# A Location-aware, Service-based Audio System

Robin Kirk and Jan Newmarch
School of Network Computing
Monash University Peninsula Campus
Melbourne, Australia
{robin.kirk, jan.newmarch}@infotech.monash.edu

*Abstract*— **With the development of computer-based location tracking systems and sensor systems a large amount of contextual information is available to the application developer. Information such as the location and motion paths of users and physical objects, and the ambient noise and temperature of the surrounding environment can be used to tailor a users experience with an application, to best suit their requirements. This paper introduces a location-aware audio streaming application that utilizes a networked multimedia architecture. This architecture facilitates the advertisement, discovery and connectivity of audio services. The streaming application uses both the user's current location, and the location of surrounding active loudspeakers to determine the best output device to use.**

***Keywords- home networks; multimedia distribution protocols; network architecture; session user and device mobility; middleware, pervasive computing, multimedia technologies; interoperability; location-based services***

## I. INTRODUCTION

Computer applications are usually not programmed to handle situational change, often leading to inappropriate program behavior. If a user is listening to loud music in the lounge room and another user enters the room, a doorbell or phone rings, a common reaction would be to reduce the music volume. Currently a user must do this manually, as the stereo is unaware of these events. In situations such as these, context-aware applications can be used to provide improved functionality and automation, thus increasing the enjoyment and satisfaction of the interaction with the device. To achieve this, the application can use location, and possibly some form of contextual information relating to a user and their physical and social environment. This information can be collected by asking the user, or implicitly by monitoring user behavior and environment.

The ability of an application to deliver multimedia to users based on their context is a powerful concept. Multimedia can be directed towards the user's location, and switch output device if the user moves. Volume can be adjusted in response to events, such as a person entering the room, or the doorbell ringing. Despite these advantages, very few context-aware multimedia systems have been developed. One such system is the Intercom system by Nagel et al [1], currently under development at the Georgia Tech Aware Home [2]. The system allows context-aware intra and inter-home communication between occupants. Features of the intercom include caller id, contextual information gathering and processing, multiple connections and "follow-me" audio. If a callee does not want to be interrupted to chat (sleeping, working) the system will inform the caller of the callee's status. Audio is output through speakers in the roof of the home, and received through wearable wireless microphones. Commands are issued via voice recognition or touch screen. The Context Toolkit [3] was used to design the intercom, allowing flexibility of location system implementation. The Intercom system is designed for voice communication only, and does not allow for movement of speakers.

Most architectures for home audio-visual systems such as the Java Media Framework (JMF) [4] or Microsoft Direct Show [5] are based on a local model, where all generators (e.g. TV tuner card) and consumers (e.g. soundcard) are all on the same machine. Even though JMF supports remote audio by means of HTTP and RTP [6], it hides these under a local programming model. Network architectures are either based on existing middleware such as C++, often extending it in some way, or build their own middleware structure oriented towards a particular view of A/V. In the first class are systems such as Multimedia System Services and the Multimedia Component Architecture [7]. In the second category are systems such as Network-integrated Multimedia Middleware (NMM) [8]. There is work on distributed A/V systems using Java such as HAVi [9] but this is quite specific to the Firewire [10] networking protocol.

This paper describes a "follow me" audio application that is aware of the user's location, and the location of speakers around the room, and outputs audio to the closest speaker to the user. The application monitors the distances from each speaker to the user, and will switch the audio stream if the closest speaker changes. Speakers under development are networked loudspeakers which consist of a basic speaker, power amplifier and a micro PC (running Linux) with a wireless network card [11].

The application uses our service-based audio architecture [12] that facilitates audio service advertisement and discovery. Use of this framework allows the application to be extended to support many transport protocols and multiple presentation formats. Furthermore, quality of service extensions may be integrated. The framework uses Jini [13] for service management. Jini is a middleware system built on Java that is able to fully exploit Java networking capabilities and object mobility.

This paper is organized as follows: Section II discusses the area of context-aware computing. Section III defines and discusses the methods available to determine the location of users and devices. Section IV describes the audio system framework that is used. Section V describes the implementation of the source and sink service, and the location aware interface. Section VI describes the test environment and observations. The paper concludes with a summary and discussion of future work.

## II. Context Aware Computing

Context is defined in the dictionary [14] as "The circumstances in which an event occurs; a setting". This may include people, objects, temperature, social setting, lighting etc. The concept of context-aware computing was first defined in 1994 by Schilt et. al.[15] as applications that "adapt according to the location of use, the collection of nearby people, hosts and accessible devices, as well as to changes of such things over time". Dey and Abowd provide a more general definition; "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"[16]. The latter definition best defines context-aware computing, as it places no emphasis or constraints on the type of contextual information that may be required. The aim of context-aware computing is to enable applications to adapt to situational change in an attempt to improve the interaction between the user and computer. This can be achieved by providing a specialized interface, automating tasks or adjusting program settings to provide a unique and personalized experience.

The first generation of context aware applications used location of users to refine and personalize applications in an office environment. The Active Badge office system [17] produced at the Olivetti Research Lab in the early 90's was the first its kind. Staff members allowed the application to track their movements around the building using wearable infrared devices. The application reported the nearest room number, so staff could be contacted on the nearest phone. The ParcTAB also had context aware applications to determine current room details and the nearest printer [18]. The Georgia Institute of Technology has been researching context-aware applications since the mid to late 90's. Projects include a context-aware Intercom, In/Out Board, Information display, whiteboard, and conference assistant [19].

A number of context-aware systems have been developed to provide location-based information to mobile users. Georgia Tech produced Cyberguide [20], a system to provide tourist information to visitors to the campus. Cyberguide users carry an Apple handheld computer that connected to a central PC via infrared to beacons around the center. The University of Lancaster created GUIDE [21], to provide tourist information on a Fujitsu handheld. Carnegie Mellon University has a tourist assistant system called SmartSight [22]. This system uses a wearable computer to answer spoken questions about local landmarks, provide translations and aid navigation.

## III. Determining Location

There are four main categories of location tracking technologies used to determine the location of a user or physical object in an indoor environment:

1. Infra-red (IR) – Fixed beacons receive IR signals from transmitters worn on the body to determine location. IR has short range, and objects and sunlight can interfere with the signal. Systems such as Active Badge [17] and Cricket [23] use IR tracking.
2. Proximity – Detecting when something is near an object. For detecting physical contact, pressure or touch sensors are used. Electronic ID tags can be used to identify users. This technology is the basis for Smart Floor [24]. Proximity location sensing does not give accurate room location unless the technology is installed on all areas of the floor.
3. Ultrasonic – This method calculates location using some triangulation from short pulses of ultrasound emitted from a transmitter worn on the body to beacons installed in the roof. As the range is limited, many beacons must be installed, increasing the cost and deployment effort. One such implementation of ultrasonic tracking is Active Bats [25].
4. Radio Frequency (RF) to determine the objects location. Recently available RF commercial software products such as the Ekahau Positioning Engine [26] and AeroScout [27] use wireless network infrastructure to determine location, as does MS RADAR [28]. Due to their low setup cost, long range and deployment effort compared to the other three location sensing techniques, RF tracking systems are currently the most popular choice for location tracking.

For the location-aware audio system, the deciding factors for what location sensing system to use were ease of integration, accuracy and cost. Electromagnetic and proximity based solutions are the most accurate, but require the purchase of expensive specialized hardware. The Ekahau and AeroScout positioning systems have a Java API, utilize existing wireless access points and are accurate to a few meters. In the end, the license fee for the Ekahau system was the more affordable of the two, and therefore was the chosen technology.

## IV. A Service based audio ARchitecture

At the most abstract layer a service based audio system consists of three players:

1. Sources of audio data
2. Sinks for audio data
3. Controller Clients

Controller clients should link sources to sinks, and leave them to decide how or if they can communicate. Section IV.A discusses the factors that determine compatibility. Figure 1

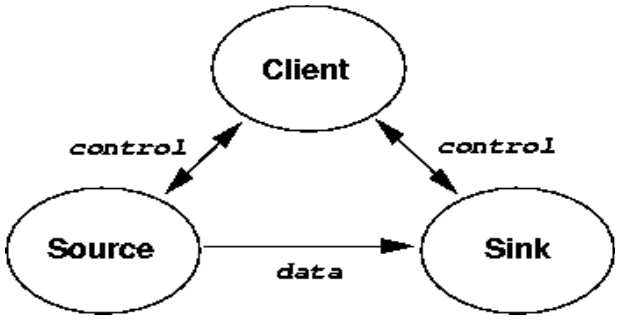


Figure 1. Communication Paths

shows the communication paths involved from a client viewpoint.

For simplicity we define two interfaces: Source and Sink. To avoid making implementation decisions about pull versus push, we have methods to tell a source about a sink, a sink about a source, to tell the source to play and the sink to record.

However, adopting such open interfaces does not address any incompatibility issues between A/V services. There is no way for a client to know if participating services can talk to each other as they may use different transport protocols, or the sink may not support the source media format. For example, if a WAV service sends the file using an array of bytes, a sink expecting an RTP transmission cannot receive the media. Streaming media protocols such as RTP were designed for client/server use, and may not cooperate from source to processor to sink. The responsibility of negotiating a transport protocol and media content must fall on the source and sink. If the source and sink fail to negotiate a valid transport and content, an exception should be thrown. This violates the principle that a service should be useable based on its interface alone, but considerably simplifies matters for controller clients.

A controller that wants to play a sequence of audio tracks to a sink will need to know when one track is finished in order to start the next. The play() and record() methods could block till finished, or return immediately and post an event on completion. The second method allows more flexibility, and as a result requires add/remove listener methods for the events.

Finally, there are the exceptions that can be thrown by the methods. Attempting to add a source that a sink cannot handle should throw an exception such as an IncompatibleSourceException. A sink that can handle only a small number of sources (for example, only one) could throw an exception if too many sources are added. A source that is already playing may not be able to satisfy a new request to play.

### *A. Design Factors*

The transport layer may be reliable (slow) TCP, unreliable (faster) UDP, HTTP (even slower), with some QOS such as RTP or some other network technology protocol such as Bluetooth [29] or FireWire.

There are an enormous number of audio formats, from encumbered formats such as MP3 [30] (for which you are required to pay license fees for encoders and decoders), unencumbered equivalents such as Ogg-Vorbis [31], compressed (MP3 and Ogg-Vorbis) or uncompressed (Sun AU[32] or waveform), lossy or lossless. In addition, there are many wrinkles in each format: little- or big-endian; 8, 16 or 32 bit; mono, stereo, 5.1; sample rate such as 44.1 kHz, 8 kHz, etc

Audio comes from many different sources: tracks off a CD, streaming audio from an FM station, speech off a telephone line. The MPEG-7 standard [33] concentrates on technical aspects of an audio signal in attempts to classify it, while the CD databases (CDDB) such as Gracenote [34] classify CDs by Artist/Title - which breaks down with compilation CDs and most classical CDs (who is the artist - the composer, the conductor or the orchestra?)

An audio stream may be "pushed", such as an FM radio stream that is always playing. Or it may be "pulled" by a client from a server, such as in fetching an MP3 file from an HTTP server

The two interfaces given earlier are enough to identify sources and sinks to a third party client (or to each other). In order to negotiate whether they can talk to each other may require more information, which can be supplied by further interfaces.

### *B. Content Interfaces*

The Java Media Framework (JMF) has methods such as getSupportedContentTypes() which returns an array of strings. Other media toolkits have similar mechanisms. This isn't type-safe: it relies on all parties having the same strings and attaching the same meaning to each. In addition to this, if a new type comes along, there isn't a reliable means of specifying this information to others. A type-safe system can at least specify this by class files.

Interfaces are more type-safe than strings: a WAV interface, an Ogg interface, etc. This doesn't easily allow extension to the multiplicity of content type variations (bit size, sampling rate, etc), but the current content handlers seem to be able to handle most of these variations, so it seems feasible to ignore them at an application level.

The content interfaces are just place-holders:

```
package presentation;


public interface Ogg extends java.rmi.Remote
{
}
```

A source that could make an audio stream available in OggVorbis format would signal this by implementing the Ogg interface. A sink that can manage OggVorbis streams would also implement this interface.

### *C. Transport Interfaces*

In a similar way, the transport mechanisms may be represented by interfaces. A transport sink will get the

information from a source using some unspecified network transport mechanism. The audio stream can be made available to any other object by exposing an InputStream. This is a standard Java stream, not the special one used by JMF. Similarly, a transport source would make an output stream available for source-side objects to write data into:

```
public interface TransportSink {

    public InputStream getInputStream();

}// TransportSink

public interface TransportSource {

    public OutputStream getOutputStream();

}// TransportSource
```

## V. LOCATION-AWARE IMPLEMENTATION

### A. Client

The location-aware implementation consists of a Jini client that is not only aware of the sources and sinks, but also the locations of all PC's or PDA's running the Ekahau client program. A music file may be chosen, and streamed from a source, to a group of possible sinks. The client then calculates the closest sink to the chosen tracked device (a PDA for example) and will play the song to that sink. A map is shown of the area, depicting the mobile user and mobile sinks. The client periodically re-calculates to determine the closest sink to the PDA, if another sink becomes closer, the stream is re-directed to that sink. The audio will continue from the point it was stopped at the source.

### B. Source/Sink

As the location-aware audio system streams audio, a "push" implementation approach at the transport layer was required. A Sink exposes an OutputStream, and the Source writes sound data read from the source file to that stream. TCP/IP sockets were used to transfer the file over the network.

At the presentation level, a Sink was implemented that is capable of handling OGG Vorbis and WAV files. Ogg Vorbis support is achieved using the OGG vorbis tritonus plug-in [35]. Using standard Java libraries, a FileInputStream is read from the socket and converted to an AudioInputStream. The file is converted to WAV on-the-fly for output to the soundcard if required.

## VI. TESTING

### A. Test Environment

A basic test environment was constructed, with two sinks, one source, one client and one PDA used for tracking purposes only. The test was conducted in a large 10x10 meter office area. Two wireless access points were placed an either end of the room.

The hardware specifications of the machines used were:

• Sinks (2) – Pentium II 350MHz 128MB RAM 802.11b PCI card.

• Source – Pentium 4 2.53GHz 512MB RAM 802.11b PCI card

• Location-aware client – Pentium M 1.4GHz Laptop 512MB RAM 802.11b Mini-PCI card

• Tracking client – IPAQ PDA 802.11b PCMCIA card

### B. Test Strategy

A test case was carried out several times in the test environment to demonstrate that the system functions as designed, and the responsiveness of the Ekahau location system. Sink, Source and location-aware client services, and the Ekahau client program on the Sinks and PDA were started. The location-aware client was then used to select the Source and Sink services and initiate the session. The PDA was initially located at one end of the room, near Sink 1. The PDA was then picked up and carried over at walking pace to the other side of the room near Sink 2. When the PDA became closer to Sink 2, the time taken for the stream to cease at Sink 1 and begin at Sink 2 was recorded. The PDA was then carried back to Sink 1. When the PDA became closer to Sink 1, the time taken for the stream to cease at Sink 2 and begin at Sink 1 was recorded.

### C. Results

The system functioned as designed in all tests that were conducted. The average time taken for the location system to trigger a Sink change was five seconds. The delay was longer than anticipated, and unsatisfactory for this application. This delay may be explained by the lack of an adequate number of wireless access points. The accuracy of the Ekahau positioning is mainly dependent on the number of audible access points installed. At least three access points is recommended, so the accuracy of the location system during testing was definitely less than the advertised accuracy of one meter.

The responsiveness of the location system is also dependent on which method call is made on the LocationEstimate object received from the Ekahau positioning engine, getAccurateLocation() or getLatestLocation(). The current implementation uses the getLatestLocation() method to minimize the delay, at the expense of greater accuracy.

## VII. CONCLUSION

In this paper we have presented an overview of the area of context-aware computing and multimedia systems, multimedia middleware, and location sensing technologies. An implementation of a location-aware audio system, which forms part of our service-based audio framework, was described in detail. An explanation of the testing carried out was provided, and the initial results discussed.

The location-aware audio system is the first stage in realizing a truly context-aware audio system. A planned variation to this is to have all nearby sinks playing the audio stream at different volumes determined by the user's location. The aim of this is to maintain the overall volume at the desired level, regardless of where the user is standing. Following this, the next stage of development will focus on developing a household audio system capable of adapting itself to the surrounding environment. Contextual information such as ambient noise levels gathered from microphones and user preferences will be used. We are also devising reasoning algorithms that will allow the audio system to react to state changes in other smart devices, such as telephones and doorbells. Additionally, quality of service extensions will be developed and integrated into the framework to deliver audio in the best way possible.

## VIII. REFERENCES


[1] K. Nagel, C. Kidd, T. O'Connell, A. Dey, and G. Abowd, "The family intercom:developing a context-aware audio communication system," presented at UbiComp2001, Atlanta, Georgia, 2001.

[2] C. Kidd, R. Orr, G. Abowd, C. Atkeson, I. Essa, B. MacIntyre, E. Mynatt, T. Starner, and W. Newstetter, "The aware home: a living laboratory for ubiquitous computing research," presented at Second International Workshop on Cooperative Buildings (CoBuild99'), Pittsburgh, PA, 1999.

[3] A. Dey and G. Abowd, "The context toolkit: aiding the development of context aware applications," presented at Workshop on Software Engineering for Wearable and Pervasive Computing, Limerick, Ireland, 2000.

[4] "Java Media Framework (JMF)" http://java.sun.com/products/java-media/jmf/

[5] "Microsoft Direct Show" http://www.microsoft.com/Developer/PRODINFO/directx/dxm/help/ds/default.htm

[6] "RTP - The real time transport protocol" http://www.cs.columbia.edu/~hgs/rtp/

[7] D. Waddington and G. Coulson, "A multimedia component architecture," presented at 1st IEEE International Workshop on Enterprise Distributed Object Computing - EDOC 97, Surfers Paradise, Gold Cost, Australia, 1997.

[8] M. Lohse, M. Repplinger, and P. Slusallek, "An open middleware architecture for network-integrated multimedia," presented at Protocols and Systems for Interactive Distributed Multimedia Systems, Proceedings of IDMS/PROMS'2002 Joint International Workshops on Interactive Distributed Multimedia Systems / Protocols for Multimedia Systems, Coimbra, Portugal, 2002.

[9] "HAVi - Home Audio/Video Interoperability Architecture" http://www.havi.org

[10] "FireWire (IEEE 1394)," http://www.apple.com/firewire/

[11] J. Newmarch, "A networked loudspeaker," presented at Australian Unix and Open Systems User Group, Queensland, 2002.

[12] J. Newmarch and R. Kirk, "A service architecture for scalable distributed audio," presented at AWESOS 2004 - 1st Australian Workshop on Engineering Service-Oriented Systems, Melbourne, Australia, 2004.

[13] K. Arnold, B. O'Sullivan, R. W. Scheifler, J. Waldo, and A. Wollrath, The Jini specification: Addison-Wesley., 1999.

[14] "Dictionary.com" http://dictionary.reference.com.

[15] B. Schilt, N. Adams, and R. Want, "Context-aware computing applications," presented at Workshop on Mobile Computing Systems and Applications, Santa Cruz, 1994.

[16] A. Dey and G. Abowd, "Toward a better understanding of context and context-awareness," Georgia Institute of Technology, GVU Technical Report GIT-GVU-99-22.

[17] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," ACM Transactions on Information Systems, vol. 10, pp. 11, 1992.

[18] P. Brown, J. Bovey, and X. Chen, "Context-aware applications: from the laboratory to the marketplace," IEEE Personal Communications, vol. 4, pp. 7.

[19] "Future computing environments website" http://www.cc.gatech.edu/fce/

[20] S. Long, R. Kooper, G. Abowd, and C. Atkeson, "Rapid Prototyping of Mobile Context-aware Applications : The cyberguide case study," presented at 2nd ACM international Conference on Mobile Computing and Networking, New York, 1996.

[21] N. Davies, K. Mitchell, K. Cheverst, and G. Blair, "Developing a context sensitive tourist guide : some issues and experiences," presented at CHI'00, Netherlands, 2000.

[22] J. Yang, W. Yang, M. Denecke, and A. Waibel, "Smart sight: a tourist assistant system," presented at 3rd International Symposium on Wearable Computers, San Francisco, California, 1999.

[23] N. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," presented at MOBICOM, 2000.

[24] R. Orr and G. Abowd, "The smart floor: a mechanism for natural user identification and tracking," presented at CHI2000, Netherlands, 2000.

[25] A. Ward, A. Jones, A. Harter, and M. Addlesee, "A new location technique for the active office," IEEE Personal Communications, vol. 4, pp. 42-47, 1997.

[26] "Ekahau positioning engine website" http://www.ekahau.com

[27] "AeroScout" http://www.aeroscout.com

[28] P. Bahl and V. N. Padmanabhan, "RADAR: An RF based in-building user location and tracking system," presented at IEEE Infocom, 2000.

[29] "Bluetooth website" http://www.bluetooth.com/

[30] "Fraunhofer IIS website" http://www.iis.fraunhofer.de/

[31] "Ogg vorbis website" http://www.vorbis.com/

[32] "Sun/NeXT sound file format (.au)," http://www.opengroup.org/public/pubs/external/auformat.html

[33] "MPEG-7 specification" http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm

[34] "Gracenote CDDB" http://www.gracenote.com/gn_products/cddb.html

[35] "Tritonus project website" http://www.tritonus.org