2. PREVIOUS AND RELATED WORK

2.1 Background

Noting that the number of devices that surround users at home, at work, and on the run is increasing; and realising that each device requires some kind of attention from the user, each consuming some amount of time, a new step towards enabling and exploiting a higher degree of intelligent behaviour of these devices is needed. A recent concept to enable the desired improvement of these devices is context awareness. The design of context-aware devices results in simplified user interfaces, applications that try to hide as much computation as possible from the user, and better services that take greater advantage of the existing infrastructure. One class of these more intelligent devices are audio-only wearable devices, for which a study was carried out, resulting in the creation of Nomadic Radio[1]. The main idea behind the Nomadic Radio was the unification of the functionality of most of the existing personal audio devices into a single audio-only intelligent wearable device that is able to anticipate the user's needs. Nomadic Radio achieves its purpose by building on text-to-speech synthesis tools, speaker and speech recognition, and spatial audio among others. With this background in mind, this thesis focuses on audio for mobile users in a quest to create more useful devices. To understand the rest of this thesis some knowledge in several areas is needed, this is present in the following.

2.1.1 Wearable Devices

There has been little agreement as to what the exact requirements are for a wearable device. One description given in [4] states that a device needs to be portable, enable hands-free use, possess a wide array of environmental sensors, and always be proactively acting on its user's behalf. This description is of quite a powerful and flexible device, but fails as a more general description as it excludes devices that are considered wearable nowadays. Another more flexible description of devices that can be considered wearable are devices that offer some kind of computing, that are worn or carried on one's person habitually and whose primary interaction is with the person wearing or carrying the device[4]. Examples of wearable devices that fit the latter definition, which will be the definition we will be assuming through the rest of this document, are laptops, cell phones, and Personal Digital Assistants (PDAs).

The performance of a wearable device can be measured according to two criteria: transparency and efficacy. This thesis will not go deeper into the

details of the performance of a wearable device, but will assume that our wearable devices are connected over wireless interfaces at all times and will assume that the details of connectivity are hidden from the user and hence not relevant for the thesis. The means of providing the appropriate selection of wireless interface is the topic of Giulio Mola's thesis[18].

2.1.2 Digital Audio Players

Music audio or audio of any kind, must be in some digital storage format to be able to exist online. In this digital form, we are able to access and listen to audio from our computers and many other digital devices[23]. Digital music is nothing more than binary data, i.e., strings of ones and zeros. One can basically find two kinds of audio players in the market, MP3 players, which are made to handle MP3 files; and real-time audio players, such as RealAudio[26] or Windows Media Player[28]. Sometimes, this division is invisible as players are designed in a very complete way and can operate as both MP3 players or real-time players at the same time.

Most audio players share features, such as standard operation buttons (play, stop, pause,...), volume controls, playlists generation (see section 2.1.3), display track (MP3 files often come with embedded track info, such as the artist's name, the title or the album), CD information, equaliser functions (RealPlayer [27] supports these functions), and compatibility with different sound formats (most players can play formats including: WAV, AIFF, MOV,..., in addition to MP3) since no single standard for digital audio online exists[23].

In this thesis, the player-client (built as one of the clients of the client/server application) generates audio output from digital audio content by selecting the appropriate player. However, only one of these players is a pre-existing player. This is mpg123, which is available in SmartBadge 4, and is obviously used for playing MPEG files. Because the results of using the FLite text-to-speech synthesis tool is a wave file, and no suitable player for wave files was implemented for SmartBadge 4, a wave player was designed (see section 3.6). Finally, as the software necessary for playing SUN AU files with SmartBadge 4 was already available, this type of audio file was also included in the player-client.

2.1.3 Playlists

In order to understand how to exploit audio for mobile users, which is the main purpose of this master's thesis, a basic application must be created. This application should build and maintain a list of content to be played;

i.e. what to output and when it should be output. This list will be dynamically modified to include new information, such as personal announcements, alerts, etc. of interest to the user.

The first approach to building such an application is to build a basic playlist with only scheduled output. This is similar to a typical broadcast radio station's playlist. Thus, a basic understanding of what a typical playlist usually includes is needed. Once we have this in mind, we can consider all the changes and additions necessary that must be made to this playlist for our specific purpose.

One of the major advantages of digital music today is being able to store dozens or hundreds of albums in a physically small digital storage device. This was popularised by the Apple iPod and Diamond Rio, but today many vendors sell such devices in a very wide variety of form factors. Additionally, one can find countless software tools for managing such digital music on the market. These tools vary from each other and differ in the way they produce a playlist, however, some concepts and features are included in almost all playlists and therefore we will examine these common features.

One of the most common characteristics of many radio station's playlists is that they consist of a mix of wave files[29] and MPEG files[24]. A typical playlist usually includes the titles to be played, their storage location (filename), date, and time following a predetermined order, selected by the user him/herself or by the programmer responsible for producing a playlist. Each element of audio content is called a *cut*. Frequently, the different music cuts to be played are organised into a database where one can include additional information such as music titles, openers, trailers, news, spots, promos, IDs, and comments[5].

Some information associated with the cut that can be useful are so called "cue points". These cue points are critical time points such as the starting point of the song, the intro time (which is the time when the vocals start), the outro time (which is the time when the vocals end), and the fade-out point[5]. These time points can be used when including new information to the playlist as one could decide to add the new content after the outro time point, instead of waiting for the song to end completely.

Some operations that applications should usually provide when dealing with a playlist are swapping cuts within the playlist, deleting cuts from the playlist, adding new cuts, creating a new playlist, and changing the cuts in a playlist.

Another interesting feature that should be contemplated is the possibility of having two mixer faders, each associated with a cut, that can be played simultaneously in order to be able to fade from one source to the other. This could be used when presenting a new personal announce to decrease the volume of the currently playing cut and increasing the volume of the announcement.

There is quite a lot of information and lots of software available for the creation and manipulation of digital playlists, such as XML playlists[6], protocols for fetching playlists[7], Java tools for manipulating playlists[8], a commercial playlist product description[9], and a complete open source broadcasting system[10] among others.

In this thesis we will produce a playlist using an application written in C. This playlist manager will maintain the usual information in playlists about title, artist, and album when possible (this information is sometimes embedded in MPEG files), along with length, size, sample rate, and other information necessary for the internal functioning of the program like priority, name of the file that contains the cut, recording type, and order of the cut inside the playlist. The functions of adding cuts to the playlist, deleting cuts, or saving the playlist in a file will be described in chapter 3, where the details of the application will be presented.

2.1.4 MPEG Audio Frame Header

MPEG audio files are built up from smaller parts called frames, which are blocks of data with a header. Generally, frames are independent items, each frame has its own header and audio information. However, due to internal data organisation in MPEG version 1 Layer III files, frames are often dependent. When one wants to extract information from an MPEG file, having a clear image of the file's structure can be highly beneficial. Figure 1 shows the structure of an MPEG file.

The first thing we encounter when reading a MPEG file is an optional list of ID3 tags. ID3 is a general tagging format for audio, which makes it possible to store meta-data about the audio inside the audio file itself. ID3 was designed to be as flexible and expandable as possible to meet new meta information needs that might arise. As described in [25], the ID3 tag is composed of a header, an optional extended header, different tags (each containing a header itself), and optional padding and a footer. The information contained in the tag is quite broad and can include the length of the MPEG file, the date of recording, the name of the composers or an

URL pointing at a webpage with information such as where the album can be bought[25].

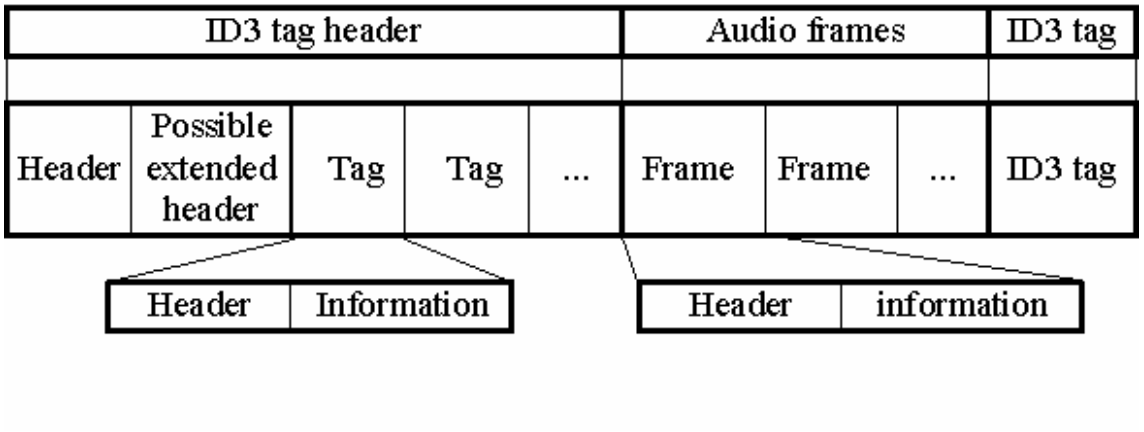

Figure 1: Structure of an MPEG file

There is no fixed order in which the tags should appear, and not all the existing tags are included in all the MPEG files. In this thesis we will look for the tag TLEN, whose information is a numeric string with the length of the song in milliseconds, but only when the MPEG file contains this tag. In the files that do not include the TLEN tag, the length of the MPEG file must be calculated using information such as bitrate, samplerate, and the size of the file in bytes (see section 3.4).

After the ID3 tag, audio frames start to appear in the file. Each frame has its own header, which is indicated by the first four bytes of each frame. In such header, information about the file's version, layer, the frame's bitrate, samplerate, and padding byte among other information is included. Not all of the frames must have the same bitrate. Bitrate switching, which means that the bitrate changes according to the content of each frame, is used in some MPEG files. This way lower bitrates may be used in frames where it will not reduce sound quality. This allows better compression while retaining high quality of more complex sounds. For a description of the exact meaning of each bit in the header see [24].

At the end of the MPEG file, another ID3 tag often appears. This is a fixed length tag that is 128 bytes in size, it is always at the very end of the file, and contains information about the file's song, artist, album, publishing year, some comment, and its genre. The genre is a number from 0 ('Blues') to 125 ('Dance Hall'), each number corresponds to one of the possible genres of an audio file[24].

2.1.5 XML – Extensible Markup Language

XML is a mark-up language for documents containing structured information[11]. Structured information contains both content (words, pictures, etc.) and some indication of what role that content plays (for example, content in a section heading has a different meaning from the same content in a footnote, which means yet something different in a figure caption, in a table, etc.). Almost all documents have some structure. A mark-up language is a mechanism to explicitly identify this structure in a document. The XML specification defines a standard way to add mark-up to documents.

XML differs from HTML. In HTML, both the mark-up tag semantics and the tag set are fixed. The World Wide Web Consortium (W3C), browser vendors, and the WWW community are continuously working to extend the definition of HTML to allow new tags. However, with these changes there is often some delay and difference in what the browser vendors have implemented. The result in compatibility is always a negative factor. In contrast, XML specifies neither semantics nor a tag set. In fact XML is really a language for describing mark-up languages. Thus XML provides a facility to define tags and the structural relationships between them[11].

XML is defined as an application profile of SGML. SGML is the Standard Generalised Mark-up Language defined by ISO 8879[30]. SGML has been the standard, vendor-independent way to maintain repositories of structured documentation for more than a decade, but it is not well suited to serving documents over the web. XML is a restricted form of SGML better suited for the Internet.

In order to understand the importance of XML, we need to understand why it was created. XML was created so that richly structured documents could be used over the web. As mentioned above, HTML and SGML, are not practical for this purpose. Some of the goals for XML are that it should be straightforward to use over the Internet, it should support a wide variety of applications, it shall be compatible with SGML, it shall be easy to write programs that process XML documents, the number of optional features in XML should be minimum and ideally zero, XML documents should be human legible, clear and easy to create, and that XML's design should be prepared quickly, be formal, and concise.

In the present thesis XML will be used to initially define the elements needed for the playlist to be used in the program. The necessary tags will be identified in XML. A possible improvement to the solution presented in

this thesis, would be a program that processes the XML document and automatically produces the C code necessary to manipulate it. Thus an alternate implementation method would be to manipulate the actual XML, for example using the Extensible Stylesheet Language Transformations (XSLT)[35], the Document Object Model (DOM)[33] or the Simple API for XML (SAX)[34] APIs .

2.1.6 Wireless LAN

A wireless LAN (WLAN) is a local area network that operates without wires in which the data is carried by radio or infrared transmissions. It offers the possibility of maintaining connectivity while the user moves about and the advantage of multiple users sharing the same network; requiring only a wireless card and authorisation to use the network, if this network is not an open one (i.e., if the network limits who may use this network). Using electromagnetic waves to transmit and receive data between users (and an access point) over the air, replaces the usual wired connections. An access point is a device that interconnects the wired and wireless networks, therefore enabling the wireless devices to communicate with devices attached to the wired network.

The IEEE 802.11 standard specifies the technologies for wireless LANs and how they can be used to provide portable, fixed and moving stations with connectivity within a local area network. The standard includes link layer encryption using the Wired Equivalent Privacy algorithm. Unfortunately, this algorithm provides only weak security. For addition details see [31].

High-bandwidth allocation for wireless LANs has made possible relatively low-cost “wiring” of classrooms. On KTH’s campus is Kista, Sweden, WLAN access is almost universally available. The low cost and ease of installation has lead to installation of wireless LAN systems where existing LANs are not already in place.

In some situations, a wireless LAN may be preferable to a wired LAN because it is cheaper to install and maintain. The coverage radii of an access point are 150 meters for indoors and 300 meters for outdoors, although better antennas, designed for the specific allocation, and the use of repeaters and other devices can enlarge an axis of the wireless cell’s area, but generally change coverage from roughly a circular cell to a more elongated shape. These facts, combined with the increasingly importance of mobility and increasing number of wearable devices, have made wireless LAN quite popular and widespread in the market. A wireless LAN

interface is now built into most laptop computers or PDAs, and many external wireless cards are sold for those devices lacking a built-in wireless LAN interface.

2.1.7 Context-awareness in Wearable Devices

Taking into account context-awareness when designing wearable devices can result in many advantages. However, there are slightly different ways in which the concept of context-awareness can be understood and defined. The term context can be seen as a knowledge about the user's and/or device's state, including things like surroundings, location, and situation. In a more general definition, context is any information that we can use for knowing the situation of a person, place, or object. In this thesis, we will consider context to be any information and data derivable by a wearable device about its state, environment, and surroundings.

Context-awareness is described as the awareness of the environment, location, situation, user, time, and current task. Context-awareness occurs when applications on wearable devices or servers use the context information when making decisions. As a result of using context information, these wearable devices achieve a higher degree of intelligence and exhibit greater independence in their behaviour. The use of this new information also should lead to "better" services (with respect to being more suitable for the user's current state) and simpler user interfaces (since much manual configuration can be eliminated).

Not all the acquired context information is relevant to applications running in the wearable devices, so a process of filtering this information must be performed in order to utilise the appropriate information at the right time while discarding information when it is not useful.

We obtain context information either by using sensors or exchanging data with other wearable devices or devices attached to the network. Sensors can be on the wearable device itself or in the environment. Their data can be transmitted over the network to any wearable device for its use. Sensors can provide speaker recognition (for use in voice recognition, for example), location based on Global Positioning System (GPS), time of day based on network attached clocks, humidity sensors (a type of environmental sensors), ... or software for monitoring. For details about collecting sensor data see Andreas Wennlund's thesis[13].

2.1.8 Nomadic Radio

As previously noted, people use a number of appliances and portable devices for a variety of tasks at home, in their workplace, and on the run; thus taking advantage of the information-rich environment that currently exists[1]. Attending to all these devices, each giving different kinds of notifications without taking into account the user's state (e.g., the user may be busy and maybe does not need to know the information included in a certain notification at the moment of reception), leads to overload of the user's limited attention. In order to be more effective, these devices should be aware of the environment in which the user is present. Nomadic Radio presents an approach to a first solution to this problem.

One of the problems of having multiple devices that Nomadic Radio addresses is the lack of differentiation in notification cues, i.e., using the same (binary) auditory cues for all the messages makes it impossible for the user to easily distinguish between urgent and important messages and non-urgent and less important ones.

Minimal awareness of the user and the user's environment enables filtering notifications which will interrupt the user. Intelligent devices can learn from prior interactions with the user and hence can take into consideration the user's behaviour, concerning the same (or similar) notification. This awareness can be used to co-ordinate notifications, so that the user's different devices work co-operatively resulting in only one notification occurring at any given time[1].

Nomadic Radio's goal was to give timely and filtered information relevant to the current user state[2]. A variety of auditory techniques must be supported in a non-visual wearable interface. For browsing and scanning messages in an easy way, spatial listening can be used. A spatial sound system is able to place individual voices in particular spatial locations. Synthetic speech and speech recognition provides hands-free navigation and control, and auditory cues provide effective awareness and notifications. All these auditory techniques must operate synchronously in order to be useful and not confuse the user[2].

The system operates as a wearable audio-only device, providing a hands-free and unobtrusive interface to a nomadic user, although a visual interface is used for setting user preferences and server-related functions. The prototype used as a platform for the development of Nomadic Radio was the SoundBeam Neckset, patented by Nortel for use in hands-free telephony. It consists of two directional speakers mounted on the user's

shoulders (necessary for providing spatial audio), and a directional microphone placed on the chest[1].

An audio-only interface has been incorporated in Nomadic Radio, and a networked infrastructure for unified messaging for wearable access. This master thesis tries to understand how to exploit audio for mobile users, building upon improvements suggested by Nomadic Radio.

2.1.9 Connectionless Transport: UDP

Two widely used transport protocols are used by Internet applications, these are the Transfer Control Protocol (TCP) and the User Datagram Protocol (UDP). Both are part of a large collection of protocols referred to as TCP/IP stack.

UDP provides simple but unreliable datagram services, it is a minimal transport service. As defined in STD 6, RFC 768[36], UDP is a connectionless protocol which is layered on top of IP and it requires neither a connection nor does it guarantee delivery. As a result, it is lightweight and efficient, but leaves to the application program all the error processing and retransmission. An application using UDP must deal directly with end-to-end communication problems (retransmission for reliable delivery, packetization and reassemble, flow control, congestion avoidance). UDP is used by applications that do not require the level of service of TCP (a reliable, connection oriented protocol) or that wish to use communications services not available from TCP.

The only services of UDP beyond IP (the underlying protocol) are checksumming of data and de/multiplexing by port number. UDP operates as a transport protocol as follows. UDP takes messages from an application process, attaches source and destination port number fields for the multiplexing/demultiplexing service, and passes the resulting "segment" to the network layer. The action that the network layer takes is to encapsulate this segment into an IP datagram, and then, make a best-effort attempt to deliver the segment to the receiving host. When the delivery is successful and the segment arrives at the receiving host, UDP uses the port numbers and the IP source and destination addresses to deliver the data in the segment to the correct application process. UDP does not implement three-way handshaking (always present in TCP) between sending and receiving transport-layer entities before sending a segment. For this reason, UDP is said to be *connectionless*[14].

Although one might consider that TCP is always preferable to UDP since it provides reliable data transfer, there are many application better suited for UDP. As UDP is connectionless, it does not introduce any delay establishing a connection; unlike TCP which uses a three-way handshake to establish a connection before starting to transfer data, as commented above.

UDP does not maintain connection state and hence does not track parameters such as receive and send buffer occupancy, congestion control parameters, or sequence and acknowledgement numbers. This enables a server devoted to a particular application to support many more active clients than if the same application were run over TCP. Additionally, a smaller segment header overhead is always more convenient than a big one as more bytes can be used for transmitting useful information. This fact makes UDP a desirable protocol for small amounts of information, as it only has 8 bytes of header overhead.

Another advantageous factor when using UDP is its unregulated sending rate. The speed at which UDP sends data is only constrained by the rate at which the application generates data, the capabilities of the source (CPU, clock rate, etc.) and the access bandwidth. However, the receiving host does not necessarily receive all the data. When the network is congested, a significant fraction of the UDP-transmitted data could be lost due to router buffer overflow. Thus, the receive rate is limited by network congestion even if the sending rate is not constrained[14].

As we can see in table 1, UDP is used for RIP routing table updates, because the updates are sent periodically, so that lost updates are replaced by more up-to-date updates. UDP is also used to carry network management (SNMP) data, since it must often run when the network is in a stressed state, i.e., when reliable, congestion-controlled data transfer is difficult to achieve. DNS runs over UDP in order to avoid TCP's connection establishment delays.

To realise advantages stated above, the transport protocol that will be used in this master thesis is UDP. It will be used for the communication between the server and the several clients. The details of this communication are described in section 3.

Table 1: Popular Internet applications and their underlying transport protocols.

Application	**Application-layer protocol**	**Underlying Transport Protocol**
electronic mail	SMTP	TCP
remote terminal access	Telnet	TCP
Web	HTTP	TCP
file transfer	FTP	TCP
remote file server	NFS	typically UDP
streaming multimedia	proprietary	typically UDP
Internet telephony	proprietary or SIP or H323	typically UDP
Network Management	SNMP	typically UDP
Routing Protocol	RIP	typically UDP
Name Translation	DNS	typically UDP

2.2 Related Work

There has been a rapid evolution in the research on context-awareness and wearable devices, since it first began in the 1980s. Today one can find lots of information on the subject thanks to the number of research projects that have been carried out. We will give short summaries of related work.

2.2.1 SmartBadge 4

SmartBadge 4 is the fourth version of the SmartBadge. It has been developed at Hewlett-Packard Laboratories in conjunction with the Royal Institute of Technology (KTH), as a prototype of future smart card systems. The first badge of this version was operational on February 2001 and in July 2001 it began running Linux. This version is a 12 layer printed circuit board with a ball grid array (BGA) mounted Intel Strong ARM SA1110 processor and SA1111 companion chip[3].

It measures 64x110 mm and is equipped with several sensors such as a 3-axis accelerometer, temperature sensors, humidity sensors and light level sensors. Some other components of SmartBadge 4 are infrared, PCMCIA, USB and compact flash port interfaces, providing a wide diversity in its connectivity and communication.

This badge can be seen as a technology platform for biometrics, context sensors, audio processing, video streaming, wireless systems, power optimisation, and measurements among others. The SmartBadge 4 has been used in courses at KTH such as the Mobile Personal Communication module of the Telecommunication course, and at University of Wollongong (Australia).

2.2.2 Active Badge

The Active Badge was developed between 1989 and 1992 at AT&T. Using Active Badges, one can be located in a building where the system is installed, as the badge repeatedly transmits an infrared signal identifying itself. This infrared signal is received by networked infrared sensors installed in the building. Each of these sensors has a unique network address and thus provides the system with information about the location of those badges whose signals it receives[15].

2.2.3 Festival-Lite

Festival-Lite (FLite) is a small, fast run-time speech synthesis engine developed at Carnegie Mellon University (CMU). It is mainly designed for small-embedded machines, in addition to large servers. It is a very portable engine which can be used on most platforms. FLite is written in ANSI C and offers text to speech synthesis in a small and efficient binary. FLite is a synthesis library that can be linked to other programs, it includes two simple voices with the distribution, an old diphone voice and an example limited domain voice, which uses the newer unit selection techniques developed by CMU[16]. The result of the text to speech synthesis is an ordinary wave file. In the present master thesis, this tool will be used for translating textual notifications into audio ones, getting closer, in this way, to an audio-only device.

2.2.4 MyCampus

MyCampus is an agent-based environment for context aware mobile services, developed at Carnegie Mellon University (CMU) and evaluated on their campus. It uses smart agents running on a PDA that uses a context server to obtain context information. The agents can suggest places for having dinner on the campus basing its decision on the user's schedule, their position, and the expected weather, for instance. Restaurants and weather information are obtained from the server and the user's schedule can be already on their PDA or on a server along with other relevant

information about the user. Users can download to their PDAs task specific agents in order to access the services they are interested in[17].

2.2.5 Wireless Diversity: Vertical Handoff Optimisation

Today, many different wireless technologies are available. Handoff mechanisms are needed to provide mobile computing communication with reliability and transparency. A good deal of the work is addressed by mobile IP itself, but addressing wireless diversity is the next step. Among this variety of wireless technologies, some technology provides wide coverage, while others provides higher bandwidth but are deployed locally, and even (directional) ad-hoc links can be used to carry IP traffic. To take full advantage of the infrastructures, a mobile device, with multiple network interfaces, should be able to dynamically switch from one link technology to another by changing from one interface to another. Hopefully, this would be totally transparent to the user. Moreover, multiple interfaces allow the device to choose, each time a new connection has to be established, which interface to select to route the datagrams over, based on the type of service desired/wanted. In Giulio Mola's thesis a possible solution is presented, which considers both vertical handoff optimisation and policy based management. The test platform is the SmartBadge v4, equipped with both GPRS link and IEEE 802.11b WLAN interfaces[18].

2.2.6 Context-Aware Support for Opportunistic Mobile Communication

The number of ways in which humans interact with each other has increased tremendously. Due to modern technologies, people thousand of miles apart are able to interact with each other, for instance, at low cost, through the internet. Further improvements of these interaction possibilities build upon context awareness and location awareness. Today, both concepts are poorly utilised as sources of information and further exploitation of them is needed. Location awareness systems collect data about the absolute or relative location of the user. This project focused on designing an application which enables users to learn of each other's presence and their location through an audio interface. Users learn of each other's presence and status by listening to the audio (wave) files generated by Festival-Lite software running on the SmartBadge 4. A key feature of this system was coupling an instant messaging system (via Jabber) to Festival-Lite[19].

2.2.7 Guided by Voices: An Audio Augmented Reality System

Lyon, Gandy, and Starner present a lightweight and inexpensive infrastructure for augmented realities that uses a simple wearable computer[20]. Whereas most traditional augmented reality systems overlay graphics onto the user' s environment, this system employs only audio. A positioning infrastructure using radio frequency (RF) transmissions was created to provide approximate location require little computational overhead. This RF based location system plays digital sounds corresponding to the user's location and current state. As a result, the system created is much less expensive and computationally complex than traditional augmented reality (AR) systems. "Guided by Voices" is an AR game developed to use a prototype infrastructure that allowed its creators to explore the unique issues involved in creating an engaging augmented reality audio environment. In the game players move around in the real world, but this triggers actions in the virtual (game) world. Some of the issues involved in creating audio-only augmented reality games were presented, along with the location infrastructure generalizable to other audio augmented realities.

2.2.8 A university licenses Napster for its students

Penn State University becomes first in the United States to offer its students a legal high quality alternative to private file sharing services. This new alternative is based on an agreement with the online music service Napster, which is a division of Roxio, Inc. Napster is the world's most recognised brand in online music, and offers access to a large catalogue of online music, with more than 500,000 tracks. A user-friendly interface allows people to quickly search for music, discover new artists, burn CDs, and transfer music to any of 40 different portable devices, among other interesting services. Members can permanently purchase songs for 99 cents each or $9.95 an album.[39].

Penn State University will make Napster's Premium Service available at no cost to its students. This service intends to meet students' interests, since recent studies revealed that the trend when listening to music goes toward digital music in PC and MP3 players. Students, who have demonstrated a voracious appetite for online music, are interested in getting extensive digital access to music. Student focus groups at Penn State have already been testing the Napster service. Thousands of testers will be deployed in the spring semester, in order to give feedback before expanding the service to the rest of university's students. For further details see [40].

2.3 Prerequisites

In order to fully understand this thesis the reader needs to have some previous knowledge and understand the basic concepts and fundamentals of data and computer communication, including wireless communication (specifically Wireless Local Area Network (WLAN)), and the principles and functions of communication protocols, specifically the TCP/IP stack.