The Royal Society Clifford Paterson Lecture, 1999
Sentient Computing

By Andy Hopper
University of Cambridge
and
AT&T Laboratories Cambridge

Sentient computing is the proposition that applications can be made more responsive and useful by observing and reacting to the physical world. It is particularly attractive in a world of mobile users and computers.

The paper presents a classification and quantification of sensor information together with a description of a method for altering the behaviour of arbitrary terminal devices. It also presents a framework for "programming with space" which can associate space-related events with actions. Consideration is given to the applications made possible by such systems.

Sentient computing

Computers have become integral to our lives, but for many the gap between man and machine is so large as to be effectively unbridgeable. Central to any good working relationship is a degree of mutual understanding between the two parties. The problem with conventional human-computer interaction is that responsibility for understanding, or the lack of it, lies wholly with the user.

The office PC illustrates this. Outwardly it seems different from earlier models of computing, such as time-sharing systems or even the mainframe. But, take away the mouse-driven user interface and the fundamentals remain unchanged. The burden for understanding lies wholly with the user. Not so long ago computers came with a team of dedicated operators. In the PC era everyone is his or her own operator. No wonder so many PCs are only used to a fraction of their potential. It does not have to be this way.

Instead of bringing the user to the computer, let us take into account that people live and work in a real physical world, and make this notion – the concept of space – integral to the operation of our computer systems. We need to make computer systems aware of the physical environment – shifting part of the onus of understanding from user to machine. Awareness comes through sensing, and that implies the need for appropriate sensor technologies to collect status and location data. Applications can now be aware of their physical environment. They know where people and devices are and what devices

can do. Crucially, the user interface is no longer based on some abstract metaphor of a physical object, but can be based on space itself. So, when I walk into my study at home, my PC seamlessly and automatically displays the desktop from my office machine – my proximity to my home PC is the prompt to the user interface. I pick up a CD cartridge in my office, and as I open it the appropriate sound track immediately starts to play. Again, a real, intuitive physical action initiates an appropriate response, made possible by an underlying computer system in which location and status data extends throughout the physical environment. I call this approach *sentient computing*.

The ultimate justification and test of sentient computing will be its capacity to deliver benefits to users, enabling them to interface directly to devices and expressing complex configuration requirements in a simple way. Reaching this goal depends on our capacity to address a broad spectrum of conceptual and technical challenges.

The central requirement in any computer application is the need to achieve the right conceptual mapping between the physical and the logical. Applications are about physical things – people, PCs, telephones, printers, whatever. A computer program is ultimately a logical abstraction, and the art of the system designer lies in bridging the conceptual gulf between these two radically different domains. The first challenge in sentient computing is to determine the appropriate meeting point between the physical notation of space and the logical constructs of our computer system. Do we do it at the ad hoc, application-specific level? That would work, but at the expense of programmers constantly 're-inventing the wheel'. Different sentient computing applications will share common features and attributes, suggesting a systems-level approach might be more appropriate, with standardised support for programs that have to capture and express the concept of space. Alternatively, we could make space an integral part of the programming language. But that would necessitate the creation of libraries for representing standardised 3D objects. Any library comprehensive enough to be universally applicable would, almost certainly, be over specified for the great majority of applications.

Underlying all this, are twin problems of computational efficiency and performance. Our logical representation of space has to be appropriate to the application in hand. But what do we mean by appropriate? Too exact a representation will make the task of maintaining our store of spatial data difficult. We want systems that react to our actions with no perceptible delay, necessitating the updating of spatial information in real time, or near-real time. So how should we associate spatial properties with things? Do we have to use a 3D representation? Under what circumstances would a 2D representation suffice? Can we use regions around things? Above all, what are the basic properties of things and the logical constructs necessary to our computing models that will turn the vision of sentient computing into an everyday reality?

The sentient computing project at AT&T Laboratories Cambridge and the University of Cambridge is an experimental programme designed to provide answers to these questions. The work of the programme is structured around three basic themes: sensors (that tell us about the spatial properties of objects), devices (the PCs, printers, and other output devices used in our applications), and the platforms (that connect sensors and devices together). Surrounding these three basic elements we have the appropriate architecture that gathers all the elements into a complete functioning system (Fig. 1).

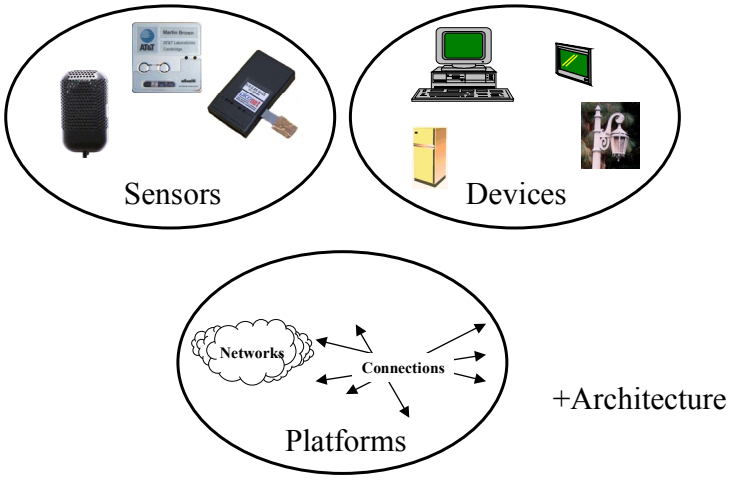


Fig 1 Components for Programming with Space

Sensors

Sensors tell us about the location or position of things. To reflect the requirements of different applications, we take three different approaches to categorising the concept of position. First, there is containment, where we say that an object is within this container, e.g. a room, so that the application could register the fact that I'm in my office or my study at home. Secondly, there is proximity, where we register that we are close to something, and finally we use co-ordinate systems, which provide a point location in space, subject to some error value. These categories are not hard and fast and can blend together. Small containers are very similar to a co-ordinate system, and proximity has much in common with the concept of containment.

Existing systems not primarily designed as sensors can generate a valuable amount of spatial information. In the case of containment, a satellite telephony system might provide an initial containment within one twentieth of the world, which, depending on the number of antennas on the satellite, can then be further partitioned into up to 50 subsections. That is still very course granularity. With GSM, the digital mobile phone system, we can do significantly better, at the expense of worldwide coverage, realising a container some 25-km across. Third generation cellular systems, such as UMTS, will offer a very similar performance. Indoor wireless LANs give us even finer granularity and provide a container about 30m in diameter.

Our first experience of developing a sensor specifically to provide spatial information originated in the early 1990s in the form of the Active Badge (Fig. 2). Personnel and equipment can be tagged using the badge, which transmits a unique infrared signal every few seconds. The transmissions are diffuse and receivers in a room pick up the signal, so the badge gives room-scale containment. It tells us who and what is where, and the software system which makes this information available to others, is still

Infra-Red Network

10 meter range

diffuse

room-scale location

Fig 2 Containment: Active Badge

very popular. The Active Badge has been the inspiration that got us started on this whole line of enquiry.

In the case of proximity - allowing, for example, a laptop and a telephone to exchange short dialling codes - promising commercial systems are starting to appear. The radio-based Bluetooth system offers a range of round 10 m, while for the infra-red based IrDA the range is more like 3 m. We have built our own wireless-based proximity system, which we call PICOnet.

PICOnet is envisaged as the minimalist building block of our system – the simplest of proximity sensors for the simplest of nodes. It appears to be always on, but uses little power, so the batteries may never have to be changed. Our PICOnet radio operates at 418 MHz, gives a data rate of approximately 5 kb/s, and has a range of around 5 meters depending on the antenna design. There are two modes of operation. In the basic mode, the PICOnet node operates as a beacon, i.e. transmit only. The node has a very low power timing circuit, so it can count time with a minimal expenditure of energy. Then at regular intervals, it transmits a very short message, ending the message with an announcement of the time of the next transmission. A slightly more complex variant combines a transmitter and a receiver. This node also transmits at regular intervals, but after every transmission it starts listening for a short period, and then it shuts down. The challenge is to exploit the facilities offered by these simple operational modes in order to effect the seamless interworking of devices in close proximity.

There are three issues we have to deal with: discovery, description and communication. Consider a situation where there are billions of PICOnet devices all over the world. They are mostly inactive, but nevertheless, should by chance two nodes happen to pass, then they have to wake up and register each other's presence. This fundamental discovery problem can be addressed in a number of ways. Probably the simplest is the bilateral rendezvous, where the node that operates as a receiver/transmitter

switches on its receiver and listens for a possible transmission from an adjacent node. Once a transmission is detected, the time for the next transmission will be known, and the two nodes can then operate in a deterministic manner. Another possibility is third-party rendezvous where a node is held permanently in receiver mode, possibly drawing power from a plentiful source. This node acts as a source of information on all local transmissions, and can therefore facilitate the discovery process for other node-pairs in its neighbourhood.

All the nodes in a PICOnet system are completely general purpose, with every node responsible for describing its services and requirements to the rest of the world. This description function is provided by a node's attribute store.

For communication between PICOnet nodes we can use an attribute store as a sort of bulletin board whereby node A posts messages to node B. There is no support for hop-by-hop routing which is the simplest way of maintaining the objective of location by proximity. If node A receives a message from node B, and node B describes itself as a fan, then node A knows it is close to a fan – there is no other way it could have got the original message.

Effecting a consistently reliable rendezvous between nodes, which spend most of their time in a deep sleep mode, remains a fundamental problem. Another challenge is how to attach such systems to those that can only operate by placing much stricter timing constraints on communicating devices, for example the Internet Protocol.

We have built a whole series of PICOnet-enabled devices (Fig. 3) to help us understand issues like the functioning of a complete system and the interoperation of PICOnet with other systems. My CD cartridge demo uses PICOnet. There is a PICOnet node in every cartridge. You pick it up, you open it and the right music starts to play from nearby speakers. It is simple, easy to grasp, and users understand it immediately.

Co-ordinate systems provide our final approach to categorising position. Outside the Global Positioning System (GPS) can be used, which, when used in combination with maps, has given rise to a large number of applications. GPS gives an error value of around 30 meters most of the time, although greater precision can be achieved. At the

Fig 3 Proximity: PICOnet Devices

Cambridge Laboratories we have been working on a co-ordinate system for indoors. This uses a tag, which incorporates ultrasonic transmitters and an array of ceiling-mounted detectors. A detector on the far side of the room will register a pulse later than a detector directly above an object. Using this differential timing information, we can calculate the position of objects to within a few centimetres almost all the time. Bats find their way around using much the same principle, so we call our system the Active Bat (Fig. 4). Fix two transmitters on a rigid object and you can work out its orientation. The Active Bat is a very versatile system; clearly there will be many sentient computing applications that do not require this level of precision and refinement. However, as a research tool, it is providing us with valuable information on what can be done when you have very detailed in positional data.

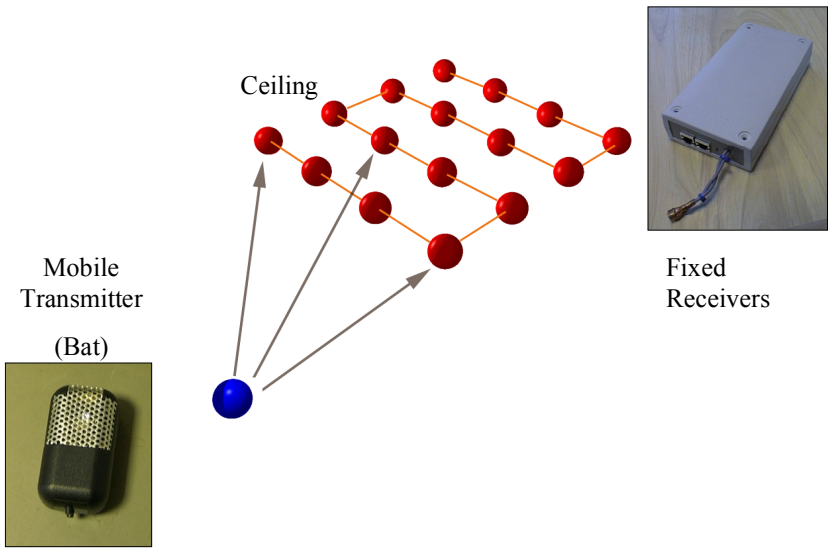


Fig 4 Co-ordinate: Active Bat

Devices and platforms

In sentient computing, a device can be anything that takes output from the distributed computing environment. Naturally, this includes conventional computing devices like workstations, PCs or the various forms of personal digital assistant (PDA), but it also embraces consumer products like refrigerators and microwaves, and new devices into the future. We need a platform for connecting and displaying on all these interesting devices in a ubiquitous way.

One way to do this is to tunnel connections to all devices using a simple device-independent protocol. We have devised one such ubiquitous platform called the Virtual Network Computer (VNC). In our approach the viewer, at the receiving end of the connection, has no state, it is just something that visually displays information. All the processing is centralised on a server at the sending end of the connection. Because the viewer has no state, it does not matter if it crashes. The application carries on running,

and the user can simply switch to another display device. The other direction, viewer to server, is also stateless – it is just key strokes and clicks - making our viewer a particularly simple version of the so-called thin client (Fig. 5.)

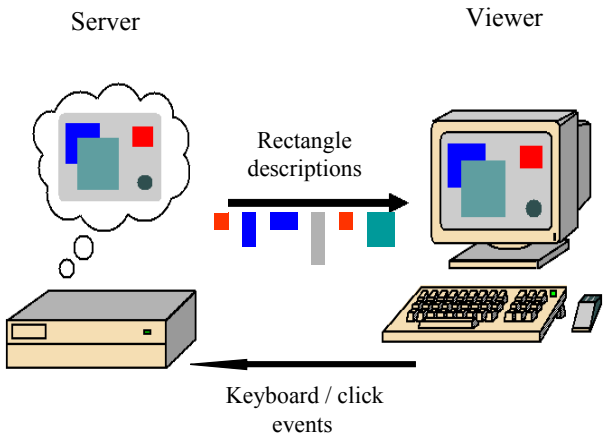


Fig 5 VNC - The Platform

The absence of state eliminates any requirement for synchronisation. You can leave your desk, go to another machine, whether next door or on the other side of the world, reconnect to your desktop and finish the sentence you were typing. Even the cursor will be in the same place. The appearance is of total mobility, although all we are doing is showing a display in different locations.

The technology underlying the VNC is a version of the remote frame buffer protocol. At the server end of the connection everything we want to display is decomposed into a series of rectangles, with every rectangle characterised by its size, colour and position on the screen. The rectangle descriptions are sent to the viewer, which recreates the original image by redisplaying the individual rectangles. As the viewer requests the next set of updates the protocol can cope with servers and clients of varying speeds. It is a bit like the old character-based dumb terminal, only now we are displaying rectangles rather than characters.

The low-level nature of the protocol is the key to device independence, providing a platform that supports the connection of any device to anything (Fig. 6). The connections can be one-to-one (fixed or mobile), and the streams can be split giving one-to-many, many-to-one, and many-to-many.

There is a potential difficulty associated with this model of stateless viewers in which everything is potentially connected to everything else. We have already established that timing constraints mean that there is a fundamental problem in effecting a rendezvous, or connection, between pairs of PICOnet nodes. Now we are postulating a model of computing built on the premise of universal interconnection.

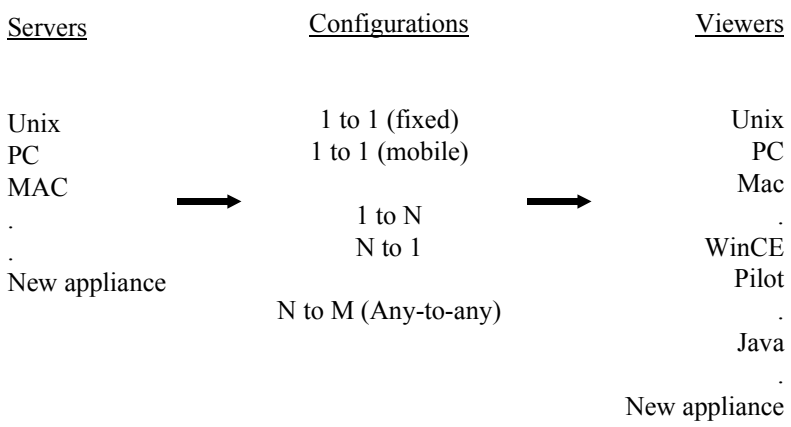


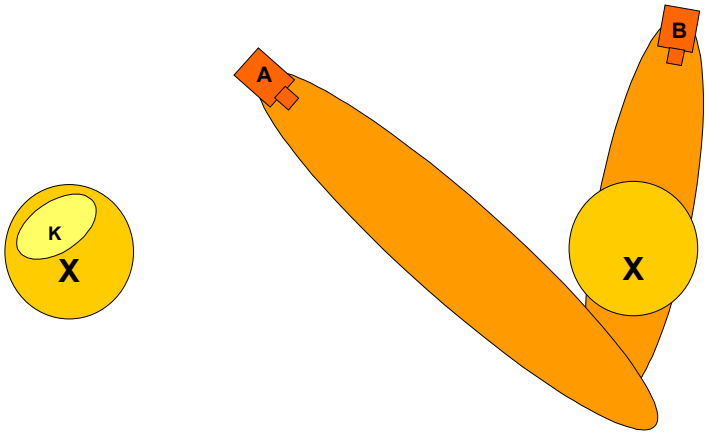
Fig 6 VNC - Configurations

Architecture

We have sensors generating a wealth of location information; we have devices and a platform for connecting to any device. Now we need to glue everything together, providing our applications with suitable abstractions to support space-aware programming.

Our sensors provide raw spatial facts about objects. They tell us where an object is, and, possibly, the direction in which an object is pointing. Location-aware applications need more than raw spatial data, they need to be notified of spatial relationships between objects that are significant in terms of advancing the execution of the application. But how do we decide whether a spatial relationship is significant? The approach we have adopted operates on the basis of zones of containment surrounding objects. In Figure 7 (image on the left) X represents a person and K a keyboard. Now suppose we have an application that needs to be notified when person X is in a position to use keyboard K – when X is possibly 'holding' K. If the zone of confinement of K overlaps the zone of confinement of X, then the holding condition is held to be true and the application receives the appropriate trigger. The situation on the right of Figure 7 indicates how this principle could be applied to support a multi-camera video conferencing system, giving participants the freedom to look in different directions while talking, or even walk around their offices.

Note Figure 7 is a 2D representation of what in reality would be a 3D environment. This simplification can be made because, in general, people and objects tend to remain relatively fixed in the vertical plane. However, the principle can be extended to 3D if required.



Person **X** is "holding" keyboard **K**

Person **X** can be "seen" by camera **B** but not by camera **A**

Fig 7 Evaluating Spatial Facts

The principle of turning raw spatial data into application-significant events through geometric containment and overlapping is reasonably straightforward. You can think of it as the mouse/desktop metaphor mapped onto the real world. However, once you start thinking about real applications, with a population possibly comprising hundreds of thousands of objects, then there is the problem of how to implement this principle in a computationally efficient manner. Every time an object moves, a calculation must be done to identify possibly significant overlaps and send the appropriate application callbacks. In a realistically sized system, there could easily be a large number of object moves every second. It is thus necessary to represent the containment regions with flexibility of precision together with reference counts of how applications have registered interest at a particular level.

Now we can put our architecture together to see how it supports applications. It starts with the sensor events, which are related to the movement of real objects. Applications register the set of objects in which they have a particular interest, and are fed callbacks indicating the occurrence of significant spatial relationships between objects. These callbacks are generated via geometric containment and overlap. When an application receives a callback, it executes an appropriate action as specified by the application program.

The operational system that has been built uses a variety of sensors; allows space representations to change quickly; provides an appropriate governing event logic; uses caches and proxies to handle large volumes of data quickly; and executes in real time to satisfy a human in the loop.

Applications and the future

Applications are the mechanism through which we can test the principles underlying sentient computing. The automatic generation of my office desktop on my

home PC is made possible by a variant of an application we call the 'follow-me desktop'. This uses an Active Bat desktop sensor, with the ability to register significant spatial relationships between the desktop and the viewer, i.e. it can determine whether the viewer is facing the desktop and vice versa. Once this containment overlap has registered, then the application tunnels the user's desktop onto the new device, be that a workstation, a PC, the refrigerator door, or some new device yet to be invented. The platform that makes this possible is the VNC technology, with its capacity to re-route whatever desktop, to wherever you like and then display it on whatever you like. I do not have the Active Bat system in my home, but I do have the Active Badge, and I can use this to register my co-location with my home PC. Four steps take me across my study to my desk, and in this time my desktop is on my screen ready for me to start work.

Much of the potential attractions of video conferencing can be undermined by the need for speakers to address a single camera throughout the duration of the call. It may seem unnatural, people want to feel free to look around, maybe even get up and walk around their room. A possible solution is to have multiple cameras in a room, combined with some technique for determining which camera to use at any instant. Machine vision and scene analysis is one of the most difficult and challenging research areas so using such technologies is unlikely to provide the same level of robustness as a tagged system. The 'follow-me video phone' uses a sensor, the Active Bat, and a display device that is fast enough to provide a VNC moving video image.

So where is this entire research theme heading? Initial applications of sentient computing will almost certainly be within vertical markets. It is possible PICOnet-based guidebooks will enrich our visits to museums and art galleries; while the VNC-based follow-me desktop has obvious attractions as a means of distributing personal desktops throughout a closed working environment such as the hospital or factory. However, it would be surprising, and not a little disappointing, if the long-term role of sentient computing was confined to such geographically restricted and application-specific domains.

We live in a world in which computing, and the technology to interconnect computers, becomes cheaper year by year. In due course, it is likely that there will be hundreds of communicating devices for each one of us. How then will all these devices be administered? How will they interoperate? And how will they be personalised to that we know how to use them? It may be this will be done automatically, through a process in which physical information about the position of objects is likely to be as important as logical information about their relationship. In short, programming with space – possibly the key to ubiquitous, pervasive, sentient computing and the communications world of tomorrow.

I thank the following who have been in involved in the sentient computing project or have otherwise helped me with this lecture: MD Addlesee, F Bennett, DJ Clarke, R Dettmer, AC Harter, SE Hodges, AH Jones, TJ Richardson, Q Stafford-Fraser, PJ Steggles, AMR Ward, MV Wilkes, KR Wood.

References can be found at: www.uk.research.att.com/~hopper/publications.html