A SURVEY OF PEER-TO-PEER ARCHITECTURES FOR SERVICE ORIENTED COMPUTING

Danny Hughes
Katholieke Universiteit Leuven, Belgium

Geof Coulson
University of Lancaster, UK

James Walkerdine
University of Lancaster, UK

Abstract

Peer-to-peer file sharing has become popular for many kinds of resource location and distribution applications including file sharing, distributed computation, multi-media messaging and content distribution. Peer-to-peer approaches also have significant potential for supporting large scale, decentralised service oriented computing. This chapter discusses each class of contemporary P2P architecture in turn and discusses the suitability of each architecture class for supporting service oriented computing. Future trends in peer-to-peer architectures are then discussed and multi-layer peer-to-peer architectures are highlighted as a promising platform for supporting service oriented computing. This chapter then concludes with a discussion of outstanding issues that must be addressed before peer-to-peer architectures can offer adequate support for service oriented computing.

INTRODUCTION

Service Oriented Computing (SOC) is a family of tools and approaches, which aim to reflect component trends towards anonymity and heterogeneity, providing systems which are better suited to today's large-scale and open distributed environments (Huhns & Singh, 2005). Individual services may be composed together by developers to provide emergent application functionality. Service-based applications, are composed of three parts: *providers*, who provide services, *consumers*, who use these services and a *registry*, which is used to publish information about available services. These architectures are supported by *service definition* languages such as WSDL (Christensen, Curbera, Meredish, & Weerawarana, 2001), *discovery protocols* such as UDDI (Bryan, Draluk, Ehnebuske, Glover, Hately, Husband, Karp, Kibakura, Kurt, Lancelle, Lee, MacRoibeaird, Manes, McKee, Munter, Nordan, Reeves, Rogers, Tomlinson, Tosun, von Riegen, & Yendluri, 2002) and *binding protocols* such as SOAP (Box, Ehnebuske, Kakivaya, Layman, Mendelsohn, Nielsen, Thatte, & Winer, 2000).

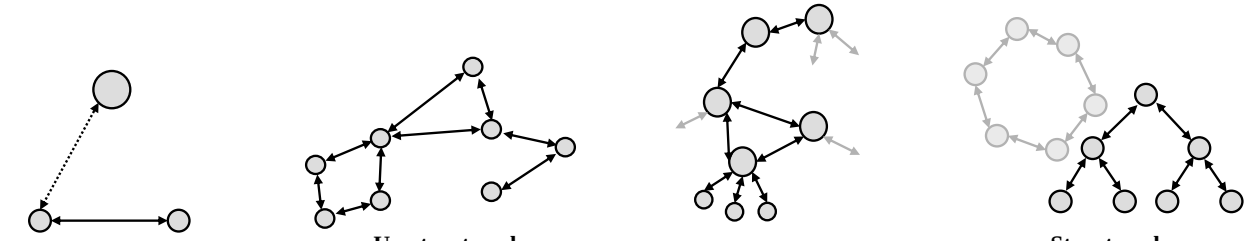
Peer-to-Peer (P2P) systems are a promising platform for supporting more flexible service oriented computing, offering decentralised and large-scale service discovery and publication. This can potentially allow any peer participating in the network to be a service provider or consumer. P2P systems are realised using a family of distributed programming approaches that emphasise co-operation between entities known as peers that are essentially equal and can both require and provide services (Shirkey, 2001). Thus far, P2P approaches have been successfully applied to fields including: resource sharing, distributed computation, multimedia messaging and content distribution.

A core component of a P2P system is its network architecture, which represents how the peers within the system are connected together. These can range from topologies that are particularly centralized in their operation (typically using some form of central hub), to fully decentralized topologies in which there is no set structure and all peers are considered equal. This chapter presents a classification of these P2P architectures and analyses their suitability for supporting the requirements of service oriented computing. In particular we look at their support for key aspects of SOC:

- Service Publication – the process of announcing your service to others
- Service Discovery – the process of discovering services that meet your requirements
- Scalability – how well the infrastructure can support a large scale environment with many service providers and consumers
- Resilience – the reliability of the infrastructure for the publication, discovery and provision of services

SURVEY OF P2P NETWORK ARCHITECTURES

As described in the introduction to this chapter, P2P systems are a promising architecture for supporting SOC and have a proven track record for supporting large scale resource location and distribution. This section introduces an in-depth survey of P2P systems conforming to the key classes of P2P architecture and discusses their suitability for supporting service oriented computing:



Semi-centralised

Example Systems
Napster, SETI@home

Un-structured decentralised

Example Systems
Gnutella (v0.4), FreeNet

Super-node decentralised

Example Systems
Gnutella (v0.6), Kazaa

Structured decentralised

Example Systems
Pastry, Chord

Figure 1 – Classes of P2P architecture

1. *Semi-centralised P2P systems* such as Napster (Merriden, 2001) and SETI@Home (Anderson, 2001) use a semi-centralised network architecture, wherein a small number of dedicated servers mediate interaction between large numbers of peers at the edge of the network.
2. *Unstructured, decentralised P2P systems* such as Gnutella 0.4 (*The Gutella Protocol Specification v0.4, Document Revision 1.2*, n.d.) and GPU (Mengotti, 2004) are entirely decentralised, connecting peers together in an essentially random manner and treating all peers equally.
3. *Super-node P2P systems* such as Gnutella 0.6 (*The Gnutella Protocol Specification v.0.6, draft*, n.d.) and Kazaa (*Kazaa Home Page*, 2001) feature decentralised server-less communication, however, all peers are not treated equally. More powerful peers on faster connections may elect to become 'super-nodes' which route a greater proportion of messages and may also provide special services.
4. *Structured, decentralised P2P systems* such as Chord (Stoica, Morris, Karger, Kaashoek, & Balakrishnan, 2001), CAN (Ratnasamy, Francis, Handley, Karp, & Shenker, 2001) and Pastry (Rowstron & Druschel, 2001) feature decentralised, server-less communication supported by a highly structured overlay network.

Specifically, each of these architectures will be critiqued on its support for *publication of service specifications* and *discovery of services*.

Semi-centralised P2P Systems

Semi-centralised P2P networks use dedicated servers to mediate the process of resource discovery and distribution, though some stages of this process may also occur via direct interaction between peers. Typically semi-centralised systems use servers to index available resources and peers. In this sense, semi-centralised systems such as Napster (Merriden, 2001) and SETI@Home (Anderson, 2001) are not really P2P, as they feature highly asymmetric communication. However, these systems can be considered P2P in the sense that they exploit unused resources available at the edge of the network and they empower participating peers to provide as well as consume services. As these systems feature elements of client-server *and* peer-to-peer design, semi-centralised P2P systems are often referred to as *hybrid* models.

Napster

Napster (Merriden, 2001) was the 'killer application' which led to the (re)emergence of the P2P paradigm. It was released in 1999 and rapidly captured the imagination of users and the research and business communities. Napster made use of the disk space and bandwidth of home-PCs to create a distributed library of digital music. Using the Napster client software, users were able to make their digital music collections available online and download music from other users. At its height, Napster served over 50 million users (Merriden, 2001). The Napster system is proprietary and protected by copyright, therefore it is not possible to provide a detailed description of the Napster protocol here, however a high-level overview of the resource location and distribution features of Napster is provided below.

Connection: Peers connect to the Napster network by firstly establishing a TCP connection to the central indexing server. This connection is authenticated by user-name and password and nodes which have behaved badly are blocked at this point. Each peer then uploads a list of all available files to the indexing server.

Resource Discovery: In order to discover resources, the downloading client will issue either a 'search' or 'browse' request to the server. A 'search' request takes a file name as argument and the server responds by returning a list of all matching files, together with connection information for the peers that are hosting them. A 'browse' request takes a peer identifier as argument and the server should respond by returning a list of all files contained on the specified host.

Resource Distribution: File transfer occurs directly between peers via Hypertext Transfer Protocol (HTTP) following the resource discovery phase. The searching peer attempts to make a HTTP connection to the peer holding the target resource on the IP address and port previously discovered and then proceeds to download the file. In cases where the serving peer is behind a fire-wall or Network Address Translation (NAT) device, the client peer may initiate a 'PUSH' request, wherein the server peer makes a TCP connection to the client and uploads the required file. If both client and server nodes are behind a firewall, then no file transfer is possible.

While at its height Napster was very successful, supporting a user community that numbered in the millions, the system was to prove critically vulnerable to scalability issues and attack, as will be discussed in the critique at the end of this section.

SETI@Home

SETI@Home (Anderson, 2001) uses the CPU cycles of home PCs to create a highly distributed super-computer dedicated to searching through information gathered by radio telescopes for signs of extra-terrestrial intelligence. Using the SETI@Home client software, users are able to donate their spare CPU cycles to a scientific experiment. SETI@Home has been a great success and at one point was the largest super-computer in the world (Anderson, 2001). Unlike the Napster model, peers do not directly interact with each other in SETI@Home. As no stage of interaction occurs directly between peers, SETI@Home can only be considered a P2P application in the context that is makes use of resources around the edge of the network.

Data Acquisition: The initial data set for SETI@Home is gathered at the Arecibo radio telescope array, which gathers radio signals from the vicinity of candidate stars, at specially selected frequencies (Anderson, 2001). The SETI@Home servers then divide this data into work units of suitable size for distribution over the Internet (300KB).

Processing: Peers connect to SETI@Home by firstly establishing a connection to the central server, which is authenticated by user-name and password. Each peer then downloads a work-unit via HTTP and proceeds to execute the algorithms contained in the client software on the work unit. When analysis of the work unit is completed, the result is uploaded back to the SETI@Home server, where it is logged.

Candidate Detection: Strong candidates that have been detected by SETI@Home peers are filtered by central servers in order to remove interference from human activities. Miscalculated or misreported work units are then removed by central servers and promising results are passed to astronomers for further analysis.

SETI@Home has been extremely successful at exploiting public interest in the search for extraterrestrial intelligence using P2P technology. The SETI@Home system remains one of the world's most powerful super-computers at over 20TFLOPS and the SETI@Home project is also the world's longest computation, having performed more than $4 \text{ X } 10^{20}$ floating point operations (Anderson, 2001).

Other Semi-Centralised P2P Systems

Alongside Napster and SETI@Home, other notable systems use a semi-centralised P2P approach. *OpenNap* (*The OpenNap Protocol Specification*, 2000), the open-source clone of Napster was released following Napster's closure in 2001. OpenNap allowed users to set-up their own Napster-style directory server, to which other OpenNap users could connect and share files. Unfortunately, while Napster's parent company had the financial resources to provide sufficiently powerful and reliable server hardware, home-run directory servers lacked the resources to support large-scale user communities and were far too unreliable. In time, these technical shortcomings drove users to abandon OpenNap.

BOINC (Anderson, 2004) is a flexible application framework based upon the SETI@Home paradigm. Whereas SETI@Home (Anderson, 2001) uses P2P technology to implement a super-computer dedicated to a specific task, BOINC provides a framework for creating P2P super-computers for any task. The BOINC network structure is very similar to that of SETI@Home and like SETI@Home, BOINC is

cross-platform. Unlike SETI@Home, BOINC is open source and designed to facilitate the deployment of existing applications.

Critique of Semi-Centralised P2P for Service Oriented Computing

Semi-centralised P2P systems can be considered as hybrid designs featuring elements of both client-server and peer-to-peer design. This being the case they naturally demonstrate only a subset of the advantages of a P2P approach. Both Napster (Merriden, 2001) and SETI@Home (Anderson, 2001) successfully leverage unused resources around the edge of the network to provide a service, however, the semi-centralised architectures they use to accomplish this have inherent limitations in terms of scalability, resilience and cost.

At its height Napster served a user community of over 50 million, therefore it is clearly possible for the semi-centralised model to scale to support large-scale service discovery and publication; however, this is dependent upon the presence of a central authority which is capable of adequately provisioning servers. As communication is highly asymmetric in semi-centralised systems, servers experience extreme loads. This necessitates specialist server hardware of adequate performance and reliability to store data on all available services and the peers providing them. While Napster proved that this model can work for large user communities where there is adequate funding by a central authority, it is inherently unsuitable for situations where no such authority exists. This is highlighted by the falling popularity of OpenNap (*The OpenNap Protocol Specification*, 2000).

As with Napster, SETI@Home suffered from significant dependability problems. Unlike Napster, where the resources being distributed are located on standard peers, in SETI@Home, work units are only distributed by the SETI@Home server, which was frequently overwhelmed during the early stages of the SETI@Home project and often had to be upgraded as the SETI@Home user community grew (Anderson, 2001). These scalability shortcomings are typical of semi-centralised P2P architectures.

Semi-centralised architectures are inherently more vulnerable than decentralised P2P architectures as central servers, such as Napster's indexing server, form single points of failure and the focal points for attack. In such systems, hardware or software failure in servers may result in failure of the entire system, and thus the unavailability of all services. Thus, while semi-centralised P2P architectrues may be the most intuitive P2P architecture to support web services (being closer in character to existing web service architectures), they offer poor scalability and reliability compared to more decentralised P2P architectures.

Unstructured, Decentralised P2P Systems

Unstructured decentralised P2P systems make connections essentially at random between peers, forming an unstructured overlay network which, unlike semi-centralised systems contains no servers or other special-purpose peers. Each peer on such a network is equally responsible for providing resources, routing network maintenance messages and routing resource discovery traffic. Unlike the *semi-centralised* systems discussed previously, unstructured decentralised networks meet all of the criteria in our definition of P2P systems. All peers are essentially equal; the network is highly decentralised and features symmetric communication. The remainder of this section discusses significant implementations of unstructured, decentralised P2P systems and concludes with a critique of unstructured, decentralised P2P networks for service oriented computing.

Gnutella 0.4

Gnutella (*The Gnutella Protocol Specification v.0.6, draft*, n.d.) is an open protocol which supports peer-to-peer resource discovery. The protocol builds an unstructured decentralised overlay network in which each host is required to forward both resource discovery and network maintenance messages. The protocol uses just five message types: PING (peer discovery search), PONG (peer discovery response), QUERY (resource discovery search), QUERYHIT (resource discovery response) and PUSH (a firewall

avoidance mechanism). Interaction on Gnutella can be considered to be structured into three phases: connecting to the network, discovering resources and transferring resources.

Connection: Acquisition of an initial host address (used to bootstrap network entry) occurs outside of the Gnutella protocol; typically via a Gnutella host cache (*The Gnutella Protocol Specification v.0.6, draft*, n.d.). A newly-arriving peer connects to an initial peer discovered in this way by initiating TCP connections to that host. Subsequently, further peers are discovered by sending a PING message. PING messages are broadcast by each peer to all of their neighbours, who in turn broadcast the message to all their neighbours and so on, 'flooding' the network. All available peers that receive a PING should respond with a PONG, which is forwarded back along the path of the incoming PING to the originating peer. PONG messages contain the network address and port on which the sending peer is listening for incoming Gnutella connections along with useful meta-data. To avoid swamping the network, all broadcast messages are tagged with a Time to Live (TTL) value, which is typically 7 hops. Peers decrement the TTL value of messages as they are routed, discarding messages where the TTL equals 0. This gives each peer a 'search horizon' of approximately 10,000 remote peers, outside of which search messages will not reach.

Resource Discovery: Peers listen for incoming QUERY messages, and contribute to their broadcast across the network by flooding them to each of their neighbours, as described above. If a peer is able to satisfy a QUERY, it responds by sending a QUERYHIT message back along the path of the incoming QUERY. QUERYHIT messages contain the network address and port on which the responding peer is listening for HTTP file-transfer connections. QUERYHITs also include the speed of the peer's connection, and a set of '*hits*' (matching file-names) which satisfy this QUERY.

Resource Distribution: Actual file-transfer, or service provision occurs outside of the Gnutella protocol. When a requesting peer receives a QUERYHIT message, it can attempt to initiate a direct download, from the target peer (whose port and IP address were specified in the QUERYHIT message) via HTTP. However, if the target peer is behind a firewall, the requesting peer can instead send a PUSH message to the target, containing details of the file requested and the network address and port to which the file should be pushed. On receiving a PUSH, the target peer establishes the HTTP connection and pushes the file to the requesting peer.

Gnutella 0.4 provided a decentralised alternative for supporting P2P file sharing following the demise of Napster. Due to its decentralised nature and lack of indexing servers; it could not be subject to the same legal attacks. Unfortunately, unstructured decentralised networks suffer from significant scalability problems (discussed in the critique at the end of this section) and for this reason these schemes have since largely been replaced by super-node architectures.

Other Unstructured, Decentralised P2P Systems

Alongside Gnutella, a number of other systems follow the unstructured P2P approach. *Freenet* (Clarke, Miller, Hong, Sandberg, & Wiley, 2002) is an anonymous and censorship resistant content publication and distribution system, which in its early incarnation used a decentralized architecture very similar to Gnutella. However, unlike Gnutella where users have control over their own file-share, Freenet automatically manages the assignment of materials across the network. As material published by a user may be stored anywhere within the network and resource discovery messages are routed through many intermediate peers, Freenet is highly anonymous. Recently, Freenet has moved towards a more structured network architecture in order to address the scalability shortcomings associated with unstructured decentralised networks.

GPU (Mengotti, 2004) implements a framework to support distributed processing over the Gnutella network. Like BOINC, GPU is open-source, cross platform and intended to be flexible; however, unlike these systems, GPU is entirely decentralised, allowing all users of the GPU network to submit tasks for processing. As GPU peers operate over the same network as Gnutella file-sharing peers, GPU has transitioned to a super-node architecture in Gnutella 0.6 (*The Gnutella Protocol Specification v.0.6, draft*, n.d.).

Critique of Unstructured Decentralised P2P Systems for Service Oriented Computing

Unstructured, decentralised P2P networks such as Gnutella (*The Gutella Protocol Specification v0.4, Document Revision 1.2*, n.d.) address some of the concerns associated with their semi-centralised predecessors in terms of resilience; however these networks are inherently incapable of offering reliable service location while remaining scalable. Unstructured decentralised P2P systems offer inherent support for complex search and are capable of scaling to large numbers of users without the need for specialised server hardware. Unfortunately, the broadcast resource discovery mechanism employed on these networks does not scale (Ritter, 2001) and thus all resource discovery messages are assigned a Time to Live (TTL) value which limits their propagation through the network. This in turn leads to an effect known as the *search horizon*, wherein nodes can only communicate with a subset of peers that are 'in range' on the Gnutella network. This makes it impossible for Gnutella to support reliable object location for large-scale networks. This limitation arises from the fact that unstructured decentralised networks treat all peers as equal and therefore bandwidth consumption due to message-passing must be sufficiently low so as not to exclude slow peers on low bandwidth links. In reality nodes are highly heterogeneous, and thus treating them as equals is undesirable as the capabilities of 'strong' nodes are not fully exploited, while the poor performance of 'weak' nodes may have a detrimental effect on their neighbours. For this reason, from version 0.6, Gnutella has adopted a hierarchical super-node network. Decentralised architectures such as Gnutella 0.4 are far more resilient than semi-centralised architectures such as Napster (Merriden, 2001) or SETI@Home (Anderson, 2001). Unlike these systems, Gnutella has no single point of failure, which makes it extremely resilient where resources are distributed evenly across the network. Thus, while unstructured, decentralised P2P systems offer better resilience and scalability than semi-centralised systems, they cannot offer large scale, reliable location of services, making them a poor fit for SOC wherein service consumers expect highly available, non-transient services.

Super-Node P2P Systems

Super-node systems differentiate nodes into two classes, *'leaf nodes'* and *'super nodes'*. Super nodes make connections essentially at random to other super nodes forming an unstructured, decentralised network, similar to those discussed in the previous section. Super nodes accept incoming connections from both super nodes and leaf nodes. Unlike super nodes, leaf nodes do not accept incoming connections, forming connections only to super nodes. This differentiation in peer responsibility means that super nodes route the majority of traffic.

Unlike unstructured decentralised P2P systems, peers in super-node networks are not equal, though the emergent network structure typically remains highly decentralised and usually features symmetric communication between super nodes. The remainder of this section discusses a significant super-node P2P system: Gnutella 0.6 (*The Gnutella Protocol Specification v.0.6, draft*, n.d.) and concludes with a critique of the super-node model of interaction for service oriented computing.

Gnutella 0.6

Gnutella 0.6 (*The Gnutella Protocol Specification v.0.6, draft*, n.d.) uses a super-node network structure. Gnutella's super node architecture is implemented through the Gnutella *ultra-peer* scheme (*The Gnutella Protocol Specification v.0.6, draft*, n.d.). The ultra-peer scheme was introduced to reduce the scalability problems caused by the flooding search used in Gnutella 0.4 (described previously). Alongside the differential connection behaviour, which is common to all super node schemes, Gnutella ultra-peers are also required to proxy for leaf nodes as specified in the Gnutella QUERY Routing Protocol (QRP) (*The Gnutella Protocol Specification v.0.6, draft*, n.d.). Ultra peers also cache PONG messages according to the Gnutella PONG caching scheme (*The Gnutella Protocol Specification v.0.6, draft*, n.d.).

As ultra-peers respond to incoming PING messages in the same way as Gnutella 0.4 peers, these nodes form an unstructured decentralised network such as that formed by Gnutella 0.4 peers, however, leaf-nodes do not respond to incoming PING messages, forming only a few connections to super-nodes – making them the *leaves* of the network graph. As each ultra-peer maintains a list of what files their leaf-

nodes neighbours are sharing, it is possible for ultra-peers to proxy for their leaf-nodes, effectively acting as distributed index servers for leaf-nodes and thus further reducing traffic for these nodes. Also, as leaf-nodes do not receive PING messages, the level of traffic that they are required to route is reduced still further.

Most Gnutella 0.6 clients allow their users to decide whether they would like to join the network as an ultra-peer or leaf-node. Those users with powerful computers and fast Internet connections are expected to elect to become ultra-peers, while users with less powerful PCs or slower Internet connections are expected to elect to become leaf-nodes. By adapting the role that each node plays in this way, their capabilities are better exploited and the contribution that each node makes to the system as a whole is somewhat optimised. Similarly, by adapting the way that each node is treated by the system, it is possible to maximize its suitability for supporting different service classes.

Other Super-Node P2P Networks

Alongside Gnutella 0.6, several other notable P2P systems use a super-node approach. *Kazaa* (*Kazaa Home Page*, 2001) is a commercial implementation of a super-node network similar to Gnutella 0.6. As Kazaa is a commercial implementation, its protocol is not open and therefore cannot be discussed in detail here. However, two key differences between Kazaa and Gnutella 0.6 are apparent. Firstly, like Napster, Kazaa uses usernames and passwords, which are authenticated when a node is boot-strapped onto the network. Secondly, Kazaa uses specialised indexing servers to augment the super-node network. This increases the performance of Kazaa, however, it does so at the expense of both scalability and resilience.

Various schemes have been proposed to better manage the election of super-nodes, for example *SG-1* (Montresor, 2004) addresses the issue of reduced resilience on super-node networks (discussed in the following critique), while *SUPS* (Pyun & Reeves, 2004) applies random graph theory to the election of super-nodes in order to ensure that super-node networks remain scalable and avoid the issues which may arise either due to either a lack of super-nodes or poor selection of super-nodes.

Critique of Super-Node P2P Systems

Super-node P2P networks such as Gnutella 0.6 (*The Gnutella Protocol Specification v.0.6, draft*, n.d.) and Kazaa (*Kazaa Home Page*, 2001) address some of the scalability issues associated with their entirely decentralised predecessors, however, super-node schemes do so at the expense of resilience and are equally incapable of providing truly reliable object location. As with unstructured decentralised P2P systems, super-node schemes offer inherent support for complex search and are capable of scaling to large numbers of users without the need for specialised server hardware. Unfortunately, while the *search horizon* of these networks is significantly larger than unstructured networks (an order of magnitude for Gnutella 0.6) it is still possible for peers to fall out of range of one another. This makes it impossible for Gnutella 0.6 to support reliable service publication.

Unfortunately, the scalability improvements offered by super-node schemes come at the cost of asymmetric network communication and to some extent, this asymmetric communication brings with it the issues associated with semi-centralised P2P systems - reduced resilience to attack and lower fault tolerance. In most cases, leaf nodes will be connected to multiple Super-Nodes and therefore the failure of a single Super-Node should not cause a service interruption for attached leaf-nodes, however, as leaf-nodes typically only connect to a few super-nodes, the failure of a super-node may cause a significant reduction in quality of service for attached leaf-nodes. This is better than the semi-centralised model, wherein the failure of an indexing server may cause a loss of service for the entire network, but worse than unstructured decentralised networks. In addition, as the failure of super-nodes is likely to have a significant effect on their leaf-nodes, they form a logical point of attack. Again this is preferable to the semi-centralised approach, but worse than unstructured, decentralised approaches. Thus, while super-node networks are a more promising platform for supporting SOC than semi-centralised or unstructured, decentralised P2P systems, they still cannot provide reliable availability of services. This is addressed to an extent by structured, decentralised systems, which are discussed below.

Structured, Decentralised P2P Systems

Structured P2P systems are designed to provide efficient resource location and routing primitives. A strictly defined network structure is maintained by controlling the way that peers connect to their neighbours, for example through the implementation of a distributed hash table (DHT). A DHT abstraction provides the same functionality as a traditional hash table, wherein nodes and resources on the network are mapped onto a unique key using a hash function. Nodes on such a network make connections to peers according to their key value and in this way; the hash function defines a specific structure for the overlay network. Typically, all nodes in structured decentralised networks are considered equal and are not differentiated in terms of functionality or responsibility as in super-node or semi-centralised networks. The remainder of this section discusses three significant structured, decentralised P2P systems: Chord (Stoica, Morris, Karger, Kaashoek, & Balakrishnan, 2001), CAN (Ratnasamy, Francis, Handley, Karp, & Shenker, 2001) and Pastry (Rowstron & Druschel, 2001). It concludes with a critique of the structured, decentralised model of interaction.

Chord

Chord (Stoica, Morris, Karger, Kaashoek, & Balakrishnan, 2001) is a simple DHT which uses a consistent hash function to map keys onto nodes. The use of a consistent hash function means that nodes will always be mapped onto the same key. Nodes join the Chord overlay based upon their key value, such that nodes close in the key space are located close to each other on the overlay network. Each Chord peer maintains information about the preceding peer and *log n* other peers in what are known as 'finger tables'. The result of this is that each peer has knowledge of the following node in the ring, then the node 2 hops away, 4 hops away and so on until the termination of the Chord ring structure. For example, peer A in the simplified Chord overlay shown in figure 1 would have knowledge of nodes {B, D, and H}. Due to the highly structured nature of the Chord network, all data can be predictably located in *O(Log N)* hops. The consistent hash function usually used with Chord is the Secure Hash Algorithm – SHA1 (Eastlake & Jones, 2001). SHA1 takes an input of up to 263 bits and produces a 160-bit key. It is computationally expensive to compute a value from a given key, but inexpensive to generate a key from a given value, thus it is difficult to deduce the IP address of a peer based upon its key.

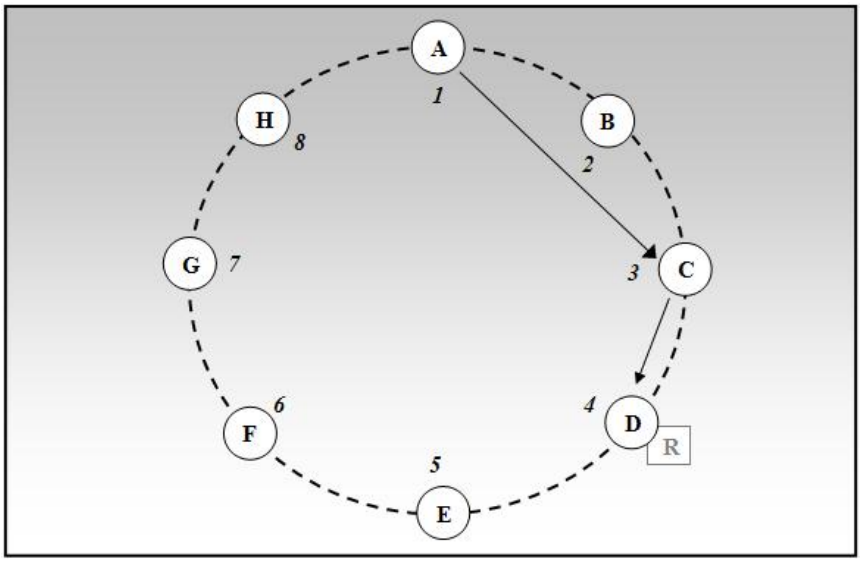


Figure 2 – Chord, a Structured Decentralised P2P Network

Chord does not implement a full resource location and distribution system; rather it implements a simple Key Based Routing (KBR) algorithm. However, Chord is often used to support distributed storage

systems, wherein each peer on the Chord overlay network is assigned responsibility for part of a distributed data set. Data is mapped onto Chord nodes by hashing with the SHA1 algorithm and each node is made responsible for storing all data which has a key of less value than its own and which is not already stored on a preceding peer. For example, if a piece of data 'D2' had a hash value of 3.5, this data would be stored onto node D. Reliable lookup of this data can then be performed in O(Log N) time from any node. The lookup path from *node A* for this resource is shown in Figure 2.

Content Addressable Network

Content Addressable Network (CAN) (Ratnasamy, Francis, Handley, Karp, & Shenker, 2001) is another example of distributed hash table that enables the insertion, look-up and deletion of key-value pairs. Key/value pairs in CAN are mapped onto a virtual Cartesian coordinate space. This space is dynamically (re)partitioned between nodes, such that each node is responsible for a section of the virtual space. Each key-value pair is mapped onto a point in the virtual coordinate space by hashing its key with a uniform hash function and then storing it on whatever CAN node is responsible for that section of the coordinate space. This is shown in figure 3 below. Alongside maintaining a database of key/value pairs which map to its zone, each node in a CAN network maintains a routing table containing each of its closest neighbours along the X axis and the Y axis in both the positive and negative direction, together with the coordinates of these nodes, such that the routing table of Node D would contain **{**E, B, A, C**}** and the routing table of node B would contain **{**E, D, A**}**. In order to route a message to a given point in the CAN, each node, simply forwards the message to the node in its routing table with the closest coordinates.

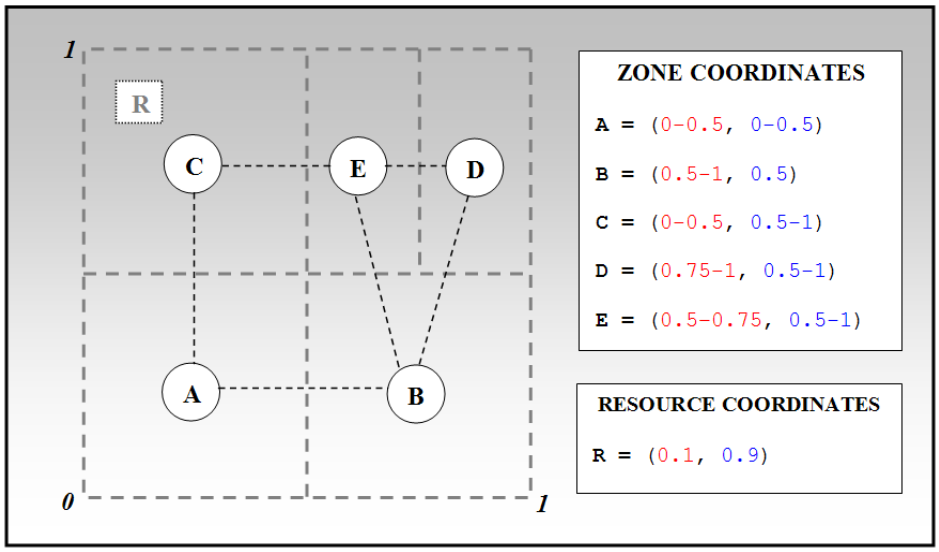


Figure 3 – CAN, a Structured Decentralised P2P Network

Unlike logarithmic DHTs such as Chord, the connection of a peer to the CAN network affects a much smaller region of the network. When nodes leave CAN, it is essential to ensure that the responsibility for the zone that the leaving node is currently responsible for is transferred to another node in order to avoid the loss of data in that zone. Suppose that *Node E* wishes to leave the CAN network. If the node's zone can be merged with a neighbour to form a valid Cartesian zone, then the key/value database is explicitly transferred. If not, then the neighbour node with the smallest zone temporarily takes control of two zones. If a node fails without transferring its zone, that zone can become inaccessible. To prevent this, each node also maintains the routing table of neighbouring peers. CAN nodes aggressively probe their neighbours and, in the event that node failure is detected send take-over messages to each node in their neighbour's routing table. In the event that multiple adjacent nodes fail, CAN nodes attempt to respond by broadcasting take-over messages across the overlay network. There are many potential paths between nodes in the CAN coordinate space, which makes the overlay particularly robust in the

presence of node failure. If the node which is the next closest hop to a target zone fails, then a peer will simply select the next closest entry from its routing table.

Other Structured, Decentralised P2P Systems

Pastry (Rowstron & Druschel, 2001) implements a scalable application level routing and object location service using a DHT. Pastry adds optimisations based upon network proximity and adapts better to the arrival and failure of nodes. As with Chord, each node has a unique node identifier and nodes join the overlay based upon their key value, such that nodes with close numeric values are located close to each other on the overlay network. Keys in Pastry are randomly assigned and a node will not necessarily be re-assigned the same ID upon re-joining the network. Pastry seeks to minimise the distance that messages travel on the underlying network using a scalar proximity metric based on the number of hops between nodes on the underlying IP network. The circular Pastry key space closely resembles the Chord ring illustrated in Figure 2. However, Pastry nodes maintain additional routing tables which are used to optimize routing based upon the underlying network structure.

Tapestry (Plaxton, Rajaraman, & Richa, 2007) is a structured, decentralised network which attempts to ensure reliability through high levels of redundancy, thus ensuring the systems resilience against both failure and attack. Like other structured networks, Tapestry supports efficient object location and retrieval; however, unlike the other systems discussed in this section, it builds a network corresponding to a Plaxton mesh (Castro, Costa, & Rowstron, 2003), rather than a ring structure as in Chord (Stoica, Morris, Karger, Kaashoek, & Balakrishnan, 2001) or a Cartesian coordinate space as in CAN (Ratnasamy, Francis, Handley, Karp, & Shenker, 2001).

Critique of Structured, Decentralised Networks

Structured decentralised networks such as Chord (Stoica, Morris, Karger, Kaashoek, & Balakrishnan, 2001), CAN (Ratnasamy, Francis, Handley, Karp, & Shenker, 2001) and Pastry (Rowstron & Druschel, 2001) address the issue of providing efficient and reliable service publication for large-scale P2P systems. However, unlike unstructured and super-node schemes such as Gnutella 0.4 (*The Gutella Protocol Specification v0.4, Document Revision 1.2*, n.d.) and Gnutella 0.6 (*The Gnutella Protocol Specification v.0.6, draft*, n.d.), these networks are not inherently capable of service discovery. While each of the structured, decentralised P2P systems discussed in this section supports efficient lookup of services, this requires that a node performing such a look-up knows the key of that resource in advance. Acquisition of this key is non-trivial and requires the implementation of a separate search mechanism. This may take the form of distributed indexing servers, which associate file-names with keys, or the use of a second overlay network, more suitable for supporting search. This overlay may, rather than being implemented directly over transport-layer protocols such as TCP, use the structured network as a routing substrate. Systems which make use of such a multi-layer P2P architecture are discussed in the *future trends* section of this chapter. Thus while structured P2P systems provide an important part of what is needed to support P2P SOC – reliable service publication and binding, they do not support complex service discovery.

SUMMARY OF P2P SUPPORT FOR SERVICE ORIENTED COMPUTING

Based upon the survey performed in the previous section, it can be seen that each of the major P2P architectures provide part of the solution to supporting service oriented computing. Unstructured or super-node systems provide efficient support for service location, while structured networks provide good support for service publication and binding. This is summarised in table 1 below:

Table 1 – P2P Architectures for Service Oriented Computing

	Semi Centralised	Unstructured	Super Node	Structured
Service Publication	**GOOD** *Reliable service location.*	**POOR** *Services may fall out of range.*	**POOR** *Services may fall out of range.*	**EXCELLENT** *Efficient and reliable service publication.*
Service Discovery	**GOOD** *Search via indexing server.*	**GOOD** *Inherent discovery through flooding search.*	**GOOD** *Inherent discovery through flooding search.*	**NONE** *No support for service discovery.*
Scalability	**VERY POOR** *Requires big, costly servers.*	**POOR** *Flooding search doesn't scale.*	**GOOD** *Improved, but un-scalable.*	**EXCELLENT** *Efficient, scalable lookup.*
Resilience	**VERY POOR** *Failure of indexing server affects all nodes.*	**EXCELLENT** *Resistant to failure & attack.*	**GOOD** *Less resistant to failure of super-nodes.*	**EXCELLENT** *Resistant to failure & attack.*

As can be seen from table 1, no single P2P architecture provides good support for service oriented computing. The next section however introduces some experimental P2P systems which use elements from various architectures to provide support for service oriented computing.

FUTURE TRENDS IN PEER-TO-PEER ARCHITECTURES

Multi-layer peer-to-peer architectures have recently begun to emerge and, though they have yet to be deployed on a large scale, they are a promising platform for SOC. These architectures attempt to combine the benefits of unstructured decentralised networks (which provide flexible service discovery) and structured decentralised networks (which provide efficient service publication). Multi-layer networks accomplish this by building a structured decentralised network, which is then used as the underlying routing and service publication substrate and is overlaid with an unstructured, decentralised network which provides support for service discovery. In multi-layer architectures, the lower overlay could be any structured, decentralised network such as CAN (Ratnasamy, Francis, Handley, Karp, & Shenker, 2001), Chord (Stoica, Morris, Karger, Kaashoek, & Balakrishnan, 2001) or Pastry (Rowstron & Druschel, 2001), while any unstructured decentralised network such as Gnutella (*The Gutella Protocol Specification v0.4, Document Revision 1.2*, n.d.) or GPU (Mengotti, 2004) could be layered on top of this to provide search. In such architectures, the structured overlay essentially fulfils the role that transport-layer protocols play in single-layer P2P systems. The remainder of this section discusses significant multi-layer P2P systems and discusses their suitability for supporting SOC.

Structella

As described in previously, a major shortcoming of structured, decentralised networks is that, unlike unstructured or super-node networks, they do not provide inherent support for complex queries such as plain-text search. Initial research on retrofitting structured decentralised networks with search capability focus on distributed index server approaches, however this has the same associated drawbacks as super-node schemes, and to a lesser extent semi-centralised systems.

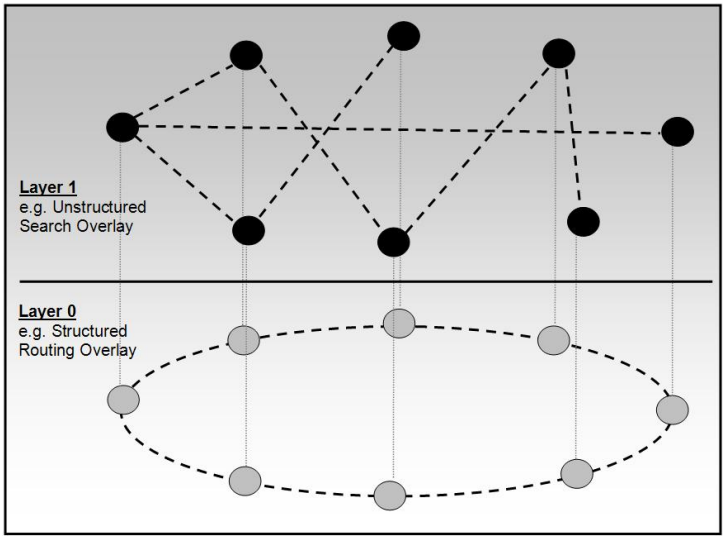


Figure 4 – Structella, a Multi-Layer Network

Structella (Castro, Costa, & Rowstron, 2003) proposes a more architecturally interesting solution; layering an unstructured decentralised network on top of a structured decentralised overlay, theoretically combining the primary benefits of both architectures: scalable object lookup and inherent support for complex queries. Structella uses the Pastry network as a routing substrate (layer 0 in figure 4). This is overlaid with a cross-layer optimised version of Gnutella (layer 1 in figure 4 to provide support for resource discovery. As with Gnutella (*The Gutella Protocol Specification v0.4, Document Revision 1.2*, n.d.), Structella *does not* manage the distribution of services across the network, allowing users to host their own content. Structella offers support for two connection mechanisms, *flooding search* and *random walk*. The connection process for Structella is virtually identical to the connection process for Gnutella (*The Gutella Protocol Specification v0.4, Document Revision 1.2*, n.d.), save that messages are routed via a structured, decentralised network rather than via TCP/IP.

The Structella resource discovery mechanism is much more efficient than that used on unstructured, decentralised networks such as Gnutella (*The Gutella Protocol Specification v0.4, Document Revision 1.2*, n.d.), as the use of Pastry's proximity-aware routing table ensures that QUERY messages should not be received more than once by each peer. In this way, Structella combines the benefits of structured networks (reliable service publication) with a more reliable and scalable version of Gnutella's service discovery facilities.

Open Overlays

The Open Overlays middleware (Grace, Coulson, Blair, Mathy, Yeung, Cai, Duce, & Cooper, 2004) provides an abstraction in which different overlay networks are universally modelled using standardised '*overlay*' component frameworks. The framework accepts '*plug-in*' components that offer various types of overlay-related behaviour from underlying transport protocols such as TCP to DHTs such as Chord (Stoica, Morris, Karger, Kaashoek, & Balakrishnan, 2001). Overlay plug-ins are themselves 'mini' component frameworks, each of which is composed of three distinct components that encapsulate the following areas of behaviour:

i. *Control*, in which the node implements cooperation with the peer control element on other nodes to build and maintain an overlay-specific topology;
ii. *Forwarding*, which determines how the overlay will route messages over the aforementioned topology
iii. *State*, information required for the overlay, for example the nearest neighbours.

As these elements expose standard interfaces, Open Overlays allows the creation of diverse P2P systems using a variety of P2P network protocols. Furthermore, the resulting system may be re-configured at runtime to modify system functionality. Overlay component frameworks form powerful building blocks which can be used to assemble systems using heterogeneous overlays, including multi-layer overlays. Furthermore, the open overlays middleware provides support for common web service technologies, as described in (Grace, Coulson, Blair, Mathy, Yeung, Cai, Duce, & Cooper, 2004).

Multi-Layer P2P Networks for SOC

Multi-layer overlay networks such as Structella (Castro, Costa, & Rowstron, 2003) address the two critical issues for supporting Service Oriented Computing (SOC) which are not adequately addressed by other architectures. Firstly, they are capable of supporting both scalable lookup and search. Though the latter depends upon cross layer optimisation between the search and routing layers.

Multi-layer networks, such as Structella (Castro, Costa, & Rowstron, 2003), inherit the scalable and reliable object location offered by the lower structured layer. Furthermore, multi-layer networks such as Structella can exploit the inherent structure of the structured routing substrate to improve the scalability of broadcast service discovery. For example, Structella uses Pastry's proximity-aware routing table (Rowstron & Druschel, 2001) to ensure that search messages are not re-broadcast to the same peers. Without such cross-layer optimisation the search layer would always demonstrate lower per-hop efficiency than single layer schemes such as Gnutella (*The Gutella Protocol Specification v0.4, Document Revision 1.2*, n.d.) due to the overhead of maintaining multiple layers.

Multi-layer overlay networks also maintain complete decentralisation and therefore enjoy a high level of resilience to node failure and attack. In these networks, as in single-layer structured decentralised networks, the failure of even a large number of nodes typically has little effect upon routing efficiency. This makes the emerging class of multi-layer P2P systems highly promising for supporting SOC, though these networks have yet to be proven through wide-scale deployment.

Current Work in the Field of P2P SOC

Amoretti et al. (Amoretti, Zanichelli, & Conte, 2008) propose web service extensions based on SOAP (Box, Ehnebuske, Kakivaya, Layman, Mendelsohn, Nielsen, Thatte, & Winer, 2000). to the JXTA P2P middleware (*JXTA Home Page*, n.d.). The JXTA P2P middleware is a generalized platform for supporting P2P networking which can use multiple underlying network types, and therefore depending upon the network architecture employed the architecture will exhibit different characteristics. By layering SOAP over JXTA, Amoretti et al. take an important step towards supporting integration between P2P SOC and traditional SOC architectures, however this work addresses neither the architectural concerns highlighted in the earlier survey nor significant outstanding challenges in the fields of reliability, security and trust, as will be discussed in the following section.

Gerke et al. (2003) propose a more decentralised approach for providing services based on an unstructured network, however, it is clear that their approach will suffer from the inherent limitations of unstructured decentralised networks, as discussed previously in this chapter. Furthermore, while Gerke et al. provide rich SOC functionality, they do not consider the critical issues of reliability, security and trust, which are of paramount importance in decentralised, unreliable and largely anonymous P2P networks.

Sacha et al. (2007) also propose a decentralised P2P SOC approach which considers the issues of node reliability using a network of trust in which peers with better reliability characteristics are moved into optimal positions in the network, while peers with sub-optimal characteristics are marginalized. The supporting implementation was developed on a structured decentralised network and thus provides good support for reliable service publication but poor support for service discovery.

Outstanding Issues in Supporting SOC using P2P technologies

This chapter has thus far focused upon supporting the basic elements of SOC computing using P2P systems – i.e. supporting large scale service discovery and publication. However there remain significant outstanding issues if P2P approaches are to be adopted for supporting SOC.

Perhaps the most critical outstanding issue is that of *security*. As modern P2P networks are inherently decentralised and lack single points of control, implementing effective security is difficult, and first requires authentication of peers to establish their identity and also the establishment of networks of *trust* (Wang & Vassileva, 2003; Koutrouli & Tsalgatidou, 2008), which can offer security assurances in lieu of a central authority. This is further complicated if services are expected to be provided by 'standard peers', as it may be possible for end users to offer malicious services which perform differently than advertised. Research into establishing networks of trust may also be useful in addressing this problem.

Peer *reliability* is another important issue. The nodes that typically compose a P2P network are inherently less reliable than more expensive and well provisioned server machines. Furthermore, these peers may leave the network at any time as users deactivate their machines. Even where machines remain connected and functional, peers may be concurrently executing competing applications, the resource usage of which can lead to diminished quality of service (QoS) characteristics. This lowered reliability demands research in two domains. The first being support for adaptation in P2P environments, which is necessary to deal with changing resource availability. The second being P2P caching approaches, that may also be applied in order to replicate services and hence avoid service unavailability due to peer failure or deactivation.

While P2P systems are a promising platform for supporting SOC, *integration with legacy web service technologies* such as WSDL (Christensen, Curbera, Meredish, & Weerawarana, 2001) and SOAP (Box, Ehnebuske, Kakivaya, Layman, Mendelsohn, Nielsen, Thatte, & Winer, 2000). will also be necessary to maximize the usefulness of P2P SOC platforms. This will allow organizations which have invested significant time and money into current SOC approaches to transition smoothly to P2P technologies. The resulting system, based around a multi-layer P2P architecture and offering end-to-end consideration of reliability and security would be an extremely powerful platform for supporting SOC.

CHAPTER SUMMARY

This chapter has introduced P2P systems and discussed how they may be used to support large scale decentralised service oriented computing. A classification of P2P architectures was presented and the suitability of each of these architectures for supporting service oriented computing was discussed. *Semi-centralised systems*, typified by Napster (Merriden, 2001) and Seti@Home (Anderson, 2001) were discussed and the problems inherent in this architecture – low scalability and resilience were highlighted. Next, *unstructured decentralised systems* were discussed along with applications that use this architecture such as Gnutella (*The Gutella Protocol Specification v0.4, Document Revision 1.2*, n.d.) and GPU (Mengotti, 2004). This P2P architecture was presented as being well suited to service discovery, but poorly suited to reliable service publication due to the inherent 'search horizon' limitations of this architecture. Following this, *super-node* architectures such as Gnutella 0.6 (*The Gnutella Protocol Specification v.0.6, draft*, n.d.) were discussed. It was shown that super-node architectures improve upon unstructured decentalised systems, but are still unable to provide reliable service publication due to the search horizon effect. Finally, *structured decentralised networks* were discussed and it was shown that this network architecture provides good support for service publication and binding, though it provides no support for service location.

While the survey of existing P2P network architectures revealed that no one architecture provides a complete solution for SOC, the emerging field of *multi-layer overlay networks* such as Structella (Castro, Costa, & Rowstron, 2003) and Open Overlays (Grace, Coulson, Blair, Mathy, Yeung, Cai, Duce, & Cooper, 2004) provides the best all-around support for SOC, offering reliable service publication via a base structured decentralised network and service discovery through an unstructured decentralised network which is overlaid on the base layer. Despite the potential of these architectures, a number of

critical issues were identified that must be addressed before P2P architectures can be used to support SOC. These issues include: security, reliability and integration with legacy web services technologies.

REFERENCES

Amoretti, M., Zanichelli, F., Conte, G. (2008, April). Enabling Peer-To-Peer Web Service Architectures With JXTA-SOAP. In *Proceedings of IADIS International Conference e-Society 2008*, Algarve, Portugal.

Anderson, D. (2001). SETI@Home. In A. Oram (Ed.), *Peer-to-Peer: Harnessing the Power of Disruptive Technologies* (pp. 67-76). Sebastopol, CA: O'Reilly and Associates.

Anderson, D. P. (2004, November). BOINC: A System for Public-Resource Computing and Storage. In *proceedings of the 5th IEEE/ACM workshop on Grid Computing (Grid '04)*, Pittsburgh, USA (pp. 4-10).

Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., & Winer, D. (2000, May 8). *Simple Object Access Protocol (SOAP) 1.1*. Retrieved from http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

Bryan, D., Draluk, V., Ehnebuske, D., Glover, T., Hately, A., Husband, Y. L., Karp, A., Kibakura, K., Kurt, C., Lancelle, J., Lee, S., MacRoibeaird, S., Manes, A. T., McKee, B., Munter, J., Nordan, T., Reeves, C., Rogers, D., Tomlinson, C., Tosun, C., von Riegen, C., & Yendluri, P. (2002, July 19). *The UDDI Version 2 API Specification*. Retrieved from http://uddi.org/pubs/ProgrammersAPI-V2.04-Published-20020719.htm

Castro, M., Costa, M., & Rowstron, A. (2003, November). Should we build Gnutella on a Structured Overlay. In *proceedings of the 2nd Workshop on Hot Topics in Networks (HotNets-II)*, Cambridge, MA, USA (pp. 131-136).

Christensen, E., Curbera, F., Meredith, G., & Weerawarana, S. (2001, March 15). *Web Services Description Language (WSDL) 1.1*. Retrieved from http://www.w3.org/TR/2001/NOTE-wsdl-20010315

Clarke, I., Miller, S. G., Hong, T. W., Sandberg, O., & Wiley, B. (2002, January/February). Protecting Freedom of Information Online with Freenet. *IEEE Internet Computing, 6*(9), 40-49. Retrieved from http://www.freenetproject.org/papers/freenet-ieee.pdf

Eastlake, D. E., & Jones, P. E. (2001, September). *US Secure Hash Algorithm 1 (SHA1), Internet Engineering Task Force, RFC3174*. Retrieved from http://www.faqs.org/rfcs/rfc3174.html

Gerke, J., Hausheer, D., Mischke, J., & Stiller, B. (2003, April). An Architecture for a Service Oriented Peer-to-Peer System (SOPPS), Praxis der Informationsverarbeitung und Kommunikation (PIK) 2/03, pages 90-95, April, 2003.

Grace, P., Coulson, G., Blair, G., Mathy, L., Yeung, W. K., Cai, W., Duce, D., & Cooper, C. (2004, October). GRIDKIT: Pluggable Overlay Networks for Grid Computing. In *proceedings of the International Symposium on Distributed Objects and Applications (DOA'04)*, Larnaca, Cyprus (LNCS 3291, pp. 1463-1481). Berlin, Germany: Springer.

Huhns, M. H., & Singh, M. P. (2005). Service-Oriented Computing: Key Concepts and Principles. *IEEE Internet Computing, 9*(1), 75-81.

JXTA Home Page. (n.d.). Retrieved from https://jxta.dev.java.net/

Kazaa Home Page. (2001). Retrieved from http://www.kazaa.com

Koutrouli, E., & Tsalgatidou, A. (2008, August). P2P Reputation Systems Credibility Analysis: Tradeoffs and Design Decisions. In *Proceedings of 12th Pan-Hellenic Conference on Informatics (PCI 2008)*, Samos, Greece (pp. 88-92). Washington, DC: IEEE Computer Society.

Mengotti, T. (2004, March). *GPU, A Framework for Distributed Computing over Gnutella*. Unpublished master's thesis in Computer Science, ETH Zurich, Switzerland.

Merriden, T. (2001). *Irresistible Forces: the Business Legacy of Napster and the Growth of the Underground Internet*. Capstone Publishing.

Montresor, A. (2004, August). A Robust Protocol for Building Superpeer Overlay Topologies. In *proceedings of the 4th IEEE International Conference on Peer-to-Peer computing (P2P'04),* Zurich, Switzerland (pp. 202-210).

Plaxton, C. G., Rajaraman, R., & Richa, A. W. (2007, June). Accessing nearby copies of replicated objects in a distributed environment. In *Proceedings of the 19th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, San Diego, CA, USA (pp. 311-320).

Pyun, Y., & Reeves, D. (2004, August). Constructing a Balanced, log(N)-Diameter Super-peer Topology. In *Proceedings of the 4th IEEE International Conference on Peer-to-Peer computing (P2P'04),* Zurich, Switzerland (pp. 210-218).

Ratnasamy, S., Francis, P., Handley, M., Karp, R., & Shenker, S. (2001, August). A Scalable Content Addressable Network In *proceedings of the ACM SIGCOMM Symposium on Communication, Architecture, and Protocols (SIGCOMM '01)*, San Diego, CA, USA (pp. 161-172).

Ritter, J. (2001, February). *Why Gnutella Can't Scale. No Really*. Retrieved from http://www.darkridge.com/~jpr5/doc/gnutella.html

Rowstron, A., & Druschel, P. (2001, November). Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware'01),* Heidelberg, Germany (pp. 329-350).

Sacha, J., Biskupski, B., Dahlem, D., Cunningham, R., Dowling, J., & Meier, R. (2007). A Service-Oriented Peer-to-Peer Architecture for a Digital Ecosystem. In *Proceedings of the IEEE International Conference on Digital Ecosystems and Technologies (DEST'07)*, Cairns, Australia.

Shirky, C. (2001). Listening to Napster. In A. Oram (Ed.), *Peer-to-Peer: Harnessing the Power of Disruptive Technologies* (pp. 21-37). Sebastopol, CA: O'Reilly and Associates.

Stoica, I., Morris, R., Karger, D., Kaashoek, F., & Balakrishnan, H. (2001, September). Chord: A scalable peer-to-peer lookup service for internet applications. In *proceedings of ACM SIGCOMM 2001*, San Diego, CA, USA (pp. 149-160). New York: ACM Press.

The Gnutella Protocol Specification v0.4, Document Revision 1.2. (n.d.). Retrieved from http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf

The Gnutella Protocol Specification v0.6, draft. (n.d.). Retrieved from http://rfc-gnutella.sourceforge.net/src/rfc-0_6-draft.html

The OpenNap Protocol Specification. (2000). Retrieved from http://opennap.sourceforge.net/

Wang, Y., & Vassileva, J. (2003). Trust and Reputation Model in Peer-To-Peer Networks. In *proceedings of the Third International Conference on Peer-to-Peer Computing (P2P'03),* Linkoping, Sweden (pp. 150-157).

KEY TERMS

Service Oriented Computing (SOC): The a family of tools and approaches, which aim to reflect component trends towards anonymity and heterogeneity, providing systems which are better suited to today's large-scale and open distributed environments.

Peer-to-Peer Systems (P2P): Peer-to-peer is a class of applications that takes advantage of resources - storage, cycles, content, human presence - available at the edges of the Internet.

Semi-Centralised P2P Systems: use a semi-centralised network architecture, wherein a small number of dedicated servers mediate interaction between large numbers of peers at the edge of the network.

Unstructured, decentralised P2P systems: are entirely decentralised, connecting peers together in an essentially random manner and treating all peers equally.

Super-node P2P systems feature decentralised server-less communication, however, all peers are not treated equally. More powerful peers on faster connections may elect to become ‘super-nodes’ which route a greater proportion of messages and may also provide special services.

Structured, decentralised P2P systems feature decentralised, server-less communication supported by a highly structured overlay network.

Multi-Layer P2P Architectures layer multiple network architectures in order to provide an emergent network architecture which combines the benefits of the component architectures.