UBIQUITOUS NETWORKING

Mobile computing devices have changed the way we look at computing. Laptops and personal digital assistants (PDAs) have unchained us from our desktop computers. A group of researchers at AT&T Laboratories Cambridge are preparing to put a new spin on mobile computing. In addition to taking the hardware with you, they are designing a **ubiquitous networking system** that allows your program applications to follow you wherever you go.

By using a small radio transmitter and a building full of special sensors, your desktop can be anywhere you are, not just at your workstation. At the press of a button, the computer closest to you in any room becomes your computer for as long as you need it. In addition to computers, the Cambridge researchers have designed the system to work for other devices, including phones and digital cameras. As we move closer to intelligent computers, they may begin to follow our every move.

The essence of mobile computing is that a user's applications are available, in a suitably adapted form, wherever that user goes. Within a richly equipped networked environment such as a modern office the user need not carry any equipment around; the user-interfaces of the applications themselves can follow the user as they move, using the equipment and networking resources available. We call these applications *Follow-me applications.*

Typically, a context-aware application needs to know the location of users and equipment, and the capabilities of the equipment and networking infrastructure. In this paper we describe a sensor-driven, or *sentient*, computing platform that collects environmental data, and presents that data in a form suitable for context-aware applications.

Context-Aware Application

A context-aware application is one which adapts its behaviour to a changing environment. Other examples of context-aware applications are 'construction-kit computers' which automatically build themselves by organizing a set of proximate components to act as a more complex device, and 'walk-through videophones' which automatically select streams from a range of cameras to maintain an image of a nomadic user. Typically, a context-aware application needs to know the location of users and equipment, and the capabilities of the equipment and networking infrastructure. In this paper we describe a sensor-driven, or *sentient*, computing platform that collects environmental data, and presents that data in a form suitable for context-aware applications. The platform we describe has five main components:

1. A fine-grained location system, which is used to locate and identify objects.
2. A detailed data model, which describes the essential real world entities that are involved in mobile applications.
3. A persistent distributed object system, which presents the data model in a form accessible to applications.
4. Resource monitors, which run on networked equipment and communicate status information to a centralized repository.
5. A spatial monitoring service, which enables event-based location-aware applications.

Finally, we describe an example application to show how this platform may be used.

Indoor Location Sensing

An ideal location sensor for use in *indoor* environments would possess several important properties. Not only would it provide fine-grain spatial information at a high update rate, but would it also be unobtrusive, cheap, scalable and robust. Unfortunately, the indoor environment is a challenging one in which to implement such a system. Radio-based location techniques (e.g. GPS), which are successful in the wide area, are afflicted by severe multipath effects within buildings. Electromagnetic methods suffer interference from monitors and metal structures, whilst optical systems require expensive imaging detectors, and are affected by line-of-sight problems in environments containing opaque objects. However, location systems that use ultrasonic techniques appear to have many desirable properties, and one such system that has been developed at AT&T laboratory is BAT Ultrasonic location system.

Indoor Location systems:-

1. Active Badge
2. Active Bat
3. Cricket
4. RADAR
5. Motionstar Magnetic Tracker
6. Easy Living
7. Smart Floor
8. Enhanced 911

Active Badge

Uses diffuse infrared technology - flooding an area with infra-red light .Each badge emits signal with unique id every 10 seconds that is received by a network of sensors .**Location** is symbolic – restricted area like a room .Range of several meters.Has difficulty in presence of sunlight

Active Bat

Infers **location** based on time of flight of **ultrasound** pulse.Each **bat** emits an **ultrasound** pulse with unique id to a grid of receivers .At the same instant a controller resets the receiver .Orientation is calculated by analysis. Distance is computed from the time interval between the reset and receiving the pulse. Accurate to within 3cm .Paging Requires large sensor infrastructure .

Cricket

Fixed **ultrasound** emitters and mobile receivers. Time gap to receive the signal is also set in the pulse to prevent reflected beams computation takes place at receiver .Decentralized architecture .Few centimeters of accuracy .Computational and power burden

RADAR

Based purely in software, building on standard RF wireless LAN technology .Uses signal strength and signal to noise ratio from wireless devices.Employs multiple base stations with overlapping coverage .Requires wireless LAN support on objects being tracked .Generalization to multi floored buildings is a problem

Motionstar Magnetic Tracker

Uses electromagnetic sensing. Axial DC magnetic-field pulses are generated .Position and orientation are found from by measuring the response on the three axes .Less than 1mm spatial resolution and 0.1° orientation .Must be within 1-3 meters of transmitter .Motion capture for animation

Easy Living

System to keep track of a room's occupants and devices .Uses real-time 3D cameras to provide vision positioning .Measures **location** to roughly 10 cm on the ground plane, and it maintains the identity of people based on color histograms .Difficult to maintain accuracy Aimed for a home environment .

Smart Floor

System for identifying people based on their footstep force profiles.Does not need device or tag .93% overall user recognition High cost factor.

Enhanced 911

Locates any phone that makes a 911 call .Reported in most instances with an accuracy of 100 meters or less. Can be enhanced for use by cell phone users. Identifying areas of traffic congestion .

BAT ULTRASONIC LOCATION SYSTEM

In order for a computer program to track its user, researchers had to develop a system that could locate both people and devices. The AT&T researchers came up with the **ultrasonic location system**. This location tracking system has three basic parts:

→ **Bats** - small ultrasonic transmitters worn by users.
→ **Receivers** - ultrasonic signal detectors embedded in ceiling.
→ **Central controller** - coordinates the bats and receiver chains.

Users within the system will wear a **bat,** a small device that transmits a 48-bit code to the receivers in the ceiling. Bats also have an imbedded transmitter which allows it to communicate with the central controller using a bidirectional 433-MHz radio link.

Bats are 3 inches long (7.5 cm) by 1.4 inches wide (3.5 cm) by .6 inches thick (1.5 cm), or about the size of a pager. These small devices are powered by a single 3.6-volt lithium thionyl chloride battery, which has a lifetime of six months. The devices also contain two buttons, two light-emitting diodes (LEDs) and a **piezoelectric** speaker, allowing them to be used as ubiquitous input and output devices, and a voltage monitor to check the battery status.

A bat will transmit an ultrasonic signal, which will be detected by receivers located in the ceiling approximately 4 feet (1.2 m) apart in a

square grid. There are about 720 of these receivers in the 10,000-square-foot building (929 m2) at the AT&T Labs in Cambridge. An object's location is found using **trilateration**, a position-finding technique that measures the objects distance in relation to three reference points.

If a bat needs to be located, the central controller sends the bat's ID over a radio link to the bat. The bat will detect its ID and send out an ultrasonic pulse. The central controller measures the time it took for that pulse to reach the receiver. Since the speed of sound through air is known, the position of the bat is calculated by measuring the speed at which the ultrasonic pulse reached three other sensors. This system provides a location accuracy of 1.18 inches (3 cm) throughout the Cambridge building.

By finding the position of two or more bats, the system can determine the orientation of a bat. The centralcontroller can also determine which way a person is facing by analyzing the pattern of receivers that detected the ultrasonic signal and the strength of the signal.

The receivers used to detect the ultrasonic signals rest above the tiles of a suspended ceiling (commonly found in office buildings). Receivers are placed in a square grid, 1.2m apart, and are connected by a high-speed serial network in daisy-chain fashion. The serial network is terminated by a DSP calculation board, which collects results from the receivers and uses them to compute transmitter positions.

A central controller coordinates the Bats and the receiver chains. When a Bat is to be located, the controller addresses it over the radio link, and it transmits a pulse of ultrasound at a known time. Once its

position has been found by the DSP calculation boards, the controller fuses the knowledge of which Bat was triggered and where a Bat was seen to be located, and passes the resulting location sighting to client middleware and applications. The central controller can also tell Bats when they are next likely to be addressed, allowing them to sleep in the intervals between polling messages and substantially increasing the battery lifetime.

The system has been designed to scale to very large buildings containing many mobile objects. A cellular radio architecture in which Bats hand over between different controllers is used, permitting spaces of any size to be covered. Objects can be given individual location qualities-of-service, depending on their level of mobility, thus sharing location resources fairly between them. Registration protocols allow Bats entering buildings to make their presence known to the central controllers, and allow the controllers to reclaim location resources again when Bats leave the tracking space.

In the Zone

With an ultrasonic location system in place, it's possible for any device fitted with a bat to become yours at the push of a button. Let's say the user leaves his workstation and enters another room. There's a phone in this room sitting on an unoccupied desk. That phone is now the user's phone, and all of the user's phone calls are immediately redirected to that phone. If there is already someone using that phone, the central controller recognizes that and the person using the phone maintains possession of the phone.

The central controller creates a zone around every person and object within the location system. For example, if several cameras are place in a room for videoconferences, the location system would activate the appropriate camera so that the user could be seen and move freely around the room. When all the sensors and bats are in place, they are included in a virtual map of the building. The computer uses a spatial monitor to detect if a user's zone overlaps with the zone of a device. If the zone's do overlap, then the user can become the temporary owner of the device. If the ultrasonic location system is working with virtual network computing (VNC) software, there are some additional capabilities. Computer desktops can be created that actually follow their owners anywhere with in the system. Just by approaching any computer display in the building, the bat can enable the VNC desktop to appear on that display. This is handy if you want to leave your computer to show a coworker what you've been working on. Your desktop is simply teleported from your computer to your coworker's computer.

Implementing a sentient system

Our project implemented the sentient computing system's model of the world as a set of software objects that correspond to real-world objects. Objects in the model contain up-to-date information about the locations and state of the corresponding real world objects. Ascertaining object positions with near human levels of accuracy requires a specially designed sensor system.

Location sensing

The location sensor, determines the 3D positions of objects within our building in real time. Personnel carry wireless devices known as Bats,

which can also be attached to equipment. The sensor system measures the time it takes for the ultrasonic pulses that the Bats emit to reach receivers installed in known, fixed positions. It uses these times of flight to calculate the position of each Bat and hence the position of the object it tags by triangulation. To allow accurate time-of-flight measurements, a wireless, cellular network synchronizes Bats with the ceiling receivers. Base stations simultaneously address a Bat over the wireless link and reset the receivers over a wired network. A wireless back channel supports the Bat's transmission of registration, telemetry, and control-button information.

Sensor scalability

The location sensing system described above has several features that make it suitable for wide-scale deployment in environments of interest to this work. It can provide different location update rates for different types of object, handle changing sets of objects to be located, and is scalable to both large numbers of objects and large areas of operation. The limited number of timeslots must be efficiently distributed between the set of Bats to be tracked. Each timeslot can be allocated to any Bat by the base station. A value called the *Location Quality of Service* (LQoS), associated with each object, indicates the desired interval between location updates for that object. The base station schedules timeslots to Bats based on the currently requested LQoS values. The scheduling environment is dynamic, and LQoS values associated with objects may change throughout the day. For example, the base station might normally monitor Bats carried by people (who move often) a few times a second, and it might monitor those attached to workstations only once every few minutes. If, however, a person were to walk up to a workstation, the workstation might be monitored more

frequently, because it would then be more likely to be moved. Scheduling information can also be used to assist,power saving in Bats. For example, if the base station knows that an object will not be located for some time, it can command the Bats associated with that object to temporarily, enter a low-power sleep state in which they do not check all incoming radio messages. The set of Bats to be tracked by the location system may change over time, as objects enter and leave its operating space. Mechanisms therefore exist for introducing new Bats into the set to be polled by the base station, and for deleting Bats from that set so that location resources are not wasted on uncontactable Bats. Bats outside the operating space of any base station enter a low-power *searching* state. When a Bat in the searching state locks on to the transmissions from a base station, it uses a slotted-ALOHA contention resolution protocol [7] to send its unique identifier to the base station via its radio transceiver, thus registering its presence with the base station. If, on the other hand, a base station allocates several timeslots to a registered Bat, but sees no indication from receivers that the Bat has responded by transmitting ultrasound, it can conclude that either the Bat is obscured or the Bat has left the operating space of the location system. The base station can resolve these possibilities by requesting that the Bat transmit its unique identifier via its radio transceiver—if repeated attempts to elicit a response from the Bat fail, the base station reclaims resources allocated to the Bat. The number of Bats that can be monitored by the system is determined by the size of their address space, which can be made as large as required. The area covered by the location system may be increased by the use ofmultiple base stations. A time-division multiplexing (TDM) strategy is used to ensure that transmissions from neighbouring base stations do not interfere with each other. All base stations use common timeslots derived from a global clock, and only one TDM channel is active in each timeslot. Base stations whose radio cells

overlap are allocated different TDM channels, and do not transmit in the same timeslots. The choice of a TDM strategy therefore limits the location rate within individual radio cells, but permits a simple, low-cost implementation of the Bat radio transceiver. When a Bat moves between radio cells, it must perform a handover of control between one base station and another. Systems which make handover decisions based on standard criteria such as received radio signal strength, link error rates or base station load could be developed. However, handover decisions can also be made by base stations using the known. Bat positions and coarse estimates of the extents of radio cells, which are made when the base stations are placed.

Managing the World Model

The location and resource status data are represented by a set of persistent CORBA objects implemented using omniORB, our own GPL-ed CORBA ORB. Each real-world object is represented by a corresponding CORBA software object. Around 40 different types of object are modelled, each by its own CORBA object type; for example, the system includes the types: Person, Computer, Mouse, Camera, Scanner, Printer, Phone and so on.

As well as the location of the real object, each software object makes available current properties of the real object, and provides an interface to control the real object, so, for example, a scanner can be set up and made to perform a scan by an application via its corresponding Scanner software object.

The set of all these persistent CORBA objects make up the world model that is seen by applications. The objects themselves take care of

transactions, fail-over, session management, event distribution and the all the other issues implicit in a large-scale distributed system, presenting a simple programming interface to the applications.

Programming with Space

Location data is transformed into containment relations by the spatial monitor. Objects can have one or more named spaces defined around them. A quad-tree based indexing method is used to quickly determine when spaces overlap, or when one space contains another, and applications are notified using a scalable event mechanism.

Systems infrastructure

The software counterparts of real-world entities are implemented as persistent distributed objects using CORBA and an Oracle 7 database. A package called *Ouija* [11] provides an object-oriented data modelling language which is used to generate an object layer on top of the relational model used by the Oracle database.

Objects are stored in the database as rows of data and associated operations written in PL/SQL, Oracle's proprietary procedural extension to SQL. They are accessed via a proxy server, which receives CORBA calls from client applications and forwards them to the appropriate PL/SQL operation via OCI, Oracle's proprietary call interface. This effectively provides a CORBA mapping for PL/SQL, because all queries on persistent state are performed by the database. All C++ code to implement the proxy is automatically generated by the Ouija utility. The

resulting three-tier model is shown in Figure . It is recognised that some information, particularly that from the Bat sensors, is updated too frequently to be stored in the database. This information therefore travels along a 'fast path' which goes only via the proxy. It is possible that thousands of objects might be stored in the system. Many of these may only be updated or queried infrequently, so their proxy implementation will be inactive much of the time. On-demand loading allows the proxies to operate as a cache of persistent objects. A proxy server creates an instance of an object in the database when it is first accessed. The proxy server can manage the number of active objects by unloading the implementations on a leastrecently- used basis. Subsequent calls to unloaded objects will cause them to be reloaded. This approach reduces the total level of resources required by the objects and provides system scalability. The on-demand mechanism also provides system robustness. Clients can assume that objects are always accessible

Updating the model

In order to populate the database described above, information about real world objects and their properties must be gathered. In many cases, this data is static and need only be asserted once. However, for those elements which are dynamic, for example keyboard activity or machine load, automatic update methods are necessary to maintain a current view of the environment. Processes called *resource monitors* are installed on all networked machines. The monitors use operating system calls to discover information about the current status of machines, and periodically report changes in status to objects in the database via CORBA interfaces. The monitors have been structured for portability and run on a wide range of operating systems. Furthermore, monitors are lightweight and have been designed so as not to impinge upon the

normal use of the machine or the communications infrastructure. Three classes of resource monitor have been implemented:

1. Machine activity, e.g. keyboard activity.
2. Machine resources, e.g. CPU usage, memory usage.
3. Network point-to-point bandwidth and latency.

The machine activity monitor, for example, wakes up once every five seconds. It checks the level of mouse and key board activity, and the identity of the logged on user. If there has been any change, the entry for the machine in the database is updated. Other monitors behave similarly. A centralised approach was chosen in order to reduce query latency to a minimum. If the information were not concentrated in one place, applications would need to collect data before acting upon it. This would require a number of network calls and increase the query latency. Storing the data in a centralised repository ensures that complex queries can be easily expressed using standard RDBMS technology. Much effort has been put into the design of the monitoring clients and in the communications infrastructure to ensure the database does not prove to be a bottleneck. We can apply filtering and caching techniques at the client level and in the middle tier to achieve this:

Update Frequency: The frequency atwhich items aremonitored is based on how quickly the item tends to change.

Relevancy: If a value has not changed significantly since the last time it was updated, it is not sent to the database.'Significance' depends on the data being monitored. For example, free space on a disk changing by 1% is not significant; a disk becoming 99% full is.

Caching: Data caching is used to improve client retrieval times. This can dramatically reduce query times. One application uses the database to provide a 'distributed ps', which lists all of a user's processes regardless of the machine they run on. Caching improves performance by a factor of

five. Use of filtering reduces the rate of updates to the database by over 90%. Evidently, there is a trade off between the absolute accuracy of the information, the response time and the resource overhead imposed by the monitors. In the current deployed system, monitors run on 50 machines.

Update frequencies range from five seconds for keyboard mouse activity, up to 45 seconds for disk free space. The monitors also update information about all long running processes, of which there are around 1500. The model is maintained to an acceptable level of accuracy with a transaction rate of around 70 transactions per minute.

Sentient Computing Applications

Sentient computing is a new way of thinking about user interfaces. Because the sentient computing system's world model covers the whole building, the interfaces to programs extend seamlessly throughout the building. As well as obvious applications like maps which update in real time, and computer desktops which follow their owner around, this leads to some surprising new kinds of application, like context-aware filing systems, and smart posters.

Browsing

The simplest class of sentient computing application is browsing. Because the model covers the whole building and is designed to be understandable by people, a browser allows a user to see what's going on anywhere in the building, without needing to go there.

One of our applications is this map which helps users contact each other by phone. As people move around the building, their positions are

updated in real time. If you want to talk to somebody, just type in their name and the map will zoom to their current location. Work out whether you should interrupt them by seeing who they are with, which way they are facing and whether they are moving -- you will see on the map if they are meeting a visitor, or talking to the boss. If you want to call them, just click on the telephone nearest them to set up a phone call. If they are already on the phone it will be coloured red, but as soon as they put it down it will go grey again.

Our Follow Me camera application selects a suitable camera to watch a person wherever they go. This makes applications like video conferencing more useful -- if you want to show the other participant a diagram on your whiteboard, you just walk over to the whiteboard and point to it, knowing that a suitable camera will be selected.

Context-Aware Information Retrieval

Sentient computing can help us to store and retrieve data. Whenever information is created, the system knows who created it, where they were and who they were with. This contextual metadata can support the retrieval of multimedia information.

In our system, each user has an `information hopper', which is a timeline of information which they have created. Two items of information created at the same time will be in the same place in the timeline -- this allows us to associate data items in a composable way, without having to maintain referential integrity. The system knows who the user was with at any point on the timeline, and the timelines of users who were working together can be merged to generate another timeline. This lets us generate records of collaborative work without any maintenance effort, by

using the sentient computing system as a kind of ubiquitous filing clerk. The timeline can be browsed using a normal web browser.

Smart Posters

Sentient computing system creates an interface that extends throughout the environment, we can treat it just like a traditional user interface and create a `button' anywhere in the environment. In a traditional GUI, a button is just an area of the screen, which usually (but not necessarily) has a label associated with it. In a sentient computing system, a button can be a small space anywhere in a building -- again, it may have a label associated with it. Of course, the label need be nothing more than a piece of paper. To press the button, the user just puts the bat on the label and clicks a button on the bat. As a bonus, because the system knows which bat was used, it knows who pressed the button.

We can create a poster with several of these buttons on it -- the poster is a user interface which can be printed out and stuck on the wall.

Ubiquitous user interfaces

We can control other devices using located tags like the bat. We have implemented a video streaming and camera control system using networked MPEG codecs and pan-tilt-zoom controllable cameras.

One of our applications of this is a distributed video bulletin board which lets users create video messages, and organise them into threads, controlling a camera by using a bat as a pointing device. Users can

control the camera's pan, tilt and zoom by pointing at things with their bat, or by wearing their bat, letting the camera track them as they move around. The bulletin board uses standard networked MPEG-1 codec boxes which transmit via IP multicast, so it could also be made to support recording of n-way video conferences.

Resource Ownership

Bats are small enough to be attachable to most small portable devices. Because it knows exactly where people and devices are, the sentient computing system can work out who is using a device at any time. We have implemented two applications. Using a standard digital camera which is located using a bat, a user can take photographs which are automatically filed in their information hopper. And using an digital audio recorder a user can make audio memos which are also automatically filed in their information hopper, together with a textual transcription of what they said. Because we know who was holding the audio memo recorder we can increase the quality of the text transcription by using a voice model and vocabulary appropriate to that user.

Over the next few years we expect wireless devices and LANs to become more widespread. But without a sentient computing system, the value of a wireless device is limited. There is a widespread assumption that radio devices themselves have some kind of innate sensing capability because useful proximity information can be inferred from radio signal strength. This assumption is incorrect, firstly because multipath effects within buildings greatly distort the relationship between signal strength and distance, and secondly because it fails to take account of environmental discontinuities like walls and floors.

CONCLUSIONS

We have presented a sentient computing infrastructure for implementing context-aware systems. Key features of the infrastructure include:

- A fine-grained sensor system which provides accurate, up-to-date location information. The finer granularity of this system in comparison to the Active Badge enables us to providemore context information to applications, with consequent benefits to the user interface.
- A rich data model reflecting the resource information required to support context-aware applications.
- A distributed system of persistent objects which can be queried by context-aware applications. This presents a highly available world model which applications can easily utilise.
- A resource monitoring system for collecting information about the computing environment. This enables applications to adjust their behaviour to accommodate system capabilities and usage patterns.
- A spatial monitoring system which allows event-based applications to be written. This enables application writers to write location-aware applications in the same event-driven style as traditional GUI applications.

These components have been integrated to support a Follow me application, Bat Teleporting, which improves upon a previously-developed system in several ways. The supporting infrastructure is robust, scalable and simultaneously usable by a large number of different applications, several of which have been prototyped. Work is underway to develop these prototypes into large-scale, fully-deployed services.

REFERENCES

- http://www.uk.research.att.com/spirit

- A New Location Technique for the Active Office. IEEE Personal Communications, Vol. 4, No. 5, October 1997, pp. 42-47.

- Sensor-driven Computing. PhD thesis, University of Cambridge, 1998.

- The Anatomy of a Context-Aware Application. Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking, MOBICOM'99, Seattle, Washington, USA, August 1999, pp. 59-68.

ABSTRACT

The ultrasonic location system is based on the principle of trilateration position finding by measurement of distances (the better-known principle of triangulation refers to position finding by measurement of angles). A short pulse of ultrasound is emitted from a transmitter (a Bat) attached to the object to be located, and we measure the times-of-flight of the pulse to receivers mounted at known points on the ceiling. The speed of sound in air is known, so we can calculate the distances from the Bat to each receiver - given three or more such distances, we have enough information to determine the 3D position of the Bat (and hence that of the object on which it is mounted).

By finding the relative positions of two or more Bats attached to an object, we can calculate its orientation. Furthermore, we can deduce some information about the direction in which a person is facing, even if they carry only a single Bat, by analysis of the pattern of receivers that detected ultrasonic signals from that transmitter, and the strength of signal they detected.

ACKNOWLEDGEMENT

First and foremost of all I thank the Almighty God who made possible this venture.

I take this opportunity to express my heartfelt gratitude and thanks to Mr. Zainul Abid (Staff incharge), Computer Science and Engineering, MESCE for giving me an opportunity to take this seminar which has been an enlighting learning experience to me. I am indebted to Prof. M.N Agnisarman Namboothiri (Head of the Department, Computer Science and Engineering, MESCE), for his valuable comments, supports and suggestions.

Finally my profound thanks to my friends and well-wishers who helped me at every turn through their valuable suggestion.

Shaheen.P.H

CONTENTS