THE STRUCTURING OF A WIRELESS INTERNET APPLICATION FOR A MUSIC-ON-DEMAND SERVICE ON UMTS DEVICES

Marco Roccetti, Vittorio Ghini, Paola Salomoni
Alessandro Gambetti, Davide Melandri, Mirko Piaggesi, Daniela Salsi

Dipartimento di Scienze dell'Informazione
Università di Bologna
Mura Anteo Zamboni 7, I-40127 Bologna, Italy
Phone: + 39 051 2094503
Fax: + 39 051 2094510

E-mail: roccetti@cs.unibo.it

ABSTRACT

Developing enhanced wireless Internet applications is becoming one of the upcoming challenges for mobile radio networks operators. In this paper we introduce and discuss the general software architecture of a wireless Internet-based application we have designed and implemented to support the distribution of Mp3-based musical songs to UMTS devices. We have examined the effects that Internet traffic has on the performance of wireless UMTS networks, due to the distribution of Mp3 files by means of our wireless application. The download time measurements we have experimentally obtained have shown that an appropriate structuring of the Internet-based wireless application may be very helpful to surmount the problems caused by the Internet standard protocols which were not especially designed for wireless environments.

Categories and Subject Descriptors

D.4.7 [**Operating Systems**]: Organization and Design – *Distributed systems, Real-time systems, Interactive systems.*

General Terms

Measurement, Performance, Design.

Keywords

Digital media on UMTS devices, Music on demand, Wireless multimedia applications, UMTS, Performance evaluation

1. INTRODUCTION

Two important tendencies have emerged, in recent years, as the major driving forces in the area of telecommunication networks. On one hand, many Internet based applications, such as browsing the Web and Electronic Mail, have determined a volume of packet data traffic that has increased at extreme rates. On the other hand, mobile radio networks have experienced an exponential trend both in the increase of network traffic and in growing importance to users [1, 2]. Due to the combination of the success of wireless networks and of the growth of the Internet, it is expected that there will be soon a greater demand for a mobile access to advanced Internet-based wireless services. Hence, it is easy to envisage that the success of these new communication technologies will depend on how much efficient the wireless radio access to those Internet-based enhanced application services will be.

It is well known that the most intensively deployed standard for second-generation mobile radio networks is the *Global System for Mobile Communications* (GSM). With the introduction of the *General Packet Radio Service* (GPRS) for mobile networks operating under GSM, packet data connections have been allowed with variable bit rates up to several tens of Kb/s. Instead, with third generation mobile systems using the *Universal Mobile Telecommunications System* (UMTS), a variable bit rate will be available of up to a few Mb/s, and this will permit connections supporting a wide range of existing and new Internet-based applications, including digital voice, video and data, as well as enhanced multimedia services [3]. Specifically, the UMTS forum specifies and describes four different traffic classes of possible services, whose quality is as follows:

- *Conversational class*, for supporting traditional real-time services, like real time telephony and video-telephony, with stringent and low delays.
- *Streaming class*, for supporting real-time traffic flows that need to preserve time relation between different entities of the stream, e.g. video on demand.
- *Interactive class*, for providing support to interactive best effort traffic such as traditional Internet applications (e.g., Web browsing),
- *Background class*, for supporting non-interactive best-effort traffic, such as, e.g., simple download of electronic mails or files.

An important advantage of these new mobile network technologies (including both GPRS and UMTS) is that packets originating from GPRS/UMTS mobile devices can be directly transmitted to data networks based on the Internet Protocol (IP networks), and viceversa. This is due to the fact that GPRS/UMTS networks support special *border nodes* (termed GSN) that use IP as the backbone protocol for transfer and routing of protocol data units.

However, in this communication scenario several important problems arise. The first problem is to decide if advanced TCP/IP based applications will be able to behave well over mobile radio communications protocols, such as UMTS for example. With regard to this fact, it is important to notice that the Internet protocols TCP and IP have not been especially designed for wireless communications. Briefly, the standard Transmission Control Protocol (TCP) provides a sliding window based ARQ (Automatic Repeat Request) mechanism that incorporates an adaptive timeout strategy for guaranteeing end-to-end reliable data transmissions between communicating peer nodes over wired connections. Since the ARQ mechanism of TCP essentially uses a "stop and resend" control mechanism for ensuring connection reliability, under question, here, is whether this TCP retransmission mechanism may trigger a TCP retransmission at the same time when the radio link level control mechanism is already retransmitting the same data. Secondly, a more significant problem of mobile wireless is that of temporary link outages. If a user, in fact, enter an area of no signal coverage, there is no way that the standard TCP protocol may be informed of this link-level outage [4]. After having considered all the above challenges, a third and final problem is strictly related to the internal architecture of those advanced Internet-based applications that should be accessed through radio interfaces. Those applications, in fact, must exhibit a high rate of robustness and availability, since the mobile access to those applications should not be influenced by possible problems occurring at the Internet side.

In this context, we have designed, developed and experimentally evaluated an Internet-based wireless application that implements a *mobile music-on-demand service* to be enjoyed on UMTS devices. The developed application permits to mobile users (i.e. clients) to download and to listen to Mp3 files [5] through UMTS devices. Specifically, our wireless application exploits the *background* traffic class of UMTS to provide its users with: i) a simple and rapid Internet-based mobile access to a music-on-demand download service, ii) a robust and widely available music-on-demand distribution system, based on the technique of replicated Web servers, iii) the possibility of interactively customize the use of the system, and finally, iv) the possibility of porting and using this application service over a wide range of UMTS devices and operating systems.

From a user's standpoint, it is worth noticing that different types of clients may exploit our developed system. In particular our wireless application may be exploited by:

- *Music Listeners*, they are single clients, equipped with a mobile UMTS device and connected to their UMTS cell, who may want to search for their favorite songs over the Internet, download them onto their UMTS devices, and finally playout them at their earliest convenience.
- *Music Producers*, they are single clients, who may wish to exploit the system in order to distribute their own music songs to be listened to on UMTS devices. At the current state of the art of our system, this kind of users needs a regular wireline Internet connection in order to upload to the system their Mp3 music resources.
- *Musical Service Providers*, they may exploit the system to organise, build and maintain structured repositories of Mp3 resources over the Internet for use from UMTS devices.

The important experiences of *Peer-to-Peer* (P2P) computing-based software systems, which have come to maturity with systems like Napster [6], Gnutella [7] and Freenet [8], have inspired our work, but our system is essentially new, in the sense that it allows a reliable and distributed song sharing service over mobile UMTS devices. In particular, in order to ensure both the availability and the responsiveness of our music-on-demand service, we have structured our system according to the special technology of the "replicated Web servers" [9, 10]. In essence, according to this technology, a software redundancy is introduced at the Internet side, namely by replicating (some of the) the music songs composing the music-on-demand service across a certain number of Web servers which are geographically distributed over the Internet. In this context, a typical approach to guarantee service responsiveness consists of dynamically binding the service client to the available server replica with the least congested connection. An approach recently proposed to implement such one adaptive downloading strategy at the Internet side amounts to the use of a software mechanism, called the Client-Centred Load Distribution (C^2LD) mechanism [11]. With this particular mechanism, each client's request of a given Web resource (e. g., a Mp3 file representing a certain song) 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 song. The mechanism periodically monitors the downloading performance of available replica servers and dynamically selects, at run-time, those replicas to which the client sub-requests can be sent, based on both the network congestion status and the replica servers workload.

As far as the protocol communication problems mentioned above are concerned, our wireless application has been structured based on the use of an *ALL-IP* approach [4], where the mobile UMTS device is simply considered as any other Internet-connected device. In essence, we have surmounted all the possible problems due to the time-varying characteristics, temporary outages, protocol interference, and high bit error rates of the radio link by resorting to a wireless session level we have developed on the top of the standard TCP protocol.

Our paper concentrates on two objectives. First, it provides a complete and succinct overview of the wireless application we have designed and implemented; second, it presents a set of preliminary experimental results that exhibit the performance of our system. The reminder of this paper is organized as follows. In Section 2 we provide a general overview of the architecture of the wireless application we have developed along with some important details about its main software components. Section 3 examines some preliminary performance results we have gathered from real-world experiments by using our system. Finally, Section 4 provides some conclusions.

2. SYSTEM ARCHITECTURE

The general system architecture of our proposed wireless application may be thought of as constructed out of the following three software components, as shown in Figure 1:

- The *Mobile Client application*. This part of our wireless application runs on the UMTS terminal, and has the responsibility of supporting the client during the subsequent activities of: i) searching the Mp3 files corresponding to the client's favorite songs over the Internet, ii) downloading them on the UMTS device, and, finally iii) playing out them as soon as they have been downloaded.
- The *Intermediate software System* (IS). This software component is hosted in an Internet server and represents the core of our system. It is in charge of managing all the communications between the UMTS device and the Internet infrastructures. In particular, the IS carries out:
  1. the *wireless communication* with the UMTS device, providing it with the access point to the Internet-based music distribution service;
  2. the *wireline* communication with the replicated Web servers. In essence, it is in charge of implementing a *reliable* and *responsive* discover-and-download service based on the user's requests.
- The replicated *Web Servers*. They are Web servers geographically distributed over the Internet which act as music repositories. In general, different Web server can be managed and administrated by different music service providers, and may also offer different set of replicated songs. Simply stated, this replication scenario can be thought of as a *loosely coupled replication system*, where, potentially, different servers support different set of music resources. Needless to say, each single musical resource may be replicated within a number of geographically dispersed Web servers.

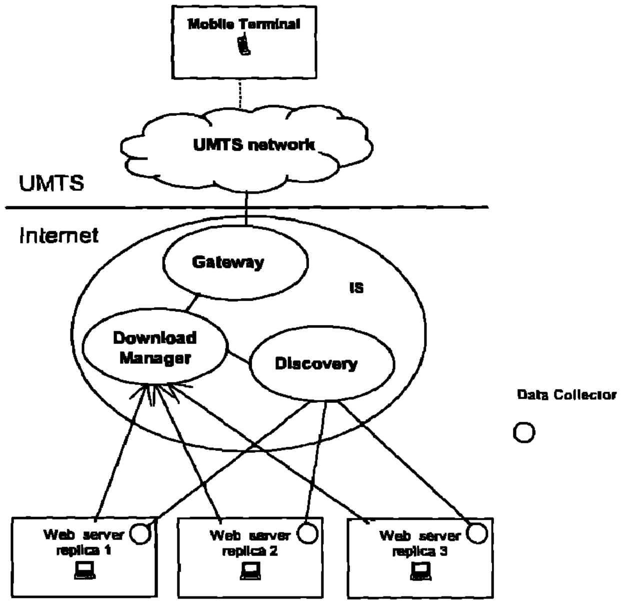


Figure 1: System Architecture

As far as the IS is concerned, it exhibits three main functions which correspond, in turn, to the following three different software sub-components of the IS:

- A *Gateway system*; the Gateway system accepts and manages all the requests for songs arriving from the client connected to a given UMTS terminal.
- A *Discovery system*; it has been developed to individuate and locate over our Internet-based system all the replicated resources (i.e., the replicated Mp3 files) which correspond to the songs that have been requested by the users.
- A *Download Manager*; it has been developed to carry out the activity of downloading from the Web server to the IS the songs that have been individuated by the Discovery system. The Download Manager performs this activity by exploiting the previously mentioned C^2LD download mechanism.

Succinctly, a typical download session works as follows: 1) a client from his/her enabled UMTS device issues to the Gateway a request for a certain Mp3 file (a song); 2) the Gateway system passes this request down to the Download Manager; 3) the Download Manager asks to the Discovery system the complete list of the Internet-based locations of all the replicated files that represent that song; 4) after having completed this phase, the Download Manager begins the download activity, by engaging all the different replica servers that maintain a copy of the requested song. This activity is carried out by exploiting the already mentioned C^2LD downloading mechanism; 5) finally, the reassembled Mp3 file is sent back from the Download Manager to the Gateway system, and from the Gateway to the UMTS terminal. Needless to say, in order for the system to work correctly, a preliminary phase has to be carried where each potential music server announces the list of musical resources it wishes to make available for sharing. Each music server which wants to add its own repository to our IS may do that by running a software application called the *Data Collector* that is in charge of communicating to the Discovery system the list of Mp3 files currently offered by that music server. The next Subsections are devoted to examine, in turn, each of the above mentioned software components of our wireless application, along with a number of relevant design and implementation details.

2.1 The Mobile Client Application

The Mobile Client application represents the interface between human users and the system services, and provides a set of *search-and-download* functions. Taking into account that a typical mobile device (such as a UMTS telephone or a PDA) possesses a very limited memory capacity and disk size, we have taken the decision to move all the "search and discovery" functions to the IS side. This entails that the Mobile Client application cannot autonomously determine which Mp3 file are precisely available, and where they are located. This even implies that users have to perform the download activity by stepping through two different phases. In the first phase, the user initiates a search for a given song, providing the system with all the available information (that may be even partial information) about song's title and author. Hence, the Mobile Client application contacts the Discovery system within the IS to verify the existence of a corresponding music resource within the system. If that resource exists, then this information is given back to the user.

(It is also worth noticing that users can perform multiple search procedures to refine their search or even to define a list of songs, called a *compilation*). In the second phase, the user can start the download activity to obtain the musical file that has been previously identified (or the required compilation). The system offers also a "*top-10* service" which provides to users the list of the 10 songs which are the most requested songs across the system. From an implementation viewpoint, it is worth noticing that our Mobile Client application has been developed with the aim to enhance portability on different mobile devices. To accomplish this goal, the software is written in Visual C++ (and, currently, developed on a Microsoft Windows CE platform [12]). Figure 2 illustrates the evolution of a full download session: at the beginning of the interaction with our wireless application, (Figure 2a, the leftmost one) the user has the intention to create a list of favorite songs by clicking on the button *RICERCA* (*search* in the Italian language).

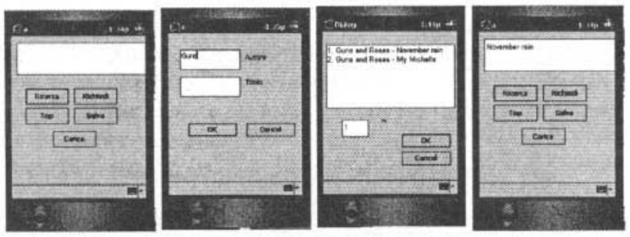


Figure 2: Snapshots from the Mobile Client Application

The search interface appears as shown in Figure 2b, and the search results are listed in Figure 2c. Finally, Figure 2d (the rightmost one) shows the updated download list with the song which has been chosen for downloading by the user. At this point, the user can perform the download activity by clicking on the button *RICHIEDI* (*request*), or even add new songs to the list by performing other searches. The button *TOP* creates a compilation of ten songs from the *top-10* service.

2.2 The Gateway System

Since traditional wireless accesses to the Internet are affected by problems of scarce bandwidth availability, typically, only a small portion of the potential Internet services is provided to wireless

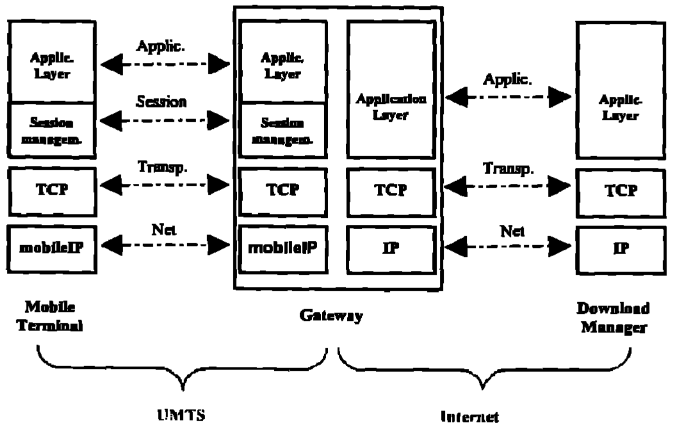


Figure 3: The Gateway Protocol Stack

users, by means of the so called *walled garden* approach [4]. Here, the transport protocol used within the mobile wireless environment is not TCP, but is instead a transport protocol that has been specifically ideated for mobile wireless. The WAP approach is exactly based on this kind of solution [13]: almost every layer of the standard TCP-IP protocol stack has been redefined and modified, with the result of loosing a full compatibility with standard IP applications which are designed for wireline environments.

Contrariwise, we have taken the decision to resort to a so called *ALL-IP* approach that exploits a standard TCP-IP stack for the communications between the Gateway system and the Mobile Client application software components. According to the adopted approach, our Mobile Client application works as any other Internet-connected device, and a standard end-to-end connection is guaranteed by using the standard TCP. Needless to say, if this approach is followed, alternative solutions must be found to all the problems which are caused by the wireless network access. We have surmounted those problems (and in particular the problems due to the frequent failures of the radio network access) by implementing, within our application, an appropriate *session* layer.

With this in view, Figure 3 shows the protocol stack we have designed and developed to support all the Gateway-related communications. In particular (as shown in the leftmost side of the Figure) the Gateway system communicates with the Mobile Client application over an UMTS link. As seen from the Figure, on the UMTS protocol stack an IP layer, based on the MobileIP (version 4) protocol, is implemented. On the top of this MobileIP level, a standard TCP layer has been built. Finally, to circumvent all the network problems due to the radio link layer, which were already mentioned in the Introduction of this paper, the application layer built on the top of TCP has been designed as constructed out of two different sub-layers:

- a *Session Layer*; this protocol layer is devoted to organize and to manage a download session which may possibly consist of different subsequent communication patterns, in the face of possible link outages.
- a *(Real) Application Layer*; this protocol layer is in charge of supporting the different connections needed to search and to download songs.

It is worth noticing, here, that our designed *session layer* provides users with the possibility of resuming a session that was previously interrupted due to subsequent temporary link outages. The session management mechanism we have designed and implemented has a greater importance for the full success of the activity of downloading Mp3 files onto an UMTS device. It is easy to understand, in fact, that very large files (e. g., songs of about 5 Mb/s) have to be delivered to the UMTS terminal, and this must be carried out in the presence of a wireless cellular access which typically exhibits a scarce connection stability and an unpredictable availability. Simply put, our *session layer* works as follows:

When the UMTS client application opens a connection to the Gateway, the Gateway assigns a unique identifier to this new session. If, eventually, the Gateway detects a network failure (i.e., the TCP connection comes out to be closed), the download status is saved at the Gateway side. In particular, a pointer to the last byte received of the Mp3 file is saved, along with the session identifier. At the same time, the identifier of the suspended session is saved at the Mobile Client application side too. As soon as the Mobile Client application is able to open a new TCP connection to the Gateway, the Client application tries to resume the interrupted session by exploiting the session identifier that was previously saved.

As a final note, it is important to remind that the session management mechanism we have developed is suitable for recovering sessions that are interrupted due to temporary link outages, but it is not adequate to recover from system failures occurring at the UMTS terminal or at the Gateway.

2.3 The Download Manager

The Download Manager is the real agent responsible for the download process, and has been built as an application on the top of the HTTP protocol. The Download Manager has been designed to be able to optimize data transmissions in the following sense:

- It maximize the service availability, i.e. it tries to maximize the percentage of successfully served requests.
- It maximize the service responsiveness, i.e., it tries to minimize the time after which a requested song is successfully downloaded at the UMTS terminal.

As already mentioned, the aforementioned objectives have been fulfilled, at the Internet side, by exploiting the technique of *replicated Web servers*. In essence, according to this technology, a software redundancy is introduced at the Internet side, namely by replicating (some of the) the music songs composing the music-on-demand service across a certain number of Web servers, geographically distributed over the Internet. In this context, a typical approach to guarantee service responsiveness and availability consists of dynamically binding the service client to the available server replica with the least congested connection [9, 10, 14]. An approach recently proposed to implement such one download strategy at the Internet side amounts to the use of a software mechanism, called the Client-Centred Load Distribution (C^2LD) mechanism [11]. In essence, C^2LD splits the client's requests for a given music resource into several sub-requests for fragments of the requested resource. Each fragment is required to a given replica server, and when a fragment is completely received from a replica server, another fragment is required. C^2LD dynamically adapts its sub-requests for fragments of different sizes depending on both the network and the server performances.

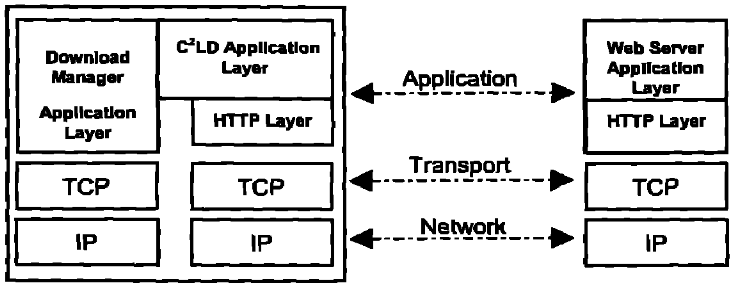


Figure 4: The Download Manager Protocol Stack

For each sub-request, an internal timeout is set, and if this timeout expires before the requested fragment is received, the fragment is required to another replica server. When all the fragments composing a given Mp3 file are received, the original file is obtained by reassembling all the fragments.Figure 4 shows the Download Manager protocol stack. As seen from the Figure, the Download Manager has to communicate with each different Web server replica (rightmost side of Figure 4) and hence forks into different children processes for each different song request. Each process uses a C^2LD application protocol, built on the top of HTTP, to download the Mp3 file from different Web servers. The interested reader may find precise information about C^2LD in [11].

It is worth concluding this Subsection by 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 replica of our system. The way in which our system is able to individuate if a requested song is in existence in the system, and where it is replicated, is discussed in the following Subsection.

2.4 The Discovery and the Data Collector

The software component of our developed system where the relevant information about songs are stored and indexed is the D*iscovery system*. 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:

- Accepting user's requests to establish a formal relationship between user requested songs and the correspondent Mp3 files stored in the system.
- Individuating the exact Internet location where a given Mp3 file (correspondent to a requested song) is located throughout the entire system composed of replicated Web servers.

To accomplish the aforementioned goal, the Discovery system is able to identify identical copies of the same MP3 file by calculating a unique 32 bit-based identifier (called the *checksum)* which is computed on the basis of each file content. Furthermore, it is important to notice that the complete lists of the information which are stored by the Discovery system are the following.

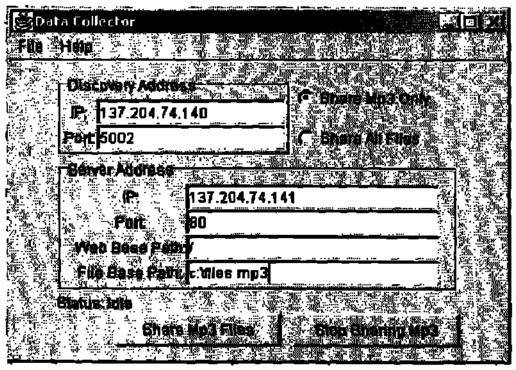
- The information needed for identifying the Mp3 file corresponding to a given song, namely: the ***File Name***, the ***File creation time***, the ***File length***, the ***Song Title*** and ***Author***, and finally the ***Checksum***.
- The information needed for localizing different identical copies of the same song, namely: the ***File Name***, the ***Host server*** (the server that maintains the requested song), the ***Path*** (along which the file is stored in the host server), the Transfer Application Protocol (e.g., HTTP) and finally the ***Checksum***.

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's Title and Author while, the latter, used to localize the requested files, is created on the basis of the checksum value mentioned above. We have devised two alternative methods for performing the calculation of the checksum:

- a *Centralized* method, according to which the entire MP3 file is transmitted from the host server to the Discovery systems that, in turn, performs the computations of the checksum, and
- a *Distributed* method, where each host server computes the checksum of its musical files and communicates the results to the Discovery system.

To minimize both the computational and the traffic overheads, we have taken the decision to implement the distributed method, where each server has to run locally a software application, termed the *Data Collector*, that 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 perform locally the checksum computation. The Data Collector is implemented as a Java applet to enhance the software portability, and also meets standard security constraints; as it can only read from the local file system, but it cannot execute 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 toward the Discovery and uploads the computed checksums to it.

Figure 5 shows a snapshot of the data collector application. As seen from the figure, two different kinds of information must be specified: the address of the Discovery system (under the form of the IP and port numbers) and the complete address of the server where the Mp3 files are stored (including the data path within the local file system).



Discovery Address
IP: 137.204.74.140
Port: 5002

Server Address
IP: 137.204.74.141
Port: 80
Web Base Path:
File Base Path: c:\files mp3

Status: Idle

Figure 5: Snapshots from the Data Collector Application

3. EXPERIMENTAL ASSESSMENT

We introduce below an experimental study we have developed in order to assess the effectiveness of our music-on-demand service. The intention behind our experimental study has been to investigate the quality of the Internet/UMTS download sessions carried out by our wireless application. In essence, we conducted 4000 experiments consisting in the download of a set of different Mp3 files. Four different replicated Web servers were exploited, at the Internet side. Instead, the communications between the Gateway and the mobile client was simulated by means of an UMTS simulator that is able to produce the transmission delay time of each frame at the radio link layer. Detailed information concerning the experimental models we adopted for our experiments are discussed in the following Subsections.

3.1 Application Level Model

As already mentioned, we used four different Web servers, geographically distributed over the Internet, providing the same set of 40 songs. The four different replica servers were respectively located in Finland, Japan, USA and New Zealand (Figure 6). Our IS application implementing the Download Manager and the Discovery system was, instead, running over a Pentium 3 machine (667 MHz, 254 MB RAM) equipped with the Windows 2000 Server operating system, and was located in Italy (Bologna).

The UMTS network was simulated by means of an UMTS simulator provided by the "Fondazione Marconi" (an Italian foundation for wireless computing). Finally, the UMTS device on which the Mobile Client application was running was simulated by means of a Pentium 2 computer (266 MHz, 128 MB RAM) equipped with the Windows CE operating system. In order to provide the reader with an approximate knowledge of the transmission times experienced over the considered Internet links, it is worth mentioning that the Round Trip Times, obtained with the ping routine, between the client and the four different servers (i.e., Finland, Japan, USA and New Zealand) measured, respectively 70, 145, 393 and 491 ms. As far as the downloading process is concerned, we have taken the two following basic assumptions:

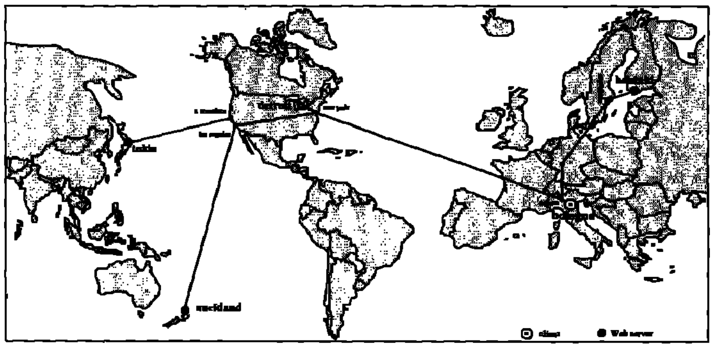


Figure 6: Web Server Replicas and Client Locations

1. *Mp3 file dimension:* in our experiments we used 40 different Mp3-based songs, whose correspondent file dimension ranged from 3 to 5 MB. The file dimension of 3-5 MB corresponds to the average file dimension of the songs maintained in the Napster system.
2. *Number of download activities*: as already mentioned, our software application provides support to two different types of downloading services: the former consists of downloading a single song, the latter amounts to the downloading of a complete song compilation. To evaluate the performance of our system under both these conditions, we conducted the following different experiments:
   - A set of independently replicated experiments consisting of the downloading of a single song.
   - A set of independently replicated experiments, with each one consisting in the download of a songs' compilation. The number of songs for each compilation ranged in the set (3, 5, 10). These three values were chosen based on the

consideration that the average disk capacity of typical Mp3 players never exceeds 50 MB.

3.2 TCP/IP and UMTS Models

Since, currently, no real measurements of UMTS wireless data are available, in our experiments the communication between the developed Gateway system (at the Internet side) and the Mobile client application running over the UMTS device was carried out through a simulated UMTS network, by exploiting the *background* traffic class. It is well known that the UMTS protocol stack consists of: a PHY (Physical) layer, a MAC (Medium Access Control) layer, an RLC (Radio Link Control) layer that implements an ARQ mechanism for ensuring reliable data transmission, and finally, a PDCP (Packet Data Convergence Protocol) layer that provides data and header compression to improve channel efficiency. On the top of this UMTS stack we have the standard IP and TCP protocols.

The UMTS network simulator we adopted is able to return after simulations a complete *Wireless Network Transmission Time* (WNTT) value computed at the PDCP layer. Needless to say, these WNTT values depend on some operational parameters, such as the amount of traffic present in the simulated cell and the number of active clients and their speeds. WNTT measurements include also the time spent for possible retransmissions at the UMTS RLC level. In our experiments, different values of this WNTT measurements were taken based on the different possible sizes of the TCP segments coming form the Internet side (namely of 120/440/920 bytes). The unique problem that stems from this hybrid approach (i.e. both experimental and simulative) is that segment errors and resulting retransmissions at the TCP level are not taken into account. To circumvent this problem, our experiments have included the possible retransmission time delays incurred at the TCP level, by exploiting an external delay introduction mechanism that was designed to take into account the typical TCP error recovery mechanism based on received ACKs. Simply stated, this delay mechanism compares the WNTT values obtained through the UMTS simulation against the *timeout* values computed by TCP. If the simulated WNTT value is larger than the correspondent TCP timeout value, then we must conclude that a retransmission must occur at the TCP level. In such one case, the WNTT value of that given TCP segment is augmented by an additional value which is chosen as equal to the next WNTT value extracted from the set of the UMTS-based simulated values. Consequently, the TCP timeout value is updated as follows. If a retransmission at the TCP level has been detected according to the method mentioned above, then the subsequent TCP timeout value is calculated as double w.r.t. to the previously computed value. If no retransmission at the TCP level has been detected, then the following adaptive formula for the calculation of the TCP timeout value, as proposed by Jacobson, is followed [15]:

$$\text{Timeout} = \text{RTT} + 4 * D,$$
$$\text{RTT} = \alpha * \text{RTT} + (1-\alpha) * M,$$
$$D = \alpha * D + (1-\alpha) * |\text{RTT} - M|,$$
$$\alpha = 7/8,$$

M = simulated value produced by the UMTS simulator.

3.3 Experimental Results

The first objective of this Subsection is to report some preliminary results concerning the Cumulative *WireLine Network Transmission Time* (WLNTT) values, that is the time spent over the wired Internet links to download a requested Mp3 file from the replicated Web server towards our IS, at the Internet side. These measurements have been compared with those that may be obtained by downloading the same Mp3 file with a standard HTTP GET method. The first row of Table 1 reports those results for Mp3 files whose size is 5 MB. The second row shows the average WLNTT percentage improvement obtained by our system (i.e., by C^2LD) w.r.t. standard HTTP. As shown in the Table, our system obtains an average percentage improvement over the fastest HTTP replica which is equal to 32 %.

Table 1: Cumulative WLNTT Results

	C^2LD	HTTP			
	(4 servers)	Finland	USA	Japan	New Zealand
Download time (seconds)	32.547	47.889	122.191	248.740	624.195
C^2LD improvement (percentage)	_	32%	73.4%	86.9%	94.7%

In addition, it is worth noticing (see Table 2) that our system, due to the adopted replication technology, is always able to carry out a complete download of the requested songs within a maximum time interval of 900 s. As shown from Table 2, this is not always true if we use the traditional HTTP protocol.

Table 2: Service Availability

	C^2LD	HTTP			
	(4 servers)	Finland	USA	Japan	New Zealand
successful download percentage	100%	98.5%	99.5%	95%	89%

The remainder of this Section is devoted to examine the Cumulative *Wireless Network Transmission Time* (WNTT) values that were obtained within our experiments, through UMTS simulations, to download entire Mp3 files to the UMTS devices.

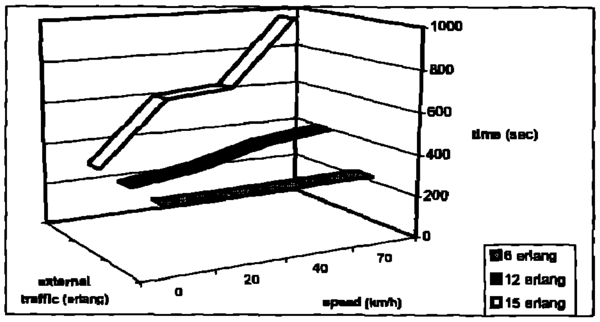


Figure 7: Cumulative WNTT Measurements

To this aim, Figure 7 reports those values depending on: i) the speed at which users move throughout the cell (expressed in

Km/h), and ii) the additional traffic in the cell (expressed in Erlang values). Several considerations are here in order. First, the obtained Cumulative WNTT results do not seem to depend on the TCP segment size (i.e., 160, 480, 960 bytes). Secondly, as shown in the Figure, the larger the traffic in the cell and the users' speed, the larger the corresponding Cumulative WNTT value. Third, the best Cumulative WNTT result may be obtained when the mobile device is still. In such one case a data rate of about 15 KB/s may be obtained.

Finally, it is worth noticing that the Cumulative WNTT values we have obtained from our simulation are in the range from 250 to about 950 seconds (from 4.5 to about 15 minutes) depending on the user's speed and on the additional traffic in the cell. Instead, with different wireless technologies, and in the absence of other external traffic, we would have obtained Cumulative WNTT values ranging from 1500 (GPRS technology at 28.8 Kb/s) to 3000 seconds (with GSM technology at 14.4 Kb/s).

4. CONCLUSIONS

In this paper we have reported on the design and the implementation of a wireless Internet-based application we have developed to support the distribution of Mp3-based musical songs to UMTS devices. We have also investigated performance issues related to the use of standard IP protocols, such as TCP, when operated over UMTS. The download time measurements we have experimentally obtained have shown that an appropriate structuring of the Internet-based wireless application may be very helpful to surmount the problems caused by the Internet standard protocols which were not especially designed for wireless environments. We wish to conclude this paper by mentioning that: i) currently, we have not already addressed any security and copyright protection issues for our wireless application. We plan to devote our future research to this important topic; 2) at the best of our knowledge, the only recent and significant results that have been presented within this particular research field describe a Internet-based wireless application which exploits a GSM connection for managing the communications along the uplink, while communications on the downlink are implemented through a Digital Broadcast Networks (DVB-T) [KEL01]. Unfortunately, in that paper only the system software architecture is discussed but no performance results are reported. This makes it very difficult to compare the two alternative approaches.

5. ACKNOWLEDGEMENTS

We wish to thank the "Fondazione Marconi of Bologna, Italy", the Department of Computer Science of the University of Bologna, Italy, and finally Microsoft Research Europe (Cambridge, UK) for the financial support to this work under a grant.

6. REFERENCES

[1] R. Kalden, I. Meirick, M. Meyer, "Wireless Internet access based on GPRS", IEEE Personal Communications, Vol. 7 N. 2 , April 2000, 8 -18

[2] D. Staehle, K. Leibnitz, K. Tsipotis, "QoS of Internet Access with GPRS", Proc. Fourth ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Rome, July 2001, 57-64

[3] UMTSForum, "What is UMTS?", http://www.umts-forum.org/what_is_umts.html

[4] G. Huston, "TCP in a Wireless World", IEEE Internet Computing, March-April 2001, 82-84

[5] MP3 resources by MPEG.ORG, http://www.mpeg.org/MPEG/mp3.html

[6] Napster official site, http://www.napster.com/

[7] Gnutella official site, http://gnutella.wego.com/

[8] Freenet Project Inc., *The Freenet project*, http://freenet.sourceforge.net

[9] M. Conti, E. Gregori, F. Panzieri, "Load Distribution among Replicated Web Servers: A QoS-based approach", Proc. 2nd ACM Workshop on Internet Server Performance, Atalanta, May 1999

[10] M. Conti, E. Gregori, F. Panzieri, "QoS-based Architectures for Geographically Replicated Web Servers", Cluster Computing 4, 2001, 105-116

[11] V. Ghini, F. Panzieri, M. Roccetti, "Client-Centered Load Distribution: A Mechanism for Constructing Responsive Web Services", Proc. 34th Hawaii International Conference on System Sciences, Maui, January 2001.

[12] Microsoft, Windows CE home page, http://www.microsoft.com/windows/embedded/ce/default.asp

[13] WapForum, "WAP Architecture Specification", http://www1.wapforum.org/tech/terms.asp?doc=WAP-100-WAPArch-19980430-a.pdf

[14] D. Ingham, S.K. Shrivastava, F. Panzieri, "Constructing Dependable Web Services", IEEE Internet Computing, Vol. 4 N. 1, January/February 2000, 25-33

[15] V. Jacobson, "Berkeley TCP evolution from 4.3-Tahoe to 4.3-Reno2", Proc. Eighteenth Internet Engineering Task Force, University of British Columbia, Vancouver, B.C.

BIOGRAPHIES

Marco Roccetti received an Italian Dott. Ing. Degree in Electronics Engineering from the Faculty of Engineering of the University of Bologna, Italy, in 1989. Since November 2000 he is a Professor of Computer Science at the Department of Computer Science of the University of Bologna. His research interests include: i) design, implementation and evaluation of multimedia computing and communication systems, and ii) performance evaluation and simulation of distributed and parallel computing systems.

Vittorio Ghini has an Italian Laurea Degree in Computer Science from the University of Bologna, Italy. He is currently a Ph.D. student at the Computer Science Department of the University of Bologna. His research interests include issues of distributed multimedia systems and quality of service over IP-based networks

Paola Salomoni has an Italian Laurea Degree in Computer Science from the University of Bologna, Italy. She is currently an Associate Professor of Computer Science at the Department of Computer Science of the same University. Her research interests focus on the design and development of distributed multimedia systems, as well as on distance teaching/learning computer based environments.