Open Peer to Peer Technologies

Independent Study CS 5974

Srikanth Koneru skoneru@vt.edu homepage

Instructor : Dr. Edward Fox fox@vt.edu homepage

Department of Computer Science, Virginia Tech

August 2002

Introduction

Definition of open peer-to-peer

Peer-to-peer applications allow us to separate out the concepts of authoring information and publishing that same information. It allows for decentralized application design, something that is both an opportunity and a challenge.

All the peer-to-peer applications, in various ways, return the content, choice, and control to ordinary users. Tiny end points on the Internet, sometimes even without knowing each other, exchange information and form communities. In these applications there are no more clients and servers, instead the communication takes place between cooperating peers.

There are many applications nowadays which are being labeled as peer-to-peer. A way to examine the distinction of whether an application is peer-to-peer or not is to check on the owner of the hardware that the service runs on. Like Napster, if the huge part of the hardware that Napster runs on is owned by the Napster users on millions of desktops then it is peer-to-peer. Peer-to-peer is a way of decentralizing not only features, but costs and administration also. By decentralizing data and therefore redirecting users so they download data directly from other user's computers, Napster reduced the load on its servers to the point where it could cheaply support tens of millions of users. The same principle is used in many commercial peer-to-peer systems. In short peer-to-peer cannot only distribute files. It can also distribute the burden of supporting network connections. The overall bandwidth remains the same as in centralized systems, but bottlenecks are eliminated at central sites and equally importantly, at their ISPs.

Search techniques are important to making peer-to-peer systems useful. But there is a higher level of system design and system use. Topics like trust, accountability and metadata have to be handled before searching is viable.

Usenet and DNS: Fathers of peer-to-peer in computers

Usenet is a news system, which has evolved some of the best examples of decentralized control structures on the Net. There is no central authority that controls this news system. Usenet is in someway the grandfather of today’s new peer-to-peer applications such as Gnutella and Freenet. Usenet is a system with no central control and copies files between computers. It has been around since 1979.

Another fixture, which has been using peer-to-peer networking from a long while, is the DNS system. It blends peer-to-peer networking with a hierarchical model of information ownership. DNS was built to distribute the data sharing (IP addresses and domain names) across the peer-to-peer Internet and it scaled remarkably well.

The lessons from Usenet and DNS are directly applicable to many contemporary peer-to-peer data sharing applications because the problems faced today by new peer-to-peer

applications systems such as file sharing are quite similar to the problems that Usenet and DNS addressed 10 or more years ago.

Napster

Napster, an application that allows people to search for and share MP3 music files, became a hot new Internet application on college campuses. In December 1999, the Recording Industry Association of America (RIAA) filed suit against Napster, alleging "contributory and vicarious" copyright infringement.

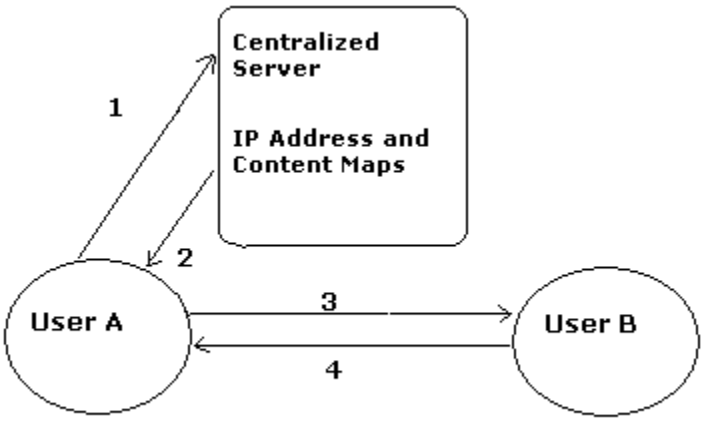


Figure: 1:User A requests a file "xyz", 2:Central server returns the IP address of a user who has file "xyz",3: User A requests "xyz" from User B,4: User B sends "xyz" to User A.

Any system that translates names into Internet numbers is a name space. Napster is a name space: When you register on Napster, you assign a name to your computer. When another Napster user wants to communicate with you, the Napster server translates this name into the Internet address of your computer. The Napster server acts as a name server and a search engine, all using proprietary protocols. (The underlying protocols are, of course, the standard Internet protocols.)

Users sign up on a Napster server with whatever name they want to use. User names are assigned on a first-come-first-served basis. Registration is instant, free, and requires no contact or other personal information.

You can use the Napster name space only to exchange files and chat, but the Napster system could easily be extended to handle web pages and e-mail, and probably will be soon. Yet, Napster has substantial security weaknesses. These limitations are described below.

Napster makes real what up until now has mostly been a straw man in the DNS debate: The DNS needs the Internet, but the Internet does not need any particular

name space. If various entities create their own name spaces like Napster does, anonymity will prosper and grow rapidly.

The famous problem Napster faced is the copyright problem. However, discussion of intellectual property rights is not pursued further in this report, since this document focuses more on the technical details involved with peer-to-peer technologies like Napster.

Gnutella

Each piece of **Gnutella** software is both a server and a client in one, because it supports bi-directional information transfer. You can be a fully functional Gnutella site by installing any of several available clients. Many different operating systems are supported. Next you have to find a few sites that are willing to communicate with you: some may be friends, while others may be advertised Gnutella sites. People with large computers and high bandwidth will encourage many others to connect to them.

You will communicate directly only with the handful of sites you've agreed to contact. Any material of interest to other sites will pass along from one site to another in store-and-forward fashion.

Because Gnutella runs over the Internet, you can connect directly with someone who's geographically far away just as easily as with your neighbor. This introduces robustness and makes the system virtually failsafe.

The protocol for obtaining information over Gnutella is a call-and-response.

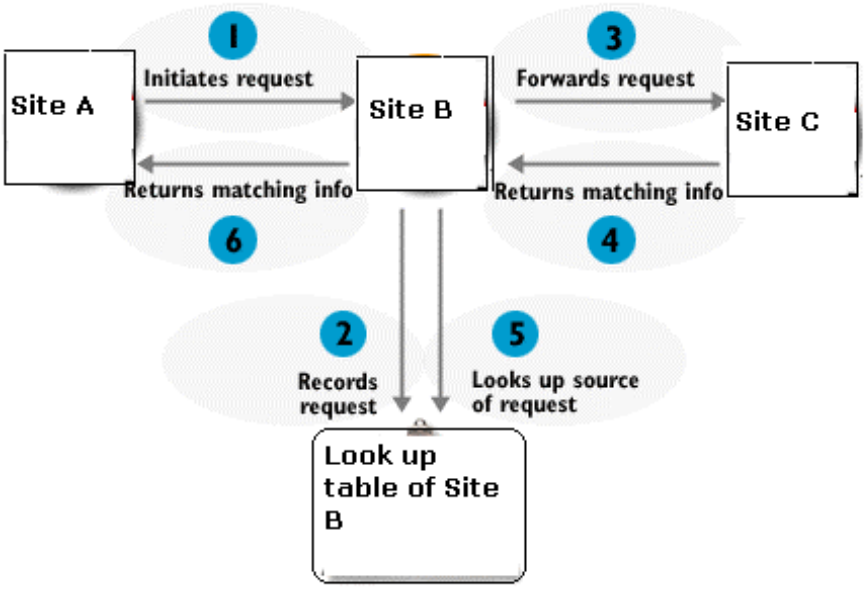


Figure: How Gnutella retrieves information

The figure shows the operation of the protocol. Suppose site A asks site B for data matching "MP3." After passing back anything that might be of interest, site B passes the request on to its colleague at site C -- but unlike mail or news, site B keeps a record that

site A has made the request. If site C has something matching the request, it gives the information to site B, which remembers that it is meant for site A and passes it through to Site A.

Each request has a unique number like an Ethernet MAC address. This helps in avoiding multiple requests. Also each site lets requests time out, simply by placing them on a queue of a predetermined size and letting old requests drop off the bottom as new ones are added.

Gnutella runs over HTTP (a sign of Gnutella's simplicity). A major advantage of using HTTP is that two sites can communicate even if one is behind a typical organization's firewall, assuming that this firewall allows traffic out to standard Web servers on port 80.

A Gnutella request has a time-to-live, which is normally decremented by each site until it reaches zero. This way there is a limit to the searching on the distributed system.

Gnutella only defines how a string is passed from one site to another, not how each site interprets the string. So depending on the site, the string may be handled by running *fgrep* or a *SQL query* or any other customized search. This flexibility allows each site to contribute to a distributed search in the most sophisticated way it can.

Limitations of Gnutella

Gnutella has problems scaling. The exponential spread of requests opens up the most likely source of disruption: denial-of-service attacks caused by flooding the system with requests. The developers have no particular solution at present, but suggest that clients keep track of the frequency of requests so that they can recognize bursts and refuse further contact with offending nodes.

Furthermore, the time-to-live imposes a horizon on each user. I may repeatedly search a few hundred sites near me, but I will never find files stored a step beyond my horizon.

Gnutella has already suffered service disruptions, mostly because of bugs in clients, and in the future it is certain to be attacked with vicious and sophisticated attempts to bring it down. While some groups of sites have slowed down temporarily or become severed from other groups, the system has never actually come down.

Another limitation of Gnutella is the difficulty in authenticating the source of the data returned. But if a digital signature infrastructure becomes widespread, clients could use that too.

Freenet

Freenet like Gnutella searches for information and returns information without telling you where it came from. Both Freenet and Gnutella are innovative in the areas of distributed information storage, information retrieval, and network architecture. But Freenet differs significantly from Gnutella in both goals and implementation.

The goals of Gnutella and Freenet are very different. Those of Freenet are more explicitly socio-political. Freenet allows people to distribute material anonymously, retrieve material anonymously, and also makes the removal of material very difficult. Also one goal in which Gnutella and Freenet differ from Napster is the goal of operating without any centralized control. Thus, a court order can shut down Napster, but shutting down Freenet or Gnutella is nearly impossible, because every user who exchanged copyrighted stuff using these protocols would need to be caught.

In addition to serving the social goals listed above, Freenet offers an intriguing possible solution to the problem of Internet congestion, because popular information automatically propagates to many sites.

The Freenet architecture and protocol is similar to Gnutella in many ways. Each cooperating person downloads a client and sends requests to a few other known clients. Requests are uniquely marked, are handed from one site to another, are temporarily stored on a stack so that data can be returned, and are dropped after each one's time-to-live expires.

The main difference between the two systems is that when a Freenet client satisfies a request, it passes the entire data to the requester. This is an option in Gnutella but is not required. Even more important, as the data passes back along a chain of Freenet clients to the original requester, each client keeps a copy. The client keeps the data so long as other people keep asking for it, but discards the data after some period of time when no one seems to want it. This achieves the transience required to meet the goals of anonymity and persistence. It lets small sites indirectly distribute large, popular documents without suffering bandwidth problems. It rewards popular material and allows unpopular material to disappear quietly. It tends to bring data close to those who want it. This is because the first request from node A to node B may have to pass through many other nodes, but the second and subsequent requests by nodes nearer to A can be satisfied by node A directly without haing to pass through all the nodes from A to B.

Freenet is more restrained in the traffic generated than Gnutella, perhaps because it expects to transfer a complete file of data for each successful request. When a Freenet client receives a request it cannot satisfy, it sends the request on to a single peer; it does not multicast to all peers as Gnutella does. In brief, searching is done depth-first and not in parallel. Freenet is being developed in Java and requires the Java Runtime Environment to run. It uses its own port and protocol, rather than running over HTTP as Gnutella does.

Freenet uses a unique identifier for each resource. This prevents malicious users from replacing resources with hoaxes. This unique identifier is Freenet's current weak point. Although someone posting material can assign any string as an identifier, Freenet chooses for security reasons to hash the string. Two search strings that differ by a single character (like "Human Rights" and “Human-Rights") will hash to very different values, as with any hashing algorithm. This hashing renders Freenet unusable for random searches. Free text searches become difficult because unless you know the keyword the serving system used, it is hard to make Freenet reach that server. For more information on Key Hashing and Keywords please refer to http://freenetproject.org/cgi-bin/twiki/view/Main/ICSI#SECTION00031000000000000000

OpenP2P

Some of the other P2P systems that have gained reputation are

Name	Description and URL
Hailstorm	HailStorm is Microsoft's system aimed towards its .NET initiative. It is a set of Web services whose data is contained in a set of XML documents, and which is accessed from the various clients (or "HailStorm endpoints") via SOAP (Simple Object Access Protocol). http://www.openp2p.com/topics/p2p/hailstorm/
Jabber	Jabber is a non-proprietary distribution model with wide platform support (including even the Newton MessagePad), wide choice of clients, interoperability with other instant messaging systems, ease of setup/use, and a large number of active subprojects ranging from clients and servers to custom-built transport server applications and various Jabber development tools. http://www.jabber.org/
JXTA	JXTA technology is a set of open protocols that allow any connected device on the network ranging from cell phones and wireless PDA's to PCs and servers to communicate and collaborate in a P2P manner. http://www.jxta.org/
Publius	Publius is a Web publishing system that is highly resistant to censorship and provides publishers with a high degree of anonymity. http://publius.cdt.org/
SETI@home	**SETI@home** is a scientific experiment that uses Internet-connected computers in the Search for Extraterrestrial Intelligence. http://setiathome.ssl.berkeley.edu/

P2P Architecture

The peer-to-peer explosion has reminded people of the power of decentralized systems. The promise of robustness, open-endedness, and infinite scalability has made many people excited about decentralization. But in reality, most systems we build on the Internet are largely centralized.

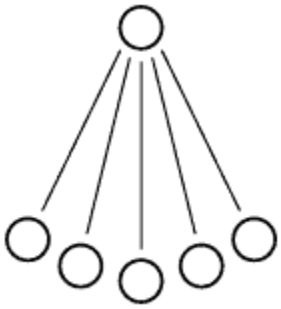


Figure: Centralized Architecture

Four basic topologies are in use on the Internet: centralized and decentralized, but also hierarchical and ring systems. These topologies can be used by themselves, or combined into one system creating hybrid systems.

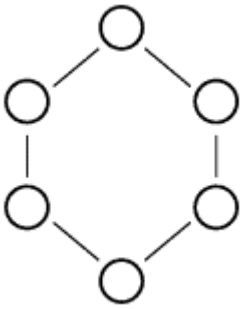


Figure: Ring Architecture

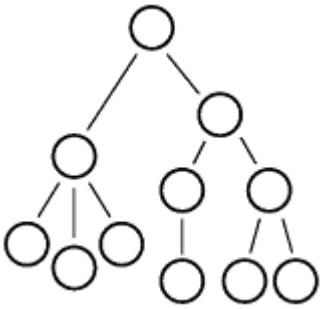


Figure: Hierarchical Architecture

SETI@Home is a fully centralized architecture with the job dispatcher as the server. And the original Napster's search architecture was centralized, although the file sharing was not.

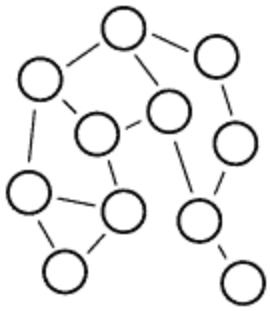


Figure: Decentralized Architecture

In decentralized systems, all peers communicate symmetrically and have equal roles. Gnutella is probably the most "pure" decentralized system used in practice today, with only a small-centralized function to bootstrap a new host. Many other file-sharing systems also are designed to be decentralized, such as Freenet. Decentralized systems are not new; the Internet routing architecture (http://www.icir.org/floyd/evolution.html) itself is largely decentralized, with the Border Gateway Protocol (http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/bgp.htm) used to coordinate the peering links between various autonomous systems.

There are limitless possibilities in combining various kinds of architectures. A centralized system could have a hierarchy of machines in the role of server. Decentralized systems could be built that span different rings or hierarchies. Systems could conceivably be built with three or more topologies combined, although the resulting complexity may be too difficult to manage.

By looking at systems in terms of their topology, it is possible to examine a wide spectrum of systems we have on the Internet today, from centralized to decentralized and with architectures in between.

Some of the properties that must be taken into consideration before designing the system are:

Manageability

> How hard is it to keep the system working? Complex systems require management: updating, repairing, and log analysis.

Extensibility

> How easy is it to grow the system, to add new resources to it? The Web is the ultimate extensible system; anyone can create a new Web server or Web page and immediately have that contribution be part of the Web.

Fault Tolerance

How well can the system handle failures? Fault tolerance is a necessity in large distributed systems.

Security

How hard is it to subvert the system? Security covers a variety of topics, such as preventing people from taking over the system, injecting bad information, or using the system for a purpose other then for what the owners intend.

Resistance to lawsuits and politics

How hard is it for an authority to shut down the system? The designers of Gnutella or Freenet consider their resistance to lawsuits to be one of their best features. Other parties consider this property to be a danger.

Scalability

How large can the system grow? Scalability is often promoted as a key advantage of decentralized systems over centralized, although the reality is more complex.

Conclusion on Topologies

A decentralized system is not always better or worse than a centralized system. The choice depends entirely on the needs of the application. The simplicity of centralized systems makes them easier to manage and control, while decentralized systems may grow more easily and be more resistant to failures or shutdowns.

As for scalability, the story is not clear. Centralized systems have limited scalability, but that limit is easy to understand. In contrast, decentralized systems offer the possibility of massive scalability, but in practice that can be very hard to achieve.

The second conclusion is the power of creating hybrid topologies. In centralized+ring systems, the ring covers many of the drawbacks of a purely centralized approach, providing easy scalability and fault tolerance. And centralized+decentralized systems are showing powerful scalability and extensibility while retaining some of the coherence of centralized systems.

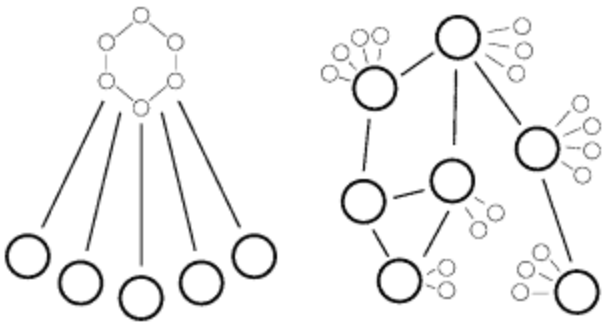


Figure: centralized+ring architecture, centralized+decentralized architecture.

Research into distributed applications and infrastructure has a very wide application. Centralized systems are evolving towards decentralization, as they grow larger and scale upward. A well-known example is how the host's file on the Internet became the Domain Name System. So centralized systems evolve toward decentralization. In an intriguing, complementary operation, decentralized or peer-to-peer sites are evolving toward centralization, also in response to growth and the need to scale upward. Gnutella now has super peers. Freenet provides gateways, JXTA Search creates a hierarchy of servers, and so on.

System designers have to evaluate the requirements for their particular area and pick a topology that matches their needs. We are not limited to a few simple topologies; topologies can be combined to make hybrids. And while centralized systems are doing a lot of the work on the Internet, there is a lot of exciting potential in decentralized systems. In particular, combining decentralized topologies with other simpler topologies is a powerful approach.

Issues in P2P

These are the issues that are involved in Peer-to-peer technologies which determine the future of P2P technologies.

Accountability	Traditionally file systems and communication media use accountability to maintain centralized control over their respective resources.
Fault Tolerance	The ability of a system to respond gracefully to an unexpected hardware or software failure.
Interoperability	Interoperability here refers to the ability to operate without problems between two gateways.
Metadata	Data about data. Metadata describes how and when and by whom a particular set of data was collected, and how the data is formatted. Metadata is essential for understanding information stored in repositories.
Performance	Performance determines the capability of the system, e.g., how long will it take to download this file and how much bandwidth a query will consume.
Reputation	Reputation is the memory and summary of behavior from past transactions. In real life, we use it to help us set our expectations when we consider future transactions. The same concept can be applied to peer-to-peer systems.
Scalability	Scalability refers to how well a system can adapt to increased demands.
Security	Refers to techniques for ensuring that data stored in a computer can be read and not compromised.
Trust	Trust in peer-to-peer, collaborative or distributed systems is about believing web sites or just surfing the web, when downloading and installing software or buying a product.

P2P and Education

Open Peer-to-peer Technologies in Education.

P2p in Education is still in its nascent stages and lots of research needs to happen in this area.

Rapid interaction, efficient data sharing, and the combined processing of inputs from many different sources; all these are supported well by peer-to-peer technologies. These features of peer-to-peer systems are amicable for the formation of various study groups in academia. Students as well as professors can make use of these features.

Kepler (http://kepler.cs.odu.edu/) is a digital library system for the individual, and it is based on the peer-to-peer model. This allows average researchers at an average university to publish results and disseminate them to a wide audience quickly and conveniently. The registration service in Kepler keeps track of the status of the registered archivelets in support of higher-level services. This is similar to the Napster centralized model, where the central server keeps track of active clients. In Kepler the publisher and the retriever of documents can be the same and hence it is peer-to-peer in nature. Scalability and unreliability (when machines go down) are issues for further research.

Kepler could be used by anyone to establish an OAI archive. Kepler's approach is based on the OAI that defines the open interface between data provider and service provider to implement digital library interoperability based on a harvesting approach. The intention of OAI had been to support data providers or archives that exist at the organizational level. To be a part of the OAI framework, a data provider needs to be 'open' in so far as it needs to support the OAI metadata harvesting protocol.

LOCKSS (**Lots of copies keep stuff safe**)
(http://lockss1.stanford.edu/uidemo/) It is an Internet "appliance", or "easy to use" software, designed to preserve access to authoritative versions of web-published materials. The current version of LOCKSS software is restricted to electronic journals. Web published materials are increasingly the authoritative versions. There are no affordable, widely available techniques for preserving this "written record". The web is an effective publishing medium (data sets, dynamic lists of citing papers, e-mail notification of citing papers, hyperlinks, searching). As web editions increasingly become the 'version of record', paper versions of the same titles are merely a subset of peer reviewed scholarly discourse. Librarians need an inexpensive, robust mechanism, that they control, to ensure their communities maintain long-term access to this important literature.

LOCKSS has the potential to become a sustainable, affordable, preservation tool and archiving system for web delivered information. LOCKSS software systematically caches content in a self-correcting P2P network. Using a decentralized, peer-to-peer network of like holdings at other participating libraries, the LOCKSS system would allow libraries to retain indefinite access to subscribed journal issues, even if the publisher's online site goes down or if he goes out of business.

Internet2 (http://www.internet2.edu/)
This project focuses on next-generation high-speed connections, where multimedia content is supported with suitable quality of service. In particular, R&D is overcoming many barriers that are holding back the deployment of peer-to-peer products in current corporate environments. Internet2 is a good test bed for basic research that can benefit work on peer-to-peer architectures.

Conclusion

Gnutella and Freenet continue to loosen the virtual from the physical, a theme that characterizes network evolution. DNS decoupled names from physical systems; URNs will allow users to retrieve documents without domain names; virtual hosting and replicated servers change the one-to-one relationship of names to systems. Perhaps it is time for another major conceptual leap, where we let go of the notion of location.

Within the peer-to-peer systems model, a number of modifications and tradeoffs can be used to tailor different sets of performance outcomes.

Freenet emphasizes high scalability and efficient searches under average conditions while it sacrifices worse case performance. At the other end, Gnutella sacrifices efficiency for faster searches and better worst-case guarantees. Ideas drawn from graph theory and the small world model can help to quantify these trade-offs and to analyze systems in concrete terms.

The peer-to-peer model encompasses a diverse set of approaches. By recognizing the wide range of possibilities available, inventing new ideas and new combinations, and using analytical methods to evaluate their behaviors, systems designers will be well equipped to exploit the power of peer-to-peer.

Will peer-to-peer replace the client/server model entirely? Most probably it won't. Client/server model remains extremely useful for many purposes, particularly where one site is recognized as the authoritative source for information and wants to maintain some control over that information. Client/server is also a much simpler model than peer-to-peer, and we should never abandon simplicity for complexity without a clear benefit. Client/server rarely presents administrative problems except where the amount of traffic exceeds the server's capacity.

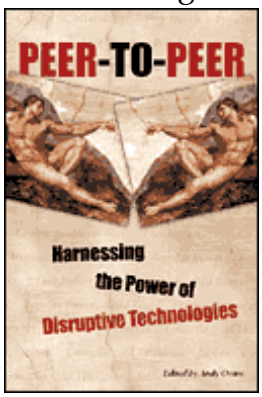
Peer-to-peer is useful when the goods you are trying to get at lie at many end points, i.e, where the value of information lies in the contribution of many users rather than the authority of one.

Peer-to-peer systems also can be a possible solution to bandwidth problems, when designed carefully. They are particularly useful in disseminating popular information and avoiding bottlenecks at the popular sites. But they can also cause bandwidth problems, either because their design adds too much overhead or because people just want large amount of data without paying for the bandwidth that can accommodate it.

The peer-to-peer models that are completely decentralized like Gnutella and Freenet are extremely valuable for research purposes. Whether or not other systems move in their direction, the viability of the most decentralized systems will help us judge the viability of peer-to-peer technology as a whole.

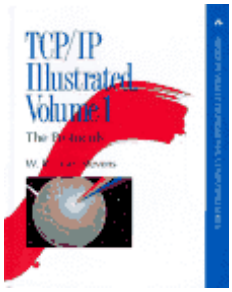
References

Most recent guide to Disruptive Technologies



Peer-to-Peer, Harnessing the Power of Disruptive Technologies, Edited by Andy Oram, O'Reilly, Cambridge, MA, USA, March 2001

Internet Routing Structure, BGP and other Networking Protocols



TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley, New Jersey, NJ, USA 1994, ISBN 0-201-63346-9

Links on the Internet

Border Gateway Protocol
www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/bgp.htm

Freenet
http://www.openp2p.com/topics/p2p/freenet/
http://freenetproject.org/cgi-bin/twiki/view/Main/WebHome

Gnutella
http://www.openp2p.com/topics/p2p/gnutella/

Hailstorm
http://www.openp2p.com/topics/p2p/hailstorm/

Internet2
http://www.internet2.edu/

Jabber
http://www.jabber.org/

JXTA
http://www.jxta.org/

Kepler
http://kepler.cs.odu.edu/

LOCKSS
http://lockss.stanford.edu

Napster
http://www.openp2p.com/topics/p2p/napster

OpenAchives Initiative
http://www.openarchives.org/

Peer-to-peer technologies
http://www.openp2p.com/

Publius
http://publius.cdt.org/

Security in Peer-to-peer Systems
http://www.openp2p.com/topics/p2p/security/

SETI at home
http://setiathome.ssl.berkeley.edu/