Mapping of Services on Bluetooth Radio Networks

J. Dunlop and N. Amanquah
University of Strathclyde -Department of Electronic and Electrical Engineering, Glasgow G1 1XW, Scotland

Ph.: +44 141 5482081, Fax:+44 141 5524968,
e-mail: {j.dunlop,nathan.amanquah}@strath.ac.uk

ABSTRACT

With the emergence of many short range wireless connectivity solutions it is expected that a wide range of devices will become connected together to form a sea of networked devices. Such an environment has considerable potential for service provision and Bluetooth has been specifically designed for operation in such an environment. However networks of this type must be equipped with the means to find and configure both resources and services when both are in a changing environment. Device and service discovery technologies are thus very important for successfully networking diverse services on ad hoc networks. Bluetooth is limited in this regard but such functionality is provided by Jini, which is a proprietary distributed computing solution offering "network plug and play" features. This paper examines the main issues to be addressed when systems such as Bluetooth and Jini are combined, and describes scenarios which have been built into a developing testbed facility designed primarily to investigate middleware functionality in ad-hoc radio network environments.

1 BLUETOOTH SYSTEM ARCHITECTURE

Bluetooth[1] is a technology specification for small form factor, low-cost, short range radio links between mobile PCs, mobile phones and other portable devices. Bluetooth technology allows for the replacement of the many proprietary cables that connect one device to another by a short-range radio link. It is designed to operate in radio environment with a high level of interference and uses a fast acknowledgement and frequency hoping scheme to make the link robust. Bluetooth operates in the unlicensed Industrial, Scientific and Medical (ISM) band, which is centred around 2.45 GHz.

The Bluetooth system supports both point-to-point and point-to-multipoint connections. Several Piconets [3] can be established and linked together in an ad-hoc fashion in which each Piconet is identified by a different frequency hopping sequence. The topology is described as a Scatternet or a multiple Piconet structure as shown in figure 1.

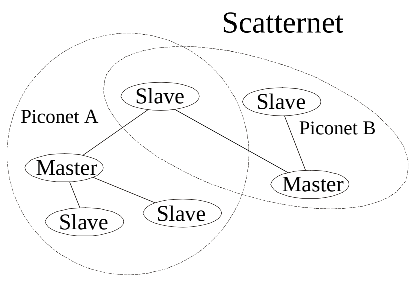

Figure 1: Bluetooth topology

Up to 10 piconets can coexist in the same location. The maximum asynchronous data rate within a Piconet is 723.2kb/s. The Bluetooth protocol stack [4] is illustrated in figure 2.

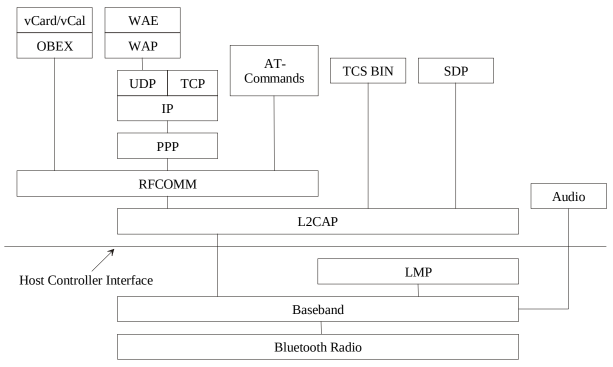

Figure 2: The Bluetooth protocol stack.

The core of the Bluetooth protocol architecture comprises a set of three protocols – the link control and adaptation protocol (L2CAP), the services discovery protocol (SDP) and the RFCOMM protocol. Device information, services and service characteristics can be queried using the SDP. As with SDP, RFCOMM is layered on top of the L2CAP. As a cable replacement protocol, it provides transport capabilities for application-level services.

2 THE NEED FOR AN APPLICATION PROGRAMMING INTERFACE (API)

Bluetooth enables connectivity of a large range of devices with a significant variation in software support capabilities. Additional advantages of Bluetooth are the small form facto and low power consumption. Unlike many other wireless links/technologies, Bluetooth also incorporates a service discovery protocol (SDP 1.0). Thus Bluetooth devices can intrinsically discover what services have been registered on a remote device by inspecting the SDP Database on that device. In the Bluetooth environment, where the set of services that are available can change dynamically based on the RF proximity of devices in motion, service discovery is qualitatively different from that in traditional network-based environments.

However, SDP 1.0 is limited as it provides access to information about services rather than access to the services themselves. Further the Bluetooth SDP 1.0 does not provide brokering of services, negotiation of service parameters or an event notification when services, or information about services, become unavailable. Thus there is a need for a session layer protocol that can establish a session between the high level applications where the brokerage of the services and essential negotiation of the service parameters may take place. Jini, which is a Java distributed computing component infrastructure offers a potential solution to this problem.

3 JINI TECHNOLOGY ARCHITECTURE

The Jini[2] architecture provides a distributed environment for service look-up, registration, and leasing. To run a Jini service, three components are required: A web server to host downloadable code for clients, a Remote Method Invocation (RMI) activation daemon that invokes long lived services when they are needed and allows them to switch to an inactive state between calls, and a Lookup Service that manages a registry of available services. Jini allows services to be easily added to and removed from a network or a co-operating community without any significant change in configuration. Participants in the network can be notified of changes to the available services if desired.

Jini is able to create a style of “plug-and-play” environment for a variety of devices and software components on a network. Services that join the community carry the code needed to use them, thus avoiding the need to install drivers and configuring before using a service. Such code is dynamically uploaded to a lookup server (service locator) when a service registers its proxy with one. A Service Item consisting of the proxy and a record of searchable attributes is kept on the lookup service. The proxy is then available to any client that may search for it and can be downloaded when a client looks up a service from a service locator. A proxy may have all the intelligence to provide a service, or contact a backend device/service by a proprietary protocol (if desired) or by RMI in order to provide the service.

Clients search lookup services for services that implement a particular interface, by a unique service ID or by inspecting informative service “attributes”. Both clients and services search for lookup services by means of Multicast or Unicast protocols. The latter is used when the location/URL of the service is known before hand.

The Jini programming model is shown in figure 3. In an operational Jini system there are 3 main components, the *Client* which requires the service, the *Service* (a trivial example would be a printer) and the *Service Locator* which acts as a locator/broker/trader between clients and services. In addition there will be a *network* which interconnects these components which will generally be running TCP/IP. In the context of the Jini system a service is a logical concept which will normally be defined and identified by a Java interface.

Referring to figure 3 when the service locator receives a request for a service from a client or a request for registration from a service provider it returns an *object* known as a registrar that acts as a proxy for the service. The proxy will then communicate with other objects in the service provider using a protocol supported by the Java RMI.

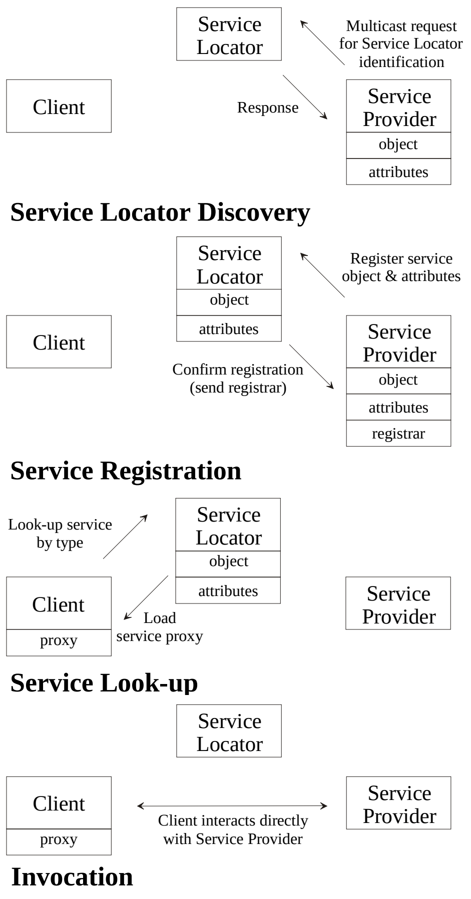

Figure 3: The Jini programming model

4 BLUETOOTH-JINI COMPLEMENTARY PAIR

Jini readily interfaces with the Bluetooth protocol stack at the TCP/IP-UDP layer. The SDP1.0 along with the Look-up Locator of Jini can establish the necessary platform for the registration and brokering of services. Bluetooth and Jini are potentially a complementary pair in service discovery and utilisation, particularly in an ad hoc network. In such a network, there is no centralised body or node to administer network addresses assigned to each participating node. TCP/IP is required for Jini and Java applications to run. To establish a TCP/IP connection, the Bluetooth LAN access profile, may be used. This is dependent on the RFCOMM (the Bluetooth adaptation of the GSM TS 07.10 for serial port emulation) and PPP protocols, which work just as they would in the Serial port profile. This profile is suitable when one device is configured as a LAN Access Point, and the client as a Data Terminal. Alternatively, the Dial Up networking profile may be used when one node has been configured as a “gateway” node. The RFCOMM (serial port connection) in conjunction with PPP may also be used directly but this may require a Dial-up server and Modem Emulation. The mode of connection will depend on the profiles implemented in the two devices establishing a link.

The LAN Access profile supports single or multiple devices or PC to PC PPP networking over serial port emulation. A LAN Access Point provides access to the

LAN by providing the services of a PPP server, and connection is over an emulated serial port (the RFCOMM) Using appropriate PPP mechanisms, a suitable IP address is assigned to the client. The PPP over RFCOMM used in this profile differs from using PPP in the Dial-Up Networking Profile in that no AT commands are used in establishing the link.

A Jini service or client may discover services, which are "near" to it in network terms. One method used is the lookup discovery, which depends on multicast of packets to addresses within the same subnet. This would include both Bluetooth enabled nodes and wired terminals that share a similar address space governed by their IP Address Subnet Mask. Packets may not be forwarded to nodes on other segments of the network using a different subnet even if they were attached to the same physical medium. This is as a result of using a multicast-based protocol. The other discovery method- the lookup locator, may locate services on different subnets provided the full IP address or URL to this location is given.

For security considerations or by virtue of network configuration, a service may not be available on the local IP subnet using multicast methods. Bluetooth devices will be able to establish a connection not withstanding the IP addresses assigned to the hosts. An entry may be made in the Bluetooth SDP to indicate the URL and port where the Lookup server or other service may be contacted by unicast methods.

5 SDP CLIENT-SERVER INTERACTION

The service discovery protocol of Bluetooth provides the means for client applications to discover the existence of services provided by server applications as well as the attributes of those services. The attributes of a service include the type or class of service offered and the mechanism or protocol information needed to utilise the service. This is illustrated in figure 4.

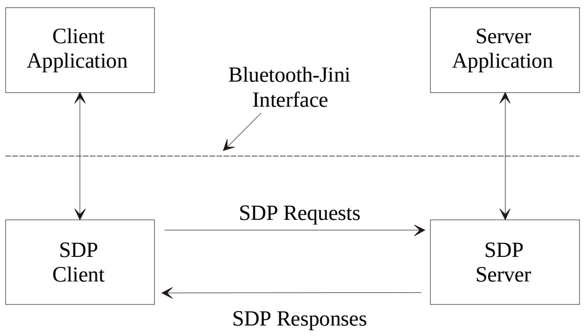

Figure 4: SDP client-server interaction

There is a maximum of one SDP server per Bluetooth device which may function both as a SDP server and as a SDP client irrespective of the number of applications on the device providing service.

6 COMMUNICATION SEQUENCE

This section describes the communications sequence designed to implement an applications programming interface which interacts with the Bluetooth service discovery protocol. The SDP client will discover the services of the SDP server using the Service Discovery Protocol. The Jini application running at the client side will manage the search at both ends as well as the start and finish of the services at the application level.

Step 1: The Client Side Java Application, figure 5, prepares the Entry forms for identifying the service in the network. These Entry Forms are then passed on to the SDP Client after appropriate mapping at the Bluetooth – Jini Interface.

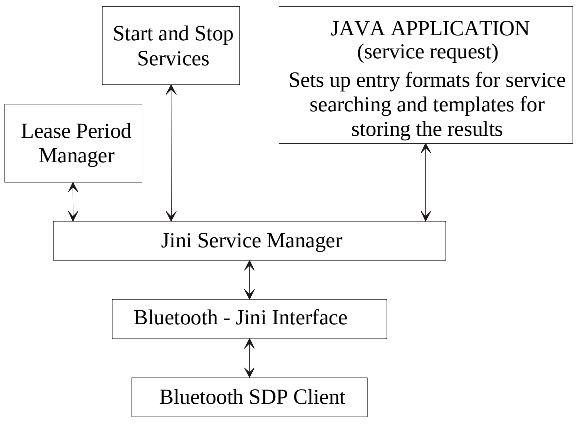
Start and Stop Services
JAVA APPLICATION (service request) Sets up entry formats for service searching and templates for storing the results
Lease Period Manager
Jini Service Manager
Bluetooth - Jini Interface
Bluetooth SDP Client

Figure 5: Client side of a Jini Application

Step 2: The SDP Client searches for the service in the network using the SDP of Bluetooth.

Step 3: The service information registered with the SDP Server, shown in figure 6, matches the request sent by the client and sends a confirmation upon the successful match. (It should be noted that the SDP Server does not have to refer to the higher end Server side Jini Application at this stage.)

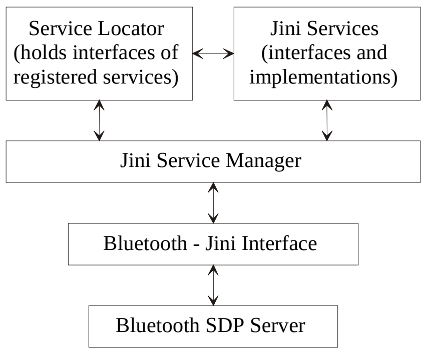
Service Locator (holds interfaces of registered services)
Jini Services (interfaces and implementations)
Jini Service Manager
Bluetooth - Jini Interface
Bluetooth SDP Server

Figure 6: Server Side of a Jini Application

Step 4: The Search results are then sent back to the SDP Client by the SDP Server (after a successful match). These results are passed back to the Client Side Application and stored in the templates created by the Java Application.

Step 5: A request for the service arrives from the Jini Service Manager through the Java Application. A suitable TCP/IP based connection is then established to the Server and then the control is carried out in the higher layers.

Step 6: After initial negotiations (i.e. lease period and other such quality of service parameters) the Jini interface is sent to the client end.

Step 7: The Start and Stop module of the Jini Service Manager will pass the start and stop parameters to the Server and the service is then run either at the Server or Client side depending on design of the service. This may require marshalling and un-marshalling of results or objects as appropriate.

Step 8: After the successful execution of the service, the client ends the contract (a typical sign off) and hence the interface implementation is completed and the lease manager finally submits the lease contract back to the server. The signalling exchanges associated with steps 1 to 8 are shown in figure 7.

Jini can handle multiple client requests. Jini also accommodates less well behaved applications (for example applications that do not do perform a normal shutdown, or simply move out of radio range) by means of its leasing technology. Jini can be employed to monitor Bluetooth links so that when the lease on a Jini service expires and is not renewed, the associated Bluetooth connection may be closed and resources released or suspended. Furthermore, Jini uses event notification to update the status of a client or service when there are changes to the available services and configuration. Some Bluetooth enabled devices may not have enough resources to run a Lookup Service and services required to run a Jini service. Nevertheless, a Jini application may still access such information on such services using a subset of the steps described.

7 Experimental Testbed

The testbed described in this paper is being developed to assess middleware functionality in ad-hoc radio networks. To assess the signalling exchanges proposed in figure 7, two networked PCs and two Bluetooth enabled PCs were set up to mimic different clients and services. (Scenarios implemented are envisioned to include several Bluetooth enabled devices.) A web server, a remote method invocation (RMI) daemon, and a service locator were run on one Bluetooth enabled PC that acted as a server. The client PC only runs a client Jini application. The arrangement implements the following scenarios (depicted in figures 8 and 9 respectively):

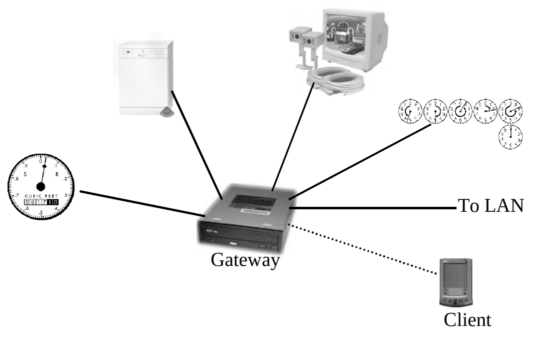

Figure 8: Wireless link between Client and Gateway

a) A Bluetooth enabled PC acting as gateway is connected by a LAN to two other PCs that offer or implement services. Two utility services (e.g. electricity and gas), a simple security system and an electrical appliance can be accessed from this gateway. The client (e.g. an emulated PDA) may browse for the list of services available, make a choice from the three presented, bind to, interact with and control the service selected.
b) A Bluetooth enabled PC acts as a gateway, and a second Bluetooth enabled PC acts as a controlled device, with a Bluetooth link to the gateway/server. A client may interact with the service offered by the second PC by accessing and interacting with the proxy on the gateway. The proxy in turn interacts with and controls the service remotely over the Bluetooth link. The testbed was restricted to one wireless link between the gateway and a service, other links being wired.

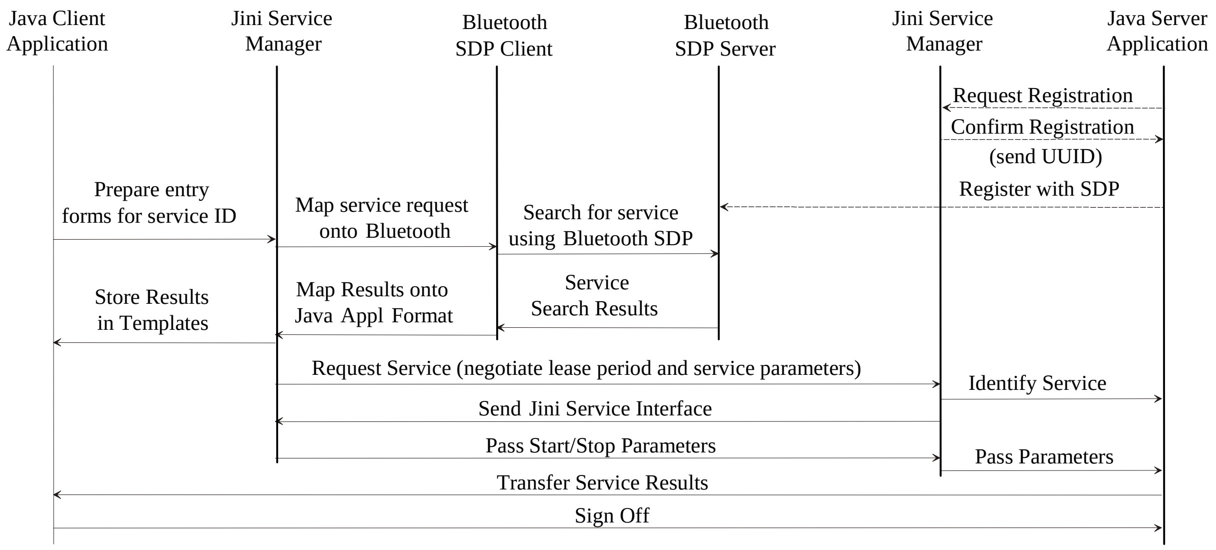
Java Client Application
Jini Service Manager
Bluetooth SDP Client
Bluetooth SDP Server
Jini Service Manager
Java Server Application
Request Registration
Confirm Registration (send UUID)
Register with SDP
Prepare entry forms for service ID
Map service request onto Bluetooth
Search for service using Bluetooth SDP
Store Results in Templates
Map Results onto Java Appl Format
Service Search Results
Request Service (negotiate lease period and service parameters)
Identify Service
Send Jini Service Interface
Pass Start/Stop Parameters
Pass Parameters
Transfer Service Results
Sign Off

Figure 7: Signalling exchanges in Jini/Bluetooth environment

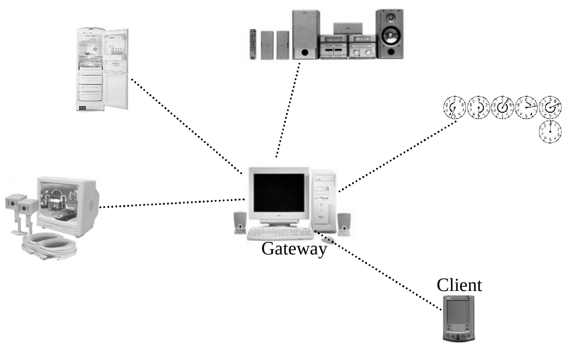

Figure 9: Wireless links between all devices and gateway.

In case a) both unicast and multicast methods are used. To establish a connection, the signalling sequence in Figure 7 is followed. Two options were explored for service discovery:

i) A client first fills out a template with a query for the existence of a Lookup Service on the remote Bluetooth device and checks if TCP/IP connection may be established with this device. If a link on a suitable profile can be established, and a Lookup Service is available directly on this Server, a TCP/IP link is set up. The IP address (or URL) of the look up service is read directly from the SDP and a unicast request is made to the Lookup service. Next, the Jini client interrogates the Lookup service for Jini services desired and binds to a service for a renewable lease period.

(ii) The Client formulates a request for a given service X. This is mapped to the local Bluetooth SDP (as depicted in Figure 7), which in turn queries the remote Bluetooth SDP for that particular service as well as the existence of a Jini Lookup service. If the service X is not available at this location a search can be performed on other devices. This option requires that all Jini services available at a Bluetooth enabled device register with the Bluetooth SDP so that remote Bluetooth devices may contact them. A "gateway services manager" can be made responsible for registering all services (local and remote), which may be accessed from this point, on the service locator with the local SDP. This is useful in an ad hoc network as a client will not then need to search all devices in range for a desired service.

A set of primitives for interacting with the SDP were defined as a Jini class which applications that wish to interact with the SDP extend. Other functions that abstract commonly used functions are also provided. These were employed in the scenarios described and found to work as desired. A number of Jini (IP based) services were created and run independently. A Bluetooth enabled device (mobile terminal) was brought within radio range. A successful service discovery revealed the services available on the network. The client could then make a selection from those presented and bind to the service.

The average time (over 5 sessions) to initialise the Bluetooth system, and perform an inquiry is 13.97s. The inquiry period is a fixed 12.8s but can be optimised. A further 3.2s is required to establish a TCP/IP session. It takes 0.45s to query the remote SDP for four service names matching a query (76 bytes of data returned) and 0.45s to write 15 attributes into the local SDP. A server takes 7.11s to start and register with the Lookup service compared to 1.44s over a wired LAN. The corresponding times taken by a client are a subject of further work. These values have been obtained from a typical Jini service, without optimising either the performance of the Bluetooth link or the Jini application.

In this testbed, a proprietary network profile was used in lieu of using the LAN access profile or Dial Up Networking profile in conjunction with a PPP server or LAN Access Point at the receiving end. Either would provide the TCP/IP connection that was desired. The relative merits of using unicast Lookup Locator or multicast Lookup Discovery Jini requests on the network is a topic for further work. Further work will also determine the effect of having more than seven devices connected to a gateway, some of which will be in a parked mode at any given time, but all taking part in a Jini community. Multi-hopping over other nodes in order to join a Jini community or to access a Jini Lookup service is an essential QoS support mechanism which will be evaluated as the testbed develops.

8 CONCLUSION

This paper has concentrated on the need for the mapping of evolving services onto the Bluetooth protocol stack. In particular it has drawn attention to limitations of the Bluetooth service discovery protocol and the need to provide an application programming interface. An implementation has been described that interfaces Jini services with the Bluetooth service discovery protocol. By using a set of suitably defined API, applications can be rapidly developed and deployed in a uniform way. The successful implementation of IP based JINI services demonstrates that Bluetooth can be adapted by running a suitable service discovery protocol over it.

This implies that new services can be created and delivered directly to clients by independent third parties which is an important feature for emerging 4G services which will involve trading of information and quality at the point of service access. This work is part of an ongoing study by the UK Virtual Centre of Excellence in Mobile and Personal Communications (MVCE) on middleware functionality in ad-hoc radio networks. The work described in this paper indicates that incorporating limited versions of Jini in embedded devices can serve as a useful platform for combining Bluetooth and Jini in service discovery in the ad hoc wireless network environment.

It is envisaged that Bluetooth will play an important part in emergence of truly virtual mobile network operators and therefore must be able to support the functionality which will be an essential element of service brokering, negotiation of service parameters and quality of service guarantees. The paper has outlined an API based on Java Jini, and illustrated the essential steps required to interface such an API with the Bluetooth SDP and described tested scenarios based on this.

9 ACKNOWLEDGEMENT

The work reported in this paper has formed part of the WA1 area of the Core 2 Research Programme of the Virtual Centre of Excellence in Mobile & Personal Communications, Mobile VCE, www.mobilevce.co.uk, whose funding support, including that of EPSRC, is gratefully acknowledged. More detailed technical reports on this research are available to Industrial Members of Mobile VCE.

10 REFERENCES

[1] Bluetooth Special Interest Group, " The Bluetooth Specification", http://www.bluetooth.com/developer/specification/specification.asp, April 2000.
[2] Sun MicroSystems, "Jini Network Technology", http://www.sun.com/jini, August 2001.
[3] Motorola, "Bluetooth in Action", http://www.mot.com/bluetooth/action/index.html April 2000.
[4] "Bluetooth Overview", http://www.palowireless.com/bluetooth, August 2001.