

# Providing Presence within P2P Systems

James Walkerdine, Lee Melville, Ian Sommerville  
Computing Department, Lancaster University, Lancaster, UK  
{walkerdi, l.melville, is} @comp.lancs.ac.uk

## ABSTRACT

There is an increasing interest in incorporating presence within Peer-to-Peer systems (P2P). However the diverse nature of P2P can have an effect on how easily presence functionality can be integrated within a design. This paper examines how presence can be supported within P2P systems, and in particular what affect the choice of underlying logical network architecture can have on this.

## 1. Introduction

A common definition for presence is the state that a user, application or hardware is in [1]. For example, a user specifying if they are free or busy. This information is made available to the rest of a system so it can be viewed by others entities (and reacted to if needs be).

The ability to convey such presence information is increasingly becoming an important aspect of many systems. In particular, presence has played a key role within the areas of CSCW (where it is commonly referred to as *awareness*), distributed systems (being aware of what services exist) and Grid computing (being aware of which nodes in the grid are available to carry out computational processing).

The use of presence brings with it contextual information which in turn can provide advantages within a system. For example, it can assist co-operation by allowing users to see whether or not other users are busy, or be used to optimise distributed computation by allowing a system to be aware of when a node is connected [2].

Presence can also play a role within industrial settings and can be particularly important for large organisations that are globally distributed. A notable problem that is often experienced is the amount of time it takes to resolve issues that involve people from more than one site [3]. In such circumstances, presence information could be used to alleviate problems by informing distant colleagues who is available, and when they are available [4].

Within Peer-to-Peer (P2P) systems presence has commonly been used in Instant Messaging applications such ICQ[5] and MSN[6], as well as in file sharing

applications such as Napster [7]. Although it has been generally used to capture whether a user or peer is on-line/off-line, presence does not have to be limited to this and could convey location, contextual, activity or application-specific information.

However the nature of P2P means that there are numerous ways in which a system can be designed, deployed and operated. This is particularly the case with respect to the underlying logical network architecture that is used, which, as shown in [8][9], can have significant impact on the properties of a system.

This paper examines how presence can be supported within P2P systems, and in particular provides an analysis of the effect the choice of underlying logical network architecture can have on the provision of presence functionality.

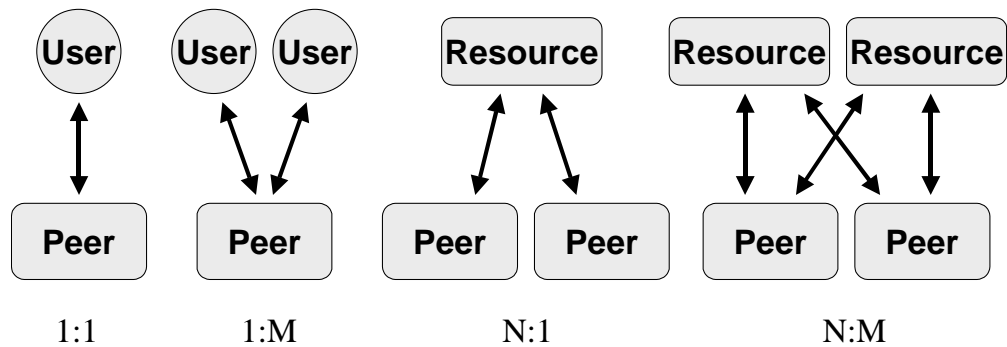
The paper begins by first discussing the main sources of presence within a P2P system. It then proceeds to identify the main technical issues that need to be addressed in order for presence to be supported, before examining the affect the choice of logical network architecture can have on resolving these issues.

The paper also provides a brief discussion of other more general design issues that may need to be considered, and how the choice of logical network architecture can also have an influence on these. The issues discussed include privacy, controlling information and real-time consistency.

## 2. Sources of Presence Information

It is possible to split presence within a P2P system into two main categories, *peer presence* and *abstract presence*.

**Peer presence** represents information that is available about the peers that are located on the network. This information typically includes whether or not the peer is currently online, but can also include other information such as the IP address of that peer or perhaps information about its network connectivity.



**Figure 1 - Possible relationships between the abstract and peer layers**

**Abstract presence** represents information that is available about the entities that utilise the peers, or represents information that is available about the peers' environment. Types of abstract presence can include users, resources or even the peer's physical environment.

- *User presence* represents information that is available about the user of a peer. This typically focuses on whether or not the user is online, but can also represent information about a user's state. For example, this could be whether or not the user is busy, or is away from their computer. There are some issues that need to be considered with user presence, however. For example, although the peer may be connected to the network, the user could have registered themselves as being off-line. Furthermore, it is perfectly possible that multiple users may make use of a single physical peer, i.e. share a computer.
- *Resource presence* represents information that is available about the resources that are connected to the peer. What constitutes a resource is difficult to fully define, however they do seem to fall into two categories, *internal* and *external* resources. An internal resource can be regarded as being those resources that contribute to the actual system, for example processing power, hard disc space, bandwidth, services, etc. External resources can be regarded as being those resources that are independent of the system and play no role within its function. These, for example, can be MP3's or documents that can be shared. Typically a P2P system supports the communication/utilisation of external resources. For example, Napster or Gnutella [10].
- *Physical presence* represents information about the physical environment of the peer. This could include location information [11], audio and video information (for example, with the use of web cams) and information about the current

environment (for example, what web site is the user currently browsing, what file are they editing, etc).

Although Peer and Abstract presence are quite distinct here they are, in reality, intrinsically linked together as the abstract layer cannot exist without a peer layer. Abstract presence can essentially be considered as an extension of peer presence. However, the relationship between the two does not have to be one to one. This is illustrated in figure 1.

An example 1:1 relationship between peer and abstract layers is that of a single user making use of a single peer, e.g., a user utilising an instant messenger application on their PC.

An example 1:M relationship between peer and abstract layers is where multiple users make use of a single peer, e.g., more than one user utilising an instant messenger application on a shared PC.

An example N:1 relationship between peer and abstract layers is where more than one peer controls a resource, e.g., a hard disc that is shared between two peers.

An example N:M relationship between peer and abstract layers is where a resource is controlled by more than one peer and peers control more than one resource, e.g. peers might share a hard disc and a printer.

There has been considerable research within the area of presence (or awareness), examining issues such as synchronous and asynchronous presence, tightly and loosely coupled presence, etc [12]. However these are beyond the scope of the work presented in this paper and so have not been considered here.

### 3. Providing presence at the peer and abstract layer

When considering presence within P2P systems, there are two key aspects of functionality that need to be taken into account. Firstly, the deployment of the presence information throughout the system and

secondly, reacting to any change to this information. To an extent, both of these aspects will be affected by the choice of underlying logical architecture used.

The two areas of functionality also possess overlap. In particular the re-deploying of presence information is likely to happen whenever a change to the information occurs. Due to this fact, this paper focuses on the 'reacting to presence change' issues that need to be considered within a P2P system, and in doing so will consider presence information deployment issues as well.

Reacting to changes in presence information can be further broken down into the following issues within the areas of: identifying when an information change can occur and then as a result of this, ensuring that interested parties have up to date information.

### When presence information may change

**Connection:** When the entity providing the presence information is connected to a P2P network. *For example, a peer's state may change from off-line to on-line when the application is started*

**Disconnection:** When the entity providing the presence information is intentionally disconnected from a P2P network. *For example, an application is terminated and so the peer's state changes to off-line.*

**Failure:** When the entity providing the presence information is accidentally disconnected from a P2P network. *For example, if there is a power cut and the peer is accidentally disconnected from the network.* In this case it is likely that the peer would not have informed the rest of the network about its change in state. This means that it may be necessary to also provide an alternative mechanism for keeping interested parties up to date with the latest presence information.

**Presence State Change:** When the entity providing the presence information changes the information that it publishes. *For example, if a user changes their state from busy to free.*

### When it may be necessary to ensure that interested parties are kept up to date

**Awareness:** When an entity connects to a P2P network it needs to be informed about the current presence information state of the network. *For example, when an instant messenger client is started, it is informed of which other (relevant) users are on-line.* Essentially this represents an entity being given the latest presence information when it connects to the P2P network.

**Awareness Update:** When the presence information being published by an entity changes. *For example, if a user changes their state from busy to free.* In this case interested parties need to be informed of this change. This can either be achieved by informing them of the change, or by republishing all the presence information.

The above lists indicate the main issues that should be considered to incorporate presence within a P2P system. The paper will now move on to examine how the choice of logical network architecture can have an impact on the provision of presence functionality (and on tackling the above issues) within a system.

## 4. Summary of P2P logical network architectures

Before analysing how the choice of P2P logical network architecture can have an affect on the provision of presence, a brief summary of the more commonly used logical network architectures is provided (see figure 2). A more detailed review and analysis of these different architectures has been presented in our previous work [8][9].

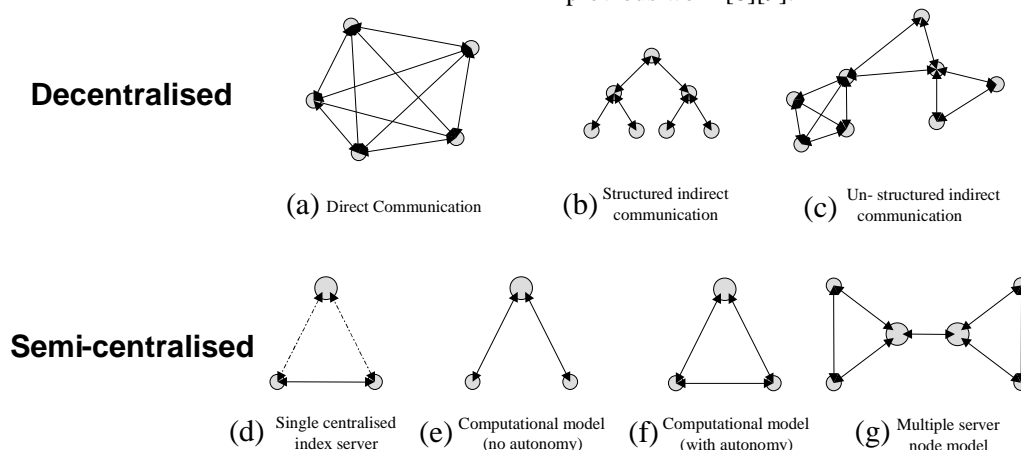


Figure 2 - P2P Architectures

## Decentralised Architectures

*Direct communication* – All nodes within the network are regarded as being equal and autonomous. No single node maintains any control over the network. Each node can communicate directly with each other. Each node is aware of each other. As a result of these characteristics, scalability is likely to be an issue.

*Structured indirect communication* - All nodes within the network are regarded as being equal and autonomous. No single node maintains any control over the network. It is not necessarily the case that all nodes can communicate directly with one another. Communication could be routed via other nodes. Nodes are connected together in a structured manner (for example, hierarchical, star, ring, etc). A degree of management may be required to ensure the structure persists.

*Unstructured indirect communication* - All nodes within the network are regarded as being equal and autonomous. No single node maintains any control over the network. It is not necessarily the case that all nodes can communicate directly with one another. Communication could be routed via other nodes. No structure is forced onto the architecture and so it can expand in an unpredictable manner. The discovery service becomes particularly important

## Semi-centralised Architectures

*Single centralised index server* - A single node acts as a lookup for all other nodes within the network. All other nodes are regarded as being equal and autonomous. All nodes can communicate directly with each other, but the index node typically facilitates this. These index nodes are a single point of failure for the architecture.

*Computational model (no autonomy)* - A single node acts as a focal point for all other nodes within the network. The remaining nodes do not possess their own autonomy. All communication is via the server node, if at all. Arguably not a true P2P architecture. The server node is a single point of failure for the architecture.

*Computational model (with autonomy)* - A single node acts as a focal point for all other nodes within the network. The remaining nodes retain a degree of autonomy. Nodes could communicate directly with one another (typically facilitated by the server node). The server node is a single point of failure for the architecture

*Multiple servers' model* - It is not necessarily the case that only one server node can exist within the network. This allows for the possibility of hybrid architectures. For example, server nodes connect together via a direct communication architecture, but collectively act as a single server node within a semi-centralised architecture.

The rest of this paper provides an analysis of how the choice of logical network architecture can affect presence support within a system. It first examines general presence design issues, before focusing on how the technical issues identified in section 3 can be affected by the choice of architecture.

## 5. General Presence Design Issues

As well as the technical issues, there are also more general design issues that may need to be considered by designers should presence support be desired. This section examines some of the key ones, including:

- *Controlling presence information*
- *Presence and privacy*
- *Presence consistency*
- *Presence as a mechanism to increase the dependability of a system*

### 5.1 Controlling presence information

Ultimately the presence information that is made available to the rest of the network is controlled by the entity that provides it (not taking into account security issues, such as Trojans). Consequently information control mechanisms need to be considered and provided.

One possibility is to provide the entity with a range of information levels, ranging from the entity providing very little information to providing a substantial amount. Such an approach can be frequently seen in many distributed applications, where a user can specify as much presence information as they feel (for example, ICQ). In theory this can be further enhanced by also specifying who/what can have access to this information. For example, all users have access to some portion of the presence information, but only my friends have access to all of it.

Anonymity can also be important, as it allows for an entity to provide presence information without actually revealing whom it is originating from. For example, an entity can provide 50 MB hard disc space to the system, but it's IP address remains hidden.

Controlling presence information, designing control mechanisms with which to do this, as well as considering the issues that may be involved, are important factors that need to be taken into account when incorporating presence within a system.

### 5.2 Presence and privacy

To a certain extent privacy is related to how the presence information is controlled. If the control mechanisms are sufficient then privacy should be less of a problem. However, there is always the possibility that the presence information that is published may be misused, for example, obtaining a users email address

and then using it to send Spam, or a peer's IP address and then performing an attack on that machine.

Again this may be limited by controlling what information is being made available (i.e. not displaying email addresses). However, in some situations it might be necessary for such information to be made accessible in order for the P2P system to fully function. For example, in Napster IP addresses need to be shared in order for a peer to download files from another peer.

To reduce the possibility of information misuse in such circumstances it is likely that suitable protection mechanisms would need to be provided. This could include drawing upon techniques such as authentication or reputation tracking [13]. In this way an entity could be sure that only those who have been granted permission (and are trusted) can have access to the information. However, the choice of logical network architecture used within the system can also affect the success of these techniques. In particular, semi-centralised architectures are likely to provide better support for their use [8][9].

If private information is to be published as presence information within a system, then it will become important to consider privacy issues, and to reduce the possibility of the information being misused.

### 5.3 Presence consistency

One issue that can be important within a system is that of insuring the presence information is up to date and valid. Depending on the nature of the system, inaccurate presence information could result in critical situations, such as incorrect business decisions being made, or just time and effort being wasted.

In order to ensure a high level of validity any updates to the published information need to be done in a near instantaneous fashion. In reality this is not going to be achieved, however the type of logical network architecture used can play a significant role in going some way to achieving it. In particular, semi-centralised architectures are likely to be most suitable for distributing quick presence updates around the network, whereas with the more decentralised architectures there is the danger of updates getting lost or delayed.

Ultimately, though, it would need to be decided whether presence mechanisms are reliable enough.

### 5.4 Presence as a mechanism to increase the dependability of a system

In our previous work [8][9], we identified a number of dependability properties that can be possessed by P2P systems. It was pointed out that many of these properties would typically require some form of monitoring mechanisms in order for them to be properly resolved. Such properties included availability, fault tolerance and maintainability.

Incorporating presence within a system might provide one mechanism with which this monitoring can be achieved. Presence mechanisms could be piggy backed and used to monitor the availability of nodes within the system, or to identify faults that may occur so that the system can act accordingly. However, in order to make this effective it is likely that a semi-centralised logical network architecture would need to be used.

Although presence could be utilised in this fashion, ultimately it needs to be decided whether presence mechanisms themselves, can be considered to be reliable, and if not, whether it is possible to make them reliable enough.

## 6. Establishing presence support in a P2P architecture

Section 4 provided a brief summary of the more common underlying logical network architectures that are used in P2P systems. For our analysis we have placed these into three categories: direct communication decentralised systems, indirect communication decentralised systems, and semi-centralised systems. This section moves on to examine how the presence issues that were identified in section 3 could be satisfied within each architecture category.

### 6.1 Providing presence within direct communication decentralised systems

Of all the different types of architecture, those that utilise direct communication between nodes are likely to always provide the best basis for supporting presence. Because each peer knows every other peer on the network it can become easier for presence information to be distributed to all peers immediately. Within such an architecture a central co-ordinator would not be needed to organise such matters. Obviously, as has been discussed elsewhere [8][9], this type of architecture suffers in terms of scalability given that a peer would have to broadcast its presence information to every other peer on the network. However if used in small-scale environments then it is the most ideal.

Table 1, suggests possible solutions for satisfying the presence issues within these types of P2P logical architectures.

Issue	Possible solutions for direct communication decentralised architectures
<b>Connection</b>	You would expect the entity to broadcast its presence information to all peers (and thus to all entities also on the abstract level), when it connects to the network.

<b>Disconnection</b>	You would expect the entity to broadcast the fact that it is disconnecting to all peers (and thus to all entities also on the abstract level).
<b>Failure</b>	In this case it is unlikely that the entity would have informed the rest of the network about its loss of connection. This means that unless peers are expected to regularly broadcast their presence information (and that of any abstract entities they might possess), or periodically poll other peers for theirs, they may be unaware that an entity has been disconnected
<b>Presence State Change</b>	You would expect the entity to broadcast any changes to its presence information to all peers (and thus to all entities also on the abstract level).
<b>Awareness</b>	Because all peers are connected to one another, when a peer (or any abstract entities it might possess) connects to the network, it should be able to automatically discover what other presence information exists on the network.
<b>Awareness Update</b>	Because all peers are connected to one another, when a peer's (or any abstract entities it might possess) presence information changes, it should automatically inform all other peers (and abstract entities) on the network of this fact. This would not happen, however, if the change were accidental (e.g., loss of power). To take into account this scenario it might be necessary for the individual entities to ping each other at regular intervals.

**Table 1 – Providing presence within direct communication decentralised architectures**

## 6.2 Providing presence within indirect communication decentralised systems

Achieving presence within indirect communication decentralised systems can be difficult (if not impossible) due to the lack of a central co-ordinator. This not only makes it difficult to co-ordinate the collection and publishing of presence information, but also to give peers, users and resources universal ID's within the system. Without these ID's it is likely to be a difficult task to identify which entity's presence information has changed. In addition, because the network can frequently change it may be difficult to achieve any form

of real-time presence information updates or even for these updates to be received. In theory, when presence information is broadcast onto the network it could be spread between peers using techniques similar to that used for searching or for peer discovery in indirect decentralised architectures. However, due to the dynamic nature of the architecture it cannot be guaranteed that all peers will receive this information, or that a peer that has received it once in the past, will receive it again (due to issues such as network partitioning, alternative message routing, etc).

As a result, it is difficult to analyse the effects these types of architecture can have on the presence issues that have been identified. Table 2, however, suggests possible solutions for satisfying the presence issues within these types of P2P logical architectures.

<b>Issue</b>	<b>Possible solutions for indirect communication decentralised architectures</b>
<b>Connection</b>	When an entity connects to the P2P network it will be able to publish its presence information but only to those other peers (and to their respective abstract entities) it is aware of. In theory, in turn these peers could then broadcast the information to the peers they are aware of.
<b>Disconnection</b>	When an entity is going to disconnect from the P2P network it will be able to publish this fact but only to those other peers (and to their respective abstract entities) it is aware of. In theory, in turn these peers could then broadcast the information to the peers they are aware of.
<b>Failure</b>	In this case it is unlikely that the entity would have been able to inform the other entities it is aware of, about its loss of connection. This means that unless peers are expected to regularly broadcast their presence information (and that of any abstract entities they might possess), or periodically poll other known peers for theirs, they may be unaware that an entity has been disconnected.
<b>Presence State Change</b>	When an entity's presence information changes, it will be able to publish this fact but only to those other peers (and to their respective abstract entities) it is aware of. In theory, in turn these peers could then

	broadcast the information to the peers they are aware of.
<b>Awareness</b>	<p>When an entity connects to the network it will most likely need to perform some sort of presence information discovery. This could operate in a similar manner to resource searching or the discovery of peers within this type of architecture. The entity could issue a presence request that would get passed to all the entities it is aware of. These would return their own presence information, whilst also forwarding the presence request to the entities they are aware of. The issues with such an approach are that it is not overly reliable, returned presence information may not be up to date, the time it takes to gather this information could vary considerably, and it cannot be guaranteed that all entities on the network would receive the request.</p> <p>It is likely that the most reliable presence information would be obtained from the local entities.</p>
<b>Awareness Update</b>	<p>It could be difficult to keep an entity's presence information up to date with interested parties. Because it cannot be assumed that the entity would be able to contact these parties directly to inform them of a change in the presence information, it is likely that a propagating broadcast method would have to be used.</p> <p>As highlighted previously, this does not provide any guarantees of reliability, and could potentially result in an interested entity not being informed of a change.</p> <p>An alternative strategy would be to make each interested entity attempt to obtain the current presence information itself. However, this could also add to the problems by swamping the network with traffic.</p>

**Table 2 – Providing presence within indirect communication decentralised architectures**

One possible solution for supporting presence within indirect communication architectures would be to create smaller groups of entities. These groups could then communicate with each other in a direct way, but also be linked to other groups. However, to achieve this it is

likely that a degree of management would be needed and ultimately this could result in upsetting the equality of the network (in essence turning it more into a semi-centralised system).

### 6.3 Providing presence in semi-centralised systems

It is likely to be easier to achieve presence within semi-centralised based systems than with some of the more decentralised alternatives (in particular with indirect communication decentralised systems) due to the central foci that exist within the system. These foci can be used to capture and publish the presence information that exists, and because all peers within the system will be in direct contact with them, this information should be reasonably consistent and up to date. Obviously the negative side is that these foci become points of failure. Should they go down then this presence information will be lost from the network.

Table 3, suggests possible solutions for satisfying the presence issues within these types of P2P logical architectures.

<b>Issue</b>	<b>Possible solutions for semi-centralised architectures</b>
<b>Connection</b>	You would expect the entity to inform a server node of its presence information when it connects to the network.
<b>Disconnection</b>	You would expect the entity to inform a server node of the fact that it is disconnecting from the network.
<b>Failure</b>	In this case it is unlikely that the entity would have informed a server node about its loss of connection. This suggests that as well as presence information updates coming from the relevant entities, the server node(s) may also need to make regular checks on the presence state of the network.
<b>Presence Stage Change</b>	You would expect the entity to inform a server node of any changes to its presence information.
<b>Awareness</b>	The server nodes would most likely be used to store and distribute presence information around the network. As a result, when an entity connects to the network to obtain current and relevant presence information, it would likely need to send a list of the entities it is interested in, to one of these server

	nodes. The server node could then return the current presence information for these entities.
<b>Awareness Update</b>	<p>Because the server nodes would most likely be used to store and distribute presence information around the network, they need to be kept informed of what entities are interested in (i.e. who is interested in who).</p> <p>To achieve this each entity would need to register these interests with a server node. In this way, when a change occurs the relevant interested parties can be kept up to date. The main danger with this approach is that amount of information the server may end up having to store, especially if a poor structure is not adopted.</p> <p>An alternative strategy would be to make each entity poll a server node for the current presence information. This, however, could result in swamping the network and server node.</p>

**Table 3 - Providing presence within semi-centralised architectures**

Within this architecture an entity could try and obtain current presence information itself by contacting the relevant entities directly. However, this would rely on using previous information about the entities that may have become out of date (peer address, for example). It is, however, a viable alternative to solely relying on the server nodes.

## 7. Summary

This paper has examined how presence can be supported within P2P systems. It has provided an initial overview of presence within P2P environments and has attempted to identify its main characteristics. It has also identified the key technical requirements that would need to be satisfied, and has analysed how the different logical network architectures can have an effect on these.

Initial analysis has shown that overall (when scalability is taken into account) semi-centralised architectures provide the best foundation for supporting presence. The server nodes within a semi-centralised system can be used to capture and distribute presence information around the network, and because all nodes connect to the server nodes there are no problems with

regards to providing entities with ID's or not being able to contact all parts of the network.

Although, generally, decentralised architectures are arguably less suitable for supporting presence, this is not the case for direct communication decentralised systems. Because all nodes connect to all other nodes, this architecture can also be a viable alternative for supporting presence. Obviously the downside (as reported elsewhere [8][9]) is that this architecture is not really scalable.

As well as the analysis, this paper has also highlighted general design issues that may need to be considered if presence support is desired. These have focused on controlling the presence information, ensuring privacy (if desired) and the importance of keeping presence information up to date. The possibility has also been raised of whether presence could be used as a mechanism to improve a P2P systems dependability.

## 8. References

- [1] Andy Oram (editor), *Peer-to-Peer: Harnessing the power of Disruptive Technologies*, pages 85-86, O'Reilly publishing, 2001
- [2] Palen, L., Social, individual, and technological issues for groupware calendar systems. *In CHI'99*, 1999
- [3] Herbsleb, J. D., Grinter, R. E., Architectures, coordination, and distance: Conway's law and beyond. *IEEE Software*, Sept/Oct 1999, 63-70.
- [4] Godefroid, P., Model Checking for Programming Languages using VeriSoft. *In ACM Symposium on Principles of Programming Languages*, January 1997, 174-186.
- [5] ICQ. Instant Messenger Application. More information at the URL <http://www.icq.com>
- [6] MSN Messenger. Instant Messenger Application. More information at the URL <http://specials.msn.com/ms/default.asp>
- [7] Napster. MP3 file sharing application. More information at the URL <http://www.napster.com>
- [8] Walkerdine, J., Melville, L., Sommerville, I., Dependability Properties of P2P Architectures, *In P2P2002*, 2002
- [9] Melville, L., Walkerdine, J., Sommerville, I., Report on the dependability properties of P2P architectures, Information Societies Technology Institute, 31st July, 2002
- [10] Gnutella. File sharing application. More information at the URL <http://www.gnutella.com>
- [11] Want, R., Hopper, A., Falcao, V., Gibbons, J., The active badge location system. *ACM Transactions on Information Systems*, 10(1):91-102, 1992.
- [12] Ramloll, R., "Supporting Cooperative Work Through Ubiquitous Awareness Filtration Mechanisms," PhD Thesis, Lancaster University, 2000
- [13] Andy Oram (editor), *Peer-to-Peer: Harnessing the power of Disruptive Technologies*, Chapter Seventeen, O'Reilly publishing, 2001