

Abstract

Piconet is a general-purpose, low-power ad hoc radio network. It provides a base level of connectivity to even the simplest of sensing and computing objects. It is our intention that a full range of portable and embedded devices may make use of this connectivity. This article outlines the Piconet system, under development at the Olivetti and Oracle Research Laboratory (ORL). The authors discuss the motivation for providing this low-level "embedded networking," and describe their experiences of building such a system. The article concludes with a commentary on some of the implications that power saving, and other considerations central to Piconet, have on the design of the system.

Piconet: Embedded Mobile Networking

Frazer Bennett and David Clarke,
The Olivetti and Oracle Research Laboratory
Joseph B. Evans and Andy Hopper,
The Olivetti and Oracle Research Laboratory
and University of Cambridge Computer Laboratory
Alan Jones, The Olivetti and Oracle Research Laboratory
David Leask, University of Cambridge Computer Laboratory

There is a great divide in mobile computing between what is desirable and what is practical. This divide is inherent and caused by, among other things, constraints in size and power as well as the lack of a reliable network connection. All these compound to make the mobile computing environment a harsh one. None of this, however, has prevented the proliferation of mobile computers, from laptops and personal digital assistants (PDAs) to very small, simple, embedded computing devices that may go almost completely unnoticed. Indeed, the extent to which these devices are embedded means we are already at the stage where people are unaware of how many computers they may use in a day.

The integration of this vast array of mobile and embedded computing objects is now the challenge. The prospect is one of a seamlessly orchestrated computing and communications infrastructure.

Clearly, there is a large variation in the communication requirements of these very different devices. However, it is our opinion that there must exist, at the very least, a "base level" of connectivity between things. This should be available to even the simplest of embedded sensing and computing objects. By providing just a small amount of wireless connectivity through which communication is possible, we make possible large numbers of new applications. The provision of such connectivity is what we call *embedded networking*.

This article describes the Piconet project underway at the Olivetti and Oracle Research Laboratory (ORL). Piconet is an attempt to understand the implications of the provision of wireless connectivity at the level described here. We recognize that to do this effectively we must build, deploy, and use a system that demonstrates these concepts. This is what we have done.

Joseph B. Evans was visiting from the University of Kansas Information and Telecommunications Technology Center, Lawrence, KS, when this work was done.

Embedded Mobile Networking

Embedded networking concerns the provision of a network which is so simple and small that it can be used by almost anything. Through embedded networks we would like everyday objects to be able to communicate in a way that has not yet been achieved. Sensors which can monitor and control the environment; telephones, fax machines, photocopiers, printers, portable computers, and PDAs; electronic access control to buildings and roads; banking and public information terminals — many of these already need a network in order to operate, but all would benefit from a common mechanism by which they are made aware of, and can communicate with, other things nearby.

The Piconet project at ORL is developing a prototype embedded network. Piconet is a low-rate, low-range, ad hoc radio network. We have developed a Piconet *node* that can be used to provide a connection to this embedded network. Piconet provides a broad range of mobile and embedded computing objects with the ability to exploit an awareness of, and connectivity to, their environment.

Sensors can use Piconet to relay information about the state of the local environment or of a particular device. Personal connectivity is improved because the multitude of mobile and fixed devices used by an individual in a day can be connected by Piconet; it might be used to personalize things nearby or allow two devices near each other to interoperate. Embedded networking is also suitable for smart information services: active diaries, alarms, information points, and electronic business cards, for example. The *proximate connectivity* that Piconet provides means these applications can be *context-aware* [1].

Technology Characteristics

The kind of applications we hope to make possible with an embedded network like Piconet impose certain constraints on the technology used to build it. Primarily the network must be low-powered, simple, and ubiquitous. It must be of reasonably short range to allow proximity to be inferred from connectivity.

Ubiquitous and Simple

The simplest device that might be connected by Piconet is a binary switch. Perhaps all that it would do is to periodically convey its state over the wireless channel. Other mobile devices connected with Piconet can interrogate the switch for its state, and discover what the switch's state implies. To make this possible, Piconet must be extremely simple and very low-powered. The wireless medium must be functional under a wide variety of conditions: indoors and outdoors, exposed and embedded, line-of-sight and diffuse. The communications protocols employed must impose very little overhead. Common mechanisms must exist by which devices can describe themselves to the world so that other devices can discover them, understand what they are, and interact with them.

Low-Power, Low-Rate, Low-Range

The requirement for low power has implications at every level of the system's design. As well as choosing low-power components, we must adopt protocols that allow a device's network interface to be switched off much of the time. The need for only a low-rate connection between devices makes this easier, since we do not need the same level of complexity inherent in the design of higher-speed networks.

The low range of Piconet has the advantage of providing information about proximity. If two devices can communicate over Piconet, by implication they are near each other. This proximity information makes context-aware applications and *personalization* possible.

Radio for Embedded Networking

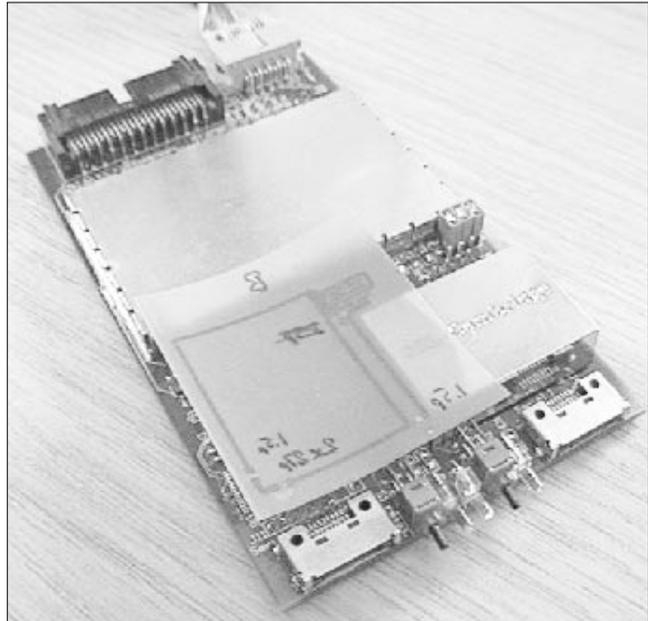
Radio is the technology used for communications in Piconet. Radio possesses the characteristics needed for ad hoc, peer-to-peer communications in virtually all configurations and environments. In order to support our model of interaction among Piconet nodes, communication must be unrestricted; that is, nodes must be able to communicate when in range, even if they are being carried in a briefcase, coat pocket, or car boot, indoors or out. Although infrared has advantages over radio such as smaller component size, lower cost, and power consumption, much of this is attributable to the maturity of infrared technology and standardization activities such as those by the Infrared Data Association (IrDA) [2]. The line-of-sight requirements of infrared, as well as the difficulties in using infrared outdoors, restrict its flexibility. It is our opinion that a future ubiquitous embedded network will have to use radio as a communications medium for the reasons cited here.

Piconet System Design

In developing a prototype Piconet node, we have compromised on size and power considerations in favor of a simple and flexible design. We want to experience what Piconet might do for us, so speed and ease of prototyping are what is important.

Our Piconet node is composed of a 418 MHz FM transceiver, an FPGA that drives the radio physical layer and provides media access control (MAC) support, and a microcontroller with a runtime environment. Figure 1 shows a prototype node that measures 127 x 74mm. RF screening cans cover most of the board. The lower lefthand quarter of the board is covered by a thin patch antenna. As well as the radio, a node incorporates two serial ports and a parallel "expansion" port as external interfaces. Through these, it is able to connect to many different types of device, giving us scope for using Piconet in a variety of applications.

Piconet's *runtime* environment allows rapid and automatic configuration of a node for a particular task. This is done at



■ Figure 1. A prototype Piconet node.

power-up by booting code and data into a node through any of its interfaces. This environment is flexible and extensible, while still respecting the fact that Piconet is essentially an embedded system. Other components include *protocol drivers* to provide various radio transport protocols and, more interestingly, an *attribute store*. The attribute store acts as both a naming and a resource description facility within each node, as well as being a more general mechanism by which nodes can convey information to each other.

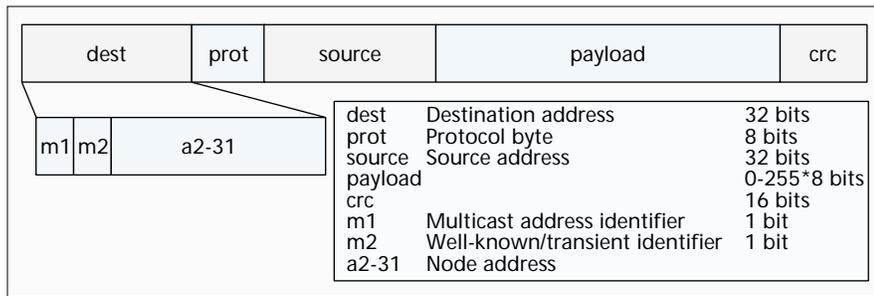
Piconet Radio

The physical range of the radios used by Piconet is constrained to around 5 m. There are several reasons for this. First, it allows us to use radios that are small, low-powered, and cheap. Second, a small spatial cell size allows greater reuse of the radio channel, thus increasing the aggregate bandwidth available. Finally, it parameterizes the system to work at *human ranges*. By this we mean that Piconet enables communication between objects within a human's immediate surroundings. Those things that are near somebody — within their *local context* — are things that can now be connected together for them by Piconet.

If one node can contact another over Piconet, it is close enough to be of use. If a few nodes are near each other as they move around, perhaps being carried by someone, they should be able to spontaneously intercommunicate.

Radio Protocols — In choosing suitable radio protocols for Piconet, we had to particularly consider its ad hoc and low-power nature. We need to support the intermittent connectivity of a continuously mobile and varying selection of nodes, each of which may only need to communicate just a simple amount of information very infrequently.

The fact that Piconet is ad hoc imposes inefficiencies on our choice of MAC protocols, as do the specific characteristics of the radios that we have used in our first prototypes. There is no base station to arbitrate communication between nodes, and the dynamic nomination of such a base station is complicated by the extreme mobility inherent in the system. As explained earlier, we expect many nodes to continuously drift in and out of contact with each other. These characteristics distinguish Piconet quite significantly from those systems outlined in [3, 4].



■ Figure 2. Piconet datagram format.

Our initial protocols are oriented toward short-lived transactions between nodes rather than long-lived streams of data. Furthermore, we must exploit the broadcast nature of the medium by supporting multicast communication. We want our protocol to be very simple. If Piconet is going to be useful to the very simplest of embedded sensing objects, elaborate protocol overheads cannot be afforded. Finally, we want to instrument radio protocols in such a way that we can gain useful information about proximity and link quality.

Existing ad hoc radio protocols include the increasingly popular IEEE 802.11 standard. IEEE 802.11 and similar protocols were not used in Piconet for several reasons. The characteristics required by the Piconet radios are simplicity of implementation and support for a very-low-rate physical layer. The former property is necessary because the intended target devices need to be extremely small and inexpensive. We would not wish such nodes to be burdened by the complex physical layers required for high bit rates, or the protocols necessary for sharing the medium among high-availability or stream-based services.

Piconet's Radio Protocol – The low-level radio protocol used by Piconet reflects the specific qualities of the radio being used. A data preamble long enough to support transmitter warmup and receiver settle time is essential, and a DC balanced encoding scheme is used. This scheme involves 4b6b encoded FM keying. It involves representing every 4-bit value as a unique 6-bit code. Each 6-bit code used has the property of containing three 1s and three 0s. This encoding is necessary to ensure that a radio's receiver does not drift from the FM signal. In addition, 4b6b encoding introduces some code redundancy, which is valuable for error detection.

The radio transceiver provides 40 kbaud, which, with the encoding overhead, gives us 28.8 kb/s. Additional overheads from link-layer and MAC functions leave us with an estimated 9600 b/s data rate, half-duplex. It is worth emphasizing that while higher bandwidth may well be desirable, this is not an issue for the first prototypes of Piconet since many new applications are made possible with even the very smallest amount of connectivity.

Our link-layer protocol uses 32-bit node addresses, and supports addressing for multicast groups. All datagrams include full source and destination addresses. The format of a datagram is outlined in Fig. 2. As well as the destination and source node addresses, the datagram header contains a *protocol byte* and *payload length* indicator. The protocol byte is used internally within a node to determine how data is to be handled. The payload may contain up to 255 bytes of data. This short packet size means that encoding and decoding for transmission is kept simple, since we only need to manage 8-bit counters inside the node. In addition, smaller packets result in better sharing of the radio channel, since no single device is transmitting for too long.

Piconet supports two types of multicast traffic in the form of *well-known* and *transient* multicast groups. As indicated in Fig. 2, the top two bits of a node address are used to identify

multicast groups. *Well-known* multicast addresses are used for predefined purposes, and each of these is universally recognized by any node that may want to make use of it. The *radio boot plea*, described later, uses a well-known multicast address.

Transient multicast addresses allow dynamic creation of multicast groups — perhaps for a temporary collaboration or sharing between nodes. When a transient multicast group is created, a random 30-bit address is nominated by one node for use by the group. Without any arbitration in the allocation of these addresses, we rely on the statistical improbability of an address clash in both time and space to avoid conflicts. We choose this address allocation strategy in order to preserve the strictly ad hoc nature of Piconet.

We use a CRC-16 at the tail of every datagram. With 4b6b encoding, however, we only use one quarter of all valid 6-bit codes, and it is much more likely that errors are interpreted as coding violations than cyclic redundancy check errors.

Piconet's MAC protocol is an extension of 1-DSMA,¹ providing support for multicast, unacknowledged, data. In order to establish that the channel is busy, Piconet relies on the detection of a valid preamble. This represents an inefficiency in the implementation of the MAC protocol, but radios with better channel measurement facilities will improve this.

Multicast traffic is supported in the MAC protocol by the addition of a *multicast backoff timer*. This timer is used to ensure that the channel is not swamped by multicast traffic which, since it is not acknowledged, is not subject to backoff imposed by transmission failure. When a node has transmitted a multicast packet, it must refrain from transmitting another for a fixed time measured by the multicast backoff timer.

Piconet Runtime

In designing Piconet's system-level components, we wanted to make it easy to interface to the many different types of device that will use it. These range from simple sensors, which may require analog, digital, or serial input, to PDAs, which may want to make more elaborate use of Piconet's facilities. Each must have a common mechanism for their use of the radio channel, particularly device discovery and description. In addition, we wanted it to be easy to configure a node for a particular task by booting a node with application code.

Another significant factor in the design of Piconet's runtime is power-saving. Mobile Piconet nodes need to save power wherever possible. This involves powering down both the processor and the radio interface most of the time.

The two main components of Piconet's runtime are the kernel and the loader. The kernel represents the environment that directs the operation of a node, and provides drivers for the node's interfaces. Applications are downloaded into a node via the loader, and run within the context of the kernel. The kernel exists in ROM, while external RAM (up to 0.5 Mbytes) is available for applications. An application is a piece of code used to configure a node to perform a specific task (e.g., to act as an interface to a sensor connected to the node).

The Piconet Kernel – The kernel has a simple multithreaded design. Within the kernel, threads communicate by passing messages between one another. This is done via pairs of

¹ 1-DSMA — Singly persistent data sensing for multiple access.

message queues managed by the scheduler. The scheduler also directs the invocation of a thread's methods. The kernel includes a number of system threads, managing some of the internal components of a node. Alongside these, application threads are used to make a node perform a particular task. A cooperative scheduling strategy is employed in which any of a thread's methods must complete before any other thread can run.

Threads are identified by their unique *ThreadIds*. System threads are given well-known *ThreadIds*. An application thread may replace any system thread. In this way the runtime environment can be upgraded in a soft manner, and a programmer has complete control over how a node's internals are directed.

The Piconet Loader – The Piconet loader provides the mechanism by which a node can be configured for a particular task. This happens by booting a node through one of its external interfaces. It is possible to boot a node through its radio interface by way of the *radio boot protocol*. This allows a boot server to dynamically configure a client with a pre-specified boot image. Once configuration is complete, the Piconet node will retain its image indefinitely.

The boot procedure begins with a *boot plea* from the unconfigured node, or client. The boot plea is issued on a well-known multicast address, and contains details about the client node- its address, home "domain," and version number. Nodes configured as boot servers may respond to the boot plea with a *boot offer*. A server does this after analysis of the boot plea in order to determine whether it has a suitable image to offer.

Upon receipt of a boot offer, the client node may decide to accept it, and in doing so issues a boot request. After this the boot image is relayed. The boot image may contain code and/or data. The boot procedure is outlined in Fig. 3.

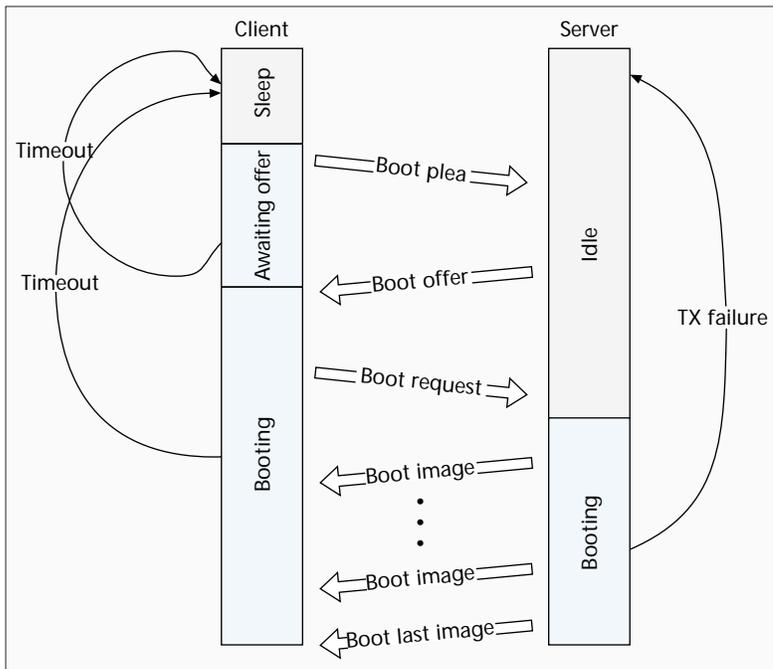
Use of the radio boot protocol extends beyond merely the initial configuration of a Piconet node. It represents a more general mechanism by which code and data can be passed between nodes. This might be useful for dynamically upgrading the interface that a node offers to a particular service, for example.

This dynamic reconfiguration of a node has implications for power saving. There is a cost trade-off between transmission and computation, particularly in a low-power device. When transmission is much more costly than computation, it may be more efficient to configure a node with a particular decompression algorithm before sending data to it.

The Attribute Store

A primary means by which Piconet nodes can interpret data received over their radio interfaces is by examining a datagram's source and destination addresses, particularly in the case of multicast addressing. Generally, however, it will be necessary to map these addresses into names and types. Furthermore, additional information about a node may well be required in order to make any sense of the data received from it. We need a common mechanism by which nodes can present this basic information to each other.

Piconet nodes neither expect nor rely on an underlying infrastructure of base stations in order to communicate. In a truly mobile ad hoc system the task of mapping addresses to names, and discovering a node's resources and services, cannot easily be centralized. In Piconet, we prefer each node to



■ Figure 3. Piconet's radio boot protocol.

be individually responsible for describing itself to the rest of the world, and for any other node to be able to interrogate it to determine what kind of services it either requires or provides. Every node is equipped with an attribute store that performs this function.

The Piconet attribute store is a simple hierarchical directory structure of (name, value) pairs. Its use is not restricted to the presentation of basic naming and type information; rather, the attribute store is available as a general resource within a node for presenting any kind of information to other nodes.

The attribute store is a resource used both internally within a Piconet node, and externally as a presentation interface to other nodes. The directory structure is soft, in that an attribute's name is merely a "/"-delimited sequence of printable characters. The length of a name is restricted to 128 characters. The attribute store has no typing other than that maintained by convention. As a result, attributes' values are managed as simple strings. We made the decision not to include any typing within the attribute store since to do so might have restricted us, and would certainly have made its implementation more complicated.

There are mechanisms for reading and writing, watching, and cleaning up the values stored in the attribute store. These mechanisms are available locally within a Piconet node, and remotely via the radio link. It is the adoption of standard names in this store that allows Piconet nodes to discover the services available in their vicinity and to make use of them.

The particular facilities the attribute store offers can be described by way of the following example. Consider a simple temperature sensor with a Piconet interface. The node will interpret information provided by the sensor and make it available over Piconet. Another node that comes across this temperature-sensing node might make the queries outlined in Table 1. Having discovered the name of the device, it decides to watch an attribute. In doing this, it is returned a *WatchHandle*. The temperature-sensing node will then issue a notification of any change in temperature by passing the *WatchHandle* and the new temperature to the watching node. An attribute watch like this will be canceled after a timeout, if, for example, the watching node disappears.

Query	Reply
GetAttribute("/name")	"/name"="Temperature Sensor"
GetAttribute("/temp/C/value")	"/temp/C/value"="17"
WatchAttribute("/temp/C/change5")	"WatchHandle 01"
	01 "/tmp/C"="24"
UnwatchAttribute(01)	

■ **Table 1.** *Queries to an attribute store.*

Security

There has been an explicit “not yet” attitude to security in Piconet. Certainly, there is great scope for experimenting with the design of security models for this kind of highly mobile computing environment. Indeed, adequate security mechanisms will become essential here. However, the feeling has been that it would not be beneficial to be burdened with these requirements at this early stage.

Experiences

Our prototype Piconet nodes are now fully operational, and we are beginning to make use of them in some preliminary applications. We are gaining valuable insights from these experiments — about the nature of the kind of network we have built, about the kind of applications it enables, and about how both of these can be improved.

Personalized Displays

By combining a Piconet node with a small, low-power display, we make a wireless, portable information point. At ORL we have built such a device around a zero-powered Bistable Cholesteric display developed by Kent Display Systems [5]. These low-powered stick-anywhere information appliances are supported by a customizable data repository. The displays are positioned around the building, in public areas and individual offices. The service is integrated with our Active Badge² system, and the information the displays convey can depend on time, location, particular user, and other events.

The Personalized Displays system imports information from a variety of sources, including the World Wide Web, the Active Badge system, and localized sensors. This information is then presented to a user’s *agent* which determines what information is displayed on which display, according to the user’s preferences.

Pico GPS

An in-car GPS unit uses Piconet to convey information about its current position, speed, and direction. The device is installed in the boot of the car. Any other node within the car can make use of the information provided by this unit. For example, a database containing mapping information might use information from the GPS unit to display the car’s current position to the driver. It could do this by representing a map on a Piconet display on the car’s dashboard.

As well as its use for navigation, the information provided by the GPS unit is logged by a PDA the driver carries in his/her briefcase. The PDA also has a Piconet node, so this happens automatically. If a properly authorized cellular phone is nearby, position reports may be sent by the PDA to prespecified destinations. When the driver returns to his/her office, the logged information is backed up from the

PDA to a central index and archive. This also happens automatically, just as the driver enters the office.

Direction

Our preliminary experiments with Piconet have shown us much about the requirements for embedded networking. We are learning about the requirements of a suitable radio, as well as the qualities protocols must have to allow power-saving operation. We outline here some of the directions we expect our work to take.

Power Control in Piconet

Piconet nodes will come in many different forms. Some of them will not be mobile, but could be plugged into permanent power supplies to provide network access, environmental sensing, display services, and so on. Others will be as small and lightweight as we can make them, having no more than a coin cell for power. The low-power nodes will have to spend most of their time in sleep modes if we are to attain good battery life. This means they can neither transmit nor receive radio transmissions much of the time. We are thus left with the problem of how two low-power nodes will ever discover each other’s presence if they are only active for short intervals.

One method is to synchronize their waking intervals, either by internal clocks or external time reference. Internal clocks will drift, but this can be overcome. We will also have an initial synchronization problem, particularly acute where two groups of differently synchronized nodes come within range of each other. External time references put extra hardware and complexity into every Piconet node, which we would like to avoid unless there is no other option. Such synchronization will also cause contention around the synchronization time if there are many nodes present, while the band may be otherwise free. This will waste power.

Another means of discovery is to design a Piconet node that can be brought out of sleep mode by a low-power RF detection circuit, perhaps like that used in the Active Badge [6]. To enable rendezvous, for example, whenever nodes have been within range for 30 s, we would ensure that every node comes out of sleep and transmits a broadcast packet containing its unique ID at least every 30 s. This would have the effect of *waking up* any nodes within range, which can decide whether they need to discover more about the transmitter based on what they know about it already. If they wish to communicate further, to find out more about the node or to open a dialog with it, they can transmit requests directly to it, which will wake it up and cause it to remain active until the dialog is finished.

Our current nodes are being kept very simple, and do not have the low-power RF detection circuit. For this we sacrifice unaided rendezvous between low power nodes. We use the scheme where every node periodically broadcasts its unique ID, and follows this with a brief interval during which its receiver is switched on. In this time, other nodes can cause it to stay active, or “wake up.” They do so by transmitting sufficient signal to trigger carrier detection or signal strength circuitry. A node can use this mechanism to detect what is in its vicinity. To do this, it will remain with its receiver switched on, and interrogate any nodes that broadcast their IDs.

This leaves us with the problem of when, and for how long, a node should listen to the channel. One way we can do this is by an external stimulus. This can be by a request from the user of a PDA, opening the case, running appropriate software, or on the arrival of an alarm time. It could be triggered

² Active Badge is a registered trademark of Ing. C. Olivetti & C. Sp.A.

by switching on a cellular phone or by the movement of a device after a period of stillness. A more important way is by a packet sent out over the radio. When a person has gathered together the nodes they wish to intercommunicate, they need only trigger one into remaining active, and it can then trigger all the others as they beacon. Within a minute or so, all nodes that are in range of the initial trigger and can intercommunicate directly will have learned of each others' presence.

Not all Piconet nodes will be short of power. Some nodes will be attached to devices with ample power supplies, ranging from cellular phones (10 W/hr) through notebooks (30 W/hr) to cars (600 W/hr) and mains (unlimited). These nodes can remain active in receive mode at all times, and will of course broadcast their IDs every 30 s as usual. We would call these *listening nodes*. When they hear the broadcast from another node, they will look it up in a cache of recently seen nodes and, if necessary, interrogate it to find out what it is and which services it offers and requires. If they have sufficiently recent information, they will not respond. When a low power node (named A) is interrogated by a listening node (named L), it can request a list of other recently seen nodes identified by their IDs. If there are some IDs it does not know about, it can immediately request details about them from the cache maintained by listening node L. If low-power node A wishes to communicate with one of these recently seen nodes (named B), it can choose to remain active for 30 s and respond directly to the broadcast from node B. It may even obtain a hint about how often and how recently that node broadcasts from listening node L. If the listening node cannot provide enough information about node B for A to make a decision, node A will remain active long enough to establish direct contact with B and obtain sufficient details. Thus, the listening node is used purely as a hint mechanism, with communications being established directly if there is any interest. This is important because the other nodes may no longer be in the vicinity, and node B may even move away with node A. Once they are in contact, A and B can sleep for predetermined periods, awakening to transmit and listen for each others' presence. This listening period will be longer than usual to account for missynchronization, but if it extends too long (perhaps 1 s), the nodes will assume they have lost contact and return to their default state.

Note that in the common case, node L will not even respond to node A's broadcasts because it has already cached information about it. We will be investigating mechanisms by which L will again respond to A's broadcasts if L has further information for A, or if A specially marks the broadcast as a query as well as an ID broadcast. If A wants to be sure of what is around right now, it can of course stay listening for the full 30 s for the broadcasts and then interrogate the nodes.

There are techniques that can be adopted to save power progressively as a power source is consumed. It may be advantageous for a node to detect a low-battery condition and take several actions. It could make its battery status available as an attribute which can be read by a *maintenance* node. Levels of service offered by a node may be reduced as the node runs short of power. Finally, and as a last resort, a node may lengthen the interval between its broadcast transmissions.

Range Extension

In certain circumstances, it would be of benefit to have control over the range of a Piconet node. In particular, nodes that had longer, or selective, range would make new applications possible. These include allowing devices within a house to intercommunicate. In these circumstances, time distribution, wake-up and switch-off controls, heat and light control, and last-hop access for modems would be possible.

An alternative to longer-range nodes is to allow the forwarding of Piconet messages on behalf of other nodes. This opens up the whole area of wireless routing protocols, and access to services by multihop routes. The limited range and low cost of Piconet nodes should allow us to provide researchers in this area with good experimental platforms on which to work. Existing protocol simulators can be used to investigate options, and can then be validated by the deployment of dozens of Piconet nodes within a single building.

Higher Data Rates

Higher data rate radios (1–2 Mb/s) may become attractive if they can be made cheaply enough, as the power per bit transmitted or received tends to be lower at higher bit rates. A high-speed radio system, developed at ORL [7], operates at 10 Mb/s, consuming 5 W, or 0.5 μ J/b. Piconet operates at 40 kb/s, consuming 250 mW or 6 μ J/b. This makes Piconet some 12 times less efficient in power consumption for the same amount of data transmitted. However, there is a trade-off with the power consumption when a node is listening over a long period for rendezvous, because our high-speed radio will consume 20 times more energy than Piconet when receiving. Piconet is presently limited to 40 kb/s by the size and cost of the available radio transceivers, but if we can move to higher bit rates, we must seriously consider a secondary radio or the use of timing protocols for rendezvous purposes.

Asynchronous Design

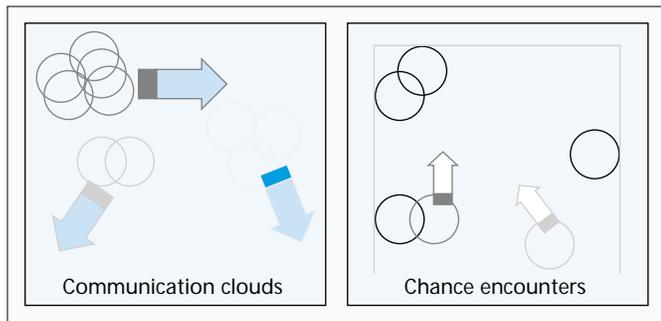
We are experimenting with the Amulet2e asynchronous processor produced by the Amulet Group at Manchester University [8]. Given the intermittent, and often infrequent, nature of Piconet communications, we feel that asynchronous design has the potential for great power savings. Our hope is that we can interrupt the processor directly upon the reception or transmission of each bit, byte, or packet, and that the processor will only work for the few cycles that most of these events demand, stopping within one cycle when there is no more work to be done. When new events occur, over either the radio or a wired interface, the processor will be up and running within a single cycle time, thus saving the power normally wasted in stabilizing clocks and resetting processor states. Substantial comparisons of the asynchronous and synchronous approaches are underway.

Piconet Applications

To describe further what we hope to achieve in building Piconet, we outline here some of the application areas it makes possible. We describe these applications as sets of *services* offered by Piconet nodes, both mobile and static. Applications include extensions and improvements to existing systems, as well as new systems. We have found it instructive to consider devices as offering services to each other over Piconet. In this way we can categorize the kinds of devices and applications we would like to see, as well as clarify what is required to make them possible.

Authorized Services

A particular set of nodes may be programmed to recognize each other and provide very specific services. For example, a PDA may recognize a telephone over which it is allowed to make calls or a printer to which it may send a document. Often, the user bringing the relevant Piconet devices into deliberate proximity will trigger these sorts of activities. Here, then, it is a user's explicit action that causes things to happen; connectivity between nodes may continue for some period in this case.



■ Figure 4. Mobile systems using embedded networks.

Piconet Upgrades to Existing Services

Imagine that we replace all the embedded controllers in modern consumer devices with Piconet nodes. This will allow those consumer devices to communicate with each other when they are in sufficiently close proximity. If we can support low bit rates (say 10 kb/s between any pair of devices), we can see some immediate benefits.

Any device that requires the time to be set (VCRs, microwave ovens, central heating controllers, clocks, radios, PDAs) can obtain an update whenever one of them is set. This update may come from a service (RDS, Teletext, Rugby MSF, GPS) that one of them receives, or may be set by the user.

If a digital communications device is linked to Piconet, that communications medium may be used to control or interact with other nearby Piconet devices. In the home, a modem on the telephone line equipped with Piconet would allow remote access to program a VCR, alter central heating, and perhaps control house lighting. A cellular phone equipped with Piconet would allow portable computers to send and receive data calls (Short Message Service, data connections, or faxes) just by being brought into sufficient proximity, or could automatically divert to a nearby wired telephone to save cellular bandwidth and call costs. A networked computer equipped with Piconet would allow access to nearby PDAs or display devices to exchange e-mail and important data files, display timely information, or help authenticate users.

New Piconet Services

Piconet provides a communication channel, but because of its limited range it is also a sensor system that provides proximity information. This information can be used to trigger actions, so Piconet can be used for *context-aware* applications.

If a Piconet node equipped with a few kilobytes of memory is allocated to an individual, it can be programmed with their personal preferences for common systems. For instance, telephone lists could be carried in the node and edited on any suitable Piconet computer. They could be used to set the user interface on Piconet-equipped telephones for speed dialing. Calls could be forwarded automatically from a suitably equipped private automated branch exchange (PABX).

More interestingly, Piconet nodes can be placed around the working environment to act as beacons. One example would be a node in a meeting room that would request other nodes to be quiet when it was told a meeting was in progress. This could be used to suppress alarms, telephone calls, and, in an ideal world, the chimes of digital watches. It would perhaps be linked to the meeting room booking facility to gain information on what the meeting was about. This could be recorded by a participant's personal Piconet node, along with the text names of other Piconet nodes at the meeting, forming a record of their activities suitable for a memory prosthesis [9] or video indexing and retrieval [10].

Further to these simple beacons, Piconet nodes can adopt a common format for location coordinates. At ORL, we promote the use of latitude, longitude, and altitude with the WGS84 datum. Nodes that are known to be static (e.g., the meeting room beacons) can be told their location and make this available to passing nodes. Nodes that are attached to positioning systems (e.g., consumer satellite navigation systems) can make dynamic location information available in the same format, for instance, in cars and aircraft. This information can be used to trigger events when other nodes pass, thus opening up all the possibilities described in [6] and [11]. Personal nodes can record location against time and so build up an accurate trace of where the node has been. PDA alarms for meetings can be related to a location and will be given earlier if the PDA is far away, but suppressed if the PDA is heading in the right direction or has arrived at the location.

As many navigators appreciate, knowing exactly where you are does not mean you know where to go. Using Piconet, we can augment direction signs with information about the routes to thousands of destinations. For instance, road signs could carry electronic information on which exit to take to any destination in the United Kingdom that can be described by postal code and street number. The user of a navigation device would enter the postal code and number of their destination. The device would pass this postal code to any nearby road signs which would respond with the appropriate exit to take and any ancillary information such as distance, expected journey time, and road condition alerts. This exit and other information would then be presented to the user. Clearly this operation would be on a very large scale if it were to accommodate vast numbers of users on motorways, for example. While the specific technology we have developed may not be appropriate in this case, the principle of short-range, context-based, wireless telemetry is still appropriate for this application.

The example of road signs and postal codes is only one combination. The Piconet could be used to guide passengers on public transport, through buildings, around museums and galleries, through tourist sites, and so on. The routing information could be made time-dependent, and could be updated swiftly in case of diversions or delays. It could also be personalized by type of route (e.g., scenic, fast, cheap, no stairs). For many destinations it is only the service that is important, as in the case of tourist information offices, public conveniences, taxi ranks, hospitals, and car parks. All this information could be written on signs placed at strategic locations, but they would soon become an eyesore and costly to install and maintain.

As a further example of the use of Piconet for sensing, consider audio tours as used in museums and outdoor tourist sites. These are often hired out in the form of a cassette player with a taped commentary which requests the user to stop the tape, walk to the next numbered site, then start the tape again. With Piconet and a random access medium, such as MiniDisc,³ the user could be freed to vary the recommended itinerary. The routes could be customized to the level of interest, or length of time for the tour. If several people were taking the tour together, their commentaries could be synchronized whenever they were together so that they were all looking at the same objects or getting the jokes at the same time. If Piconet nodes on doorways and pathways were connected into a backbone network, lost tour members could be located. Finally, at the end of the tour, a personalized printout containing the route followed and more details of the items where most time was spent could be automatically generated.

³ MiniDisc is a registered trademark of Sony Corporation.

Mobility and Communication

To understand the importance of embedded networking to mobility, we describe two different ways in which mobile systems can use services available through an embedded network. Figure 4 illustrates these.

Communication Clouds — If a set of communication endpoints move around but remain in range of each other, we have a *cloud of devices* as might be carried by a person, in luggage, in vehicles, or between a small group of people. Such devices can be made aware of each other via Piconet, and can offer services to each other. They will often be authorized to use each other's services, and may be able to do this for extended periods. For example, my PDA in one pocket may be authorized to use my mobile phone in the other to send and receive short messages.

Chance Encounters — If one endpoint moves around, occasionally seeing other nodes that provide services to which it has no special authorization, this is a *nomadic* node. The sort of services it might use are those that tell it about its environment — position and local facilities, and perhaps those that allow it to personalize another node — by configuring it in a way that is suitable for a particular user. For example, a telephone may be primed with my commonly dialed numbers, because it detects a node owned by me nearby.

Resource Discovery and Description

Another property that characterizes Piconet is *decentralized* resource discovery and description. Given the highly mobile and ad hoc nature of such a network, every device must be able to independently describe itself to a sufficient level in order for it to be useful to others. The advantage of this decentralized approach is that we are not tied to a central naming service, which would be inappropriate in an ad hoc network. Given the *proximate* nature of embedded networking, knowing about things that are not nearby is less important. Some of the ideas about this approach to resource description were first outlined in [12].

Discussion

We have built a prototype embedded network, Piconet. It is one component of a framework of systems and services being envisioned and developed at ORL to support our view of mobility and communications.

With Piconet we can demonstrate that radio is the preferred medium for this kind of short-range ad hoc communications between embedded and mobile devices. Piconet does not predetermine the specific types of device that can make use of this connectivity, but provides a simple and flexible mechanism by which we can experiment with empowering all sorts of devices.

To this end, we have kept Piconet's system design simple, in particular that of the attribute store. Within the attribute store, we expect that with *attribute grouping*, supported by the store's hierarchy, we can experiment with type systems, automatic resource timeout, and recovery. In addition, we will be able to build generalized sensor interfaces and a variety of distributed mobile applications. We will then be in a strong position to specify the minimum and maximum functionality expected of any node.

We envision that ubiquitous embedded networks such as Piconet will become the basis for the personalization of the work and home environments. We expect that further advances in radio and processing technologies will provide for more powerful and flexible embedded network devices which will be able to perform an increasing variety of tasks. In addi-

tion, we will see smaller devices that may be easily incorporated into any appliance. Should such embedded communication technology become commonplace, the creation of new applications for the interaction of objects will become trivial.

References

- [1] B. Schilit, N. Adams and R. Want, "Context-Aware Computing Applications," *IEEE Wksp. Mobile Comp. Sys. and Appls.*, Dec. 8-9, 1994.
- [2] Infrared Data Association specs, v. 1.1. June 1996.
- [3] V. Bharghavan *et al.*, "MACAW: A Media Access Protocol for Wireless LANs," *Proc. Sigcomm '94*.
- [4] C. Fullner and J. J. Garcia-Luna-Aceves, "FAMA-PJ: A Channel Access Protocol for Wireless LANs," *Proc. Mobicom '95*.
- [5] Kent Displays Inc., 343 Portage Blvd, Kent, OH 44240 USA.
- [6] R. Want and A. Hopper, "Active Badges and Personal Interactive Computing Objects," *IEEE Trans. Cons. Elect.*, vol. 38, no. 1, Feb. 1992.
- [7] J. Porter and A. Hopper, "An ATM Based Protocol for Wireless LANs," ORL tech. rep. 94.2.
- [8] "The Amulet2e Microprocessor," Dept. Comp. Sci., Univ. Manchester.
- [9] M. Lamming *et al.*, "The Design of a Human Memory Prosthesis," *Comp. J.*, vol. 37, no. 3, 1994.
- [10] M. Brown, "Open-Vocabulary Speech Indexing for Voice and Video Mail Retrieval," *Proc. 4th ACM Int'l. Multimedia Conf.*, Boston, MA, Nov. 1996.
- [11] P. J. Brown, "Context-Aware Applications: From the Laboratory to the Marketplace," *IEEE Pers. Commun.*, this issue.
- [12] D. C. Oppen and J. K. Dalal, "The Clearinghouse: A Decentralized Agent for Locating Named Objects in a Distributed Environment," *ACM Trans. Office Info. Sys.*, July 1993.

Additional Reading

- [1] R. Want *et al.*, "The ParcTab Ubiquitous Computing Experiment," Xerox PARC tech. rep. CSL 95-1, Mar. 1995.

Biographies

FRASER BENNETT (FBennett@orl.co.uk) is a research engineer at the Olivetti and Oracle Research Laboratory in Cambridge, England. He is a graduate of Bristol University, where he studied mathematics. His research interests include location technologies, mobility, and wireless networking.

DAVID CLARKE is a research engineer at the Olivetti and Oracle Research Laboratory in Cambridge, England. He is a graduate of the University of Cambridge, where he studied engineering, and is a member of the IEE. His research interests include the design of low-power electronics for mobile use.

JOSEPH B. EVANS [StM '82, M '89] received a B.S.E.E. from Lafayette College in 1983, and M.S.E., M.A., and Ph.D. degrees from Princeton University in 1984, 1986, and 1989, respectively. Since 1989 he has been at the University of Kansas, first as an assistant professor and since 1994 as an associate professor of electrical engineering and computer science. Prior to this he held a postdoctoral position in the Network Systems Research Department of AT&T Bell Laboratories in Holmdel, New Jersey, where he was involved in the design of a high-performance integrated network. While at Princeton he was awarded an AT&T Bell Laboratories Graduate Fellowship for 1984-1988, during which he was also an employee of Bell Laboratories, working in the field of speech processing algorithms for packet network. His current research interests include high-speed (gigabit) networks, mobile networking, digital signal/speech processing, special-purpose computer architecture, and VLSI implementations.

ANDY HOPPER is vice president of research of Ing. C. Olivetti & C. SpA, Italy, director of the Olivetti and Oracle Research Laboratory (ORL) in Cambridge, chief technical officer of Advanced Telecommunications Modules Limited, and chairman of Telemedia Systems Ltd. He is also a reader in computer technology at the University of Cambridge and a fellow of Corpus Christi College. His research interests include networking, multimedia, and mobile systems. He received a B.Sc. from the University of Wales and a Ph.D. from the University of Cambridge.

ALAN JONES is a principal research engineer at ORL. He was awarded a B.A. in physics in 1981, a computer science diploma in 1982, and a Ph.D. in computer science in 1986, all from the University of Cambridge, England. He has worked on control systems, sensor devices, optimization, simulation, real-time multimedia, and a variety of mathematically oriented projects. Current interests include communication and navigation systems, as both a keen user and a developer.

DAVID LEASK is currently undertaking a Ph.D. at the University of Cambridge. He received B.E., B.Sc., and M.Eng.Sc. degrees from the University of Melbourne, Australia. His research interests include low-power wireless networks, broadband communications, and security protocols.