Ubiquitous Peer-to-Peer Applications in Wireless Ad hoc Networks

Interim Report

Neil Madhvani
nm300e@doc.ic.ac.uk

MEng in Information Systems Engineering
Imperial College, London

January 2004

Supervisor: Dr Naranker Dulay
Second Marker: Dr Emil Lupu

Contents

1 Introduction

The primary objectives of this interim report are to provide a summary of the work carried out so far as part of this ISE4 individual project, to take some time to reflect on this progress, and to propose a roadmap from now until completion in June 2004.

2 Progress to date

This section looks firstly at the original motivation and objectives, as determined in October 2003 when this project was proposed. Following on from this is a summary of findings from the various papers that I have read around this area. We then discuss how the project has evolved and moved into a slightly different direction with more specific objectives.

2.1 Original project objectives

My original areas (mainly developed over the 2003 summer holidays) revolved around the idea of Voice over IP (VoIP) technologies, particularly involving cellular networks. It is apparent that whilst the number of mobile phone users is increasing on a daily basis, and there is growing use of VoIP over the Internet with recently-announced protocols such as SIP (Session Initiated Protocols), very few services currently exist that exploit the synergy between these two worlds. For many applications, such as group conferencing of relatively 'local' users, it simply isn't efficient to deliver voice traffic over the cellular network, due to the relatively long call setup time and excessive bandwidth usage. My view is that a 'best-effort' walkie-talkie style service delivered via an overlay network utilising Bluetooth and WiFi could potentially be much more effective, but clearly there are a number of issues that require investigation, including how to carry out 'least-cost' multi-hop routing and device discovery.

After meeting with my supervisor, we came to the conclusion that it would be more interesting to investigate ubiquitous peer-to-peer applications in general across wireless ad hoc networks. Voice could then potentially be one of the services delivered across such a network. The following was subsequently agreed as the original kick-off proposal:

In the space of just a few years, peer-to-peer (P2P) networking has become a computing phenomenon. Millions of Internet users are communicating with each other through P2P file sharing software programs that allow a group of computer users to share text, audio and video files stored on each other's computers. However, as it becoming increasingly evident, P2P networks have capabilities and uses that stretch far beyond 'file trading'.

With the proliferation of mobile devices such as cellular telephones and PDAs, and the increasingly pervasive nature of wireless technology, it is clear that there is no longer a need for us to restrict peer-to-peer applications to fixed computers on wired networks, or indeed to constrict cellular users to traditional client/server applications such as WAP (Wireless Application Protocol). P2P and wireless technology are an ideal match, and deploying the two together will enable us to exploit a wide range of new opportunities that were previously not feasible. The inherent nature of mobile devices, in that they are suitably lightweight and portable to be carried around by people, makes them ideal instruments to form the peers of a global wireless data network, where information can be shared between individuals in a ubiquitous manner.

Unfortunately current 2G and 3G mobile networks are not suitable in isolation for delivering mobile P2P services, due to the relatively high cost of data transmission, latency and limited footprint. A mobile ad hoc network is a system comprised of mobile devices that act as both hosts and routers, communicating wirelessly in an arbitrary way without an existing network infrastructure. The devices on an ad hoc network are free to move about and the topology of this kind of network is therefore dynamic. A key feature is multi-hop support, which for example could allow a device that is outside traditional mobile coverage to still be able to access services by relaying requests to another device that is in range over a technology such as Bluetooth. Alternatively, it may be more economical to send data using a nearby WiFi hotspot rather than through a GSM mobile network – the potential practical uses of ad hoc technology are only limited by imagination. By introducing a peer-to-peer architecture over such a scheme could enable users to access fully decentralised applications and to discover new resources as and when they become available.

The aim of this project is to first of all investigate recent developments in the wireless mobile, peer-to-peer and ad hoc network arenas, in order to fully understand the challenges that must be overcome for such a scheme to be commercially viable in the near future.

The second phase of the project will build upon the research phase above and will involve the design and implementation of a real peer-to-peer ad hoc wireless system to demonstrate the benefits of this next generation technology. The goal is to be able to demonstrate a set of applications such as multicast push-to-talk voice and location-based services on actual mobile devices running J2ME (Java 2 Mobile Edition). The devices will be capable of discovering each other, and adapting the types of service presented to the end user based on the methods of communication available, e.g. Bluetooth, GSM, WiFi. A universal framework will need to be defined in order to allow new applications to be added in the future. Issues of security and confidentiality in such a distributed environment are clearly important and will be taken into consideration during the design.

2.2 Summary of research phase

I spent several weeks carrying out a survey of existing work in the areas of mobile computing, peer-to-peer networking, cellular telephony and ad hoc networking arenas in order to identify technical challenges. The following is a summary of the types of areas that I looked into:

- **Peer-to-peer networking** – services such as Napster and Gnutella which have been popular for file-sharing on the Internet, the JXTA project, Distributed.Net and SETI@home which exploit the abundance of clock cycles.
    - I had a specific look at the Free World Dialup VoIP network, particularly at the lightweight SIP protocol which is very similar to HTTP. As part of this I read some work [TURN03] by Kenneth Turner on VoiceXML and CRESS.
- **Wireless WANs** – I looked at various cellular technologies [WEB01] such as FDMA, TDMA and CDMA, plus some of the applications such as WAP, i-mode, CHTML and newer 3G services. I was particularly impressed by the Nextel PTT (Push to Talk) service in the US which offers a coast-to-coast walkie-talkie service for a flat monthly fee.
    - I concluded that cellular airtime is still relatively expensive and not suitable for frequent group communications. The speed is also quite restrictive, e.g. for file sharing.
    - I also looked at FastChat [WEB02], a push-to-talk style service launched recently in the UK. This works over GPRS, but only supports Symbian handsets at the moment.
- **Wireless LANs** – I looked at WiFi, Bluetooth [GROT01], proprietary low-range radio services such as Cybiko [WEB03], infra-red, and Zimmerman intra-body networks [KORT02].
- **Ad hoc networking** – The IETF work on MANET is quite interesting [WEB04], and I found that there is quite a bit of work going on relating to efficient distributed routing algorithms. Security is also an important issue. The AODV and DSR routing algorithms have been proposed for multi-hop peer-to-peer networks.
- **Hardware and software for mobile devices** – I considered technologies such as Symbian, J2ME (Java 2 Micro Edition) and Microsoft .NET Compact Framework. I got the impression that the latter is quite flexible and provides a rich API for development. It also means that development can be done in C# or VB.NET rather

than C++, as is the case with Symbian. J2ME is far too restrictive – the API is extremely limited.

I also looked at other research projects that had similar overall interests to mine, i.e. bringing p2p applications onto a wireless network. There were some useful outputs from this, including:

- **University of Oregon, Proem Project** [KORT02] – they have developed a framework for wireless p2p applications and the paper advocates impromptu collaboration.
- **The Virginia Polytechnic Institute and State University, Wireless Ad hoc messenger** [WEB05] – sponsored by Microsoft Research, based on .NET CF, uses multi-hop routing for a text-based p2p chat service across iPAQs.
- **Motorola Labs and Purdue University, MOBY** [HOR02] – a Jini-based platform with an interesting security model.
- **University of Florida, Konark** [DES03] and [HEL03] – a service discovery and delivery protocol for ad hoc networks.

My conclusion from the research that I carried out was that whilst there were a number of interesting areas that would benefit from additional research, such as security isssues and delivering multimedia (e.g. video) services over such a network with QoS implications, it would be quite difficult to demonstrate such a framework using the hardware available for the project.

As a result of further discussions with Dr. Dulay a couple of weeks ago, it was therefore decided that it might be better to change the focus of the project slightly, and to consider how consumer devices in a personal area network could carry out self-management through the specification of policies.

2.3 Autonomic policy-based device interaction in Personal Area Networks

Work on this more specific idea has only been ongoing for the last couple of weeks, and is motivated by the AMUSE project [WEB06], which is looking at the autonomic management of ubiquitous systems for e-Health. We thought it would be interesting to take this idea and apply it to Personal Area Networks (PANs), made up of consumer devices such as phones, PDAs, MP3 players etc. Consumers nowadays are carrying an increasing number of these devices, resulting in an increasing amount of time spent configuring and operating them. For example, a user may have to search for the volume control on his MP3 player when a call comes in on his mobile phone, thereby wasting valuable time and causing unnecessary inconvenience. To date, very few attempts have been made to automate the management of these consumer devices.

The AMUSE project advocates the concept of a self-managed cell (SMC) as the basic architectural pattern for implementing self-management at both local and integrated levels. The cell would therefore contain various devices as well as administrative functionality, e.g. discovery and policy management services. This brings about several areas of interest:

- Specification of policies in a relatively high-level form, and the subsequent compilation of these into software objects that can be executed. Development of a common framework for hardware vendors to adhere to when developing new consumer devices, so that they can join a self-managed cell ubiquitously.
- How to group consumer devices in a hierarchical fashion, enabling policy inheritance.
- Devices in this environment are likely to appear and disappear, e.g. when switched off. The cell management functionality needs to consider how to react when these types of events occur.
- Discovery of devices within the cell. Managing devices that have different capabilities, perhaps via adapters.
- How these cells should interact when in proximity of each other, e.g. can a device belong to and be managed by one than one cell, and the conflict resolution that is necessary for the policy rules.
- The intention is to keep user input to a minimum, i.e. an autonomic system. However we could use user feedback to refine the policies.
- Efficient interaction between devices. It is clearly important to look at ways of minimising the impact of the management functionality of the battery performance of these devices. Such a system should therefore aim to avoid unnecessary device communication.

There are a number of other useful sources of information in this arena. Dr Mitchell Waldrop [MITC03] in his recent article on *Autonomic Computing: The Technology of Self-Management*, refers to a “continuous control loop”, i.e. each component of the system (hardware and software) should now only know how its assigned tasks but should also have internal mechanisms that constantly monitor its own operation, and make corrections as needed. A key feature of such a scheme is that each device would handle as much as possible locally – and yet still have the means to call on the larger system when it needs help. Dr Waldrop also suggests that the scheme could be recursive, so that when the call for help reaches that system, it may decide to call for help to a still larger system for help.

Project Oxygen [WEB07] at MIT which has an objective of "*bringing abundant computation and communication, as pervasive and free as air, naturally into people's lives*" has already demonstrated the benefits of self-management in a range of applications. They distinguish between *basic physical* and *basic virtual* objects. The former senses or actuates a physical entity, whereas the latter collects, generates and transforms information, e.g. extracting information from an incoming electronic form and sending the results on to a particular device. The project also advocates the use of a scripting language to enable the tasks that need to be automated to be specified easily and rapidly.

My intention is to use relevant parts of the Ponder Policy Specification Language [DAMI01] which has been developed by the DSE group. Most pertinent are obligation policies, which have the following notation:

```
inst oblig policyName "{"
        on                  event-specification ;
        subject [<type>]    domain-Scope-Expression ;
        [ target [<type>]   domain-Scope-Expression ; ]
        do                  obligation-action-list ;
        [ catch             exception-specification ; ]
        [ when              constraint-Expression ; ]           "}"
```

An example of a policy we could use for the self-management of consumer devices is:

```
inst oblig incomingCellularCall {
        on          eventIncomingCellularCall(callerID) ;
        subject     s = /dev/cellular/phones ;
        target      t = /dev/music ;
        do          t.mute() -> t.playCallerID(callerID) ;
        when        s.profile != "do not disturb" ;
}
```

The above policy is specified by cellular phone devices, and applies to all devices that live in the /dev/cellular/phones namespace. When a call comes in, all devices that are in the /dev/music namespace will be asked to mute, and to then play out the incoming caller's ID to the user. However these actions will not be carried out if the phone's profile is in the "do not disturb" mode, i.e. the user doesn't want the music to stop when a call comes in!

As specified in the Ponder language, the basic policy constraints can be derived from:

- **Subject/target state** – reflected by attributes at an object's interface.
- **Action/event parameters**
- **Time constraints**, e.g. between 0200 and 0400.

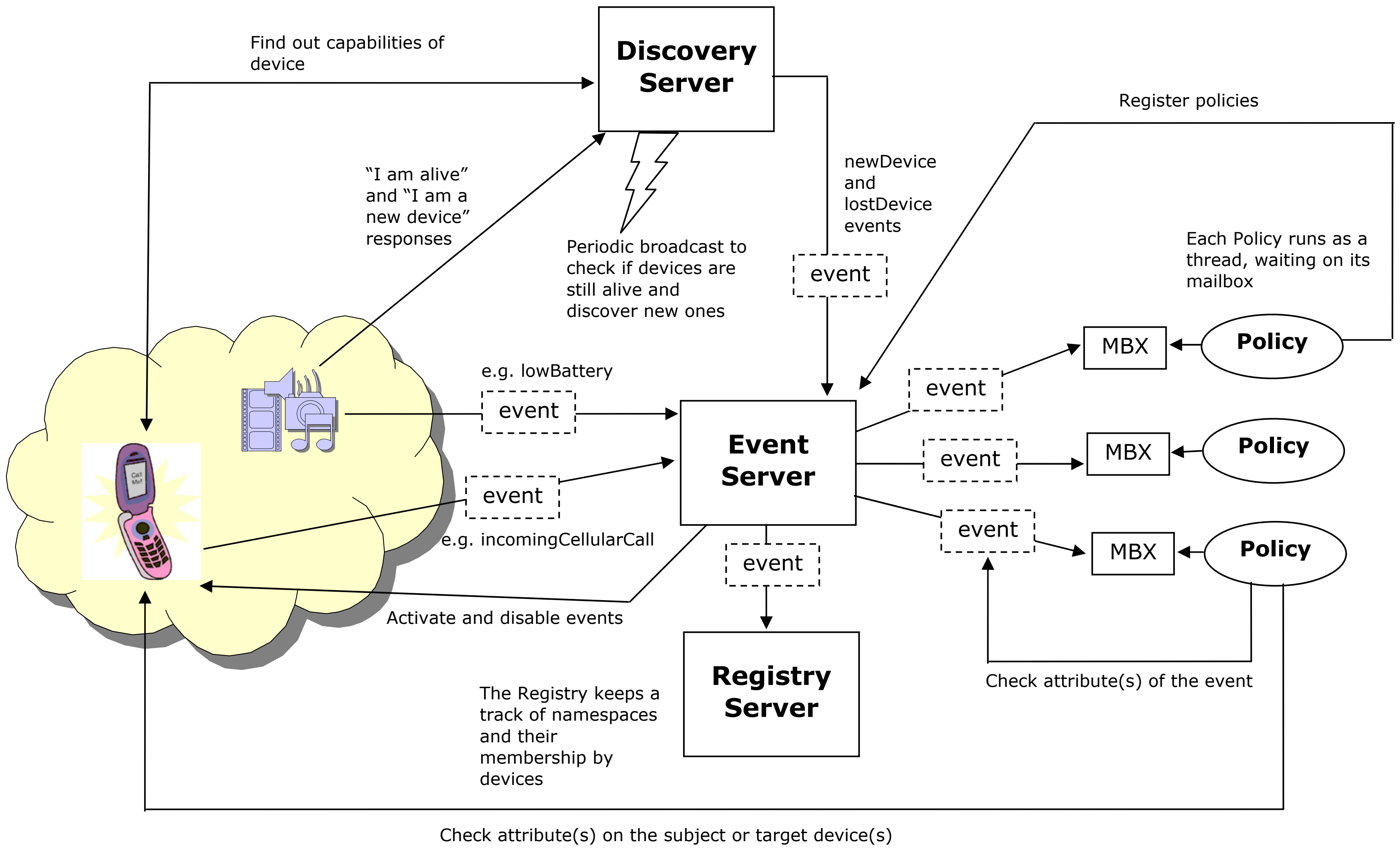
It also seems appropriate to use similar policies to define the behaviour of the management components within the cell – this is almost like meta-policy. For example, we might want to use a policy to specify how frequently the discovery server polls for devices, perhaps depending on the amount of battery power left. Ponder supports meta-policies, which can be used to resolve conflicts between overlapping subjects and targets.

3 Moving into the implementation…

3.1 Proposed architecture

This section presents my ideas with regards to a basic framework implementation. The intention here is to develop a relatively simple but demonstrable system, which has the ability to read in policies, handle events generated by devices within the cell and allow for actions to be performed on those devices autonomically. We are also interested in dealing with the discovery and loss of devices and generating appropriate events.

The diagram on the following page illustrates some of the key interactions in the system.


Find out capabilities of device
Discovery Server
"I am alive" and "I am a new device" responses
Periodic broadcast to check if devices are still alive and discover new ones
newDevice and lostDevice events
event
Register policies
Each Policy runs as a thread, waiting on its mailbox
MBX
Policy
e.g. lowBattery
event
Event Server
event
MBX
Policy
event
e.g. incomingCellularCall
event
event
MBX
Policy
Activate and disable events
Registry Server
The Registry keeps a track of namespaces and their membership by devices
Check attribute(s) of the event
Check attribute(s) on the subject or target device(s)

Policies

The intention is for policies to simply be software objects. Whilst they will be originally specified using Ponder syntax, the compilation into objects will be done before execution. Initially, the compilation will be done by hand, however as a future extension it might be worth looking into ways of refining the existing Ponder compiler to provide the ability to generate code for this system.

Each policy will run as a thread and will block on its individual mailbox. When messages (events) become available, the thread will be woken up to carry out any necessary actions.

Events & the Event Server

Events will be represented as objects as well. These will be created by the device and sent across (serialised) to the event server. Using inheritance, we can create specialised events, e.g. temperatureEvent which will inherit from the Event base class. A temperatureEvent may contain a member that stores the temperature reading that caused the event to occur.

The Event Server will send Activate and Disable messages to the devices dependent on which events it wishes to listen for. The Event Server bases this information on the collection of policies that it knows about. This means that devices will only send out events if there are policies that are interested in them.

It may be worth using a thread pool for the Event Server. These worker threads would be used to despatch incoming events out to the relevant policies concurrently.

Registry & devices

A directory-like hierarchical structure for device grouping. e.g.

/dev/music/mp3_player

/dev/cellular/nokia_8910_phone

/dev/pda/hp_ipaq_h5550

A domain can have only one parent at most, but many children, but devices can belong to one or more domains.

The Registry Server will receive events about new devices, or ones that are no longer part of the cell. It needs to be able to work out where in the namespace a device should be added and in some cases if a device should be placed in more than one namespace.

The Device class itself will be abstract (an interface) – different types of devices will then extend this, e.g. PhoneDevice which will also be an interface. The actual adapters that talk to the physical hardware, e.g. WindowsSmartPhone can then use multiple inheritance via interfaces to implement functionality that that device can provide. So a WindowsSmartPhone may implement SimplePDADevice as well as PhoneDevice.

Discovery

The Discovery Server will send out broadcast packets every now and then (configurable via a policy!) to check the status of existing devices and to see if there are any new ones. This mechanism will probably use IP multicast with UDP packets.

To discover device capabilities, the Discovery Server is likely to send out a direct UDP packet to the relevant device, and the device will respond accordingly.

Start-up behaviour

- Start device registry
    - Load last known topology if found
- Start discovery server
- Start event server
- Start policy server
    - deploy ‘internal’ policies – e.g. to discovery and event servers – this defines the behaviour of these components.
    - create a thread for each standard policy and start execution.

Periodic flush of device topology to disk, so that we can reload it if the system needs to be restarted.

3.2 Hardware and software platforms

The proposed development environment is .NET Compact Framework using Visual Studio .NET 2003. These components are already installed on my PC and were used by me over the Christmas break to develop a sample UDP/TCP chat application across an iPAQ Pocket PC and laptop with WiFi card in order to learn the .NET API and the C# language. The C# language has some very useful features such as delegates which Java does not support.

In terms of hardware, I currently have an iPAQ H5550 with Bluetooth and WiFi and a Windows Smartphone Developer Kit (including red-e

Smartphone) on loan from the DSE group. I'm trying to find out if I can get hold of a Smartphone that supports the Bluetooth PAN profile.

4 Project planning

4.1 Proposed milestones

Target date	Activities
End of Week 3, Spring Term	Hand in Interim report and meet with supervisor and second marker to discuss ideas. Formalise a design for the basic architecture.
End of Week 5, Spring Term	Translate design into C# code, setting up skeleton structure for classes etc.
End of Week 11, Spring Term	Bulk of development work complete, and a view of what extra functionality can be added once the exam period is over.
After Exams (likely to be End of Week 3, Summer Term)	Resume work on project. Meet with supervisor to review progress.
End of Week 6, Summer Term	Development work should be almost complete by this stage.
End of Week 7, Summer Term	The report and presentation should be almost at completion stage.

4.2 Outstanding issues register

Issue	Status	Opened	Closed	Notes
Clarification of Ponder syntax – how to specify "for all" devices in a group, or to refer to specific devices?	O	21/01/04		
We have iPAQ devices but for the demo can we obtain a Bluetooth Smartphone?	O	21/01/04		If not, maybe we can simulate a phone device.
Communication between remote objects using RMI/Remoting or simple TCP/UDP sockets?	O	23/01/04		.NET CF doesn't support remoting, and I don't think J2ME supports RMI but this needs investigation. The Socket

				implementation in .NET CF however is quite easy to work with.

Status values are **O** (Open, under investigation), **C** (Closed), **H** (On Hold).

4.3 Risk register

Risk	Severity	Raised	Mitigation notes
.NET CF may not be able to handle a large number of threads efficiently, i.e. to support one thread per policy object	MEDIUM	25/01/04	The design may have to be changed if this model proves to be too slow.
Lack of availability of hardware for the demonstration	MEDIUM	25/01/04	I will require at least 2 iPAQ h5550 or similar devices, and ideally a Bluetooth Smartphone. Without at least 3 devices it will be hard to demonstrate the results of this work.
Learning curve for Microsoft .NET API and C# may prove too difficult given the project timescale	LOW	25/01/04	Was HIGH risk, now LOW - I spent the Christmas holidays learning about .NET and C# with some sample applications, so I now have a better idea of what is achievable.
Hard to interface into real consumer devices	MEDIUM	25/01/04	We will have to simulate devices such as MP3 players, since getting data from and interfacing into a real one is likely to prove infeasible.

Severity values are **CRITICAL**, **HIGH**, **MEDIUM** and **LOW**.

5 Bibliography

[TURN03] Turner, K. J., "Representing New Voice Services and Their Features", Proc. Feature Interactions in Telecommunication Networks VII, pp 123-140, IOS Press, Amsterdam, June 2003

[GROT01] Groten D., Schmidt, J. R., "Bluetooth-based Mobile Ad Hoc Networks: Opportunities and Challenges for a Telecommunications Operator", IEEE VTS 53rd Vehicular Technology Conference, VTC 2001 Spring, pp. 1134-1138, May 2001.

[KORT02] Kortuem G., Schenider J., Preuitt D., Thompson T. G. C., Fickas S., Segall Z., "When Peer-to-Peer comes Face-to-Face: Collaborative Peer-to-Peer Computing in Mobile Ad hoc Networks", First International Conference on Peer-to-Peer Computing (P2P'01), August 27 - 29, 2001, Lingköping, Sweden.

[HOR02] Horozov T., Grama A., Vasudevan V., Landis S., "Moby – A Mobile Peer-to-Peer Service and Data Network", Proceedings of the International Conference on Parallel Processing (ICPP'02), 2002.

[MITC03] Waldrop, M. M., "Autonomic Computing: The Technology of Self-Management", The Future of Computing Project, Woodrow Wilson International Centre of Scholars, July 2003.

[DAMI01] Damianou, N., Dulay, N., Lupu E., Sloman M., "The Ponder Policy Specification Language", Proc. Policy 2001: Workshop on Policies for Distributed Systems and Networks, Bristol, UK, 29-31 Jan. 2001, Sptringer-Verlag LNCS 1995, pp 18-39.

[SVE03] Sventek, J., "AMUSE – Autonomic Management of Ubiquitous Systems for e-Health", PowerPoint Presentation Slides, University of Glasgow.

[DES03] Desai B., Verma V., Helal S., "Infrastructure for Peer-to-Peer Applications in Ad-Hoc Networks", Submitted to 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, February 2003.

[HEL03] Helal S., Desai N., Verma, V., Lee, C., "Konark – A Service Discovery and Delivery Protocol for Ad-Hoc Networks", Proceedings of the Third IEEE Conference on Wireless Communication Networks (WCNC), New Orleans, March 2003.

[PER00] Perkins, C., E., "IP Address Autoconfiguration for Ad Hoc networks", draft-ietf-manet-autoconf-00.txt, IETF Mobile Ad Hoc Networking Working Group, 10 July 2000.

[JOH94] Johnson, D. B., "Routing in Ad Hoc Networks of Mobile Hosts", Proceedings of the IEEE Workshop on Mobile Computing Systemsn and Applications, December 1994.

[SUN01] Sun, J., "Mobile Ad Hoc Networking: An Essential Technology for Pervasive Computing", Proc. International Conferences on Info-tech & Info-net, Beijing, China, C:316-321.

[WEB01] Wireless Wide Area Networking Technologies, http://www.rhowireless.com/wan/Default.htm

[WEB02] FastChat, http://www.fastchat.co.uk

[WEB03] Cybiko, http://www.cybiko.com

[WEB04] IETF Mobile Ad Hoc Networking (MANET) Working Group, http://protean.itd.nrl.navy.mil/manet/manet_home.html.

[WEB05] The Virginia Polytechnic Institute and State University, Multihop Ad Hoc Instant Messaging using Microsoft .NET Compact Framework and Pocket PCs, http://people.cs.vt.edu/~irchen/microsoft-grant/Website_HTML_Files/index.html.

[WEB06] AMUSE: Autonomic Management of Ubiquitous Systems for e-Health, http://www.doc.ic.ac.uk/~ecl1/projects/AMUSE/.

[WEB07] MIT Project Oxygen, http://oxygen.lcs.mit.edu/.

[WEB08] The JXTA Project, Sun Microsystems, http://www.jxta.org.

[WEB09] JXTA for J2ME, Sun Microsystems, http://jxme.jxta.org.