

## CHAPTER

## 1

## Introduction to Peer-to-Peer

The early developers of ARPANET and the Internet allowed themselves to dream about a day when computers around the world would be linked together to share resources in a peer-to-peer fashion. In 1962, one of those early dreamers, J.C.R. Licklider of MIT, wrote a now-famous series of memos in which he described an “Intergalactic Network” of interconnected computers. His vision led to the creation of the Network Control Program (NCP), the first “host-host” networking protocol and the precursor to TCP/IP.

The host-host concept—which we now call peer-to-peer—was crucial in developing the Internet. Every computer on the network was an equal: Each computer could access the resources of any other computer on the network while making its own resources available. Communication among hosts was also equal: No computer was seen as a client or component of another, and all computers shared more-or-less equal bandwidth access to one another.

Several events have conspired to change the Internet landscape from primarily peer-to-peer to the now more familiar client-server architecture. The Internet has gradually become more commercial, and corporations build firewalls around their information to control access. Millions of people “log on” to the Internet using desktop computers that cannot match the power of the servers that form the backbone of the Internet. And many popular Internet applications and services, including the World Wide Web and FTP, are based on a client-server architecture.

In the last few years, however, peer-to-peer (P2P) technology has once again had a profound effect on the Internet and the distribution of information and resources. P2P is being aggressively hyped in the media, but there are a wide variety of opinions as to what exactly P2P is:

- Clay Shirky (Internet consultant, writer, and speaker) once said that “P2P is a class of applications that takes advantage of resources—storage, cycles, content, human presence—available at the edges of the Internet.”
- Li Gong of Sun Microsystems wrote that “The term peer-to-peer networking is applied to a wide range of technologies that greatly increase the utilization of information, bandwidth, and computing resources in the Internet. Frequently, these P2P technologies adopt a network-based computing style that neither excludes nor inherently depends on centralized control points.”
- Ed Dumbill of XML.com said, “P2P is whoever says they’re P2P.”

Right now Mr. Dumbill’s definition seems to be winning the popular vote. One of the purposes of this chapter is to help you formulate your own answer to the question, “What is P2P?” The next section begins our discussion of peer-to-peer technology with a quick review of network topologies. In the final section, we look at Napster, Morpheus, instant messaging, and Usenet from a P2P perspective; and then discuss some of the legal, technical, and security issues that have arisen from the use of these and similar applications.

## What Is a Peer-to-Peer Architecture?

If we take a moment to consider the Internet itself, we will see that there are millions of computers connected in the network at any given time. All of the computers are theoretically connected to one another, and information stored on any of the systems can be accessed. As a whole, the topology or layout of the computers on the Internet is a grouping of machines spread out in various locations. Within each of the groups or subnets, computers will be visible to other computers on the subnet and sometimes to the outside Internet.

Some of the computers will be servers and host information. The machines at Yahoo! that serve up contents are *web servers*. Browsing to Yahoo! on your local computer turns the machine into a *client*. This type of client-server interaction is happening for hundreds of thousands of computers at the same time. While a client machine is browsing to Yahoo!, it could also be sharing a local drive with group members. In this situation, the machine will become a server to any client that tries to access files on the local drive.

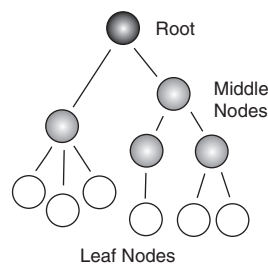
In most peer-to-peer systems, the division between a server and a client is blurred. The computer itself might be connected to other computers using a token-ring topology, but a peer-to-peer system might have a completely different architecture. The peers could all be communicating with a central server, like Napster.

In most cases, peers will be connected to one another over the Internet using either the TCP or HTTP protocol. As you probably already know, TCP/IP is the fundamental protocol for transferring information over the Internet. The HTTP protocol is built on top of TCP/IP and allows communication between computers using port 80. HTTP is very popular for peer-to-peer systems because most organizations keep port 80 clear on their firewalls for web browser traffic.

Several network topologies can be used for connecting P2P systems. In this section, we discuss the major P2P network topologies in order to explain how information can be transmitted between peers effectively.

## The Hierarchical Topology

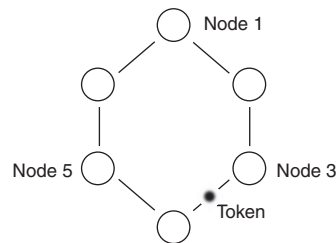
One of the most common topologies is the *hierarchy*. Every time you type a website URL into your browser, you are using a system called DNS, or Domain Name Server. This system is set up in a hierarchy, with root servers at the very top levels. The hierarchy topology looks like Figure 1.1. For several years now, critics have called for an overhaul of the DNS architecture because the root servers represent a single point of failure. However, because the entire system is based on replication and the chance of the DNS system going down is very small, no real work has occurred in this area.



**Figure 1.1** The hierarchy network topology.

## The Ring Topology

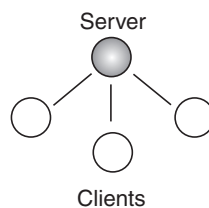
*Token Ring* is a network topology that uses the concept of passing a single token around to the computers connected in a ring pattern. When a machine receives the token, it can send information out onto the network. The ring topology isn't used much anymore for common networks, but does provide an interesting pattern for load-balancing a single-server system or hierarchy. The top rung of a hierarchy topology could actually be a ring of servers that balance the network requests. Figure 1.2 shows what a ring topology looks like.



**Figure 1.2** The ring network topology.

## The Client-Server, or Centralized, Topology

By far the most common topology is the *client-server*, or *centralized*, topology. The terminology of client-server has been with us for many years; more recently, the term *centralized* has been used to describe a system in which a single computer, the server, makes services available over the network. Client machines contact the server when the services are needed. Obviously, the more clients in the system, the larger the server must be. At some point, the server will need to be replicated in order to handle the traffic volume from all clients. Figure 1.3 shows an example of the centralized topology.

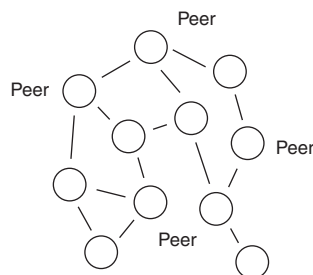


**Figure 1.3** The client-server, or centralized, network topology.

## The Decentralized Topology

The *decentralized* topology is a network topology that comes closest to being truly peer-to-peer. There is no central authority, only individual computers that

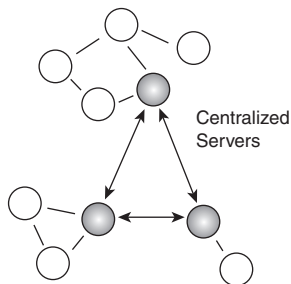
are able to connect and communicate with any of the other computers on the network. When a packet of information starts its travels on the Internet, it is basically traveling through a decentralized topology. Information within the packet itself tells each computer where to send the packet next. Figure 1.4 shows an example of a decentralized network topology. Basically, all of the peers in the system act as both clients and servers, handling query requests and downloads while also making media searches and download request themselves. The KaZaA and Gnutella applications use this decentralized topology for their P2P systems.



**Figure 1.4** The decentralized network topology.

## The Hybrid Topology

In the *hybrid* topology shown in Figure 1.5, we have an example of a situation where the individual computers are considered clients when they need information. The client that needs information will contact a central server (the centralized servers are distributed in the example shown in Figure 1.5) to obtain the “name” of another client where the requested information is stored. The requesting client will then contact the client with the information directly. With the hybrid, a computer can be either a client or a server. This is the topology used for the Napster system—individual peers contact a localized server for searching and proceed to contact peers directly for media downloading.



**Figure 1.5** The hybrid network topology.

## Examples of Peer-to-Peer Systems

This section describes several well-known peer-to-peer applications. We briefly examine how each one shares resources and services so that by the end of this section you'll have a clearer idea of what functional, real-world P2P really means.

### Napster

Napster is the software that thrust the peer-to-peer concept into the limelight. As you probably know, Napster was developed to allow the sharing of MP3 music files created from CDs and other sources. Later in the chapter, we briefly discuss why Napster isn't the peer-to-peer powerhouse it once was, but for now let's look at how Napster works:

1. A prospective user downloads the Napster peer software from a Napster primary or mirror web site.
2. Once installed and launched, the peer software attempts to connect to a central Napster server, where the user is required to choose a username and password.
3. The user can have the peer software search his or her local hard drive for MP3 files to share with others. If this option is selected, the user's hard drive will be searched and the names of any media files will be transferred to the central server. Note that only the *filenames* are transferred.
4. The user can search for media in the Napster network. The peer software will transfer the search string to the central server, which will return a list of files found and connection information about the peer computers where the files reside, including the username, the IP address, the port to connect to, the connection speed, and the file size.
5. The Napster peer making the request will attempt to directly contact the Napster peer on the remote computer where the target file resides. At this point, the central server is no longer involved in the file transfer.

For Napster, the central server is just a large database containing a list of all files found in all clients in the network. The system worked very well, and many of the peers who had a fast connection to the Internet were typically slammed with file requests. If your system was being swamped with requests to download files, you could set a limit to allow only  $N$  number of active downloads at any one time. If a new download request came into the peer, the peer would respond with a message indicating that the download request was added to the queue. The queue would automatically keep track of all download requests and move them into an active state as older download requests finished.

From this explanation, it is clear that the Napster system uses a hybrid topology. Without the centralized server, all of the peers in the system would have media to share and also want to search for media, but they wouldn't know about one another. The centralized server is responsible for keeping a database of all peers and what they have to share available to all peers in the network.

## Gnutella

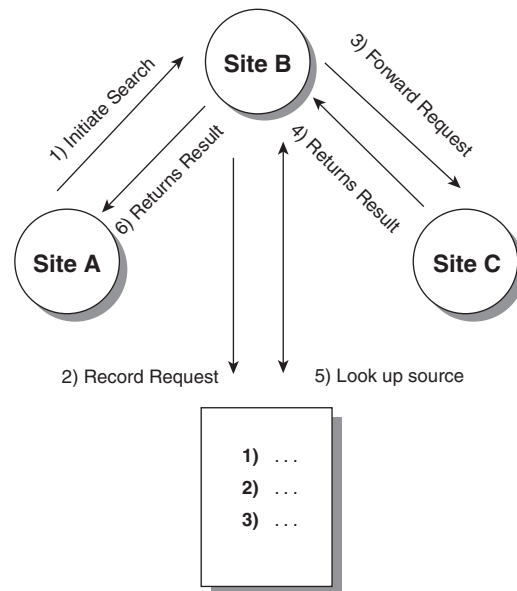
As Napster became successful, other P2P products such as Gnutella were created to enable information sharing. One feature that distinguishes Gnutella is that it uses the HTTP protocol to transfer information. HTTP is used by web browsers to contact web servers, so in a sense a Gnutella peer is actually a transparent web site “server” with links for each of the pieces of media being shared. A Gnutella peer contacts a Gnutella “server” in much the same way that a standard web browser contacts a web server.

The topology for Gnutella is decentralized, and there is no centralized authority. So, if there is no central authority, how does a peer obtain a list of the media files available in the network?

The real key to Gnutella is its search capability. With no central server, the peers need to be able to determine what files are available in a fashion that is both quick and effective. The search mechanism works by creating a search packet with a *max hops* value that indicates the maximum number of times the search packet will be propagated in the Gnutella network before it is returned to the peer that originated it. So, if a packet has a max hops value of 3, the packet will be allowed to be propagated throughout the Gnutella network a maximum of three peers away from its peer of origin. As each peer receives the packet, it decrements an internal counter. When the internal counter reaches zero, the search packet is no longer forwarded to other peers.

When a peer requests a search, the search packet is sent to all the peers that the requesting peer knows about. Those peers will immediately send all media file descriptions matching the search string in the packet back to the requesting peer; the peers will then forward the search packet to all the peers they know about.

The search process does have a problem in that you can never be sure your search packet reaches a peer that has the information you want. Further, the process of broadcasting the search packet to all known peers has a predictable consequence of using high bandwidth. If you increase the maximum hops allowed for the search, you might find your information, but there will also be a penalty in search time. In addition, when the search result is sent back to the originator, it must pass back through the same sequence of peers it initially traveled. Figure 1.6 shows an example of passing a search request between three sites.



**Figure 1.6** The Gnutella search process.

In this example, Site A is requesting all files that match the phrase *Rush*. Site A sends the search packet to Site B. Site B creates a list of the items requested by Site A (assigning a unique request number to the search packet), sends any matches it has locally back to Site A, and then forwards the search packet to Site C. If Site C has any matching files, the results will not be directly sent to Site A, but instead to Site B. When Site B receives the results from Site C, it checks its list, sees that Site A originally requested the information, and forwards it to Site A.

## Morpheus/KaZaA

Another file-sharing system, called Morpheus, was developed by MusicCity to replace a central server system with a decentralized system. Morpheus is based on FastTrack's P2P Stack, also used in the KaZaA file-sharing system. Morpheus and KaZaA aren't limited in their sharing of file types. You will find audio, video, images, documents, and even software in the two applications' networks. The two systems have improved the technologies involved in file download and search on a decentralized system by allowing for file restarts during download and by keeping lists of multiple peers who have the same file available.

Although the peers are basically decentralized, a central server is still used for providing username/password functionality and maintaining the overall network. In addition, the systems use a pure hybrid topology, as shown earlier. When a peer logs on, it is associated with a peer hub. These peer hubs are responsible for maintaining a list of the media files on their peers, and assisting

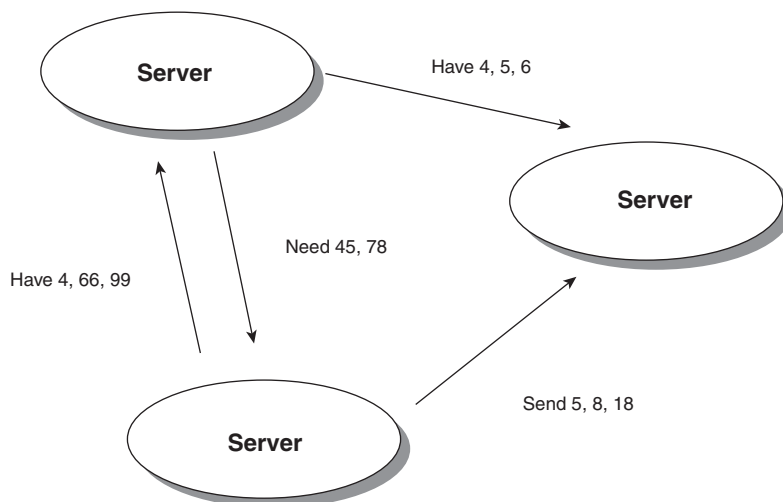
in the search requests between peers and peer hubs. If you are a peer on the network, and you have high bandwidth and a good deal of CPU power, your peer can be elected a peer hub automatically; however, you can select an option to not become a peer hub.

The hub peers are very important to the overall efficiency of the P2P network. Individual peers don't need to send requests to every peer in the network or worry about a max hops value—they send requests to their hub peer. The hub peer can quickly answer requests with information about media residing on the other peers in the hub. At the same time, the hub peers can contact other hub peers to find even more results. The amount of network traffic involved in a search is drastically reduced.

Media is transferred between peers on a purely peer-to-peer basis with no intermediary peers propagating them. The files are transferred using HTTP in order to reach peers behind firewalls. It should be noted that all transfers display the IP address of the machines involved in the transfer.

## Usenet

One of the oldest peer-to-peer systems, Usenet is based on a hybrid topology in which server nodes will be contacted for information from clients, and the server nodes will communicate with one another to ensure widespread distribution of information. The server nodes in Usenet are really the peers in the network. They have information to share and also request updates as needed from other peers. For the most part, the server nodes will all contain the same information if they choose to keep all newsgroups. Figure 1.7 shows an example of how the server nodes communicate to keep one another up-to-date.



**Figure 1.7** The Usenet server message-sharing process.

Client applications will connect to the server peers to obtain a listing of title messages available. When a message is selected, the actual data behind the title will be sent. It is obvious from this description that the developers of Usenet built an early prototype of a peer-to-peer system.

## Instant Messaging

Instant messaging systems, such as AOL Instant Message (AIM) and Yahoo! Instant Message, typically work in a topology in which central servers are used to coordinate your group of connections. When you log into an instant messaging service, the server will keep a temporary list of your contacts. The server will check to see if any of your contacts is currently logged into the system. If so, the IP and port information for the instant messaging client on the contact will be provided to your client, and your client information will be provided to the contact. From that point on, all communication between you and the contact will be in a peer-to-peer fashion.

## Extreme Peer-to-Peer: Distributed Computational Engines

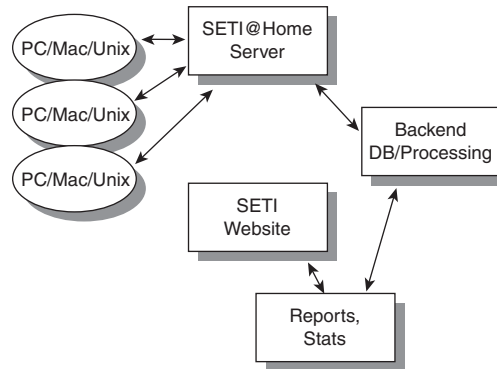
Several years ago, a tremendous piece of software was created that revolutionized the way Grand Challenge and computationally complex problems were solved. This software follows the traditional habit of building larger and larger supercomputers to solve problems. While distributed.net ([www.distributed.net](http://www.distributed.net)) wasn't the first system to use the idle CPU cycles of machines on a network, they are certainly the biggest. Using client software that runs as a background process in a "nice" mode, distributed.net has announced its intention to break RSA Labs' 56-bit secret-key code in response to a challenge put forth by RSA Labs.

When installed, the client software contacts a central server and requests packets of encrypted data that it will attempt to decrypt using a brute-force algorithm. When those packets have been processed, the results are sent to the central authority and a new set of packets is retrieved.

The resulting computation power and magnitude of machines involved is enormous. For example, there are on average 33,000 participants active on any given day. All of those participants have a minimum of one computer, and some are using entire labs. The participants are working through 92,141,082 keys per second, which equates to roughly .01% of the entire keyspace every day.

When the popularity of distributed.net rose, another group decided to use the same principle in the search for extraterrestrials. SETI@home (which stands for Search for Extraterrestrial Intelligence) produced a client that works in much the same way. The client receives signal data from the SETI installation,

and applies an algorithm to the information. The results are sent back to a central server, and more data is retrieved for processing. Figure 1.8 shows the transfer part of the system, which can be loosely called a peer-to-peer network.



**Figure 1.8** A traffic example within SETI.

## Warnings

While peer-to-peer systems offer great benefits in resource distribution, communication, and problem solving, developers and users alike should be aware of unresolved issues involved in using them. In this section, we briefly discuss four broad issues to raise your awareness level and spur your own conversations and research.

## Workplace Policies

Recently, there was a case in which a university system administrator installed a distributed computational engine on some of the computers under his control. When the university found out, he was charged with theft. In effect, he had stolen both CPU utilization and bandwidth of those computers. While there is much argument over the dollar figures provided by the university, you must be careful about how machines in your care are used.

The company I work for has a strict policy: No peer-to-peer systems or software shall operate on company computers. Before using a peer-to-peer system at work, check with those who make and enforce the policy.

## Intellectual Property

One of the major faults commonly associated with the Internet is that it enables users to inappropriately distribute copyrighted material quickly and on a global basis. This issue has caused many heated arguments among consumers,

distributors, copyright owners, artists, civil libertarians, and others; but it all comes down to what the law says about intellectual property. If you buy something and offer it for sale or give it away, there is usually no problem because the property has been transferred to another individual (one notable exception to this is software that is typically licensed, not purchased). If you buy something and make a copy of it, and then you offer that copy for sale or give it away, you are no longer transferring the property you purchased—instead, you are transferring a copy and keeping the original. The law says this is theft, and this is ultimately why Napster failed. Be careful what you offer to other peers in a peer-to-peer system.

## Bandwidth Costs

Peer-to-peer applications eat bandwidth for lunch, and as we all know, there are no free lunches. In many cases, a peer-to-peer application will be in violation of an ISP's service agreement because it can act as a server. ISPs put these policies in place to guard their precious bandwidth.

In addition to the bandwidth costs, there is a real concern for peer-to-peer systems that use broadcast mechanisms to locate other peers. As messages propagate across the Internet, more and more “garbage” broadcast packets are bouncing off the infrastructure. There have even been cases of denial of service occurring because of the volume of broadcast messages occurring on a network. In some cases, it might be preferable for peer-to-peer applications to operate on private networks with a limited connection to the “outside world.”

## Security

Internet applications are known for security holes. What kind of access does a remote peer have to your computer when it makes a request to the peer software you are running? Do you really know what information is gathered and sent to some remote server? What about the application itself? Is it secure? Buggy? Does it contain a Trojan horse? Most recently, it was released that the KaZaA client contained a stealth application designed to use the spare CPU cycles of the machine it was installed on. The stealth nature of the application allowed it to process work undetected by the computer's owner. These are all questions and situations that you must ask when using peer-to-peer software. And as a developer, you have a responsibility to ensure that the peer-to-peer applications you create are secure.

## Summary

---

This chapter has enumerated several clear and not-so-clear examples of peer-to-peer applications. Most of the applications, which allow the sharing of information over the Internet, have had a profound effect on society. Sometimes, the effect is to challenge the norm, and in other cases, it provides the ability to get more work done through increased communication. Throughout this book, we will learn about the tools available and necessary to build peer-to-peer systems.

