The R2 Low-Power Messaging and Rendezvous Layer

Power efficient communications for Tetherless Computing

J.N. Weatherall,
Laboratory for Communications Engineering,
University of Cambridge Engineering Department, Trumpington Street, Cambridge, UK
Email: jnw22@cam.ac.uk

Recent advances in microprocessor and wireless technologies have created scope for a new class of device – the *tetherless* computer. Tetherless computers encompass not only those devices traditionally regarded as computers but also other electronic household devices. Tetherless computing may even extend, as will be illustrated, to networking non-electronic home items. The R2-Layer provides a simple yet flexible low-power rendezvous and messaging interface suitable for tetherless devices.

1 Introduction

Twenty years ago, your television set would have been controlled by a set of knobs on its front panel and may well have been supplied in a plastic wood-effect case. Your radio would have generally been mono rather than stereo and would require tuning manually. The volume of their output would have been controlled by a variable resistor knob labelled "volume" and connected directly into part of the amplifier circuit. If you had a telephone then it would have a dialling wheel.

Ten years ago, a television set would have been supplied in any colour you wanted provided it was black and would be operated by a small panel of buttons linked to a simple microprocessor. The device would still require manual tuning but be capable of scanning the broadcast signal for available channels. Radios would now be stereo for the most part and some would be able to tune themselves automatically. They would now be controlled by a set of buttons linked back to a microprocessor, as would your telephone.

Five years ago, your television, video recorder, and hi-fi systems would have each have had their own microprocessor-based infrared remote controller. The controllers would generally have a similar set of buttons on them to the ones on the devices themselves, rather than adding any extra functionality. When people asked you for your telephone number, you might now have to get them to specify whether they wanted your home, office, or mobile number.

Two trends are apparent if one looks more closely at the above observations. The first is that embedded processing technology is becoming cheaper and at the same time more powerful. The second is that, more recently, wireless technologies have become cheaper and more reliable and hence have become more popular.

2 Motivating Applications

Given the trends described above, it seems reasonable to expect that, within the next ten years or so, more and more devices in the home will become both microprocessor-controlled and network-aware. On the surface it might well appear that the technology transition required will be smooth and painless. The following example situations illustrate why this is not actually the case.

2.1 Multimedia Management

Increasingly popular in the home, multimedia allows users to benefit from a wide variety of information sources in a unified way. By splitting traditional media equipment such as hi-fis, televisions and recording equipment into their fundamental components, such as speakers, visual displays and storage media, their functionality can be distributed and recombined in more useful ways.

Placing the fundamental components on a shared network allows them to be combined flexibly but this raises the issue of how to specify the connections to be made. In addition, there are devices such as headsets and user interfaces which should not be attached to a wired network in order to avoid *tethering* the user to the devices they are using.

2.2 Home Monitoring

In a home environment there are a wide variety of things that are not traditionally networked, in most cases because the cost of making them networked would far outweigh the benefits.

A good example might be a home-owner's house-plants. Using the availability of cheap sensors capable of interacting with other devices, a user can place moisture and alkalinity sensors on all the plants in their home and configure them to notify the user when the plants need watering, or need their soil replacing.

The availability of cheap sensor technology makes it possible for the user to be alerted to non-computer-related problems that need their attention without requiring them to continually monitor their home themselves.

2.3 The Active Bicycle

In addition to the standard set of lights featured on most

bicycles, the Active Bicycle also features a range of sensors and a small LCD display, all powered by either long-life batteries or solar cells.

To augment the sensors commonly found on bicycles, such as speedometers and odometers, the owner can simply buy additional wireless sensors for factors such as humidity, temperature, wind speed and pollution level, and attach them to the bike.

The sensors all communicate back to the LCD panel to display their status and the panel also allows the bike's lights to be manipulated easily from its position on the handlebars.

When the cyclist approaches the bike, a device upon their person, perhaps a ring, watch or pager, communicates with the bike and arranges that pager messages and travel warnings received by the cyclist's own devices appear on the bike's display. The cyclist's devices may also log status information from the bike's sensors and report, for example, how fast the user travelled or how polluted the route they chose was.

3 Related Work

3.1 HAN, HAVi and Jini

The Home Area Network[HAN99] project aims to unify access to home devices by attaching them to a home-area ATM network. Doing so allows individual multimedia streams to be routed flexibly around the home in order to support, for example, roaming music or videoconferencing applications.

The HAVi[HAVi98] architecture is a standard designed to allow devices connected to an IEEE Std. 1394 "High Performance Serial Bus" network to interact in a uniform way. Its design is tailored to allow interoperation of home audio and video devices, building on several established standards including Java bytecodes.

The Jini[Jini99] project also uses Java bytecodes to provide extensibility of the system software. Like HAN and HAVi, it too relies on the presence of a shared home network.

All of these projects support the application of multimedia management well, as might be expected. However, the need for each networked device to be wired to the others and the associated power and processing requirements make them unsuitable for home monitoring or for ad hoc applications such as the bicycle.

3.2 X10 and Jini

The X10[X10] home control architecture avoids the need for extensive network wiring in the home by piggy-backing control signals on the existing mains power wiring. The result is a low-bandwidth home network ideal for remote and automated control of powered devices.

3.3 BlueTooth

BlueTooth[Blue99] networks consist of clusters of cooperating devices communicating over a shared radio channel. The radio reaches only a few metres and is accessed using spread-spectrum techniques, allowing bandwidths sufficient to support several audio streams. BlueTooth networks could therefore be used in all three applications, owing to their wireless ad hoc nature. The power requirements of BlueTooth make it unsuitable for tasks involving devices will not be recharged regularly.

3.4 PEN

AT&T Labs' Prototype Embedded Network (PEN – previously Piconet)[Pico97] project uses short-range, low-bandwidth, low-power radio to transmit small amounts of control and state change information between devices.

Although the system has inherently low power requirements, making it a suitable target platform for all three applications, it still requires too much power for embedded sensor applications if left continually active. This limitation can be overcome using suitable power management schemes to shutdown the radio hardware when not required.

A key decision in the design of the system was to use peer-to-peer communications, with no base-stations or "master" devices. As a consequence, the power tradeoffs to be made differ from those found in many existing cellular networks.

4 R2-Layer Modes of Operation

Two types of interaction may be identified from the example applications described previously. Certain system components will naturally operate using unsolicited transmissions, initiated and controlled by the sender. Other components will more closely fit a solicited model, in which one device broadcasts data in a well-known way and it is up to receivers to control receipt of the data, as and when they require it.

In order to exploit both techniques and thus obtain the best possible combination of flexibility and power saving, the R2 Low-Power Messaging Layer was designed.

The R2-Layer forms a generic but flexible low-power radio interface on top of which application-specific protocols may be constructed. Fundamental to the design of the R2-Layer is the notion of there being two basic modes of operation for ad hoc communication systems;

4.1 Transmitter-Managed Mode

Traditional ad-hoc, low-power communication layers such as the PEN R-Layer [RLayer2000] tend to be *transmitter-managed*. This means that responsibility for ensuring rendezvous between two devices to exchange data lies with the device wishing to transmit rather than the target device.

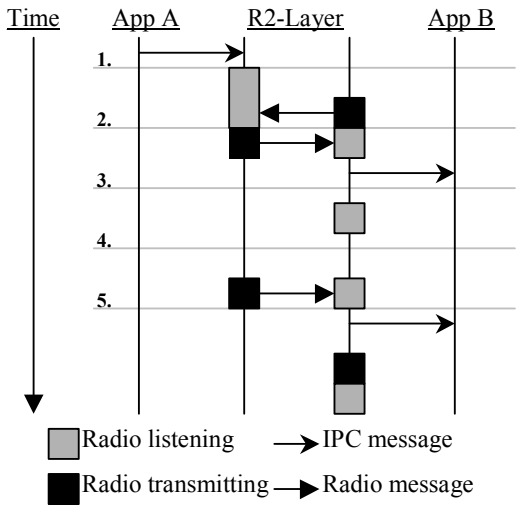
Using the R-Layer, for example, devices wishing to receive data transmit a broadcast beacon packet at regular intervals, which transmitting devices listen for to establish when to send data to the receiver. In this way

the emphasis is very much on the party wishing to send data taking responsibility for coordinating the data transfer.

While a transmitter-managed system works well for RPC-style interactions, it is less suitable for interactions involving monitoring of state. In a transmitter-managed environment each sensor, for example, must be aware of each and every device to which it must send state-change updates. In order for a sensor to be aware of interested devices each device receiving sensor data must previously have sent a request to the sensor to register itself. Sensors must therefore accept incoming radio requests and receivers must beacon so that sensors know when to transmit data to them. As more receivers are added the computational, radio bandwidth and power loads on the sensor increase.

Figure 1 presents an example of transmitter-managed operation.

Fig 1. Transmitter-Managed Communication



1. Application A queues a message with the R2-Layer, to be transmitted to application B.
2. The sending device powers up until a beacon is received from the target device.
3. The sending device transmits the data to coincide with the receiver's listen window.
4. No further data is queued so the transmitter remains inactive.
5. A queues more data to be sent to B. The R2-Layer transmits the data in the next available listen window.

4.2 Receiver-Managed Mode

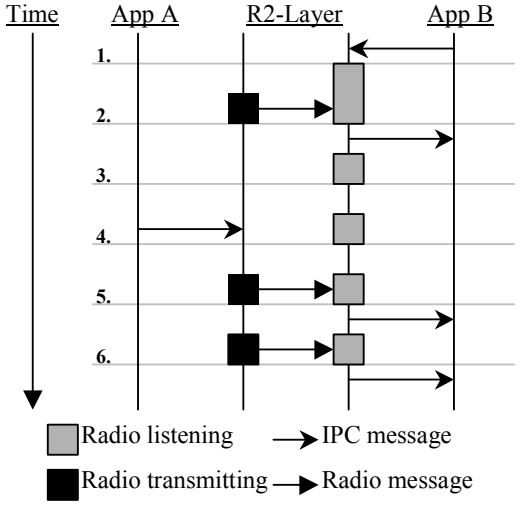
Systems involving a large amount of monitoring of the state of distributed devices may be better suited to a receiver-managed mode of operation.

In this mode the sensors embed their state information into each beacon packet they transmit. Actuator devices such as lamps, which wish to monitor a light sensor, for example, are responsible for ensuring they power up their radio at suitable intervals to catch the sensor's beacons.

To support sensor applications in which state is consistent for prolonged periods but may change at a moment's notice, optional transmission slots provide a way for sensors to both conserve power and deliver fast response times.

Figure 2 illustrates the operation of a receiver-managed scheme.

Fig 2. Receiver-Managed Communication



1. Application B informs the R2-Layer that it wishes to receive beacons from application A.
2. A's R2-Layer beacons on its behalf. B's R2-Layer catches the beacon and passes it to B.
3. B listens for A's optional transmission slot but A has no new data to transmit and does not use the slot.
4. A passes a new beacon payload to the R2-Layer.
5. The R2-Layer transmits the new payload to B in an optional transmission slot, between beacons.
6. The R2-Layer transmits the normal beacon packet.

The sensor no longer needs to accept registrations of interest, nor keep a table of interested parties, nor send the same data multiple times. Because the protocol now requires no incoming messages to be dealt with by sensors, they can be simplified considerably by completely removing the reception side of their implementation.

A receiver-managed system thus provides far better computational, bandwidth and power scalability to large numbers of receiving devices. The primary disadvantage of this approach is that there is now no way to detect transmission failure to the receivers, since multicast packets have no acknowledgement mechanism[1].

5 R2-Layer Structure

The prototype R2-Layer implementation has been built on the PEN platform. The PEN platform features a variable amount of SRAM and Flash-RAM, a 16-bit CPU and some extremely primitive radio hardware, in its most basic form. The R2-Layer is designed,

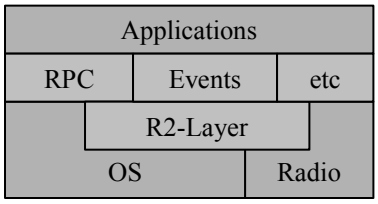
[1]It is assumed that sensor devices will generally be too primitive to support reliable multicast algorithms. Such devices may even be incapable of receiving acknowledge packets.

however, to be suitable for use with other shared-access media, such as infrared.

In order to support the widest possible variety of PEN devices, the R2-Layer implementation is divided up into a set of small modules, so that each device need only include the R2-Layer code necessary to its operation.

Each R2-Layer module provides APIs serving a different aspect of the protocol's functionality. The modules are factories for the relevant component classes, allowing multiple higher-level protocols to co-exist over the R2-Layer on a single device, as shown in Figure 3.

Fig 3. An Example R2-Layer System Structure



The modules' APIs are kept as concise as possible, to avoid creating artificial dependencies between unrelated code fragments. For example, the lowest level radio power management, transmission and reception APIs are separated into three distinct modules – R2POWER, R2TX and R2RX. Although the R2TX and R2RX modules cannot operate without the presence of the R2POWER module, which is required to power the radio hardware in order for communication to take place, its functionality doesn't intrinsically belong to either of the R2TX or R2RX modules and bundling it with either one of them would limit the flexibility of the system.

5.1 Receiver-Managed Modules

Three modules are directly involved in dealing with receiver-managed communication.

The R2TX module is the most fundamental module required by simple transmitter devices. It is responsible for powering the radio hardware up at user-specified intervals and transmitting broadcast beacon packets. R2TX objects include methods to alter the beacon interval and to allow user-specified data to be attached to the beacon packets. The R2TX module depends upon the R2POWER module and the Operating System's radio interface.

Clients wishing to use the optional-transmission slots of the R2-Layer protocol create both an R2INFORM and an R2TX object, each with their own interval defined. When the beacon payload is to be changed, the new payload is passed to the R2INFORM object, which passes it down to the R2TX module and also queues it to be sent in the next optional-transmission slot. If the payload doesn't change then only the normal beacon packets will ever be sent.

Receivers create an instance of the R2WATCH module and specify the addresses of other devices they wish to receive beacons from. The R2WATCH module will then attempt to power the radio only when beacons from "watched" devices are expected, in much the same way as a FLEX [FLEX99] pager must synchronise with the base-station to catch pager notifications at regular intervals.

The client may control whether or not the R2WATCH module also watches for optional-transmission packets.

5.2 Transmitter-Managed Modules

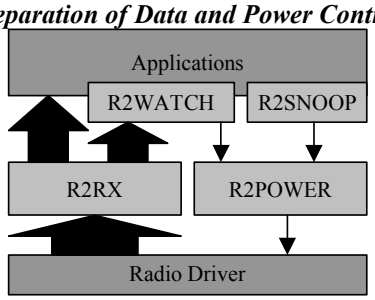
Two modules are involved in transmitter-managed communication.

The R2RECEIVE module is used by the receiving device to control power management of the radio. The client creates R2TX and R2RECEIVE objects and specifies an interval to each. The R2RECEIVE instance will advertise "listen windows" at the desired intervals, via the R2TX object's beacons. It will power up the radio hardware to satisfy these advertisements but will not actually accept incoming data. Instead, clients register themselves directly with the R2RX module and receive incoming packets from that.

The power management and packet unmarshalling tasks are kept completely distinct, as shown in figure 4.

The transmitting device uses R2SEND objects to transmit packets to other devices. The R2SEND module requests access to incoming beacon packets via the R2RX module and uses the R2POWER module to power up the hardware. Once a beacon is received from the target device, the R2SEND module forwards the queued data packet and powers the radio hardware down again.

Fig 4. Separation of Data and Power Control Paths



5.3 Other Modules

In addition to those described above, modules are provided to perform other services such as ad hoc device discovery.

A client wishing to be notified when new devices arrive creates an instance of the R2SNOOP module, as shown in figure 4. This module powers the radio up and down in such a way as to ensure that at least one beacon from each local device will be heard within a specified interval. The client must also register with the R2RX module in order to actually receive the incoming beacons.

6 R2-Layer Performance

6.1 The Light Sensor and The Lamp

Any wireless solution to the tetherless computing

problem must be extremely low-power and capable of operating for prolonged periods on a set of betteries or, better still, using parasitic power, perhaps from solar cells.

To illustrate an environment in which the R2-Layer may out-perform similar transmitter-managed communication schemes, the Light-Sensor and Lamp example was constructed.

In this example, a simple light sensor is attached to a wireless device, which takes readings from the sensor and makes them available over the wireless channel to nearby lamps, so that they can switch on when it gets dark.

The following sections evaluate the relative performance characteristics of the three tested implementations. All values are approximate and based on short-term observed behaviour. Effects such as long-term power cell deterioration, for example, are not taken into account.

6.2 Prototype #1: Reference Implementation

The first implementation was primarily intended to demonstrate that the PEN hardware platform and operating system functioned correctly.

Sensors keep a table of interested lamps, to which updates must be sent. Lamps register using a custom, RPC-like protocol.

Both devices keep their radio hardware powered at all times, requiring approximately 30mA of current.

In order to respond to incoming packets in a timely fashion, the CPU must also remain powered at all times, draining a further 30mA. While the CPU is operating it will be accessing off-chip memory, incurring a further penalty of 25mA.

With a steady drain of around 80mA, a single 9V cell will power lamp or sensor for 18 hours on average.

6.3 Prototype #2: Power Management

The second prototype aimed to demonstrate the effectiveness of the "Rendezvous Layer" or "R-Layer", a power-aware messaging layer for PEN. This prototype uses the same protocol & design as the original prototype, but running over the R-Layer power management software.

To obtain usable response times, the lamps were set to beacon every 2s, to receive updates, and the sensors every 10s, to receive registrations. Each beacon consists of transmit and receive phases and takes around 50ms. At all other times, the radio, CPU and main memory are powered down and the device drains only 60µA. Figures 5 & 6 illustrate the resulting performance.

Every time the light level changes, the sensor will need to power up for anywhere between 50ms and 2000ms, in order to transmit a reading to the client lamp. As more clients are added, the average time required to send the message to all clients tends towards the 2000ms limit.

Because readings are marshalled and then queued for up to 2000ms before being transmitted, they may be correspondingly "stale" upon receipt.

Fig 5. Sensor performance - static & variable light

Sensor performance	Static	Variable
Live time (ms)	50	-
Interval (ms)	10000	-
Average Current (mA)	0.5	80
Lifetime (days)	~150	0.75

Fig 6. Lamp performance

Lamp performance	
Live time (ms)	50
Interval (ms)	2000
Average Current (mA)	2

6.4 Prototype #3 – Event Distribution

The third implementation uses the R2-Layer's receiver-managed mode to both simplify the sensor device and improve the system's power consumption (Figures 7 & 8).

Sensors beacon for 20ms every 10s and have optional transmission slots every 2s. Each beacon contains the current light level reading. Because the sensors beacon but do not need to listen for registration messages, they need only power up for around 20ms for each beacon.

Fig 7. Sensor performance - static & variable light

Sensor performance	Static	Variable
Live time (ms)	20	20
Interval (ms)	10000	2000
Average Current (mA)	0.22	0.86
Lifetime (days)	~365	~100

Fig 8. Lamp performance

Lamp performance	
Live time (ms)	30
Interval (ms)	2000
Average Current (mA)	1.2

Similarly, lamps now power up to receive readings every 2s but are not required to transmit beacons, reducing the required "live" time to only 30ms.

If the light level is changing rapidly then the sensors will transmit in every optional-transmission slot. If more lamps watch the same sensor, no extra load is added and no extra bandwidth is required.

The sensor can now be implemented on a tiny PIC and operated from a solar cell. The readings received by lamps are no longer "stale" as in prototype #2.

For complex devices, state changes could still be propagated using the original RPC scheme, if required. The benefits to simple devices gained in using beacons to convey state changes offset the small degree of extra complexity required in the client devices, and the difference in implementations can be made transparent to applications through appropriate abstractions if necessary.

7 Protocol Mapping

The primary motivation behind the development of the R2-Layer was the desire to support high-level distributed computing abstractions such as RPC and Distributed Events in an efficient manner, suitable for embedded devices.

By augmenting an RPC Interface Definition Language such as that of OMG's CORBA[OMG] suitably, we can allow application programmers to convey implicit semantic information in interfaces, through the use of alternative abstractions such as Distributed Events. The presentation layer can then use this extra information to decide how to best use the available communications schemes. Because this avoids having the application access lower level system components directly, the layered structure of a conventional network stack as described by the OSI reference model is not violated.

For example, consider a simple OMG IDL interface such as:

```
interface VerySimpleSensor {
        readonly attribute Units Value;
};
```

Given this interface, an IDL compiler for a simple embedded target device could produce code which would embed the sensor's Value in beacon packets, allowing nearby devices to retrieve the value using receiver-managed interaction and thus reduce the load on the sensor.

Such a technique could be used to allow the components of the Active Bicycle, described previously, to communicate with each other and to be accessed by standard ORBs on palm- or lap-top computers as if they were fully equipped objects.

Conventional mappings of similar interfaces would require the sensor device to accept incoming RPC requests and to respond to each query individually, hence suffering from the limitations of low-power RPC previously described.

8 Summary

In this paper we have described the notion of Tetherless Computing and have illustrated it with several example applications. The difficulties involved in implementing these applications efficiently using existing technologies have motivated the design of a new low-power communications layer, upon which a wide variety of conventional distributed computation abstractions may be built.

The resulting protocol allows even resource poor, power-starved and computationally limited devices to interact on a peer-to-peer basis with more standard computer hardware.

By using suitable mappings of standard abstractions over this low-power transport, applications can tailor the transport's behaviour to their requirements without the programmer having to perform menial power management, rendezvous and marshalling tasks directly. Not only does this simplify life for the programmer but it reduces the scope for mistakes at low levels of an embedded system and increases the likelihood that disparate devices will be able to communicate, even if only at a basic level.

9 Acknowledgements

The author wishes to thanks Andy Hopper[2] and Sai-Lai Lo[3] for their time in supervising this work, and the PEN project group at AT&T Labs Cambridge for their support. Thanks also to Scott Mitchell for his proof-reading skills.

10 References

[Blue99] - *"Specification of the BlueTooth System – CORE",* from www.bluetooth.com, 1999

[FLEX] - *"FLEX White Papers",* from http://www.motorola.com/MIMS/MSPG/CTSD/white_papers/flex_white_paper.html, 1997

[HAN99] – *"Home Area Networks"*, from http://www.cl.cam.ac.uk/Research/SRG/netos/han, 1999

[HAVi98] – *"HAVi V1.0 Specification",* from http://www.havi.org, 1998

[Jini99] – *"Jini Architecture Specification",* from http://www.sun.com/jini/specs, 1999

[Pico97] – *"Piconet – Embedded Mobile Networking",* F.Bennett, D.Clarke, J.B.Evans, A.Hopper, A.Jones and D.Leask, IEEE Personal Communications, Vol. 4, No. 5. pp 8-15. October 1997

[OMG] – *"CORBA Specification",* from http://www.omg.org, 1999

[RLayer2000] – *"Low Power Rendezvous in Embedded Wireless Networks",* T.Todd, F.Bennett and A.Jones, to be published in proceedings of MobiHOC 2000.

[X10] – *"X-10 Transmission Theory",* from http://www.x10.com/technology1.htm, 1998

[2] Laboratory for Communications Engineering, Cambridge University Engineering Department.

[3] AT&T Labs Cambridge, Cambridge, UK.