers/year_2001.shtml.
[6] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, R.H. Katz, A comparison of mechanisms for improving TCP performance over wireless links, IEEE/ACM Transactions on Networking 5 (6) (1997) 756–769.
[7] K. Wang, S.K. Tripathi, Mobile-end transport protocols: an alternative to TCP/IP over wireless links, in: Proceedings of the IEEE Infocom'98, San Francisco, CA, USA, 1998, pp. 1046–1053.
[8] C. Hesselman, H. Eertink, A. Peddemors, Multimedia QoS adaptation for inter-tech roaming, in: Proceedings of the Sixth IEEE Symposium on Computers and Communications (ISCC'01), Hammamet, Tunisia, July 2001, pp. 554–561.
[9] W. Zhuang, Y. Gan, K. Loh, K. Chua, Policy-based QoS management architecture in an integrated UMTS and WLAN environment, IEEE Communications Magazine 41 (11) (2003) 118–125.
[10] M. Roccetti, P. Salomoni, V. Ghini, S. Ferretti, Bringing the wireless Internet to UMTS devices: a case study with music distribution, Multimedia Tools Appl. Int. J. 25 (2) (2005) 217–251.
[11] V. Ghini, F. Panzieri, M. Roccetti, Client-centered load distribution: a mechanism for constructing responsive Web Services, in: Proceedings of the 34th International Conference on System Sciences, Maui, Hawaii, USA, January 2001.
[12] M. Roccetti, P. Salomoni, V. Ghini, S. Ferretti, S. Cacciaguerra, Delivering music over the wireless Internet: from song distribution to interactive karaoke on UMTS devices, in: B. Furht, M. Ilyas (Eds.), Wireless Internet Handbook: Technologies, Standards and Applications, CRC Press, Boca Raton, USA, 2003, pp. 537–565.
[13] V. Jacobson, Congestion avoidance and control, ACM Computer Communications Review 4 (18) (1988) 314–329.
[14] V. Jacobson, Berkeley TCP evolution from 4.3-Tahoe to 4.3 Reno, in: Proceedings of the 18th Internet Engineering Task Force, University of British Colombia, Vancouver, BC, 1990.
[15] S. Mascolo, C. Casetti, M. Gerla, M. Sanadidi, R. Wang, TCP Westwood: bandwidth estimation for enhanced transport over wireless links, in: Proceedings of the ACM Mobicom 2001, Rome, Italy, July 2001, pp. 287–297.
[16] J.C. Hoe, Improving the start-up behavior of a congestion control scheme for TCP, in: Proceedings of the ACM Sigcomm'96, Antibes, France, August 1996, pp. 270–280.
[17] P. Bhagwat, P. Bhattacharya, A. Krishna, K. Tripathi, Using channel state dependent packet scheduling to improve TCP throughput over wireless LANs, Wireless Networks 3 (1) (1997) 91–102.
[18] R. Wang, M. Valla, M.Y. Sanadidi, M. Gerla, Using adaptive rate estimation to provide enhanced and robust transport over heterogeneous networks, in: Proceedings of the 10th IEEE International Conference on Network Protocols, Paris, France, November 2002, pp. 12–15.
[19] M. Gerla, S.S. Lee, G. Pau, TCP Westwood simulation studies in multiple-path cases, in: Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), San Diego, CA, USA, July 2002.
[20] S. Mascolo, A. Grieco, G. Pau, M. Gerla, C. Casetti, End-to-end bandwidth estimation in TCP to improve wireless link utilization, in: European Wireless Conference, Florence, Italy, 2002.
[21] D. Ingham, S.K. Shrivastava, F. Panzieri, Constructing dependable Web services, IEEE Internet Computing 4 (1) (2000) 25–33.
[22] A. Takeshita, Toward mobile multimedia in next generation networks, in: Proceedings of the IEEE Applications and the Internet Workshop, January 2001, pp. 194–198.
[23] G. Prem Premkumar, Alternate distribution strategies for digital music, Communications of the ACM 46 (9) (2003) 89–95.
[24] F. Fitzek, M. Reisslein, A prefetching protocol for continuous media streaming in wireless environments, IEEE Journal on Selected Areas in Communications 19 (6) (2001) 2015–2028.

[25] S. Valaee, J. Gregoire, Resource allocation for video streaming in wireless environment, in: Proceedings of the IEEE International Symposium on Wireless Personal Multimedia Communications (WPMC 02), Honolulu, Hawaii, USA, October 2002.
[26] Q. Zhang, W. Zhu, Y.Q. Zhang, Network-adaptive scalable video streaming over 3G wireless networks, in: Proceedings of the IEEE International Conference on Image Processing (ICIP'01), October 2001.
[27] E. Gustafsson, A. Jonsson, Always best connected, IEEE Wireless Communications 10 (1) (2003) 49–55.
[28] G. Fodor, A. Eriksson, A. Tuoriniemi, Providing quality of service in always best connected networks, IEEE Communications Magazine 41 (7) (2003) 154–163.
[29] M. Frodigh, S. Parkvall, C. Roobol, P. Johansson, P. Larsson, Future generation wireless networks, IEEE Personal Communications 8 (5) (2001) 10–17.
[30] K. Ahmavaara, H. Haverinen, R. Pichna, Interworking architecture between 3GPP and WLAN systems, IEEE Communications Magazine 41 (11) (2003) 74–81.
[31] M.M. Buddhikot, G. Chandranmenon, S. Han, Y. Lee, S. Miller, L. Salgarelli, Design and implementation of a WLAN/CDMA2000 interworking architecture, IEEE Communications Magazine 41 (11) (2003) 90–99.
[32] J.W. Floroiu, R. Ruppelt, D. Sisalem, J. Voglimacci Stephanopoli, Seamless handover in terrestrial radio access networks: a case study, IEEE Communications Magazine 41 (11) (2003) 110–116.
[33] C. Perkins, IETF-RFC 3344, IP Mobility Support for IPv4, August 2002.
[34] Q. Zhang, C. Guo, Z. Guo, W. Zhu, Efficient mobility management for vertical handoff between WWAN and WLAN, IEEE Communications Magazine 41 (11) (2003) 102–108.
[35] D. Johnson, C. Perkins, J. Arkko, Mobility Support in IPv6 (draft), August 2003.
[36] P.A. Bernstein, Middleware: a model for distributed services, Communications of the ACM 39 (2) (1996) 86–97.
[37] G.M. Køien, T. Haslestad, Security aspects of 3G-WLAN interworking, IEEE Communications Magazine 41 (11) (2003) 82–88.

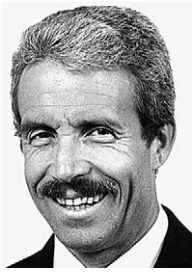
Vittorio Ghini is a Research Associate at the Department of Computer Science of the University of Bologna. He received the Laurea degree (with honors) and the Ph.D. in Computer Science from the University of Bologna respectively in 1997 and in 2002. His current research interests include QoS management at the middleware level, Web performance, network emulation and architectural design of Internet-based multimedia system.

Giovanni Pau is an assistant researcher at the Computer Science Department of the University of California, Los Angeles. He obtained the Laurea Doctorate in Computer Science and the Ph.D. in Computer Engineering from the University of Bologna (Italy). He has served on the Program Committee of several International Conferences and Workshops, and recently was a Co-Chair of the 1st International IEEE Workshop on Networking Issues in Multimedia Entertainment. His research interests include Wireless Multimedia, Peer to Peer and Multimedia Entertainment.

Marco Roccetti is a professor at the Department of Computer Science of the University of Bologna. He received his Laurea Degree in Electronics Engineering from the University of Bologna. For the past 15 years he has held different research and management positions at the University of Bologna where he is currently the scientific director of the Master Program in Communication and Information Technology. He is a member of a number of international conference program committees and he was the program chair for the SCS International Conference on Simulation and Multimedia in Engineering Education (2002–2003). He is also co-chair for the 1st IEEE Workshop on Networking Issues in Multimedia Entertainment (2004) and is serving as co-guest Editor for a Special Issue on Networked Computer Entertainment of the IEEE Communications Magazine. He also serves as Editor-in-Chief of the E-Letter publication of the Multimedia Technical Committee of the IEEE Communications Society. He is active in several Italian and European projects and has authored and co-authored more than 90 technical refereed papers published in the proceedings of international conferences and journals. His research interests include digital audio and video for multimedia communications, wireless multimedia and network-centric computer-based entertainment.

Paola Salomoni is an Associate Professor of Computer Science at the Department of Computer Science of the University of Bologna. In October 1992 she received the Italian Laurea degree (with honors) in Computer Science from the University of Bologna. From 1995 to 2001 she was a Research Associate in the Department of Computer Science of the University of Bologna. She is active in several Italian and European projects and has served on the program

committees of several international conferences and workshops. Her research interests include distributed multimedia systems and services, tools and techniques for E-learning environments, and computer entertainment.



Mario Gerla received a graduate degree in engineering from the Politecnico di Milano in 1966, and the M.S. and Ph.D. degrees in engineering from UCLA in 1970 and 1973, respectively. After working for Network Analysis Corporation from 1973 to 1976, he joined the Faculty of the Computer Science Department at UCLA where he is now a Professor. His research interests cover the performance evaluation, design and control of distributed computer communication systems, high speed computer networks, wireless LANs and, ad hoc wireless networks. He has worked on the design, implementation and testing of various wireless ad hoc network protocols (channel access, clustering, routing and transport) within the DARPA WAMIS and GloMo projects. Currently he is leading the ONR MINUTEMAN project at UCLA, the main focus of which is the design of a robust, scalable wireless ad hoc network architecture for unmanned intelligent agents in defense and homeland security scenarios. He is also conducting research on QoS routing, multicasting protocols and TCP transport for the Next Generation Internet (see www.cs.ucla.edu/NRL for recent publications). Professor Gerla is an IEEE Fellow.