Piconet

Embedded Mobile Networking

Frazer Bennett[*], *David Clarke*[*], *Joseph B. Evans*[*α 1],

Andy Hopper[*α], *Alan Jones*[*], *David Leask*[α]

[*] ***The Olivetti and Oracle Research Laboratory***
24a Trumpington Street
Cambridge
CB2 1QA
United Kingdom
http://www.orl.co.uk/

[α] ***University of Cambridge Computer Laboratory***
New Museums Site
Pembroke Street
Cambridge
CB2 3QG
United Kingdom
http://www.cl.cam.ac.uk/

Abstract

Piconet is a general purpose, low powered, 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 paper outlines the Piconet system, under development at the Olivetti and Oracle Research Laboratory (ORL). We discuss the motivation for providing this low-level 'embedded networking', and describe our experiences of building such a system. We conclude with a commentary of some of the implications that power-saving, and other considerations central to Piconet, have on the design of the system.

1 Introduction

There is a great divide in mobile computing between what is desirable and what is practical. This divide is inherent and caused by, amongst 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 PDAs to very small, simple, embedded computing devices that may go almost completely unnoticed. Indeed, the extent to which these devices are embedded means that we are already at the stage where people are unaware of how many computers they may use in a day.

[1] Visiting from University of Kansas, Info. and Telecomm. Tech. Center , Lawrence, KS 66045, USA

It is the integration of this vast array of mobile and embedded computing objects that 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 paper 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 recognise that to do this effectively we must build, deploy and use a system that demonstrates these concepts. This is what we have done.

2 Embedded Mobile Networking

Embedded networking concerns the provision of a network that 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 personalisation things nearby, or allow two devices near to each other to inter-operate. 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 that these applications can be *context aware* **[Shilit94]**.

3 Technology Characteristics

The kind of applications that 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.

3.1 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.

3.2 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 powered components, we must adopt protocols that allow a device's network interface to be switched off for 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, then by implication they are near to each other. This proximity information makes *context-aware* applications and *personalisation* possible.

3.3 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 outdoors. 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 standardisation activities such as those by IrDA **[IrDA96]**. 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.

4 Piconet System Design

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

Our Piconet node is composed of a 418MHz FM transceiver, a FPGA that drives the radio physical layer and provides MAC support, and a microcontroller with a runtime environment. Figure 1 shows a prototype node – it measures 127x74mm. RF screening cans cover most of the board. The lower left-hand 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.

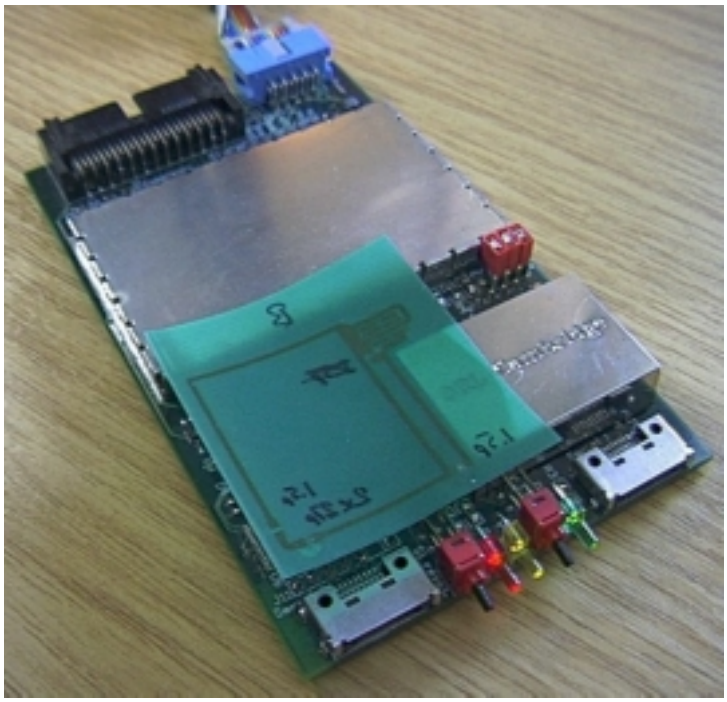

Figure 1 - Prototype Piconet hardware

Piconet's *runtime* environment allows rapid and automatic configuration of a node for a particular task. This is done at power-up by booting code and data into a node through any one 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.

4.1 Piconet radio

The physical range of the radios used by Piconet is constrained to around five metres. 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 re-use of the radio channel, so increasing the aggregate bandwidth available. Finally, it parameterises the system to work at *human ranges*. By this we mean that Piconet enables communication between objects that are within a human's immediate surroundings. Those things that are nearby to somebody – that are 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 to each other as they move around, perhaps being carried by someone, then they should be able to spontaneously inter-communicate.

Radio protocols

In choosing suitable radio protocols for Piconet, we had to particularly consider its ad-hoc and low-powered 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 be continuously drifting in and out of contact with each other. These characteristics distinguish Piconet quite significantly from those systems outlined in **[Bharghavan94]** and **[Fullmer95]**.

Our initial protocols are oriented towards 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 can not 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 IEEE802.11 standard. IEEE802.11 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 amongst 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 warm-up 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 four-bit value as a unique six-bit code. Each six-bit code used has the property of containing three ones and three zeroes. 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 40kbaud that, with the encoding overhead, gives us 28.8kb/s. Additional overheads from link-layer and MAC functions leave us with an estimated 9600 bit/s data-rate, half-duplex. It is worth emphasising that whilst 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 Figure 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 eight-bit counters inside the node. In addition, smaller packets result in a better sharing of the radio channel, since no single device is transmitting for too long.

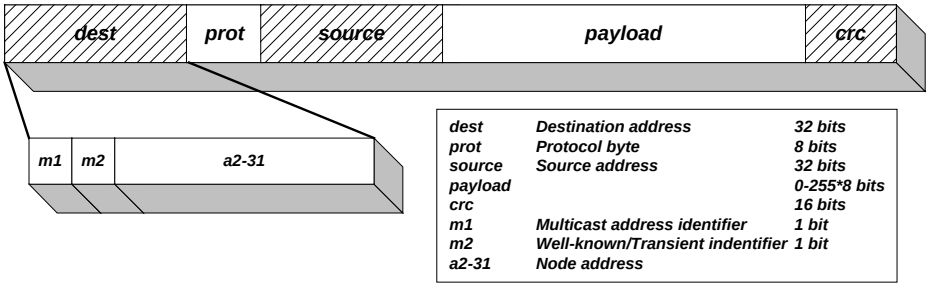

Figure 2 - Piconet datagram format

Piconet supports two types of multicast traffic in the form of *well-known* and *transient* multicast groups. As indicated in Figure 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 recognised 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 six-bit codes, and it is much more likely that errors are interpreted as coding violations than CRC 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 back-off 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 back-off 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 back-off timer.

4.2 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 analogue, 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, in particular device descovery 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 for most of the time.

[1] 1-DSMA – *Singly persistent Data Sensing for Multiple Access*

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 megabytes) is available for applications. An application is a piece of code which is used to configure a node to perform a specific task – to act as an interface to a sensor connected to the node, for example.

The Piconet Kernel

The kernel has a simple multi-threaded design. Within the kernel, threads communicate by passing messages between one another. This is done via pairs of *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 co-operative 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 Figure 3.

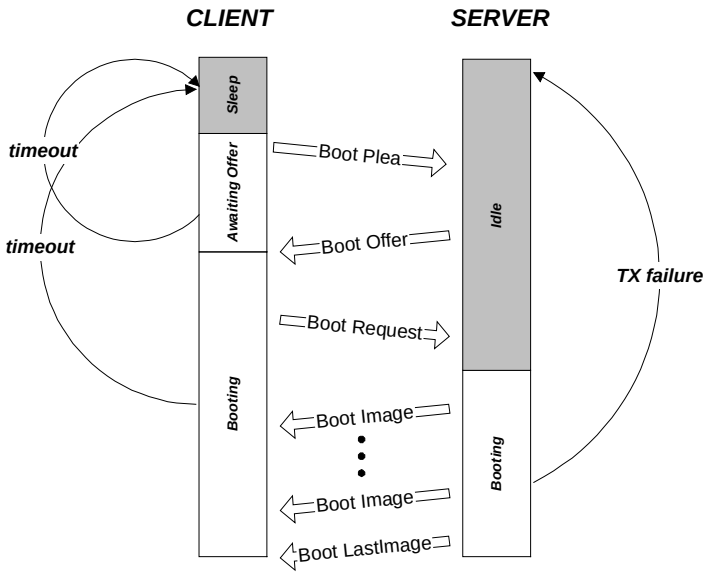

Figure 3 - Piconet's radio boot protocol

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 powered 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.

4.3 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, in particular 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 upon an underlying infrastructure of base-stations in order to communicate. In a truly mobile ad-hoc system the task of mapping of addresses to names, and discovering a node's resources and services, cannot easily be centralised. In Piconet, we prefer each node to 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, but 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 may 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 that are available in their vicinity, and to make use of them.

The particular facilities that 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 which comes across this temperature sensing node might make the queries outlined in Table 1. Having discovered the name of the devices, 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 to the temperature, by passing the WatchHandle and the new temperature to the watching node. An attribute watch like this will be cancelled after a timeout – if the watching node disappears, for example.

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

4.4 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.

5 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 that it enables, and about how both of these can be improved.

5.1 Personalised displays

By combining a Piconet node with a small, low-powered 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 **[Kent97]**. These low-powered stick-anywhere information appliances are supported by a customisable data repository. The displays are positioned around the building – in public areas and individual offices. The service is integrated with our Active Badge[1] system, and the information that the displays convey can dependent on time, location, particular user, and other events.

[1] Active Badge is a registered trademark of Ing. C. Olivetti & C. S.p.A.

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

5.2 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 which the driver carries in his briefcase. The PDA also has a Piconet node, so this happens automatically. If a properly authorised cellular 'phone is nearby, position reports may be sent by the PDA to pre-specified destinations. When the driver returns to his 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 his office.

6 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 that protocols must have to allow power-saving operation. We outline here some of the directions which we expect our work to take.

6.1 Power Control in Piconet

Piconet nodes will come in may 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 etc. 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 that they can neither transmit nor receive radio transmissions for 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 synchronise 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 synchronisation problem, particularly acute where two groups of differently synchronised nodes come within range of each other. External time references put extra hardware and complexity into every Piconet node, so we would like to avoid that unless we have no other option. Such synchronisation will also cause contention around the synchronisation 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 **[Want92]**. To enable rendezvous, for example whenever nodes have been within range for thirty seconds, we would ensure that every node comes out of sleep and transmits a broadcast packet containing its unique Id at least every thirty seconds. 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 dialogue with it, they can transmit requests directly to it, which will wake it up, and cause it to remain active until the dialogue 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 detect 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 in which we can do this is by an external stimulus. This can be by a request from the user of a PDA, by opening the case, running appropriate software, or on the arrival of an alarm time. It could be triggered by the switching on of 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 that they wish to intercommunicate, they only need to trigger one into remaining active, and it can then trigger all of the others as they beacon. Within a minute or so, all nodes which are in range of the initial trigger and can intercommunicate directly will have learnt 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 (10Whr) through notebooks (30Whr) to cars (600Whr) and mains (unlimited). These nodes can remain active in receive at all times, and will of course broadcast their Ids every thirty seconds 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 node and, if necessary, will interrogate it to find out what it is and what 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 that it does not know about, it can immediately request details about them from the cache maintained by the listening node L. If the low power node A wishes to communicate with one of these recently seen nodes (named B), it can choose to remain active for thirty seconds, 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 the listening node L. If the listening node cannot provide enough information about node B for A to make a decision, then node A will remain active for 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 as 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, awaking to transmit and listen for each others presence. This listening period will be longer than usual to account for mis-synchronisation, but if it extends for too long (perhaps 1 second), then the nodes will assume that 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 seconds 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.

6.2 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 multi-hop 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.

6.3 Higher data rates

Higher data rate radios (1-2 Mbit/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 **[Porter94]**, operates at 10Mbit/s, consuming 5W, or 0.5 microjoules per bit. Piconet operates at 40kbit/s consuming 250mW or 6 microjoules per bit. This makes Piconet some 12 times less efficient in power consumption for the same amount of data transmitted. However, there is a tradeoff with the power consumption when a node is listening over a long period for rendezvous, as our high-speed radio will consume 20 times more energy than Piconet when receiving. Piconet is presently limited to 40kbit/s by the size and cost of the available radio transceivers, but if we can move to higher bit rates, then we must seriously consider a secondary radio or the use of timing protocols for rendezvous purposes.

6.4 Asynchronous design

We are experimenting with the Amulet2e asynchronous processor produced by the Amulet Group at Manchester University **[Amulet]**. 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, either over the radio or a wired interface, the processor will be up and running within

a single cycle time, thus saving the power that is normally wasted in stabilising clocks and resetting processor states. Substantial comparisons of the asynchronous and synchronous approaches are underway.

7 Piconet Applications

To describe further what we hope to achieve in building Piconet, we outline here some of the application areas which 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 categorise the kinds of devices and applications that we would like to see, as well as clarify what is required to make them possible.

7.1 Authorised services

A particular set of nodes may be programmed to recognise each other and provide very specific services. For example, a PDA may recognise 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.

7.2 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 10kbit/s between any pair of devices) then 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, then 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, 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 to authenticate users.

7.3 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, and 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 dialling. Calls could be forwarded automatically from a suitably equipped 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 that 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 **[Lamming94]** or video indexing and retrieval **[Brown,M96]**

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 to the WGS84 datum. Nodes that are known to be static (for instance, 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 **[Want92]** and **[Brown96]**. 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 that 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 UK that can be described by postcode and street number. The user of a navigation device would enter the postcode and number of their destination. The device would pass this postcode to any nearby road signs which would respond with the appropriate exit to take and any ancillary information such as distance, expected journey time, road condition alerts etc. 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. Whilst the specific technology that 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 postcodes is only one combination. The Piconet could be used to guide passengers on public transport, through buildings, around museums and galleries, through tourist sites etc. The routing information could be made time dependent, and could be updated swiftly in case of diversions or delays. It could also be personalised by type of route - scenic, fast, cheap, no stairs, for example. For many destinations, it is only the service that is important, as in the case of tourist information offices, public conveniences, taxi ranks, hospitals, car parks. All of 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[*], the user could be freed to vary the recommended itinerary. The routes could be customised to the level of interest, or length of time for the tour. If several people were taking the tour together, their commentaries could be synchronised whenever they are together, so that they are all looking at the same objects, or get the jokes at the same time. If Piconet nodes on doorways and pathways were connected into a backbone network, then lost tour members could be located. Finally, at the end of the tour, a personalised printout, containing the route followed and more details of the items where most time was spent could be automatically generated.

7.4 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, below, 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 about 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 authorised 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 authorised to use my mobile phone in the other, for sending and receiving short messages.

Chance Encounters

If one endpoint moves around, occasionally seeing other nodes that provide services to which it has no special authorisation, 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 personalise 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 dialled numbers, because it detects a node owned by me nearby.

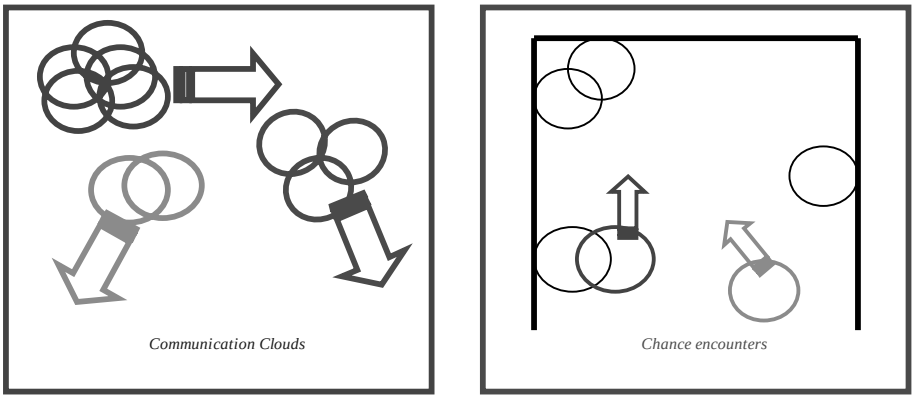

Figure 4 Mobile Systems using Embedded Networks

---

* MiniDisc is a registered trademark of Sony.

7.5 Resource discovery and description

Another property that characterises Piconet is *de-centralised* 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 de-centralised approach is that we are not tied in to a central naming service, which would be inappropriate in an ad-hoc network. Given the *proximate* nature of embedded networking, knowing about things which are not nearby is less important. Some of the ideas about this approach to resource description were first outlined in **[Oppen83].**

8 Discussion

We have built a prototype embedded network, Piconet. It is one component of a framework of systems and services which are 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 generalised 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 personalisation 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 that will be able to perform an increasing variety of tasks. In addition, 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.

9 References

Bharghavan94 MACAW: A Media Access Protocol for Wireless LANs. *Vaduvur Bharghavan et al.* Sigcomm 94.

Fullmer95 FAMA-PJ: A Channel Access Protocol for Wireless LANs. *ChaneL. Fullmer and J.J Garcia-Luna-Aceves.* Mobicom 95.

IrDA96 Infrared Data Association specifications, version 1.1. June 1996.

Lamming94 The Design of a Human Memory Prosthesis. *Mik Lamming et al.* Computer Journal, Vol 37, no.3. 1994.

Oppen83 The Clearinghouse: A Decentralized Agent for Locating Named Objects in a Distributed Environment. *Derek C Oppen and Jogen K Dalal* ACM Transactions on Office Information Systems. July 1993.

Porter94 An ATM Based Protocol for Wireless LANs. *John Porter and Andy Hopper* ORL Technical Report 94.2.

Schilit94 Context-Aware Computing Applications. *Bill N. Schilit, Norman Adams and Roy Want.* IEEE Workshop on Mobile Computing Systems and Applications. *December 8-9, 1994.*

Want92 Active Badges and personal interactive computing objects. *Roy Want and Andy Hopper*. IEEE Transactions on Consumer Electronics, 38(1). Feb 1992.

Want95 The ParcTab Ubiquitous Computing Experiment. *Roy Want et al.* Xerox PARC Technical Report CSL 95-1. March 1995

Brown96 Context-aware applications:from the laboratory to the marketplace. *P.J. Brown*. This volume.

Brown,M96 Open-Vocabulary Speech Indexing for Voice and Video Mail Retrieval. *Martin Brown et al.* Proceedings of the Fourth ACM International Multimedia Conference, Boston, Mass. November 1996.

Kent97. Kent Displays Inc., 343 Portage Blvd, Kent, AH 44240 USA

Amulet. The Amulet2e Microprocessor. *Department of Computer Science, University of Manchester*.