CHAPTER

8 Peer-to-Peer Computing and Overlay Networks

CHAPTER OUTLINE

© 2012 Elsevier, Inc. All rights reserved.

SUMMARY

In this chapter, we study *peer-to-peer* (P2P) networks and their applications. P2P overlay networks are built over a large number of client computers through virtual networking over the Internet. We present unstructured, structured, and hybrid overlay networks. *Distributed hash tables* (DHTs) are studied for fast routing and locality preservation operations. Fault tolerance and churn resilience are studied to improve dependability. On the application side, P2P networks are used for distributed file sharing and content distribution applications. Trust management and copyright protection are studied with reputation systems and content poisoning techniques. Examples are given to illustrate the design principles of P2P networks and the protocols applied in real-life applications. We will cover P2P social networks in Chapter 9.

8.1 PEER-TO-PEER COMPUTING SYSTEMS

P2P computing systems are widely used in distributed file sharing, messaging, online chatting, video streaming, and social networking. It has been estimated that P2P applications on the Internet accounted for about 40 percent to 70 percent of the Internet traffic worldwide from 2008 through 2010 and the trend is increasing in light of the popularity of online applications such as Facebook, Twitter, and YouTube. Unlike a traditional distributed system, P2P networks are autonomous and self-organized with free participation from the peers or clients. In a P2P network, peers share computing and data resources with others. All peers offer a mixture of online services on a voluntary basis. There are economical advantages as well as legal problems in today's P2P networks and their open applications.

As shown in Figure 8.1, Internet traffic due to e-mail is quite flat and getting lower, compared with other traffic components. FTP traffic has also declined over the past 15 years. The major traffic surge on the Internet has really been in P2P requests and responses. Even web traffic is declining with the rise of social networking and digital content downloads. By 2007, 70 percent of Internet traffic was attributed to P2P applications. This is our motivation for studying P2P systems in this chapter.

8.1.1 Basic Concepts of P2P Computing Systems

Communication involving two equal role users can be treated as P2P communication; as such, it can be said that earlier distributed systems also followed a P2P model. Basically, P2P computing is a technique that takes advantage of computational resources (such as storage, CPU cycles, and bandwidth) and content resources (such as content files) available at the edges of the Internet to complete huge tasks, such as distribution of content to a large audience and distributed search engine and CPU-bound computing tasks.

The resources at an edge network have intermittent availability, as they are added and removed unexpectedly all the time. P2P computing eliminates the coordination by a central server, and no peer has a global view of the entire system. Instead, peers act as both servers and clients, sharing resources and services directly with one another. P2P networks share a number of common characteristics, which are summarized in the following list.

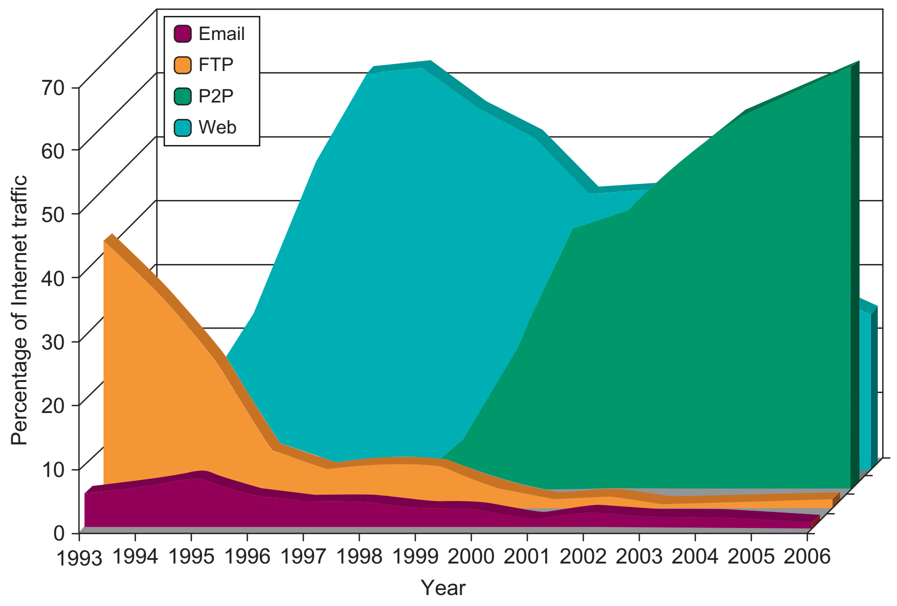


FIGURE 8.1

The distribution of Internet traffic based on application trends.

(*Courtesy of CacheLogic Research, 2007*)

- **Decentralization** In a pure P2P computing system, peers are equal and there is no central server to coordinate the system. Each peer has a partial view of the whole system to construct the overlay network, and controls its data and resources.
- **Self-organization** Self-organization means no central management point is required to organize the peer computing resources and the data resources spreading over the peers. The resources in P2P computing systems are dynamic or fluctuate in the sense that they come and go at any time, at will. It would be very costly to dedicate a server to manage these systems.
- **Ad hoc connectivity and dynamics** The availability of peers is unpredictable. A peer may join or leave the system at any time. The overlay topologies and the system scales will vary enormously and rapidly. P2P computing systems should provide stable services even in extra dynamic environments.
- **Anonymity** In a decentralized P2P network, peers resort to detour paths to send/receive requests (i.e., two nodes communicate through some middle nodes). This ensures senders' anonymity. Anonymity can be also achieved through hashing.
- **Scalability** The P2P model eliminates the single point of failure problem related to the traditional centralized client/server model. Each peer only maintains a limited number of system states and shares resources directly with others. These features enable the high scalability of P2P computing systems.
- **Fault tolerance** In a P2P network, all peers are created equal and no peer can dominate the system or become a single point of failure to pull down the entire system. Resources could be stored in multiple peers. These features facilitate fault tolerance.

8.1.1.1 Distinction between Client/Server and P2P Architectures

The traditional client/server architecture is formed with many client hosts surrounding a server node; see Figure 8.2(a). A P2P computing system is formed with peer hosts that are fully distributed without using a centralized server; see Figure 8.2(b). In other words, the client/server architecture is server-oriented. The server splits a task into many parts and dispatches the parts to the clients. The clients complete the task assigned to them independently and seldom communicate with one another. The clients require resources from the server and the server delivers the required resources to the clients. On the contrary, in a P2P network, participating clients (peers) all play equal roles. They exchange resources with one another directly. Peers are autonomous, self-organizing, and thus less structured, less secure, and less controllable than those of client/server systems.

8.1.1.2 Three P2P Network Models

Servers are also used in P2P networks, but they have a different role. Napster, a representative P2P file-sharing service, uses an index server to manage all file links shared by all peers. Figure 8.3(a) shows the Napster model. Peers register shared data information on the index server. The address information of file owners is given to the requester from the index server. Then the file is transferred between the requester and the owners directly. The Gnutella 0.4 file-sharing system adopts a pure P2P model, which is illustrated in Figure 8.3(b). Peers only index their own shared files.

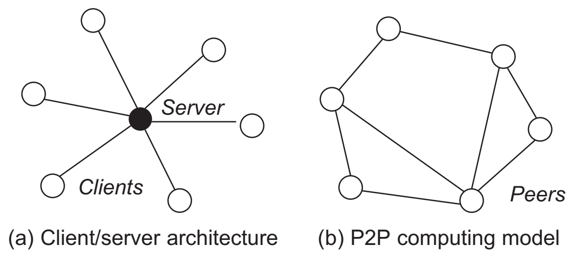


FIGURE 8.2

P2P network model versus client/server architecture.

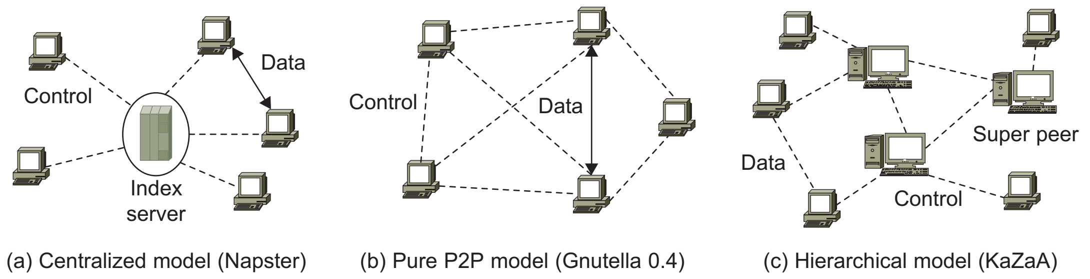

Data
Control
Index server
Control
Data
Data
Super peer
Control
(a) Centralized model (Napster)
(b) Pure P2P model (Gnutella 0.4)
(c) Hierarchical model (KaZaA)

FIGURE 8.3

Evolution of P2P network systems from centralized Napster to flat Gnutella and hierarchically structured KaZaA using super nodes.

Since there is no central index server, file request messages are flooded to their neighbor peers. Gnutella 0.6 systems, KaZaA, and Skype leverage the third model, the hierarchical model; see Figure 8.3(c). A portion of powerful peers are selected as super nodes in the system. The requests are only flooded among powerful peers, hence increasing scalability. P2P distributed computing systems always maintain central servers for task management and communication to client peers. However, client peers do not need to communicate. Therefore, peers in this context refer to computing systems that make their resources available. P2P platforms act as a middleware foundation to enable easy development of P2P systems. The platforms provide security services, communication services, and standard services (indexing, searching, and file sharing).

8.1.1.3 P2P Applications

P2P networks are widely used in Internet applications. Table 8.1 lists some typical P2P applications and examples. File sharing is the most popular P2P application. P2P content networks deliver data objects to all audiences. However, unlike client/server networks, the system uses peer resources to distribute content. Peers locate users and relay the messages directly.

Example 8.1 The Skype Network for Online Video Telephony

Skype is a popular P2P *Voice over IP (VoIP)* service which attracts more than 500 million registered users and more than 40 million active users daily. Skype leverages P2P techniques for user profile storage, user search, media transfer, and NAT traversal. It also supports instant messaging and file transfers. Instead of the standard *Session Initiation Protocol (SIP)*, Skype uses a proprietary VoIP network. Figure 8.4 shows the P2P architecture of the Skype online VoIP system.

Skype applies the hybrid P2P model, similar to KaZaA. In fact, these two systems are developed by the same design group. Any node with a public IP address and having sufficient CPU, memory, and network bandwidth is a candidate to be a super node. However, users cannot manually control whether it is a super node or ordinary host. An ordinary host connects to a super node, in order to join the system for VoIP services.

Table 8.1 Typical P2P Applications and Example Network Systems

P2P Application Area	Brief Description	Examples of P2P Networks
File sharing	Queries are forwarded by peers collaboratively. File objects are transferred directly between peers.	Napster, Gnutella, KaZaA, eDonkey
Content distribution	Nodes forward the content they have received to other peers until all nodes receive the content.	BitTorrent, PPlive
P2P instant messaging	Text messages or video and audio data are relayed by peers in the systems, instead of central servers.	Skype, MSN
Distributed computing	Computationally intensive tasks are split into subtasks. Nodes process subtasks independently.	SETI@Home, Folding@Home
Collaboration support	Events and messages exchanged are relayed instantly to all other peer members in the group.	Groove, Magi, AIM
Platforms	Platforms are general frameworks that provide support for P2P computing or use P2P mechanisms.	JXTA, Microsoft .NET

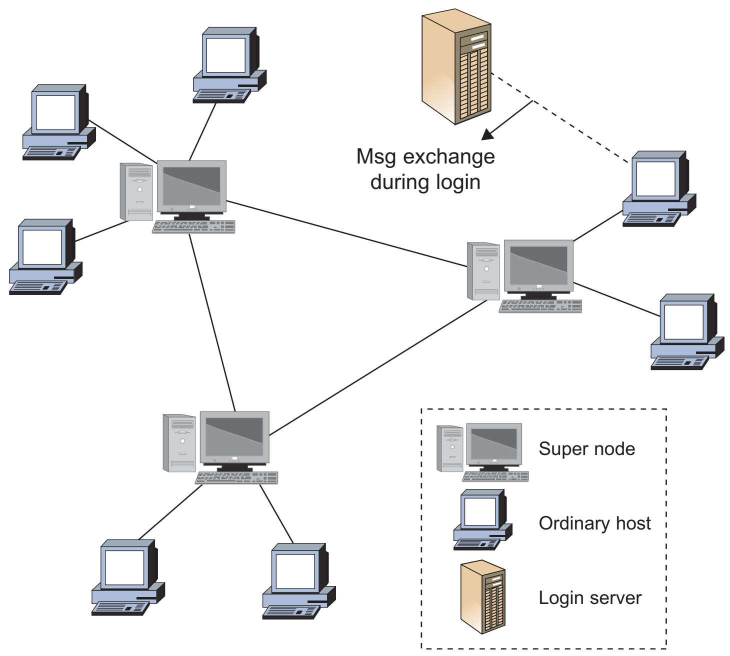


FIGURE 8.4

The Skype architecture and its major components.

A super node or ordinary host is also called a Skype client. Each client maintains a list of reachable nodes. This list is called the *host cache*, and it contains the IP addresses and ports of a number of super nodes (typically 200). A number of super nodes, which are run by the service provider, are hardcoded into the client host cache. After the first login, a client learns other super nodes' information to fill its cache. The login server is the only central component in Skype. It stores usernames and passwords for user authentication. During its login process, a client first chooses a valid entry from its cache to build a TCP connection. If there is no valid entry in the cache, the client will report a login failure. The client then contacts the login server for authentication.

A client detects whether it is behind the network address translator (NAT)/firewall or not during its login process. Online and offline user information is stored and propagated in a decentralized fashion, and so are user search queries. Users' buddy lists are stored locally on the Skype clients. Skype claims it uses an advanced Globe Index technique for quick user search. However, it is unclear how it works since all the communications, including both control messages and media data, are encrypted. Skype clients prefer to use TCP for call signaling and UDP for media transfers. However, if it is impossible to communicate through UDP due to a NAT/firewall, TCP connections are used. In the case that the caller or callee is behind a NAT/firewall, a valid node which has a publicly accessible IP address is used for signaling and media data relay. The relay nodes may change during conservations. ■

Example 8.2 The SETI@Home Project for Distributed Supercomputing

SETI is an acronym for *Search for Extra-Terrestrial Intelligence*. The goal of SETI is to analyze radio signals from the outer space for detecting intelligent life outside Earth. The analysis work is a computationally intensive task and requires Tflops/second (*Tera-floating-point operations per second*) processing power. The SETI@Home project aims to leverage the available CPU cycles at the edge of the Internet. The project was released by the Space Sciences Laboratory at the University of California, Berkeley in May 1999. SETI@Home has logged more than 5.2 million participants worldwide and has been considered the largest distributed computing project. Figure 8.5 illustrates the workload distribution process.

SETI@Home shares SERENDIP's (Search for Extraterrestrial Radio Emissions from Nearby Developed Intelligent Populations) data source, which was recorded on removable tapes and mailed to the laboratory at UCB. SERENDIP collects signal data sources from outer space at Arecibo using a telescope. The data sources are divided into small work units which are formed by dividing the 2.5 MHz signal into 256 frequency bands, each about 10 KHz wide. These work units are stored on the servers and are sent to participant computers. The programs on client sites analyze the work units during idle cycles. Finally, the results are reported back to the servers.

Users who would like to donate their computing resources download BOINC (Berkeley Open Infrastructure for Network Computing), which is a software that enables users to donate their idle computer time to science projects like SETI@Home. BOINC automatically downloads work units and manages analysis progress. The screensavers on users' computers display some analysis information when users are not using their computers. SETI@Home uses the concept of P2P in the sense that it takes advantage of available computing resources at the edge of the Internet. However, users' computers do not communicate directly.

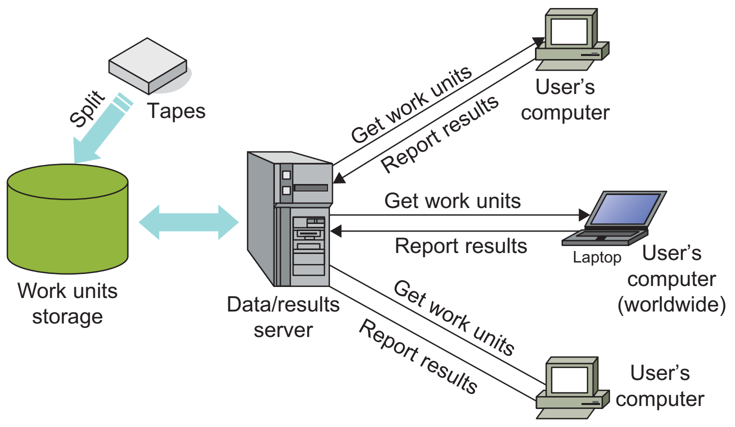


FIGURE 8.5

Workload distribution process in SETI@Home for volunteer computing.

8.1.2 Fundamental Challenges in P2P Computing

A P2P computing system eliminates the need for a central coordination point. This makes the system more scalable and flexible. In this section, we introduce 11 fundamental technical challenges in designing and using a P2P network in real-life applications. Solutions to these issues will be studied in subsequent sections.

8.1.2.1 Heterogeneity in Peer Resources

Basically, P2P computing faces three sources of heterogeneity problems: namely *hardware, software*, and *network* requirements. Peers offer many different hardware hosts and platform architectures. Incompatibility does exist between the software and the OS at different hosts. Different network connections and protocols make it even more complex to apply in some real-life applications. A P2P computing system aggregates diverse resources at the peer nodes. The system should be designed to handle node heterogeneity transparently.

8.1.2.2 Scalability for Future Growth

This is a fundamental goal of P2P computing systems. The system is able to support web-scale computers from hundreds to millions, and as the system size grows, the efficiency should not be affected negatively, or should just degrade gracefully. System scaling is directly related to performance and bandwidth. For example, a decentralized P2P file-sharing system uses message forwarding for information searching. If the forwarding path length grows linearly with the system size, the system cannot be scaled to a larger size.

With flooding, the messages consume a considerable network bandwidth. Gnutella version 0.4 took flooding as a query forwarding scheme, and hence was not scalable. Gnutella version 0.6 and KaZaA leverage some powerful and stable nodes as super peers. Each peer is either a super peer or assigned to a super peer. Queries are only flooded among super peers, which saves bandwidth consumption. The super-node model is more scalable in this sense.

8.1.2.3 Effective Search of Desired Peers

Data resources spread over the peer nodes in P2P systems. To use or to access a data object, a data locating operation (or search operation) among the peer nodes is performed first. Apparently, the data location is important in terms of affecting collective performance. However, there is no central index server for the shared resources, and each peer only has a partial view of the whole system. It is difficult to find the right peers for queries. The design of efficient data location algorithms is a unique challenge in P2P systems.

Search can be classified into two categories: blind searches and informed searches. In a blind search, peers only have the indices of their own data. When receiving a query, a peer forwards it to either all its neighbors or a subset of its neighbors without considering from which path it can receive the query response. In an informed search, peers maintain indices of the data residing on other peers. For example, a peer keeps track of the data indices of the peers that r hops away from it. Peers forward queries to the neighbors that can respond with high probability. The design challenge in a blind search is to find the right peers quickly with low-bandwidth consumption, while the challenge in an informed search is to track peers' data indices with low control overhead.

8.1.2.4 Data Locality and Network Proximity

These are two design objectives in modern P2P applications. Data locality means data with similar attribute values is kept on close peers. It is an efficient way to support complex query operations and enable fast data location. Network proximity is measured by the closeness of two peers in the underlying physical IP network. By taking network proximity into account, one can build a P2P overlay network in such a way that physically close peers are connected with one another. Thus, query messages or data items are likely to be exchanged among physically close peers, which not only improves *Quality of Service* (QoS) but also saves network bandwidth. Figure 8.6 illustrates the advantage of network proximity awareness.

Figure 6.5(c) shows a part of the IP network. C1 and C3 belong to one *autonomous syst*em (AS), while C2 and C4 belong to another. If the nodes form an overlay as shown in Figure 8.6(a), a message from C3 to C1 would traverse the link between two ASes two times. This prolongs the message delivery delay and consumes unnecessary backbone network bandwidth between AS1 and AS2. If network proximity is considered and a proximity-aware overlay is built as in Figure 8.6(b), the message would not transfer the link between two autonomous systems. It is a challenge to find physically close nodes in a decentralized P2P overlay with a central server and no node has the global knowledge.

8.1.2.5 Routing Efficiency

As in IP network, routing routines are demanded in forwarding a message to its destination. They directly affect the performance of P2P systems. A pure P2P system avoids single points of failure. However, it still faces failures related to disconnections and unreachability, partitions, and peer node failures. The system should run properly even if there are failures, and it should repair failures in an ad hoc way. Replication is an efficient way to improve system availability and fault tolerance. Maintaining the consistency of replicas in a distributed system is always a big challenge.

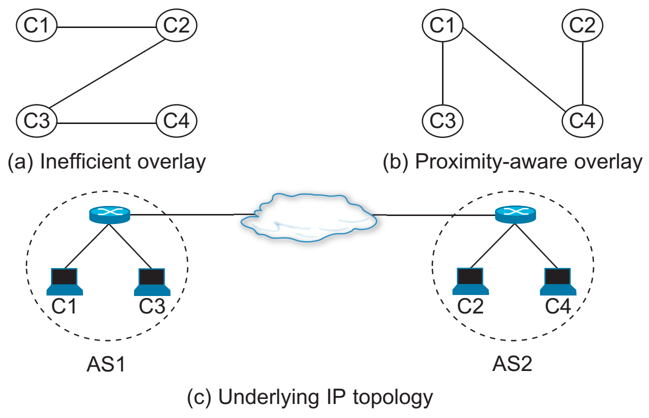


FIGURE 8.6

Building a proximity-aware P2P overlay network.

8.1.2.6 Free-Riding Avoidance

Another challenge in P2P computing systems is the free-riding problem. P2P computing systems aggregate resources at the edge of the Internet to improve performance. The participant peers may be selfish and unwilling to contribute anything. These peers are called *free riders*. The free-riding problem is common in systems that rely on voluntary contributions from individuals since people are not willing to pay the cost of public goods when they hope someone else will bear the cost instead. Thus, incentive mechanisms to encourage contributions from individual peers are required. The incentive mechanism should evaluate peers' contributions correctly and give awards to the contributors, while maintaining anonymity and defending malicious attacks.

8.1.2.7 Anonymity and Privacy

Peers in P2P systems like to hide their information from others. Anonymity should be an option to peers, especially for peers in P2P communication systems. There are three levels of anonymity: namely initiator anonymity, responder anonymity, and mutual anonymity. The first two levels of anonymity can be further seen as directional anonymity, while the third level can be seen as bidirectional anonymity. Basically, directional anonymity can be implemented using onion routing.

Example 8.3 Onion Routing to Protect Anonymity of Peer Machines

In onion routing, the initiator knows everything. However, intermediate routers does not. Onion routing is used to hide the identify of the originator of the final destination. This is done by encryption of the incoming message and the next hop information recursively using public keys. It creates a number of layers for the message. Each layer contains the information of the next hop nodes (called *onion routers*). Each onion router removes one layer and forwards the remaining message to the next hop onion router. This way, any node knows only its immediate sender and who is the next receiver.

Figure 8.7 shows an example of onion routing. The message is sent from A to B. The identity of A is hidden from C, D, and A, while the message itself is hidden from C and D. Onion routing can be improved using encryption with symmetric keys. Mutual anonymity requires hiding both the initiator and responder's identities, and no others are able to guess the two parties. Encrypting messages and building agents for peers are possible ways to implement mutual anonymity. However, these techniques bring high overhead and conflict with some design goals of P2P systems. For example, using encryption incurs cryptography cost, while using agent conflicts with decentralization.

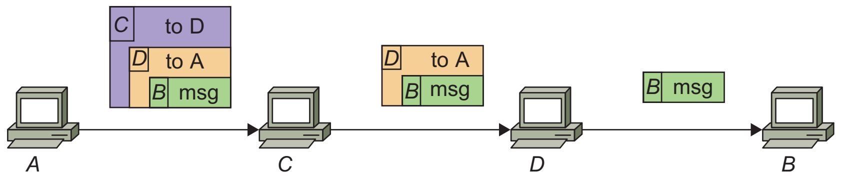


FIGURE 8.7

An example of onion routing for anonymous communication.

8.1.2.8 Trust and Reputation Management

Lack of trust among peers poses another problem. Peers are strangers to one another, and inevitably some of them act evilly or in an uncooperative way. It is required in P2P systems to provide a trustworthy environment in which peers' trust is measurable and malicious peers are punishable. However, P2P systems are fully distributed. Transactions are performed directly between peers. No central server keeps track of the transactions. Hence, building a trustworthy P2P environment is a unique challenge.

8.1.2.9 Network Threats and Defense against Attacks

Due to the decentralization and ad hoc characteristics, attacks in P2P systems can be easily performed. *Denial of service* (DoS) and *distributed denial of service* (DDoS) attacks are possible by flooding messages to particular peers and claiming that the victim has all files requested. QoS attacks can be implemented by serving files slowly or sending different files to requesters. The anonymity design of P2P systems helps malicious peers easily hide behind others.

Example 8.4 Defense against DDoS Attacks

Figure 8.8 shows a DDoS attack by two malicious nodes through flooding queries throughout the entire network. The malicious peers flood one or more network nodes with unwanted queries or messages so that their communication buffers get overflowed, and thus TCP connections get denied. The flooded node is saturated by these messages and is unable to handle more queries from good peers. A well-designed P2P network should be able to handle this type of flooding attack to any node in the system. Sometimes the attack could be initiated from outside the P2P network and use some of the peer nodes simply as zombies.

8.1.2.10 Churn Resilience

The peers in P2P computing systems are from clients at the edge of the Internet. They come, leave, and even fail at random. Faulty nodes no longer forward messages. In some cases, failure of a node would cause others to disconnect from the whole system or lose index information. For example, in KaZaA, if a super node fails, the nodes assigned to it would lose connections to the system and

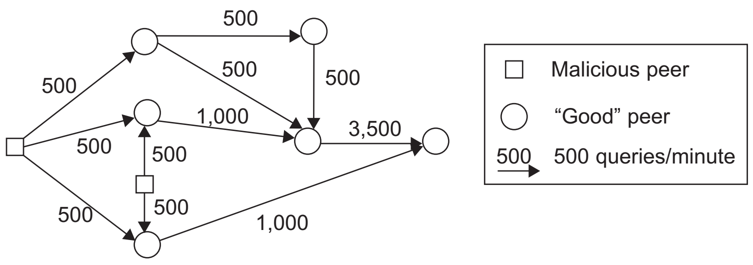


FIGURE 8.8

An example of a DDoS attack in a small P2P network through message flooding.

have to find other super nodes to connect with. The index information kept in the faulty super node would be lost. Node failure imposes a great challenge to achieve fault tolerance in P2P networks. Since there is no central coordinator, peers use periodic heartbeat messages to check the status and availability of other peers.

8.1.2.11 Collusive Piracy Prevention

Online piracy has hindered the legal and commercial use of P2P file sharing. The illegal file contents are from the peers that have the legal contents and spread them to the pirates. This is called collusion. An agent recruits clients to send illegal download requests to suspected peers. For each request, if the suspected peer returns a clean copy, the feedback is 1. Otherwise, the feedback is −1. If the sum of feedbacks exceeds a predefined threshold, the suspected peer is recognized as a colluder.

Collusive piracy is the main source of intellectual property violations within the boundary of a P2P network. Paid clients (colluders) may illegally share copyrighted content files with unpaid clients (pirates). Such online piracy has hindered the use of open P2P networks for commercial content delivery. Lou and Hwang [32] have developed a proactive content poisoning scheme to stop colluders and pirates from alleged copyright infringements in P2P file sharing. The basic idea is to detect pirates in a timely manner with identity-based signatures and timestamped tokens. This scheme stops collusive piracy without hurting legitimate P2P clients by targeting poisoning on detected violators exclusively.

8.1.3 Taxonomy of P2P Network Systems

This section classifies the existing or proposed P2P networks by their structures and functionality. P2P network systems are identified by their names or by a short description in Figure 8.8. Some of the well-known networks will be treated in more detail in examples in subsequent sections.

8.1.3.1 Unstructured P2P Overlays

The neighborhood relationships in an unstructured P2P overlay network are randomly connected without much constraint. Unstructured overlays are easy to build up or down when user anonymity and minimal administrative overhead are desired. At the time of this writing, many P2P networks can be started with no structural constraints. Well-known examples include Napster, Gnutella, KaZaA, Skype, BitTorrent, and eMule. Here are some interesting features in an unstructured P2P overlay:

- Data is distributed randomly over peers.
- The overlay can start with centralized control and gradually shifted to fully decentralized control.
- There are no broadcasting mechanisms (or if there are, they are constrained).
- Flooding queries to entire network may generate heavy network traffic.
- There is no guarantee of a deterministic search result.
- The TTL (time to live) limit on a query message may reach the entire network.

8.1.3.2 Structured P2P Overlays

In a structured P2P overlay network, peer nodes are organized into predefined and restrictive structures, such as a ring, a chord, or a tree to be characterized in Section 8.2. Good examples of structured overlays include Chord, CAN, Pastry, and Tapestry. Many P2P networks may start with a random structure, but later adopt a more efficient overlay structure to speed up searches and downloads with enhanced security and efficiency. Here are some useful features of structured overlays:

- Structured routing mechanisms in the overlay
- Add application-level overlays over the peers
- Reduce routing hops, compared with using random graphs
- Avoid flooding and eliminate hotspots
- Guarantee the search results
- Provide load balancing among the peer nodes
- Provide enhanced scalability and fault tolerance
- Preserve data locality if desired
- Promote self-organization under constraints
- Provide enhanced security protection
- Support node heterogeneity

8.1.3.3 Taxonomy by Structure and Functionality

Figure 8.9 maps existing P2P systems according to their design style and functionality. The bottom row denotes centralized systems built in the earlier years. The middle row covers unstructured P2P systems that are decentralized, like Gnutella, FastTrack, and DirectConnect. The upper row is for decentralized and structured P2P systems. At the lower-left corner is Napster, which is centralized and unstructured. At the upper-right corner are P2P resource managers, P2P directory services, and P2P databases that are all decentralized using structured overlays are expected.

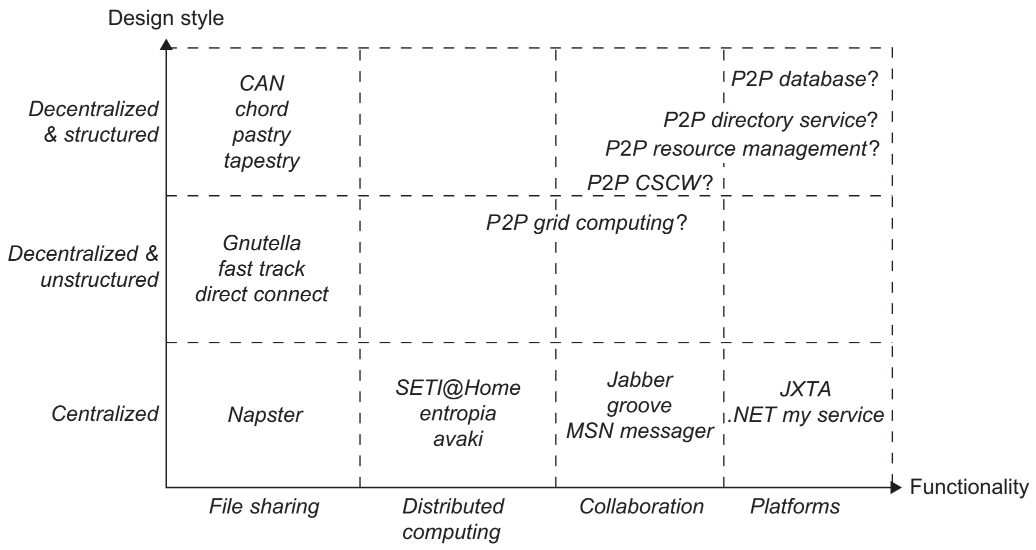


FIGURE 8.9

Classification of various P2P systems by functionality and control style.

(*Courtesy of Min Cai, EE 657 Report, University of Southern California, 2007*)

CSCW refers to *computer-supported cooperative work.* CAN, Chord, Pastry, and Tapestry are now all using structured overlays with decentralized control. Functionally speaking, the P2P grids are mainly for distributed computing and collaboration applications that have no fixed structures. SETI@Home has centralized control for distributed supercomputing applications. The rest of the P2P networks fall in various squares in the 2D design space as shown. Note that some lattice squares are empty or are labeled with a question mark, meaning no systems are being built yet under that category.

8.2 P2P OVERLAY NETWORKS AND PROPERTIES

Overlay networks are built on top of physical IP networks. The nodes are end hosts from the physical networks, while the links are virtually built among nodes with TCP connections or simply pointers to IP addresses. The virtual links do not need to have the same weights. Different weights can be designed for different types of links. Since end hosts are dynamic, overlay networks require protocols for overlay maintenance. A new node contacts another node which is already on the overlay network for bootstrapping. Nodes in overlay networks leverage periodic heartbeat messages to verify neighbor aliveness. If a current neighbor fails, a node may connect to another node according to the maintenance protocol.

Recall that in Figure 1.16, one can map some nodes from physical IP networks to an overlay network built with virtual links. The mapping is shown by the vertical dashed lines between physical hosts to virtual nodes called peers. Overlay networks do not require additional physical infrastructure; hence they are easy to deploy and use. The overlay topology can change based on application. Node failures are easily recovered since nodes can choose others to connect with. There is no restriction on the communication protocol. Application designers can define any protocol they like. The underlying physical networks are transparent to the overlay designers. However, to better utilize network resources (e.g., network proximity), one should take physical networks into consideration.

A P2P network is a type of overlay network. According to overlay graph properties, P2P networks are classified into two categories: namely *unstructured* and *structured* overlays. Unstructured overlay networks are always built based on random graphs. Nodes select their neighbors randomly from all the peers in the overlay. Structured overlay network graphs, however, have predefined structures (e.g., ring, hypercube, etc.). Each peer has a unique identifier and can only build neighborhood relationships to the peer nodes whose identifiers have satisfied the predefined requirements. Some hybrid P2P overlay networks, which leverage the advantages of both unstructured and structured overlay networks, are also used to reach the advantages of these two types of overlay networks.

8.2.1 Unstructured P2P Overlay Networks

To build a good unstructured P2P overlay, the node degree (i.e., the number of neighbors) of a peer and the expected number of nodes that are traversed from one node to another should be kept as small as possible. In addition, the join or departure operations cannot cause extensive modifications of the graph. Finally, the overlay can determine the forwarding path easily, when some nodes fail or depart unexpectedly.

8.2.1.1 Construction by Random Graphs

The Erdos-Renyi random graph [38] can be considered as a baseline model for unstructured P2P overlay networks. The probability p that any two vertices (nodes) have an edge is equal and independent. The probability that a vertex has a degree of k is as follows:

$$P\{d = k\} = \binom{n-1}{k} p^k (1-p)^{n-1-k} \tag{8.1}$$

The expected degree of a vertices is $E\{d\} = (n-1)p$. Apparently, if $p = c/n$, then for a large n, $P\{d = k\} = \frac{c^k}{k!} e^{-c}$. That is how the node degree follows a Poisson distribution. For unstructured P2P systems, the Erdos-Renyi random graph is too random to design a distributed algorithm for determination of the forwarding path from one peer to another peer.

8.2.1.2 Small-World Model

Small-world graphs [2] have two prominent properties: a short mean-shortest path length and high cluster coefficient. For P2P systems, these two properties are important. The former one helps us to reduce the expected hop count between peer nodes and the latter one facilitates the handling of the flash crowd. Suppose each node has two types of neighbors, namely nearby neighbors and distant neighbors. The nearby neighbors of a node i are the nodes whose distances are smaller than p. Node i also has q distant neighbors that are selected randomly from the nodes that are out of this scope. A node i builds a link to any node j, where the distance of i and j is smaller than p, that is, $d(i,j) < p$. Node i builds a link to a distant node v with probability $d(i,v)^{-r}$, where r is the clustering exponent. A search query is forwarded to the closest neighbor to final destination. The clustering exponent r affects the search time.

8.2.1.3 Scale-Free Graph

The scale-free graph model is also widely used in the context of P2P applications. The node degree follows a power-law distribution, which means the probability that a node has a degree of k is proportional to $k^{-\alpha}$, where α is a constant and always in the range (2, 3). The graph diameter (i.e., the longest minimum distance of any two nodes) is of the order of $O(\ln n)$, where n is the number of nodes in the graph. Thus, the diameter almost remains unchanged when the graph scale grows. In practice, many real networks are thought to be scale-free, such as social networks and the Internet graph.

In the Gnutella 0.4 overlay network, the node degree follows a power-law distribution with α approximately equal to 2.3 [41]. This is mainly because new join nodes are likely to build overlay connections to the nodes already with high degrees. A scale-free network is also resilient to the random node failures. However, if the network is attacked by knowledgeable attackers and some of the high-degree nodes break down, the graph may be partitioned into several disjointed parts or the graph diameter will grow quickly. Thus, to build a resilient scale-free graph based on P2P overlays, high-degree node identities should be hidden if possible.

8.2.1.4 P2P Networks for Distributed File Sharing

The most popular applications that are built on top of unstructured P2P networks are P2P file-sharing systems. Table 8.2 compares the P2P file-sharing systems Napster, Gnutella, and KaZaA. These three systems represent three distinct types of P2P service models. Basically, in these

Table 8.2 P2P File-Sharing Applications Using Unstructured Overlay Networks

P2P Network	Napster	Gnutella	KaZaA
Architecture	Centralized model	Pure P2P model	Hierarchical with super peers
File indices	Centralized index servers keep all links	Each node keeps file indices for itself only	Super node keeps the indices for the nodes that connect to them
Search algorithm	Queries are sent to servers, which return the requested files	Queries are flooded on entire overlay Matching peers reply to originators	Queries to super nodes with flooding among themselves and reply to the originators
Search time (hops)	$O(1)$	$O(\log n)$	$O(\log m)$, m counts supernodes
Search cost (#msgs)	$O(1)$	$O(d^* n)$, d is node degree	$O(c^*$ m), c is supernode degree
File fetch method	Originator gets the file from a single peer	Query originator gets the file directly from a peer	Originator gets the file from multiple peers concurrently
Scalability	Limited by the central servers	Limited by the high search cost	Relatively good scalability due to super peers
Fault tolerance	Single point of failure	Good fault tolerance due to equality among peers	Super peers are vulnerable to attacks

file-sharing systems, data is distributed randomly over the peers. Flooding on the overlay is used to find files in response to requests. The TTL is imposed to limit the flooding, in order to save heavy traffic generated by flooding. Moreover, there is no guarantee on the search results.

The major difference among these systems lies in the fact that Napster applies a centralized control, Gnutella uses pure P2P networking, and KaZaA applies a hierarchical structure. Napster has a constant search time. Flooding in Gnutella takes the longest search time across the entire network. KaZaA performs in between, because binary search is conducted only across the super nodes. Both Napter and Gnutella have limited scalability. KaZaA has better scalability. However, these three P2P file-sharing services have all suffered greatly from illegal downloads by copyright violators in recent years.

Example 8.5 Gnutella (Version 0.4) Networks with Flooding Search

The Gnutella file-sharing application leverages the pure P2P model. Each peer has equal roles and acts as either a server when uploading files to others or a client when downloading files from others. A peer is also called a *servant*. Gnutella defines a number of messages, called *descriptors*. Flooding-based search is used to locate highly replicated items, as illustrated in Figure 8.10. Each node sends the requests to multiple neighboring nodes, which can send to more nodes, and thus flood the entire network. This flooding scheme may create heavy traffic to saturate the network. Therefore, the messages to send or to forward and the hop count in forwarding must be limited.

Figure 8.11 shows the descriptor header and payload. The header consists of 22 bytes. The descriptor ID uses the first 16 bytes. It is a unique identifier for the descriptor in the network. The payload descriptor

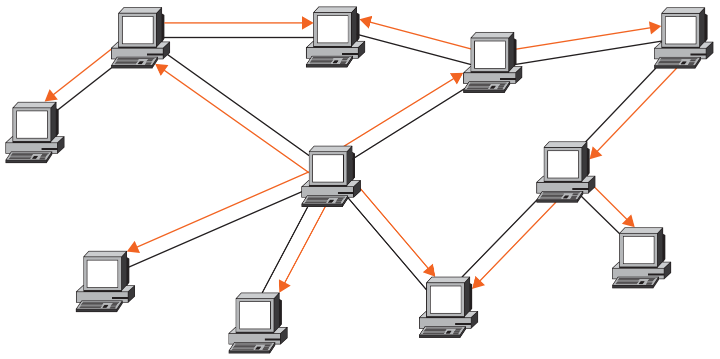


FIGURE 8.10

Flooding scheme used in Gnutella to search for peers that can provide files of digital content.

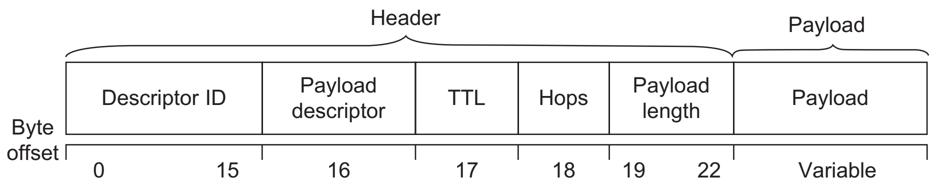


FIGURE 8.11

Gnutella packet descriptor format.

only uses 1 byte. Gnutella defines five main types of descriptor, including *Ping* (0X00), *Pong* (0X01), *Query* (0X40), *QueryHit* (0X80), and *push* (0X80). The TTL and Hops fields use one byte each. TTL is initialized by the descriptor originator and is decreased by one once the descriptor traverses one servant. When TTL reaches 0, the descriptor is discarded.

The TTL is used to limit the horizon of the descriptor in order to save the communication cost of unlimited flooding. The Hops field counts the number of hops the descriptor has traveled. The last four bytes in the header is for payload length, which is followed by variable payload. For example, The Ping has no payload, while other four descriptors, payload is filled with bytes of variable size.

The Ping descriptor is used by servants to discover others. A servant that receives a Ping sends back one or more Pong descriptors along the path of Ping with a reverse direction. A Pong consists of the address of the responsible servant and the amount of data it shares. Query descriptors are for search, while a QueryHit descriptor is the response to a Query. The Push descriptor enables firewalled servants to share files with others. A new servant contacts bootstrap nodes to join. After joining the system, a servant communicates with others using descriptors. It sends a Ping to discover more servants to connect with.

In general, the Ping and Query descriptors are sent in a flooding fashion. A servant that receives such a descriptor will forward the descriptor to its neighboring servants. Clearly, if the horizon of the flooding is not limited, such descriptors would, on average, be forwarded to *d* neighbors by each peer, where *d* is the average number of neighboring peers of each node. Thus, an *n*-node Gnutella system must use *d** *n* messages, within which (d-1)*n messages are duplicated. Such a large number of messages limits system scalability.

Gnutella limits the horizon of flooding with a TTL in each descriptor. The TTL is decreased by one each time the descriptor is forwarded. If the TTL expires, that is, it reaches 0, the descriptor is discarded without further forwarding. By default, the TTL is set at 7 since within this horizon, 95 percent of servants can be reached. However, the requested files may be owed only by the unreached 5 percent of servants. In such cases, the requests cannot be satisfied, even though files that match the requests do exist.

The flooding search in Gnutella is suitable for highly replicated files (i.e., hay), not for rarely replicated ones (i.e., needle). Since users often request hay, Gnutella works well in practice. The request originator downloads the file from one peer directly using HTTP. From version 0.6, Gnutella leverages powerful peers as *ultrapeers* to construct hierarchical architecture similar to KaZaA's. The descriptors are only flooded among ultrapeers, greatly reducing search cost and improving scalability.

8.2.2 Distributed Hash Tables (DHTs)

As shown in Figure 8.12, a *distributed hash table* (DHT) is middleware which offers information search or table lookup services for a distributed system, especially for P2P systems. A hash table is often represented by (*key*, *value*) pairs. DHT stores such pairs in an *identifier space*. For example, an identifier space, represented by 64-bit binary strings, can accommodate 2^{64} keys, representing the same number of data objects. Using separate hash functions, the DHT maps the data objects into the keys. The (key, data) pairs are distributed to the peers in a P2P network. The identifier of a peer is the hash value of its address using certain hash function such as SHA-1.

The ownership of the identifier space is distributed among the peer nodes. DHTs provide a lookup service similar to a hash table: (*key, value*) pairs used in other mapping services. Any participating peer can retrieve the value associated with a given key. In a P2P network structured with DHT support, the mapping from keys to values is distributed among the peer nodes. Any peer joining or leaving should cause minimal disruption to the entire P2P network. An efficient DHT design should scale to an extremely large identifier space such as 2^{64} or even greater such as 2^{256}.

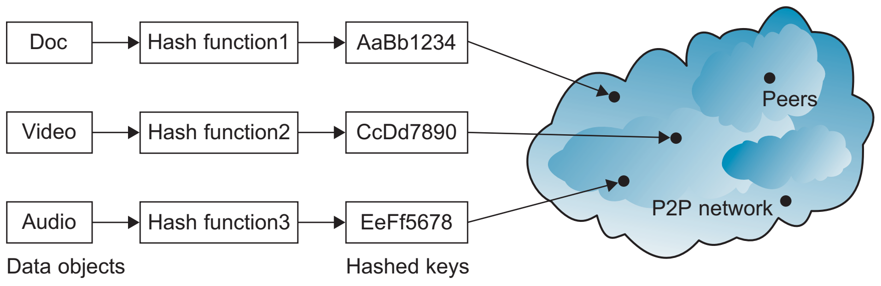


FIGURE 8.12

Key mapping in distributed hash tables (DHTs).

The DHT applies *consistent hashing* to map keys to nodes. This consistent hashing should be able to define an abstract notion of proximity or distance between two nodes. Keys are mapped to the nearest nodes with the minimum distance or maximum proximity. When a new node joins or an existing node leaves, consistent hashing causes a small impact on existing nodes by reassigning their keys. For n nodes with k keys, each node is responsible for, at most, $(1+\varepsilon)/n$ of the whole ID space with a high probability. The addition or deletion of a node requires $O(1/n)$ keys to reassign, where $\varepsilon = O(\log n)$. Thus, the DHT is suitable for P2P systems where nodes leave and join frequently.

The DHT provides two primitive operations: *put* (*key, data*) and *get* (*key*). The *put* primitive stores the date or data index with the *key* on the node which has the closest identifier to the key, while the *get* primitive retrieves the node that stores the data or data index with the *key*. Peer nodes form an overlay network. Every node keeps several links to a subset of other nodes. The *get* request for a key with identifier k is forwarded over the overlay to the nearest node, until the request arrives at the node with an identifier closest to k.

Popular P2P networks that use DHTs include BitTorrent's distributed tracker, the Bitcoin monetary network, the Kad network, the Storm botnet, YaCy, and the Coral Content Distribution Network. Some prominent research projects including the Chord project; the PAST storage utility; P-Grid, a self-organized and emerging overlay network; and the CoopNet content distribution system also apply the DHT in their overlay constructions. Others apply the DHT in resource discovery in grid computing systems. Resource discovery searches for suitable resource types to match with user application demands.

8.2.2.1 DHT Deployment

The DHT acts as a substrate to provide two primitives: *put* (*key*, *data*) and *get* (*key*). The basic idea is to map nodes and keys into the identifier space and assign keys to closer nodes in this space. This enables fast search with guaranteed lookup success and provable bounds on the search time. Moreover, such overlay networks avoid high search costs by flooding, and thus are more scalable. However, it should be noted that DHT-based structured P2P overlays only directly support exact-match search, rather than keyword search. Figure 8.13 shows the role of a DHT in mapping P2P user applications to desired source machines through proper TCP/IP connections.

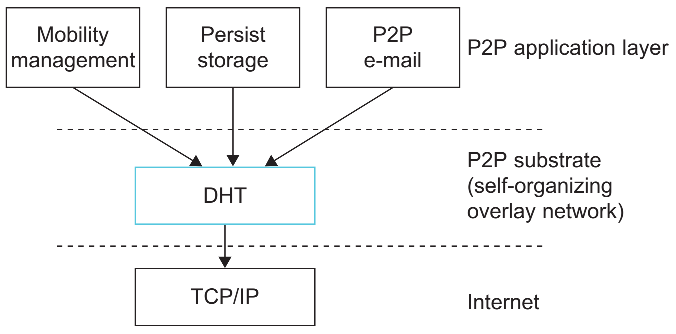


FIGURE 8.13

DHT for fast and secure search and other Internet applications.

8.2.3 Structured P2P Overlay Networks

In a ring structure, each node has two neighbors: its successor and its predecessor along the ring. If a node A is inserted at the interval between B and C, it has to take these two nodes as its neighbors. In a tree structure, a node can only build neighboring links with its child nodes and parent node. Although any P2P overlay network that has a logical predefined structure is considered structured, the most popular structured overlays are DHT-based.

Structured P2P networks employ a globally consistent protocol to ensure that any node can efficiently route a search to some peer that has the desired file, even if the file is extremely rare. Such a guarantee necessitates a more structured pattern of overlay links. By far the most common type of structured P2P network is the DHT, in which a variant of consistent hashing is used to assign ownership of each file to a particular peer, in a way analogous to a traditional hash table's assignment of each key to a particular array slot. Table 8.3 summarizes various structured P2P overlay networks; three examples follow the table. An *n*-node network with *b* bits per node identifier is considered here.

- **Distributed hash tables** Use distributed hashing with key lookup to avoid flooding at the expense of losing data locality. Good examples are Tapestry, Pastry, Chord, and CAN.
- **Tree-structure system** Preserves data locality with a tree-structured data access hierarchy. A good example is the TerraDir system.
- **Skip-list-based systems** Accelerates query processing by key ordering rather key lookup. Two good examples are Skip Graphs and SkipNet.

Due to the random selection of neighboring peers, there often exists a topology mismatch problem between overlay and physical topologies in P2P systems. Location awareness or network proximity awareness solves the mismatch problem by allowing nodes to select neighboring peers on overlay networks according to peers' locations on physical networks.

In DHT-based structured P2P overlays, the neighbors of a node are not selected randomly from all the peers on the overlay, but only from the ones whose identifiers are subject to special requirements. Addition or deletion of a node requires few recovery operations, which facilitates large-scale deployment. There are different overlay structures for DHT-based overlay networks, which yield

Table 8.3 A Comparison of Three Structured P2P Overlay Networks

Overlay Networks	Chord [45]	CAN [40]	Pastry [39]
Identifier Space	Chordal ring with direct fingers connecting to nodes in distance intervals	*d*-dimensional coordinates	High-radix identifier space that divides nodes into nested proximity groups
Routing state	log *n*	2*d*	$2 \times 2^b \log_{2^b} n$
Lookup Protocol	Matching data key and node identifier	Mapping key to a point in the *d*-dim coordinate	Matching key and prefix of node identifier
Routing complexity	$O(\log n)$	$O(d \times n^{1/d})$	$O(\log_{2^b} n)$
Number of messages per join/leave operation	$\log^2 n$	2*d*	$\log_{2^b} n$

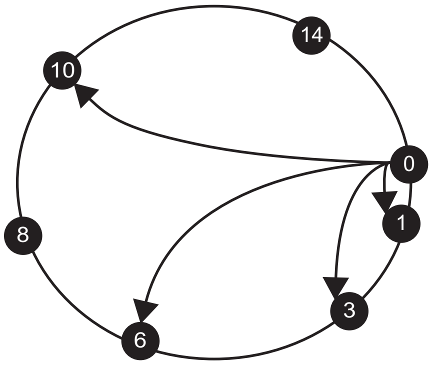
different structured P2P systems, such as Chord, Pastry, CAN, and Kademlia. This section focuses on the structures of these structured P2P systems. In Table 8.3, n is the number of nodes in the P2P network. The quantity 2^b for specifying Pastry is the base of the identifier space. The "log" implies the binary base 2.

A good example of DHT systems, *Chord*, is introduced in Example 8.6. The identifier space of m-bit strings is mapped to a unidirectional logical ring modulo 2^m. The distance between the identifier id_1 and the identifier id_2 is measured by the interval traveled from id_1 to id_2 in a clockwise direction. Nodes are placed at the position where the node's identifier can be found. A node with identifier id_x stores the data or data index with key identifiers falling into the interval from the immediate predecessor of id_x to id_x itself. Each node keeps m chordal links to the nodes whose identifiers immediately succeed $(x+2^i)\text{mod}2^m$ for $0 \leq i \leq m-1$.

Example 8.6 Chord System—A DHT-Based Chordal Ring

Chord is a typical DHT system which organizes peers in a unidirectional ring augmented with chord links for scalable search [45]. Nodes and keys are mapped to the same m-bit identifier (ID) space using SHA-1 as the base hash function. A peer's ID is generated by hashing the peer's IP address, while a key's ID is assigned by hashing the data key. Identifiers are represented on the ring modulo $2m$. A peer is responsible for the ring interval from the previous ID to its own ID. That is, the data (or the index) with keys mapped to this interval is stored on this peer. The distance from an ID1 to another ID2 is the interval from ID1 along the clockwise direction of the ring to ID2.

Besides the predecessor and successor on the ring, each node maintains a *finger table* which consists of the nodes pointed by the chord links from that node. The i-th $(0 \leq i < m)$ entry in a node's finger table refers to the node whose identifier is 2^i away along the clockwise direction, where m is the identifier length. A 3-bit identifier ring with four nodes and the finger table of node 8 are shown in Figure 8.14. Node *N6* is



Gap	Span	Scsr
1	(1,2)	1
2	(2,4)	3
3	(4,8)	6
8	(8,1)	10

Node 0 finger table

FIGURE 8.14

An example of a Chord network with a 16-key search space. The finger table establishes the direct links between the populated nodes located in various span intervals.

2^2 away from *N1*. Although there are *m* entries in a peer's finger table, on the average, a peer's finger table contains *O*(log *n*) nodal information, where *n* is the total number of nodes in the search space.

In Figure 8.14, N1's finger table has three rows. However, the first two rows contain the same node's information, that is, N3's information. This is stemmed from the fact that the identifier range which starts from a node to its $\lfloor m - \log n \rfloor$-th entry is within distance $2^m/n$. Since nodes are randomly distributed on the identifier ring, an average of one node is within the range $2^m/n$. Thus, on average, the number of nodes in a node's finger table is $O(m - \lfloor m - \log n \rfloor + 1) = O(\log n)$. DHT-based search is often used to locate rarely replicated items.

To improve robustness, each node maintains a successor list. The list starts from its successor down the ring. If the current successor fails, a node replaces the fault successor with the first live entry in the list. A newly coming node of ID *X* first contacts a bootstrap node, which forwards the join message to *X*'s predecessor *Y*. The new node joins as *Y*'s successor and *Y* finds entries to fulfill the finger table for the new node. The nodes whose finger tables should contain *X* update their tables. The join/leave operation of a node requires using $\log^2 n$ messages. Chord uses a stabilization procedure running periodically on each node to update the successor and finger table entries.

Example 8.7 The CAN System: A Multidimensional Mesh Architecture

CAN (*Content Addressable Network*) organizes nodes in a virtual *d*-dimension Cartesian coordinate space on a *d*-torus. The coordinate space is dynamically split into *n* disjoint zones and each node owns one, where *n* is the number of nodes. An example of a 2D coordinate space which is partitioned into five nodes is given in Figure 8.15. Nodes store their neighbors' information (i.e., IP addresses and the zone information that they own) as their route tables. Two nodes are neighbors if their zones span overlap along (*d*-1) dimensions and abut along one dimension.

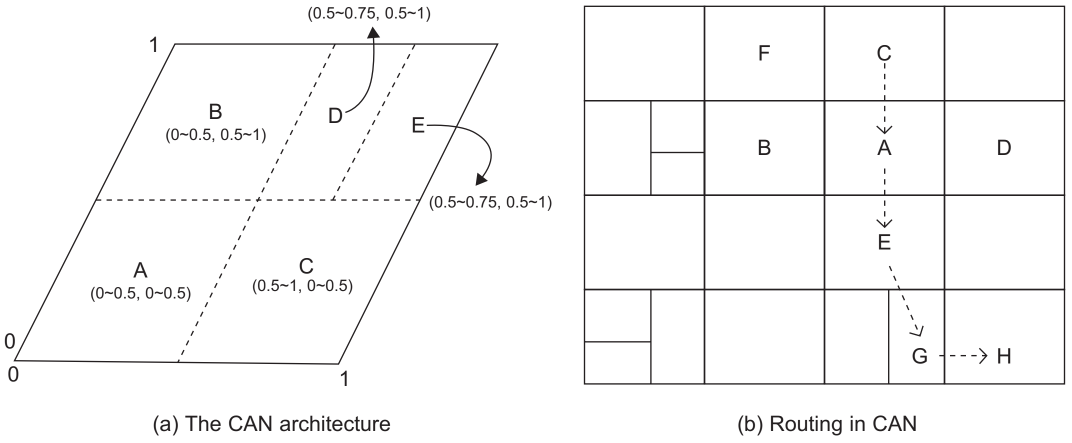


FIGURE 8.15

A CAN network constructed by the repeated partitioning of the 2D coordinated space and its routing process.

(*Courtesy of S. Ratnasamy, P. Francis, and R. Karp [40]*)

In Figure 8.15(a), node B and node C are neighbors of node A, but node D is not A's neighbor. Keys are mapped to coordinate space using uniform hash functions. Each key is associated with a d-dimension vector, which represents its position in the coordinate space. If a key's position in the coordinate space falls in the zone owned by a node i, node i stores the pointer to the object related to the key or directly stores the object. In a d-dimension coordinate space, each node maintains $2d$ neighbors on average and the route complexity is of the order of $O((d/4)(n^{1/d}))$.

A new coming node X contacts a bootstrap node, which is located using the DNS service. The bootstrap node supplies the address information of some randomly chosen peers populated in the overlay. The new node randomly chooses a point P in the coordinate space as its position and asks a populated node to route the join message to the node Y whose zone contains P. The node Y splits its zone into two parts with equal quota and hands over one subzone to the incoming node.

The CAN overlay uses a greedy routing algorithm. A node greedily forwards the message to its neighbor with coordinates closest to the destination. Figure 8.15(b) shows a 2D CAN overlay and the routing path from node C to node H. In a CAN overlay with d-dimension coordinates, on the average, each dimension has $n^{1/d}$ nodes. On a torus, there are $1/4n^{1/d}$ nodes between any two nodes on the average. Since a message to its destination traverses all d dimensions, this yields the routing complexity, which is $O((d/4)(n^{1/d}))$.

8.2.4 Hierarchically Structured Overlay Networks

A hybrid P2P overlay network holds the features of unstructured and structured overlay network at the same time. There are always two types of approaches to building hybrid P2P overlay networks. The first involves adding structured overlays on top of unstructured overlays. For example, one can embed Chord on top of a Gnutella overlay. The flooding search protocol in Gnutella is suitable for locating highly replicated files (hays). However, for queries of rarely replicated files (needles), flooding requires a long time for a response, or even gives no response. A hybrid P2P overlay is often built by maintaining the primary components while getting the secondary components for free. In Figure 8.16, the route table of a Pastry node is selected. The global random peer information is provided by the leaf set of the Pastry node. In this way, the overhead of Pastry is related to locate nearby nodes to fill the routing tables.

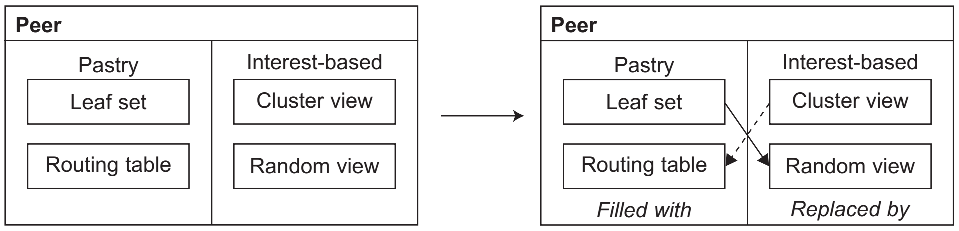


FIGURE 8.16

The intuition of the hybrid P2P overlay construction by keeping the primary components and getting the secondary components for free.

Example 8.8 KaZaA Network Using Super Nodes to Form a Backbone Overlay

KaZaA connects peers in a two-layer hierarchical structure (see Figure 8.17). The upper layer consists of *super nodes* that are more powerful in terms of CPU, bandwidth, and stability. Lightweight user nodes are assumed at the second layer, and are connected to a parent super node. A super node acts as a hub for user nodes. A user node uploads to its parent super node the filename, content hash, and file description of each requested file. Thus, each super node maintains the file indices for all of its children nodes. The super node provides each child with a super-node list, which contains up to 200 super nodes in the system.

In 2006, more than 3 million peers of KaZaA shared more than 3,000 terabytes of digital content, including MP3s, videos, and games. At one time, more than 50 percent of Internet traffic was generated by KaZaA users. On average, it takes 56 minutes to connect to the end super node. Each super node has about 100 to 150 children. In total, the system has roughly 30,000 super nodes. Each super node has TCP connections to 30–50 other super nodes. The system is not fully connected. Among the super nodes there is 0.1 percent connectivity with an average of 23 minutes to connect two super nodes.

Each super node acts like a mini-Napster hub by tracking the content and IP addresses of its descendants. The super node is represented with its IP address, port number, workload, timestamp, and similar data. This list is cached at the user node for use in bootstrapping and failure recovery. KaZaA encrypts all its signaling traffic. A query from a user node is first submitted to its parent super node. If any files match the query, the super node directly answers the query. Otherwise, the super node forwards the query to other super nodes using TTL-limited flooding. KaZaA uses the UUHash to identify files uniquely. Nodes request files from multiple peers using this unique identifier in order to enable parallel downloading.

The UUHash hashes the first 300 KB of a file using MD5, yielding a 128-bit hash value. It applies a smallhash function to 300 KB blocks at file offsets 2^m MB, with m being an integer from 0 until the offset reaches the end. The smallhash generates a 32-bit hash value. The MD5 hash and this smallhash are concatenated, yielding a 160-bit hash as the file identifier. The UUHash is simple since it only hashes a fraction of the file. This weakness allows users to easily modify large parts of the file without changing the UUHash identifier. However, copyright infringement is still a serious problem. Both DDoS attacks on super nodes and content pollution have been encountered in the past.

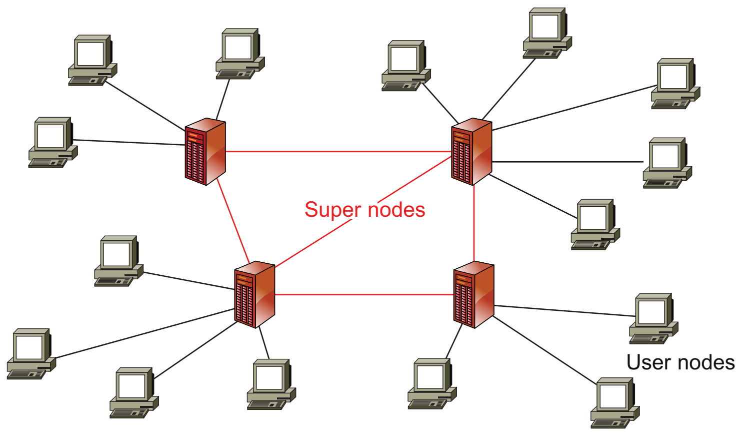


FIGURE 8.17

The KaZaA architecture built with super nodes forming a backbone overlay.

The system automatically switches to a new download server when the current server fails to deliver. The system also gives the estimated download time to users. Super nodes are relatively stable; thus node churns have a limited effect. The bandwidth on a super node is low, even in relaying VoIP traffic. The two-layer structure takes heterogeneity in capacity into account and enables network proximity awareness by assigning user nodes to nearby super nodes. Flooding is performed on top of the upper layer, thus improving scalability. However, it still does not provide real guarantees on search scope and search time.

Example 8.9 Embedding Chord on Top of a Gnutella Network

Figure 8.18 shows the hybrid architecture that results after embedding Chord on top of Gnutella. For any query, the DHT-based Chord responds almost within the same time as long as there is at least one file to satisfy the query. Thus, Chord performs well for needle queries.

For queries to hays, it takes longer to respond. In Gnutella version 0.6, a portion of the powerful nodes (ultrapeers) are selected as hubs for ordinary nodes (leaf peers). Queries for hays are forwarded using flooding, in order to be responded to, while queries for needles follow the Chord search protocol. Hence, queries to hays and needles can respond faster.

8.2.4.1 Overlay Complexity Analysis

Table 8.4 summarizes the functional features and complexity characteristics of five P2P networks. All five are self-organized and cannot support node heterogeneity. Data locality is not preserved in

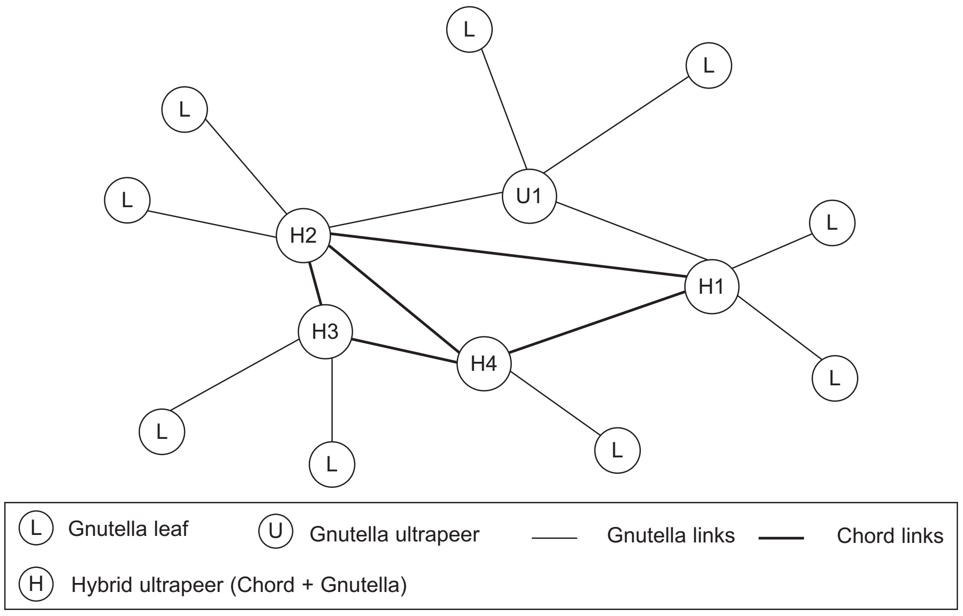


FIGURE 8.18

A hybrid P2P architecture of Gnutella and Chord.

Table 8.4 Complexity Properties of Six P2P Overlay Networks

Features	**Gnutella** [60]	**Chord** [45]	**CAN** [40]	**TerraDir**	**SkipNet**
Overlay structure	Random	Circular identifier space	Multi-dim. Cartesian space	Tree data hierarchy	Distributed skip list
Design parameters	N/A	r: Immediate successors	d: dimensionality	N/A	N/A
Routing hops	$O(N)$	$O(\log N)$	$O(d \cdot N^{1/d})$	$O(h)$, tree height	$O(\log N)$
Node state	$O(1)$	$O(\log N)$	$O(2\,d)$	$O(1)$	$O(2\log N)$
Node join	$O(1)$	$O(\log N)$	$O(2\,d)$	$O(1)$	$O(\log N)$
Load balancing and reducing query hotspots	N/A	Via consistent hashing and virtual nodes	Partitioned space, zone reassignment, caching and replication	Via caching and replication	Consistent hashing the suffix of object name
Fault handling	Flooding to other valid neighbors	Successors list, application layer replication	Multiple realities, overloading zones	Node replication, routing to next best prefix node	Redundant neighbors at level 0, survive failures at boundary
Network proximity	No	Proximity routing to finger's successor with lowest latency	Topology-aware overlay and proximity routing	No	Proximity neighbor selection via P-Table and C-Table
Support node heterogeneity	No	No, but might through virtual nodes	No, but might split zones proportional to node capabilities	No	No
Self-organization	Yes	Yes	Yes	Yes	Yes
Guarantee determinate searching results	No, only within broadcast scope	Yes	Yes	Yes	Yes
Data locality	No	No, destroyed by uniform hashing	No, destroyed by mapping to hyper space via uniform hashing	Yes	Yes
Applications	N/A	CFS [27]	Data-centric sensor networks	Resource discovery	Global event notification service

Courtesy of Min Cai, EE 657 Report, University of Southern California, 2006.

all five overlay structures. With uniform routing, both Chord and CAN destroy data locality easily. Network proximity is supported only in Chord, CAN, and SkipNet. They all (except Gnutella) guarantee determinate search results. Gnutella applies a random graph. Chord applies a chordal identifier space using fingers to identify immediate successors.

CAN applies a multidimensional Cartesian space. TerraDir has a tree data hierarchy and SkipNet uses a distributed skip list. The major attraction of Gnutella is its constant node state and node joining time, but its routing time is the longest, $O(n)$. Flooding in Gnutella is an expensive way to search, but it supports fault tolerance. The rest all support fault tolerance using replication techniques. $O(\log n)$ routing speed is achieved in Chord and SkipNet. The routing time in TerraDir is proportional to the tree height (h) applied. The dimension d of CAN affects its performance greatly.

8.3 ROUTING, PROXIMITY, AND FAULT TOLERANCE

In this section, two fundamental techniques related to P2P systems will be discussed, namely routing and locality awareness. In P2P systems, peer nodes are organized in decentralized and self-organizing overlay networks. The routing algorithms, which indicate how to get to a specific node from any node, should be distributed and only relay on nodes in the local view of the whole system. Locality awareness, also called proximity awareness, is a mechanism to let peer nodes be prone to connect with physically close peers, in order to reduce average overlay link latency and backbone network bandwidth consumption. The P2P overlay network is ad hoc. Thus, these mechanisms are required to tolerate and recover peer failures and disconnections.

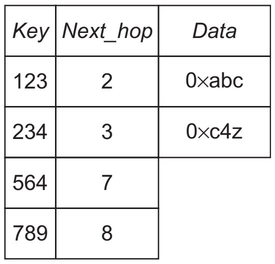
8.3.1 Routing in P2P Overlay Networks

Unstructured P2P overlay networks give no indication of how to locate a specified node since node neighborhoods are randomly selected and are not subject to constraints. The routing algorithms in unstructured P2P overlay networks are always based on blind flooding. When receiving a message, a peer node simply forwards it to all its neighbors except the one from which it receives the message. Suppose the number of nodes is n and the average node degree (i.e., the number of neighbors) is k. The number of messages used for locating a node is $n(k-1)$ on average.

The number of messages used for locating a node is $n(k-1)$ on average. Since a message follows the shortest path from the source to the destination, the route complexity, which is defined as the expected number of overlay hops for a message to reach a specified node from any node, is directly determined by the overlay graph diameter. Overlay networks based on small-world graphs have a short diameter, allowing for fast routing. Freenet [10] is such an overlay network.

Example 8.10 Freenet—An Unstructured P2P Network

Freenet is a decentralized storage service, which provides good anonymity and censorship resilience. Its architecture belongs to unstructured overlay networks since it allows any node to be a neighbor to any other node. Each node contributes a storage space to the system. Nodes and files are mapped to the



Key	Next_hop	Data
123	2	0×abc
234	3	0×c4z
564	7	
789	8	

FIGURE 8.19

An example of a datastore stack of a node in the Freenet.

same ID space using hash functions. A file is encrypted and stored at a node with the ID closest to the file's ID. Files may be split into multiple chunks, with additional chunks added to provide redundancy. Each chunk is handled independently over distributed nodes.

Freenet uses a key-based route protocol to find requested files. Both publishers and readers of files are hidden from the outside. Data files that are stored on a node form a datastore stack, which is illustrated as an example in Figure 8.19. The stack is divided into two parts. The top part stores the keys, node references of the next hop, and data, while the bottom part only stores keys and node references. When receiving a new file insertion request, a node puts the information at the top of the stack. If the stack is full, LRU (least recent used) is used to swap out old entries. Unlike Gnutella, in Freenet, even if the file publishers leave the network, files are still available.

The routing in Freenet is similar as the routing used in IP. The datastore stacks are taken as routing tables. For a request of a file with key *k*, if the key can be found in its routing table, the node stops forwarding the request and replies to the upstream requester. Otherwise, the node forwards the request to the node indicated by the next hop reference of the entry with the closed key to *k*. The horizon of a request is limited by the TTL. If data is not found when the TTL expires, a failure is reported. If data is found, it is cached on each node along the request path.

The same process is used for insertion of data: The data to be inserted is cached on each node along the path to the target node. Thus, a request to a data file will cause the file to be replicated more. The request originator may connect the node that answered the request, while discarding old nodes that are least recently used. This is called path folding, and would cause a clustered structure. Path folding makes the simple routing protocol used in Freenet well performed. Anonymity is achieved by randomly modifying the source of messages as they traverse the network.

DHT-based structured P2P overlay networks have strict and predefined structures, which facilitate message routing. Each node is associated with a globally unique identifier. The routing process to a destination node gradually reduces the identifier's distance from the message processing node to the destination node. Although different structured overlays have different routing algorithms, the routing complexities are $O(\log n)$, where n is the number of nodes. We describe the routing algorithms for three representative structured P2P overlays in the following examples.

Example 8.11 Table-Lookup Search in a DHT-Based Chord Network

Chord overlay networks resort to finger tables to efficiently route messages. When receiving a message whose destination is *y*, a node *x* forwards it to the node in its finger table which is the closest one to *y* in the identifier space. To this end, node *x* checks its finger table in a reverse order. The first node whose identifier is smaller than y is chosen as the next step node. For example, in Figure 8.20, seven nodes are populated in a 16-node chord structure. The minimum routing path from node 0 to node 8 is via node 6.

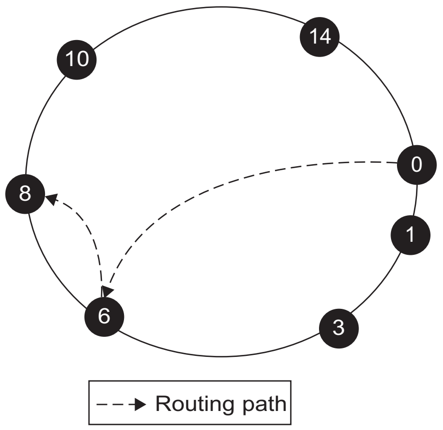
The routing complexity for the chord is $O(\log n)$ for an n-node system. Suppose that node x selects the i-th entry in its finger table as the next step node z. Then node z is at least 2^i away from x in the identifier space. Since the identifier of the $(i+1)$-th entry of x's finger table is not smaller than y, node y is at most 2^{i+1} away from x in the identifier space. Therefore, the distance between node x and z is at least equal to the distance between nodes z and y, which means node z halves the distance between nodes x and y. After $\log n$ steps, the distance between the source node to the destination is equal to $2^m/n$, where m is the identifier length. Since the populated nodes are randomly distributed, an average of one node exists in each of the range $2^m/n$.

■

8.3.2 Network Proximity in P2P Overlays

P2P overlay networks are logical structures over IP networks. While randomized overlay graphs are fault tolerant and have a low diameter, they ignore the network proximity information on the IP networks. This causes two physically close peers to be far from each other in the overlay network, while two nodes that are close in the overlay network are far from each other in the physical network. This scenario is denoted as topology mismatching. The problem also exists in structured P2P overlay networks. For the CAN network, (Example 8.6, also Figure 8.15a), if nodes C and E belong to the same autonomous system, while A and G belong to another, the messages from C to G traverse the links between the two domains three times, which inevitably prolongs the delay and consumes unnecessary network bandwidth.

As to the structured P2P overlay networks, node neighborhoods are so rigid that it is hard to optimize them in terms of network proximity awareness. There are three types of approaches to



Gap	Span	Scsr
1	(1,2)	1
2	(2,4)	3
4	(4,8)	6
8	(8,1)	10

Node 0 finger table

FIGURE 8.20

An example of the table lookup search in a Chord overlay network.

exploit network proximity in structured P2P overlays: *geographic layout, proximity routing, and proximity neighbor selection.*

8.3.2.1 Geographic Layout

Node identifiers are no longer random. Physically close nodes are assigned adjacent identifiers, and therefore are close in the overlay network. Since many structured P2P overlay networks (e.g., Chord, Pastry, Tapestry) rely on the randomness of node identifiers to guarantee robustness and performance, this approach is only suitable for some special overlays, such as CAN. For example, the landmark binning scheme can be leveraged to build proximity-aware CAN overlay networks. In this scenario, m publicly accessible nodes are chosen as landmark nodes. Each node measures the distance to these nodes and sorts them in terms of increasing distances, which yields an order of landmark nodes. The order represents the "bin" the node belongs to.

Physically close nodes are likely to belong to the same or similar bins. The coordinate space of a CAN overlay network is divided into $m!$ zones, one zone per bin. A new node randomly selects a point from the zone which associates with its bin and joins. This achieves the goal that physically close nodes are close in overlay networks. However, nodes are no longer randomly distributed on the coordinate space. It is possible that a small portion of the nodes occupy most of the identifier space. These nodes are vulnerable to overloading.

8.3.2.2 Proximity Routing

There are always multiple paths between any two nodes in P2P overlay networks. For the CAN in Example 8.7 (Figure 8.15b), the node C can follow either path C-A-B or path C-F-B to get to node B. The idea behind proximity routing is to select the path with minimum delay. However, in distributed overlay networks, finding this path is as complex as a Traveling Salesman Problem, which is NP-hard. A heuristic way is that each node selects the closest neighbor as the target node to forward messages. However, this may increase routing path length. For example, in a Chord overlay network, if each node selects its successor as the target node, the routing complexity is $O(\text{n})$. Therefore, the candidate target nodes are limited to a small number. The best example overlay for proximity routing is the Pastry network. We introduce the Pastry architecture in the following example.

Example 8.12 Pastry—A Proximity-Routing Overlay with Nested Groups

The Pastry is a structured overlay, which has an identifier space coded with high-radix digits. The purpose is to subdivide the peer nodes into nested groups with layers of subgroups. those nodes in the same innner-most subgroup are very close neighbors, known as *proximity subgroups.* The overlay using high radix identifiers to search the nodes in the nested groups or subgroups. For example, a 128-bit identifier space can specify a very large P2P network with identifier viewed as base-16 numbers (base 2^b for $b=4$) like 65A1FC04. The identifiers are divided into groups with 16 subgroups per each group.

Each data key is stored in a node with the closest identifier. Node addressing defines the subgroups. Nodes in the same subgroup know each other's address, Thus, the search time can be significantly reduced for being in the same proximity group. Furthermore, each node knows the IP address of one delegate node in some of the other groups. Each node needs to know the IP addresses of all up nodes in its inner group. The proximity routing is applied to perform fast search in $\log_{2^b} n$ steps, where $2^b = 16$.

For $n = 2^{16}$ nodes or 64 K nodes, it takes at most 4 routing steps ($\log_{16}$ 32K) to route the messages among 16 groups, compared with $\log_2$64K = 16 steps needed in using Chord.

With node failures, the Pastry is definitely faster than Chord and CAN. During failure recovery, the Pastry may take $O(n)$ routing hops at the worst case. Each routing table entry points to a node close to the local node in the proximity space, among all node identifiers with the appropriate prefix. In a way, Pastry has the best locality properties. The expected distance traveled by a message is within a small constant in the proximity space. Pastry is suitable to use in building very large P2P networks. The fast lookup search within proximity space makes Pastry very suitable to use in very large-scale P2P networks. Details of Pastry can be found in [21].

■

8.3.2.3 Proximity Neighbor Selection

Nodes choose physically close nodes as their neighbors in overlay networks. In structured P2P networks, the identifiers of a node's neighbors should be subject to some constraints. For example, in Pastry networks, the nodes in the *i*-th row of a node's routing table should share *i*-digit long prefix with the node identifiers. This approach can also be applied in Chord overlay networks. Instead of fixing the *i*-th entry of node *x*'s finger table to the node whose identifier is $(x + 2^i) \bmod 2^m$ $(0 \leq i < m$, m is the length of identifiers), node *x* selects the physically closest node whose identifier falls in the range $[x + 2^i, x + 2^{i+1}]$.

This change does not affect routing complexity. Landmark clusters with the dimension reduction algorithm is a wildly used scheme. Each node measures the delay to *m* landmark nodes which are publicly accessible. The measurement yields an *m*-dimension vector which acts as the node position in an *m*-dimension landmark space. The idea behind this is that physically close nodes are also close in the landmark space. Then, space filling curves are used to map the *m*-dimension vectors to a space with lower dimensions. For Chord, the vectors are mapped to one dimension number. The space-filling curves preserve the proximity information. The mapped results are stored on the overlays based on their specific protocol, which facilitates archival of close nodes. The landmark cluster scheme is coarse-grained and cannot distinguish two physically close nodes. Thus, RTT measurement can be further combined.

Compared with structured overlay networks, unstructured networks are easier to optimize in terms of network proximity, because the node neighborhood relationships are more flexible in unstructured overlay networks. The optimization operation is a process of deleting high-cost overlay connections and adding low-cost ones. The key problem is how to identify high/low-cost overlay connections and cut/establish them in a decentralized way. A simple approach is that each node actively measures link costs (i.e., latencies) to its neighbors. The high-cost overlay links are disconnected if they are redundant. Nodes probe one another to locate closer nodes and build the overlay connections.

8.3.3 Fault Tolerance and Failure Recovery

This section covers fault tolerance and failure recovery techniques developed for upholding P2P operations under faulty and failure conditions.

8.3.3.1 Faults and Node Failures

Node failures also have a vital impact on P2P overlay connectivity. The links to a node would be disconnected if the node failed. The extent of the impact on overlay connectivity caused by node failures depends on the graph property of the overlay and the degree (i.e., number of neighbor links) of the faulty node. For instance, in a power-law graph-based P2P overlay network like Gnutella 0.4, failures of a subset of randomly selecting nodes would not fragment the overlay into disconnected portions. However, failures of some high-degree nodes would easily shatter the overlay into pieces of disconnected components.

DHT-based structured P2P systems implement $O(\log n)$ lookup time by building $O(\log n)$ neighbor links for every node. The overlay neighbors of a node should be selected according to the predefined rules, rather than selected randomly. For example, in Chord, the i-th entry in a node x's finger table should be the successor of $(x+2^i) \bmod 2^m$, where $0 \leq x \leq m-1$ and m is the length of the ID. Structured P2P systems rely on these neighbors to accelerate the lookup service, rather than guarantee overlay connectivity. Thus, a neighbor failure would delay resource lookups. Overlay connectivity is guaranteed in other ways. In Chord, each node maintains $O(\log n)$ immediate successors to achieve this goal.

The aforementioned node faults are simply considered crash failures or silent failures, which means the fault nodes just do not respond to the messages sent to them (i.e., they stay silent). Another type of fault is the Byzantine failure, which means the fault nodes behave inconsistently when communicating with others. Byzantine failures are often triggered by attacks and discussed in the context of DHT-based P2P systems. DHT-based P2P systems are more vulnerable in the case of Byzantine failures. An adversary can elaborately select the IP addresses for Byzantine peers and where the peers join the network in order to put them in critical positions. When processing lookup requests, the Byzantine peers provide inconsistent answers for the same request, which delays the lookup service or even makes the system useless. On the other hand, unstructured P2P systems flood the lookup requests. Thus, there are multiple disjointed paths from the query source to the nodes which satisfy the requests.

Example 8.13 Failure Effects on Real-Life P2P Networks

The impact of node failures on a P2P network depends on the system architecture and the protocols applied. In a central index server model such as Napster, the failures of the index servers would pull down the entire system, since the resource lookup service would be completely down, while in a pure P2P model such as Gnutella 0.4, failures of individual nodes would have very little impact on lookup services because all nodes would keep the indexes by themselves.

In a super-node hybrid model like KaZaA, failure of the super nodes would cause index information loss for those peers that connect with it. This implies that the resource lookup service will be downgraded or shut down. In a DHT structured P2P network such as Chord, the resource index information is distributed almost evenly among all the nodes. Thus, the failures of the nodes inevitably deteriorate the performance of the lookup service, but do not necessarily pull down the whole system. The resource sharing service between two nodes would be interrupted if either node fails. P2P networks resort to service backup and resource relocation to resume file-sharing services.

8.3.3.2 Analysis of Failure Recovery

Since failures occur frequently, P2P systems demand safe recovery from node failures. Chord leverages stabilization operations to fix node failures. Another method is to let a node periodically check the aliveness of a random neighbor in its finger table. In the case of an i-th entry failure in a chordal ring, node x relocates the successor to $(x + 2^i) \bmod 2^m$, where $0 \le i \le m - 1$. Nodes also periodically verify their immediate successor. In Pastry, a failure of a routing table entry is detected when forwarding lookup requests. As to the method for new node selection from backup node lists, the *random replacement* strategy performs better than others in terms of churn if the distribution of node duration time is highly skewed. Here, the new node is randomly chosen from several nodes to replace the faulty one. The idea behind this is that the strategy is biased in favor of longer duration nodes.

8.3.3.3 Fault Tolerance Techniques

Unlike traditional client/server model-based distributed systems, no node in P2P systems has a global view. Nodes rely on themselves to detect faults and recover from failures in a fully decentralized fashion. P2P overlay networks ensure stable throughput by providing redundancy. For example, each node has multiple neighbors for service providers. The question is how many redundancies are adequate. In Chord overlay networks, the successor lists are the key component for connectivity and routing correctness.

Each node maintains a sequential successor list with length $O(\log n)$, where n is the number of nodes. If the current successor fails, a node replaces it with the first live entry in its successor list. Suppose that node fails independently with probability 1/2. The probability that all the nodes in a node's successor list fail is $(\frac{1}{2})^{\log_2 n} = 1/n$. Thus, with high probability every node is aware of its immediate live successor, which is sufficient to guarantee overlay connectivity.

In unstructured P2P overlay networks, node neighborhoods are random and flexible. Nodes are not required to maintain restrictive structures. Thus, they are considered more reliable than structured P2P overlays. The reliability of structured P2P networks means predefined structures are maintained, while reliability demands at least one path between any pair of nodes.

In an overlay network based on a random graph with n nodes, if the probability p with which there is a link between each pair of nodes is $(\log n + c + o(1))/n$, the overlay network is connected with probability $e^{-e^{-c}}$. This result can be easily extended to a scenario where there are node and link failures. Suppose link failures are independent of each other with probability q. In this case, to achieve the connectivity probability $e^{-e^{-c}}$, the probability is $p = (\log n + c + o(1))/(1 - q)n$.

8.3.3.4 Failure Analysis

If any node fails with a probability $e^{-a} (a > 1)$ independently, the base ring is connected with a high probability that is more than $1 - n^{1-a}$, where e is the base of the natural logarithm. The results confirm that REM overlay is as resilient as the one in which each node maintains $\log n$ random neighbors in the presence of random failures. In the preceding analysis, the failure resilience of overlay networks is evaluated under random failures. However, in a proximity-aware overlay, the failures of neighbor nodes may be correlated. Physically close nodes may fail concurrently due to network congestion. Further work should evaluate the fault tolerance for proximity-aware overlays.

8.3.4 Churn Resilience against Failures

A P2P network often faces disturbance from high node churns. The churns come from unexpected node joins, departures, or failures. Node failures or sudden departures impose a great adverse effect on network performance. This is caused by unavailability of data stored on the faulty nodes. The nodes requesting service from the faulty nodes need to relocate the services elsewhere. Hence, a P2P overlay network should be designed to be fault tolerant and churn resilient. This section is devoted to overcoming the difficulties associated with churn and failure conditions that are frequently encountered in a P2P network. A *churn-resilient protocol (CRP)* is presented to alleviate churn effects.

8.3.4.1 Churn-Resilient Protocol (CRP)

A solution to increase fault tolerance in proximity-aware overlays is to guarantee connectivity of critical links, such as random links among clusters or nodes. This can be achieved by providing backup nodes to build the critical links. For example, in GoCast [47], each node maintains a node cache as backups. Another example is CRP [24] for construction of proximity-aware and churn-resilient P2P overlay networks.

The topology of the CRP-enabled overlay design looks like a ring augmented with extra chord links. However, the overlay topology differs from Chord in that the extra links are not fixed fingers from nodes. Instead, the chord links are established among nodes satisfying certain proximity properties. Both network proximity and capacity proximity are applied in CRP. Network proximity is measured by the latency or closeness of two nodes in physical IP networks. This proximity enables construction of low-latency chord links. Capacity proximity is measured by the closeness of nodes with respect to node capacities. Capacity proximity allows the grouping of nodes with similar capacities into clusters.

8.3.4.2 Overlay Construction Using CRP

The initial base ring with three nodes is shown in Figure 8.21(a). The ring is assumed to be unidirectional with clockwise links shown by dotted edges. We assume that all ring links are weighted with one unit. Node A's successor and predecessor are B and C, respectively. The fourth node D is inserted into the interval between A and C in Figure 8.21(b). A chord link weighted with 0.7 is added between nodes B and D. In Figure 8.21(c), the fifth node E is inserted between nodes C and D. Two new chord links are added, with weights of 0.6 and 0.8.

A newly inserted node may establish new chord links with other nodes by removing some existing chord links. An existing chord link is removed when it has higher weight than the added new links. In Figure 8.21(d), a new node F is inserted between nodes A and D. Two new chord links are added to connect F with nodes B and E replacing the old link between B and E in Figure 8.21(c). The newly added chord links must have lower weights than those of the links being replaced. The overlay is dynamically adapted by replacing higher-weight chord links with lower ones, without changing any node's degrees.

In Figure 8.21(e), the links between B and D, A and E in Figure 8.21(d) are replaced with two new lower-weight links between A and D, B and E. Figure 8.21(f) shows the overlay after the departure of node E in Figure 8.21(d). The links associated with E are removed. The base ring is always connected as nodes keep their correct successors. To improve the connectivity, the CRP borrows the idea of the successor list from Chord. Each node maintains a list of $O(\log n)$ successor

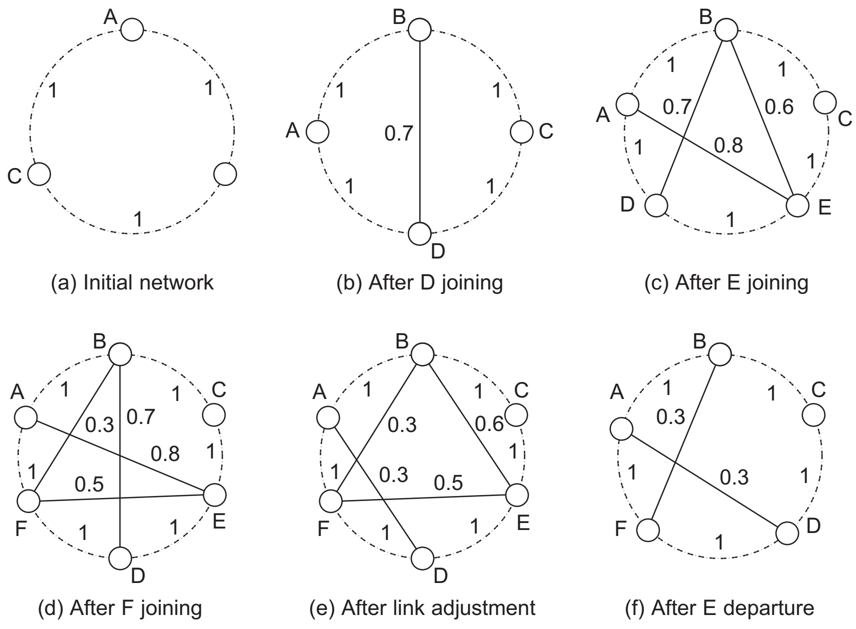


FIGURE 8.21

An example CRP-enabled overlay network design in six steps.

(*Courtesy of Li, Xie, Hwang, and Li [24]*)

nodes down the base ring. If a node detects a faulty successor, it replaces that successor with the first live node in the list. In Figure 8.21(e), for example, node C keeps two successor nodes E and D. If node E fails or leaves, C takes D as its new successor.

8.3.4.3 Performance of CRP Scheme

Figure 8.22 depicts the relative performances of CRP and four other tree-based algorithms. The notation CRP:q denotes a CRP system with a message replication ratio q. CAM-Chord [56] builds on Chord overlays. It takes the heterogeneity of node capacities into account. ACOM [9] implements data dissemination by combining scope-limited flooding and random walks on overlay networks. GoCast [47] and Plumtree [23] apply a tree-based gossip scheme for data dissemination. It can be found that CRP outperforms all other designs in terms of delivery time due to proximity awareness. The message replication in CRP is close to 0 and can be controlled by adjusting q.

The CRP-enabled overlay networks consider both network proximity and capacity proximity. It uses a hybridization of scope flooding and tree traversing to disseminate data promptly. The overlay networks provide redundant proximity links for tree structures for fault resilience. The churn-resilient CRP-enabled overlay networks provide adequate backup links for the tree structure. Once a node x receives a departure notification from its parent or detects the failure of its parent by heartbeat messages, it actively switches its tree parent. The node x prefers its proximity chord neighbor as the new parent to save on delivery time.

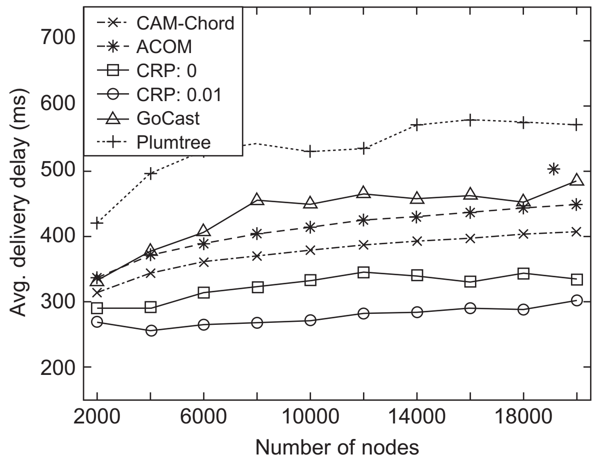


FIGURE 8.22

Average delivery time of five data dissemination schemes over a P2P network.

(*Courtesy of Z. Li, et al.* [24])

8.4 TRUST, REPUTATION, AND SECURITY MANAGEMENT

The anonymous and dynamic nature of peer activities makes P2P networks very vulnerable to abuses by selfish and malicious peers. For example, most P2P file-sharing networks (e.g., Gnutella) consist of autonomous peers with special self-interests. There is no efficient way to prevent malicious peers from joining open networks. To encourage resource sharing among peers and combat malicious peer behavior, trust and reputation management becomes critical to maintain the normal operation of P2P networks.

Without trust, peers will have little incentive to contribute their resources to other peers. The peers may hesitate to interact with unknown peers due to the concern of receiving corrupted or poisoned files or being exploited by malware. Identifying trustworthy peers demands reputation systems in commercial P2P applications, such as for online shopping, auctioning, content delivery, pay-per-transaction, etc.

8.4.1 Peer Trust and Reputation Systems

This section characterizes the trust model needed to establish harmony among the peers. We will define trust matrix for aggregating peer reputations in P2P reputation system development.

8.4.1.1 Peer Trust Characteristics

There are two ways to model the trust or distrust among peers: namely *trust* and *reputation*. Trust refers to the belief of one peer on another, based on his direct experiences with the peer. Reputation is a collective opinion on a peer by other peers based on recommendations. To face the reality of an open P2P network, one has to assume that the participating peers in a P2P system do not trust one another unless proven otherwise.

A fair *reputation system* is thus needed to establish the trust or distrust among peers, based on recorded historical behaviors of the peers. The purpose is to distinguish good peers from bad ones through a scientific screening process. The quality of the reputation system is primarily indicated by its accuracy and effectiveness in updating periodically.

8.4.1.2 Trust Matrix for Computing Reputations

Consider an overlay network for global reputation aggregation in a P2P system over five peers, denoted by nodes N1, N2,…, N5 of a directed graph shown in Figure 8.23. The trust relationships among the peers are represented by a square *trust matrix* $M(t) = [m_{ij}(t)]$, where $m_{ij}(t)$ is the *local score* issued by node *i*in evaluating node *j* at time *t*. All trust scores are fractions in the range (0, 1) with 0 meaning no trust (or no contact) and 1 for 100 percent trust. Therefore, trust is relatively measured by these fractions. For the five-node network, the following trust matrix at certain time *t* is achieved. Note that all row sums are 1. A zero entry, $m_{ij}(t) = 0$, means node *i* does not evaluate node *j* for lack of direct contact. The diagonal scores are all zero, meaning no peer evaluates himself.

$$M(t) = \begin{bmatrix} 0 & 0 & 0 & 0.2 & 0.8 \\ 0.6 & 0 & 0 & 0 & 0.4 \\ 0 & 0.7 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & 0 & 0 \\ 0.9 & 0 & 0 & 0.1 & 0 \end{bmatrix} \tag{8.2}$$

The edge labels are the *local scores* issued between all (source, destination) pairs. The fraction value inside each node in Figure 8.23 is the *global reputation score* of that peer at time *t*. This global score is resulted from aggregating all local scores issued by all peers toward the peer being evaluated. However, all local scores must be weighted by their own global reputations. In another words, the global reputation is the weighted sum of all local scores. For clarity, all the global reputation scores are normalized so that their sum is always a "1". For example, the

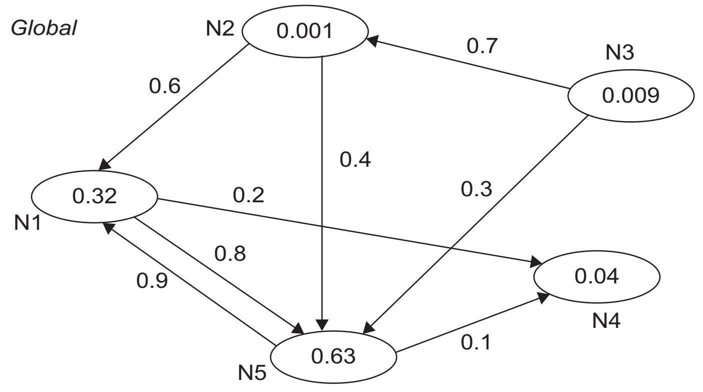


FIGURE 8.23

A directed graph showing the trust relationship among five peers in a P2P network.

global scores of the five peers are represented by a *reputation vector* of five components which add up to 1.

$$V(t) = \{v_1(t), v_2(t), v_3(t), v_4(t), v_5(t)\} = \{0.32, 0.001, 0.009, 0.04, 0.63\} \tag{8.3}$$

8.4.1.3 Reputation Systems

A *reputation system* calculates the global reputation score of a peer by considering the opinions (i.e., *feedback*) from all other peers that have interacted with this peer. After a peer completes a transaction(e.g., downloading a music file), the peer will provide his feedback for other peers to use in future transactions. By making the reputation scores publicly available, peers are able to make informed decisions about which peers to trust.

The eBay reputation system is a simple and successful one, since it has a centralized authority to manage all user feedback scores. However, in an open and decentralized P2P system, peers will not have any centralized authority to maintain and distribute reputation information. Instead, most existing P2P reputation systems calculate global reputation scores by aggregating peer feedback in a fully distributed manner. Building an efficient P2P reputation system is a challenging task due to several intrinsic requirements of large-scale P2P systems.

To measure peer reputations, one can set up an evaluation system. After each transaction, the two participant parties evaluate each other by giving an honest score. This is like what we do in online auction systems such as eBay. Each peer's reputation can be retrieved from the voting system. However, not every peer is trustworthy. The scores from malicious peers are useless, while the scores from more trustworthy peers are more useful. This advises us to assign different weights to votes based on the voting peers' reputations. It should be noted that a peer's reputation may be different from other peers. The reputations can be expressed with a reputation matrix.

8.4.1.4 Global Reputation Aggregation

As global reputation scores are aggregated from local feedback, the distribution property of feedback plays a significant role in the design of an efficient reputation system. Surprisingly, most previous work either ignored the distribution of peer feedback or assumed an arbitrary random distribution, which could be misleading. To achieve global reputation scores, each peer calculates its own parts and all peers cooperatively compute a global matrix. For example, at time $t+1$, the global reputation score of node N5 is computed as follows:

$$\begin{aligned} v_5(t+1) &= m_{15}(t) \times v_1(t) + m_{25}(t) \times v_2(t) + m_{35}(t) \times v_3(t) \\ &= 0.8 \times 0.32 + 0.4 \times 0.001 + 0.3 \times 0.009 = 0.2573 \end{aligned} \tag{8.4}$$

Similarly, the remaining global scores for the other four peers are computed to yield the following updated global reputation vector:

$$\begin{aligned} V(t+1) &= \{v_1(t+1), v_2(t+1), v_3(t+1), v_4(t+1), v_5(t+1)\} \\ &= \{0.5673, 0.0063, 0, 0.1370, 0.2573\} \end{aligned} \tag{8.5}$$

This vector is unnormalized. Dividing each peer score by the total sum of all component scores, one can obtain the following *normalized global reputation vector*. Note that the sum of the five peer reputation scores should be 1 in the normalized vector.

$$\begin{aligned} V(t+1) &= \{v_1(t+1), v_2(t+1), v_3(t+1), v_4(t+1), v_5(t+1)\} \\ &= \{0.5862, 0.0065, 0, 0.1416, 0.2657\} \end{aligned} \tag{8.6}$$

8.4.1.5 Design Objectives of Reputation Systems

The following six key issues should be addressed in the design of a cost-effective P2P reputation system:

- **High accuracy** To help distinguish reputable peers from malicious ones, the system should calculate the reputation scores as close to their real trustworthiness as possible.
- **Fast convergence speed** The reputation of a peer varies over time. The reputation aggregation should converge fast enough to reflect the true changes of peer behaviors.
- **Low overhead** The system should only consume limited computation and bandwidth resources for peer reputation monitoring and evaluation.
- **Adaptive to peer dynamics** A peer joins and leaves an open P2P system dynamically. The system should adapt to these peer dynamics instead of relying on predetermined peers.
- **Robust to malicious peers** The system should be robust to various attacks by both independent and collective malicious peers.
- **Scalability** The system should be able to scale to serve a large number of peers in terms of accuracy, convergence speed, and extra overhead per peer.

8.4.1.6 Three Representative Reputation Systems

The PeerTrust system [53] was developed at Georgia Institute of Technology. The system computes the reputation score of a given peer as the average feedback weighted by the scores of the originators. Five trust attributes were suggested in this system. The EigenTrust system was developed at Stanford University [20], which aggregates reputation information using the eigenvector of the trust matrix over the peers. EigenTrust relies on a good choice of some pretrusted peers. This assumption may be overly optimistic in a distributed computing environment because the pretrusted peers may change with time. The PowerTrust reputation system [57] was developed at the University of Southern California. PowerTrust will be covered in Section 8.4.3. Table 8.5 compares these three systems in four technical aspects.

8.4.2 Trust Overlay and DHT Implementation

This section introduces a trust overlay concept for fast aggregation of trust information in global reputation calculations. Then a DHT implementation of the reputation system is described.

8.4.2.1 Trust Overlay Network (TON)

This is a virtual network on top of a P2P system. A TON is represented by a directed graph in Figure 8.24. The graph nodes correspond to the peers. The directed edges or links are labeled with the feedback scores between two interacting peers. The feedback score is issued by a peer (source of the link) for the service provided by the interacting peer (destination of the link). For example, node N_5 after downloading music files from nodes N_2 and N_7 issues the feedback scores, 0.7 and 0.3, to the two provider nodes, respectively. If a node gets more than one service from the same provider, this consumer generates a newly updated score after each transaction.

Table 8.5 Comparison of Three P2P Reputation Systems

Reputation System	Local Trust Evaluation	Global Reputation Aggregation	Implementation Overhead	Scalability and Reliability
EigenTrust at Stanford University [20]	Uses sum of positive and negative ratings	Uses pretrust peers to compute global scores from trust matrix	Moderate overhead experienced in assigning score managers and in messaging for global score aggregation	Limited scalability and reliability if pretrust peers leave
PeerTrust at Georgia Tech [53]	Normalized rating on each transaction	Peer calculates trust score over five factors in a distributed manner	Moderate overhead experienced in global score calculation over five factors and on the establishment of the trust manager	Partially scalable and resistant to malicious peers
PowerTrust at USC [57]	Uses Bayesian method to generate local trust scores	Distributed ranking of power nodes and LWR strategy to aggregate global reputation scores	Low overhead in using locality-preserving hashing to locate power nodes. Global aggregation time drops sharply with look-ahead random walk strategy applied	Highly scalable and robust with dynamic peer join and leave and malicious peers

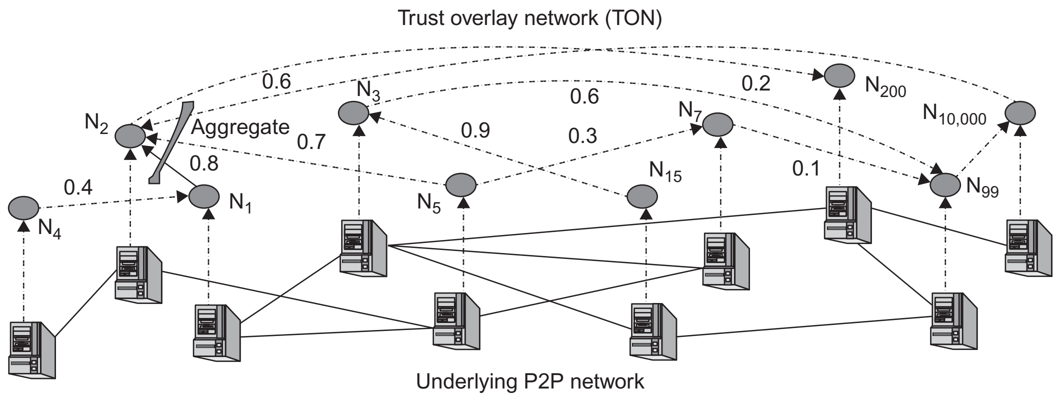


FIGURE 8.24

A trust overlay network for a P2P trust managment, where the edge is weighted by the peer feedback score for the service provided. The global reputation of a peer (node) is calculated by the weighted sum of all local trust scores received on the incoming edges.

(*Courtesy of R. Zhou and K. Hwang [57]*)

Our system can incorporate different methods to generate feedback scores, such as Bayesian learning. In a TON, every node keeps feedback scores for its neighbors. Because every peer has its own criteria to generate feedback scores, the feedback will be normalized to *local trust scores*. Each node N_i is rated with a *global reputation score* v_i. This global reputation is aggregated from local trust scores weighted by the global reputations of all in-degree neighbors. For example, the global reputation score of N_2 could be calculated by weighting three incoming local scores (0.8, 0.7, 0.6) from N_1, N_5, and $N_{10,000}$, respectively: $v_2 = 0.8\ v_1 + 0.7\ v_5 + 0.6\ v_{10000}$. Given $v_1 = 0.04$, $v_5 = 0.0007$, and $v_{10000} = 0.000005$, then $v_2 = 0.8 \times 0.04 + 0.7 \times 0.0007 + 0.6 \times 0.0000005 = 0.032 + 0.00049 + 0.00003 = 0.032493$. It is interesting to note that node N_1 has a much higher reputation score v_1, compared with v_5 and v_{10000} in this example. Thus, node N_1 carries more weight in the global reputation aggregation process.

8.4.2.2 DHT Implementation

Basically, distributed reputation ranking requires two different hash overlays. One assigns peers to their score managers and the other ranks the peers by their global reputation scores. Figure 8.25 presents a five-node system built on top of a chord with a 4-bit circular hash space. Node N_{15} is the score manager of node N_2 whose global reputation is 0.2. Node N_{15} hashes the reputation value 0.2 using a simple LPH function $H(x) = 32x$. The resultant hash value is 6.4. Node N_{15} sends out *Sort_Request*{ *key* = 6.4, (0.2, N_2, N_{15}) } message, which is routed to node N_8. Node N_8 stores the triplet (0.2, N_2, N_{15}), since it is the successor node of hash value 6.4.

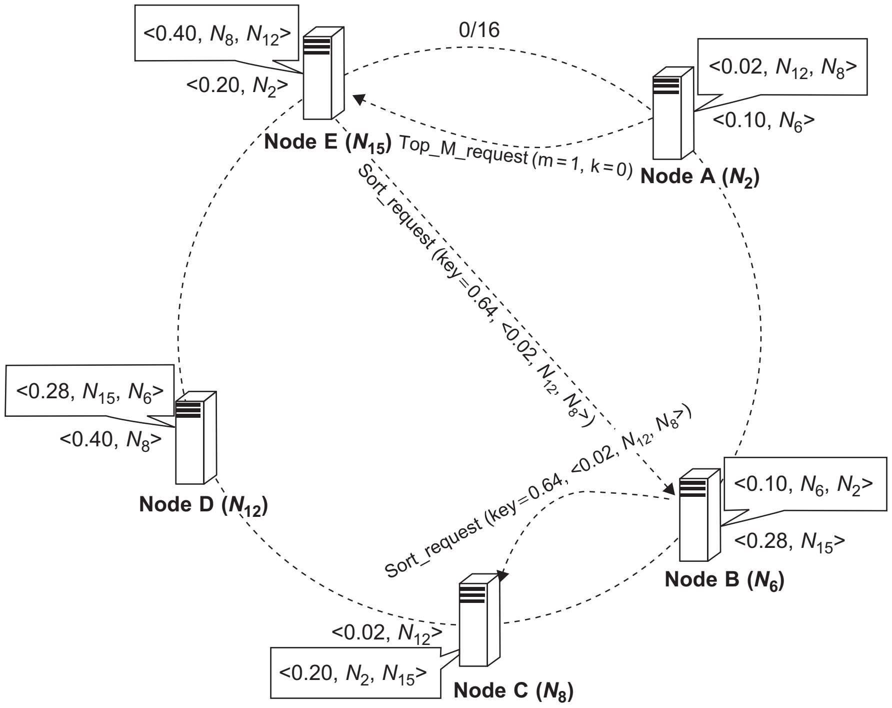


FIGURE 8.25

Distributed reputation ranking using the locality-preserving hash function over a DHT-based P2P system.

(*Courtesy of R. Zhou and K. Hwang [57]*)

Node N_2 is the successor node of the maximum hash value 15, so node N_2 initiates the process to find m power nodes. Node N_2 is responsible for the hash values in the union of two ranges [15, 16] ∪ [0, 2]. Since it has no corresponding triplets within the range [15, 16], it stores zero triples with the highest reputation values, that is, $k = 0$. Therefore, it sends a *Top_m_Request*($m = 1,\ k = 0$) message to its predecessor node N_{15}, which finds its stored triplet with value 0.4 being the highest one. So, node N_8 is the most reputable node in this example system. Multiple LPH functions could be used to prevent cheating by the malicious peers.

8.4.3 PowerTrust: A Scalable Reputation System

Figure 8.26 shows the major building blocks in a PowerTrust system. First, a TON is built on top of all peers (nodes) in a P2P system. All peers evaluate one another, whenever a transaction takes place between a peer pair. Therefore, all peers send *local trust scores* to one another frequently. These scores are considered the raw data input to the PowerTrust system. The system aggregates the local scores to calculate the global reputation score of each participating peer. All global scores form a *reputation vector*, $V = (v_1, v_2, v_3, \ldots, v_n)$, which is the output of the PowerTrust system. All global scores are normalized with $\Sigma_i\, v_i = 1$, where $i = 1, 2, \ldots, n$ and n is the TON network size.

The system is built with five functional modules, as shown in Figure 8.26. The *regular random walk* module supports *initial reputation aggregation.* The *look-ahead random walk (LRW)* module is used to update the reputation score periodically. To this end, the LRW also works with a *distributed ranking module* to identify the power nodes. The system leverages the power nodes to update the global scores' reputation. PowerTrust achieves high aggregation speed and accuracy, robustness to resist malicious peers, and high scalability to support large-scale P2P applications.

8.4.3.1 Reputation Convergence Overhead

The *convergence overhead* is measured as the number of iterations before the global reputation convergence. Convergence means the distance between two consecutive reputation vectors is smaller

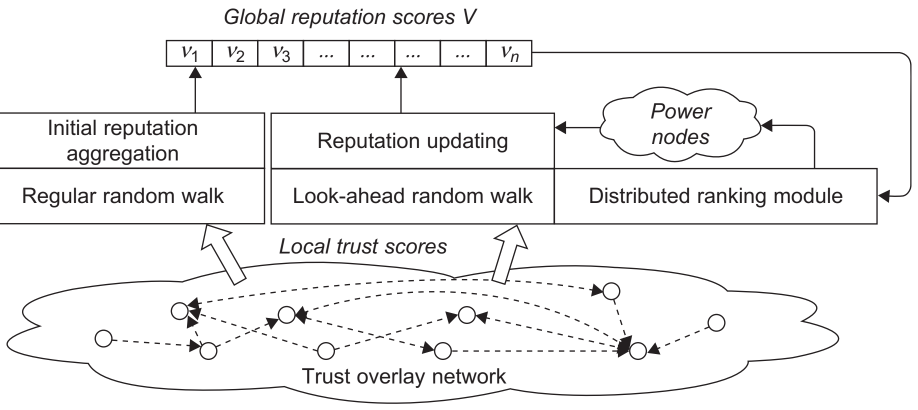


FIGURE 8.26

Functional modules in the PowerTrust system for trust score collection and global reputation aggregation.

(*Courtesy of Zhou and Hwang [57]*)

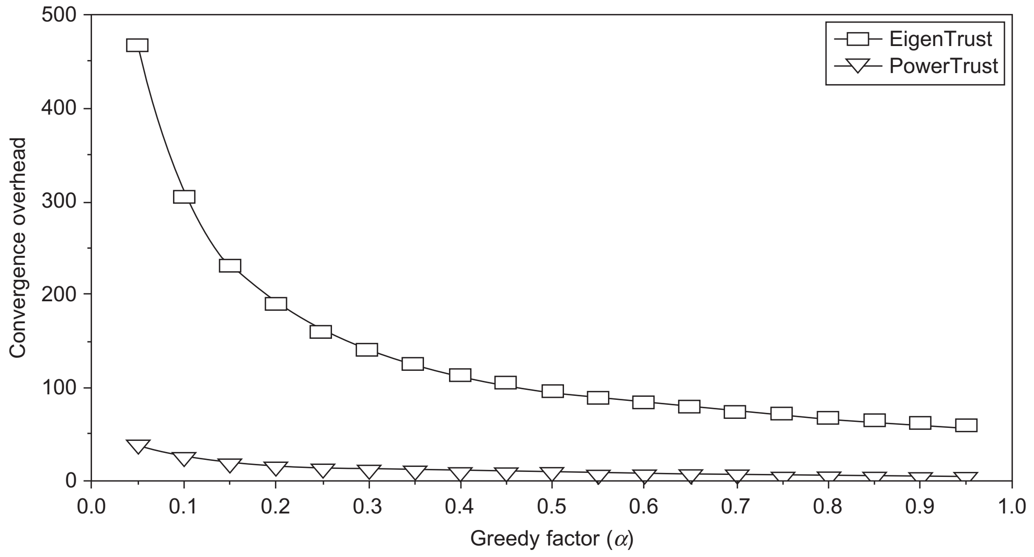


FIGURE 8.27

Convergence overhead of two reputation systems in a P2P network.

(*Courtesy of Zhou and Hwang [57]*)

than the threshold. The EigenTrust approach relies on a few pretrusted nodes to compute the global reputations. They assumed that some peers are known to be trustworthy, essentially among the very first few peers joining the system.

For fairness, the same number of power nodes equal to that of the pretrusted nodes used in EigenTrust is chosen. Figure 8.27 shows the convergence overheads for these two reputation systems. Here, the power nodes in PowerTrust and the pretrusted node in EigenTrust are allowed to leave freely. A sharp drop in iteration count in using PowerTrust to a flat small number less than 50 is observed, when α increases from 0.15 to 1, while EigenTrust still requires more than 100 iterations to converge. The EigenTrust system overhead can reach as high as 400 iterations as the system increases to 4,000 nodes.

The EigenTrust system converges very slowly. The system cannot guarantee its convergence, when the pretrusted nodes are allowed to leave the system freely. In the PowerTrust system, the power nodes are reelected after each aggregation round. Based on the distributed ranking mechanism, the score managers of the departing power nodes notify the system to replace them in a timely manner with other, more qualified power nodes. The decrease in computation overhead means a significant traffic reduction on the network, and less work for all peers involved. The low overhead in using the PowerTrust system makes it attractive for performing highly scalable P2P applications.

8.4.3.2 Query Success Rate

The PowerTrust system was applied in simulated P2P file-sharing experiments. There are more than 100,000 files in the simulated P2P systems. The number of copies of each file in the system is determined by a content Power-law distribution with $\beta = 1.2$. Each peer is assigned with a number

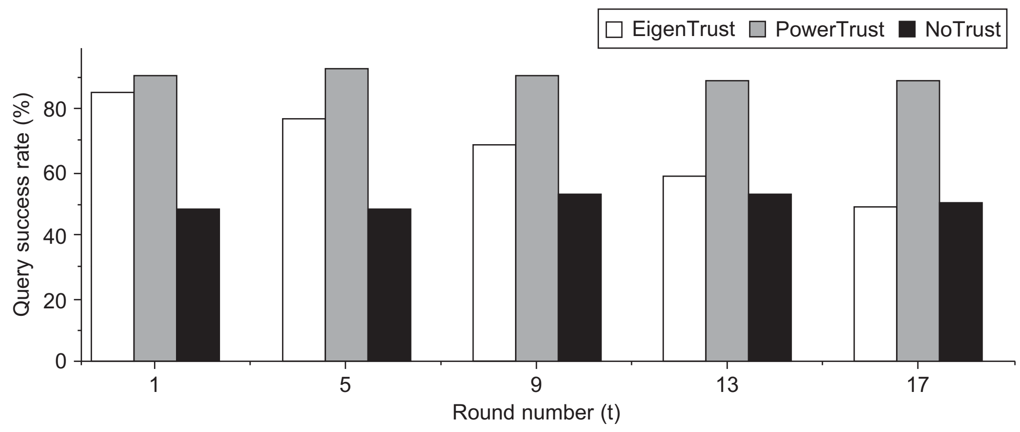


FIGURE 8.28

Query success rate in distributed file sharing using two reputation systems to compensate for the free departure of power nodes or pretrusted peers.

(*Courtesy of Zhou and Hwang [57]*)

of files based on Sarioiu distribution. At each time step, a query is randomly generated at a peer and completely executed before the next query/time step. The query distribution determines which file each query searches for. The queries are ranked according to their popularity. A Power-law distribution with $\beta = 0.63$ for queries ranked 1 to 250 and $\beta = 1.24$ for the remaining lower ranking queries is used. This distribution models the query popularity in existing P2P systems. When a query for a file is issued, the file owner with the highest global reputation is selected to download the desired file.

The query success rate is measured by the percentage of successful queries over the total number of queries issued. Every node may respond to a query with unauthentic files. For simplicity, this behavior is modeled as inversely proportional to the node's global reputation. Both cases of allowing or disallowing power nodes or pretrusted nodes to leave the system are considered. The case of a *no-trust system*, meaning no trust management in the P2P system, is also evaluated. The no-trust system randomly selects a node to download the file without considering reputation. Figure 8.28 shows the results without the departure of power nodes or pretrusted nodes.

There are 1,000 queries issued after each round of global reputation aggregation. The query success rate of PowerTrust is maintained at 90 percent after just one round of reputation aggregation. The query success rate of EigenTrust drops from 85 percent to 50 percent after several aggregation rounds. This is because the pretrusted nodes may leave the system randomly. The EigenTrust query success rate drops to a 50 percent low, equal to that of a no-trust system, after 17 rounds of reputation aggregation.

8.4.4 Securing Overlays to Prevent DDoS Attacks

The security of a P2P network may be in doubt if the peer nodes behave maliciously to attack other innocent peer machines. The attackers could be hidden in the P2P group. They can view or modify packet routing or can collude with other peers to stage the attacks or to steal the contents. What can be done to hinder the attacks is crucial to achieve harmonious operation in a P2P network. Four classes of network attacks often occur in a P2P network:

1. A P2P network could enter churn condition, if large number of peers join or leave the network rapidly or randomly. Churn could create inconsistent behavior or resources deadlocks.
2. Distributed denial of service (DDoS) attacks are caused by flooding attacks on a target node.
3. Routing attackers attempt to re-route messages in order to steal contents or to stage DDoS attacks.
4. Storage/retrieval attacks are caused by attackers to prevent the delivery of the requested data.

To become resilient to network churns, one can force nodes to sign all messages so that inconsistency can easily detected. To cope with DDoS attacks, one can replicate the content and spread over the network. The SOS example below presents a more sophisticated solution. For *identity spoofing attacks* caused by a peer node to use other node's identity or to deliver bogus contents. One solution is to insist that all peers must have certificates signed by a trusted authority. Another solution to prevent proofing is tying the identity with protected IP address or send query to verify the address.

To prevent routing attacks, one solution is to use a converging policy iteratively so that each hop moves closer to the destination in the ID space. Another solution is to provide multiple paths to re-route messages around the attackers. Onion routing is also used to hide the identify of the originator of the final destination. This done by encryption of the incoming message and the next hop information recursively using public keys. This way, any node only knows who was the immediate sender and who is the next receiver. To combat storage/retrieval attacks, one can replicate the data object to enhance availability.

Example 8.14 SOS Scheme to Prevent DDoS Attacks [21]

SOS is an architecture for *Secure Overlay Services*, which was proposed by Keromyts, et al. [21] in 2002. The main goal of SOS to allow a majority of legitimate or authenticated users to communicate with a secure checkpoint node, where DDoS attacks can be stopped to be forwarded to the final target machine. This is made possible by making the target address difficult to replicate or legitimate users may be mobile using changing IP addresses. Good examples of SOS include FBI, police or bank officers, who always check with their central database.

A SOS system will protect legitimate users and innocent target machines from DDoS attacks. At the same time, the attackers are blocked or caught. To this end, we need to screen good users into a pre-approved class. Attackers should not allowed to attack a bound number of distributed end systems. The construction of an SOS system requires to perform filtering, to hide the proxy as *secret servlets,* to build reliable overlays with hidden proxies, and to advertise the trustworthy access points for users to join the P2P system. The SOS can be also assisted by using the Chord with SOAP servers or adding redundancy in SOAP servers. More details on the effectiveness of SOS preventing DDoS attacks can be found in [21].

8.5 P2P FILE SHARING AND COPYRIGHT PROTECTION

P2P technology enables files to be shared among peers in a distributed way freely. The client first performs search operations to locate peers holding the files the client wants. The client downloads the files directly from these provider nodes. The ultimate goal is to deliver content to all requesters as fast as possible. Caching of P2P content is helpful to improve content downloading speed and traffic locality. This section reviews fast file search methods. Replicas and consistency techniques to enhance delivery speed and success rate are discussed as well.

8.5.1 Fast Search, Replica, and Consistency

For P2P file-sharing applications, search algorithms play the most import role. A search algorithm determines which nodes hold the specified files. The performance of a search algorithm is evaluated by two metrics: query path length and message overhead. Query path length is measured by the average number of nodes a query message traverses before reaching the destination, while message overhead is measured by the average number of query messages forwarded for one search operation.

Search algorithms for file-sharing applications based on structured P2P overlays and unstructured P2P overlays are different. In a structured P2P overlay application, keys of data objects and nodes are mapped to the same identifier space and the data object information is stored on the node that holds the identifier zone within which the key is mapped. Search algorithms are like routing algorithms in overlay networks. In an unstructured P2P overlay application, however, each node often only holds the information of the data objects that it shares. The query message has to traverse a lot of nodes until it reaches the destination. In the following paragraphs, we discuss algorithms for structured and unstructured P2P overlay applications.

In structured P2P overlay networks, each node is responsible for a portion of the identifier space. For example, in Chord, a node is responsible for the range from its predecessor to the node on the ring identifier space. The information of the data objects whose keys are mapped to the identifier range held by node x is stored on x. Node x is called the successor of the key. Searching a key is equal to routing the query message to the successor of the key. Hence, the performance of search algorithms is similar to that of routing algorithms, which means a key is always located in $O(\log n)$ hops with the overhead of $O(\log n)$ messages, where n is the number of nodes in the application.

Search in applications based on unstructured P2P overlay networks is more complicated. Basically, the search algorithms can be classified as blind search and informed search. Blind search algorithms are for applications where nodes do not maintain the metadata of files shared by others, while informed search algorithms are for applications where nodes store some information of the files shared by others to facilitate the search. Blind search algorithms are often referred to as flooding algorithms.

In flooding algorithms, when a query message is first received, a node forwards it to k randomly selected neighbors, until the TTL of the message shrinks to 0. The query messages are forwarded round by round like a wave spreading from the source. Flooding algorithms inevitably bring considerable message overhead. A simple method to reduce overhead is to limit the TTLs of query messages. However, this would increase the probability that objects far away from the query nodes cannot be located. Various algorithms have been proposed to reduce message overhead, such as expanding ring and random walk.

In expanding ring algorithms, a query node is associated with a schedule $\{t_1, t_2, \ldots, t_i\}$. Initially, the TTL of a query message is set to t_1 by the query node. If the query cannot be satisfied, the TTL grows to t_2. Eventually, the TTL grows to the maximum value t_i. The successive retries facilitate overhead reduction at the cost of a slight increase in query path length.

Random walk algorithms reduce overhead from another aspect. A query node deploys *k* independently random walkers. Each walker carries a query message and walks on the overlay graph, until the desired object is located. Random walk algorithms suffer from the TTL selection problem just like flooding algorithms do. However, in random walk algorithms, each round only increases the number of visited nodes by *k*. Therefore, it is reasonable for walkers to periodically check with the query node to determine whether to continue or stop. Random walk algorithms generally outperform expanding ring algorithms and save on message overhead by two orders of magnitude at a cost of a slightly increase in query path length compared to flooding algorithms.

The principles of scalable blind search algorithms for unstructured P2P overlays are summarized as follows. First, the algorithms should adopt adaptive termination. Simply limiting the TTL values does not work well as that would adversely affect search scope. Second, the number of visited nodes should grow slowly with the increase in search round. This is because only a small portion of the nodes are required to be visited. If the number of visited nodes grows quickly (e.g., exponentially in flooding algorithms), most of the messages generated at the last several hops are duplicated.

The message overhead incurred by blind search algorithms can be greatly reduced if nodes keep some information regarding the files shared by others. Informed search algorithms are designed for this purpose. Each node keeps some information regarding the files shared by others and the information is used to guide the forwarding of query messages. The idle case is that each node keeps all the files shared by others, and thus the query path is only one hop long. However, this is not feasible because it requires not only unaffordable storage, but also lots of synchronize overhead among peers.

Search performance can also be enhanced through overlay topology adaptation. In Gnutella, one can take heterogeneity of node capacity into account by assigning more loads on high-capacity nodes. High-capacity nodes adaptively build more overlay links and keep the file indices of the nodes one hop away. Since random walk is biased toward the high-degree node, most queries are forwarded to the high-capacity nodes and satisfied there. Another topology adaptation is to consider semantic interests of nodes. Semantically relevant nodes are organized into the same semantic clusters on overlays. Queries are forwarded to semantically relevant clusters for answers. The idea behind this is that if nodes are semantically relevant, it is likely that they are relevant to the same queries. BitTorrent is the most popular P2P content distribution application. It uses swarming content distribution.

Example 8.15 BitTorrent File-Sharing Network

BitTorrent is a P2P content distribution protocol that is widely used in the current Internet. It is reported by Ipoque that BitTorrent content distribution systems account for 27 percent to 55 percent of Internet traffic. A BitTorrent system leverages the upload bandwidth of participant peers to achieve high system capacity. The nodes downloading the same file constitute a torrent. There is no communication between different torrents. A typical BitTorrent system consists of four components as shown in Figure 8.29.

Seeds are the nodes that have already downloaded the whole file. There is at least one initial seed in the system at startup. A node that has not gotten the whole file is called a downloader. The tracker tracks the nodes' information, including the address and download process information. The tracker also knows the seeds in the system. The web server hosts the torrent file, which stores the metadata of the file to be distributed and the tracker's information.

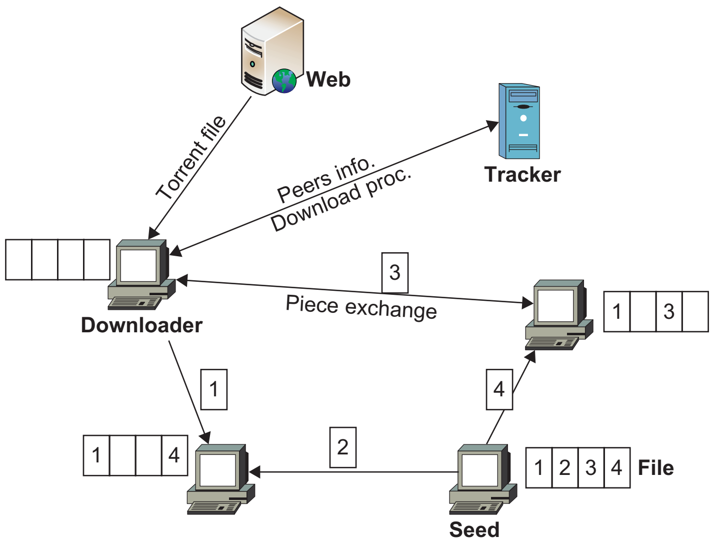


FIGURE 8.29

The BitTorrent system architecture.

(*Courtesy of "BitTorrent Protocol Specification"; www.bittorrent.org/protocol.html, 2006*)

The file is divided into pieces of equal size, ranging from 32 KB to 4 MB, with a typical size of 256 KB. The publisher of the file creates a hash for each piece using the SHA-1 hash function. The hash for each piece is used to verify the integrity of the piece during downloading and is stored in the torrent file. A file piece may be further divided into blocks. In this case, each piece is a trading unit, while a block is a request unit. Only after all the blocks of a piece have been downloaded can this piece be publicly shared with others.

A new node first gets the torrent file from a web site. By parsing the file, it gets the tracker's address and asks the tracker for peers' information. Usually, the tracker responds with a peer list containing 50 randomly selected peers. The number of TCP connections initialized by a peer is limited to 35 and the total number of connections of a peer is limited to 55. If ever a client fails to maintain at least 20 connections initialized by it, it recontacts the tracker to obtain additional peers. To update the tracker's global view of the system, active nodes periodically (every 30 minutes) report their states to the tracker or at the time of joining or departure.

Each peer has a bit-vector indicating the pieces' availabilities on it, one bit for each piece. The bit-vectors are exchanged between neighbors. If peer *y*'s neighbor *x* finds that *y* has a piece that it does not have, *x* sends an "interest" message to *y* to request that piece. The decision by *y* of whether and when to send the requested piece to *x* is based on an incentive strategy called *Tit-for-Tat (TFT)*. All connections are choked by default. Each peer periodically (every 10 seconds) updates its download rates from all of its neighbors, and chooses four neighbors.

The TFT is an incentive strategy to combat against free riders, the peers that contribute nothing or very limited upload bandwidth. A peer also uses an optimistic unchoking strategy to unchoke a uniformly chosen peer at random every 30 seconds, which enables the nodes that have no or very limited file pieces to start up. When unchoked by a neighboring peer, a node selects the file piece that is the least replicated among its neighbors. This selection is called *rarest first (RF)* (locally) piece selection. However, for a new peer that has no file piece, it randomly chooses a piece to download. The RF piece selection scheme is used to enable random distribution of file pieces on overlay networks, which facilitates the diversities of pieces on neighboring peers.

Figure 8.30 shows the hierarchical structure of multiple swarms built up in a BitTorrent network. A swarm is a collection of peer machines that form a virtual cluster under the coordination of special trackers. Interswarm traffic is possible if the local swarm cannot satisfy a user request. Multiple trackers exchange updated directory information to the global view consistently. The idea is quite similar to that of the KaZaA architecture shown in Figure 8.11. The purpose is to upgrade management efficiency and shorten search time for users.

By convention, the name of a torrent file has the suffix (.torrent). Torrent files have an “announce” section, which specifies the URL of the tracker, and an “info” section, containing the suggested names for the files, their lengths, the piece length used, and an SHA-1 hash code for each piece, all of which are used

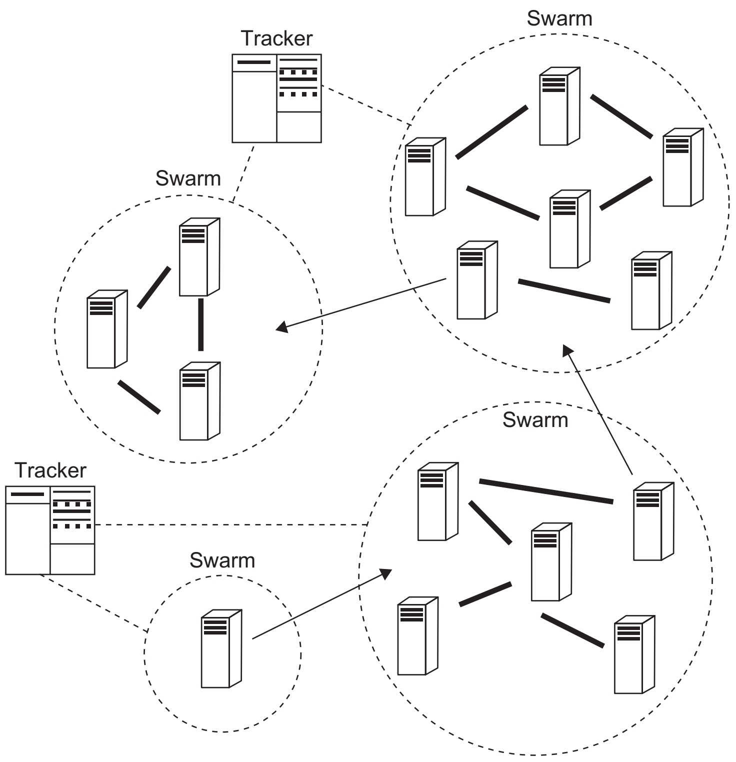


FIGURE 8.30

An operational view of a BitTorrent system built with multiple swarms of resource nodes under the coordination of special trackers.

by clients to verify the integrity of the data they receive. The tracker maintains lists of the clients currently participating in the torrent. Alternatively, in a decentralized tracking which works without a tracker, every peer acts as a tracker. Azureus was the first BitTorrent client to implement such a system through the DHT. An alternative and incompatible DHT system, known as Mainline DHT, was later developed and adopted by BitTorrent, μTorrent, and others.

■

8.5.1.1 Replica and Consistency

Replication is critical to improving search performance in P2P file-sharing applications. The number of replicas for an object is affected by the popularity of that object. Intuitively, more replicas should be created for highly popular objects. The number of nodes that hold the desired objects has a direct impact on search performance. This is quite intuitive in that as a greater number of desired objects scatter over the overlays, it becomes easier to find them.

A question related to replication concerns maintaining consistency of replicas. If the data object is modified or updated by a replica node (i.e., the node maintains a replica of the object), the updated content should be propagated to all the other replica nodes. Consistency maintenance schemes provide support for applications dealing with frequently updated contents. Two key issues of consistency maintenance concern how to track the replicas of an object and how to propagate the updated contents to the replica nodes. The centralized scheme dedicates one node (often the root node of the object) to keep the information of all the replica nodes and relies on this node to forward the updated content. However, it lacks scalability.

In the locality-aware and distributed consistency maintenance scheme [25], the nodes that store the replicas of an object constitute a replica group. The centralized scheme is sufficient for the replica group with fewer replica nodes. Thus, our focus is on larger replica groups; for example, replica groups of popular objects. An auxiliary hierarchical overlay network is established for each replica group: The upper layer is DHT-based and consists of powerful and stable replica nodes of the key, and the ordinary replica nodes at the second layer attach to physically close upper-layer nodes. Figure 8.31 shows the structure, where the upper layer is based on a Chord ring structure.

When an updated replica emerges on a node, the corresponding upper-layer node dynamically initializes a *d*-ary source-specific tree on top of the upper layer by continuously partitioning the DHT identifier space. Suppose upper-layer replica node *x* receives an update request from one of the ordinary replica nodes that attaches to it. Initially, node *x* holds the whole identifier space represented by the Chord ring structure. It divides the identifier space into *d* equal parts. The first node along the clockwise direction within each part is selected as the representative node of that part. All the representative nodes constitute the root node *x*'s child node set. Each identifier part is further partitioned into *d* subparts of equal size, and so on, until there is only one replica node in the identifier part.

The updated content is propagated along the tree structure in a top-down fashion and the upper-layer replica node takes charge of forwarding the updated content to the ordinary replica nodes that attach to it. Since the upper-layer node and the ordinary replica nodes that attach to it are physically close, the updated contents are transferred quickly and network bandwidth consumption is greatly reduced. The tree structure is destroyed from bottom to top after the update operation completes, in order to save on maintenance overhead. Now, let's briefly analyze the time (measured by logical hops on the overlays) and the control overhead required by an update operation.

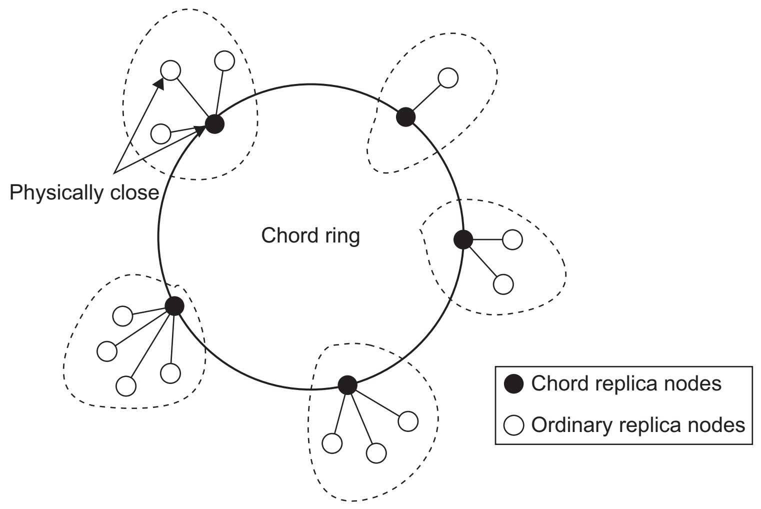


FIGURE 8.31

Auxiliary structure for a replica group.

8.5.2 P2P Content Delivery Networks

This section applies P2P technology to digital content networking and distribution. First, three generations of *content delivery networks (CDNs)* are reviewed. Then we discuss how P2P technology can enhance the QoS of CDNs.

8.5.2.1 Three Generations of CDNs

Table 8.6 compares three generations of CDNs in the past 30 years. The early days of online distribution systems follow the client/server architecture using FTP or HTTP services. This generation is limited by the processing power of a single server. For distributing small files, this approach is still acceptable. Good examples include the Apache, GetRight, and CuteFTP services. The second generation is marked by the use of multiple surrogate servers and large-scale distribution networks mainly for video streaming and software distribution.

The Akamai CDN is a good example of a large-scale CDN as illustrated in Example 8.14. Finally, the P2P file-sharing networks mark the appearance of the third generation of CDNs. BitTorrent and eMule are good examples. P2P networks definitely are more cost-effective than the first two generations. However, severe abuses on copyright violations in P2P file-sharing systems have hindered the use of open P2P networks for commercial content distribution. We discuss the online piracy problem in the next two subsections. Selective content poisoning poses a possible solution to this problem.

Example 8.16 A Global Content Distribution Network

In 2004, more than 3,000 companies used CDNs. They spent more than $20 million monthly. CDN providers have doubled their revenue almost every year since then. In 2005, worldwide CDN revenue grew to $450 million in the delivery of news, film, sports, music, and video content. Figure 8.32 shows the

Table 8.6 Three Generations of Content Distribution Networks

System Generations	Network Architecture	Download Bandwidth	Content Applications	Example Systems and Web Sites
First generation: client/server system	Clients access server using FTP and HTTP	Limited by single-server bandwidth	Small files with limited downloads	Apache (www.apache.com/) GetRight (www.getright.com/) CuteFTP (www.cuteftp.com/)
Second generation: CDNs	Clients connect to multiple servers	Load balancing among servers	Video streaming, virus signatures	Akamai (www.akamai.com/) SyncCast (www.synccast.com/)
Third generation: P2P file-sharing system	P2P overlay network	Low; peers are not secured and are unreliable	Large files, speed is not critical	BitTorrent (www.bittorrent.org/) eDonkey (www.edonkey.com/) eMule (www.emule.org/)

Courtesy of Lou and Hwang [31].

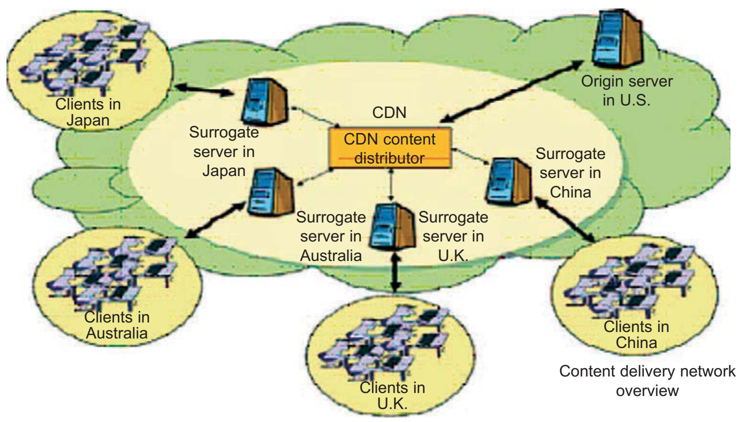


FIGURE 8.32

The concept of a global CDN using surrogate servers in major user regions or countries.

(*Courtesy of Pallis and Vakali [37]*)

conceptual architecture of a global-scale CDN, modeled after the Akamai Technologies system. By 2006, the Akamai CDN provided 80 percent of overall content delivery services. At that time, more than 12,000 surrogate servers were used in various parts of the world in more than 1,000 networks in 62 countries.

This number is now much greater. Surrogate servers are used as cache centers for distributing digital content more cost-effectively in terms of transmission, storage, access speed, and fault tolerance requirements. Each surrogate center is responsible for distributing the requested content to all end users in each country or region. The CDN content distribution center coordinates the traffic among the servers. For copyright protection reasons, open P2P technology is not used here for commercial content distribution. ■

A P2P network does not require many expensive servers to deliver content. Instead, content is distributed and shared among the peers. P2P networks feature improvements over conventional CDNs in terms of content availability and system scalability. Electronic publishing was hindered by the rapid growth of copyright violations. The major source of illegal P2P content distribution lies in peer collusion to share copyrighted content with other peers or pirates.

Digital watermarking is often considered for digital copyright protection. Digital watermarking is injected into a content file so that when a pirated copy is discovered, authorities can find the origin of the piracy via a unique watermark in each copy. In a P2P network, all peers are sharing exactly the same file (if it is not poisoned), which effectively defeats the purpose of watermarking. Thus, watermarking is not a suitable technology for P2P file sharing. By subdividing a large file into smaller chunks, P2P content delivery allows a peer to download multiple chunks from different sources. File chunking increases availability and decreases download time. Table 8.7 summarizes three major P2P CDN families.

P2P networks in the same family have some common features. They are variants or descendants of their predecessors: *BitTorrent*, *Gnutella*, and *eMule*, respectively. These families are distinguished primarily by the file-chunking or hashing protocols applied. At the time of this writing, none of these P2P networks has been built with satisfactory support for copyright protection. The BitTorrent family applies the strongest hashing at the individual *piece* or chunklevel, which is most resistant to poisoning. The Gnutella family applies file-level hashing, which is easily poisoned. The eMule family applies part-level hashing with fixed chunking.

Our analytical and experimental results show that the eMule family demonstrates a moderate level of resistance to content poisoning. The ability to detect and identify poisoned chunks is different in the three P2P network families. The BitTorrent family keeps clean chunks and discards poisoned chunks.

Table 8.7 File Chunking and Hashing Schemes in P2P Content Networks

P2P Networks	BitTorrent Family	Gnutella Family	eMule Family
Chunking Scheme	Divides files into fix-sized chunks (256 KB), called pieces	Peers negotiate the chunk size at runtime, 64 KB/chunk	Divides into 9,500 KB parts, each has 53 (180 KB) chunks
Hash Distribution	SHA hashing at piece level, embedded in index file	SHA hashing applied to entire file to generate a unique file ID, no chunk-level hashing	MD-4 hashing at part level, peers exchange hash set to detect corrupted contents
Poison Resistance	Poison detected at piece level, each is handled independently	Poison detected only after downloading the entire file, heavy overhead if poisoned	Poison detected at part level, works if part hash set is not poisoned
Download Policy	Keeps clean and discards poisoned pieces, repeats download until all chunks are clean	Repeats downloading entire file until all chunks are clean, a time-consuming policy	Keeps clean and discards poisoned parts, downloads until all parts are clean
Example Networks	BitTorrent, Snark, BitComet, BNBT, BitTyrant, Azureus, JTorrent, etc.	Gnutella, KaZaA, LimeWire, Phex, Freenet, BareShare, Swapper, Ares, etc.	eMule, aMule, Shareaza, iMule, Morpheus, eDonkey, FastTrack, etc.

The Gnutella family must download the entire file before any poisoned chunks can be detected. Because of part-level hashing, the eMule family will either keep or discard the entire part of 53 chunks.

8.5.2.2 Content Delivery Methods

P2P content distribution applications take advantage of the abundant computing resources at the edge networks to distribute content. Nodes which are interested in the same content form a P2P overlay network. The data content is sent out from sources, stored, and forwarded among peers on the overlay networks. Basically, there are three approaches to distributing data content: flooding-based, tree-based, and swarm-based. In flooding-based approaches, data content is flooded on the overlay networks. There are too many duplicated deliveries to be tolerated.

The tree-based approach follows a tree structure and delivers the content along the tree. Since peer nodes are likely to join, leave, and even fail at any time at will, it is a great challenge to maintain the tree structure. One can also build multiple trees. Data content is partitioned into several chunks. Each chunk is delivered along one tree. A node resides at all the trees to get the data objects. Multiple tree-based approaches enable fast content delivery and improve the robustness of data distribution. However, vulnerability to node churn is still the most critical concern for this kind of approach.

Swarm-based approaches are built on top of mesh overlay networks. They are also called data-driven approaches. Data content is pulled by the participants, rather than distributed actively by the data sources. Data content is also partitioned into small chunks. Each node keeps two "windows" indicating what it has and which chunk it requires, respectively. Nodes periodically exchange their "windows" with their neighbors and get the missing chunks actively.

Although swarming content distribution requires more time than the tree-based approach, it is widely used and deployed in current applications (e.g., PPlive, BT, etc.) because of its scalability and ease of implementation, among other features. The order of the chunks that a node gets is random, that is, no specified schedule is imposed. This is reasonable for applications that do not care about the arrival order of the chunks (e.g., BitTorrent). However, other applications require control of orders (e.g., the live streaming applications). In these applications, a chunk must appear on a node before its playback deadline, or it is useless. Basically, the rule for a node is to get the chunks whose deadlines are approaching and the chunks which are rarely replicated among its neighbors.

Recently, P2P has become popular in the context of dynamic information dissemination. Nodes that are interesting in the same event are organized in a P2P overlay network. Data items emerge at nodes dynamically, which means the data sources are no longer fixed. Data items are replicated and forwarded among peers. The delivery algorithms in this context are required to distribute the items appearing in any node to all the others as soon as possible with relative small network resource consumption. Swarm-based approaches are not suitable in this scenario because they suffer from a delay-versus-overhead trade-off. That is, to minimize data delivery delay, the intervals of "window" information exchange should be small.

However, this would increase control overhead. On the other hand, increasing the intervals would prolong the delivery delay. Building one tree for each potential data source is also not a good idea due to the considerable maintenance overhead involved. Building one undirected tree structure for all the sources is optimal for controlling overhead. However, if the data source is a leaf node, it would take as many as $2h$ hops to transmit the data item to all members, where h is the height of tree. Another concern is how to maintain the tree structure under node churn.

8.5.3 Copyright Protection Issues and Solutions

P2P networks are cost-effective in delivering large files to massive numbers of users. Unfortunately, today's P2P networks are grossly abused by illegal distribution of music, games, video streams, and popular software. These abuses have not only resulted in heavy financial loss in the media and content industries, but also hindered the legal commercial use of P2P technology. The main sources of illegal file sharing are peers who ignore copyright laws and collude with pirates. To solve this peer collusion problem, a copyright-compliant system for legalized P2P content delivery is proposed [31].

The goal is to stop collusive piracy within the boundary of a P2P CDN. In particular, the scheme appeals to protecting large-scale perishable contents that diminish in value as time elapses. Traditional CDNs use a large number of surrogate content servers over many globally distributed WANs. The content distributors need to replicate or cache contents on many servers. The bandwidth demand and resources needed to maintain these CDNs are very expensive. A P2P content network significantly reduces the distribution cost, since many content servers are eliminated and open networks are used. P2P networks improve content availability, as any peer can serve as a content provider. P2P networks are inherently scalable, because more providers lead to faster content delivery.

Honest or legitimate *clients* are those that comply with the copyright law not to share contents freely. *Pirates* are peers attempting to download some content file without paying or authorization. The *colluders* are those paid clients who share the content with pirates. Pirates and colluders coexist with law-abiding clients. *Content poisoning* is implemented by deliberate falsifying the file requested by the pirate. The media industry backed by the *Recording Industry Association of America* (RIAA) and *Motion Picture Association of America* (MPAA) has applied unscreened brute force content poisoning to deter piracy in open P2P file-sharing networks. However, their prevention results are mixed and controversial.

The media industry fight against all P2P file-sharing services by brute force. Legitimate clients can still enjoy the flexibility and convenience provided by an open P2P network. The scheme stops pirates from downloading copyrighted files, even in the presence of colluding peers. A reputation scheme is used to detect these colluders. A copyright-protected P2P network should benefit both the media industry and Internet user communities [6]. Figure 8.33 suggests a copyright-protected P2P network. As originally developed by Lou and Hwang [31], the network is built over a large number of peers. Four types of peers coexist in the P2P network: clients (honest or legitimate peers), colluders (paid peers sharing content with others without authorization), distribution agents (trusted peers operated by content owners for file distribution), and pirates (unpaid clients downloading content files illegally).

To join the system, clients submit requests to a *transaction server* that handles purchasing and billing matters. A *private key generator (PKG)* is installed to generate private keys with *identity-based signatures (IBSes)* for securing communication among the peers. The PKG's role is similar to a certificate authority (CA) in PKI services. The difference lies in the fact that the CA generates the public/private key pairs, while PKG only generates the private key.

The transaction server and PKG are only used initially when peers are joining the P2P network. With IBS, the communication between peers does not require an explicit public key, because the identity of each party is used as the public key. In our system, file distribution and copyright protection are completely distributed. Based on past experience, the number of peers sharing or requesting the same file at any point of time is in the hundreds. Depending on the variation of the swarm size, only a handful of distribution agents are needed. For example, it is sufficient to use

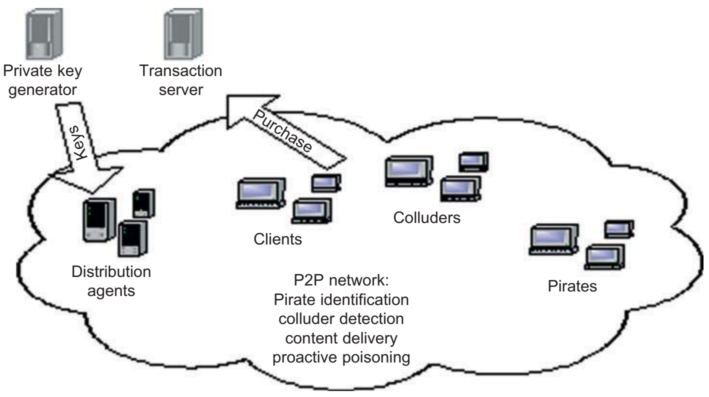


FIGURE 8.33

A secure P2P platform for copyright-protected content delivery.

(*Courtesy of Lou and Hwang [31]*)

Table 8.8 Mechanisms for Copyright Protection in P2P Networks

Mechanism	Protocol Requirements
Secure file indexing	File index format is modified to include token and IBS signature.
PAP	Peer sends digital receipt to bootstrap agent and obtains an authorization token. The token must be refreshed periodically.
Proactive content poisoning	The token and IBS signature check all download requests and responses, sending clean or poisoned contents, accordingly.
Random collusion prevention	Distribution agents randomly recruit decoys to probe for colluders. Collusion reports are weighted against client trust rates.

10 PC-based distribution agents to handle a swarm size of 2,000 peers. These agents authorize peers to download and prevent piracy by unpaid peers.

Paid clients, colluders, and pirates are all mixed up without visible labels. The copyright-protection network is designed to distinguish them automatically. Each client is assigned a *bootstrap agent*, selected from one of the distribution agents, as its entry point. In current P2P networks, a peer can self-assert its username without verification. Therefore, the peer endpoint address (IP address + port number) instead of the username is used to identify a peer. A peer is considered fully connected if it is reachable via a listening port on its host.

Table 8.8 summarizes the key protocols and mechanisms used to construct a trusted P2P system. In this approach, the modified file index format enables pirate detection. Peer Authorization Protocol (PAP) authorizes legitimate download privileges to clients. The content distributor applies content poisoning to disrupt illegal file distribution to unpaid clients. The system is enhanced by randomized collusion detection. In the system, a content file must be downloaded fully to be useful. Such a restraint is easily achievable by compressing and encrypting the file with a trivial password that is known to every peer. This encryption does not offer any protection of the content, except to package the entire file for distribution.

8.5.4 Collusive Piracy Prevention in P2P Networks

Figure 8.34 illustrates the concept of proactive content poisoning. If a pirate sends a download request to a distribution agent or a client, based on the protocol definition it will receive poisoned file chunks. If the download request was sent to a colluder, it will receive clean file chunks. If a pirate shares the file chunks with another pirate, it could spread the poison. Therefore, it is critical to send poisoned chunks to pirates to deny their requests. Otherwise, the pirate still can assemble a clean copy from those colluders who have responded with clean chunks. With poisoning, we exploit the limited poison detection capability of P2P networks and force a pirate to discard the clean chunks downloaded with the poisoned chunks. The rationale behind such poisoning is that if a pirate keeps downloading a corrupted file, the pirates will eventually give up their attempts out of frustration.

8.5.4.1 Randomized Colluder Detection

As reported in the GossipTrust paper [58], the gossip protocol and power nodes play a crucial role in speeding up the reputation aggregation process in a P2P network. Randomized gossiping can reach consensus among all peers in a distributed manner. This approach exploits massive concurrency among millions of active nodes in a very large P2P network. Figure 8.34 shows a simplified GossipTrust system to identify colluders.

The idea is to associate each {*peer, file*} pair with a *collusion rate*. The "0" rate means the peer was never reported as a colluder. Otherwise, the peer is getting a collusion report of "1," meaning it has shared clean content with illegal download requesters. This collusion rate is accumulative in the same way eBay collects peers' reputation scores. Figure 8.35 illustrates the collusion detection process. Distribution agents randomly recruit clients, called *decoys*, to send illegal download requests to

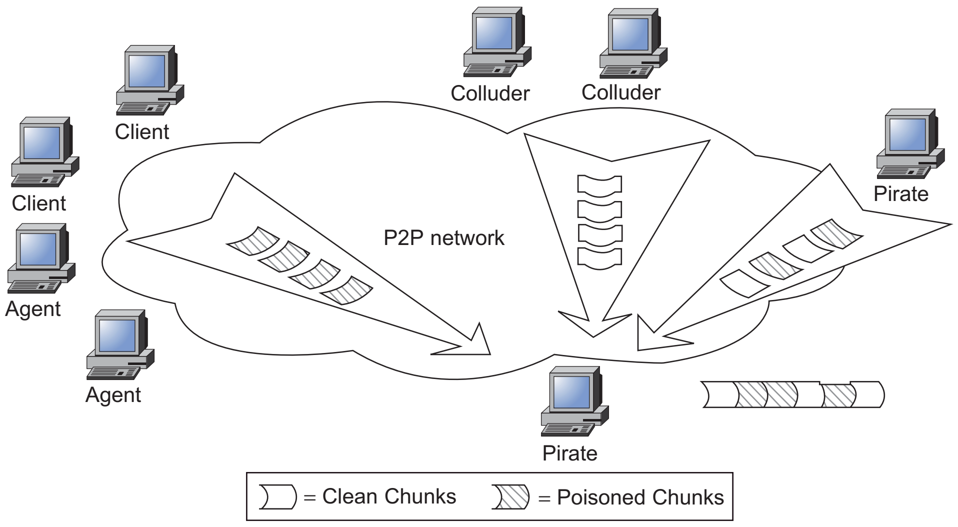


FIGURE 8.34

Proactive poisoning in a trusted P2P network, where clean chunks (white) are received by legitimate clients and poisoned chunks (shaded) by detected pirates.

(*Courtesy of Lou and Hwang [31]*)

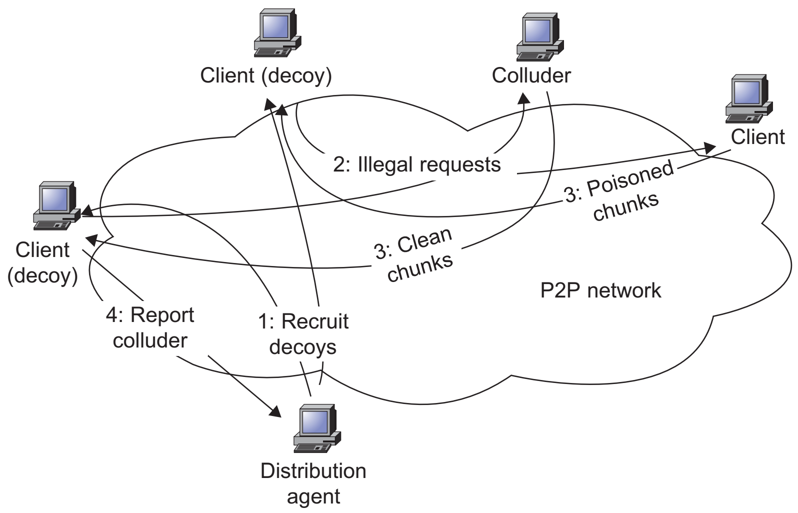


FIGURE 8.35

Distribution agent randomly recruiting some clients to probe suspected peers; peer collusion is reported when a peer supplies a copyrighted file to an illegal requestor.

(*Courtesy of Lou and Hwang [31]*)

suspected peers. If an illegal request is returned with a clean file chunk, the decoy reports the collusion event. Since the decoy is randomly chosen, there exists a risk that the report is not trustworthy either by error or by cheating. Thus, a reputation system is needed to screen the peers.

A new PAP was developed to distinguish pirates from legitimate clients. Detected pirates will receive poisoned chunks in their repeated attempts. Pirates are thus severely penalized with no chance to download successfully in a tolerable amount of time. Based on simulation results, a 99.9 percent prevention rate was found in Gnutella, KaZaA, and Freenet, while for eMule, eDonkey, Morpheus, and others the rate is between 85 percent and 98 percent. This prevention system is shown to be less effective in protecting poison-resilient networks such as BitTorrent and Azureus. However, the poisoning approach opens up low-cost P2P technology for copyrighted content delivery. The advantage lies mainly in minimum delivery speed, higher content availability, and copyright compliance (see Figure 8.36).

8.5.4.2 Pirate Download Time and Success Rate

To explore the limit of the trusted P2P system, various file sizes have been experimented. The *protection success rate* β is defined by the failure rate of pirates to download a file within a *tolerance threshold* θ of time. The θ is set at 20 days for a 700 MB CD-ROM image file and 30 days for a 4.5 GB movie file.

Figure 8.36 simulates all three P2P network families with 100 paid clients, 900 colluders, and 1,000 pirates. This scenario of 90 percent colluders is very unlikely in a real-life P2P network. This experiment is designed to approximate a worst-case scenario that all pirates adopt an aggressive peer selection policy. The percentage of pirates that fail to download a clean copy within the time frame θ as an approximated success rate β is measured. All β curves start with 100 percent initially.

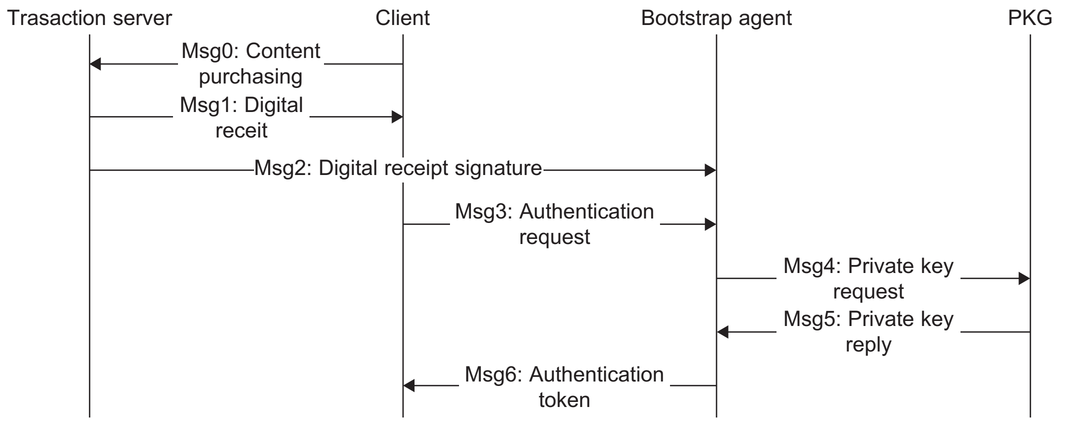

Trasaction server
Client
Bootstrap agent
PKG
Msg0: Content purchasing
Msg1: Digital receit
Msg2: Digital receipt signature
Msg3: Authentication request
Msg4: Private key request
Msg5: Private key reply
Msg6: Authentication token

FIGURE 8.36

The protected peer joining process for copyrighted P2P content delivery using seven messages to secure communications among four parties involved.

(*Courtesy of Lou and Hwang [31]*)

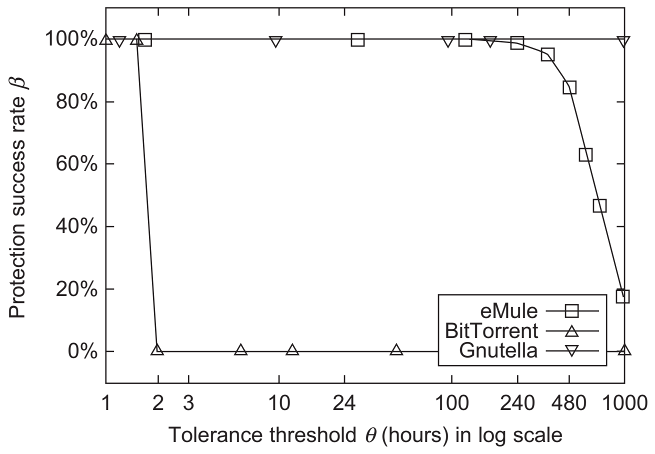


FIGURE 8.37

Protection success rate of a simulated eMule network by prolonging the delivery delay to an intolerable magnitude.

(*Courtesy of Lou and Hwang [31]*)

Without poisoning, the average download time for a client is 1.5 hours. The Gnutella family, including KaZaA and LimeWire, has a near-perfect success rate (higher than 99.9 percent). The eMule network has an average of 85 percent after tolerating up to 20 days. These success rates are satisfactory, because most pirates will give up trying after a few days without success. The Gnutella family including the KaZaA and LimeWire network has the highest success rate, close to 99.9 percent, for both file sizes.

With a tolerance threshold of 20 days in Figure 8.37, the eMule network has an average 98 percent success rate for the 4.5 GB file and 85 percent success rate for the 700 MB file. In comparison, an average pirate in a BitTorrent network takes about 100 minutes to download the 700 MB file, only marginally longer than a paid client. Hence the success rate drops rapidly before two hours. This implies that our system does not protect BitTorrent well due to its strong resistance to content poisoning.

8.6 BIBLIOGRAPHIC NOTES AND HOMEWORK PROBLEMS

A good tutorial on P2P systems was given by Ross and Rubestein [38] at IEEE Infocom 2004. The key concepts of P2P are introduced in Wikipedia [50]. Buford, et al. have treated P2P networking and applications in a 2008 book [6]. The problem of building overlays for unstructured P2P networks has been studied in [13,28,52]. Enabling techniques for P2P computing are discussed in [5]. NetTube [8] leverages the high clustering property of videos and P2P techniques for improving YouTube services. Keromytis, et al. [21] reported the SOS architecture and operational requirements. The Skype protocol was analyzed in [3] and measured in [15].

Reported work on consistent hashing can be found in [21]. Chord was first proposed by Stoica, et al. [45] and extended by a large number of researchers [12,16]. CAN was proposed by Ratnasamy, et al. [40]. Pastry was treated by Rowstron and Drusche [39]. Kademlia was proposed in [35] and measured in [44]. Gnutella was studied in [60] and measured in [17], and Freenet in [10]. Hybrid overlays are studied in [32]. Gnutella and Napster were also measured and compared by Sarioiu, et al. in [41]. Network proximity for structured P2P networks was studied in [7,16,54], while for unstructured P2P networks, studies can be found in [28,52]. The small-world networks is studied in Kleinberg [22].

P2P churn resilience and fault tolerance are treated in [12,14,23,24,51]. Trust models for P2P computing have been studied in [19,27,43,49]. Three P2P reputation systems—EigenTrust, PeerTrust, and PowerTrust—are reported in [20,53,57], respectively. P2P file-sharing and content distribution technologies can be found in [1,25,37,31,37,41,46]. Data replication and network proximity are studied in [11,33]. Overlay multicasts were extensively discussed in [9,47,48,56]. P2P video streaming is studied in [8,18,29,36]. BitTorrent has been studied in [4,26,30,59].

Ross and Rubestien [38] have suggested four important research areas for P2P systems: (1) Locality in DHT-based systems, (2) Using DH to hash or not to hash, (3) Building a trusted systems from autonomous and untrusted collections, and (4). Building P2P systems to operate in high-churn environments. These challenges are partially solved in some recent work on churn-resilient load balancing [42], massive data dissemination [24], P2P trust management [43,57,58], and copyright protection [31]. A reputation system for social networking is reported in [55].

Acknowledgements

This chapter was jointly written by Zhenyu Li of the Chinese Academy of Sciences (CAS) and by Kai Hwang of the University of Southern California. The authors would like to acknowledge the technical support from Gaogang Xie of CAS Institute of Computing Technology.

References

[1] S. Androutsellis, D. Spinellis, A survey of P2P content distribution technologies. ACM. Comput. Surv. (December) (2004).

[2] L. Amaral, A. Scala, M. Barthelemy, M. Stanley, Classes of small-world networks. Natl. Acad. Sci. 97 (21) (2000).

[3] S.A. Baset, H. Schulzrinne, An analysis of the Skype peer-to-peer Internet telephony protocol, in: Proceedings of IEEE INFOCOM, April 2006.

[4] R. Bindal, P. Cao, W. Chan, et al., Improving traffic locality in BitTorrent via biased neighbor selection, in: Proceedings of IEEE ICDCS, 2006.

[5] B. Bloom, Space/time tradeoffs in hash coding with allowable errors. Commun. ACM. 13 (7) (1970) 422–426.

[6] J. Buford, H. Yu, E. Lua, P2P Networking and Applications, Morgan Kaufmann, December 2008. Also www.p2pna.com.

[7] M. Castro, P. Druschel, Y.C. Hu, A. Rowstron, Exploiting network proximity in distributed hash tables, in: Proceedings of the International Workshop on Future Directions in Distributed Computing, June 2002.

[8] X. Cheng, J. Liu, NetTube: exploring social networks for peer-to-peer short video sharing, in: Proceedings of IEEE Infocom, March 2009.

[9] S. Chen, B. Shi, S. Chen, ACOM: any-source capacity-constrained overlay multicast in non-DHT P2P networks, in: IEEE Transactions on Parallel and Distributed Systems, September 2007, pp. 1188–1201.

[10] I. Clarke, O. Sandberg, B. Wiley, T.W. Hong, Freenet: a distributed anonymous information storage and retrieval system, in: ICSI Workshop on Design Issues in Anonymity and Unobservability, June 2000.

[11] E. Cohen, S. Shenker, Replication strategies in unstructured peer-to-peer networks, in: ACM SIGCOMM, 2002.

[12] A. Fiat, J. Saia, M. Young, Making chord robust to Byzantine attacks, in: Proceedings of the European Symposium on Algorithms (ESA), 2005.

[13] A.J. Ganesh, A.M. Kermarrec, L. Massoulié, Peer-to-peer membership management for gossip-based protocols. IEEE. Trans. Computers. 52 (2) (2003) 139–149.

[14] P.B. Godfrey, S. Shenker, I. Stoica, Minimizing churn in distributed systems. in: Proceedings of ACM SIGCOMM, 2006.

[15] S. Guha, N. Daswani, R. Jain, An experimental study of the Skype peer-to-peer VoIP system, in: Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS), February 2006.

[16] K.P. Gummadi, R. Gummadi, S.D. Gribble, et al., The impact of DHT routing geometry on resilience and proximity, in: Proceedings of ACM SIGCOMM, 2003.

[17] M. Hefeeda, O. Saleh, Traffic modeling and proportional partial caching for peer-to-peer systems, in: IEEE/ACM Transactions on Networking, December 2008, pp. 1447–1460.

[18] Y. Huang, et al., Challenges, design and analysis of a large-scale P2P VOD system, in: Proceedings of ACM SIGCOMM 2008, Seattle, August 2008.

[19] K. Hwang, D. Li, Trusted cloud computing with secure resources and data coloring, IEEE. Inte. Comput. (September) (2010) 14–22.

[20] S. Kamvar, M. Schlosser, H. Garcia-Molina, The Eigentrust algorithm for reputation management in P2P networks, ACM WWW '03, Budapest, Hungary, 2003.

[21] A. Keromytis, V. Misra, D. Rubenstein, SOS: secure overlay services, in: Proceedings of ANM SIGCOMM'02, Pittsburg, PA. August 2002.

[22] J. Kleinberg, The small-world phenomenon: an algorithmic perspective, in: Proceedings 32nd ACM Symposium on Theory of Computing, Portland, OR. May 2000.

[23] J. Leitao, J. Pereira, L. Rodrigues, Epidemic broadcast trees, in: Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems, October 2007.

[24] Z. Li, G. Xie, K. Hwang, Z. Li, Churn-resilient protocol for massive data dissemination in P2P networks, in: IEEE Transactions on Parallel and Distributed Systems, accepted to appear 2011.
[25] Z. Li, G. Xie, Z. Li, Efficient and scalable consistency maintenance for heterogeneous peer-to-peer systems, in: IEEE Transactions on Parallel and Distributed Systems, December 2008, pp. 1695–1708.
[26] Z. Li, G. Xie, Enhancing content distribution performance of locality-aware BitTorrent systems, in: Proceedings of IEEE Globecom, December 2010.
[27] L. Liu, W. Shi, Trust and reputation management, in: IEEE Internet Computing, September 2010, pp. 10–13. (special issue).
[28] Y. Liu, L. Xiao, X. Liu, L. M Ni, X. Zhang, Location awareness in unstructured peer-to-peer systems, in: IEEE Transactions on Parallel and Distributed Systems, February 2005, pp. 163–174.
[29] J. Liu, S.G. Rao, B. Li, H. Zhang, Opportunities and challenges of peer-to-peer internet video broadcast, in: Proceedings of the IEEE Special Issue on Recent Advances in Distributed Multimedia Communications, Vol. 96 (1), 2008, pp. 11–24.
[30] T. Locher, P. Moor, S. Schmid, R. Watenhofer, Free riding in BitTorrent is cheap, in: Proceedings of ACM HotNets, November 2006.
[31] X. Lou, K. Hwang, Collusive piracy prevention in P2P content delivery networks, in: IEEE Transactions on Computers, Vol. 58, (7) 2009, pp. 970–983.
[32] B.T. Loo, R. Huebsch, I. Stoica, J. M. Hellerstein, The Case for a Hybrid P2P Search Infrastructure, 3rd International Workshop on Peer-to-Peer Systems, February 2004.
[33] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker, Search and replication in unstructured peer-to-peer networks, in: Proceedings of the ACM International Conference on Supercomputing, June 2002.
[34] B. Maniymaran, M. Bertier, A.-M. Kermarrec, Build one, get one free: leveraging the coexistence of multiple P2P overlay networks, in: Proceedings of the International Conference on Distributed Computing Systems, Toronto, June 2007.
[35] P. Maymounkov, D. Mazières, Kademlia: a peer-to-peer information system based on the XOR metric, in: Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS), March 2002.
[36] N. Magharei, R. Rejaie, Y. Guo, Mesh or multiple-tree: a comparative study of live P2P streaming approaches, in: Proceedings of IEEE INFOCOM, Alaska, May 2007.
[37] G. Pallis, A. Vakali, Insight and perspectives for content delivery networks, Commun. ACM 49 (1) (2006) 101–106.
[38] K. Ross, D. Rubenstein, Peer-to-peer systems, in: IEEE Infocom, Hong Kong, 2004, (Tutorial slides).
[39] A. Rowstron, P. Druschel, 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), Heidelberg, Germany, November 2001, pp. 329–350.
[40] S. Ratnasamy, P. Francis, M. Handley, R. Karp, A scalable content-addressable network, in: Proceedings of ACM SIGCOMM, August 2001.
[41] S. Saroiu, P. Gummadi, S. Gribble, A measurement study of peer-to-peer file sharing systems, in: Multimedia Computing and Networking (MMCN '02), January 2002.
[42] H. Shen, C. Xu, Locality-aware and churn-resilient load balancing algorithms in structured peer-to-peer networks, in: IEEE Transactions on Parallel and Distributed Systems, Vol. 18 (6), June 2007, pp. 849–862.
[43] S. Song, K. Hwang, R Zhou, Y.K. Kwok, Trusted P2P transactions with fuzzy reputation aggregation, in: IEEE Internet Computing, November–December 2005, pp. 18–28.
[44] M. Steiner, T. En-Najjary, E. Biersack, Long term study of peer behavior in the KAD DHT, in: IEEE/ACM Transactions on Networking, Vol. 17 (6) 2009.
[45] I. Stoica, R. Morris, D. Liben-Nowell, et al., Chord: a scalable peer-to-peer lookup service for Internet applications, in: IEEE/ACM Transactions on Networking, Vol. 11 (1) February 2003, pp. 17–32.
[46] K. Spripanidkulchai, B. Maggs, H. Zhang, Efficient content location using interest-based locality in peer-to-peer systems, in: Proceedings of IEEE INFOCOM, 2003.

[47] C. Tang, R.N. Chang, C. Ward, GoCast: gossip-enhanced overlay multicast for fast and dependable group communication, in: Proceedings of the International Conference on Dependable Systems and Networks, Yokohama, Japan, June 2005, pp. 140–149.
[48] V. Venkataraman, K. Yoshida, P. Francis, Chunkyspread: heterogeneous unstructured tree-based peer to peer multicast, in: 14th IEEE International Conference on Network Protocols, November 2006, pp. 2–11.
[49] Y. Wang, J. Vassileva, Trust and reputation model in peer-to-peer networks, in: Third International Conference on Peer-to-Peer Computing, August 2003.
[50] Wikipedia, Peer to Peer. http://en.wikipedia.org/wiki/Peer-to-peer, 2010 (accessed 14.08.2010).
[51] R.H. Wouhaybi, A.T. Campbell, Phoenix: supporting resilient low-diameter peer-to-peer topologies, in: IEEE INFOCOM, 2004.
[52] L. Xiao, Y. Liu, L.M. Ni, Improving unstructured peer-to-peer systems by adaptive connection establishment, in: IEEE Transactions on Computers, Vol. 54 (9), September 2005, pp. 1091–1103.
[53] L. Xiong, L. Liu, Peertrust: supporting reputation-based trust for peer-to-peer electronic communities, in: IEEE Transactions on Knowledge and Data Engineering, Vol. 16 (7), 2004, pp. 843–857.
[54] Z. Xu, C. Tang, Z. Zhang, Building topology-aware overlays using global soft-state, in: Proceedings on the International Conference on Distributed Computing Systems, 2003.
[55] M. Yang, Y. Dai, X. Li, Bring reputation system to social network in the maze P2P file-sharing system, in: IEEE 2006 International Symposium on Collaborative Technologies and Systems (CTS 2006), Las Vegas, 14–17 May 2006.
[56] Z. Zhang, S. Chen, Y. Ling, R. Chow, Capacity-aware multicast algorithms in heterogeneous overlay networks, in: IEEE Transactions on Parallel and Distributed Systems, February 2006, pp. 135–147.
[57] R. Zhou, K. Hwang, PowerTrust for fast reputation aggregation in peer-to-peer networks, in: IEEE Transactions on Parallel and Distributed Systems, April 2007, pp. 460–473.
[58] R. Zhou, K. Hwang, M. Cai, GossipTrust for fast reputation aggregation in peer-to-peer networks, in: IEEE Transactions on Knowledge and Data Engineering, September 2008, pp. 1282–1295.
[59] Z. Zhou, Z. Li, G. Xie, ACNS: Adaptive complementary neighbor selection in BitTorrent-like applications, in: Proceedings of IEEE ICC 2009, Germany, 2009.
[60] Y. Zhu, Y. Hu, Enhancing search performance on Gnutella-Like P2P systems, in: IEEE Transactions on Parallel and Distributed Systems, Vol. 17 (12), December 2006, pp. 1482–1495.

HOMEWORK PROBLEMS

Problem 8.1

You studied the basic idea of DHT in Section 8.3.1. One important DHT-based protocol is Kademlia [35]. The unique feature of Kademlia is that it has been successfully used in eDonkey/eMule file-sharing systems. Study the Kademlia paper and compare it with Chord and CAN in terms of routing complexity, maintenance cost, and scalability. Finally, download the source code of eMule at www.emule-project.net and study how it works. If possible, modify the client protocol to measure the eMule system. For measurement, please refer to the paper authored by M. Steiner [44].

Problem 8.2

P2P VoD (video on demand) systems are different from P2P live streaming systems since P2P-VoD has less synchrony in the users sharing video content, and thus it is very difficult to leverage end users' resources. Please read the paper [18] authored by Huang, et al. Then list the challenges of designing P2P VoD systems and explain how PPlive solves these challenges.

Problem 8.3

List the three types of approaches to exploit network proximity in structured P2P overlays and their advantages and disadvantages. Section 8.3.3 points out that, in Chord overlay networks, a node x can take the physically closest node whose identifier falls in the range $[x+2^i, x+2^{i+1}]$ as its i-th figure entry, which does not affect routing complexity. Why does this hold?

Problem 8.4

Building network proximity-aware overlays would degrade the performance of search algorithms such as flood-based algorithms and random walk-based algorithms since nodes are clustered into groups and fewer links are built among the groups. For example, a random walk query message may be forwarded within a cluster for many hops before reaching the cluster where the resource is located. You are asked to design a search algorithm for clustered overlay networks.

Problem 8.5

Data delivery can also been implemented on top of a DHT-based structure. The representative approach is CAM-Chord [56], which builds a capacity-aware Chord overlay and delivers data items on top of it. CAM-Chord brings no redundant message, that is, to deliver an item to n nodes, only n message is required. However, maintaining such a structure is nontrivial. Read the paper and compare CAM-Chord with the ACOM [9] and CRP approaches [24]. Please focus on following performance metrics: hop complexity, proximity awareness, overlay maintenance overhead, and data replication ratio.

Problem 8.6

In Table 8.9, compare the routing algorithms of Chord, Pastry, and CAN by briefly filling in the missing table entries. Let n be the network size (number of peer nodes) and d be the dimension size. A few of the entries are already given as examples.

Problem 8.7

P2P traffic locality can also been improved by deploying cache at the gateway of the network provider. However, different from the HTTP access pattern, the popularity distribution of objects (files)

Table 8.9 Comparison of Structured P2P Networks

Features	Chord	Pastry	CAN	Kademlia
Route state			$O(d)$	
Network structure	Chordal ring			
Routing complexity		$O(\log n)$		
Fault tolerance				

does not follow Zipf law. Read the paper [17] and explain how the popularity distribution affects the cache algorithm. Furthermore, discuss the advantages and shortcomings of different methods to preserve locality.

Problem 8.8

In Section 8.5.1, you studied the basic search algorithms that do not consider semantic information of shared content. It has been identified in [60] that building semantic overlays can significantly improve search performance. Read this paper and their related works. You are asked to describe the reasons why semantic overlays can enhance search performance.

Problem 8.9

YouTube is one of the largest web sites for user-generated videos. Tens of thousands of videos are viewed each day by a large number of users. YouTube naturally leverages P2P techniques to alleviate server loading. However, unlike traditional videos, user-generated videos are small. Study the NetTube paper [8] and discuss the solution for using P2P techniques in YouTube content distribution.

Problem 8.10

Analyze the fault tolerance of a Chord overlay with respect to the size of the successor list. Is it sufficient to guarantee a stable Chord ring with high probability in the case that the successor list size is $O(\log n)$ (n is the number of nodes), when every peer fails with probability ½?

Problem 8.11

BitTorrent is wildly used for file sharing. Files are divided into pieces. Nodes exchange the piece availability information with their neighbors. When a node finds there is a piece that its neighbor has but it has not downloaded yet, it sends a request for the piece to that neighbor. By default, the piece size is 256 KB. Perform simulations using the code in http://theory.stanford.edu/~cao/biased-bt.html to evaluate how piece size affects content distribution performance for files of various sizes.

Problem 8.12

Skype is a P2P-based VoIP system. It has been widely used and studied. Read the papers on Skype [3]. Then describe the overlay structure of Skype, the bandwidth consumption characteristic, and the impact of access capacity constraint on VoIP quality.

Problem 8.13

In mesh-based pull methods, each node advertises to its neighbors which data block it has received. The neighbors explicitly request messages if needed. Tree-based push methods, on the other hand, deliver data blocks along one or multiple tree structures. It is considered that mesh-based methods suffer from a trade-off between control overhead and delay, while tree-based methods have a trade-off between continuity and delay [48]. Read a survey paper on content delivery algorithms [36] and a comparison study on mesh- and tree-based methods [29].You may consider combining these two methods by leveraging network coding in tree-based systems.

Problem 8.14

BitTorrent systems use the so-called "Tit-for-Tat" incentive scheme to enable contribution awareness in content delivery. Some researchers point out that this mechanism is sufficient to ensure fairness, but others are doubtful of this and free-riding is still cheap. Study these three papers in [20,29,37]. Then answer the following two questions: (1) Which factor makes free riding so easy? (2) Do you think "Tit-for-Tat" is sufficient to solve the problem? Justify your answers with reasoning.

Problem 8.15

Match the overlay networks on the left with their best-match descriptions on the right:

_____eMule	**a**) A structured overlay using a coordinated multi-dimensional search space
_____BitTorrent	**b**) An unstructured overlay for anonymous and decentralized storage services
_____Gnutella	**c**) A P2P network for video telephony applications
_____Napster	**d**) An overlay coded with high-radix nested groups for proximity routing
_____SETI@Home	**e**) A P2P file-sharing network using multiple file index trackers
_____KaZaA	**f**) The first P2P network for MP3 music delivery using a centralized control
_____Chord	**g**) A secure overlay service for preventing DDoS or network attacks
_____Pastry	**h**) An unstructured overlay using flooding search for file providers
_____Freenet	**i**) A decentralized P2P network built with multiple super nodes
_____CAN	**j**) A P2P file-sharing network applying hashing at a 9.5 MB part level
_____Skype	**k**) A DHT-based overlay structure with a O(log n) lookup time
_____SOS	**l**) A P2P network for distributed processing of outer space signals

Problem 8.16

Consider a Chord overlay network with 64 keys identified by 6-bit key identifiers. Assume six nodes (0, 4, 7, 12, 32, 50) are populated at present. The rest are not assigned yet. Answer the following questions using this Chord graph:

a. What are the data keys and node keys stored in node 0?

b. Work out the finger table for node 7. Describe how node 7 uses the finger table to find the shortest routing path to reach the home node of a document file where data key 3 is located.

c. Describe how to insert a new node 45 into this Chord structure. You need to show the shortest path on the Chord graph leading to the node holding key 45.

d. Use the finger table at the predecessor node to perform five lookup operations to establish the finger table for this new node 45.