

# **Mobile P2P Networking for Play, Gaming and Learning**

Ph.D. progress report by

**Bent Guldbjerg Christensen**  
**bentor@daimi.au.dk**  
**Department of Computer Science**  
**Aarhus University**

February 2006

## Preface

My interest in mobile systems and hypermedia precedes my Ph.D. project by several years. I developed, together with my study group, my first hypermedia system in the autumn of 2000 as part of the Hypermedia course at Aarhus University. The system was a compiler engine written in Python which could compile guided tour maps created with the Webvise open hypermedia system (Sandvad et al., 2001), into interactive 3D VRML models. The system, Webvise3D, was later demonstrated at a demo session during the 12th Hypertext conference in 2001 (Poulsen et al., 2001). I received my master's degree in computer science in the spring of 2003. Together with Allan Hansen, I investigated W3C's proposed XLink standard as an open hypermedia linking format and compared it to a number of other open hypermedia link models. Key to our work was also a prototype, the Xspect system, demonstrating that XLink indeed shared many of the qualities found in open hypermedia systems and the prototype also showed how XLink linkbases could be integrated with common Web browsers using standards such as XSLT and JavaScript. I presented the results of our investigation at the International Workshop on Open Hypermedia Systems Core Concepts and Research Directions during the 13th Hypertext conference in 2002 (Christensen and Hansen, 2002) and at the 12th International World-WideWeb Conference in 2003 (Christensen et al., 2003). Thus, the results were presented to both the open hypermedia research community and the Web community. In February 2003 I was hired as a research assistant at the Centre for Interactive Spaces to work as programmer on the research project ContextIT, a project with two private partners, Euman A/S and TDC/InnovationLab, and the computer science departments at the universities in Aalborg and Aarhus. In the project I worked on the initial version of a hypermedia framework named HyCon. The HyCon framework was designed to encompass annotations, links, and guided tours associating locations and RFID- or Bluetooth-tagged objects with maps, Web pages, and collections of resources. The HyCon architecture extends upon earlier location based hypermedia systems by supporting authoring in the field and by providing access to browsing and searching information through a novel geo-based search (GBS) interface for the Web. The HyCon framework and the first prototype of HyConExplorer was documented in the article: "*HyCon: A framework for context-aware mobile hypermedia*" in the New Review of Hypermedia and Multimedia journal in late 2003 (Bouvin et al., 2003). The ContextIT project ended in December 2003 and the paper: "*Integrating the Web and the World: Contextual trails on the move*", summarised the joint results of the developed systems at the 15th Hypertext conference in 2004 (Hansen et al., 2004a). Since January 2004 the development and tests of the HyCon framework and HyConExplorer system continued in the iSchool project which took place in the Centre for Interactive Spaces. This resulted in the four papers: "*eBag - the Digital School Bag*" (Brodersen et al., 2004), "*RSS as a Distribution Medium for Geo-spatial Hypermedia*" (Hansen et al., 2005), "*eBag - a Ubiquitous Web Infrastructure for Nomadic Learning*" (Brodersen et al., 2005), and "*Supporting Mobile and Nomadic Learning*" (Bouvin et al., 2005) of which I presented the last two at the 14th International World Wide Web Conference in Chiba, Japan in May 2005. The papers discusses the evaluation of the HyConExplorer in a school setting where the system was used to support pupils during project work in the field. This work was combined with my early Ph.D. studies which I started in September 2004 as part of the Nomadic Play project with the Lego Company as private partner. Especially the

experiences with evaluation of concepts and prototypes with pupils in their daily environment, has been a useful headstart in my further research on mobile ad hoc networking for play, gaming and learning. Together with Martin Brynskov and Martin Ludvigsen, I conducted design workshops with children in different age groups to obtain an understanding of children's use of technology today and to try out a participatory design process which resulted in the paper "*Designing for Nomadic Play: A case study of participatory design with children*" presented at the 4th International Conference for Interaction Design and Children in Boulder, Colorado in June 2005 (Brynskov et al., 2005). In August 2005, I participated in a workshop entitled "Ambient Computing in a Critical, Quality of Life Perspective" with the position paper "*Do Social Computing Make You Happy? A Case Study of Nomadic Children in Mixed Environments*" (Christensen, 2005) where I discussed visions and goals for using mobile and ambient computing technologies. Based on my experiences with mobile technologies for children in learning environments, I began developing a peer-to-peer framework called LightPeers, to support that. My work on the LightPeers framework so far is to be presented at the 3rd MiNEMA workshop on Middleware for Mobile Environments, February 7-8, 2006 in Leuven, Belgium with the research paper "*LightPeers - A Framework Supporting Nomadic Learning in Mixed Environments with Mobile Ad Hoc Networking*" (Christensen, 2006).

With this brief review of my previous work, I hope to have given an overview of my background and contextualized my current research in "Mobile P2P Networking for Play, Gaming and Learning" with the LightPeers framework. A full list of my publications can be found in appendix A. In this report I will describe the current status of the LightPeers project and the future plans for my Ph.D. studies.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Problem Domain . . . . .	1
1.2	Research Issues for Investigation . . . . .	2
1.3	Result of the Investigation . . . . .	2
1.4	Method of Research Used . . . . .	3
1.5	Structure of Report . . . . .	4
<b>2</b>	<b>Ubiquitous P2P Computing</b>	<b>5</b>
2.1	P2P . . . . .	5
2.2	The JXTA Framework . . . . .	7
2.2.1	The JXME Project . . . . .	10
2.3	Proem . . . . .	11
2.4	Discussion and Summary . . . . .	13
<b>3</b>	<b>The LightPeers Framework</b>	<b>15</b>
3.1	Architecture . . . . .	15
3.2	Datamodel . . . . .	18
3.3	Contingency handling . . . . .	21
3.4	Implementations . . . . .	23
3.5	Discussion and Summary . . . . .	23
<b>4</b>	<b>Prospective Application Domains</b>	<b>26</b>
4.1	Interactive Sports Equipment . . . . .	26
4.2	Nomadic Learning . . . . .	27
4.3	Mock Games . . . . .	28
<b>5</b>	<b>Future Work and Conclusion</b>	<b>29</b>
5.1	Future Work . . . . .	29
5.2	Conclusion . . . . .	29
<b>A</b>	<b>Publications</b>	<b>34</b>

## List of Figures

1	The architectures for client-server and P2P networks. . . . .	6
2	The three layer model of JXTA: Core, Services, and Applications layers .	8
3	The Proem Runtime architecture consisting of: protocol stack, peerlet engine, and a collection of services . . . . .	12
4	The five LightPeers Framework Layers: Network, P2P transport, Service, Application, and Sensor layers. . . . .	16
5	An alternative perspective on the P2P transport Layer and Service Layer where the components are grouped into Reflection components and Data handling components. . . . .	18
6	The state of a peer from the LightPeers datamodel serialized as an XML structure. . . . .	19
7	The state of a session from the LightPeers datamodel serialized as an XML structure. . . . .	20
8	A LightPeers message produced from the pPost application. . . . .	21
9	The Debug Architecture . . . . .	22
10	Gym horse setup with P2P sensoring enabled by LightPeers . . . . .	27

## List of Tables

1	Comparison of P2P protocols and frameworks . . . . .	14
2	Comparison of P2P protocols and frameworks including LightPeers . .	24

# 1 Introduction

In this report, the research results of the first part of my Ph.D. studies is described, and directions on areas for future work are presented. The report is entitled "Mobile P2P Networking for Play, Gaming and Learning" which reflects the research topic of my work.

## 1.1 Problem Domain

The visions for Ubiquitous and Pervasive Computing as described by Weiser (1991, 1993), present the idea of ubiquitous computing power available through numerous mobile and stationary computer devices distributed throughout our environment and accessible through a ubiquitous network. Part of this vision is today realized in the form of personal mobile devices. Mobile and nomadic users typically carry a number of such devices with them: laptops for presentations, PDAs for note taking, digital cameras and SmartPhones with built-in cameras for documenting interesting situations, or digital audio recorders and players for voice recording and audio playback. The reason why it is only partly fulfilled is that Mark Weiser went even further in his visionary description of ubiquitous computing or embodied virtuality:

"[...] some of us use the term "embodied virtuality" to refer to the process of drawing computers out of their electronic shells. [...] Most of the computers that participate in embodied virtuality will be invisible in fact as well as in metaphor. Already computers in light switches, thermostats, stereos and ovens help to activate the world. These machines and more will be interconnected in a ubiquitous network." (Weiser, 1991)

The computing devices will according to Weiser, be drawn out of their electronic shells, and they will even become invisible because they do not look like computers and because they become a natural part a everyday life that do not draw the user's direct attention. This part of Weiser's vision have been an inspiration for the work I have been involved with at the Center for Interactive Spaces<sup>1</sup>, where new ways of interfacing with computing devices have been explored and still are. Weiser mentions further that the computing devices will be interconnected through a ubiquitous network, but how this could be accomplished is not elaborated further. The main topic of my studies so far has been on this issue, of how to realize such ubiquitous networking.

Technologies for such ubiquitous networking include especially wireless network technologies e.g., WLAN, Bluetooth<sup>2</sup>, ZigBee<sup>3</sup>, GPRS, and more, because they allow the devices to be mobile and to interconnect in the field, or to be nomadic in the sense that a mobile device can connect to a stationary device in a fixed infrastructure.

Network topologies building on top of these technologies and supporting device mobility, ad hoc interconnections, or peer-to-peer (P2P) communication each have

---

<sup>1</sup><http://www.interactivespaces.net>

<sup>2</sup><http://www.bluetooth.org>

<sup>3</sup><http://www.zigbee.org>

their own advantages and shortcomings when used in certain application domains. The promising JXTA project (Bondolo et al., 2004) defines a set of open protocols that should allow a connected device on the network ranging from cell phones and wireless PDAs to PCs and servers to communicate and collaborate in a P2P manner. The vision is indeed to provide a ubiquitous network for all kinds of devices, however in practice the implementations of the JXTA protocols are dominated by Java versions and have not yet been successfully implemented for limited devices, such as mobile phones, despite the effort in the JXME project (Hamada et al., 2005). I will draw on the experiences learned from existing implementations of networking techniques to discover the requirements for a ubiquitous network for mobile limited devices.

The technological solutions for ubiquitous networking is interesting, but becomes even more relevant when coupled with a use case or a specific application domain, where the technology provides enhanced or maybe entire new ways of working, learning, or playing. For P2P systems, a lot of attention has been devoted to file sharing applications for stationary PCs (Clarke, 1999; Cohen, 2002; Fanning, 1999), but the area of mobile P2P systems seems promising for a whole new domain of applications, where the physical location and context come into play. What are the new possibilities and limitations for this domain?

## 1.2 Research Issues for Investigation

Based on the problems and prospects found with the visions, current realizations, and existing application domains of ubiquitous computing, I will formally investigate these issues:

1. Which requirements should be prioritized and met by a lightweight architecture and framework, including datamodel and protocols of framework components, for supporting mobile ad hoc P2P network for limited devices?
2. Which domain specific demands exist in the prospective application domains, and can a generic substrate be identified and supported?

## 1.3 Result of the Investigation

The result of investigating the two issues described above includes a prototype implementation of the LightPeers framework for supporting mobile ad hoc P2P networking. Upon the framework a number of applications should demonstrate different aspects supported by the framework for specific domains. The LightPeers framework are described in section 3 on page 15.

The results demonstrated by LightPeers are based on and generalised from a survey over existing network topologies and frameworks. The survey is presented in section 2 on page 5.

Some of these results have already been described in the papers: "*Designing for Nomadic Play: A case study of participatory design with children*" (Brynskov et al., 2005) presented at the 4th International Conference for Interaction Design and Children.

And the position paper "*LightPeers - A Framework Supporting Nomadic Learning in Mixed Environments with Mobile Ad Hoc Networking*" (Christensen, 2006) to be presented at the 3rd MiNEMA Workshop on Middleware for Mobile Environments.

#### 1.4 Method of Research Used

My work with LightPeers has been of a highly practical nature, and the approach to the research issues favours experiments and practical results with the technologies and their use over merely theoretical discussions. I have therefore used methods that support this approach. The methods are taken from the field of experimental system development with a special focus on prototyping. Below I will discuss the concept of prototyping and my use of it.

Prototypes are not exclusively built for testing purposes, but are also used as learning and communication tools for both developers and users involved in the development of the system. This means that it is not the prototype itself that is the primary object of interest but rather the process of prototyping and learning. However, depending on how the prototype is developed, the prototype as a whole (or parts of it) can evolve into the final system.

Prototyping in software development can be applied in different ways and with different perspectives in mind. Floyd (1984) defines a taxonomy for prototyping consisting of the three forms: exploratory, experimental, and evolutionary prototyping.

Exploratory Prototyping is often used in the early stages of software development where the focus is on clarifying the requirements of the future system. The prototypes can be used to investigate new and unknown areas or they can simply be used as "idea generators". In the latter case the prototypes are often mock-ups or "throw-away" prototypes which are not designed to be part of the final system but as a means of communication between users and developers. Experimental Prototyping is used to create prototypes which can be evaluated in experimental use. The areas of concern can be e.g., usability (is the proposed interface good enough?) or it can be functionality (does a specific sub system or algorithm perform as expected?). In the case of usability the experiments can take place in cooperation with the users in use-like situations. This may also be the case with functionality testing, but Floyd writes:

"We may also be interested in other aspects such as demonstrating the technical feasibility of a software idea or determining the efficiency of a part of the system. In such cases, it may be appropriate to use prototyping within the development team, for example to aid communication to design and specification or to help structure the implementation of a large system." (Floyd, 1984)[p.3]

Thus, experiments with prototypes can also be used to demonstrate ideas and working solutions and to communicate experiences within the developer organization.

The goal of my work is to investigate mobile P2P networking for specific application domains and thereby find answers to the questions outlined earlier on. The prototype implementation should demonstrate both advantages and disadvantages of the approach: what was possible and what should be done differently. Hence, my

method is strongly influenced by exploratory prototyping. The prototypes are part of a learning process and a learning vehicle used to explore new and unknown areas and to test and demonstrate my ideas.

For nearly all the projects conducted at the Center for Interactive Spaces, a participatory design (PD) (Greenbaum and Kyng, 1991) approach have been used with exploratory prototyping, and this is also the case the Nomadic Play project, which I am taking part in. Thus, in the project we have involved the potential end-users in both analysing, designing, and testing the systems in an iterative development process. We chose to do that, so the developers and designers could get a better understanding of the practises which we design for and to demonstrate the developed systems in real use-situations with real users.

In the evaluations to come I plan to use the prototypes as a means to present my ideas and to communicate the lessons learned.

## 1.5 Structure of Report

The rest of this report is structured as follows. Section 2 on the following page includes a survey of network topologies with special regards to systems and frameworks for ubiquitous P2P computing.

In section 3 on page 15 the LightPeers framework, including components and data-model, is described and discussed.

Section 4 on page 26 presents prospective application domains for the LightPeers framework.

Section 5 on page 29 concludes the current status of my work and outlines my plan for further research.

In appendix A on page 34 a full list of publications, which I have authored or co-authored, is provided.

## 2 Ubiquitous P2P Computing

Ubiquitous computing as described by Weiser (1991, 1993), is also known under names such as embodied virtuality, augmented reality, pervasive computing, ambient computing, embedded computing, and more, and each of these names corresponds to a certain current in the overall wave of integrating computation into the environment, rather than having computers as distinct objects. Likewise, there exists a number of technologies with different approaches to fulfill the vision of ubiquitous computing including:

*mobile ad hoc networks (MANETs)*: are self configuring networks of mobile devices connected wirelessly, which can form ad hoc network topologies without any fixed infrastructure present. Thus, making the network independent and robust. Especially the Internet Engineering Task Force (IETF) has with its MANET working group<sup>4</sup> been in a leading position in the MANET community with its work on standardizing IP routing protocol functionality making it suitable for wireless routing application within dynamic topologies.

*sensorwebs*: are often huge clusters of sensors monitoring and exploring the environment retrieving sensed information and sending it back to central servers for storage or further computation. Movers behind this trend includes NASA with its Sensorweb project<sup>5</sup> and the Smart Dust project at UC Berkeley<sup>6</sup> supported financially by the Defense Advanced Research Projects Agency (DARPA).

*peer-to-peer (P2P)*: is the notion of a computer network that depends on the computing power and bandwidth of each participant in the network instead of concentrating it on a low number of central servers. Each participant act as both a server and a client (depicted in subfigure 1(a) on the next page) as opposed to a central server and multiple clients (shown in subfigure 1(b) on the following page) which is known, e.g., from the World Wide Web with a central Web server and multiple Web browsers or Web clients.

The approach I have chosen for investigating my research issues for ubiquitous computing belongs to the category of P2P systems, and especially mobile P2P system and architectures. Therefore, a survey of P2P systems and frameworks is presented for background knowledge and later comparison with the LightPeers framework.

### 2.1 P2P

P2P systems and architectures have changed over time due to shortcomings in existing architectures or exploitation of new possibilities, and this evolution can be presented as different generations of P2P architectures as done by Harjula et al. (2004). Three generations of P2P architectures are described, and they provide a useful overview of past and current trends of P2P architectures. The first generation

<sup>4</sup><http://www.ietf.org/html.charters/manet-charter.html>

<sup>5</sup><http://sensorwebs.jpl.nasa.gov/>

<sup>6</sup><http://robotics.eecs.berkeley.edu/~pister/SmartDust/>

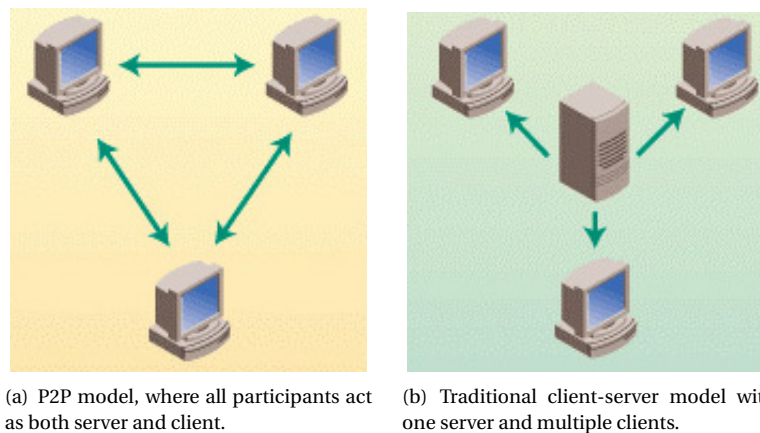


Figure 1: The architectures for client-server and P2P networks.

of P2P architectures consisted of peers and a dedicated server cluster for maintaining a central index or meta data of the shared resources and connected peers. The connections between peers were P2P. Napster (Fanning, 1999), made in 1999, is an example of such a first generation system. When a user of a peer in the Napster network is querying for a resource, the peer will connect to one of Napster's central server clusters and receive the resulting list of matching resources. If the user then choose to download the resource, a connection directly to the peer where the resource is located, is established. The server-centric architecture has a number a challenges exposed by handling the servers: need of maintenance, single point of failure, cost of running servers, poor scalability, and legal issues.

Second generation of P2P architectures are characterized by a pure P2P architecture, because each peer in such a network have equal capabilities and functionalities, and no dedicated servers are necessary. Peers of these pure P2P architectures are often called *servents*, from the words server and client. The Gnutella protocol (Frankel and Pepper, 2002; Ripeanu, 2001) is an example of a second generation P2P architecture. The company Nullsoft developed the protocol in 2000, but soon after it was reverse engineered and made open source. When a user of a servent in the Gnutella network is querying for a resource, the servent forwards the query to all its neighbor servents which again will forward to query to their neighbor servents. This recursive broadcast will continue until the resource is found or the Time-to-live (TTL) field of the query reaches zero. The TTL field is decreased by one for every servent it passes. Notice, that even if the resource is found, the query only stops from that particular servent and continues to propagate in other areas of the network until the TTL field makes it terminate. The actual transfer of the resources is accomplished through direct P2P connections between servents, like it was the case with Napster. The pure P2P architecture of Gnutella overcomes some of the challenges with first generation architecture such as Napster's. It is independent of central servers, and thereby not depending on maintenance, single point of failure, and cost of running servers. The recursive broadcast mechanism does however introduce a new challenge regarding scaling of the number of participating servents, because the network is flooded by queries as discussed by Kwok et al. (2003).

The third generation of P2P architectures tries to overcome the challenges of earlier generation architectures, by combining the strategies of the two previous generations. With this third generation peers are divided into two kinds, namely *peers* and *super-peers*. Peers are lightweight peers typically controlled by the user, and super-peers are functioning as relays for peers and other super-peers. The super-peers act much like the servants of second generation architectures, and regular peers use the super-peers as access points to the P2P network. Therefore, the regular peers are also sometimes referred to as *edge-peers*. Super-peers are typically also used for firewall and network address translation (NAT) traversal. As an example of a third generation P2P architecture, the Jabber<sup>7</sup> framework, which uses the XMPP (Saint-Andre, 2004) protocols, is suitable. Jabber networks consist of a dynamic network clients and servers, where the servers act as super-peers and clients as regular peers. The first application of Jabber is an asynchronous, extensible instant messaging platform, and an Instant Messaging (IM) network that offers functionality similar to systems such as AIM, MSN, and Yahoo, and is currently being used by, e.g., the Google Talk<sup>8</sup> service. The JXTA (Bondolo et al., 2004) framework is also a prime example of a third generation architecture. In short a JXTA network consists of peers (in JXTA called edge-peers) and super-peers (in JXTA known as *rendezvous* and *relay* peers) which are used as gateways for peers. The JXTA framework is described in detail in section 2.2.

The initiative for making a P2P framework to ease system development by identifying and abstracting common tasks does not belong to the LightPeers framework project alone. Other efforts have been made in the area of P2P frameworks, and they are now described for later comparison with LightPeers. The JXTA framework has been selected since, it is widely regarded as the most comprehensive open P2P framework.

## 2.2 The JXTA Framework

The JXTA project (Bondolo et al., 2004) defines an open set of P2P protocols, and the primary goal of the project is to provide a platform with the basic functions necessary for a P2P network. The project objectives are split into three:

- **Interoperability:** by using the proposed protocols P2P, various P2P services can discover and communicate with each other.
- **Platform independence:** the protocols are designed to be independent of programming languages and deployment platforms.
- **Ubiquity:** JXTA are designed to be accessible by any device with a digital heartbeat.

The JXTA Project is logically divided in three layers as depicted in Figure 2 on the following page.

---

<sup>7</sup><http://www.jabber.org>

<sup>8</sup><http://www.google.com/talk/>

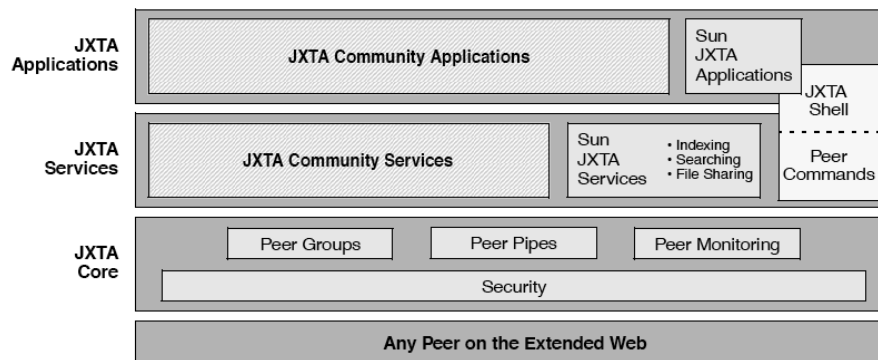


Figure 2: The three layer model of JXTA: Core, Services, and Applications layers

**Core:** This layer encapsulates essential primitives that are common to P2P networking, including peers, peer groups, discovery, communication, monitoring, and security primitives. This layer is ideally shared by all P2P devices so that interoperability is possible.

**Services:** This layer includes network services that is optional for a P2P network to provide, but are common or useful for P2P environments. Examples of network services include search and indexing, storage systems, and file sharing.

**Applications:** This layer includes among more: P2P instant messaging, content management and delivery, P2P email systems, distributed auction systems. The boundary between services and applications is not fixed, and an application to one user can be viewed as a service to another user.

The datamodel for JXTA is based on peers and peer groups. A peer is any device on the network that implements some of the JXTA protocols. Each peer operates independently from other peers, and is uniquely identified by a peer id. A peer can have one or more network interfaces for use with the JXTA protocols, and each of these interfaces can be advertised as peer endpoints. Peer endpoints are used when creating point-to-point connections (which in JXTA is called pipes) between two peers.

Peers can form transient or persistent relationships in peer groups. A peer group is a set of peers, again identified by a unique peer group id. A peer group can have its own membership policy from open to strictly protected. The motivation for creating and joining a peer group is often one of the following reasons: to create a secure environment, to create a scoping environment, or to create a monitoring environment.

In the Core layer, a number of peer group services are defined and optionally provided by a specific peer implementation for group management. This includes the following services:

- *Discovery Service:* Is used for members of a group to search for peer group resources shared between members. E.g., specific peers, other peer groups, or services.

- *Membership Service*: For a peer to join a peer group, the new peer has to be accepted by existing member or a representative part of them depending on the membership acceptance policy. This could include a vote among members which is accomplished through the Membership Service.
- *Access Service*: Controls permission for peers requesting access to other peers' resources or group services, if access control is enforced.
- *Pipe Service*: Manages pipe connections among group members.
- *Resolver Service*: Is used to send out queries to group members.
- *Monitoring Service*: Enables that a peer can monitor other peers in the same group.

For each of these Core layer services JXTA defines a protocol for querying and retrieving information.

Peers can offer network services on their own for other peers to use. Network services are divided into peer services and peer group services, depending on how the network service is shared. A peer service is only accessible from a single peer, and if that peer should brake down or loose network connection, the service is unavailable. A peer group service is composed of several instances of the same services on different peers in a peer group. If a single peer fails, another peer in the group can replace it, and the group service is not affected by this.

An implementation of a network service is in the JXTA runtime environment called a module. Modules can implement certain behaviors, where a behavior is an implementation independent module description. These module description are used to advertise other peers or peer groups about a module and should enable a peer to describe itself.

When two JXTA peers have made a connection through a pipe between network services, the basic unit of data sent is called a Message. A message is structured in message elements which is name-value pairs. Message handling including sending and receiving message are handled by the Pipe Service and the Endpoint Service. JXTA messages can be presented in two ways: in XML or as a binary format, an each software platform binding defines how a message is converted to and from a native data structure.

A key feature of the JXTA environment is the use of advertisements to broadcast descriptions of peer, peer groups, and services. Advertisements is language independent meta-data defined as XML documents. Peers discover resources by searching for advertisements describing the required resource. Several advertisement types are defined and below is a list of the ones which will later be used for comparison with datamodel of LightPeers:

- *Peer Advertisement*: is used to announce specific information about a single peer, such as peer id, name, and available endpoints.
- *Peer Group Advertisement*: describes the peer group id, name, and services.

- *Module Class Advertisement*: holds a description of a module class, and is used to announce the existence of a particular module class with module id, name, and description.
- *Module Spec Advertisement*: defines a module specification, by providing a reference to documentation on how to implement a certain module to create interoperable implementations. The advertisement contains furthermore a name, description, and id.
- *Module Impl Advertisement*: describes an implementation of a module specification, and contains a reference to the specification as a module spec id. It also contains name and unique id.

When advertisements and other messages are being transmitted between peers, JXTA supports an advanced security model including: confidentiality, authentication, authorization, data integrity, and refutability. This will not be described in depth here, because security have not yet been in focus for the LightPeers framework.

There is no requirements made by the JXTA protocols of how a peer should behave when receiving an advertisement or any other message with regards to forwarding or routing. In practice however, there typically exists four kinds of peers:

- *Minimal edge peer*: which can send and receive messages, but do not cache advertisements or messages for other peers.
- *Full-featured edge peer*: can send and receive messages, and often caches advertisements. Such a peer will typically reply to discovery requests based its own advertisements and cached advertisements from other peers.
- *Rendezvous peer*: is like a full-featured peer, but do also forward discovery requests on advertisements to help other peers discover resources.
- *Relay peer*: maintains routing information for routes of messages to other peers. Relay peers are often used to bridge different networks, e.g., for peers on each side of a firewall.

The Rendezvous and Relay kind of peers are not mutually exclusive, and can indeed be combined onto a single peer.

### 2.2.1 The JXME Project

One of the three main objectives for JXTA is ubiquity, that is to make JXTA accessible by any device with a digital heartbeat. This is hardly the case with current implementations of JXTA which is dominated by Java versions made for devices with desktop computer capabilities or more. A promising initiative for changing this, is the JXTA for Java 2 Platform, Micro Edition<sup>9</sup> (J2ME), also known as JXME (Hamada et al., 2005). The goal of the JXME 1.0 project is to implement a JXTA peer for the limited J2ME environment on, e.g., a mobile phone.

---

<sup>9</sup><http://java.sun.com/j2me/>

However, the JXME developers have not found it feasible to implement a full JXTA edge peer for the J2ME environment. Instead an approach where a proxy peer on a less limited device acting on behalf of the J2ME peer, is introduced. The Proxy peer: creates, publishes and discovers pipe advertisement, it creates, joins and discovers peer groups, and furthermore it compacts and relays all messages. The proxy peer stores all the incoming messages for the JXME peer, and then the J2ME peer periodically polls the proxy to get all the incoming messages for it.

The JXME 2.0 project called Arrakis is an initiative to overcome the shortcomings of JXME 1.0, thus to support a proxy-less version fully compliant and interoperable with other full JXTA peer. However, only little progress have been achieved so far<sup>10</sup>.

### 2.3 Proem

Proem is an open community platform for mobile P2P applications (Kortuem, 2002; Kortuem et al., 2001), and it provides a package with tools for development and deployment of mobile P2P applications. The vision for Proem includes a list of six objectives:

- **Adaptability:** Proem is designed to support timely response to changes in the operating environment, e.g., connectivity changes and resource availability.
- **Universality:** Proem is a platform and an architecture for building diverse mobile P2P systems, rather than a system for one particular purpose.
- **Interoperability:** between heterogenous hardware and software platforms is supported by the design of Proem.
- **Platform independence:** Proem is designed to be independent of programming language and system platforms. This is accomplished by using Web standards such as: HTTP and XML.
- **Extensibility:** Developers is allowed and should be able to change the internals of Proem's core components.
- **High-level development support:** The most prominent objective of the Proem design is to provide a simple yet powerful development platform.

Proem has been developed with research focus on how to build mobile ad hoc information systems which assist users in their everyday social interactions with other people, which could be on the work place, but also informal encounters during, e.g., shopping or hanging out with friends. Kortuem et al. (2001) refer to these proximity-based ad hoc interactions using mobile devices as impromptu collaboration.

The Proem architecture which is presented in Figure 3 on the next page, consist of three parts: the protocol stack, the peerlet engine, and a collection of services.

The datamodel for Proem consists of four types of entities: a peer which is any mobile device, an individual that can own and use one or multiple peers, data space

---

<sup>10</sup><http://jxme.jxta.org/currentwork.html>

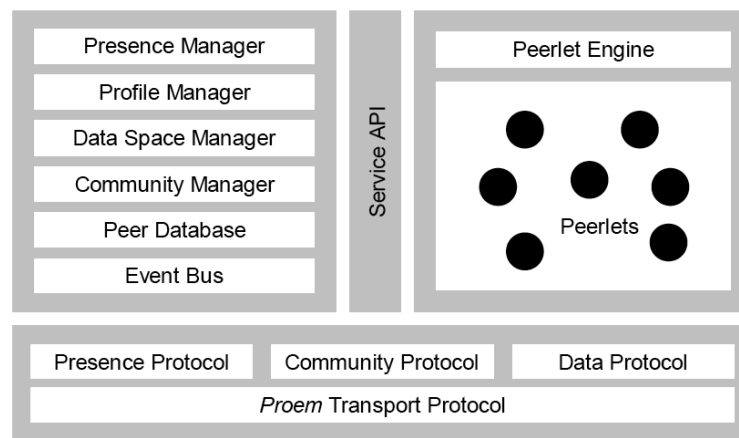


Figure 3: The Proem Runtime architecture consisting of: protocol stack, peerlet engine, and a collection of services

that is a set of data items managed by a group of peers, and a community which is a set of entities (peers, individuals, data spaces, and other communities). Each entity is identified by a name which is expressed by an URI, and one entity can optionally have multiple names. Another form of references to entities are profiles which are XML-based meta data structures.

The Proem protocols consists of:

- *Transport protocol*: which is a connectionless asynchronous communication protocol. Data transmitted through the transport protocol is represented by XML messages.
- *Presence protocol*: is used for peers to announce their presence throughout the network with profiles.
- *Data protocol*: defines messages that allow peers to share data items with data spaces. A data space is replicated on all the peers sharing it.
- *Community protocol*: is used for membership management of communities.

The Proem runtime system is just one instance of a peer that implements the Proem protocols. However, the Proem runtime system provides an event-based environment with the peerlet engine. The peerlet engine is responsible for the instantiation, execution, and termination of peerlets, and allows the peerlets to be added or removed at runtime.

The last part of the Proem architecture is a set of services that provides peerlets with access to commonly used functionality. The *presence manager* announce a peer's presence and discovers other peers nearby. The *data manager* controls persistent storage and access control. The *community manager* is responsible for keeping track of a peer's community memberships. The *peer database* holds a persistent log of encounters with other peers and supports that peerlets can store custom meta

data of other peers. And finally the *event bus* provides event-based communication for services and peerlets by using a publish-subscribe model. All of the services are in fact implemented as peerlets themselves and can be modified and reinstated by developers.

Furthermore, a peerlet development kit (PDK) has been assembled based on a Java implementation of the peerlet engine and services. The PDK offers Java APIs, Java interfaces, and classes for communication, data management, and event handling.

## 2.4 Discussion and Summary

P2P systems and architectures have changed over time due to shortcomings in existing architectures or exploitation of new possibilities, and this evolution have been presented as three different generations of P2P architectures. The JXTA platform which has been presented, belongs to the third of these generations where the use of two kinds of peers: peers and super-peers, is characterizing for that generation. JXTA is a full-fledged P2P architecture and provides a matured platform for desktop and laptop environments. These environments offer wideband network connections, high processing power, and large memory capacity, when compared to facilities of mobile devices. Third generation architectures can as such be used for mobile limited environments and if successful, the mobile peers seamlessly extends the stationary peer infrastructure. But as many of the third generation architectures are designed with desktop environment in mind, they are often not suitable to implement for mobile devices. This was also the the case with JXTA which was too heavy to run on a limited device. The reaction from the JXTA community was to develop a light version, JXME, for mobile devices, but JXME is made exclusively for the J2ME platform so far.

Instead of trying to make a comprehensive and heavy third generation architecture fit on a limited device, the Proem architecture takes its outset in designing for mobile limited devices. With the constraints of the mobile environments, some characteristics of second generation P2P architectures are becoming feasible again. The Proem architecture is, e.g., a pure P2P network without any super-peers. The problems with flooding as experienced with Gnutella is overcome by Proem, since connections are only possible between physically close peers, and thereby limiting the number of reachable peers making multicasting and broadcasting feasible.

Properties for the P2P architectures and systems are summarised in Table 1. Notice, how Proem has properties of earlier second generation P2P architectures, making it suitable for mobile P2P on limited devices.

The table includes two empty rows, namely Contingency handling and Context-aware. I see these two properties as important and interesting features of a P2P framework, and which existing frameworks lacks. Contingency handling is especially important since, ad hoc network on limited devices, are extremely vulnerable. I think of Context-awareness is a natural extension of the network, where not only the "network sensor" is used but also other sensor input from the physical context is used. My work with LightPeers is indeed motivated by these two challenges. In the next section I will describe the status of my work with LightPeers and how these two challenges are approached.

	<b>JXTA</b>	<b>JXME</b>	<b>Proem</b>
Designed for limited devices		x	x
Contingency handling			
Context-aware			
Extensible			x
Language independent	x		x
Pure P2P			x
Built-in Nat and firewall traversal	x	x	
Built-in discovery	x	x	x
Built-in security	x	x	x

Table 1: Comparison of P2P protocols and frameworks

### 3 The LightPeers Framework

LightPeers is a framework designed for mobile P2P systems, and the primary goal of the project is to provide a lightweight platform available on limited devices. The project objectives are split into six:

- **Lightweight:** the framework is designed for mobile P2P on limited devices.
- **Ubiquitous:** LightPeers is designed to be implementable on a variety of devices.
- **Platform independent:** the LightPeers framework is designed to be independent of programming languages and deployment platforms.
- **Interoperable:** LightPeers supports interoperability by using the XML-based SOAP message format, and a freeform tagging mechanism for resource description.
- **Context-aware:** LightPeers is designed to support sensors beyond discovering other peers, but also for, e.g., location sensors.
- **Contingency handling:** LightPeers should be prepared for break-downs and failures of peer devices and software components and have a contingency plan built-in for such situations.

The architecture and datamodel of the LightPeers framework is presented in the next sections followed by a comparison with existing frameworks.

#### 3.1 Architecture

The following description of the LightPeers framework takes a starting point in describing what a single device in a LightPeers network should implement to participate.

The LightPeers framework is divided into the five layers: Network Layer, P2P transport Layer, Service Layer, Application Layer, and Sensor Layer as seen in Figure 4.

The Network Layer contains components for handling the fundamental communication technology of the device. This could be all sorts of transceivers from traditional WiFi cards to Bluetooth or ZigBee hardware to more specialized and product specific hardware. The functionality that should be supported by each component in the Network Layer has to be able to transmit and receive data packets (messages). Thus a mechanism to split data packets into appropriate sized units according to hardware limitations should be supported at this layer. E.g. this could be the case for communication devices only offering the medium access control layer, as is the case for some ZigBee ready devices.

The P2P transport Layer encapsulates the basic components to support an ad hoc networking structure and data exchange. The key component in the framework is the *Discovery* component that facilitates devices running the LightPeers framework

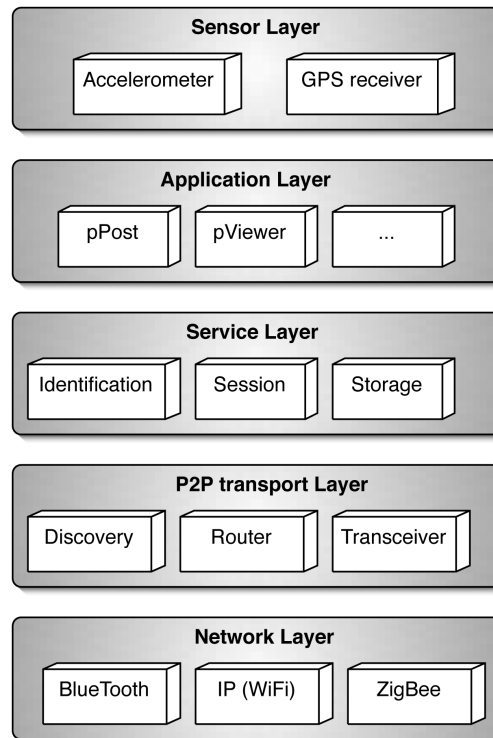


Figure 4: The five LightPeers Framework Layers: Network, P2P transport, Service, Application, and Sensor layers.

to discover each other. The component is responsible of maintaining an updated representation of the LightPeers network which the host device participates in. This is an especially important task as the framework is intended to be used with highly mobile devices, thus the network topology is likely to be of a very dynamic nature. The mobile devices are potentially less reliable as participants in a network than traditional stationary desktop computers or servers due to several reasons.

- The mobile devices are likely to be physically moved out of network reach,
- power supply for mobile devices is indeed a limited resource, and could be used up untimely, and
- as the devices typically both are handheld and operated by the user at the same time, they are in danger of being dropped or otherwise roughly handled.

The *Discovery* component should aggregate the discovery mechanisms for all of the Network Layer components. This may result in a situation where a certain device can be contacted through several means of communication media, e.g., through WiFi and ZigBee at the same time. In such a case the *Router* component decides, based on a simple priority metric, which communication media to send data packets through. The routing metric for this setup could be extended to include param-

eters such as stability of communication media and power usage. The *Router* component is tightly coupled to the *Transceiver* component. When the network topology is established by the *Discovery* component, the *Transceiver* component can be used as a single point of transmitting and receiving data packets for a LightPeers-enabled device. The *Transceiver* component includes message queues for incoming and outgoing data packets, and a newly arrived data packet in one of these queues is examined by the *Router* component for determining its next hop (if not already at destination) and priority.

The third layer is the Service Layer in which common service components are logically grouped. One of these, is the *Identification* component that is used to retrieve information about peers on a discovered device. The notion of a peer is here used as a digital identity representing a person or a software agent. In practice a mobile device is often also a personal device, thus a one-to-one correspondance between peer and device is the result. However, there is no conceptual restriction in the framework for a peer to be present at several devices, or a device having several peers present at the same time. A session is a group with one or more peers participating in a shared activity. The *Session* component offers state information about sessions that peers on a device are participating in. Notice, that the *Identification* and the *Session* components only handles information about local peers and sessions with local peers participating. The data being sent from and to the *Identification* and *Session* components are detailed in the datamodel section further down. The last component of the Service Layer is the *Storage* component, which is dedicated to handle shared storage among peers participating in a session, thus a session can have a session store where each peer can contribute appropriately.

The Application Layer is the fourth layer of the framework and is where the actual applications are logically placed. Thus, a LightPeers programmer would have access to the components of the Service Layer below and the Sensor Layer above. This is realised in the reference implementation by a skeleton application (pFoo) having references for these components from the start.

The final and fifth layer, the Sensor Layer, is where components for accessing sensor equipment are grouped. The reason to encapsulate sensors with components is to allow sharing of the equipment among applications on the same device, and to facilitate sharing of the sensor as a resource among peers participating in the same session. Each sensor component implements an interface that requires the component to implement: a simple data pull mechanism, and a notification mechanism when sensed data is available. Sensors of special interest for use on mobile devices is the movement and motion sensors. This information can be used actively to, e.g., tune the discovery protocol.

An alternative perspective on the basic components in the P2P transport Layer and the Service Layer can be useful to distinguish their functionality in the framework. One grouping of components handles how to: discover LightPeer devices, identify peers on devices, and exchange information about session state. These components handle information primarily used to maintain the ad hoc network infrastructure by examining their LightPeer context and offer information about their own state. In Object Oriented languages like Object C, C++, and Java this property of examining objects at runtime is called Object Reflection. Inspired by this, the group of components is called Reflection components, as can be seen in Figure 5. The other group-

ing contains components for ordinary data handling, i.e., transmitting and receiving data packets, routing data packets, and storing data. This group of components is therefore called Data handling components.

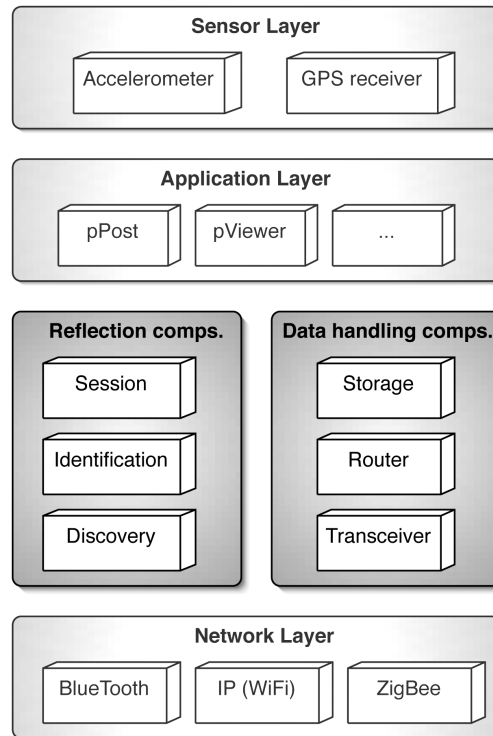


Figure 5: An alternative perspective on the P2P transport Layer and Service Layer where the components are grouped into Reflection components and Data handling components.

### 3.2 Datamodel

In the reference implementation the datamodel used to represent, e.g., a peer or a session, is expressed as XML structures. In fact each object in the datamodel implements an interface with the two methods *toXML* and *fromXML* that allow serialization and deserialization of an object state which is useful when transmitting an object between devices. The choice of using XML structured data for transmissions may seem in conflict with the idea of having a lightweight framework because of the overhead of ASCII characters being sent and the need of XML parsing at the destination. The reason for this choice is foremost because XML offers a simple way of expressing possibly complex data structures, and secondly because debugging of data flow is more accessible in the development phase. However, when need be, a compact version of the transmission protocol in a binary format, can be expressed as a mapping to and from the XML based datamodel. The LightPeers datamodel is quite simple and describes the following three object types: peers, sessions, and messages. Notice, that the two first objects belong to Reflection components and the

---

```

<?xml version="1.0"?>
<peer xmlns="http://lightpeers.daimi.au.dk">
  <name>foo</name>
  <guid>...</guid>

  <endpoints>
    <endpoint-guid>...</endpoint-guid>
    <endpoint-guid>...</endpoint-guid>
  </endpoints>
  ...
</peer>

```

---

Figure 6: The state of a peer from the LightPeers datamodel serialized as an XML structure.

last one to the Data handling components. The state of a peer which is serialized to XML and ready for transmission would look like the example shown in Figure 6.

Notice, that a peer in the LightPeers framework is a digital identity representing a person or a software agent. A peer is modelled with a `guid` element for unique identification, a `name` element for easy recognition, and a collection of device endpoints (`endpoint-guid`) where the peer can be contacted. The endpoints are referred to by their ids, which are again globally unique ids. This is the minimum required information about a peer, but further optional descriptions may be provided, e.g., an icon, person age or phone number.

A LightPeers session conceptually consists of a group of participating peers, a set of applications used by the peers to communicate and share data, and a list of shared resources (often hardware based) which are offered by endpoints at the devices in use. An XML example of such a session is presented in Figure 7, where the session also has a `guid` element for unique identification, a `name` element for easy human recognition, and a `creator` element that refers to the owner peer of the session. An important feature of a LightPeers session is the possibility to use tags to describe applications and shared resources of peers participating in the session. The tags are used like freeform meta-data as known and used by the systems: Del.icio.us<sup>11</sup>, Flickr<sup>12</sup>, and CiteULike<sup>13</sup>. Furthermore, the session description is allowed to contain more detailed or application specific elements, e.g., a logo or a highscore list. The session is a key concept in the LightPeers framework, because sessions define the context or environment of all peer collaboration.

Application specific data is packaged into LightPeers messages. The message format is structured as simple SOAP 1.2 (Gudgin et al., 2003) messages as seen in Figure 8 with a SOAP Header and Body elements. As with the data exchanged by the Reflection components, the SOAP based message format are not restricted to be represented as XML structures (Gudgin et al., 2003)[sec. 1.3]. In the Header element the participating endpoints, the recipients, the sender, the message id, and the expire-time of the message is listed. The message protocol is based on a simple multicast to the peers participating in the session. The Body element contains the content of

---

<sup>11</sup><http://del.icio.us/>

<sup>12</sup><http://www.flickr.org/>

<sup>13</sup><http://www.citeulike.org>

---

```

<?xml version="1.0"?>

<session xmlns="http://lightpeers.daimi.au.dk">

  <name>Fieldtrip session</name>
  <guid>...</guid>

  <creator>
    <peer>...</peer>
  </creator>

  <participants>
    <peer>...</peer>
    <peer>...</peer>
    ...
  </participants>

  <applications>
    <application>
      <guid>...</guid>
      <name>pPost</name>
      <tags>
        <tag>pPost</tag>
        <tag>IM</tag>
      </tags>
    </application>
    <application>
      <guid>...</guid>
      <name>HyConExplorer</name>
      <tags>
        <tag>HyConExplorer</tag>
        <tag>hypermedia</tag>
        <tag>browser</tag>
      </tags>
    </application>
  </applications>

  <shared resources>
    <resource>
      <tags>
        <tag>storage</tag>
      </tags>
      <endpoint>...</endpoint>
    </resource>
    <resource>
      <tags>
        <tag>accelerometer</tag>
        <tag>bluetooth</tag>
      </tags>
      <endpoint>...</endpoint>
    </resource>
    ...
  </shared resources>
  ...
</session>

```

---

Figure 7: The state of a session from the LightPeers datamodel serialized as an XML structure.

---

```

<?xml version="1.0"?>
<soap:Envelope xmlns:soap=\
"http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://lightpeers.daimi.au.dk">
  <soap:Header xmlns:lp=\
"http://lightpeers.daimi.au.dk/message">
    <lp:session-endpoints>
      <lp:endpoint lp:visited="false">...</lp:endpoint>
      <lp:endpoint lp:visited="true">...</lp:endpoint>
    </lp:session-endpoints>
    <lp:recipients>
      <lp:peer lp:visited="false">...</lp:peer>
      <lp:peer lp:visited="false">...</lp:peer>
    </lp:recipients>
    <lp:sender>
      <lp:peer>...</lp:peer>
    </lp:sender>
    <lp:message-info>
      <lp:message-id>...</lp:message-id>
      <lp:expire-time>...</lp:expire-time>
    </lp:message-info>
  </soap:Header>
  <soap:Body xmlns:m=\
"http://lightpeers.daimi.au.dk/pPost">
    <m:content>
      Hi Morten,
      I have added some nice pictures from the field
      trip to the session storage. See ya, Bent
    </m:content>
  </soap:Body>
</soap:Envelope>

```

---

Figure 8: A LightPeers message produced from the pPost application.

the message.

### 3.3 Contingency handling

The challenge of supporting contingency handling has been approached together with visiting professor Ken Anderson from University of Colorado, Boulder. We have discussed various architectures and framework, and the design is still under development. Our work so far, is the break the down the contingency handling mechanism into two parts: a debug/monitor architecture and a contingency handling engine.

The Debug Architecture is depicted in figure where three devices with software components are presented in the example. These components are LightPeers components or could in fact be any component on the device.

The basic setup is as follows: The sensors in the Debug architecture are a type of components used to monitor one or more components. This could both local or

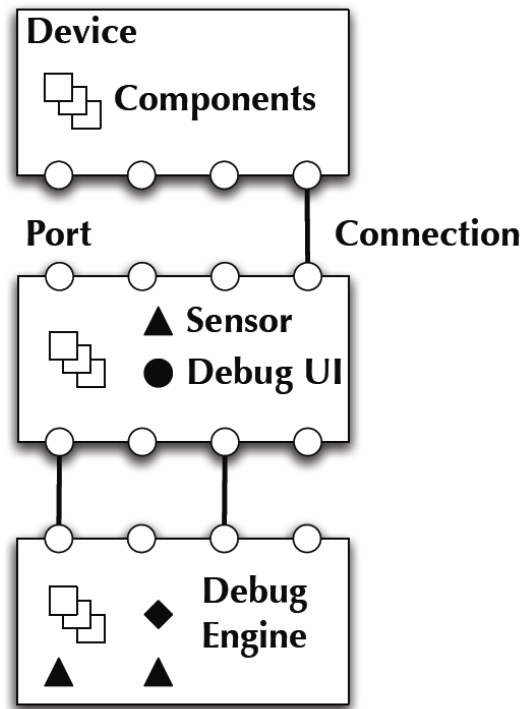


Figure 9: The Debug Architecture

remote components. The Debug engine is a components that communicate only to sensors, other debug engines, or debug UIs. A UI is a type a component which is able to display information provided by a debug engine.

The debug engines should be provided with a description of the components which are to be monitored. The description or configuration contains the devices, the type and number of components on each device, and the connections that exist in setup. The debug engines should be able to deploy sensors locally or remotely on other devices. The setup could be implemented with direct connections between engines and sensors, or an event based publish-subscribe model could be used.

Components that are monitored by a sensor, should implement a pre-defined interface. A different approach is to hardwire the sensor with(in) a collection of components, if somehow they cannot be contacted via an interface. However, if the sensor is tightly coupled with the components in this way, the sensor is also likely to be inflicted by the same break-downs and failures as the components it monitors.

Even though the basic setup of the Debug architecture is becoming stable, a lot of open questions still need to be answered. Open issues include: Do all sensors generate the same type of events? Do we want to have only one sensor per application? How is a sensor identified and found?

Issues regarding the debug engines are also being discussed. E.g., if two debug engines exist in the same setup description and one of them break-downs, how can the

other debug engine take over the monitoring from the first debug engine? How do we support setup description switching at runtime?

Work ahead of us is the implement prototypes of the debug architecture components: sensors, debug engine, UI, for the LightPeers components. The actual contingency handling should be based on the information from the debug architecture, and this is also future work.

### 3.4 Implementations

The first platform chosen was the Nokia 6600 Symbian mobile phone with integrated Bluetooth. This seemed ideal, since the platform is quite ubiquitous and relatively low cost. The first implementation demonstrated that Bluetooth was not the most suitable technology for very dynamic and ad hoc networks, since the discovery time of other devices could be around 10 seconds. The Bluetooth implementation on the Nokia 6600 was furthermore a bit more restricted than first assumed, because it did not have support for Bluetooth Scatternets. This limited network to simple piconets to be of a maximum size of seven peers theoretically. On the Nokia 6600 only three connections at a time was supported. A new platform was chosen with the requirement of being mobile and have an integrated WiFi card. So, the Pocket PC with integrated WiFi was chosen as platform for developing the reference implementation.

The second prototype of the LightPeers framework has been implemented on the .Net Compact Framework environment in the C# language. The hardware platform used is the Fujitsu Siemens Pocket LOOX 720 with an Intel XScale PXA272 processor 520 MHz, a display resolution of 480 x 640 and 16 bits color depth. It has Bluetooth, Wi-Fi integrated, and a CompactFlash, SD/MMC (SDIO compatible) expansion slot.

### 3.5 Discussion and Summary

In this section, the LightPeers framework has been introduced. The framework development is focused on facilitating mobile P2P applications by making the framework: lightweight, ubiquitous, platform independent, interoperable, context-aware, and able to handle contingencies.

LightPeers is not the only initiative of developing a framework for mobile P2P applications on limited devices. As described and discussed in the previous section 2 on page 5 about P2P architectures, the JXME and Proem architectures also aim at providing common P2P functionality for mobile devices, but in different ways. JXME inherits some strong properties from its JXTA proxy peer including security and firewall traversal, which is left out (for now) in the design of LightPeers. Security in LightPeers is only present if application developers implements support for confidentiality and authentication as part of their application.

Proem allows developers to modify and extend core services in the runtime environment. This is powerful feature which makes the platform extensible and updatable even at runtime. LightPeers has not support for such a feature yet, but the possibility to exchange applications and services among peers has been considered. This

	<b>LightPeers</b>	<b>JXTA</b>	<b>JXME</b>	<b>Proem</b>
Designed for limited devices	x		x	x
Contingency handling	x			
Context-aware	x			
Extensible				x
Language independent	x	x		x
Pure P2P	x			x
Built-in Nat and firewall traversal		x	x	
Built-in discovery	x	x	x	x
Built-in security		x	x	x

Table 2: Comparison of P2P protocols and frameworks including LightPeers

feature also implies a need for basic security or a trust mechanism, as applications from alien peers potentially could behave maliciously.

JXTA uses advertisements to describe resources, and for the JXTA modules alone there exists three kinds of advertisements, namely module class advertisements, modules spec advertisements, and module advertisements. In LightPeers services and resources in general are described with freeform tags, which is very lightweight compared to JXTA. The tags themselves do not as such dictate a certain type or semantic, but the semantics of the tags are negotiated implicitly by the manner in which they are used. Thus, the same tag can have widely different meaning across different peer communities or domains. This could be described as domain specific interoperability.

An inherent feature of mobile devices is change of physical location and context, and with LightPeers a sensor layer for handling shared sensors for, e.g., location has been designed, but not fully implemented yet. This is inspired by the area of Sensorwebs and my earlier work with context-aware hypermedia, where the notion of context for mobile devices proved useful (Bouvin et al., 2003).

The main contributions so far in fulfilling the objectives of LightPeers can be summarised as the following:

- design of a lightweight framework, including datamodel and protocols, for mobile P2P able to run on devices with limited resources,
- proposal of a freeform community tagging mechanism of applications and resources, and
- distilling the handling of shared sensors in a dedicated logical layer of the framework.

In Table 2 the comparison between P2P protocol and framework presented in table 1 on page 14 have been updated with the properties of the LightPeers framework.

There still is a lot of work to be done with the LightPeers project. The plan for the work ahead of me includes:

- to have a stable LightPeers setup or kit which can be used for evaluations,
- to continue the work with Ken Anderson on the Debug Architecture, and
- to build an actual contingency handling engine on top of the Debug Architecture.

## 4 Prospective Application Domains

The current interest and popularity of P2P systems for common use as well as in research started out with file sharing applications in the years 1999 and 2000. Although file sharing probably still today is the best-known application of P2P systems, applications of P2P systems go beyond that, and can be divided into at least the following four categories (as introduced by Androutsellis-Theotokis and Spinellis (2004)) across generations: Information sharing, Communication, Collaboration, and Distributed computation .

- **Information sharing** refers to applications where peers share information. Examples of such systems are: Napster, Freenet, and BitTorrent (Clarke, 1999; Cohen, 2002; Fanning, 1999).
- **Communication** is crucial when collaborating, and the communication needs only to be directly between peers. Among such systems exist: ICQ (ICQ server maps from ICQ-specific addresses to IP addresses, the communication is P2P) and Skype (Baset and Schulzrinne, 2004).
- **Collaboration** between humans is in essence P2P, and the collaboration situation is supported by the system. Systems of this kind include: Groove<sup>14</sup> and Distributed Knight (Damm and Hansen, 2002).
- **Distributed Computation** is where otherwise unused computing power and network bandwidth at the "edge" of the Internet is organised in a P2P network and each peer is used for computing a small fraction of a large problem space. E.g., The Grid<sup>15</sup> and GreenTea<sup>16</sup>.

These applications of P2P systems are examples of dedicated systems for a specific task. The LightPeers framework can be used for applications in all four of these categories. The prospective applications of LightPeers which have been selected for further work and for evaluating the soundness and robustness of LightPeers, include: LightPeers based Sports Equipment, a LightPeers-enabled nomadic learning, and LightPeers as backbone for the pervasive play concept Mock Games.

### 4.1 Interactive Sports Equipment

Research in developing innovative interactive sports equipment and new interaction techniques has been discussed in the Center of Interactive Spaces and an application on SmartGym for the Højteknologifonden on this topic has been made. My work with the LightPeers framework has been the used as technological approach in the application.

In such a project the goal would be to go beyond the traditional pulse watch for runners, and enrich classical sports equipment, e.g., for sports and gymnastics as well as invent new types of equipment.

<sup>14</sup>Groove - Virtual office. <http://www.groove.net/>.

<sup>15</sup><http://www.grid.org>

<sup>16</sup><http://www.greenteatech.com/>

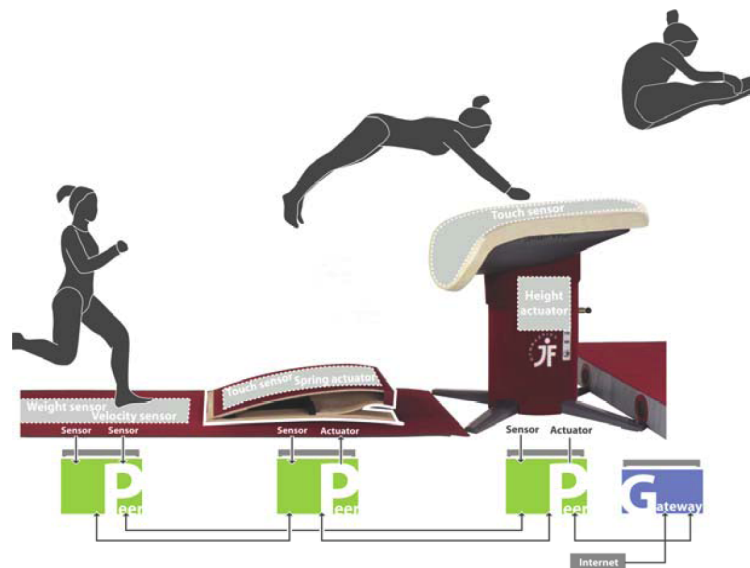


Figure 10: Gym horse setup with P2P sensing enabled by LightPeers

An example of enriching traditional sports equipment could be a setup of springboard enhanced with a peer controlling spring-strength, a mattress enhanced with a weight and a velocity sensing peer, and a gym horse with a touch sensor peer and height adjustment actuator peer as depicted in Figure 10. The setup as a whole will log data to be used later on for debriefing or for being used in math or physics classes as real life data examples. The sensor peers could furthermore measure the condition of the equipment for maintenance purposes.

Using the LightPeers framework for the SmartGym project would provide a challenging evaluation application of the sensor support and handling in LightPeers. In addition logged data from the SmartGym equipment would be easy accessible in conjunction with a LightPeers-enabled nomadic learning. The peers in this scenario are not very mobile, since the sports equipment forms a group.

## 4.2 Nomadic Learning

From our studies conducted at the Center for Interactive Spaces of public school activities (Brodersen et al., 2005; Iversen and Nielsen, 2003; Nørregaard et al., 2003) we see that learning becomes more project oriented and nomadic in the sense, that pupils spend their daily life as "nomads" in transit between many physical places ("oases") such as classrooms, labs, workshops, libraries, museums, the city, nature, clubs, and at home.

The hypermedia system HyConExplorer (Bouvin et al., 2004; Hansen et al., 2004b) was built to support nomadic and project oriented learning characterised by ongoing production of material anywhere with the devices at hand. This also involves field trips (e.g., for sample taking in a biology lesson or interviews with people outside the school area), where material is collected and shared. The HyConExplorer is

build upon the HyCon framework which uses a server-centric network model. Thus, when pupils collect material in the field with mobile phones, tablets, and laptops the material is transmitted back to the central server through often very unstable and thin network connections such as GPRS.

By using LightPeers as underlying network layer on the various devices, support for ad hoc communication and collaboration among group members would be accomplished. Material could be shared among peers in the field without having to be transmitted back and forth to the server through cumbersome connections. If a particular device in a peer group in the field should have a wideband network connection, this connection could be shared among peers with LightPeers.

An interesting challenge of using LightPeers with an existing application such as HyConExplorer (which is actually a suite of applications for numerous devices) would be, to make the integration without having to change the original application, in this case HyConExplorer. Thus, in general to be able to LightPeers-enable applications which were not prepared for P2P networking from the outset. A similar approach from the hypermedia research area is known as open hypermedia (Christensen and Hansen, 2002; Davis et al., 1994; Grønbaek et al., 1999), where applications not prepared for linking from the start are augmented with linking facilities by an external open hypermedia system. The peers are semi-mobile because they typically move around in project group in this scenario.

### 4.3 Mock Games

Mock Games is a concept for pervasive play described by Martin Brynskov and Martin Ludvigsen also affiliated with the Nomadic Play project in the Center of Interactive Spaces. Based on experiences from the cooperative design workshops with children that we organized and described (Brynskov et al., 2005), Brynskov and Ludvigsen have elaborated further on their ideas resulting in the Mock Games concept and the paper "Mock Games: A New Genre of Pervasive Play" submitted for publication.

Their definition of Mock Games states: *A Mock Game is a type of peer interaction that combines elements of pervasive gaming and social transformative play. It is a role-based game of emergence involving social reality, explicitly formed by and forming communities. It invites humor and friendly conflict as primary ingredients in social interaction. Real identities are constructed and blended with play roles.*

The LightPeers framework could be used as backbone technology for devices used for the peer interaction proposed by Mock Gaming. It would be an interesting evaluation to offer the LightPeers framework as a testbed for both the developers of Mock Games and the actual Mock Gaming. The peers are highly mobile in this scenario.

## 5 Future Work and Conclusion

### 5.1 Future Work

From the start of March until August/September 2006 I am visting professor Blair MacIntyre at Georgia Tech, Atlanta. I this period I plan to continue my work with LightPeers, but I also wish to combine it with projects conducted at MacIntyre's Augmented Environment Lab, to be part of their research enviroment. My goal is to fully participate in their research enviroment, and work with LightPeers, resulting in a paper on LightPeers for augmented enviroments.

Furthermore, my work with the debug architecture together with Ken Anderson for mobile P2P framework, is ongoing and should be able to continue when I am abroad. The debug architecture would be a very strong mechanism when developing application upon LightPeers and in fact for the actual developing of the LightPeers framework itself. This should hopefully result in a paper on a distributed debugging architecture for a mobile P2P framework.

Finally, a concluding paper on domain specific demands for mobile P2P frameworks should wrap up my experiences with using LightPeers in various domains and hopefully be able to extract a generic substrate.

Contingency handling is a huge and interesting research area, and could be build on top of the debug architecture. However, the work with implementing contingency handling it is probably realisticly reaching beyond the time limit of my project.

### 5.2 Conclusion

In this report, the research results of the first part of my Ph.D. studies has been described, and directions on areas for potential future work were presented.

The research issues have been investigated, and the results of the investigation have been presented. The first issue was:

*Which requirements should be prioritized and met by a lightweight architecture and framework, including datamodel and protocols of framework components, for supporting mobile ad hoc P2P network for limited devices?*

This included a survey of P2P architectures and systems, which resulted in a design for the LightPeers framework with the objectives of being: lightweight, ubiquitous, platform independent, interoperable, context-aware, and able to handle contingency. In fulfilling these objectives the main contributions so far have been: a proposal of a freeform community tagging mechanism of applications and resources, and distilling the handling of shared sensors in a dedicated logical layer of the framework.

The second issue was:

*Which domain specific demands exist in the prospective application domains, and can a generic substrate be identified and supported?*

Several prospective application domains have been presented, namely: a LightPeers-enabled for nomadic learning, a LightPeers based SmartGym, and LightPeers as backbone for the pervasive play concept Mock Games. As implementing the LightPeers framework still is work in progress, the prospective application domains have not been investigated thoroughly for domain specific demands yet. However, for a LightPeers-enabled HyConExplorer, an interesting challenge would be, to make the integration without having to change the original application. Using the LightPeers framework for an Interactive SchoolGym would provide a challenging evaluation application of the sensor support and handling in LightPeers. The LightPeers framework used as backbone technology for devices used for Mock Gaming, would be an interesting evaluation of the framework as a testbed for the developers of Mock Games and the actual Mock Gaming.

## References

- Androutsellis-Theotokis, S. and Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371.
- Baset, S. A. and Schulzrinne, H. (2004). An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. <http://www1.cs.columbia.edu/~library/TR-repository/reports/reports-2004/cucs-039-04.pdf>.
- Bondolo, Jaltman, Tra, and Yeager (2004). JXTA v2.0 Protocol Specification. <http://spec.jxta.org>.
- Bouvin, N. O., Brodersen, C., Hansen, F. A., Iversen, O., and Nørregaard, P. (2004). Tools of contextualization: Extending the class room to the field. Submitted for publication to the Personal and Ubiquitous Computing Journal.
- Bouvin, N. O., Christensen, B. G., Grønbæk, K., and Hansen, F. A. (2003). HyCon: A framework for context-aware mobile hypermedia. *The New Review of Hypermedia and Multimedia*, 9:59–88.
- Bouvin, N. O., Christensen, B. G., Hansen, F. A., and Nielsen, K. L. (2005). Supporting mobile and nomadic learning. In *WWW '05: Workshop Proceedings of the 14th international conference on World Wide Web*.
- Brodersen, C., Christensen, B. G., Grønbæk, K., Dindler, C., and Iversen, O. S. (2004). eBag - the Digital School Bag. In *Proceedings of the Fourth Danish Human-Computer Interaction Research Symposium*.
- Brodersen, C., Christensen, B. G., Grønbæk, K., Dindler, C., and Sundararajah, B. (2005). ebag: a ubiquitous web infrastructure for nomadic learning. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 298–306, New York, NY, USA. ACM Press.
- Brynskov, M., Christensen, B. G., Ludvigsen, M., Collins, A., and Grønbæk, K. (2005). Designing for nomadic play: A case study of participatory design with children. In *Proceedings of the 4th International Conference for Interaction Design and Children*.
- Christensen, B. G. (2005). Do Social Computing Make You Happy? A Case Study of Nomadic Children in Mixed Environments. In *Proceedings of "Ambient Computing in a Critical, Quality of Life Perspective", under The Fourth Decennial Aarhus Conference*.
- Christensen, B. G. (2006). LightPeers - A Framework Supporting Nomadic Learning in Mixed Environments with Mobile Ad Hoc Networking. In *Proceedings of the 3rd MiNEMA Workshop on Middleware for Mobile Environments*.
- Christensen, B. G. and Hansen, F. A. (2002). XLink—linking the Web and open hypermedia. In *Proceedings of the Open Hypermedia Systems Working Group Meeting 8.0*, pages 9–18. FernUniversität Hagen, Germany.
- Christensen, B. G., Hansen, F. A., and Bouvin, N. O. (2003). Xspect: bridging open hypermedia and XLink. In *Proceedings of the 12<sup>th</sup> International World Wide Web Conference*, pages 490–499, Budapest, Hungary. W3C, ACM Press.

- Clarke, I. (1999). Freenet. <http://freenetproject.org/>.
- Cohen, B. (2002). BitTorrent. <http://www.bittorrent.com/>.
- Damm, C. and Hansen, K. (2002). Distributing Knight — Using type-based publish/subscribe for building distributed collaboration tools. In *Proceedings of NWP-ER*.
- Davis, H. C., Knight, S., and Hall, W. (1994). Light hypermedia link services: A study of third party integration. In *Proceedings of the 1994 ACM European Conference on Hypermedia Technology*, pages 41–50, Edinburgh, UK. ACM.
- Fanning, S. (1999). Napster. <http://www.napster.com>.
- Floyd, C. (1984). A systematic look at prototyping. In Budde, editor, *Approaches to Prototyping*, pages 1–18. Springer Verlag.
- Frankel, J. and Pepper, T. (2002). Gnutella. <http://www.gnutella.com>.
- Greenbaum, J. and Kyng, M., editors (1991). *Design at Work - Cooperative Design of Computer Systems*. Lawrence Erlbaum Associates Publishers., Hillsdale, NJ.
- Grønbaek, K., Sloth, L., and Ørbæk, P. (1999). Webwise: browser and proxy support for open hypermedia structuring mechanisms of the World Wide Web. In *Proceedings of the 8<sup>th</sup> International World Wide Web Conference*, pages 253–267, Toronto, Canada. W3C.
- Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., and Nielsen, H. F. (2003). SOAP 1.2. W3C Recommendation, W3C. <http://www.w3.org/TR/soap12-part1/>.
- Hamada, Kuldeep, and Tra (2005). JXTA-J2ME 2.0. <http://jxme.jxta.org/>.
- Hansen, F. A., Bouvin, N. O., Christensen, B. G., Grønbaek, K., Pedersen, T. B., and Gagach, J. (2004a). Integrating the web and the world: Contextual trails on the move. In *Proceedings of the 15th ACM Conference on Hypertext and Hypermedia - HT04*. ACM.
- Hansen, F. A., Bouvin, N. O., Christensen, B. G., Grønbaek, K., Pedersen, T. B., and Gagach, J. (2004b). Integrating the Web and the World: Contextual trails on the move. In de Roure, D. and Ashman, H., editors, *Proceedings of the 15<sup>th</sup> ACM Hypertext Conference*, Santa Cruz, CA, USA. ACM Press.
- Hansen, F. A., Christensen, B. G., and Bouvin, N. O. (2005). RSS as a Distribution Medium for Geo-spatial Hypermedia. In *Proceedings of the 16th ACM Conference on Hypertext and Hypermedia*.
- Harjula, E., Ylianttila, M., Ala-Kurikka, J., Rieki, J., and Sauvola, J. (2004). Plug-and-play application platform: towards mobile peer-to-peer. In *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 63–69, New York, NY, USA. ACM Press.
- Iversen, O. S. and Nielsen, C. (2003). Using digital cultural probes in design with children. Poster paper in the Proceedings of the Interaction Design and Children 2003 conference (IDC'03).

- Kortuem, G. (2002). Proem: a middleware platform for mobile peer-to-peer computing. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):62–64.
- Kortuem, G., Schneider, J., Preuitt, D., Thompson, T. G. C., Fickas, S., and Segall, Z. (2001). When peer-to-peer comes face-to-face: Collaborative peer-to-peer computing in mobile ad hoc networks. In *P2P '01: Proceedings of the First International Conference on Peer-to-Peer Computing (P2P'01)*, page 75, Washington, DC, USA. IEEE Computer Society.
- Kwok, S. H., Lui, S. M., Cheung, R., Chan, S., and Yang, C. C. (2003). Searching behavior in peer-to-peer communities. In *ITCC '03: Proceedings of the International Conference on Information Technology: Computers and Communications*, page 130, Washington, DC, USA. IEEE Computer Society.
- Nørregaard, P., Andersen, J., Dindler, C., Frich, J., Iversen, O. S., and Nielsen, C. (2003). Networking news—a method for engaging children actively in design. In *Proceedings of the 26<sup>th</sup> Information Systems Research Seminar in Scandinavia*.
- Poulsen, S., Fjord-Larsen, M., Hansen, F. A., and Christensen, B. G. (2001). Visualizing guided tours with w3d. In *Proceedings of the 12th ACM Conference on Hypertext and Hypermedia*.
- Ripeanu, M. (2001). Peer-to-peer architecture case study: Gnutella network. In *P2P '01: Proceedings of the First International Conference on Peer-to-Peer Computing (P2P'01)*, page 99, Washington, DC, USA. IEEE Computer Society.
- Saint-Andre, P. (2004). Extensible Messaging and Presence Protocol (XMPP): Core. Request for Comments: 3920, Jabber Software Foundation. <http://www.ietf.org/rfc/rfc3920.txt>.
- Sandvad, E., Grønbæk, K., Sloth, L., and Knudsen, J. L. (2001). A metro map metaphor for guided tours on the Web: the Webwise guided tour system. In *Proceedings of the 10<sup>th</sup> International World Wide Web Conference*, pages 326–333, Hong Kong. W3C.
- Weiser, M. (1991). The computer for the 21<sup>st</sup> century. *Scientific American*, 265(3):66–75.
- Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Commun. ACM*, 36(7):75–84.

## A Publications

1. Bouvin, N. O., Christensen, B. G., Grønbæk, K., and Hansen, F. A. (2003). Hy-Con: A framework for context-aware mobile hypermedia. *The New Review of Hypermedia and Multimedia*, 9:59–88.
2. Brodersen, C., Christensen, B. G., Grønbæk, K., Dindler, C., and Iversen, O. S. (2004). eBag - the Digital School Bag. In *Proceedings of the Fourth Danish Human-Computer Interaction Research Symposium*.
3. Brodersen, C., Christensen, B. G., Grønbæk, K., Dindler, C., and Sundararajah, B. (2005). ebag: a ubiquitous web infrastructure for nomadic learning. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 298–306, New York, NY, USA. ACM Press.
4. Brynskov, M., Christensen, B. G., Ludvigsen, M., Collins, A., and Grønbæk, K. (2005). Designing for nomadic play: A case study of participatory design with children. In *Proceedings of the 4th International Conference for Interaction Design and Children*.
5. Christensen, B. G. (2005). Do Social Computing Make You Happy? A Case Study of Nomadic Children in Mixed Environments. In *Proceedings of "Ambient Computing in a Critical, Quality of Life Perspective", under The Fourth Decennial Aarhus Conference*.
6. Christensen, B. G. (2006). LightPeers - A Framework Supporting Nomadic Learning in Mixed Environments with Mobile Ad Hoc Networking. In *Proceedings of the 3rd MiNEMA Workshop on Middleware for Mobile Environments*.
7. Christensen, B. G. and Hansen, F. A. (2002). XLink—linking the Web and open hypermedia. In *Proceedings of the Open Hypermedia Systems Working Group Meeting 8.0*, pages 9–18. FernUniversität Hagen, Germany.
8. Christensen, B. G., Hansen, F. A., and Bouvin, N. O. (2003). Xspect: bridging open hypermedia and XLink. In *Proceedings of the 12<sup>th</sup> International World Wide Web Conference*, pages 490–499, Budapest, Hungary. W3C, ACM Press.
9. Hansen, F. A., Bouvin, N. O., Christensen, B. G., Grønbæk, K., Pedersen, T. B., and Gagach, J. (2004). Integrating the web and the world: Contextual trails on the move. In *Proceedings of the 15th ACM Conference on Hypertext and Hypermedia - HT04*. ACM.
10. Hansen, F. A., Christensen, B. G., and Bouvin, N. O. (2005). RSS as a Distribution Medium for Geo-spatial Hypermedia. In *Proceedings of the 16th ACM Conference on Hypertext and Hypermedia*.
11. Poulsen, S., Fjord-Larsen, M., Hansen, F. A., and Christensen, B. G. (2001). Visualizing guided tours with w3d. In *Proceedings of the 12th ACM Conference on Hypertext and Hypermedia*.