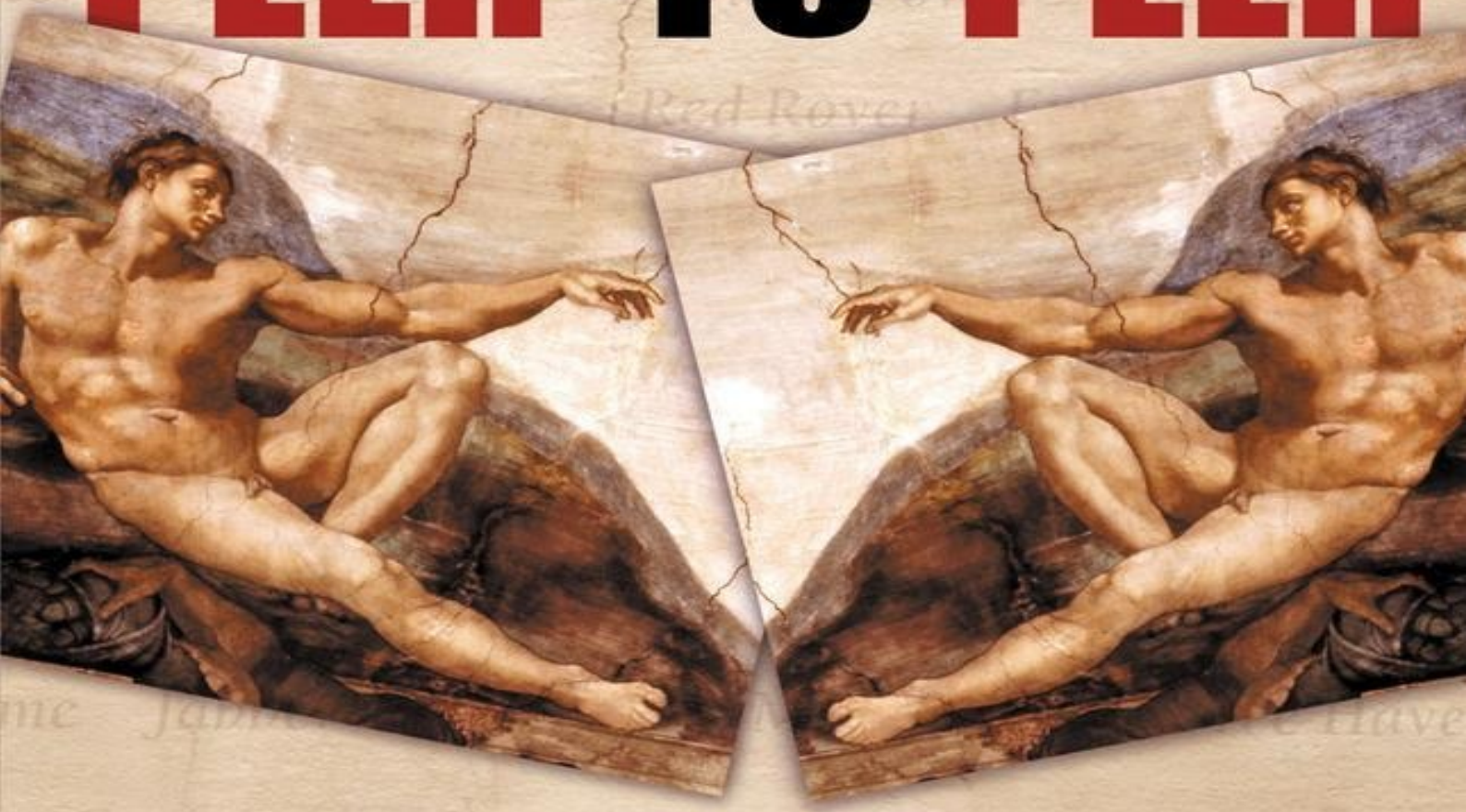
PEER-TO-PEER



Harnessing the Power of Disruptive Technologies

Edited by Andy Oram

Peer to Peer

Edited by

Andy Oram

Beijing • Boston • Farnham • Sebastopol • Tokyo

Special Upgrade Offer

If you purchased this ebook directly from oreilly.com, you have the following benefits:

- DRM-free ebooks — use your ebooks across devices without restrictions or limitations
- Multiple formats — use on your laptop, tablet, or phone
- Lifetime access, with free updates
- Dropbox syncing — your files, anywhere

If you purchased this ebook from another retailer, you can upgrade your ebook to take advantage of a
these benefits for just $4.99. Click here to access your ebook upgrade.

Please note that upgrade offers are not available from sample content.

Preface

Andy Oram, O’Reilly & Associates, Inc.

The term *peer-to-peer* rudely shoved its way to front and center stage of the computing field around the middle of the year 2000. Just as the early 20th-century advocates of psychoanalysis saw sex everywhere, industry analysts and marketing managers are starting to call everything they like in computers and telecommunications “peer-to-peer.” At the same time, technologists report that fear and mistrust still hang around this concept, sometimes making it hard for them to get a fair hearing from venture capitalists and policy makers.

Yes, a new energy is erupting in the computing field, and a new cuisine is brewing. Leaving sexiness aside, this preface tries to show that the term peer-to-peer is a useful way to understand a number of current trends that are exemplified by projects and research in this book. Seemingly small technological innovations in peer-to-peer can radically alter the day-to-day use of computer systems, as well as the way ordinary people interact using computer systems.

But to really understand what makes peer-to-peer tick, where it is viable, and what it can do for you, you have to proceed to the later chapters of the book. Each is written by technology leaders who are working ‘round the clock to create the new technologies that form the subject of this book. By following their thoughts and research, you can learn the state of the field today and where it might go in the future.

Some context and a definition

I mentioned at the beginning of this preface that the idea of peer-to-peer was the new eyebrow-raiser for the summer of 2000. At that point in history, it looked like the Internet had fallen into predictable patterns. Retail outlets had turned the Web into the newest mail order channel, while entertainment firms used it to rally fans of pop culture. Portals and search engines presented a small slice of Interne offerings in the desperate struggle to win eyes for banner ads. The average user, stuck behind a firewall at work or burdened with usage restrictions on a home connection, settled down to sending email and passive viewing.

In a word, boredom. Nothing much for creative souls to look forward to. An Olympic sports ceremon that would go on forever.

At that moment the computer field was awakened by a number of shocks. The technologies were not precisely new, but people realized for the first time that they were having a wide social impact:

Napster

This famous and immensely popular music exchange system caused quite a ruckus, first over its demands on campus bandwidth, and later for its famous legal problems. The technology is similar to earlier systems that got less attention, and even today is rather limited (since it was designed fo pop songs, though similar systems have been developed for other types of data). But Napster had a revolutionary impact because of a basic design choice: after the initial search for material, clients connect to each other and exchange data directly from one system's disk to the other.

SETI@home

This project attracted the fascination of millions of people long before the Napster phenomenon, and it brought to public attention the promising technique of distributing a computation across numerous personal computers. This technique, which exploited the enormous amounts of idle time going to waste on PCs, had been used before in projects to crack encryption challenges, but after SETI@home began, a number of companies started up with the goal of making the technique commercially viable.

Freenet

Several years before the peer-to-peer mania, University of Edinburgh researcher Ian Clarke started to create an elegantly simple and symmetric file exchange system that has proven to be among the purest of current models for peer-to-peer systems. Client and server are the same thing in this system; there is absolutely no centralization.

Gnutella

This experimental system almost disappeared before being discovered and championed by open source developers. It is another file exchange system that, like Freenet, stresses decentralization. I potential for enhanced searches is currently being explored.

Jabber

This open source project combines instant messaging (supporting many popular systems) with XML. The emergence of Jabber proclaimed that XML was more than a tool for business-to-busine (B2B) transaction processing, and in fact could be used to create spontaneous communities of ordinary users by structuring the information of interest to them.

.NET

This is the most far-reaching initiative Microsoft has released for many years, and they've announced that they're betting the house on it. .NET makes Microsoft's earlier component technology easier to use and brings it to more places, so that web servers and even web browsers can divide jobs among themselves. XML and SOAP (a protocol for doing object-oriented programming over the Web) are a part of .NET.

Analysts trying to find the source of inspiration for these developments have also noted a new world of sporadically connected Internet nodes emerging in laptops, handhelds, and cell phones, with more such nodes promised for the future in the form of household devices.

What thread winds itself around all these developments? In various ways they return content, choice, and control to ordinary users. Tiny endpoints on the Internet, sometimes without even knowing each other, exchange information and form communities. There are no more clients and servers — or at least, the servers retract themselves discreetly. Instead, the significant communication takes place between cooperating peers. That is why, diverse as these developments are, it is appropriate to lump them together under the rubric peer-to-peer.

While the technologies just listed are so new we cannot yet tell where their impact will be, peer-to-peer is also the oldest architecture in the world of communications. Telephones are peer-to-peer, as i the original UUCP implementation of Usenet. IP routing, the basis of the Internet, is peer-to-peer, even now when the largest access points raise themselves above the rest. Endpoints have also historically been peers, because until the past decade every Internet- connected system hosted both servers and clients. Aside from dial-up users, the second-class status of today's PC browser crowd didn't exist. Thus, as some of the authors in this book point out, peer-to-peer technologies return the Internet to its original vision, in which everyone creates as well as consumes.

Many early peer-to-peer projects have an overtly political mission: routing around censorship. Peer-to-peer techniques developed in deliberate evasion of mainstream networking turned out to be very useful within mainstream networking. There is nothing surprising about this move from a specialized and somewhat ostracized group of experimenters to the center of commercial activity; similar trends can be found in the history of many technologies. After all, organizations that are used to working within the dominant paradigm don't normally try to change that paradigm; change is more likely to come from those pushing a new cause. Many of the anti-censorship projects and their leaders are featured in this book, because they have worked for a long time on the relevant peer-to-peer issues an have a lot of experience to offer.

Peer-to-peer can be seen as the continuation of a theme that has always characterized Internet evolution: loosening the virtual from the physical. DNS decoupled names from physical systems, while URNs were meant to let users retrieve documents without knowing the domain names of their hosts. Virtual hosting and replicated servers changed 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. Welcome to the Heisenberg Principle as applied to the Internet.

The two-way Internet also has a social impact, and while this book is mostly about the technical promise of peer-to-peer, authors also talk about its exciting social promise. Communities have been forming on the Internet for a long time, but they have been limited by the flat interactive qualities of email and network newsgroups. People can exchange recommendations and ideas over these media, but they have great difficulty commenting on each other's postings, structuring information, performing searches, or creating summaries. If tools provided ways to organize information intelligently, and if each person could serve up his or her own data and retrieve others' data, the possibilities for collaboration would take off. Peer-to-peer technologies could enhance almost any

group of people who share an interest — technical, cultural, political, medical, you name it.

How this book came into being

The feat of compiling original material from the wide range of experts who contributed to this book i a story all in itself.

Long before the buzz about peer-to-peer erupted in the summer of 2000, several people at O'Reilly & Associates had been talking to leaders of interesting technologies who later found themselves identified as part of the peer-to-peer movement. At that time, for instance, we were finishing a book on SETI@home (*Beyond Contact*, by Brian McConnell) and just starting a book on Jabber. Tim O'Reilly knew Ray Ozzie of Groove Networks (the creator of Lotus Notes), Marc Hedlund and Nelso Minar of Popular Power, and a number of other technologists working on technologies like those in this book.

As for me, I became aware of the technologies through my interest in Internet and computing policy. When the first alarmist news reports were published about Freenet and Gnutella, calling them mechanisms for evading copyright controls and censorship, I figured that anything with enough powe to frighten major forces must be based on interesting and useful technologies. My hunch was borne out more readily than I could have imagined; the articles I published in defense of the technologies proved to be very popular, and Tim O'Reilly asked me to edit a book on the topic.

As a result, contributors came from many sources. Some were already known to O'Reilly & Associates, some were found through a grapevine of interested technologists, and some approached u when word got out that we were writing about peer-to-peer. We solicited chapters from several peopl who could have made valuable contributions but had to decline for lack of time or other reasons. I am fully willing to admit we missed some valuable contributors simply because we did not know about them, but perhaps that can be rectified in a future edition.

In addition to choosing authors, I spent a lot of effort making sure their topics accurately represented the field. I asked each author to find a topic that he or she found compelling, and I weighed each topi to make sure it was general enough to be of interest to a wide range of readers.

I was partial to topics that answered the immediate questions knowledgeable computer people ask when they hear about peer-to-peer, such as "Will performance become terrible as it scales?" or "How can you trust people?" Naturally, I admonished authors to be completely honest and to cover weaknesses as well as strengths.

We did our best, in the short time we had, to cover everything of importance while avoiding overlap. Some valuable topics could not be covered. For instance, no one among the authors we found felt comfortable writing about search techniques, which are clearly important to making peer-to-peer systems useful. I believe the reason we didn't get to search techniques is that it represents a relatively high level of system design and system use — a level the field has not yet achieved. Experiments are being conducted (such as InfraSearch, a system built on Gnutella), but the requisite body of knowledge is not in place for a chapter in this book. All the topics in the following pages — trust, accountability, metadata — have to be in place before searching is viable. Sometime in the future, when the problems in these areas are ironed out, we will be ready to discuss search techniques.

Thanks to Steve Burbeck, Ian Clarke, Scott Miller, and Terry Steichen, whose technical reviews were critical to assuring accurate information and sharpening the arguments in this book. Thanks also to th many authors who generously and gently reviewed each other's work, and to those people whose aid listed in particular chapters.

Thanks also to the following O'Reilly staff: Darren Kelly, production editor; Leanne Soylemez, who

was the copyeditor; Rachel Wheeler, who was the proofreader; Matthew Hutchinson, Jane Ellin, Sara Jane Shangraw, and Claire Cloutier, who provided quality control; Judy Hoer, who wrote the index; Lucy Muellner and Linley Dolby, who did interior composition; Edie Freedman, who designed the cover of this book; Emma Colby, who produced the cover layout; Melanie Wang and David Futato, who designed the interior layout; Mike Sierra, who implemented the design; and Robert Romano and Jessamyn Read, who produced the illustrations.

Contents of this book

It's fun to find a common thread in a variety of projects, but simply noting philosophical parallels is not enough to make the term peer-to-peer useful. Rather, it is valuable only if it helps us develop and deploy the various technologies. In other words, if putting two technologies under the peer-to-peer umbrella shows that they share a set of problems, and that the solution found for one technology can perhaps be applied to another, we benefit from the buzzword. This book, then, spends most of its tim on general topics rather than the details of particular existing projects.

Part I contains the observations of several thinkers in the computer industry about the movements tha have come to be called peer-to-peer. These authors discuss what can be included in the term, where it is innovative or not so innovative, and where its future may lie.

Chapter 1 describes where peer-to-peer systems might offer benefits, and the problems of fitting such systems into the current Internet. It includes a history of early antecedents. The chapter is written by Nelson Minar and Marc Hedlund, the chief officers of Popular Power.

Chapter 2 tries to tie down what peer-to-peer means and what we can learn from the factors that mad Napster so popular. The chapter is written by investment advisor and essayist Clay Shirky.

Chapter 3 contrasts the way the public often views a buzzword such as peer-to-peer with more constructive approaches. It is written by Tim O'Reilly, founder and CEO of O'Reilly & Associates, Inc.

Chapter 4 reveals the importance of maximizing the value that normal, selfish use adds to a service. It is written by Dan Bricklin, cocreator of Visicalc, the first computer spreadsheet.

Some aspects of peer-to-peer can be understood only by looking at real systems. Part II contains chapters of varying length about some important systems that are currently in operation or under development.

Chapter 5 presents one of the most famous of the early crop of peer-to-peer technologies. Project Director David Anderson explains why the team chose to crunch astronomical data on millions of scattered systems and how they pulled it off.

Chapter 6 presents the wonderful possibilities inherent in using the Internet to form communities of people as well as automated agents contacting each other freely. It is written by Jeremie Miller, leade of the Jabber project.

Chapter 7 covers a classic system for allowing anonymous email. Other systems described in this book depend on Mixmaster to protect end-user privacy, and it represents an important and long-standing example of peer-to-peer in itself. It is written by Adam Langley, a Freenet developer.

Chapter 8 offers not only an introduction to one of the most important of current projects, but also an entertaining discussion of the value of using peer-to-peer techniques. The chapter is written by Gene Kan, one of the developers most strongly associated with Gnutella.

Chapter 9 describes an important project that should be examined by anyone interested in peer-to-peer. The chapter explains how the system passes around requests and how various cryptographic key permit searches and the retrieval of documents. It is written by Adam Langley.

Chapter 10 describes a fascinating system for avoiding censorship and recrimination for the distribution of files using electronic mail. It is written by Alan Brown, the developer of Red Rover.

Chapter 11 describes a system that distributes material through a collection of servers in order to prevent censorship. Although Publius is not a pure peer-to-peer system, its design offers insight and

unique solutions to many of the problems faced by peer-to-peer designers and users. The chapter is written by Marc Waldman, Lorrie Faith Cranor, and Avi Rubin, the members of the Publius team.

Chapter 12 introduces another set of distributed storage services that promotes anonymity with the addition of some new techniques in improving accountability in the face of this anonymity. It is written by Roger Dingledine, Michael Freedman, and David Molnar, leaders of the Free Haven team.

In Part III, project leaders choose various key topics and explore the problems, purposes, and promis of the technology.

Chapter 13 shows how to turn raw data into useful information and how that information can support information seekers and communities. Metadata can be created through XML, RDF, and other standard formats. The chapter is written by Rael Dornfest, an O'Reilly Network developer, and Dan Brickley, a longstanding RDF advocate and chair of the World Wide Web Consortium's RDF Interes Group.

Chapter 14 covers a topic that has been much in the news recently and comes to mind immediately when people consider peer-to-peer for real-life systems. This chapter examines how well a peer-to-peer project can scale, using simulation to provide projections for Freenet and Gnutella. It is written by Theodore Hong of the Freenet project.

Chapter 15 begins a series of chapters on the intertwined issues of privacy, authentication, anonymity and reliability. This chapter covers the basic elements of security, some of which will be well known to most readers, but some of which are fairly novel. It is written by the members of the Publius team.

Chapter 16 covers ways to avoid the "tragedy of the commons" in shared systems — in other words, the temptation for many users to freeload off the resources contributed by a few. This problem is endemic to many peer-to-peer systems, and has led to several suggestions for micropayment systems (like Mojo Nation) and reputation systems. The chapter is written by leaders of the Free Haven team.

Chapter 17 discusses ways to automate the collection and processing of information from previous transactions to help users decide whether they can trust a server with a new transaction. The chapter i written by Richard Lethin, founder of Reputation Technologies, Inc.

Chapter 18 offers the assurance that it is technically possible for people in a peer-to-peer system to authenticate each other and ensure the integrity and secrecy of their communications. The chapter accomplishes this by describing the industrial-strength security system used in Groove, a new commercial groupware system for small collections of people. It is written by Jon Udell, an independent author/consultant, and Nimisha Asthagiri and Walter Tuvell, staff of Groove Networks.

Chapter 19 discusses how the best of all worlds could be achieved by connecting one system to another. It includes an encapsulated comparison of several peer-to-peer systems and the advantages each one offers. It is written by Brandon Wiley, a developer of the Freenet project.

Appendix A lists some interesting projects, companies, and standards that could reasonably be considered examples of peer-to-peer technology.

Peer-to-peer web site

O’Reilly has created the web site http://openp2p.com to cover peer-to-peer (P2P) technology for developers and technical managers. The site covers these technologies from inside the communities producing them and tries to profile the leading technologists, thinkers, and programmers in the P2P space by providing a deep technical perspective.

We’d like to hear from you

Please address comments and questions concerning this book to the publisher:

O’Reilly & Associates, Inc.
101 Morris Street Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)
(707) 829-0515 (international or local)
(707) 829-0104 (fax)

We have a web page for this book, where we list errata, examples, or any additional information. You can access this page at:

http://www.oreilly.com/catalog/peertopeer

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, software, Resource Centers, and the O’Reilly Network, see our web site at:

http://www.oreilly.com

Part I. Context and Overview

This part of the book offers some high-level views, defining the term “peer-to-peer” and placing current projects in a social and technological context.

Chapter 1. A Network of Peers: Peer-to-Peer Models Through the History of the Internet

Nelson Minar and Marc Hedlund, Popular Power

The Internet is a shared resource, a cooperative network built out of millions of hosts all over the world. Today there are more applications than ever that want to use the network, consume bandwidth and send packets far and wide. Since 1994, the general public has been racing to join the community of computers on the Internet, placing strain on the most basic of resources: network bandwidth. And the increasing reliance on the Internet for critical applications has brought with it new security requirements, resulting in firewalls that strongly partition the Net into pieces. Through rain and snow and congested Network Access Providers (NAPs), the email goes through, and the system has scaled vastly beyond its original design.

In the year 2000, though, something has changed — or, perhaps, reverted. The network model that survived the enormous growth of the previous five years has been turned on its head. What was down has become up; what was passive is now active. Through the music-sharing application called Napste and the larger movement dubbed “peer-to-peer,” the millions of users connecting to the Internet have started using their ever more powerful home computers for more than just browsing the Web and trading email. Instead, machines in the home and on the desktop are connecting to each other directly forming groups and collaborating to become user-created search engines, virtual supercomputers, an filesystems.

Not everyone thinks this is such a great idea. Some objections (dealt with elsewhere in this volume) cite legal or moral concerns. Other problems are technical. Many network providers, having set up their systems with the idea that users would spend most of their time downloading data from central servers, have economic objections to peer-to-peer models. Some have begun to cut off access to peer to-peer services on the basis that they violate user agreements and consume too much bandwidth (for illicit purposes, at that). As reported by the online News.com site, a third of U.S. colleges surveyed have banned Napster because students using it have sometimes saturated campus networks.

In our own company, Popular Power, we have encountered many of these problems as we create a peer-to-peer distributed computing resource out of millions of computers all over the Internet. We have identified many specific problems where the Internet architecture has been strained; we have also found work-arounds for many of these problems and have come to understand what true solution would be like. Surprisingly, we often find ourselves looking back to the Internet of 10 or 15 years ago to consider how best to solve a problem.

The original Internet was fundamentally designed as a peer-to-peer system. Over time it has become increasingly client/server, with millions of consumer clients communicating with a relatively privileged set of servers. The current crop of peer-to-peer applications is using the Internet much as i was originally designed: as a medium for communication for machines that share resources with each other as equals. Because this network model is more revolutionary for its scale and its particular implementations than for its concept, a good number of past Internet applications can provide lesson to architects of new peer-to-peer applications. In some cases, designers of current applications can learn from distributed Internet systems like Usenet and the Domain Name System (DNS); in others, the changes that the Internet has undergone during its commercialization may need to be reversed or modified to accommodate new peer-to-peer applications. In either case, the lessons these systems

provide are instructive, and may help us, as application designers, avoid causing the death of the Internet.[1]

A revisionist history of peer-to-peer (1969-1995)

The Internet as originally conceived in the late 1960s was a peer-to-peer system. The goal of the original ARPANET was to share computing resources around the U.S. The challenge for this effort was to integrate different kinds of existing networks as well as future technologies with one common network architecture that would allow every host to be an equal player. The first few hosts on the ARPANET — UCLA, SRI, UCSB, and the University of Utah — were already independent computin sites with equal status. The ARPANET connected them together not in a master/slave or client/server relationship, but rather as equal computing peers.

The early Internet was also much more open and free than today's network. Firewalls were unknown until the late 1980s. Generally, any two machines on the Internet could send packets to each other. Th Net was the playground of cooperative researchers who generally did not need protection from each other. The protocols and systems were obscure and specialized enough that security break-ins were rare and generally harmless. As we shall see later, the modern Internet is much more partitioned.

The early "killer apps" of the Internet, FTP and Telnet, were themselves client/server applications. A Telnet client logged into a compute server, and an FTP client sent and received files from a file serve But while a single application was client/server, the usage patterns as a whole were symmetric. Every host on the Net could FTP or Telnet to any other host, and in the early days of minicomputers and mainframes, the servers usually acted as clients as well.

This fundamental symmetry is what made the Internet so radical. In turn, it enabled a variety of more complex systems such as Usenet and DNS that used peer-to-peer communication patterns in an interesting fashion. In subsequent years, the Internet has become more and more restricted to client/server-type applications. But as peer-to-peer applications become common again, we believe the Internet must revert to its initial design.

Let's look at two long-established fixtures of computer networking that include important peer-to-peer components: Usenet and DNS.

Usenet

Usenet news implements a decentralized model of control that in some ways is the grandfather of today's new peer-to-peer applications such as Gnutella and Freenet. Fundamentally, Usenet is a system that, using no central control, copies files between computers. Since Usenet has been around since 1979, it offers a number of lessons and is worth considering for contemporary file-sharing applications.

The Usenet system was originally based on a facility called the Unix-to-Unix-copy protocol, or UUC UUCP was a mechanism by which one Unix machine would automatically dial another, exchange file with it, and disconnect. This mechanism allowed Unix sites to exchange email, files, system patches, or other messages. The Usenet used UUCP to exchange messages within a set of topics, so that students at the University of North Carolina and Duke University could each "post" messages to a topic, read messages from others on the same topic, and trade messages between the two schools. Th Usenet grew from these original two hosts to hundreds of thousands of sites. As the network grew, so did the number and structure of the topics in which a message could be posted. Usenet today uses a TCP/IP-based protocol known as the Network News Transport Protocol (NNTP), which allows two machines on the Usenet network to discover new newsgroups efficiently and exchange new messages in each group.

The basic model of Usenet provides a great deal of local control and relatively simple administration

A Usenet site joins the rest of the world by setting up a news exchange connection with at least one other news server on the Usenet network. Today, exchange is typically provided by a company's ISP. The administrator tells the company's news server to get in touch with the ISP's news server and exchange messages on a regular schedule. Company employees contact the company's local news server, and transact with it to read and post news messages. When a user in the company posts a new message in a newsgroup, the next time the company news server contacts the ISP's server it will notify the ISP's server that it has a new article and then transmit that article. At the same time, the ISP's server sends its new articles to the company's server.

Today, the volume of Usenet traffic is enormous, and not every server will want to carry the full complement of newsgroups or messages. The company administrator can control the size of the news installation by specifying which newsgroups the server will carry. In addition, the administrator can specify an expiration time by group or hierarchy, so that articles in a newsgroup will be retained for that time period but no longer. These controls allow each organization to voluntarily join the network on its own terms. Many organizations decide not to carry newsgroups that transmit sexually oriented or illegal material. This is a distinct difference from, say, Freenet, which (as a design choice) does no let a user know what material he or she has received.

Usenet has evolved some of the best examples of decentralized control structures on the Net. There is no central authority that controls the news system. The addition of new newsgroups to the main topic hierarchy is controlled by a rigorous democratic process, using the Usenet group *news.admin* to propose and discuss the creation of new groups. After a new group is proposed and discussed for a se period of time, anyone with an email address may submit an email vote for or against the proposal. I a newsgroup vote passes, a new group message is sent and propagated through the Usenet network.

There is even an institutionalized form of anarchy, the *alt.** hierarchy, that subverts the *news.admin* process in a codified way. An *alt* newsgroup can be added at any time by anybody, but sites that don' want to deal with the resulting absurdity can avoid the whole hierarchy. The beauty of Usenet is that each of the participating hosts can set their own local policies, but the network as a whole functions through the cooperation and good will of the community. Many of the peer-to-peer systems currently emerging have not yet effectively addressed decentralized control as a goal. Others, such as Freenet, deliberately avoid giving local administrators control over the content of their machines because this control would weaken the political aims of the system. In each case, the interesting question is: how much control can or should the local administrator have?

NNTP as a protocol contains a number of optimizations that modern peer-to-peer systems would do well to copy. For instance, news messages maintain a "Path" header that traces their transmission from one news server to another. If news server A receives a request from server B, and A's copy of message lists B in the Path header, A will not try to retransmit that message to B. Since the purpose o NNTP transmission is to make sure every news server on Usenet can receive an article (if it wants to) the Path header avoids a flood of repeated messages. Gnutella, as an example, does not use a similar system when transmitting search requests, so as a result a single Gnutella node can receive the same request repeatedly.

The open, decentralized nature of Usenet can be harmful as well as beneficial. Usenet has been enormously successful as a system in the sense that it has survived since 1979 and continues to be home to thriving communities of experts. It has swelled far beyond its modest beginnings. But in many ways the trusting, decentralized nature of the protocol has reduced its utility and made it an extremely noisy communication channel. Particularly, as we will discuss later, Usenet fell victim to spam early in the rise of the commercial Internet. Still, Usenet's systems for decentralized control, it

methods of avoiding a network flood, and other characteristics make it an excellent object lesson for designers of peer-to-peer systems.

DNS

The Domain Name System (DNS) is an example of a system that blends peer-to-peer networking wit a hierarchical model of information ownership. The remarkable thing about DNS is how well it has scaled, from the few thousand hosts it was originally designed to support in 1983 to the hundreds of millions of hosts currently on the Internet. The lessons from DNS are directly applicable to contemporary peer-to-peer data sharing applications.

DNS was established as a solution to a file-sharing problem. In the early days of the Internet, the way to map a human-friendly name like *bbn* to an IP address like *4.2.49.2* was through a single flat file, *hosts.txt*, which was copied around the Internet periodically. As the Net grew to thousands of hosts ar managing that file became impossible, DNS was developed as a way to distribute the data sharing across the peer-to-peer Internet.

The namespace of DNS names is naturally hierarchical. For example, O'Reilly & Associates, Inc. owns the namespace *oreilly.com*: they are the sole authority for all names in their domain, such as http://www.oreilly.com. This built-in hierarchy yields a simple, natural way to delegate responsibilit for serving part of the DNS database. Each domain has an *authority*, the name server of record for hosts in that domain. When a host on the Internet wants to know the address of a given name, it queries its nearest name server to ask for the address. If that server does not know the name, it delegates the query to the authority for that namespace. That query, in turn, may be delegated to a higher authority, all the way up to the root name servers for the Internet as a whole. As the answer propagates back down to the requestor, the result is cached along the way to the name servers so the next fetch can be more efficient. Name servers operate both as clients and as servers.

DNS as a whole works amazingly well, having scaled to 10,000 times its original size. There are several key design elements in DNS that are replicated in many distributed systems today. One element is that hosts can operate both as clients and as servers, propagating requests when need be. These hosts help make the network scale well by caching replies. The second element is a natural method of propagating data requests across the network. Any DNS server can query any other, but in normal operation there is a standard path up the chain of authority. The load is naturally distributed across the DNS network, so that any individual name server needs to serve only the needs of its clien and the namespace it individually manages.

So from its earliest stages, the Internet was built out of peer-to-peer communication patterns. One advantage of this history is that we have experience to draw from in how to design new peer-to-peer systems. 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 15 years ago.