From Client-Server to P2P Networking

Lu Liu and Nick Antonopoulos

Abstract Peer-to-peer (P2P) networks attract attentions worldwide with their great success in file sharing networks (such as Napster, Gnutella, Freenet, BitTorrent, Kazaa, and JXTA). An explosive increase in the popularity of P2P networks has been witnessed by millions of Internet users. In this chapter, an investigation of network architecture evolution, from client-server to P2P networking, will be given, underlining the benefits and the potential problems of existing approaches, which provides essential theoretical base to drive future generation of distributed systems.

1 Introduction

As a new design pattern, peer-to-peer (P2P) has been widely used in the design of large-scale distributed applications. An explosive increase in the popularity of P2P file sharing applications has been witnessed by millions of Internet users.

As an emerging technology, P2P networks attract attention worldwide, ranging from casual Internet users to venture capitalists. At the same time, the innovations of P2P networks also offer many interesting avenues of research for scientific communities. In the last few years, great research achievements have been made on P2P resource sharing and data transfer. Network architectures are starting to evolve from centralised client-server architectures to distributed P2P architectures or hybrid architectures between client-server and P2P.

Lu Liu
School of Computing, University of Leeds, Leeds, West Yorkshire, LS2 9JT, United Kingdom, e-mail: luliu@comp.leeds.ac.uk

Nick Antonopoulos
Department of Computing, University of Surrey, Guildford, Surrey, GU2 7XH, United Kingdom, e-mail: n.antonopoulos@surrey.ac.uk

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_3,

In this chapter, the network architecture evolution is discussed from client-server to P2P. A summary of recent solutions for resource discovery in P2P networks is also given underlining the benefits and the potential problems of these solutions.

2 Network Architecture Evolution

2.1 Client-Server Architecture

Many today's Internet applications, such as WWW, FTP, email, are distributed by using the client-server architecture (Fig. 1). In the client-server architecture, many clients request and receive services from the server(s). All the contents and services are stored and provided by a server(s). Content and services can be discovered and utilised efficiently by querying the centralised server(s), if the server(s) is available and capable of serving all clients at the same time.

Fig. 1 Client-server architecture

However, such centralisation of the client-server architecture raises a series of issues which are caused by the limitation of resources at the server side, such as network bandwidth, CPU capability, Input/Output (I/O) speed and storage space. A server could be overloaded if too many requests are received. In order to cope with these limitations, the centralised server(s) needs to bear the high costs of providing sufficient resources. For instance, Google clusters more than 200,000 machines to give successful Web indexing services [1].

Moreover, the centralisation of the client-server architecture also leads to the problem of single-point-of-failure. If the centralised server(s) is removed or is not available for use, no alternative in the architecture can take its place and all services on the server(s) will be lost.

2.2 Grid Architecture

"Grid Computing" is rapidly emerging from the scientific and academic area to the industrial and commercial world. Current Grid computing systems are prominent implementations of client-server architecture for distributed computing.

The vision of the Grid Computing is to provide high performance computing and data infrastructure supporting flexible, secure and coordinated resource sharing among dynamic collections of individuals and institutions known as "virtual organizations" (VO) [2, 3]. The main focus of Grid architecture is on interoperability among resource providers and users in order to establish the sharing relationship, which needs common protocols at each layer of architecture. It is intended to offer seamless and uniform access to substantial resources without having to consider their geographical locations. Resources in the Grid can be high performance supercomputers, massive storage space, sensors, satellites, software applications, and data belonging to different institutions and connected through the Internet. The Grid provides the infrastructure that enables dispersed institutions (commercial companies, universities, government institutions, and laboratories) to form virtual organisations (VOs) that share resources and collaborate for the sake of solving common problems [2, 3].

2.3 Peer-to-Peer Architecture

P2P networks are decentralised distributed systems and enable computers to share and integrate their computing resources, data and services. Although concepts of P2P and Grids have a significant amount of overlap, they were originally proposed to address different domains. Whereas P2P is generally applied to a wide range of technologies that can greatly increase the utilization of collective natural resources at the edge of the Internet, such as information, bandwidth and computing resources, Grids are intended to promote interoperability and extensibility among various applications, platforms and frameworks [4].

In contrast to the existing Grid paradigms, P2P architecture does not rely on a centralised server to provide services (Fig. 2), which offers an appealing alternative to the existing Grid models especially for large-scale distributed applications. In the P2P model, each peer node (also known as servent) acts as both client and server, requesting resources from as well as routing queries and serving resources for other peer nodes. A P2P network is a logical overlay network over a physical infrastructure as illustrated in Fig. 2, which provides a virtual environment for P2P

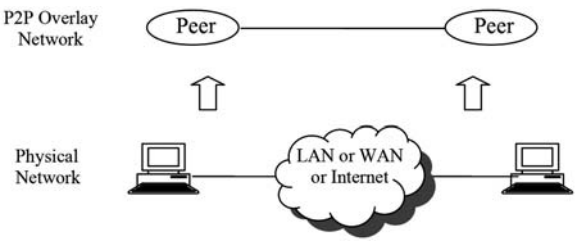

Fig. 2 Peer-to-peer architecture

developers to easily design and implement their own communication environment and protocols on the top of existing networks.

In recent years P2P networks have grown seemingly exponentially in popularity and utilisation. P2P file sharing has become one of the most popular Internet activities. Today's popular P2P file-sharing applications, such as Kazaa and Gnutella, have more than one million users each at any point of time [5]. According to research results from Cachelogic [6], about 50–65% of "downstream traffic" (i.e. from ISPs to endpoint devices) and 75–90% of "upstream traffic" (i.e. from endpoint devices to ISPs) on the Internet are the results of P2P file-sharing applications.

The popularity of P2P is motivated by the benefits it offers to end users. Compared to the client-server architecture, the advantages of P2P are listed below:

- P2P frees users from the traditional dependence on central servers, which enables end users to easily share resources (e.g. music, movies, games and other software). End users can share or retrieve resources directly from their connected machines without any further need to upload them to a centralised server.
- P2P applications are more resilient than those built on the client-server architecture by removing the single-point-of-failure.
- P2P distributes the responsibility of providing services from centralised servers to each individual peer node in the network.
- P2P exploits available bandwidth, processor, storage and other resources across the entire network. P2P interactions are only between individual peers which eliminate the bottleneck of centralised servers.
- Most P2P applications use virtual channels for communication which break the obstacles of corporate private networks, such as firewalls and Network Address Translation (NAT).
- P2P has better availability as each peer node can obtain content from multiple peer nodes. If one peer node is overloaded or experiences a hardware failure, other peer nodes in the network can still handle requests.

3 Evolution of Peer-to-Peer Networks

There are many interesting types of P2P applications, including file sharing, instant messaging, VoIP, Streaming media, High Performance Computing, search engine. Among them, file sharing, one of the most popular on-line activities [7], is the initial motivation behind many of successful P2P networks. P2P file sharing has become one of the most popular Internet activities. Today's popular P2P file-sharing applications, such as Kazaa and Gnutella, have more than one million users each at any point of time [5]. In this section, the history of the P2P file sharing networks is discussed along with the most popular file sharing applications.

Existing P2P file sharing networks can be divided into three categories [8] according to the degree of network centralisation: centralised P2P networks, decentralised P2P networks and hybrid P2P networks.

3.1 Centralised Peer-to-Peer Networks

Although P2P is often seen as an opposite model to the centralised client-server paradigm, the first generation P2P systems (e.g. Napster) started with the concept of centralisation. However, in contrast to traditional client-server systems, the server(s) in centralised P2P networks only keeps the meta-information about shared content (e.g. addresses or ID of peer nodes where the shared content is available) rather than storing content on its own.

- **Napster**

Napster was the first widely-used P2P music sharing service. Before Napster came along, Internet users only passively operated their connected computers, such as browsing news or checking email. With the increased popularity of Napster, ordinary Internet users started opening their PCs to actively contribute resources and played more important roles for the Internet.

Compared to follow-up P2P applications, Napster utilises a simple but highly efficient mechanism to share and search files in the network. To participate in the Napster network, new users need to register to the Napster server and publish a list of files they are willing to share. To search for a shared file in the network, users can request the Napster server and retrieve a list of providers hosting the files which match the query. File transfer takes place without the Napster server participating. The requested file is transferred directly between the requester and the provider as shown in Fig. 3.

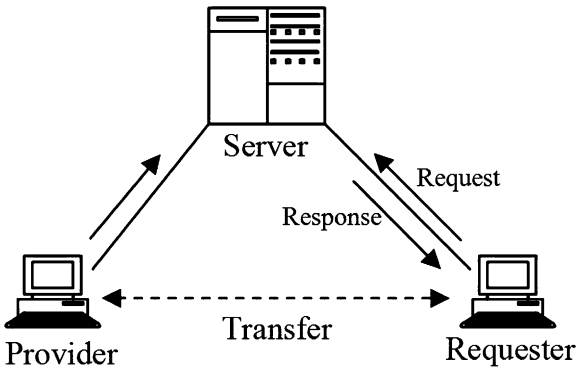

Fig. 3 Example of Napster network

- **BitTorrent**

BitTorrent is designed to distribute large amounts of data without incurring the corresponding consumption in server and bandwidth resources. The original BitTorrent (before version 4.2.0) can be looked at as a Napster-like centralised P2P system. In order to share a file or a group of files, users need to create a small

.torrent file that contains the address of the tracker machine that launches the file distribution. The .torrent file is published on well-known web-sites, so that other users can find and download the .torrent file of interest using web search engines. The .torrent file is opened by the BitTorrent client software. The client software connects to the tracker machine and receives a list of peer nodes that are participating in transferring the file. In order to distribute a file efficiently, a file is broken into smaller fragments (typically 256 KB each) for transmission. The client, attempting to download the file, simultaneously connects to these peer nodes that are participating in file transfer, and downloads different pieces of the file from different peer nodes. In the meantime, the client can also upload downloaded pieces to other participants.

3.2 Decentralised Peer-to-Peer Networks

To address the problems of centralised P2P networks (such as scalability, single-point-of-failure and legal issues), decentralised peer-to-peer networks become widely used, which do not rely on any central server.

- **Gnutella**

The Gnutella network is a decentralised file-sharing P2P network, which is built on an open protocol developed to enable peer node discovery, distributed search, and file transfer.

Each Gnutella user needs a Gnutella client software to join the Gnutella network. The Gnutella client software on initial use can bootstrap and find a number of possible working peer nodes in the network and try to connect to them. If some attempts succeed, these working peer nodes will then become the new node's neighbours and give the new node their own lists of working nodes. The new node continues to connect to these working peer nodes, until it reaches a certain quota (usually user-specified). The new node keeps the peer nodes it has not yet tried as backup. When a peer node leaves the P2P network and then wants to re-connect to the network again, the peer node will try to connect to the nodes whose addresses it has stored. Once the peer node re-connects into the network, it will periodically ping the network connections and update its list of node addresses.

In contrast to Napster, the Gnutella network is a decentralised P2P file-sharing network not only for file storage, but also for content lookup and query routing. Gnutella nodes take over routing functionalities initially performed by the Napster server. Figure gives an example of query propagation over the Gnutella network. In the Gnutella network, each peer node uses a Breadth-First Search (BFS) mechanism to search the network by broadcasting the query with a Time-to-Live (TTL) to all connected peer nodes. TTL represents the number of times a message can be forwarded before it is discarded. Each peer node receiving the query will process it, check the local file storage, and respond to the query if at least one matched file is found. Each peer node then decreases TTL by one and

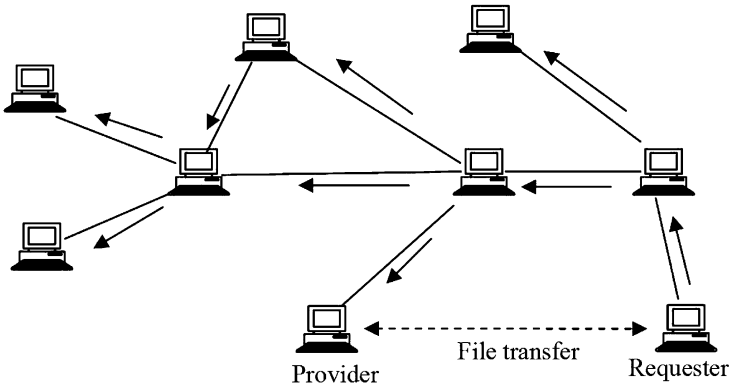

Fig. 4 Query propagation over the Gnutella network

forwards the query to all of its neighbours. This process continues until TTL decreases to zero.

The Gnutella network does not rely on a central server to index files, which avoids the single-point-of-failure issue and the performance bottleneck at the server side. Instead, many peer nodes are visited by flooding queries to see whether they have a requested file. The obvious drawback of Gnutella is that it generates potentially huge network traffic by flooding queries.

- **Freenet**

Freenet is a decentralised P2P data storage system designed to provide electronic document exchange through strong anonymity. In contrast to Gnutella, Freenet acts as a P2P storage system by enabling users to share unused local storage space for popular file replication and caching. The stored information is encrypted and replicated across the participating computers.

In Freenet, a file is shared with an ID generated from the hash value of the name and description of the file. Each peer node forms a dynamic routing table to avoid network flooding. A routing table includes a set of other peers associated with the keys they are expected to hold. To search a required file, the query is forwarded to the peer node holding the nearest key to the key requested. If the query is successful, the reply is passed back along the route the query comes in through. Each peer node that forwards the request will cache the reply and update the routing table by a new entry associating the data source with the requested key.

3.3 Hybrid Peer-to-Peer Networks

To avoid the observed problems of the centralised and decentralised P2P networks discussed above, hybrid P2P networks are emerging recently to provide trade-off solutions with a hierarchical architecture.

- **Kazaa**

Kazaa reorganises peer nodes into a two-level hierarchy with supernodes and leaves. Supernodes are capable and reliable peer nodes that take more responsibility for providing services in the network. A supernode is a temporary index server for other peer nodes. The peer nodes with high computing power and fast network connection automatically become supernodes.

Similarly to the bootstrapping method used in the Gnutella network, a newly joined node will attempt to contact an active supernode from a list of supernodes offered by Kazaa client software. The newly joined node will send a list of files it shares to the connected supernodes and further retrieve more active supernodes from the connected supernodes for future connection attempts.

In Kazaa, each leave node begins a lookup by sending a lookup request to its connected supernode as shown in Fig. 5. The supernode not only checks the local index for the file requested, but also communicates with other supernodes for a list of addresses of peer nodes sharing the files. When a supernode discovers the requested file from its local index, it will respond to the original supernode. The file is transferred directly between the query originator and the target peer node that shares the file as shown in Fig. 5.

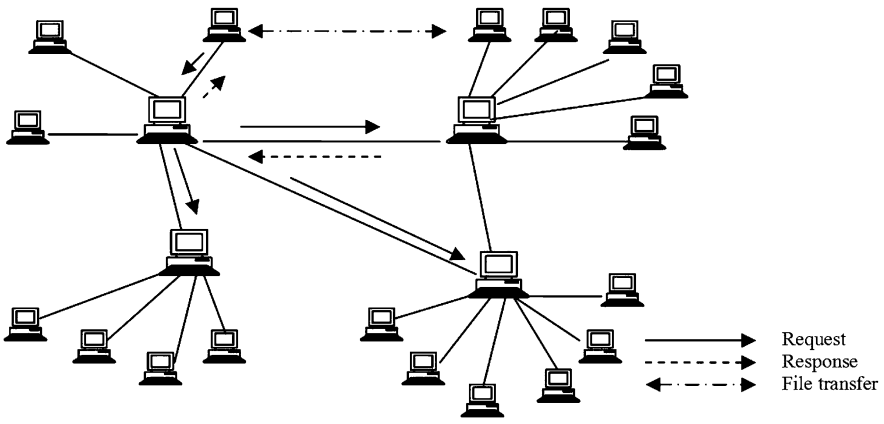

Fig. 5 Example of Kazaa network

- **Gnutella2**

Similarly to Kazaa, the peer nodes in the Gnutella2 network are classified into two categories: hubs and leaves. A leaf keeps only one or two connections to hubs. A hub acts as a proxy to the Gnutella2 network for the leaves connected to it. Queries are propagated among the hubs and only forwarded to a leaf if a hub believes it can answer the query.

- **JXTA**

JXTA is an open source P2P platform developed by Sun Micro-systems. The JXTA Application Programming Interface (API) hides many programming details, which makes a JXTA application writing much easier than developing a P2P application from scratch.

Similarly to Kazaa and Gnutella2, JXTA maintains a hierarchical network structure with rendezvous peers and edge peers. Different from Kazaa, the rendezvous peers in the JXTA network call the Shared Resource Distributed Index (SRDI) service to distribute indices to other rendezvous peers in the network. When a peer node searches for a file, it will send the query to the connected rendezvous peer and also multicast the query to other peers on the same subnet. If the rendezvous peer finds the information about the requested resources on its local cache, it will notify the peers that publish the resources and these peers will respond directly to the query originator. If the rendezvous peer cannot find the requested information locally, a default algorithm is used to go through a set of rendezvous peers for a rendezvous peer that caches the requested information [9].

As discussed above, hybrid P2P networks combine the techniques of both the centralised Napster and the decentralised Gnutella. However, since only a limited number of peer nodes are responsible for the query processing and routing, existing hybrid P2P networks still have the capability bottlenecks of the supernodes, which are also vulnerable to planned attacks [10].

4 Peer-to-Peer Search Systems

Since resource and service discovery in Grids involves a lot of elements in common with resource discovery in P2P networks, P2P search approaches are applicable for service discovery in large-scale Grid systems, which could help to ensure Grid scalability [11]. In this section, existing P2P search systems are investigated by classifying them into two broad categories: structured and unstructured P2P systems.

4.1 Structured P2P Systems

Structured P2P systems have a dedicated network structure on the overlay network which establishes a link between stored content and the IP address of a node. Distributed Hash Tables (DHTs) are widely used for resource discovery in the structured P2P systems like Chord [12], ROME [13], Pastry [14], CAN [15], and Kademlia [16].

In DHT-based P2P systems, each file is associated with a key generated by hashing the file name or content. Each peer node in these systems is responsible for

storing a certain range of keys. The network structure is sorted by routing tables (or finger tables) stored on individual peer nodes. Each peer node only needs a small amount of "routing" information about other nodes (e.g. nodes' addresses and the range of keys the node is responsible for). With routing tables and uniform hash functions, peer nodes can conveniently put and get files to and from other peer nodes according to the keys of files.

- **Chord**

Chord [12] is a well-known DHT-based distributed protocol aimed to efficiently locate the peer node that stores a particular data item. Peer nodes are arranged in a ring that keeps the keys ranging from zero to $2^m - 1$. A consistent hashing is used to assign items to nodes, which provides load balancing and only requires a small number of keys to move when nodes join or leave the network [12]. The consistent hash function assigns each node and each key an ID using SHA-1.

In Chord, each peer node maintains a finger table pointing to $O(\log N)$ other nodes on the ring. Given a ring with 2^m peer nodes, a finger table has a maximum of m entries. The Chord routing algorithm utilizes the information stored in the finger table of each node to direct query propagation. For example, a node sends a query for a given key k to the closest predecessor of k on the Chord ring according to its finger table, and then asks the predecessor for the node it knows whose ID is the closest to k. By repeating this process, the algorithm can find the peer nodes with IDs closer and closer to k. A lookup only requires $O(logN)$ messages in a N-node Chord network and $\frac{1}{2}\log_2 N$ hops on average [12].

Unlike some other P2P models (e.g. Gnutella and JXTA) that provide a set of protocols to support P2P applications, Chord provides support for just one operation: given a key, it maps the key onto a node. In Chord, peer nodes are automatically allowed to participate in the network using the standard Chord protocol, no matter whether the nodes are useful and capable or not. Chord needs monitoring and selection functions in order to support and optimise its deployment over the real networks.

- **ROME**

ROME (Reactive Overlay Monitoring and Expansion) [13, 17] is an additional layer built upon the standard Chord protocol allowing control over the size of the network overlay via the selection and placement of peer nodes on the Chord ring.

Figure [18] shows the ROME architecture running on the top of Chord. ROME includes a set of processes (ellipses) and data structures (rectangles) [18]. Processes are comprised of a traffic analyser to monitor network traffic without changing the underlying Chord protocol, and operations to add, replace and remove nodes from the ring. Data structures of ROME store monitoring data, a copy of Chord data and ROME specific data (e.g. bootstrap server's address). ROME can provide an optimal size of Chord ring by monitoring the workload on each node to solve the problems of under-load or over-load nodes by adding, replacing and removing nodes. ROME provides more efficient and fault-tolerant resource discovery with message cost saving than the standard Chord [18].

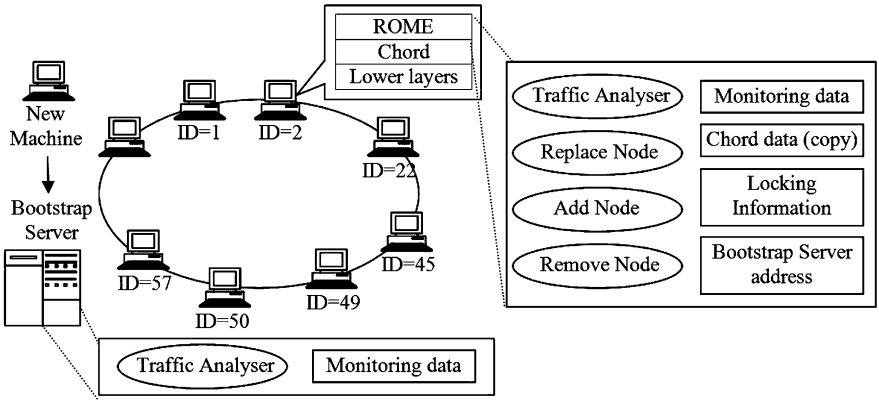

Fig. 6 ROME architecture

- **Accordion**

Accordion [19] is a DHT protocol extended from Chord, which bounds its communication overhead according to a user specified bandwidth budget. Accordion borrows Chord's protocols for maintaining a linked list in which the ID space is organized as a ring as in Chord. Different from Chord using a fixed routing table, Accordion can automatically adapt itself to achieve the best lookup latency across a wide range of network sizes and churn rates. Accordion maintains a large routing table when the system is small and relatively stable. When the system grows too large or suffers from high churns, Accordion shrinks its routing table on each peer node for lower communication overhead. By remaining flexible in the choice of routing table size, Accordion can operate efficiently in a wide range of operating environments.

- **Pastry**

Pastry [14] is a prefix-based routing system using a proximity metric. Similarly to Chord, Pastry organizes peer nodes in a 128-bit circular node ID space. At each step, a query message is forwarded to a numerically closer node to a given key. In a network consisting of N nodes, the message can be routed to the numerically closest node within $\log_{2^b} N$ hops, where b is a configurable parameter.

In contrast to Chord which uses one-dimensional tables, a node's routing table is organized into two dimensions with $\log_{2^b} N$ rows and $2^b - 1$ entries per row. Each entry in row n of the routing table points to a node whose node ID shares the present node's ID in the first n digits, but whose $(n+1)th$ digit is different from the $(n+1)th$ digit in the present node's ID. The routing procedure involves two main steps. Given a message, the node first checks whether the key is within the range of its leaf set. If so, the message is sent directly to the destination node. Otherwise, the message is forwarded to the node that shares a common prefix with the key by at least one more digit.

- **CAN**

Content-Addressable Network (CAN) [15] generalizes the DHT methods used in Chord and Pastry. A CAN identifier space can be looked at as a d-dimensional version of Chord (which is one-dimensional) and Pastry (which is two-dimensional) identifier space. For a d-dimensional space partitioned into n equal zones, each node maintains $2d$ neighbours and the average routing path length is $\frac{1}{4}(n^{\frac{1}{d}})$. Higher dimensions in the identifier space reduce the number of routing hops and only slightly increase the size of routing table saved in each node. Meanwhile, fault tolerance of routing is improved by higher dimensionality of CAN, since each node has a larger set of neighbours to select as alternatives to a failed node.

- **Kademlia**

Kademlia [16] not only uses the basic DHT methods (e.g. unique ID, routing table with $< key, value >$ pairs), but also provides a number of desirable features superseding other previous DHTs (e.g. Chord, CAN and Pastry).

Kademlia is based on the calculation of the "distance" between two nodes on the overlay with an XOR metric. Each Kademlia node stores contact information about other peer nodes in the local routing tables. When a Kademlia node receives a message from another node, it will update the appropriate entry for the sender's node ID. Thus, the peer nodes which issue or reply to a large number of queries will become widely known, which enables more capable nodes to take on more workload in the network.

To lookup a specific key, the query originator searches the local routing tables for α nodes with the closest distance to the query originator and then contacts them in parallel. Each recipient node replies with the information about the peer node which is closer to the key. The query originator resends the lookup to nodes it has learned about from previous RPCs.

4.2 Unstructured P2P Systems

In contrast to structured P2P systems, unstructured P2P systems do not maintain network structure, where address and content stored on a given peer node are unrelated. Although existing search methods in unstructured P2P systems are heterogeneous, most of them are dedicated to solving observed issues of flooding mechanisms which can be classified into two broad categories: blind search and informed search, according to whether the algorithm needs additional indices about the locations of resources.

4.2.1 Blind Search

In a network with a blind search method, peer nodes do not maintain additional information about resource locations. The advantages of blind search methods are

that they do not need any communication overhead to maintain the additional indices about resource locations and are extremely resilient in the highly dynamic networks. Flooding is the fundamental approach of blind search where queries are forwarded to all connected peer node to see whether they have a requested file. In order to solve the massive traffic problem caused by flooding, several improved search methods have been presented recently [20–22].

Random walker [20, 21] is a well-known blind search method, which enables peer nodes to forward a query to a randomly chosen neighbour rather than broadcast the query to all of their neighbours. Each neighbour repeats this process until the required file is discovered. The random walkers can find targets more efficiently while significantly reducing the traffic compared to Gnutella's flooding method [20]. In order to increase the probability of resource discovery, pro-active replications are used to increase the density of copies of each object. The study in [20] shows that the square-root replication distribution is optimal in terms of minimizing overall search traffic, which replicates files in proportion to the square-root of their query probability.

Yang and Garcia-Molina [22] presented an iterative deepening technique for resource discovery in unstructured P2P networks. Iterative deepening enables query originators to use successive BFS (Breadth-First Search) queries with increasing depths, until the request is satisfied or the maximum depth is reached.

4.2.2 Informed Search

In contrast to blind search, informed search methods enable peer nodes to maintain additional information about other peer nodes in the network, e.g. network topology and resource locations. Existing informed search solutions can be classified in two groups: mechanisms with specialised index nodes and mechanisms with indices at each node.

The search mechanisms with specialised index nodes have been used in current P2P applications. For example, Napster employs a centralized server to maintain such additional information. Kazaa and JXTA utilize a set of semi-centralized supernodes to maintain the extra information about their leaves and other supernodes.

However, systems with specialised index nodes are vulnerable to attack by centralizing indices in a small subset of peer nodes. New methods have emerged in recent years by distributing indices to each individual peer node in the network. Such methods can be further classified into the following three approaches according to their design principles.

- **Topology optimisation**

In many P2P applications, topology significantly affects the performance of resource discovery. The first approach intends to reduce search cost by adapting and optimising the overlay topology of network. Each peer node is expected to keep topology information about its neighbours or neighbours' neighbours.

A topology optimisation method used in Gia [23] puts most of the peer nodes in the network within a short reach of high capacity nodes, which leads to the situation

that the high capacity nodes are also the nodes with a high node degree of connectivity (a large number of links). This protocol ensures that the well-connected nodes which receive a large proportion of queries actually have the capability to handle these queries.

Gia enables each peer node to connect high capacity nodes as neighbours. Alternatively, the studies in [24, 25] present topology optimisation methods by preferentially selecting low-cost and low-latency connections. These methods address the topology mismatch problem between P2P logical overlay networks and physical under-lying networks (which incurs a large volume of redundant traffic on P2P networks) by deleting inefficient overlay links and adding efficient ones.

- **Statistical information about neighbours**

The second approach enables peer nodes to route queries to the neighbours that are likely to have the requested files in accordance with the maintained statistical information about neighbours.

Tsoumakos et al. [26] introduced an adaptive and bandwidth-efficient algorithm for searching in unstructured P2P networks, called Adaptive Probabilistic Search (APS). Different from the above topology optimisation methods, the search algorithm of APS is not allowed to alter the overlay topology. In APS, each peer node keeps an index describing which objects were requested by each neighbour with a success ratio of previous searches. The probability of choosing a neighbour to find a particular file is dependent on the success ratio. In APS, searching is based on the simultaneous deployment of k walkers and probabilistic forwarding. The query originator sends queries to k of its N neighbours. If each of these nodes can find a matched object in its local repository, the walker terminates, otherwise it gets forwarded to one of neighbours. The procedure continues until all k walkers have completed. The update takes a reverse path back to the query originator to adjust probability accordingly either with success or failure. According to the results shown in [26], APS can discover many more objects with much higher success rates than random walkers [20, 21] algorithm.

Adamic et al. [21] showed that many peer nodes can be reached by forwarding queries to peer nodes with the highest degree (number of links) in the neighbourhood and thus it can get many results back. Yang and Garcia-Molina [22] tested this model by forwarding queries to the peer node which had the largest number of neighbours. Yang and Garcia-Molina also designed and simulated several other heuristics to help in selecting the best peer nodes to send a query, for example, the peer node that returned the highest number of results from previous searches or the peer node that had the shortest latency.

The study in [27] evaluated a similar heuristic that the peer nodes with more files are more likely to be able to answer the query, called Most File Shared on Neighbourhood (MFSN) which forwards the query to the neighbouring node sharing the largest number of files. In accordance with these previous studies above, these heuristic-based methods can generally achieve better performance than random walkers.

In contrast to the above studies using only one heuristic parameter, DSearch method in [28] puts two heuristics in consideration. In order to achieve a high

success rate and efficiency, queries are forwarded to the "good" nodes that are more likely to have the requested files or can find the requested files with a high probability in future hops. Two heuristics are considered to determine the "goodness" of a neighbour: the number of shared files on the neighbour and the expected number of accessible files in future hops via the neighbour. DSearch utilizes high-degree nodes to find a wider range of accessible nodes, while using the neighbouring nodes that shares a large number of files to discover the requested files. From the simulation results, DSearch achieves a better performance when compared to the methods with only one heuristic.

The principle behind these heuristic-based methods is to forward queries to more capable peer nodes, for example, the peer nodes with the highest success rate of searches, the peer nodes with the highest degree, or the peer nodes with most shared files. However, such capable nodes may be over-loaded and become the victims of their own success. Traffic unbalance is a significant limitation to these methods.

- **Cached semantic information**

In contrast to the second approach of maintaining simple statistical information, the third approach enables each peer node to keep a routing index which contains detailed semantic information about content of shared files. This information can be collected by exchanging indices regularly with other peer nodes or caching the historic record regarding the results of previous queries.

Routing Indices

Routing Indices (RI) [29] enable peer nodes to create query routing tables by hashing file keywords and regularly exchanging those with their neighbours. Peer nodes normally maintain additional indices of files offered by their overlay neighbours and neighbours' neighbours. In the RI system, shared documents are classified into different categories of topics. Each peer node maintains the RI indicating how many documents of which categories could be found through that neighbour. When a new connection is established in the RI system, two peer nodes will aggregate and exchange their own RI to each other. In order to keep data in RI up-to-date, peer nodes will notify neighbours about the changes in the local RI caused by addition/removal of shared documents or joining/leaving of other neighbours.

When a peer node receives a query, the peer node needs to compute the goodness of each neighbour for a query. The number of documents that may be found in a path is used as a measure of goodness. If more documents are found through a particular neighbour, this neighbour will be selected to forward a query.

Sripanidkulchai's model

The idea of constructing "friend lists" has also been used in the Sripanidkulchai's model [30], which presents a content location solution in which peer nodes loosely organize themselves into an interest-based structure. When a node joins the system, it first searches the network by flooding to locate content. The lookup returns a set of nodes that store the content. These nodes are potential candidates to be added to a "shortcut list ". One node is selected at random from the set and added. Subsequent

queries will go through the nodes in the shortcut list. If a peer node cannot find content from the list, it will generate a lookup with Gnutella protocol. From their results, a significant amount of flooding can be avoided which makes Gnutella a more competitive solution.

NeuroGrid

NeuroGrid [31] is an adaptive decentralized search system. NeuroGrid utilizes the historic record of previous searches to help peer nodes make routing decisions. In NeuroGrid, peer nodes support distributed searches through semantic routing by maintaining routing tables at each node [31]. In the local routing tables, each peer node is associated with keywords regarding the content it stores. When a peer node forwards a query, it will search for the peer nodes that are associated with the query keywords. In each hop, NeuroGrid intends to find M matched peer nodes and then forwards the query to these peer nodes. If there are not M matches found, the algorithm will randomly select peer nodes in the routing table until M peer nodes are selected.

In NeuroGrid, users' responses to search results are stored and used to update the meta-data describing the content of remote peer nodes. NeuroGrid can learn the results from previous searches to make future searches more focused and semantically routes queries according to the cached knowledge. However, NeuroGrid is only effective for previously queried keywords and not suitable for networks where peer nodes come and go rapidly [32].

5 Future Trends

Some potential future trends are outlined in this section.

5.1 Self-organising Systems

P2P is beneficial when removing a centralised server. On the other hand, new mechanisms are required to compensate for the server, especially for resource discovery and network maintenance. However, owing to the lack of a centralised server, any attempts of additional control could be difficult to achieve in the distributed P2P architecture. In contrast, self-organisation could be a good way to solve the control issues in the decentralised P2P architecture. Self-organisation is a process where the organisation of a system spontaneously increases without being managed by an outside source.

Human society is a self-organising system. Social networks are formed naturally by daily social interactions. A social community is a group of people with common interests, goals or responsibilities which is formed spontaneously. By using social networks, people can find some acquaintances that potentially have knowledge about the resources they are looking for.

Similarly to social networks, where people are connected by their social relationships, two autonomous peer nodes can be connected if users in those nodes are interested in each other's data. If peer nodes can self-organise themselves like social networks, a large communication cost for peer group construction, maintenance and discovery can be saved, which will significantly improve overall performance of P2P networks.

5.2 Hybrid Systems

Since large-scale resource sharing is one goal of Grids, P2P networks and Grid systems are starting to converge to a common structure, leading to application of P2P techniques to Grid systems [33]. Some research has been done to try connecting Grid Services and P2P networks (e.g., [4, 31, 32]). Unfortunately, there is still not a good solution to seamlessly pull the two domains together. There are many boundaries to connect Grids and P2P networks because of different architectures, standards and protocols between them. Current Grid service standards need to be extended, especially for service descriptions and annotations, by including additional attributes for peer's specifications.

6 Conclusions

In this chapter, the evolution of the network architecture has been investigated from client-server to P2P networking. P2P is beneficial when removing a centralised server. On the other hand, new mechanisms are required to compensate for the server. The main advantage of P2P architecture lies in its good scalability, agility, resilience and availability. On the contrary, its major challenges lie in its efficiency, dependability and security.

To address these challenges, hybrid systems combining the techniques of both Grid and P2P computing could be potential solutions for the design of next generation distributed systems. By introducing P2P techniques into Grid computing, Grid systems could be more scalable and resilient, removing the sing-point-of-failure. With the cooperation of Grid computing environments, the usage of P2P networks could be also broadened from simple file provision to more advanced services, such as sharing redundant computing power for complicated scientific calculation and sharing extra bandwidth for real-time video transmission.

References

1. Vance, A.: Google goes gaga for Opteron. Available from: `http://www.theregister.co.uk/2006/03/04/goog_opteron_sun/`
2. Foster, I., Kesselman, C. (eds.): The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (1999)

3. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of Supercomputer Applications 15 (2001) 200–222
4. Qu, C., Nejdl, W.: Interacting the Edutella/JXTA Peer-to-Peer Network with Web Services. Second International Conference on Peer-to-Peer Computing (2002)
5. Zhao, S., Stutzbach, D., Rejaie, R.: Characterizing Files in the Modern Gnutella Network: a Measurement Study. SPIE/ACM. Multimedia Computing and Networking, San Jose, CA (2006)
6. Cachelogic.: Trends and Statistics in Peer-to-Peer. (2005) Available from: `http://creativecommons.nl/nieuws/wp-content/uploads/2006/04/CacheLogic_AmsterdamWorkshop_Presentation_v1.0.ppt`.
7. Oberholzer, F., Strumpf, K.: The Effect of File Sharing on Record Sales: An Empirical Analysis. Journal of Political Economy 115 (2006) 1–42
8. Risson, J., Moors, T.: Survey of Research towards Robust Peer-to-Peer Networks: Search Methods. Computer Networks 50 (2006) 3485–3521
9. Traversat, B., Abdelaziz, M., Pouyoul, E.: Project JXTA: A Loosely-Consistent DHT Rendezvous Walker. Sun Microsystems, Inc (2003)
10. Lo, V., Zhou, D., Liu, Y., GauthierDickey, C., Li, J.: Scalable Supernode Selection in Peer-to-Peer Overlay Networks. Second International Workshop on Hot Topics in Peer-to-Peer Systems, La Jolla, California (2005)
11. Mastroianni, C., Talia, D., Verta, O.: A P2P Approach for Membership Management and Resource Discovery in Grids. 2005 International Symposium on Information Technology: Coding and Computing, Las Vegas, NV (2005)
12. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. ACM SIGCOMM, San Diego, CA (2001)
13. Salter, J., Antonopoulos, N.: ROME: Optimising DHT-based Peer-to-Peer Networks. 5th International Network Conference, Samos, Greece (2005)
14. Rowstron, A., Druschel, P.: Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems. IFIP/ACM International Conference on Distributed Systems Platforms, Heidelberg, Germany (2001)
15. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. ACM SIGCOMM, San Diego, CA (2001)
16. Maymounkov, P., Mazi'eres, D.: Kademlia: A Peer to Peer Information System Based on the XOR Metric. Internation Workshop on Peer-to-Peer Systems, Cambridge, MA (2002)
17. Antonopoulos, N., Exarchakos, G.: G-ROME: A Semantic Driven Model for Capacity Sharing Among P2P Networks. Journal of Internet Research 17 (2007) 7–20
18. Salter, J.: An Efficient Reactive Model for Resource Discovery in DHT-Based Peer-to-peer Networks. Department of Computing, Vol. PhD. University of Surrey, Guildford, Surrey (2006)
19. Li, J., Stribling, J., Morris, R., Kaashoek, M.F.: Bandwidth-Efficient Management of DHT Routing Tables. 2nd Symposium on Networked Systems Design and Implementation, Boston, MA (2005)
20. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and Replication in Unstructured Peer-to-Peer Networks. ACM SIGMETRICS, Marina Del Rey, CA (2002)
21. Adamic, L.A., Lukose, R.M., Puniyani, A.R., Huberman, B.A.: Search in Power Law Networks. Physical Review 64 (2001) 046135-046131–046135-046138
22. Yang, B., Garcia-Molina, H.: Efficient Search in Peer-to-Peer Networks. International Conference on Distributed Computing Systems, Vienna, Austria (2002)
23. Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., Shenker, S.: Making Gnutella-Like P2P System Scalable. ACM SIGCOMM, Karlsruhe, Germany (2003)
24. Xiao, L., Liu, Y., Ni, L.M.: Improving Unstructured Peer-to-Peer Systems by Adaptive Connection Establishment. IEEE Transactions on Computers 54 (2005) 176–184
25. Liu, Y., Xiao, L., Liu, X., Ni, L.M., Zhang, X.: Location Awareness in Unstructured Peer-to-Peer Systems. IEEE Transactions on Parallel and Distributed Systems 16 (2005) 163–174

26. Tsoumakos, D., Roussopoulos, N.: Adaptive Probabilistic Search for Peer-to-Peer Networks. Third International Conference on Peer-to-Peer Computing, Linkoping, Sweden (2003)
27. Li, X., Wu, J.: A Hybrid Searching Scheme in Unstructured P2P Networks. International Conference on Parallel Processing, Oslo, Norway (2005)
28. Liu, L., Antonopoulos, N., Mackin, S.: Directed Information Search and Retrieval over Unstructured Peer-to-Peer Networks. the International Computer Engineering Conference, Cairo, Egypt (2006)
29. Crespo, A., Garcia-Molina, H.: Routing Indices for Peer-to-Peer Systems. International Conference on Distributed Computing Systems, Vienna, Austria (2002)
30. Sripanidkulchai, K., Maggs, B., Zhang, H.: Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. IEEE Infocom, San Francisco (2003)
31. Joseph, S.: NeuroGrid: Semantically Routing Queries in Peer-to-Peer Networks. International Workshop on Peer-to-Peer Computing, Pisa, Italy (2002)
32. Joseph, S.: P2P MetaData Search Layers. International Workshop on Agents and Peer-to-Peer Computing, Melbourne, Australia (2003)
33. Marzolla, M., Mordacchini, M., Orlando, S.: Peer-to-peer Systems for Discovering Resources in a Dynamic Gridstar, Open. Parallel Computing 33 (2007) 339–358
34. Wang, M., Fox, G., Pallickara, S.: A Demonstration of Collaborative Web Services and Peer-to-Peer Grids. International Conference on Information Technology: Coding and Computing (2004)
35. Prasad, V., Lee, Y.: A Scalable Infrastructure for Peer-to-Peer Networks Using Web Service Registries and Intelligent Peer Locators. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (2003)