# Dependability within Peer-to-Peer Systems

James Walkerdine, Lee Melville, Ian Sommerville
*Computing Department, Lancaster University, Lancaster, LA1 4YR, UK*
*{walkerdi, l.Melville, is}@comp.lancs.ac.uk*

## Abstract

*There is an increasing interest in using Peer-to-Peer (P2P) architectures as a basis for software systems. However, by their very nature, achieving dependability within a P2P system can be difficult. This paper provides an initial analysis of the main issues that need to be considered when developing a dependable P2P system. It examines the key properties that can influence the dependability of a P2P system, and discusses the relationship between system dependability and the choice of logical network architecture.*

### 1 Dependability and Peer-to-Peer

A system's dependability can be thought of as being its trustworthiness [9]. The difficulty when attempting to measure dependability is that it is typically a context sensitive property. While one user might regard a system to be dependable for the particular activities they use it for, another user might regard it to be undependable for their activities. Traditionally dependability has also been regarded as multi-dimensional, in that it can be influenced by a variety of other attributes. Key attributes include availability, reliability, responsiveness, safety and security [9].

Peer-to-Peer (P2P) computing has become very popular in recent years. Essentially it can be thought of as a class of application that takes advantage of the resources and services that are available at the edge of the Internet [8]. There is an increasing interest in using P2P as a basis for software systems within industry, where it can be used to support activities such as communication between workers. However for P2P technology to be adopted within such an environment it also needs to be dependable and achieving dependability within a P2P system can be difficult.

In particular P2P systems possess a number of specific properties that can have an influence on the system's dependability attributes (security, reliability, etc). For example, the type of peer discovery mechanism used can influence the responsiveness of a P2P system. A broadcast discovery mechanism can result in slower performance than with using a centralised peer lookup server. Consequently, when considering dependability within P2P system design, it is also becomes necessary to consider these specific P2P properties.

Dependability within P2P systems is further complicated by the numerous P2P logical network architectures (abstractions of the underlying physical network) that exist and no single architecture is likely to be suitable for all application types. For example, Napster [7] benefits most from a semi-centralised architecture (provides more efficient resource searching), whereas a decentralised architecture is more suitable for FreeNet (better anonymity support) [2]. The different types of logical network architecture can also influence the P2P properties and general system dependability. Decentralised P2P systems are likely to be better suited at handling denial of service attacks, the central authority provided by semi-centralised systems would be better suited for handling peer certification. Designers, when deciding on a suitable logical network architecture, also need to take into account the dependability requirements of the system. In some cases this will restrict the logical network architecture options that are available.

This paper describes some of our current work within the EU funded P2P ARCHITECT project, which seeks to develop methods and tools to support the building of dependable P2P software systems. The paper aims to identify the main properties (additional to availability, reliability, responsiveness, safety and security) that can influence a P2P systems dependability, and how these in turn can be influenced by or influence the choice of underlying logical network architecture.

The paper begins by presenting the key P2P dependability properties that we have derived from studying existing dependability properties and applying and analysing their importance within a P2P environment. The paper then summarises the common types of P2P logical network architectures, and provides an initial discussion of the relationship between the properties and architecture.

### 2 Dependability Properties of P2P Systems

To structure and help readability the key dependability properties that we have identified have been placed into three categories.

*External properties* – properties that can be only viewed externally (for example, by the user).

*Internal properties* – properties that can be viewed from within the system (for example, by a system component).

*Hybrid properties* - properties that can be viewed both internally and externally.

A brief summary of the properties is provided here. A more detailed description is provided in [6].

### 2.1 External properties

*Scalability* - Scalability is the ability of a system to operate without a noticeable drop in performance despite increases or decreases in its overall operational size. Peer-to-peer systems, by their very nature, are designed to be distributed over many peers. Accordingly, catering for scalability should play a fundamental role when designing for a dependable peer-to-peer system, and can have influences on dependability attributes such as reliability, availability and responsiveness.

*Survivability* - Survivability is the capability of a system to fulfil its mission in a timely manner in the presence of attacks, failures, or accidents [1]. Survivability in P2P systems raises some interesting issues, as in some cases the inherent redundancy can help attain survivability, whilst the lack of a central control can hinder it. Furthermore a system's survivability can also have an influence on its dependability attributes such as reliability and availability.

*Maintainability* - Maintainability represents the ease in which the system can be changed after it has been delivered and is in use [9]. A major issue with P2P maintenance is updating peer software. It cannot be assumed that all peers will be updated and this can be particularly important for critical issues (for example security patches). Given the range of issues that maintainability can affect and given the distributed nature of P2P, it should be viewed as being a critical property when designing a dependable P2P system.

*Manageability* - Manageability reflects the ease in which the system as a whole can be managed (which can be important in business environments). Although not an obvious dependability property, a system's manageability can influence other issues such as security and maintainability. Its importance is also further raised given that it is often harder to control all aspects of a P2P system.

*Repairability* - A system is regarded as being repairable if it allows defect correction with minimal effort. A repairable system needs to be able to detect the faults and then perform corrective maintenance. As P2P systems are distributed in nature the main difficultly is how to identify and repair defects that can occur throughout the network. This is especially the case for decentralised systems and so can make repairability an important property for when designing dependable P2P systems.

*Trust* - Trust can depend on a range of properties. For example, it can be influenced by the perceived dependability of the system, its security measures, or the behaviour of other users. Actually defining and measuring trust is a difficult task due to its subjective nature. As a result in terms of dependability it is a complex property to consider. However there is a link between trust and dependability, and within distributed systems trust (of users and machines) can be a crucial issue.

### 2.2 Internal properties

*Network evolution* - Studies of existing P2P systems have shown that the logical network architecture used can evolve over time [5]. Although P2P systems set out to give all their peers equal status, in reality this is very difficult to achieve due to the different resources available to each peer. Because P2P systems are likely to evolve during use, a dependable system would need to be able to cater for this eventuality. The evolution of the system could have implications for dependability attributes such as security and maintainability.

*Legacy versions* - It is likely that as new versions of the P2P software are released, not all peers within the network will upgrade. Consequently you can have the scenario where multiple versions of the software are run across the network. A P2P system needs to be able to still operate despite the different versions of the software that might be running on the peers. For designing a P2P system that is upgradeable whilst also maintaining system dependability, legacy is going to be an issue that needs to be tackled.

*Fault Tolerance* - Fault tolerance is the ability for a system to continue giving a correct service following the manifestation of a fault either through errors in the system design, implementation or introduced following an attack [9]. Fault tolerance is a particularly important issue within P2P given its distributed nature. Designers would need to decided how faults will be recognised and dealt with in a distributed manner. Clearly the ability of a system to resist and tackle faults will have an influence on its perceived dependability.

*Connection bandwidth* - How peers are connected together in a P2P network can vary considerably, from a user connecting via a modem, to a machine connected via a T3 connection. Consequently the amount of network bandwidth available to a single peer can vary considerably. Ideally a dependable P2P system should be able to operate no matter the connection bandwidth, and should be designed in such away to avoid hindering

system attributes such as reliability, availability and responsiveness.

*Intermittent peer connectivity* - Due to the very nature of peer-to-peer, it cannot be assumed that peers within such a network are connected at all times. When designing a peer-to-peer system, it is important to cater for a peer's intermittent connectivity and not assume that a peer will always be connected. Likewise it is important to make sure that intermittent connectivity will not affect other system dependability attributes (for example, reliability, availability and responsiveness).

*Peer Discovery* - P2P systems need the ability to discover other peers that reside on the network. Typically this is either achieved by using a central lookup service, or by propagating discovery messages around the network. The centralised approach tends to be more efficient but less fault tolerant, and the propagation approach is prone to poor responsiveness. Consequently, the dependability of P2P system is going to be influenced to an extent by the method of peer discovery that is used.

*Peer addressing* - Due to the rapid uptake of the Internet it is no longer feasible to guarantee every host with a fixed IP address. Dynamic IP's are seen as a possible solution; however within P2P systems their use can result in peers becoming difficult to reach due to their high dynamic IP address turnaround. P2P's dynamic nature, and the lack of permanent IP's presents a challenge that designers need to take into account. Failure to properly tackle peer addressing can have an influence on a number of dependability attributes (for example, reliability, availability and responsiveness).

*Load balancing* - Load balancing is where the load that is placed on components within a system is balanced to ensure that a component is not overworked or, alternatively, underused. Within P2P systems load balancing can be important when deciding how much of a peers' resources to draw upon. The lack of load balancing can have significant effect on the systems dependability. For example, attributes such as availability and responsiveness can all be affected due to load imbalance resulting in certain peers being hard to reach or simply failing due to overload.

### 2.3 Hybrid Properties

*Responsibility, accountability and reputation* - A key challenge within P2P systems is enforcing rules of social responsibility. To minimise breakdowns in this (for example, spreading Spam) systems need to provide means to track occurrences to the originator and also provide mechanisms to make users accountable for their actions. Responsibility and accountability are also linked to trust, and can be applied from the hardware level through to the user. Although difficult to quantify they are, nevertheless, properties that can influence the users perception of the dependability of a system.

*Data integrity* - It is important for the data that is stored and manipulated by a system to maintain its integrity. P2P systems provide both advantages and disadvantages for maintaining data integrity. On the one hand their decentralised nature increases the chances of data becoming corrupted or invalid as it passed around, on the other hand it also provides redundancy. Given the importance of data integrity within a distributed system, it is a property that must be considered when designing a dependable system.

*Adaptability* - Adaptability is a systems ability to adapt to a dynamically changing environment. P2P networks are very unpredictable in nature; a P2P system needs to be able to adapt when changes occur to ensure its continued operation. Given the very dynamic nature of most P2P systems, a dependable P2P system should be able to easily adapt to change in order to ensure that attributes such as reliability and responsiveness are not compromised.

## 3 Dependability Properties and Logical Network Architectures

Although the above properties are all issues that designers should consider when developing a dependable P2P system, they in turn are also influenced by the type of logical network architecture that is used as the basis for that system. For example, a semi-centralised styled logical network architecture will be better suited for use in a safety critical system due to the fact that a central peer is able to monitor the whole system. Consequently in order to meet the dependability requirements for a system, designers will also need to consider which logical network architecture can best assist them in achieving this task.

### 3.1 Overview of P2P Logical Network Architectures

Peer-to-peer systems are built up around a collection of nodes that are networked together in some fashion. These nodes are typically personal computers but there are no reasons why they cannot be anything with a 'digital heartbeat', be it PDA's, sensors, consumer electronics, network routers or storage systems. Communication that passes between these nodes can, for example, involve the transference of data or the relaying of commands from a server node to a client node.

Logical network architectures (or overlay) represent an abstraction of the physical network architecture (the physical network connections), and consider just the nodes and connections between them.

From examining existing peer-to-peer systems it is apparent that two core types of logical network architecture exist. *Pure P2P* or *Decentralised*, where

each node within the network is regarded as an equal and no control nodes exists, and *Hybrid* or *Semi-centralised*, where there exists at least one control node that performs an authoritative role within the network. Within this paper we use the terms decentralised and semi-centralised for the two architecture types.

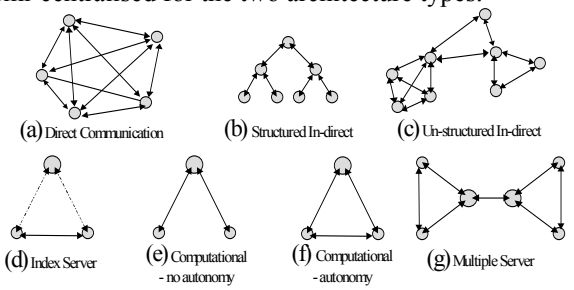


**Figure 1 - P2P Logical Network Architectures**

Figure 1 illustrates seven of the more commonly used peer-to-peer logical network architectures. These have been described in detail in our previous work [6], but a summary is provided here.

*Direct communication architectures (a)* represent those in which all peers can communicate directly with each other and hence are also directly aware of each other. The main disadvantage of this architecture is its lack of scalability as it becomes unfeasible for each peer to 'know' every other peer. *Structured Indirect communication architectures (b)* differ in that it is not necessarily the case that all peers can communicate directly with one another. Instead, peers that are not linked can communicate via other nodes. These types of architectures are also structured so that they conform to a type of network topology (for example, hierarchical, ring, and star). *Unstructured Indirect Communication architectures (c)* are essentially the same as their structured counterparts with the difference being their more freeform nature. Peers can be connected and removed from any part of the network, resulting in a varied density of peers throughout.

*Single Centralised Index Server architectures (d)* possess a single peer that can act as a lookup for all other peers within the system. Peers communicate directly with each other, but with the aid of this central index peer. Although having an index peer does provide certain benefits, it also becomes a single point of failure. The two computational architectures are similar except that the server peer instead co-ordinates the computation that is carried out within the system. With *Computational without Autonomy architectures (e)*, the remaining peers of the network do not possess their own autonomy and are essentially controlled by the server peer. With *Computational with Autonomy architectures (f)*, the remaining peers maintain a degree of autonomy (allowing them, for example to communicate with each other). Finally, *Multiple Server Node architectures (g)* represent the possibility of architectures that possess more than one server peer. This can provide advantages such as increasing a P2P systems reliability by removing a potential single point of failure, or improving Quality-of-Service should a single server become overused.

### 3.2 The influence the architecture can have on a system's dependability

To illustrate the relationship between the dependability properties and the logical network architectures, we provide a brief analysis involving two of the above-discussed architectures. Due to the subjective nature of a system's perceived dependability and because it will also be significantly influenced by the actual application or framework that is being used, only a theoretical analysis is provided at this time. A more detailed analysis that considers all the architectures is provided in [6].

#### *3.2.1 Using an Unstructured Indirect Communication architecture*

The main characteristics of this type of architecture are the lack of server nodes and its free form nature. This means that the individual peers that make up the systems need to be independent and self-contained (possess the functionality to operate independently). In terms of providing a basis for a dependable system, this provides both advantages and disadvantages.

The lack of server peers removes the problem of the system possessing single points of failure. Should a peer fail, the system should still continue to operate without too significant a performance loss. Such a system is more resilient to attacks and faults, thus potentially making it better for tackling aspects of dependability such as *survivability* and *fault tolerance* as well as broader reliability, availability and security issues.

However, the lack of central points of control or focus makes it difficult to monitor or control the system as whole. This makes it less suitable for safety critical systems where system monitoring is critical, as well for supporting *management* and *maintenance*. Furthermore, although this architecture may help a system survive a fault or attack, because there is no central control when such a fault/attack occurs there is nothing to monitor and co-ordinate the recovery of the system.

The freeform nature of this architecture type means that it can easily *adapt* to changes within the network. If a routing peer is disconnected it should be possible to route a message via another set of peers. Likewise the architecture allows for *network evolution* so that issues such as overload on peers, or inefficient use of connection bandwidth can be addressed. Gnutella [3], which uses such an architecture structure, is an example of where network evolution has been seen to occur.

The main disadvantage of the architecture's flexible nature, however, is that of *scalability* and system responsiveness. The routing of messages between two nodes from either side of the network can make the responsiveness of the system very slow, if not impossible (due message lifetimes, etc).

*3.2.2 Using a Single Centralised Index Server architecture*

This type of architecture uses a server peer that acts as a lookup for the rest of the network. This does mean, however, that the 'client' peers typically rely on the server peer in order for the system to properly operate.

Using a server peer provides a number of advantages, in particular the fact that it can be used to manage or control aspects of the system. This makes it better suited for systems where *safety*, *maintainability* or *manageability* are important. It also means that the system as whole can better respond to attacks or faults. Should such occur, the server peer could monitor this and initiate suitable recovery procedures.

A server peer can also help in supporting *trust* within a system, making it relatively easy to implement *accountability* techniques. Using a server peer as a lookup can also help system responsiveness particularly with *peer discovery*. Rather than having to broadcast a peer discovery message through the network, a peer can directly lookup the target peer's address on the server.

The drawback of an architecture that uses a server peer, is that it acts as a single point of failure. Due to the rest of the networks reliance on the server, should it be compromised then the dependability of the whole system will be affected (in many cases it may mean the system is unusable, a good example being of when the servers go down with instant messenger applications such as ICQ [4]). This single point of failure therefore has a notable influence on the systems *fault tolerance* and *survivability*, as well as on general system reliability, availability and security issues.

## 4 Summary and Conclusions

This paper has examined key issues that should be considered when developing a dependable P2P system.

Dependability is a difficult attribute to measure. Not only is it context sensitive and subjective, it is also influenced by a number of dependability attributes. Achieving dependability within P2P systems is further complicated by their distributed nature. This results in the need to consider further properties, such as intermittent peer connectivity and bandwidth, which can also have an influence on system dependability. Finally, the relationship that exists between system dependability and the logical network architecture types, also means that the choice of architecture used can have an impact on a system's dependability (and vice versa).

This paper has summarised key properties that can influence a P2P system's dependability. These properties highlight additional issues that designers should also consider when developing a dependable P2P system. The paper has also provided a brief overview of the more commonly used logical network architectures, and illustrated, with initial theoretical analysis, how the choice of architecture can help or hinder some of these properties. This demonstrates the importance of the logical network architecture within the design process, and how that the designers should choose the architecture based on the dependability requirements of the system. Consequently when developing a dependable P2P system the required properties and how they relate to possible logical network architectures should be considered at an early stage.

It is our intention to extend the initial analysis work that is presented here by assessing specific implementations (for example, Napster) in more detail. Although the results would be specific to the individual implementation, they would go someway to quantifying the initial analysis provided here and the effect the choice of logical network architecture can have on system dependability. The results of this work will be presented in a future paper.

## 5 Acknowledgements

This work has been funded by the European Commission within the P2P ARCHITECT project (IST-2001-32708).

## 6 References

[1] Ellison, R. J., Fisher, D. A., Linger, R. C., Lipson, H. F., Longtaff, T. A., Mead, N. R., SURVIVABILITY: Protecting Your Critical Systems, In IEEE Internet Computing, 3 (6), 55-63, Nov/Dec, 1999.

[2] The Free Network P2P project (FreeNet). Anonymous file sharing application. More information can be found at http://freenet.sourceforge.net/

[3] Gnutella. File sharing application. More information at can be found at http://www.gnutella.com

[4] ICQ. Instant Messenger Application. More information at the URL http://www.icq.com

[5] Kan, G., Gnutella, *in Peer-to-Peer: Harnessing the power of Disruptive Technologies*, Oram, A., (editor), O'Reilly publishing, 2001

[6]Melville, L., Walkerdine, J., Sommerville, I., Report on the dependability properties of P2P architectures, Information Societies Technology Institute, IST-2001-32708, 31st July, 2002

[7]Napster. MP3 file sharing application. More information can be found at http://www.napster.com

[8]Shirky, C., Listening to Napster, *in Peer-to-Peer: Harnessing the power of Disruptive Technologies*, Oram, A., (editor), O'Reilly publishing, 2001

[9]Sommerville, I., *Software Engineering*, 6th Edition, Addison Wesley publishing, 2000