Oral Comprehensive Exam: Position Paper

Content Distribution in Unstructured Peer-to-Peer Networks

Daniel Stutzbach
Department of Information and Computer Science
University of Oregon
agthorr@cs.uoregon.edu

December 9, 2004

1 Introduction

Previously, my research has focused on studying Peer-to-Peer File Transfer mechanisms through simulation under Professor Daniel Zappala. This formed the basis for my DRP and a paper currently under submission to a conference. More recently, I have focused on measurement studies of the Gnutella file-sharing network, working with Professor Reza Rejaie. Based on that work, we have a workshop paper under submission describing new measurement techniques and tools, and a conference paper under submission presenting an analysis of the Gnutella topology and its properties. We also have an additional workshop paper under submission exploring the dynamics of peer session length and arrival rate.

This is my position paper for the Oral Comprehensive Exam requirement of the Ph.D. program of the Computer and Information Science Department at the University of Oregon. In recent years, peer-to-peer file sharing applications have witnessed explosive growth. With millions of simultaneous users[1] and, in one University of Washington study [SGD+02], generating three times as much traffic as the Web, this is an important component of the Internet. In this paper, I explore three related topics: Peer-to-Peer Search, Peer-to-Peer File Transfer, and Peer-to-Peer Streaming.

Search From a users point of view, all file-sharing applications—Kazaa, eDonkey 2000, Gnutella—operate in basically the same way. The user types in what they're looking for, the application does a simple substring match on other users' filenames, and displays the results. Users can then select which files they'd like to download. Under-the-hood, these applications use different techniques to perform the search. The bulk of this position paper surveys measurement studies on existing file-sharing networks (Section 3) and simulation-driven studies that explore more efficient search techniques (Section 4). While the existing work in this area is considerable, as we shall see, there are some significant gaps.

Transfer Early file-sharing applications, such as Napster and the original Gnutella, used a simple HTTP-like protocol for the download. In effect, this was a client-server transfer between two peers that were located using the peer-to-peer search. Modern systems bring peer-to-peer techniques to the transfer mechanism with a technique known as *swarming*, which allows peers to use several sources in parallel and share fragments of the file before they've completed their download. There has been relatively little published work on this aspect of file-sharing systems. Section 5 covers the design of file transfer in peer-to-peer systems.

While file-sharing applications include two major pieces of functionality, search and transfer, a few applications take a component-based approach and implement only one of these features. The most notable example is BitTorrent, a pure swarming application that includes no search functionality.

[1] As reported by www.slyck.com

Streaming Closely related to swarming is peer-to-peer streaming, which shares many of the same technical problems and solutions. It introduces the additional constraint that packets need to arrive promptly to be useful. On the other hand, with Multiple Descriptor Coding, it relaxes the condition that every packet from the source must reach every destination. Because there are no major deployed peer-to-peer streaming systems, this papers examines work that characterize the load and user-behavior of client-server streaming systems, in addition to the few research papers that propose peer-to-peer streaming mechanisms (Section 6).

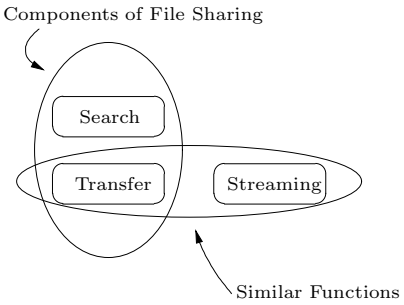

Figure 1: Components Covered

	Search	Transfer	Streaming
Examples	Gnutella Kazaa Gia	Slurpie Bullet BitTorrent	CoopNet PRO
Signaling	Keyword Search Overlay Maintenance	Locating and Selecting Blocks Overlay Maintenance	Selecting Layers or Frames Overlay Maintenance
Overlay	Flat, Ad-hoc Mesh Ultrapeers & Leaves	Flat, Ad-hoc Mesh Block-seeking Mesh	Delay-optimized Tress Bandwidth-optimized Mesh
Data Units	Best-effort Queries over TCP	Reliable Blocks over TCP	Best-effort Frames over UDP

Table 1: Topics in this paper

Relationship Figure 1 shows the relationship between the topics in this paper. Search and Transfer are both part of file-sharing applications, while Peer-to-Peer Transfer and Peer-to-Peer Streaming solve similar problems and operate in similar ways. Table 1 shows these three topics and illustrates some ways in which they are similar and different.

2 Measurement Techniques

Researchers have conducted several measurement studies on peer-to-peer file sharing systems over the last few years. These studies employ one of five basic techniques, each offering a different view with certain advantages and disadvantages. Before examining the measurement results in detail, it's useful to briefly examine these techniques.

Intercept These studies eavesdrop on peer-to-peer sessions passing through a router at an ISP or university.

Participate These studies instrument peer-to-peer software and allow it to run in its usual manner.

Crawl These studies walk the peer-to-peer network, capturing its topology.

Probe These studies select a subset of the peers in the network and probe them at regular intervals.

Centralize These studies rely on logs maintained by a central server.

Intercept	Participate	Crawl	Probe	Centralize
[GDS+03] (K) [LBBSS02] (K) [LRW03] (K) [SGD+02] (K,G) [SW04] (K,G,DC) [CWVL01] (S)	[HA04] (G) [KAD+04] (G) [KLVW04] (G) [Mar02] (G) [Sri01] (G) [AH00] (G) [EIPN04] (B)	[cli00] (G) [ABJ01] (G) [RFI02] (G)	[CLL02] (N,G) [SGG02] (N,G) [BSV03] (O) [FHKM04] (D)	[IUKB+04] (B) [VAM+02] (S) [SMZ03] (S) [hCGR+04] (S)

Table 2: File sharing measurement studies, grouped by technique. The system under study is shown in parenthesis. B=BitTorrent, D=eDonkey 2000, DC=Direct Connect, G=Gnutella, K=Kazaa, N=Napster, S=Streaming, O=Overnet

Table 2 summarizes the peer-reviewed studies in each category, and lists the particular systems that were examined. Studies which intercept data tend to prefer to examine Kazaa, one of the most popular peer-to-peer system in use today. [SGD+02] shows that in 2002, Kazaa traffic was between one and two orders of magnitude larger than Gnutella traffic. However, others studies tend to focus on Gnutella, which has several open source implementations available and open protocol specifications. Over the past few years, Gnutella has gained considerable market share relative to Kazaa, and currently has about half as many concurrently active users, according to www.slyck.com. Other popular file-sharing communities such eDonkey 2000, Warez, and Overnet remain largely unstudied.

Intercept Studies which intercept peer-to-peer traffic suffer from two fundamental limitations. First, because they look at only a cross-section of network traffic, usage patterns may differ compared to other user populations. For example, [GDS+03] and [SGD+02] both examine peer-to-peer traffic at the University of Washington. Because these users have exceptionally high bandwidth and an uneven age distribution, they may have different usage characteristics than, for example, a typical home broadband user. [LBBSS02] and [LRW03] provide measurements from an Israeli Internet Service Provider (ISP). Interestingly, the University of Washington users mostly upload content while the Israeli users mostly download it. [SW04] overcomes this limitation by capturing data at several routers in a Tier-1 ISP.

Second, interception only provides information about peers which are actively sending or receiving data. It cannot reveal any information about peers which are up but idle, and it is not possible to tell with certainty when the user has opened or closed the application. These caveats aside, these studies are quite useful in providing insight into file sharing usage patterns.

This technique is predominantly used to study bulk data movement such as HTTP-like file transfers and streaming, where it is relatively easy to identify a flow when it starts and just count the bytes.

Participate Several studies have instrumented open-source Gnutella clients to log information to disk for later analysis. While the studies in the previous section focused on file transfers, these studies focus on the overlay construction and keyword search features. These studies use a small number of vantage points into the network, relying on the knowledge that Gnutella does not construct its overlay in a topologically-aware fashion. Queries pass through these monitoring peers where observations are made.

Crawl A crawler is a program which walks a peer-to-peer network, asking each node for a list of its neighbors, similar to the way a web-spider operates. This is the only technique for capturing a full snapshot of the topology, which is important for simulation. However, capturing the whole topology is tricky, particularly for modern, large networks with millions of peers. If the crawler is too slow, it captures a distorted picture of the topology.

Probe With probing, a miniature crawl of the network is performed to collect a set of peers, then the peers are probed at regular intervals for several hours or days. This technique has the advantage of being straightforward. The primary problem with this approach is ensuring that the chosen peers are selected uniformly at random. Unfortunately, none of the papers address this critical issue. It is likely that their samples are inheritly biased towards peers with high uptime, since they are more likely to be around during the initial crawl.

Centralize The final measurement technique is to use the logs from a centralized source. For obvious reasons, this technique is of limited usefulness in peer-to-peer systems. The only case where this technique has proven useful is BitTorrent, which uses a centralized rendezvous point called a *tracker*.

3 Measurement Results

In the following subsections, we will examine what is known about peer-to-peer systems in the real world. We divide the different measurable aspects of peer-to-peer systems into six broad categories:

Churn affects the performance of all peer-to-peer systems, as they need to adjust as peers come and go. Measurements of churn examine the distribution of session lifetimes, downtimes, peer availability, and related characteristics.

File Characteristics are an important element of file sharing and streaming systems. They describe how the popularity of objects is distributed, how large the objects are, how many files each peer shares, etc.

Peer Characteristics include qualities such as the distribution of bandwidth and latency.

Query Characteristics describe the nature of the queries that users enter. This includes the relative popularity of queries, the frequency at which users enter queries, and how common duplicate queries are.

Topology is the way that peers are connected to one another. From this raw data, analysis leads to graph properties such as the node degree distribution, path lengths, and the clustering coefficient.

Implementation Characteristics are ways in which different vendors have chosen to implement the same protocol. Studying these in the real world allows for a true comparison between different algorithms on real data.

3.1 Churn

Churn is composed of several related concepts. *Session duration* or *uptime* is the length of time a peer is connected to a peer-to-peer network. *Lifetime* is the duration from the first time a peer connects to a peer-to-peer network—ever—to the very last time it disconnects. *Availability* is the percentage of time that a peer and its resources are connected to the peer-to-peer network; in other words, availability is the sum of all of a peer's uptime divided by the peer's lifetimes. *Downtime* is the duration between two successive sessions.

Prior work [RGK04] has summarized measurements of session length; their table is adapted as Table 3. The results for the median session time are conflicting, ranging from as low as one minute to as long as one hour. This may be due to genuine differences in user behavior, or it may be due to problems with some or all of the studies. All of these studies rely on probing, which can be biased, or interception, which misses quiet peers.

Citation	Systems Observed	Session Time
[SGG02]	Gnutella, Napster	50% $\leq$ 60 min.
[CLL02]	Gnutella, Napster	31% $\leq$ 10 min.
[SW04]	Kazaa	50% $\leq$ 1 min.
[BSV03]	Overnet	50% $\leq$ 60 min.
[GDS+03]	Kazaa	50% $\leq$ 2.4 min.

Table 3: Observed session lengths in various peer-to-peer file sharing systems. Adapted from [RGK04].

The median, however, does not tell the full story. Examination of the distribution of session lengths reveals that while many sessions are short, some sessions are very long. One study [CLL02] found around 20% of sessions last longer than two hours, and fit session lengths to a log-quadratic distribution.

Availability Another angle to churn is examining the *availability* of peers: what percentage of the time a particular peer is up, regardless of how long those periods are. [BSV03] examines availability in the Overnet peer-to-peer system[2], using data from January 2003. Peers in Overnet are assigned unique identifiers, so they

[2]Overnet is a second-generation system made by the creators of eDonkey 2000.

measure the presence of *particular nodes*, rather than a *particular IP address*. In their measurements, they find that the ratio of host identifiers to IP addresses is roughly 1:4, suggesting that IP-address based measurements may dramatically overcount the total number of hosts in the system over time. Almost 40% of probed hosts use more than one IP address in a single day. The availability of hosts is heavily dependant on the time of day, and therefore cannot be modeled as independent random events. However, when they examine the interdependence between hosts, they find that there is little direct correlation between the availability of one host and another. The availability itself proved difficult to measure and the results vary depending on the size of the measurement window, due to the significant fraction of hosts with very low availability that can only be observed in a large window.

BitTorrent As a somewhat different application, BitTorrent is worth examining separately. Users start the BitTorrent application to download a particular file, rather than to search and perhaps download several files. Also, BitTorrent is typically used for very large files that may take hours to complete even with a high-speed connection. [IUKB+04] examines the five-month tracker log for the RedHat 9 ISO images. 81% of peers leave before the download completes, with 60% of those peers remaining for less than 1000 seconds (16.7 minutes). However, peers that complete the download linger for six and a half hours, on average. These peers are known as *seeds*, while those still downloading are known as *leeches*. The seeds contribute more than twice the amount of data sent by leeches, with the portion of seeds was regularly more than 20% of all active peers, peaking at 40% during the first 5 days. This suggests that the performance of BitTorrent may rely heavily on peers remaining in the system long after their downloads complete.

Streaming Examining a Brazilian "reality television" show, [VAM+02] find that client inter-arrivals follow a Pareto distribution. However, further analysis shows that it can also be modeled as a series of Poisson processes which shift over time, which matches the Poisson distribution observed by other researchers [CWVL01, hCGR+04]. Session lengths fit a log-normal distribution and downtimes fit an exponential distribution. This is significantly different from the distribution of heavy-tailed session-lengths of file-sharing applications.

However, in a study of Akamai between October 2003 and January 2004, [SMZ04] show that session durations are heavy-tailed, and using interception [CWVL01] shows that more than 90% of streams have a total length less than 10 minutes, but the remaining streams have lengths ranging from 10 minutes to 6 weeks. This suggests that session durations may vary depending on the nature of the content. Moreover, since a television show has a relatively short duration, it isn't possible for the session durations to be heavy-tailed.

Surges in popularity are common, and virtually all short-duration streams experienced a burst of traffic on start-up [CWVL01].

Client turnover is very high. Roughly half of new clients to a stream are "one-timers" [SMZ04]. They tune in once, leave soon thereafter, and never access the stream again. However, many of the other clients are long-lived. This reflects user's viewing habits; they tune in to see if they like a program, and either decide to continue viewing it, or they leave.

Open Issues There are noticeable discrepancies in the measurements of session durations, and the implications of very short sessions on network stability are not well understood. The downtime distribution and any correlations between uptime, downtime, and future uptime also have not been studied.

3.2 File Characteristics

File characteristics in file-sharing systems can be examined from two different angles: files *stored* and files *transferred*. Although these two things are related, they have slightly different properties.

Files Stored The distribution of stored files in file-sharing systems is heavily skewed, following a Zipf distribution [FHKM04]. In fact, the 10% most popular files make up 50% of all stored bytes! [CLL02] The most popular files are around 4 megabytes in size; however, the 3% of files which are videos make up 21% of all stored bytes.

Users have noticeable interests, with 30% of files having a correlation of at least 60% with at least one other file. When two peers have 10 files in common, there's an 80% chance they have at least one more file in common. The probability is close to 100% if they have at least 50 files in common [FHKM04]. In fact, generating a graph

by treating users as nodes and assigning an edge when two users have more than N files in common, results in a small world [LRW03].

The number of stored files per user is also heavily skewed, with two studies finding around 67% of all users sharing no files at all [AH00, FHKM04]. However, [SGG02] found this number to be much smaller, at just 25%, though their results agree that most users share few files, and that a relatively small fraction of users share the majority of files.

Files Transferred The characteristics of files being transferred are even more lopsided. Comparing Kazaa traffic characteristics with web traffic, Kazaa has a request rate two orders of magnitude lower, but objects which are three orders of magnitude larger. In one study [SGD+02], peer-to-peer traffic accounts for three-quarters of all HTTP data, and the number of simultaneously open Kazaa connections is twice that of the Web. Likewise, Kazaa accounts for more than twice as much traffic overall compared to the Web.

More than 90% of file transfers are of small files (less than 10 MB), while most bytes are part of large files (larger than 100 MB). File sizes are also unevenly distributed. They tend to be around 100 KB (pictures), 2 MB to 5 MB (music), or larger than 700 MB (movies, games) [GDS+03]. These long transfers are significantly different from the Web which is inheritly interactive.

The most popular 10% of transferred files account for 60% of all transfers, and the most popular 5% account for 50% of all transfers. That's approximately 45,000 songs which can be stored in 175 GB [GDS+03], suggesting that caches would be extremely effective for file-sharing applications. Several studies [SGD+02, LBBSS02, LRW03] support this conclusion with observations that users frequently download files from distant parts of the Internet that have already been downloaded by a nearby user. [GDS+03] shows that 86% of bytes were already available locally. [LBBSS02] takes this a step further and implements a transparent cache, achieving a 67% savings using a 300 GB disk drive.

[GDS+03] makes the interesting observation that preferences for files, measured by transfers, do not follow a Zipf distribution. Instead, popular files are of roughly equal popularity, while unpopular files are Zipf. This contrasts with both the Web and the files stored on each peer. They explore this further and attribute this difference to the fact that Web users will access the same object repeatedly, while Kazaa users typically download an object at most once. They validate this hypothesis by showing that if repeated access from the same user are removed from web data (such as if a cache is in use), then the Web access distribution has approximately the same shape as the Kazaa distribution. However, this does not explain why stored files, which are a product of transfers, follow a Zipf distribution. Perhaps this is a result of some files being popular for a long time, while others dwindle.

Roughly 30% of transfers are recurrently popular over the course of a few days, while many other files experience brief popularity, then fade [LRW03]. Over the course of a month, 15% of transfers remain steadily popular.

Streaming [VAM+02] examine the workload imposed by a Brazilian "reality television" show from a 28-day period in early 2002 and including 1.5 million sessions and 8 terabytes of data. They were able to keep track of individual users via a unique identifier stored in each client. This allowed them to monitor how often users tune in to the program, showing that user interest follows a Zipf distribution. In other words, there is a small number of users who are very interested in the program, and a large number of users who are slightly or uninterested.

More recently, [SMZ04] presents data gathered at Akamai between October 2003 and January 2004. This data includes 70 million requests for 5,000 distinct URLs. They find that popularity follows a 2-mode Zipf distribution with the most popular items relatively similar in popularity. Interestingly, this matches the result in [Sri01] for file-sharing items. This could be due to a similar only-view-once effect. However, this result conflicts with an earlier study [CWVL01], which found a pure Zipf distribution.

Most session have short durations, less than 10 minutes, and modest sizes, less than one megabyte. However, a few sessions (3%) are larger and longer and are responsible for nearly half of all bytes downloaded [CWVL01].

Open Issues The shift in popularity of files is not entirely understood, requiring observation over longer periods of time. This data may be useful beyond network research, and, indeed, the publishing industry or researchers studying the evolution of culture may already have an understanding of how these trends evolve in other contexts, such as album sales.

It would be interesting to further examine the correlations in taste in conjunction with shifts in popularity, to see if it is possible to *predict* whether a user will be interested in a particular object. Speculative caching would then be possible, or perhaps a decentralized recommendation system.

Finally, no existing measurement studies examine the swarming download feature found in many modern file sharing clients. We do not know how well they work or where their bottlenecks might be. We do not even have well-understood metrics for understanding their behavior. As file-sharing networks continue to proliferate and swarming downloads become more common, this is likely to become a significant area of future research.

3.3 Peer Characteristics

Another case of heavy skew is the fact that some IP addresses account for a dramatically disproportionate amount of traffic. In [SW04], less than 10% all IP addresses running Kazaa contributed 99% of the total traffic. The top 0.1% contributed 33% of the traffic, while the top 1% contributed 73%.

[SGG02] present one of the most comprehensive studies of Gnutella and Napster, based on data collected in May of 2001. After collecting a pool of hosts, they measure the bottleneck bandwidth, latency, availability, and the number of shared files for each host. Each of these qualities varies by many orders of magnitude.

Correlations exit between some of the metrics they measure. Bottleneck bandwidth and the number of uploads have a positive correlation, while bottleneck bandwidth and the number of downloads have a negative correlation. In other word, those with high bandwidth tend to be uploading many files, while those with low bandwidth have to spend more time downloading. Interestingly, no significant correlation exists between bottleneck bandwidth and the number of files stored on a peer.

Streaming Bandwidth varies considerably among peers [hCGR+04], and therefore leveraging heterogeneity is critical. An early study [CWVL01] found that most streaming is at low bit-rates: 81% are transmitted at less than 56 kbps. However, may have shifted in the last few years as bandwidth becomes steadily less expensive.

One study examines a basic single-tree overlay multicast system in real-world usage, with a peak of 280 simultaneous viewers , [hCGR+04]. This work highlights several practical difficulties not normally addressed in overlay papers. Specifically, it addresses the difficulty in coping with Network Address Translation (NAT) and firewalls. These devices can establish outgoing connection, but cannot accept incoming connections. This limits the connectivity of the network because two such devices cannot peer with one another.

Open Issues While [SGG02] presents us with an impressive battery of information on peer bandwidth characteristics, this attribute is perhaps the one most likely to change over time. Also, their measurement techniques are somewhat limited compared to the ideal of all-pairs bandwidth and delay characteristics needed for simulation. This is a tricky problem.

The distribution of NAT and firewall devices in the network is a serious problem limiting the connectivity of peer-to-peer applications and largely ignored by researchers. An in-depth study of the types of NAT and their frequency would be invaluable for designing peer-to-peer systems that can cope with connectivity limitations.

3.4 Query Characteristics

Queries follow a Zipf distribution, except for the most popular queries which are of roughly equal popularity [Sri01]. However, while both queries and stored files follow a Zipf distribution, they don't match. There is little relationship between sharing many files and responding to many queries. In other words, some users have a large collection of rarely-sought files while other users have small collections of highly popular items [AH00].

An impressive 40% of all queries are duplicates, using different query identifiers[3] [Mar02]. These results have led several researches to suggest caching query results [Sri01, Mar02].

Queries and query-hits appear in bursts, across a variety of timescales [Mar02]. One study, [KLVW04], provides a comprehensive analysis of queries, breaking down the number of queries observed by time of day and geographical region. It includes distributions for the number of sessions that generate queries, the time until the first queries, the query inter-arrival time, and the length of the session. In short, it provides a framework for generating a synthetic query workload as seen from a single peer.

[3]Query identifiers are used by the flooding protocol to prune duplicate queries.

An interesting quirk in query propagation in flooding networks is the "short-circuit effect", first pointed out by [ABJ01]. In simulation and analysis of scoped flooding networks such as Gnutella, it is normally assumed that all hops take equal time to traverse. A message spreads out through the network, and any duplicates along longer paths are dropped. However, due to unequal latencies, it is possible for the message along the longer overlay path to arrive first. When this occurs, the horizon of the network is shortened, since this message has a lower TTL than in the ideal model. Using a small captured topology with randomly assigned network latencies, they show that the short-circuiting effect is significant. For a TTL of 5, it reduces the search horizon by 58% on average.

Open Issues The relationship between queries, files stored, query replies, and files transferred is not yet well understood. The fact that storing many files and responding to many queries are uncorrelated is surprising and suggests further research is needed.

Also, no existing work examines the relationship between the composition of a query and the number of *distinct* results returned. Put another way, all existing work considers the relationship between queries and results. However, queries are composed of a set of search *terms* and results must match all the terms. Little work has been done to examine how many terms are in each query, the relative popularity of individuals terms, and how different terms affect the final result set. This data would be import for investigating term-based indexing options, such as using a Distributed Hash Table as described in Section 4.1.

While several studies have recommended result-caching based on their measurements, no studies have been able to quantify how frequently the cached data will become invalid due to changes in the network.

3.5 Topology

In 2000, a company called "Clip2" developed a Gnutella crawler and published their results on the web. Although not validated by peer-review, their analysis and topology captures have been widely used in simulation studies of improvements for Gnutella-like networks [ALHP01, JZ03, LZXN03, LZXN04, LLX+04]. In [cli00], Clip2 presents analysis of data they captured between June and August of 2000. They counted between 1,000 and 8,000 active peers at this time using their crawler which took a bit less than an hour to survey the entire topology.

Their analysis suggests that the Gnutella network has a power-law degree distribution, although the highest degree is 20. [ALHP01] repeats this analysis on similar data provided by Clip2, with a maximum degree of 12. Several later studies [CGM02, GMS04, JZ03, LRS02, WXLZ04] rely on this result, simulating Gnutella using random power-law topologies. [LCC+02] show that power-law networks are in fact worse for unstructured search than random graphs.

[RFI02] implement a crawler and use it to examine properties of the overlay topology. Their crawler uses a client-server architecture running on roughly 50 computers to crawl a 30,000 node network in a few hours. Their crawls were conducted in November 2000 through May 2001. The size of the network grew from 2,063 to 48,195 peers over that time. They performed all-pairs shortest-path computations and plotted the distribution of path lengths. 95% of paths are within 7 hops, with most paths being 4 or 5 hops long. In their November capture, the degree distribution was power-law. By March, it is modal, with a roughly even distribution of degree among low-degree nodes and a power-law distribution among high-degree nodes. Finally, they show that the Gnutella topology is poorly matched to the underlying Internet topology.

Open Issues Thus far, no published papers have quantified the accuracy of their crawlers, making it difficult to determine how accurate the captured topologies are. Given that crawls may take a few hours, and peer uptimes may be just a few minutes [CLL02, SW04, GDS+03], it is very possible that these topologies are highly inaccurate, leading to a drastically distorted picture of the network.

Also, all of the captured topologies are dated, studying Gnutella when it was over an order of magnitude smaller than it is today, and before the introduction of ultrapeers. Finally, it would be interesting to capture the topology of other peer-to-peer file sharing networks to compare and contrast them.

3.6 Implementation Characteristics

[KAD+04] explores the way in which different Gnutella implementations bootstrap to connect to the network initially. They compare the operation of LimeWire 2.8, Gtk-Gnutella 0.91, Mutella 9.4.3, and Gnucleus 1.8.6.0. Each implementation employs a local cache of nodes discovered in previous sessions and resort to contacting special

rendezvous points, called GWebCaches [Dm03], if the local cache proves unhelpful. However, the details of each implementation differs substantially. LimeWire's performance is the best; it differentiates between ultrapeers and leaves in its cache, prioritizes the cache by age, and includes the hard-coded addresses of two orders of magnitude more GWebCaches than the other implementations. Not only does LimeWire connect faster, but it begins receiving query replies sooner once connected. Unfortunately, GWebCaches are not performing well; most of the load is concentrated on just a few and the caches have many entries for hosts that are no longer present in the system.

[HA04] brings to light the issue of congestion in the overlay. Each peer connection uses TCP, a reliable transport protocol with congestion control. When the network is congested, TCP will slow down its sending rate. However, this means that the application will need to slow how fast it's feeding data to TCP. While the application can buffer data to smooth out transient network conditions, it eventually needs to start dropping data if the queue is continuously growing faster than it can be emptied. [HA04] examines the approaches taken by three different Gnutella implementations: LimeWire, Mutella, and Gtk-Gnutella. Their study includes a comprehensive explanation of the algorithms employed by these implementations, measurements comparing their performance, and simulations exploring alternative algorithms. Again, LimeWire shows the best overall performance.

Open Issues These studies have left out BearShare, one of the most popular Gnutella implementations, since it is not open source. Nevertheless, these studies are very useful since they touch on topics otherwise neglected in the research literature. Overlay congestion control, in particular, is not well-understood, with little solid research to guide developers in choosing appropriate algorithms for handling application-layer bottlenecks.

4 Design of File-Sharing Networks

File sharing applications are composed of several inter-linked components. In the following subsections we review published approaches for improving *(i)* indexing, *(ii)* query routing, and *(iii)* overlay construction. We also explore attempts to model file sharing networks. We begin by examining the most radical departure from existing systems: attempting to index file-sharing networks using Distributed Hash Tables. Table 4 summarizes the papers covers in this section, illustrating where the research community has placed the most focus.

DHT	Indexing and Querying		Overlay Construction	Modeling	Miscellaneous
[APHS02] [CCR03] [LHSH04]	[ALHP01] [GMS04] [LCC+02] [LRS02] [YVGM04] [WXLZ04]	[CS02] [YGM02] [YGM03] [LLS04] [CGM02] [SMZ03]	[LZXN03] [LZXN04] [LLX+04] [WC04] [PRU01]	[GFJ+03] [ZA03] [YGM01]	[CRB03] [SGM04] [BFLZ03] [ZZA02]

Table 4: Research on peer-to-peer keyword search

4.1 Indexing with a Distributed Hash Table

A few papers suggest using Distributed Hash Tables (DHTs) [SMK+01, RFH+01, RD01] for the search function in peer-to-peer file sharing systems. In these systems, peers still store whichever files they please, and the DHT is used as a distributed indexing system.

[APHS02] discusses a DHT called P-Grid and builds a Gnutella-like system called Gridella over it. In their system, peers get a portion of the search space based on their connections to other peers. Then, they fall-back to Gnutella-like flooding to locate files that belong in their search space. They index those to make lookups more efficient for other nodes. However, this early paper does not address any of the typical problems that DHTs face: handling high peer churn and balancing load when some terms are exponentially more popular than others.

[CCR03] takes a different approach with a system called "Structella", which builds the overlay using Pastry [RD01]. Then, queries are sent over the structured overlay using flooding or random walks. They use the structure

of the network to ensure that nodes do not receive any duplicate queries, thereby cutting cost considerably compared to flooding over a small-world network.

[LHSH04] observes that a hybrid system may be the best approach. In their design, queries are done using conventional flooding with a low TTL. Searches for popular items will be satisfied easily with this small, cheap flood. If not many results are found, then the lookup is done using a DHT which *only* indexes rare items. This ensures that no peers will shoulder a grossly disproportionate load.

The tricky part is determining which items are rare. They conduct a few measurements using the Participate approach, and show that small result sets tend to contain only rare items, while large result sets contain a mix of common and rare items. Thus, they suggest using the heuristic that items returned in small result sets are rare and are indexed in the DHT. Using measurement, they show that 18% of queries return no results, even though for two-thirds of those queries a match does exist in the network. Thus, their approach would both increase results and decrease load.

Open Issues Much work remains to be done in this area. Although several steps have been made in the right direction, a comprehensive, scalable, and robust system for indexing files using a DHT has not yet been presented. Nevertheless, one widely-deployed peer-to-peer system, Overnet, uses the Kademlia DHT [MM02], presenting a unique opportunity for future measurement studies.

4.2 Indexing and Query Routing

Aside from DHTs, two basic strategies have been proposed for improving the efficiency and efficacy of search versus basic flooding. One is to reduce the number of duplicate messages by walking the network instead of flooding it. The other is to replicate indexing information so queries search more with each peer they pass through.

[ALHP01] was the first to propose a walk strategy. In their scheme, each node indexes itself plus all of its neighbors. Queries walk through the network, one peer at a time, preferring peers with the highest degree first. They show with simulation that, in a power-law graph, this results in searching most of the nodes very quickly. The query will reach the high-degree nodes within a few hops and, since they have many neighbors, these nodes index a significant fraction of the network. However, their scheme results in *every query* being routed through the highest degree nodes. [LRS02] build on this idea by introducing a dynamic scheme to adjust node degree based on throughput. It still seems unlikely that network operators at high-bandwidth locations would approve of their users routing *all* the query traffic for a large file-sharing network.

[GMS04] takes a slightly different approach and suggest using *random* walks. Through analysis and simulation, they show that random walks perform better than flooding when either *(i)* ultrapeers are used, or *(ii)* identical searches are reissued periodically when the overlay has not changed significantly.

[CS02] examines different approaches to replicating index information in the overlay. They show that having records proportional to the popularity of the file is suboptimal. Proportional indexing is done implicitly by most file-sharing networks today since each instance of a popular file creates an index entry on that peer. Interestingly, they also show that having an equal number of records per unique file performs just as poorly! For proportional indexing, there is too little indexing of the considerable number of less popular items. With uniform indexing, too many resources are being spent indexing the least popular items. The optimal solution is *square-root replication*, in which the number of index entries is proportional to the square-root of the popularity of related queries, striking the right balance between common and uncommon files.

One difficulty with walking is that it incur substantially more latency than flooding. [LCC+02] solves this problem by using k walkers in parallel. Using simulation, they show that 16 to 64 walkers typically achieves good results, with latencies comparable to flooding. With 32 walkers, the overhead is reduced by two-orders of magnitude compared to flooding. They also discuss practical ways to implement the square-root replication proposed in [CS02].

Another challenge with walking is the danger that a long-walk may be lost somewhere in the overlay, greatly limiting the search horizon. The Gnutella UDP Extension for Scalable Searches protocol (GUESS) [DF02] addresses this issue by keeping the state at the searcher. It individually queries each node along the walk, retrieving a list of other GUESS-enabled ultrapeers in addition to results from each node queried.

[YVGM04] evaluates GUESS and explores different policies for caching the list of ultrapeers to query. They show that prioritizing the caching of nodes with many files improves performance. However, congestion control

for GUESS is still an open issue. Because query sessions are single, brief transactions, remaining TCP-friendly is tricky.

Building on the suggestions of [Sri01] and [Mar02], [WXLZ04] explores a query-caching system for Gnutella, called DiCAS. In this scheme, each node is assigned a random n-bit label. Queries are hashed into n-bits and query results are cached by peers with a matching label. Thus, in a 2-bit system, each peer would cache roughly one-quarter of all query results. Their simulations show a moderate cache size reduces query traffic by 54% and response time by 33%. However, they do not address the problem of how to expire stale cache data.

Finally, [LLS04], [CGM02], and [SMZ03] suggest optimizing queries based on common interests. [LLS04] and [CGM02] rely on using additional semantic knowledge about the content to restructure the network. [SMZ03] proposes adding temporary "short-cut" links to nodes the previously responded to one of the user's queries. Later queries are then sent to the short-cut peers first, then flooded conventionally if the short-cut peers produce enough matches. Each peer maintains just a handful of short-cuts. Through trace-driven simulation, they show that shortcuts can reduce load by a factor of 3 to 7.

Open issues In summary, we have several techniques that offer significant improvements over flooding. Square-root replication coupled with k-random walks over a network-aware overlay is a promising set of optimizations. However, these approaches have not been carefully examined using populations as large as today's file sharing networks. Also, square-root replication and other result-caching schemes may not fair well in light of short peer lifetimes where the cached data will become invalid quickly. Moreover, none of these techniques are efficient for queries that don't match any files in the system; for those queries, a hybrid solution using a DHT may be the only efficient solution.

Also, while significant work has been done in this area, Gnutella clients have headed off in a slightly different direction. Future work will need to reconcile this difference by studying the deployed implementations and see how well they compare with the techniques already proposed in the literature.

4.3 Overlay Construction

Several studies examine building more efficient overlay topologies. The majority of these techniques are aimed at overcoming the mismatch noted in [RFI02] between the overlay and underlying topology.

[LLX+04] propose a system called Location-aware Topology Matching (LTM) which selectively rewires the topology based on network locality. In an impressive set of simulations, they show this results in faster searches and less traffic while maintaining good connectivity.

[LZXN03] and [LZXN04] take a slightly different and complementary approach by logically disabling nearby redundant overlay links. In their system, neighbors keep track of their neighbors' neighbors and avoid sending queries through obviously redundant links.

[WC04] develops a system called Phenix, which builds a power-law overlay in a distributed way that does not reveal which nodes are the high-degree nodes. Their goal is to construct the power-law overlay while making it difficult for attackers to cripple the network by attacking only the high-degree nodes.

[YGM03] examines topologies that use super-peers. Their most interesting result is demonstrating that high-degree, low-TTL networks are both more efficient and more effective than low-degree, high-TTL networks.

Open Issues In the absence of a solid understanding of churn and how it affects the topology, it's difficult to understand the implications of additional rewiring or deliberate attempts to alter the topology. Causing additional turbulence in the overlay may be detrimental. Once churn and topology are better understood, overly construction techniques will need to be revisited.

4.4 Modeling

Only a few attempts have been made to develop comprehensive models to describe peer-to-peer file sharing systems. The earliest attempt, [YGM01], models Napster-like systems that feature a centralized index and peer-to-peer downloads. However, due to legal issues, no systems in this category exist anymore.

[GFJ+03] presents a model for the search component of peer-to-peer systems, including systems based on centralized indexing, flooded queries, and DHTs. Their model supports several classes of users, distinguishing freeloaders versus contributors and high versus low bandwidth users. Interestingly, their results show that

freeloaders frequently do not adversely affect the performance of contributors. In their model, peers remain in the system long enough to download n files, where $n > 1$. Unfortunately, this may not be the case, particularly in light of the low session times reported in [CLL02]. They also assume that peers remain only orders of magnitude longer than it takes to setup all of their connections, indexing, etc. This, too, may not be the case.

In contrast, [ZA03] models the spread of a file in a peer-to-peer system. Each peer may be up or down, and may have the file, uninterested in the file, downloading the file, or waiting to download the file. They break these into distinct states, and simply track the number of peers in each state rather than keeping track of each peer individually. This makes their system faster and memory efficient.

Open Issues The models developed thus far are useful, but have limited applicability. Of course, this always the tradeoff with models: it's easier to work with a simple model, but it makes more simplifying assumptions. Also, because peer-to-peer systems are evolving so rapidly, models can quickly become outdated. For example, the model of [ZA03] does not account for swarming downloads.

4.5 Miscellaneous

The problem of server selection has been extensively studied in the literature. However, the problem is more complicated in peer-to-peer systems. The server selection problem is typically cast as a *local* optimization problem, where the goal is to find the optimal measurement technique and heuristics to allow a client to select the best server. In the peer-to-peer context, this model is not always sufficient. Peers are typically more bandwidth-constrained than servers, so there is greater contention for resources and server-peers allow only a limited number of concurrent uploads. Furthermore, because some peers will be more useful in propagating a file throughout the system, there is a *global optimization* element to the peer selection problem. In addition to the client-component selecting peers, the server-component must choose which peers to upload to.

[ZZA02] presents an initial exploration of this problem. They assume the existence of a search protocol which returns a list of peers that can serve the content. Through simulation, they consider the simple case where peers allow any number of uploads and have global knowledge. In this scenario, they found that the greedy approach worked best: each client-peer selects the server-peer with the maximum value of $\frac{b}{n+1}$ where b is the server-peers uplink bandwidth, and n is the number of existing upload sessions. They also provide an initial exploration of the case where server-peer's have a finite number of upload slots.

Another largely unexamined problem is overlay congestion. The only design paper dedicated to this topic is [SGM04], which proposes allocating bandwidth to peers based on the number of query replies seen from that peer. They examine the effectiveness of their system through simulation on a 250 node random graph, using fixed available bandwidth at each peer. Unfortunately, they do not compare their system with algorithms in existing Gnutella implementations.

[CRB03] sets the ambitious goal of designing a next-generation Gnutella system called Gia. This system used one-hop index replication coupled with a walk biased towards higher-bandwidth nodes. They compare Gia through simulation with a flooding protocol, the k-random-walkers protocol from [LCC+02], and a super-peer protocol.

Gia also employs a unique overlay flow-control mechanism based on tokens. Each node assigns tokens to its neighbors, distributing the tokens proportional to the neighbor's advertised capacity. The tokens act as a right-to-transmit for one query. Because they do not evaluate their flow-control mechanism separately, it is difficult to compare with the one from [SGM04] or those in real Gnutella implementations.

5 Design of Swarming-like Systems

Few papers have proposed swarming download schemes. [KG99], [SRS02], and [PS02] propose using a peer-to-peer approach to alleviate flash crowds by re-sharing files that have been downloaded in full. In [KG99] and [PS02], peers access a central server and may be referred to a peer that has already downloaded the file. In [SRS02], peers search an overlay for a peer if the central server is overwhelmed. [BLM99] proposes allowing clients to download from several Web servers at once. This allows conventional server-selection techniques to be skipped, sine the clients will implicitly download most of the file from the fastest servers. To avoid the complexities of choosing which bits to download from which server, they employ a type of *Forward Error Correction* (FEC) called Tornado

codes. FEC encodes k packets into n packets such that the original k packets can be reconstructed using any ϵk packets of the n, where $n > k$ and ϵ is close to, but greater than, 1.

The workshop paper [MM03] proposes a full swarming system using FEC *rateless codes*. They use FEC to improve the probability that peers will have data useful to other peers. For rateless codes, n is "practically infinite" and coding is computationally inexpensive: encoding each block is $O(1)$ and decoding the whole file is $O(\epsilon k)$. However, no formal analysis has been performed to demonstrate that peers have difficulty finding non-redundant data sources in non-FEC swarming systems, so it's unclear what benefit the additional complexity provides. They do not provide any evaluation of the performance of their system.

Slurpie is another swarming system with the unique design goal of keeping the load on the root server constant, regardless of the number of Slurpie clients [SBB04]. To achieve this goal, Slurpie clients attempt to estimate the total number of clients, n, in the swarm. Then, if a peer can't find a source for the block it's seeking, every 4 seconds it downloads the piece from the root server with probability $\frac{4}{n}$. On average, there will be fewer than 3 concurrent connections to the server, yet the there will be at least one connection to the server 90% of the time.

Slurpie clients decide uniformly at random which block they would like to download next, then look for a peer that can provide them with that block. This *block-seeking* approach is quite different from BitTorrent, which establishes its peers first and then downloads whichever blocks they have available.

Slurpie performs better than BitTorrent in some limited experiments, which show that Slurpie performs better when 50 clients on a 100 Mbps switch are downloading a 1 GB file from a source on a 10 Mbps connection. Unfortunately, this doesn't provide much information about how Slurpie would perform in a true flash-crowd scenario. Also, their experiments do not help us to isolate which aspects of their implementation allow Slurpie to outperform BitTorrent in that setting.

Several models for BitTorrent exist [QS04, YdV04]. They show that after a period of exponential growth in system capacity, the system enters steady-state behavior. They also show re-sharing before completing the download dramatically improves flash crowd response for large files. However, in their model BitTorrent responds slowly to an additional burst of demand after the system is already in steady-state, suggesting an avenue for improvement.

Bullet is a swarming system [KRAV03], inspired by overlay multicast. It forms a tree from the content source and uses this tree to propagate random samples about which peers have which blocks of the file. These gossiped messages use Bloom filters to summarize information about which blocks a peer has. Peers download blocks from their parent in the tree, as well as contact other peers in the system that have blocks they need. Peers allocate bandwidth to their children based on how many grandchildren they possess, and try to send a different block to each child. They evaluate their scheme using the ModelNet emulation environment and on PlanetLab. They show that Bullet performs better than a traditional single-tree overlay multicast scheme; however, this does little to improve our understanding of how it compares to other swarming techniques.

Open Issues In summary, the design space of swarming protocols is still largely unexplored with only a few research papers on the topic. Part of the difficulty may be that we do not yet have a good set of realistic, yet practical, scenarios for comparing implementations, nor insightful metrics for locating bottlenecks. As such, it's difficult to make useful comparisons between two swarming systems, except in extremely limited environments.

Swarming systems are complicated with many inter-related components. Each implementation needs algorithms to select peers, to choose which blocks to retrieve from which peers, to choose how many peers to connect to, and a way to find peers. Each of these components affects the behavior of the others, which makes studying them in isolation difficult. Also, we do not yet have a solid understanding of swarming systems that are deployed and in operation in the real world, which makes it difficult to improve them incrementally.

There are two basic approaches. We can design entirely new systems and compare them head-to-head, or gradually improve existing solutions. However, we lack good metrics and simulation environments to compare two systems head-to-head and we lack the knowledge and the metrics to understand where an existing system is bottlenecked. These issues will need to be resolved before further progress can be made.

6 Design of P2P Streaming Systems

Peer-to-peer streaming is an up-and-coming technology evolving from overlay[4] multicast [KLL03, BBK02, CRhZ00, JGJ+00, KB04, BLBS03, ZLG04]. While IP multicast was by nature limited to a single distribution tree, overlay multicast has considerably more flexibility. [BLB+04] shows that by sending a small percentage of packets outside of the regular tree, they can improve the overall delivery rate. Like swarming, peer-to-peer streaming is a peer-to-peer distribution mechanism where peers use multiple sources and re-share data as soon as it arrives. Unlike swarming, streaming attempts to deliver the data in-order so that the client application can playback the content in real-time. Where swarming splits files into blocks, peer-to-peer streaming splits the stream into stacked *layers*. The receiver needs the lowest layer to continue playback. Higher layers improve the quality of the playback.

CoopNet [PWC03] uses *Multiple Descriptor Coding* (MDC) [Goy01]. Unlike layered coding, *any* single sub-stream will provide low-quality playback with MDC, and any additional sub-streams will improve quality. Though trace-driven simulation, they explore the relative performance of different centralized tree-construction algorithms and the tradeoffs of how many trees to use. Their simulations show that CoopNet performs well, except in the case where peers leave abruptly and repairs take more than 5 seconds to complete. However, their simulations do not examine the case where peers have heterogeneous or changing bandwidth characteristics.

One study uses the data from [SMZ04], coupled with heuristics to estimate client bandwidth characteristics, to evaluate the feasibility of using a generic peer-to-peer streaming protocol [SGMZ04]. For a single-tree protocol, they simulate a few different tree-construction policies and find that a minimum-depth join algorithm resulted in the fewest interruptions. They try using uptime-so-far as a predictor of future uptime, and joining to parents who were the most likely to remain up. However, this performs poorly. It results in very tall trees such that when a node high in the tree finally does fail, the number of interruptions is very significant. They also find that using multiple trees with MDC results in fewer interruptions.

PRO uses a decentralized, gossip-based mechanism for locating potential parents [RS04] suggests. Each peer maintains a modest cache of other peers in the system, along with their network coordinates as determined by a system such as GNP [NZ02], HYP [ST04], or Vivaldi [DCKM04]. Peers periodically exchange gossip messages with other peers in their local image, transferring the records most likely to be useful to the other peer. In this way, peers gradually learn about the peers closest to them and select a subset of them as sources. [RO03] complements this scheme with a receiver-driven protocol for downloading packets from a set of senders. In other words, this is a "pull" scheme whereas the traditional multicast schemes are "push".

Most of these schemes rely on an underlying congestion-control algorithm. TCP is inappropriate because it *(i)* enforces reliability which is undesirable in an application where old data is useless, and *(ii)* may experience large oscillations in throughput. RAP [RHE99] and TFRC [FHPW00] are rate-based protocols which provide TCP congestion-fairness while providing more stable throughput.

Open Issues While efficient streaming over the Internet has been a dream of researchers for many years, neither IP nor overlay multicast have been successful in realizing it. Peer-to-peer streaming is a new approach that promises to overcome the remaining barriers by leveraging some of the same techniques that make other peer-to-peer applications so successful. However, the real-time component remains a significant technical barrier which new projects like CoopNet and PRO attempt to overcome. PRO is still in the early stages of its development, and will be the first decentralized, peer-oriented streaming protocol.

7 Summary and Conclusions

This paper surveys this research literature in three related areas, examining different proposes approaches and measurements made of real systems:

1. Peer-to-Peer Search
2. Peer-to-Peer Transfer
3. Peer-to-Peer Streaming

[4] Also called "application-layer multicast"

We find that some aspects of these are moderately well understood, while others have gone mostly untouched. The most well-examined aspects include the following:

Index and query routing over an unstructured overlay Square-root replication coupled with k-random walks over a network-aware overlay is a promising set of optimizations that greatly improve efficiency relative to flooding.

Query, download, and file distributions in file-sharing networks The popularity of files follows a pure Zipf distribution, while the popularity of queries and transfers is Zipf for less-popular files, but relatively equal for the most popular items. Users tend to have distinct preferences and generating a graph based on the files results in a small-world.

User viewing habits of streams User preferences for streams also follow a Zipf distribution for unpopular items and a relatively equal distribution for the most popular items. Churn among streaming viewers is relatively well understood, due to the availability of centralized logs.

The least understood aspects include the topics listed below. My existing and future research will focus on a subset of these areas, and may revisit some of the well-understood topics in light of new information.

Topological characteristics of unstructured networks The few topological studies are based on crawls that are likely in accurate. Moreover, these crawls are old and the Gnutella network has grown by one or two orders of magnitude since then and introduced ultrapeers. No topology studies have been done of other peer-to-peer networks.

Effects of churn on overlay stability, performance, and structure There are conflicting results on churn in file-sharing networks, and further work needs to be done to establish a definitive analysis. Thus far, no work has been done to examine the effects of churn on the overlay.

Tools for workload generation Even in instances where distributions are well understood, they are often only presented as graphs and not fit precisely so that they can be used to generate workloads. As a result, papers use different simulation environments to evaluate their designs.

Effectiveness of existing swarming solutions Swarming downloads are still poorly understood. Substantial anecdotal evidence suggests BitTorrent performs substantially better than a single server, but we don't know how much better or where its bottlenecks lie. No studies have examined the swarming implementations built-in to file-sharing applications.

How to construct a good swarming system Few studies have proposed alternative swarming solutions and we have a relatively weak understanding of different ways to approach the problem. once good metrics and simulation models have been developed, it will become easier to compare and contrast different solutions.

How to construct a good peer-to-peer streaming system Few studies have proposed alternatives for peer-to-peer streaming. This, too, is a relatively open field, although the performance metrics are at least better understood.

References

[ABJ01] F. S. Annexstein, K. A. Berman, and M. A. Jovanovic. Latency effects on reachability in large-scale peer-to-peer networks. In *Symposium on Parallel Algorithms and Architectures*, pages 84–92, Crete, Greece, 2001.

[AH00] E. Adar and B. A. Huberman. Free riding on gnutella. *First Monday*, 5(10), October 2000.

[ALHP01] Lada A. Adamic, Rajan M. Lukose, Bernardo Huberman, and Amit R. Puniyani. Search in power-law networks. *Physical Review E*, 64(46135), 2001.

[APHS02] K. Aberer, M. Punceva, M. Hauswirth, and R. Schmidt. Improving data access in P2P systems. *IEEE Internet Computing*, 6(1):58–67, Jan/Feb 2002.

[BBK02] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *SIGCOMM*, August 2002.

[BFLZ03] Daniel Bernstein, Zhengzhu Feng, Brian Levine, and Shlomo Zilberstein. Adaptive Peer Selection. In *International Workshop on Peer-to-Peer Systems*, 2003.

[BLB+04] Suman Banerjee, Seungjoon Lee, Ryan Braud, Samrat Bhattacharjee, and Aravind Srinivasan. Scalable Resilient Media Streaming. In *Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2004.

[BLBS03] Suman Banerjee, Seungjoon Lee, Bobby Bhattacharjee, and Aravind Srinivasan. Resilient multicast using overlays. In *SIGMETRICS*, pages 102–113, San Diego, CA, June 2003.

[BLM99] John W. Byers, Michael Luby, and Michael Mitzenmacher. Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads. In *INFOCOM*, pages 275–283, New York, NY, March 1999.

[BSV03] R. Bhagwan, S. Savage, and G. Voelker. Understanding availability. In *International Workshop on Peer-to-Peer Systems*, 2003.

[CCR03] Miguel Castro, Manuel Costa, and Antony Rowstron. Should we build Gnutella on a structured overlay? In *Hotnets-II*, Cambridge, MA, November 2003.

[CGM02] Arturo Crespo and Hector Garcia-Molina. Routing indices for peer-to-peer systems. In *ICDCS*, 2002.

[cli00] clip2.com. Gnutella: To the Bandwidth Barrier and Beyond, November 2000.

[CLL02] Jacky Chu, Kevin Labonte, and Brian Neil Levine. Availability and Locality Measurements of Peer-to-Peer File Systems. In *ITCom: Scalability and Traffic Control in IP Networks II Conferences*, July 2002.

[CRB03] Yatin Chawathe, Sylvia Ratnasamy, and Lee Breslau. Making Gnutella-like P2P Systems Scalable. In *SIGCOMM*, 2003.

[CRhZ00] Y. Chu, S. G. Rao, and h. Zhang. A Case for End System Multicast. In *SIGMETRICS*, June 2000.

[CS02] Edith Cohen and Scott Shenker. Replication Strategies in Unstructured Peer-to-Peer Networks. In *SIGCOMM*, 2002.

[CWVL01] Maureen Chesire, Alec Wolman, Geoffrey M. Voelker, and Henry M. Levy. Measurement and Analysis of a Streaming Media Workload. In *USENIX Symposium on Internet Technologies and Systems*, 2001.

[DCKM04] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A Decentralized Network Coordinate System. In *SIGCOMM*, 2004.

[DF02] S. Daswani and A. Fisk. Gnutella UDP Extension for Scalable Searches (GUESS) v0.1. Gnutella Developer Forum, August 2002.

[DGMY03] Neil Daswani, Hector Garcia-Molina, and Beverly Yang. Open Problems in Data-Sharing Peer-to-Peer Systems. In *International Conference on Database Theory*, 2003.

[Dm03] Hauke Dmpfling. Gnutella Web Caching System: Version 2 Specifications Client Developers' Guide. http://www.gnucleus.com/gwebcache/newgwc.html, June 2003.

[EIPN04] D. Erman, D. Ilie, A. Popescu, and Arne A. Nilsson. Measurement and Analysis of BitTorrent Signaling Traffic. In *Nordic Teletraffic Seminar*, Oslo, Norway, August 2004.

[FHKM04] F. Le Fessant, S. Handurukande, A.-M. Kermarrec, and L. Massoulie. Clustering in Peer-to-Peer File Sharing Workloads. In *International Workshop on Peer-to-Peer Systems*, 2004.

[FHPW00] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *SIGCOMM*, August 2000.

[GDS+03] Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan. Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload. In *SOSP*, 2003.

[GFJ+03] Zihui Ge, Daniel R. Figueiredo, Sharad Jaiswal, Jim Kurose, and Don Towsley. Modeling Peer-Peer File Sharing Systems. In *INFOCOM*, 2003.

[GMS04] Christos Gkantsidis, Milena Mihail, and Amin Saberi. Random Walks in Peer-to-Peer Networks. In *INFOCOM*, 2004.

[Goy01] V. K. Goyal. Multiple Description Coding: Compression Meets the Network. *IEEE Signal Processing Magazine*, 18:74–93, 2001.

[HA04] Qi He and Mostafa Ammar. Congestion Control and Message Loss in Gnutella Networks. In *Multimedia Computing and Networking*, Santa Clara, CA, January 2004.

[hCGR+04] Yang hua Chu, Aditya Ganjam, Sanjay G. Rao, Kunwadee Sripanidkulchai, Jibin Zhan, and Hui Zhang. Early Experience with an Internet Broadcast System Based on Overlay Multicast. In *USENIX*, 2004.

[IEPN04] D. Ilie, D. Erman, A. Popescu, and Arne A. Nilsson. Measurement and Analysis of Gnutella Signaling Traffic. In *IPSI*, Stockholm, Sweden, September 2004.

[IUKB+04] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. A. Felber, A. Al Hamra, and L. Garces-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In *PAM*, April 2004.

[JGJ+00] J. Jannotti, D. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole Jr. Overcast: Reliable Multicasting with an Overlay Network. In *OSDI*, October 2000.

[JZ03] Song Jiang and Xiaodong Zhang. FloodTrail: An Efficient File Search Technique in Unstructured Peer-to-Peer Systems. In *Globecom*, San Francisco, CA, December 2003.

[KAD+04] Pradnya Karbhari, Mostafa Ammar, Amogh Dhamdhere, Himanshu Raj, George Riley, and Ellen Zegura. Bootstrapping in Gnutella: A Measurement Study. In *PAM*, April 2004.

[KB04] Gu-In Kwon and John W. Byers. ROMA: Reliable Overlay Multicast with Loosely Coupled TCP Connections. In *INFOCOM*, 2004.

[KG99] K. Kong and D. Ghosal. Mitigating server-side congestion in the Internet through pseudoserving. *IEEE/ACM Transactions on Networking*, 7(4):530–544, August 1999.

[KLL03] Min Sik Kim, Simon S. Lam, and Dong-Young Lee. Optimal Distribution Tree for Internet Streaming Media. In *International Conference on Distributed Computing Systems*, 2003.

[KLVW04] Alexander Klemm, Christoph Lindemann, Mary Vernon, and Oliver P. Waldhorst. Characterizing the Query Behavior in Peer-to-Peer File Sharing Systems. In *Internet Measurement Conference*, Taormina, Italy, October 2004.

[KRAV03] Dejan Kostic, Adolfo Rodriguez, Jeannie Albrecht, and Amin Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *SOSP*, 2003.

[LBBSS02] Nathaniel Leibowitz, Aviv Bergman, Roy Ben-Shaul, and Aviv Shavit. Are file swapping networks cacheable? Characterizing P2P Traffic. In *IWCW*, 2002.

[LCC+02] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and Replication in Unstructured Peer-to-Peer Networks. In *International Conference on Supercomputing*, 2002.

[LHSH04] Boon Thau Loo, Ryan Huebsch, Ion Stoica, and Joseph Hellerstein. The Case for a Hybrid P2P Search Infrastructure. In *International Workshop on Peer-to-Peer Systems*, 2004.

[LLH+03] Jinyang Li, Boon Thau Loo, Joe Hellerstein, Frans Kaashoek, David R. Karger, and Robert Morris. On the Feasibility of Peer-to-Peer Web Indexing and Search. In *International Workshop on Peer-to-Peer Systems*, 2003.

[LLS04] M. Li, W. Lee, and A. Sivasubramaniam. Semantic Small World: An Overlay Network for Peer-to-Peer Search. In *International Conference on Network Protocols*, Berlin, Germany, October 2004.

[LLX+04] Yunhao Liu, Xiaomei Liu, Li Xiao, Lionel M. Ni, and Xiaodong Zhang. Location-Aware Topology Matching in P2P Systems. In *INFOCOM*, 2004.

[LR01] Dmitri Loguinov and Hayder Radha. Measurement Study of Low-bitrate Internet Video Streaming. In *Internet Measurement Workshop*, San Francisco, CA, November 2001.

[LRS02] Qin Lv, Sylvia Ratnasamy, and Scott Shenker. Can heterogeneity make Gnutella scalable? In *IPTPS*, 2002.

[LRW03] Nathaniel Leibowitz, Matei Ripeanu, and Adam Wierzbicki. Deconstructing the Kazaa Network. In *WIAPP*, 2003.

[LZXN03] Yunhao Liu, Zhenyun Zhuang, Li Xiao, and Lionel M. Ni. AOTO: Adaptive Overlay Topology Optimization in Unstructured P2P Systems. In *Globecom*, San Francisco, CA, December 2003.

[LZXN04] Y. Liu, Z. Zhuang, L. Xiao, and L. M. Ni. A Distributed Approach to Solving Overlay Mismatching Problem. In *International Conference on Distributed Computing Systems*, 2004.

[Mar02] Evangelos P. Markatos. Tracing a large-scale Peer to Peer System: an hour in the life of Gnutella. In *CC Grid*, 2002.

[MM02] Petar Maymounkov and David Mazieres. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. In *International Workshop on Peer-to-Peer Systems*, 2002.

[MM03] Petar Maymounkov and David Mazieres. Rateless Codes and Big Downloads. In *International Workshop on Peer-to-Peer Systems*, 2003.

[NZ02] T. S. Eugene Ng and Hui Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *INFOCOM*, 2002.

[PRU01] G. Pandurangan, P. Raghavan, and E. Upfal. Building low-diameter p2p networks. In *IEEE Symposium on Foundations of Computer Science*, 2001.

[PS02] Venkata N. Padmanabhan and Kunwadee Sripanidkulchai. The Case for Cooperative Networking. In *International Workshop on Peer-to-Peer Systems*, 2002.

[PWC03] V. N. Padmanabhan, H. J. Wang, and P. A. Chou. Resilient Peer-to-Peer Streaming. In *International Conference on Network Protocols*, Atlanta, GA, November 2003.

[QS04] Dongyu Qiu and R. Srikant. Modeling and Performance Analysis of Bit Torrent-Like Peer-to-Peer Networks. In *SIGCOMM*, 2004.

[RD01] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany, November 2001.

[RFH+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. In *SIGCOMM*, 2001.

[RFI02] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design. *IEEE Internet Computing Journal*, 6(1), 2002.

[RGK04] Sean Rhea, Dennis Geels, and John Kubiatowicz. Handling Churn in a DHT. In *USENIX*, pages 127–140, 2004.

[RHE99] Reza Rejaie, Mark Handley, and Deborah Estrin. RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet. In *INFOCOM*, New York, NY, March 1999.

[RO03] Reza Rejaie and Antonio Ortega. PALS: Peer to Peer Adaptive Layered Streaming. In *Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2003.

[RS04] Reza Rejaie and Shad Stafford. A Framework for Architecting Peer-to-Peer Receiver-driven Overlays. In *Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2004.

[SBB04] Rob Sherwood, Ryan Braud, and Bobby Bhattacharjee. Slurpie: A Cooperative Bulk Data Transfer Protocol. In *INFOCOM*, 2004.

[SGD+02] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An analysis of Internet content delivery systems. In *Symposium on Operating Systems Design and Implementation*, pages 315–327, 2002.

[SGG02] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts. *Multimedia Systems Journal*, 8(5), November 2002.

[SGM04] Qixiang Sun and Hector Garcia-Molina. SLIC: A Selfish Link-based Incentive Mechanism for Unstructured Peer-to-Peer Networks. In *International Conference on Distributed Computing Systems*, 2004.

[SGMZ04] Kunwadee Sripanidkulchai, Aditya Ganjam, Bruce Maggs, and Hui Zhang. The Feasibility of Supporting Large-Scale Live Streaming Applications with Dynamic Application End-Points. In *SIGCOMM*, 2004.

[SMB02] Tyron Stading, Petros Maniatis, and Mary Baker. Peer-to-Peer Caching Schemes to Address Flash Crowds. In *International Workshop on Peer-to-Peer Systems*, 2002.

[SMK+01] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *SIGCOMM*, San Diego, CA, August 01.

[SMZ03] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang. Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. In *INFOCOM*, 2003.

[SMZ04] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang. An Analysis of Live Streaming Workloads on the Internet. In *Internet Measurement Conference*, Taormina, Italy, October 2004.

[Sri01] K. Sripanidkulchai. The popularity of Gnutella queries and its implications on scalability. http://www-2.cs.cmu.edu/ kunwadee/research/p2p/paper.html, January 2001.

[SRS02] Angelos Stavrou, Dan Rubenstein, and Sambit Sahu. A Lightweight, Robust P2P System to Handle Flash Crowds. In *International Conference on Network Protocols*, Paris, France, November 2002.

[ST04] Yuval Shavitt and Tomer Tankel. On the Curvature of the Internet and its usage for Overlay Construction and Distance Estimation. In *INFOCOM*, 2004.

[SW04] Subhabrata Sen and Jia Wang. Analyzing Peer-To-Peer Traffic Across Large Networks. *IEEE/ACM Transactions on Networking*, 12(2):219–232, April 2004.

[VAM+02] Eveline Veloso, Virgilio Almeida, Wagner Meira, Azer Bestavros, and Shudong Jin. A Hierarchical Characterization of a Live Streaming Media Workload. In *Internet Measurement Workshop*, Marseille, France, November 2002.

[WC04] Rita H. Wouhaybi and Andrew T. Campbell. Phenix: Supporting Resilient Low-Diameter Peer-to-Peer Topologies. In *INFOCOM*, 2004.

[WXLZ04] Chen Wang, Li Xiao, Yunhao Liu, and Pei Zheng. Distributed Caching and Adaptive Search in Multilayer P2P Networks. In *ICDCS*, 2004.

[XHHB02] Dongyan Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava. On Peer-to-Peer Media Streaming. In *International Conference on Distributed Computing Systems*, 2002.

[YdV04] Xiangying Yang and Gustavo de Veciana. Service Capacity of Peer to Peer Networks. In *INFOCOM*, 2004.

[YGM01] Beverly Yang and Hector Garcia-Molina. Comparing Hybrid Peer-to-Peer Systems. In *VLDB*, 2001.

[YGM02] Beverly Yang and Hector Garcia-Molina. Improving search in peer-to-peer systems. In *ICDCS*, 2002.

[YGM03] Beverly Yang and Hector Garcia-Molina. Designing a Super-Peer Network. In *International Conference on Data Engineering*, March 2003.

[YVGM04] Beverly Yang, Patrick Vinograd, and Hecter Garcia-Molina. Evaluating GUESS and Non-Forwarding Peer-to-Peer Search. In *IEEE International Conference on Distributed Systems*, 2004.

[ZA03] L. Zou and M. Ammar. A File-Centric Model for Peer-to-Peer File Sharing Systems. In *International Conference on Network Protocols*, Atlanta, GA, November 2003.

[ZLG04] Ying Zhu, Baochun Li, and Jiang Guo. Multicast with network coding in application-layer overlay networks. *IEEE Journal on Selected Areas is Communications*, 22(1), January 2004.

[ZZA02] L. Zou, E. W. Zegura, and M. H. Ammar. The effect of peer selection and buffering strategies on the performance of peer-to-peer file sharing systems. In *MASCOTS*, Fort Worth, Texas, October 2002.