Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems

Daniel Stutzbach, *Member, IEEE*, Reza Rejaie, *Senior Member, IEEE*, and Subhabrata Sen, *Member, IEEE*

***Abstract*—In recent years, peer-to-peer (P2P) file-sharing systems have evolved to accommodate growing numbers of participating peers. In particular, new features have changed the properties of the unstructured overlay topologies formed by these peers. Little is known about the characteristics of these topologies and their dynamics in modern file-sharing applications, despite their importance.**

This paper presents a detailed characterization of P2P overlay topologies and their dynamics, focusing on the modern Gnutella network. We present Cruiser, a fast and accurate P2P crawler, which can capture a complete snapshot of the Gnutella network of more than one million peers in just a few minutes, and show how inaccuracy in snapshots can lead to erroneous conclusions—such as a power-law degree distribution. Leveraging recent overlay snapshots captured with Cruiser, we characterize the graph-related properties of individual overlay snapshots and overlay dynamics across slices of back-to-back snapshots. Our results reveal that while the Gnutella network has dramatically grown and changed in many ways, it still exhibits the clustering and short path lengths of a small world network. Furthermore, its overlay topology is highly resilient to random peer departure and even systematic attacks. More interestingly, overlay dynamics lead to an "onion-like" biased connectivity among peers where each peer is more likely connected to peers with higher uptime. Therefore, long-lived peers form a stable core that ensures reachability among peers despite overlay dynamics.

***Index Terms*—File sharing, Gnutella, measurement, overlay topology, peer-to-peer.**

I. Introduction

The Internet has witnessed a rapid growth in the popularity of various Peer-to-Peer (P2P) applications during recent years. In particular, today's P2P file-sharing applications (e.g., FastTrack, eDonkey, Gnutella) are extremely popular with millions of simultaneous clients and contribute a significant portion of the total Internet traffic [1]–[3]. These applications have evolved over the past several years to accommodate growing numbers of participating peers. In these applications, participating peers form an overlay which provides connectivity among the peers, allowing users to search for desired files. Typically, these overlays are *unstructured* where peers select neighbors through a predominantly ad hoc process—this is different from *structured* overlays, i.e., distributed hash tables such as Chord [4] and CAN [5]. Most modern file-sharing networks use a *two-tier* topology where a subset of peers, called *ultrapeers*, form an unstructured sparse graph while other participating peers, called *leaf peers*, are connected to the top-level overlay through one or multiple ultrapeers. More importantly, the overlay topology is continuously reshaped by both user-driven dynamics of peer participation as well as protocol-driven dynamics of neighbor selection. In a nutshell, as participating peers join and leave, they collectively, in a decentralized fashion, form an unstructured and dynamically changing overlay topology.

This work focuses on developing an accurate understanding of the topological properties and dynamics of large-scale unstructured P2P networks, via a case study. Such an understanding is crucial for the development of P2P networks with superior features including better search, availability, reliability and robustness capabilities. For instance, the design and simulation-based evaluation of new search and replication techniques has received much attention in recent years [6]–[9]. These studies often make certain assumptions about topological characteristics of P2P networks (e.g., a power-law degree distribution) and usually ignore the dynamic aspects of overlay topologies. However, little is known today about the topological characteristics of popular P2P file sharing applications, particularly about overlay dynamics. An important factor to note is that properties of unstructured overlay topologies cannot be easily derived from the neighbor selection mechanisms due to implementation heterogeneity and dynamic peer participation. Without a solid understanding of the topological characteristics of file-sharing applications, the actual performance of the proposed search and replication techniques in practice is unknown and cannot be meaningfully simulated. In this case study, we examine one of the most popular file-sharing systems, Gnutella, to cast light on the topological properties of peer-to-peer systems.

Accurately capturing the overlay topology of a large scale P2P network is challenging. A common approach is to use a topology crawler [10], [11] that progressively queries peers to determine their neighbors. The captured topology is a *snapshot* of the system as a graph, with the peers represented as vertices and the connections as edges. However, capturing accurate snapshots is inherently difficult for two reasons: (i) overlay

Manuscript received May 25, 2006; revised October 20, 2006, February 7, 2007, and February 9, 2007; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor K. Ross. This material is based upon work supported by the National Science Foundation (NSF) under Grant Nets-NBD-0627202 and an unrestricted gift from Cisco Systems. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF or Cisco.

D. Stutzbach is with Stutzbach Enterprises, LLC, Dallas, TX 75206 USA (e-mail: daniel@stutzbachenterprises.com).

R. Rejaie is with the Computer and Information Science Department, University of Oregon, Eugene, OR 97403 USA (e-mail: reza@cs.uoregon.edu).

S. Sen is with AT&T Labs-Research, Florham Park, NJ 07932 USA (e-mail: sen@research.att.com).

Digital Object Identifier 10.1109/TNET.2007.900406

topologies change as the crawler operates and (ii) a non-negligible fraction of peers in each snapshot are not directly reachable by the crawler. When a crawler is slow relative to the rate of overlay change, the resulting snapshot will be significantly distorted. Furthermore, verifying the accuracy of a crawler's snapshots is difficult due to the absence of authoritative reference snapshots. We introduce techniques for studying the accuracy of a crawler in Section II-E.

Previous studies that captured P2P overlay topologies with a crawler either rely on slow crawlers, which inevitably lead to significantly distorted snapshots of the overlay [10]–[12], or capture only a portion of the overlay [13], [14] which is likely to be biased (and non-representative) [15]. These studies do not examine the accuracy of their captured snapshots and only conduct limited analysis of the overlay topology. More importantly, these few studies (except [14]) are outdated (more than three years old), since P2P filesharing applications have significantly increased in size and incorporated several new topological features over the past few years. An interesting recent study [14] presents a high level characterization of the two-tier Kazaa overlay topology. However, the study does not explore graph properties of the overlay in detail. Finally, to our knowledge, the dynamics of unstructured P2P overlay topologies have not been studied in detail in any prior work.

A. Contributions

This paper presents 1) Cruiser, a fast crawler for two-tier peer-to-peer systems such as Gnutella, and 2) detailed characterizations of both graph-related properties as well as the dynamics of unstructured overlay topologies based on recent large-scale and accurate measurements of the Gnutella network using the crawler.[1]

We have recently developed a set of measurement techniques and incorporated them into a fast parallel P2P crawler, called *Cruiser*. Cruiser can accurately capture a complete snapshot of the Gnutella network of more than one million peers in just a few minutes. Its speed is several orders of magnitude faster than any previously reported P2P crawler, and thus its captured snapshots are significantly more accurate. Capturing snapshots rapidly also allows us to examine the dynamics of the overlay with finer granularity, which was not feasible in previous studies. In Section II, we present the different techniques used in Cruiser to achieve its high speed, including leveraging the two-tier structure, a distributed architecture, asynchronous communications, and choosing appropriate timeout values. We also present techniques for quantifying the measurement inaccuracy introduced by crawl speed and present evidence that the error in Cruiser's snapshots is reasonably small.

Using Cruiser, we have captured several hundred snapshots of the Gnutella network. We use these snapshots to characterize the Gnutella topology on two levels:

- *Graph-related Properties of Individual Snapshots*: We treat individual snapshots of the overlay as graphs and apply different forms of graph analysis to examine their properties, in Section III.
- *Dynamics of the Overlay*: We present new methodologies to examine the dynamics of the overlay and its evolution over different timescales, in Section IV.

We investigate the underlying causes of the observed properties and dynamics of the overlay topology. Our main findings can be summarized as follows:

- In contrast to earlier studies [10], [11], [20], we find that node degree does not exhibit a power-law distribution. We show how power-law degree distributions can be a result of measurement artifacts.
- While the Gnutella network has dramatically grown and changed in many ways, it still exhibits the clustering and the short path lengths of a small world network. Furthermore, its overlay topology is highly resilient to random peer departure and even systematic removal of high-degree peers.
- Long-lived ultrapeers form a stable and densely connected *core overlay*, providing stable and efficient connectivity among participating peers despite the rapid dynamics of peer participation.
- The longer a peer remains in the overlay, the more it becomes clustered with other long-lived peers with similar uptime.[2] In other words, connectivity within the core overlay exhibits an "onion-like" bias where the longest-lived peers form a well-connected core, and peers with shorter uptime form layers with biased connectivity to each other and to peers with higher uptime (i.e., inner layers).

B. Why Examine Gnutella?

eDonkey, FastTrack, and Gnutella are the three most popular P2P file-sharing applications today, according to Slyck.com [1], a website which tracks the number of users of different P2P applications. We elected to first focus on the Gnutella network due to a number of considerations.

First, a variety of evidence indicates that the Gnutella network has a large and growing population of active users and generates considerable traffic volume. Fig. 1 depicts the average size of the Gnutella network over an eleven month period ending February 2005, indicating that network size has more than tripled (from 350 000 to 1.3 million peers) during our measurement period. We also observed time-of-day effects in the size of captured snapshots, which is a good indication of active user participation in the Gnutella network. Also, examination of Internet2 measurement logs[3] reveal that the estimated Gnutella traffic measured on that network is considerable and growing. For example, for the 6 week period 10/11/04–11/21/04, the Gnutella traffic on Internet2 was estimated to be 79.69 terabytes, up from 21.52 terabytes for a 6 week period (02/02/04–03/14/04) earlier that year.

Second, Gnutella, which was one of the first decentralized P2P systems, has evolved significantly since its inception in 2000. While it is among the most studied P2P networks in the literature, prior studies are at least three years old and consider the earlier flat-network incarnation. A detailed measurement study

[1]Earlier versions of different components of this work appeared in [16]–[19].

[2]Throughout this paper, by "uptime" we mean the time that has elapsed since the beginning of the peer's session.

[3]http://www.netflow.Internet2.edu/weekly/

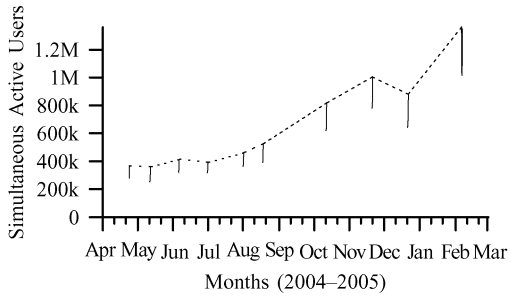


Fig. 1. Change in network size over months. Vertical bars show variation within a single day.

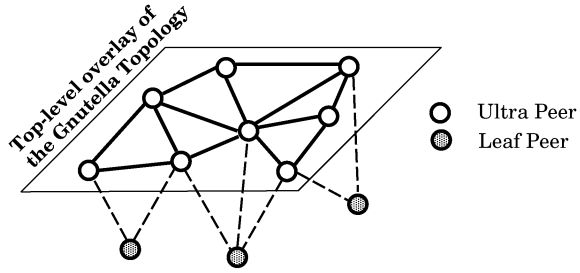


Fig. 2. Two-tier topology of modern Gnutella.

of the modern two-tier Gnutella network is therefore timely and allows us to compare and contrast the behavior today from the earlier measurement studies and gain insights into the behavior and impact of the two-tier topologies adopted by most modern P2P systems.

Third, our choice was also influenced by the fact that Gnutella is the most popular P2P file-sharing network with an open and well-documented protocol specification. This eliminates (or at least significantly reduces) any incompatibility error in our measurement that could potentially occur in other proprietary P2P applications that have been reverse-engineered, such as FastTrack/Kazaa and eDonkey.

The rest of this paper is organized as follows. We describe the key features of Cruiser in Section II, empirically explore the impact of crawling speed on snapshot accuracy, and quantify the accuracy of Cruiser's snapshots. Section III presents a detailed characterization of graph-related properties of individual snapshots and discusses the implications of our findings. In Section IV, we examine overlay dynamics, their underlying causes, and their implications for the design and evaluation of P2P applications. Section V presents an overview of related work, and Section VI concludes the paper.

II. Capturing Accurate Snapshots

In this section, we begin with a brief overview of modern Gnutella as an example of a two-tier P2P system, and describe the various technical challenges to capturing accurate snapshots of a dynamic unstructured P2P system. We then present the design of a fast parallel P2P crawler, called *Cruiser*, which incorporates a set of measurement techniques we develped to address the above challenges. Finally we explore and quantify the accuracy of the snapshots gathered by Cruiser.

A. Modern Gnutella

We briefly describe a few key features of modern Gnutella [21] that are related to our study. The original Gnutella protocol has limited scalability due to its flat overlay and simple flooding scheme. To improve scalability, modern Gnutella clients adopt a *two-tier* overlay architecture. As shown in Fig. 2, in this architecture a subset of peers, called *ultrapeers* (or super-peers), form a *top-level* overlay while the majority of participating peers, called *leaf peers*, are connected to the top-level overlay through one or multiple ultrapeers. Ultrapeers communicate with one another using a superset of the original Gnutella protocol.[4] When a leaf peer cannot find an available ultrapeer, it reconfigures itself as an ultrapeer after verifying that it has high bandwidth and can receive incoming connections (i.e., is not firewalled). In this way, the network maintains a proper ratio between ultrapeers and leaf peers. FastTrack (or Kazaa) and eDonkey also use some variation of this model.

Another key feature is *Dynamic Querying* [22], which adjusts the query scope to gather only enough results to satisfy the user (typically 50–200 results). An ultrapeer forwards a query to a subset of top-level neighbors using a low TTL. From that point, the query propagates normally until the TTL expires, and the ultrapeer waits for results. If enough results are found, the query terminates. Otherwise, the ultrapeer estimates how many more peers must be searched. The ultrapeer then sends the query via additional neighbors with a TTL estimated to return sufficient results. This process is repeated if necessary. Each ultrapeer estimates the number of visited ultrapeers through each neighbor based on the following formula: $\sum_{i=0}^{\mathrm{TTL}-1}(d-1)^i$. This formula assumes that all peers have the same node degree, d.

B. Challenges

To accurately characterize P2P overlay topologies, we need to capture *complete* and *accurate* snapshots. By "snapshot", we refer to a graph that captures all participating peers (as nodes) and the connections between them (as edges) at a particular time. The only way to capture a complete snapshot is to crawl the overlay. Given information about a handful of initial peers, the crawler progressively contacts participating peers and collects information about their neighbors. In practice, capturing accurate snapshots is challenging for two reasons:

(i) **The Dynamic Nature of Overlays:** Crawlers are not instantaneous and require time to capture a complete snapshot. Because of the dynamic nature of peer participation and neighbor selection, the longer a crawl takes, the more changes occur in participating peers and their connections, and the more *distorted* the captured snapshot becomes. More specifically, any connection that is established or closed during a crawl (i.e., *changing connections*) is likely to be reported only by one end of the connection. We note that there is no reliable way to accurately

[4]Initially the top-level overlay was composed of a mixture of Ultrapeers and ordinary peers. After being deprecated for a few years, ordinary peers are no longer permitted anywhere in the overlay. All peers must be either ultrapeers or leaf peers.

resolve the status of changing peers or changing connections. In a nutshell, any captured snapshot by a crawler will be distorted, where the degree of distortion is a function of the crawl duration relative to the rate of change in the overlay.

(ii) **Unreachable Ultrapeers:** A significant portion of discovered peers in each snapshot are not directly reachable since they have departed, reside behind a firewall, or are overloaded. Therefore, information about the connections between unreachable ultrapeers will be missing from the captured snapshots.

Using either partial crawls [13] or via passive monitoring [23] is not a reliable technique for gathering accurate snapshots for the following reasons: (i) in the absence of adequate knowledge about the properties and dynamics of the overlay topology, it is difficult to collect unbiased samples. For example, partial crawling of the network can easily result in a snapshot that is biased towards peers with higher degree [15]; similarly passive monitoring by its nature is limited to information gleamed from the communications that are visible to the monitoring station(s). Also for both partial crawls and passive monitoring, the introduced bias and its extent is unknown, making it impossible to derive representative characterizations for the whole network; and (ii) some graph-level characteristics of the overlay topology, such as the mean shortest path between peers (which we discuss in Section III-B) cannot be derived from partial snapshots. Because of these reasons, we attempt to capture snapshots that are as complete as possible and use them for our characterizations.

C. The Gnutella Cruiser

To minimize the distortion in captured snapshots caused by the dynamic nature of the overlay, we have developed a fast Gnutella crawler, called *Cruiser*. While the basic crawling strategy employed by Cruiser is similar to other crawlers, it improves the accuracy of captured snapshots by significantly increasing the crawling speed (i.e., reducing crawl duration) by using the following techniques.

First, Cruiser leverages the two-tier structure of the modern Gnutella network by only crawling ultrapeers. Since each leaf must be connected to an ultrapeer, this approach enables us to capture all the nodes and links of the overlay by contacting a relatively small fraction of all peers. Overall, this strategy leads to around an 85% reduction in the duration of a crawl without any loss of information.

Second, Cruiser crawls hundreds of peers in parallel using asynchronous communications. While parallelism improves performance, attempting to employ too much parallelism leads to high CPU load and eventually an inability to keep up with network traffic. Cruiser implements an adaptive load management mechanism to ensure its CPU remains busy but does not become overwhelmed. Towards this end, Cruiser monitors its CPU load and adjusts its maximum number of parallel connections using an AIMD algorithm similar to TCP's congestion control mechanism. In practice, Cruiser typically runs with close to 1000 parallel connections, contributing an additional speed-up of nearly three orders of magnitude, compared to sequential crawling (e.g., [10]).

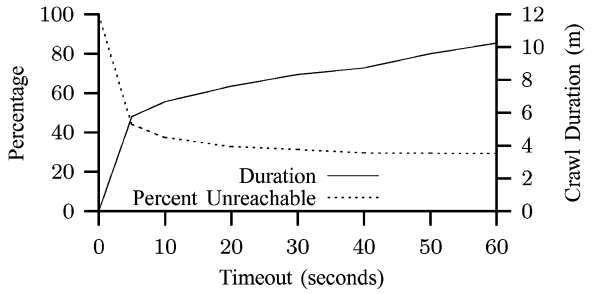


Fig. 3. Effects of the timeout length on crawl duration and snapshot completeness.

Third, Cruiser employs a master-slave architecture in order to further increase the level of parallelism and utilize the resources of multiple PCs. The master process coordinates multiple slave processes that crawl disjoint portions of the network in parallel. The master-slave architecture provides an additional speedup proportional to the number of slave machines.

Fourth, Cruiser uses an appropriate timeout length when waiting for responses from peers. When peers are unresponsive, it could take a long time to wait for TCP to time out. In our systems, a full TCP timeout to an unresponsive address takes more than 3 min. While this is suitable for many interactive and automated applications, we conducted an evaluation of the cost versus benefit of different timeout lengths for crawling. Fig. 3 shows the duration of the crawl and the percentage of peers that were unreachable as a function of the timeout length. This figure shows that very low timeouts (less than 10 s) result in a dramatic increase in the number of unreachable ultrapeers, while longer timeouts do not significantly decrease the percent of unreachable ultrapeers. In other words, if a peer has not responded after 10 s, it is unlikely to ever respond. There are diminishing returns for using longer timeout lengths, as the crawl duration (and thus distortion) continues to increase. Therefore, we use a timeout of 10 s, providing an additional speedup of more than a factor of two.

We have experienced other system issues in the development of Cruiser that are worth mentioning. In particular, we needed to increase the limit on the number of open file descriptors on the host systems. Otherwise, many connection attempts return immediately with an automatic "Connection Refused" error. In a similar vein, we increased the number of connections that our lab firewall could track to prevent the firewall from dropping packets due to this constraint.

These techniques collectively result in a significant increase in crawling speed. *Cruiser can capture the Gnutella network with one million peers in around 7 min using six off-the-shelf 1 GHz GNU/Linux boxes in our lab. Cruiser's crawling speed is about 140 k peers/minute (by directly contacting 22 k peers/minute). This is orders of magnitude faster than previously reported crawlers (i.e., 2 hours for 30 k peers (250/minute) in [10], and 2 min for 5 k peer (2.5 k/minute) in [13])*. While our crawling strategy is aggressive and our crawler requires considerable local resources, it is not intrusive for the remote peers, since each ultrapeer is contacted only once per crawl.

Postprocessing: Once information is collected from all reachable peers, we perform some postprocessing to remove

any obvious inconsistencies that might have been introduced due to changes in the topology during the crawling period. Specifically, we include edges even if they are only reported by one peer, and treat a peer as an ultrapeer if it neighbors with another ultrapeer or has any leaves. Due to the inconsistencies, we might over-count edges by about 1% and ultrapeers by about 0.5%.

D. Effect of Unreachable Ultrapeers

In this section, we carefully examine the effect of unreachable ultrapeers on the accuracy of captured snapshots. Unreachable ultrapeers can introduce the following errors in a captured snapshot: (i) including unreachable ultrapeers that departed, (ii) missing links between unreachable ultrapeers and their leaves, and (iii) missing links between two unreachable ultrapeers.

Interestingly, our measurements revealed that some of the unreachable ultrapeers are actually overwhelmed ultrapeers that sporadically accept TCP connections and can be contacted after several attempts. This transport-layer refusal means that the application is not able to call *accept()* sufficiently fast, leading to a TCP listen buffer overflow. We also noticed that connections to most of these overwhelmed ultrapeers exhibit long RTT ($>$1 sec) and little to no loss. Since latency due to a long queue in a router is typically accompanied by packet loss, this suggests the peer's CPU may be the bottleneck and the operating system is taking a long time to process the packets. Despite this finding, we did not incorporate a multiple attempt strategy into the crawler for two reasons: (i) it only marginally increases the number of reachable peers at the cost of significantly increasing the duration of each crawl which in turn increases distortion in captured snapshots; and (ii) it is intrusive and may exacerbate the existing problem.

It is important to determine what portion of unreachable ultrapeers are departed versus firewalled or overloaded, because each group introduces a different error on the snapshot. However, there is no reliable test to distinguish the three cases, because firewalls can time out or refuse connections depending on their configuration. Previous studies assume that these unreachable ultrapeers are either departed or firewalled and exclude them from their snapshots.

To determine the status of unreachable ultrapeers, we devise the following technique to identify the fraction of unreachable ultrapeers that departed. We perform back-to-back crawls to capture two snapshots. We can then conclude that the unreachable ultrapeers in the first snapshot that are missing from the second snapshot departed in the first snapshot. This approach reveals that departed peers constitute only 2–3% of peers in each snapshot.

Finally, we examine those unreachable ultrapeers that time out. Since overwhelmed ultrapeers refuse connections, we hypothesized that this group of peers is firewalled. To verify this hypothesis, we randomly selected 1000 (about 3% of) peers that were unreachable due to time out, and re-contacted them every 5 min for 7 hours.[5] Interestingly, more than 92% of these peers were never reachable at all. This suggests that the timeout is a good indicator for firewalled peers. *In summary, our investigation reveals that in each crawl, 30%–38% of discovered peers are unreachable. In this group, the breakdown is as follows: 2%–3% departed, 15%–24% firewalled, and the remaining unreachable ultrapeers (3%–21%) are either also firewalled or overwhelmed ultrapeers*. Since Cruiser only needs to contact *either* end of an edge, it is able to discover at least 85%–91% of edges. Since firewalled peers cannot directly connect together (i.e., cannot be located at both ends of a missing edge) and they constitute more than half of the unreachable ultrapeers, the actual portion of missing edges is considerably smaller.

[5]Note that each attempt translates into several attempts by TCP to establish a connection by sending SYN packets.

E. Quantifying Snapshot Accuracy

In this section, we rigorously examine the accuracy of captured snapshots by Crusier. Snapshot accuracy can not be directly measured since there is no reference snapshot for comparison. Therefore, we indirectly quantify the the effect of crawling speed and duration on two dimensions of snapshot accuracy: completeness and distortion.

Impact of Crawling Speed: To examine the impact of crawling speed on the accuracy of captured snapshots, we adjust the crawling speed (and thus the crawl duration) of Cruiser by changing the number of parallel connections that each slave process can open. Using this technique, Cruiser can effectively emulate the behavior of previously reported crawlers which have a lower degree of concurrency.

We define the *edge distortion* of a snapshot as the percentage of edges that are reported by only one of two contacted peers, i.e., those created or torn down during the crawl. Unfortunately, there is no straightforward way to validate a snapshot to check for peer distortion. Instead, we examine the sets of peers from two snapshots captured back-to-back (P_1 and P_2). The first snapshot (P_1) serves as a reference snapshot, captured at maximum speed, while we vary the speed of the second snapshot (P_2). We then define the *peer distortion* as $\delta = |P_1 \Delta P_2|/(|P_1| + |P_2|)$, where Δ denotes the symmetric difference operation. In other words, peer distortion is 0% if the snapshots are identical and 100% if the snapshots do not have any common peers.

Fig. 4 depicts peer and edge distortion as a function of crawl duration. This figure demonstrates that the accuracy of snapshots decreases with the duration of the crawl, because the increased distortion reflects changes in the topology that occur *while the crawler is running*. Crawlers that take 1–2 hours (comparable to those in earlier works) have a peer distortion of 9%–15% and an edge distortion of 31%–48%, while at full speed, Cruiser exhibits a peer distortion of only 4% and an edge distortion of only 13%.

Completeness of Snapshots: To examine the completeness of snapshots captured by Cruiser, we keep track of the following variables during each crawl: the number of discovered top-level peers, the number of leaves, the number of links between ultrapeers, and the number of links to leaves. Fig. 5 presents variations of these four variables as a function of the number of contacted peers in a sample crawl. Note that the number of discovered top-level peers as well as leaves curve off which is evidence that Cruiser has captured nearly all the participating

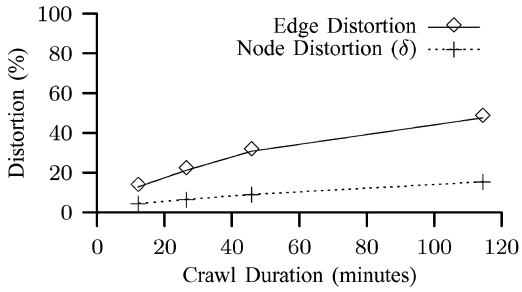


Fig. 4. Effect of crawl speed on the accuracy of captured snapshots.

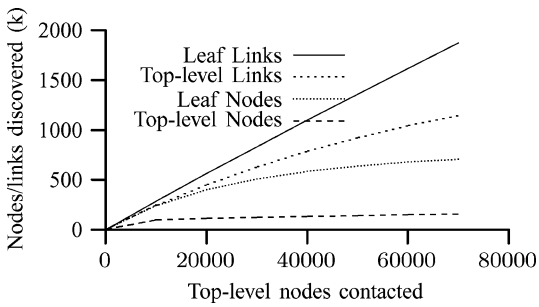


Fig. 5. Cumulative dicovered information about overlay nodes and links as a function of number of contacted peers.

peers. Links between top-level peers somewhat curves off. Finally, links to leaves is necessarily linearly increasing with the number of top-level peers because each top-level peers provide a unique set of links between itself and its leaves.

Completeness–Duration Tradeoff: To examine the completeness–duration tradeoff for captured snapshots, we modified Cruiser to stop the crawl after a specified period. Then, we performed two back-to-back crawls and repeated this process for different durations. Fig. 6 demonstrates the completeness–duration tradeoff. During short crawls (on the left side of the graph), δ is high because the captured snapshot is incomplete, and each crawl captures a different subset. As the duration of the crawl increases, δ decreases which indicates that the captured snapshot becomes more complete. Increasing the crawl length beyond 4 min does not decrease δ any further, and achieves only a marginal increase in the number of discovered peers (i.e., completeness). This figure reveals a few important points. First, there exists a "sweet spot" for crawl duration beyond which crawling has diminishing returns if the goal is simply to capture the population. Second, for sufficiently long crawls, Cruiser can capture a relatively unstretched snapshot. Third, the change of $\delta = 4\%$ is an upper-bound on the distortion due to the passage of time as Cruiser runs. The relatively flat delta on the right suggest that around 4% of the network is unstable and turns over quickly.

In summary, our evaluations reveal that (i) Cruiser captures nearly all ultrapeers and the pair-wise connections between them and the majority of connections to leaves; (ii) both node distortion and edge distortion in captured snapshots increases linearly with the crawl duration; and (iii) snapshots captured by Cruiser have little distortion. In particular, we found that two back-to-back snapshots differed only 4% in their peer populations.

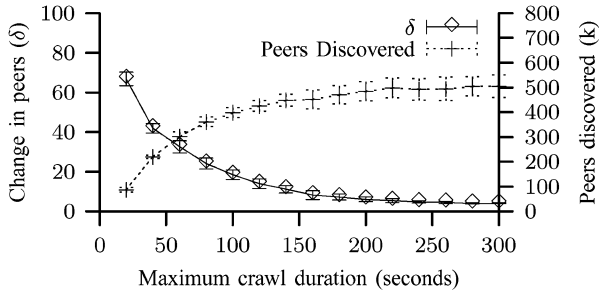


Fig. 6. Error as a function of maximum crawl duration, generated by running two crawls back-to-back for each x-value and computing δ. Averaged over 8 runs with standard deviation shown.

F. Data Set

We make use of several hundred snapshots of the Gnutella network captured during the past eleven months (Apr. 2004–Feb. 2005) with Cruiser. In particular, we collected back-to-back snapshots for several two-day intervals as well as snapshots from several times of the day to ensure that captured snapshots are representative. In Section III, we use four of these snapshots to illustrate graph properties of the overlay topology. In Section IV, we use sets of hundreds of back-to-back snapshots to examine how the overlay topology evolves with time.

III. Overlay Graph Properties

The two-tier overlay topology in modern Gnutella (as well as other unstructured P2P networks) consists of ultrapeers that form a "spaghetti-like" top-level overlay and a large group of leaf peers that are connected to the top-level through multiple ultrapeers. We treat individual snapshots of the overlay as graphs and apply different forms of graph analysis to examine their properties. We pay special attention to the top-level overlay since it is the core component of the topology. Throughout our analysis, we compare our findings with similar results reported in previous studies. As the top-level of the modern Gnutella network grew out of the original Gnutella topology, we compare properties of the top-level of the modern Gnutella network with earlier work on the original Gnutella topology. However, it is important to note that we are unable to determine whether the reported differences (or similarities) are due to changes in the Gnutella network or due to inaccuracy in the captured snapshots of previous studies.

Table I presents summary information of four sample snapshots after postprocessing. The results in this section are primarily from the snapshots in Table I. However, we have examined many other snapshots and observed similar trends and behaviors. Therefore, we believe the presented results are representative. Presenting different angles of the same subset of snapshots allows us to conduct cross comparisons and also relate various findings.

In this section, we explore the node degree distribution in Section III-A, the reachability and pairwise distance properties of the overlay in Section III-B, small world characteristics in Section III-C, and the resilience of the overlay in Section III-D.

Implementation Heterogeneity: The open nature of the Gnutella protocol has led to several interoperable implementations. It is important to determine the distribution of different

TABLE I
SAMPLE CRAWL STATISTICS

Crawl Date	Total Nodes	Leaves	Top-level	Unreachable	Top-Level Edges
09/27/04	725,120	614,912	110,208	35,796	1,212,772
10/11/04	779,535	662,568	116,967	41,192	1,244,219
10/18/04	806,948	686,719	120,229	36,035	1,331,745
02/02/05	1,031,471	873,130	158,345	39,283	1,964,121

TABLE II
DISTRIBUTION OF IMPLEMENTATIONS

Implementation:	LimeWire	BearShare	Other
Percentage:	74%–77%	19%–20%	4%–6%

implementations (and configurations) among participating peers since the implementation design choices directly affect the overall properties of the overlay topology. This will help us explain some of the observed properties of the overlay. Table II presents the distribution of different implementations across discovered ultrapeers. This table shows that a clear majority of contacted ultrapeers use the LimeWire implementation. We also discovered that a majority of LimeWire ultrapeers (around 94%) use the most recent version of the software available at the time of the crawl. These results reveal that while heterogeneity exists, nearly all Gnutella users run LimeWire or BearShare.

We are particularly interested in the number of connections that are used by each implementation since this design choice directly affects the degree distribution of the overall topology. For LimeWire, this information can readily be obtained from the source code: LimeWire attempts to maintain 27 neighbors at minimum and will accept up to 32 at maximum. However, implementations such as BearShare are not open-source. Additionally, users can always change the source code of open implementations. Thus, we must still collect this information from running ultrapeers in action.

Our measurements reveal that LimeWire's and BearShare's ultrapeer implementations prefer to serve 30 and 45 leaves, respectively, whereas both try to maintain around 30 neighbors in the top-level overlay.

A. Node Degree Distributions

The introduction of the two-tier architecture in the overlay topology along with the distinction between ultrapeers and leaf peers in the modern Gnutella protocol demands a close examination of the different degree distributions among different group of peers.

Node Degree in the Top-Level Overlay: Previous studies reported that the distribution of node degree in the Gnutella network exhibited a power-law distribution [10], [11], [24] and later changed to a two-segment power-law distribution [10], [20]. Fig. 8(a) depicts the distribution of node degree in log-log scale among all peers in the top-level overlay for the 10/18/04 snapshot (labeled "Fast Crawl"). This distribution has a spike around 30 and does not follow a power-law, which would exhibit a line-like tail when plotted in log-log scale. A key question is *to what extent this difference in degree distribution is due to the change in the overlay structure versus error in captured snapshots by earlier studies*. To examine this question, we captured a distorted snapshot by a slow crawler[6] which is similar to the 50-connection crawler used in an earlier study [10]. Fig. 8(a) depicts the degree distribution based on this distorted snapshot in log-log scale, which is similar to the power-law distribution reported in [10, Fig. 6].[7] If we further slow down the crawling speed, the resulting snapshots contain greater edge distortion and the derived degree distribution looks more similar to a single-piece power-law distribution, the result reported by earlier studies [11], [24]. *To a slow crawler, peers with long uptimes appear as high degree because many short-lived peers report them as neighbors. However, this is a mischaracterization since these short-lived peers are not all present at the same time. More importantly, this finding demonstrates that using distorted snapshots that are captured by slow crawlers can easily lead to incorrect characterizations of P2P overlays*.

Fig. 7(a) presents the degree distribution of top-level peers for the four snapshots presented in Table I, in linear scale. Because we were unable to contact every top-level peer, the distribution in Fig. 7(a) is biased slightly low since it does not include all edges.[8] To address this problem, we split the data into Figs. 7(b) and (c), which depict the neighbor degree distribution for reachable and unreachable peers, respectively. The data in Fig. 7(b) is unbiased since it includes data only from peers we contacted successfully, i.e., we discovered every edge connected to these peers. The spike around a degree of 30 is more pronounced in this figure. Fig. 7(c) presents the observed degree distribution for unreachable top-level peers (i.e., overloaded or NATed). This distribution is biased low since we cannot observe the connections between pairs of these peers. In this data, a much greater fraction of peers have an observed degree below 30, compared to Fig. 7(b). Many of these peers probably have a true degree closer to 30, with the true distribution likely similar to that in Fig. 7(b).

The degree distribution among contacted top-level peers has two distinct segments around a spike in degree of 30, resulting from LimeWire and BearShare's behavior of attempting to maintain 30 neighbors. The few peers with higher degree represent other implementations that try to maintain a higher node degree or the rare user who has modified their client software. The peers with lower degree are peers which have not yet established 30 connections. In other words, the observed degree for these peers is temporary. They are in a state of flux, working on opening more connections to increase their degree. To verify this hypothesis, we plot the mean degree

[6]To reduce the crawling speed, we simply limited the degree of concurrency (i.e., number of parallel connections) to 60 in Cruiser.

[7]To properly compare these snapshots with different sizes, the y axis in Fig. 8(a) is normalized by the number of peers in the snapshot.

[8]The degree distribution for all the presented results is limited to 50, which includes all but a small percentage ($<1\%$) of peers with larger degree that are discussed later.

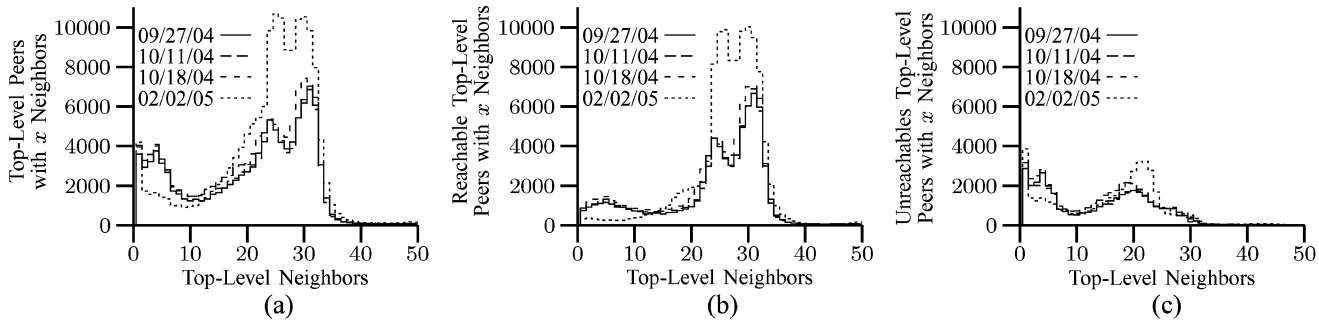


Fig. 7. Different angles of the top-level degree distribution in Gnutella topology. (a) Top-level degree distribution; (b) reachable degree distribution; (c) unreachable degree distribution.

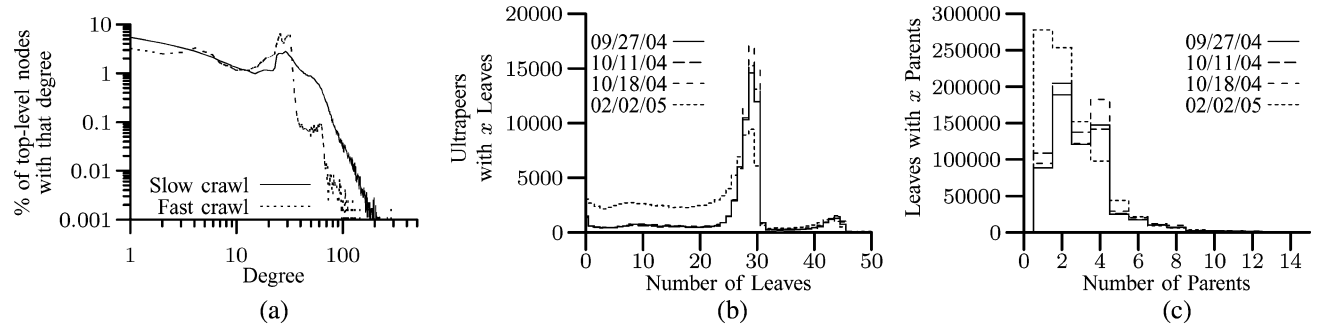


Fig. 8. Different angles of degree distribution in Gnutella. (a) Observed top-level degree distributions of a slow and a fast crawl; (b) degree distribution from ultrapeers to leaves; (c) leaf parents.

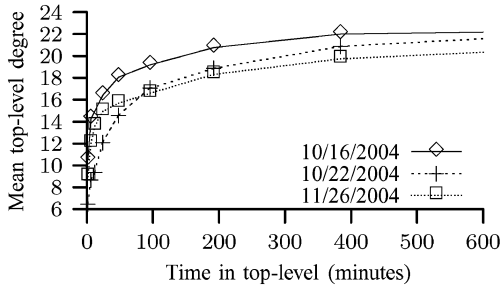


Fig. 9. Mean degree as a function of uptime. Standard deviation is large (7–13).

of peers as a function of their uptime in Fig. 9, which shows uptime and degree are correlated. The standard deviation for these measurements is quite large (around 7–13), indicating high variability. When peers first arrive, they quickly establish several connections. However, since node churn is high, they are constantly losing connections and establishing new ones. As time passes, long-lived peers gradually accumulate stable connections to other long-lived peers. We further explore this issue in Section IV when we examine overlay dynamics.

Node Degree For Leaves: To characterize properties of the two-tier topology, we have examined the degree distribution between the top-level overlay and leaves, and vice versa. Fig. 8(b) presents the degree distribution of connections from ultrapeers to leaf peers. A distinct spike at 30 is clearly visible, with a secondary spike at 45. The first two spikes are due to the corresponding parameters used in the LimeWire and BearShare implementations, respectively. This figure shows that a significant minority of ultrapeers are connected to less than 30 leaf peers, which indicates availability in the system to accommodate more leaf peers.

In Fig. 8(c), we present the degree of connectivity for leaf peers. This result reveals that most leaf peers connect to three ultrapeers or fewer (the behavior of LimeWire), a small fraction of leaves connect to several ultrapeers, and a few leaves ($<0.02\%$) connect to an extremely large number of ultrapeers (100–3000).

Implications of High Degree Peers: In all degree distributions in this subsection, we observed a few outlier peers with an unusually high degree of connectivity. The main incentive for these peers is to reduce their mean distance to other peers. To quantify the benefit of this strategy, Fig. 10(a) presents the mean distance to other peers as a function of node degree, averaged across peers with the same degree. We show this for both the top-level overlay and across all peers. This figure shows that the mean path to participating peers exponentially decreases with degree. In other words, there are steeply diminishing returns for increasing degree as a way of decreasing distance to other peers.

Turning our attention to the effects of high-degree peers on the overlay, for scoped flood-based querying, the traffic these nodes must handle is proportional to their degree for leaves and proportional to the square of their degree for ultrapeers. Note that high-degree ultrapeers may not be able, or may not choose, to route all of the traffic between their neighbors. Thus, they may not actually provide as much connectivity as they appear to, affecting the performance of the overlay.

During our analysis, we discovered around 20 ultrapeers (all on the same/24 subnet) with an extremely high degree (between 2500 to 3500) in our snapshots. These high-degree peers are widely visible throughout the overlay, and thus receive a significant portion of exchanged queries among other peers. We directly connected to these high degree peers and found they

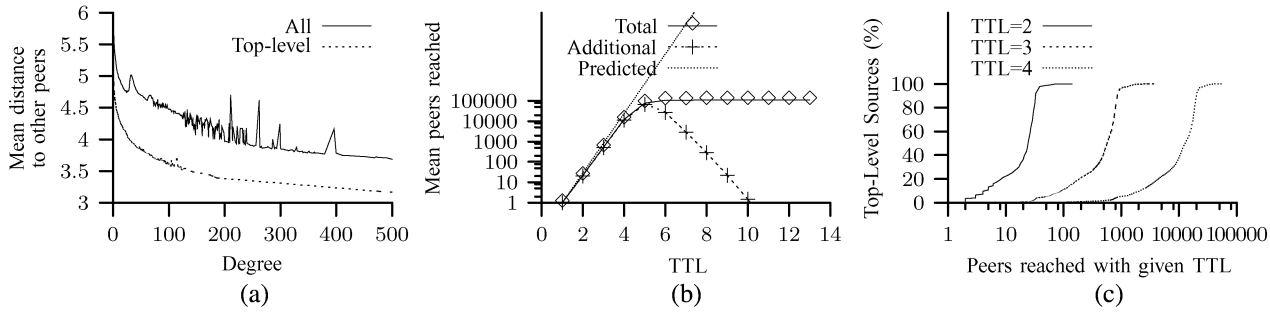


Fig. 10. Reachability, diameter, and shortest path in the Gnutella topology. (a) Correlation between ultrapeer's degree and its mean distance from other ultrapeers from the 10/18/04 snapshot; (b) mean top-level peers searched by TTL from the 9/27/2004 snapshot; (c) cumulative top-level peers searched CDF.

do not actually forward any traffic.[9] We removed these inactive high degree peers from our snapshots when considering path lengths since their presence would artificially improve the apparent connectivity of the overlay.

B. Reachability

The degree distribution suggests the overlay topology might have a low diameter, given the moderately high degree of most peers. To explore the distances between peers in more detail, we examine two equally important properties of overlay topologies that express the reachability of queries throughout the overlay: (i) the reachability of flood-based queries, and (ii) the pairwise distance between arbitrary pairs of peers.

Reachability of Flood-Based Query: Fig. 10(b) depicts the *mean* number of newly and total visited peers as a function of TTL, averaged across top-level peers in a single snapshot. The shape of this figure is similar to the result that was reported by Lv *et al.* [20, Fig. 3] which was captured in October 2000, with a significantly smaller number of peers (less than 5000). Both results indicate that the number of newly visited peers exponentially grows with increasing TTL up to a certain threshold and has diminishing returns afterwards. This illustrates that the dramatic growth of network size has been effectively balanced by the introduction of ultrapeers and an increase in node degree. Thus, while the network has changed in many ways, the percentage (but not absolute number) of newly reached peers per TTL has remained relatively stable. Fig. 10(b) also shows the number of newly visited peers predicted by the Dynamic Querying formula (assuming a node degree of 30), which we presented in Section II-A. This result indicates that the formula closely predicts the number of newly visited peers for TTL values less than 5. Beyond 5, the query has almost completely saturated the network.

Fig. 10(c) shows a different angle of reachability for the same snapshot by presenting the Cumulative Distribution Function (CDF) of the number of visited peers from top-level peers for different TTL values. This figure shows the distribution of reachability for flood-based queries among participating peers. We use a logarithmic x-scale to magnify the left part of the figure for lower TTL values. The figure illustrates two interesting points: First, the total number of visited peers using a TTL of n is almost always an order of magnitude higher compared to using a TTL of $(n-1)$. In other words, TTL is the primary determinant of the mean number of newly visited peers independent of a peer's location. Second, the distribution of newly visited peers for each TTL is not uniform among all peers. As TTL increases, this distribution becomes more skewed (considering the logarithmic scale for x axis). This is a direct effect of node degree. More specifically, if a peer or one of its neighbors has a very high degree, its flood-based query reaches a proportionally larger number of peers.

Pair-Wise Distance: Fig. 11(a) shows the distribution of shortest-path lengths in terms of overlay hops among all pairs of top-level peers from four snapshots. Ripeanu *et al.* [10] presented a similar distribution for the shortest-path length based on snapshots that were collected between November 2000 and June 2001 with 30 000 peers. Comparing these results reveals two differences: (i) the pairwise path between peers over the modern Gnutella topology is *significantly more homogeneous in length, with shorter mean value* compared with a few years ago. More specifically, the old snapshot shows 40% and 50% of all paths have a length of 4 and 5 hops whereas our results show a surprising 60% of all paths having a length of 4; and (ii) the results from our snapshots are nearly identical; whereas in [10], there is considerable variance from one crawl to another. In summary, *the path lengths have become shorter, more homogeneous, and their distribution is more stable*.

Effect of Two-Tier Topology: To examine the effect of the two-tier overlay topology on path length, we also plot the path length between all peers (including leaves) in Fig. 11(b). If each leaf had only one ultrapeer, the distribution of path length between leaves would look just like the top-level path lengths (Fig. 11(a)), but right-shifted by two. However, since each leaf peer has multiple parents, the path length distribution between leaves (and thus for all peers) has a more subtle relationship with Fig. 11(a). Comparing Figs. 11(a) and (b) shows us the cost introduced by using a two-tier overlay. In the top-level, most paths are of length 4. Among leaves, we see that around 50% of paths are of length 5 and the other 50% are of length 6. Thus, getting to and from the top-level overlay introduces an increase of 1 to 2 overlay hops.

C. Small World

Recent studies have shown that many biological and man-made graphs (e.g., collaborations among actors, the electrical grid, and the WWW graph) exhibit "small world" properties. In these graphs, the mean pairwise distance between nodes is small

[9]It appears that these peers monitor exchanged messages among other participating peers. They could be trying to locate copyright infringement among Gnutella users, collecting ratings information to measure which songs consumers might like to buy, or performing a measurement study of their own.

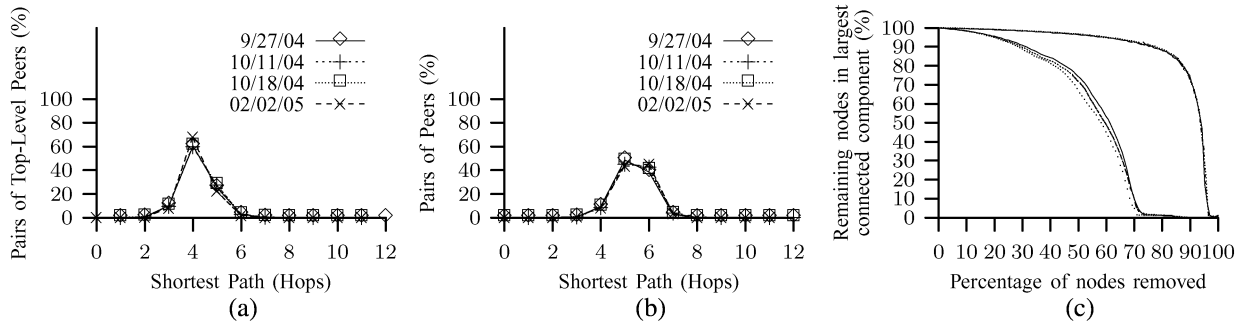


Fig. 11. Different angles on path lengths and resilience. (a) Ultrapeer-to-ultrapeer shortest paths; (b) distribution of path lengths across all pairs of peers; (c) nodes remaining in largest component as a function of nodes removed. Top and bottom lines are random and pathological removal.

TABLE III
SMALL WORLD CHARACTERISTICS

Graph	L_{actual}	L_{random}	C_{actual}	C_{random}
New Gnutella	4.17–4.23	3.75	0.018	0.00038
Old Gnutella [12]	3.30–4.42	3.66	0.02	0.002
Movie Actors [25]	3.65	2.99	0.79	0.00027
Power Grid [25]	18.7	12.4	0.08	0.005
C. Elegans [25]	2.65	2.25	0.28	0.05

and nodes are highly clustered compared to random graphs with the same number of vertices and edges. A study by Jovanovic *et al.* [12] in November—December 2000 concluded that the Gnutella network exhibits small world properties as well. Our goal is to verify to what extent recent top-level topologies of the Gnutella network still exhibit small world properties despite growth in overlay population, an increase in node degree, and changes in overlay structure. The clustering coefficient of a graph, C_{actual}, represents how frequently each node's neighbors are also neighbors, and is defined as follows [25]:

$$C(i) = \frac{D(i)}{D_{\max}(i)}, \quad C_{\text{actual}} = \frac{\sum_i C(i)}{|V|}$$

$D(i)$, $D_{\max}(i)$ and $|V|$ denote the number of edges between neighbors of node i, the maximum possible number of edges between neighbors of node i, and the number of vertices in the graph, respectively. For example, if node A has 3 neighbors, they could have at most 3 edges between them, so $D_{\max}(A) = 3$. If only two of them are connected together, that's one edge and we have $D(A) = 1$ and $C(A) = 1/3$. $C(i)$ is not defined for nodes with fewer than 2 neighbors. Thus, we simply exclude these nodes from the computation of C_{actual}. Table III presents ranges for the clustering coefficient (C_{actual}) and mean path length (L_{actual}) for the Gnutella snapshots from Table I as well as the mean values from four random graphs with the same number of vertices and edges (i.e., C_{random} and L_{random}). Because computing the true mean path lengths (L_{random}) is computationally expensive for large graphs, we used the mean of 500 sample paths selected uniformly at random. We also include the information presented by Jovanovic *et al.* [12] and three classic small world graphs [25].

A graph is loosely identified as a small world when its mean path length is close to random graphs with the same number of edge and vertices, but its clustering coefficient is orders of magnitude larger than the corresponding random graph (i.e., L_{actual} and L_{random} are close, but C_{actual} is orders of magnitude larger than C_{random}). All three classic small world graphs in the table exhibit variants of these conditions. Snapshots of modern Gnutella clearly satisfy these conditions which means that modern Gnutella still exhibits small world properties. The observed clustering could be a result of factors like peer bootstrapping, the peer discovery mechanism, and overlay dynamics. Further analysis is needed to better understand the underlying causes. Section IV shows how peer churn is one factor that contributes to clustering.

D. Resilience

We also examine the resilience in different snapshots of the Gnutella overlay topology using two different types of node removal: (i) random removal, and (ii) pathologically removing the highest-degree nodes first. An early study [13] conducted the same analysis on Gnutella based on a partial topology snapshot, finding that the overlay is resilient to random departures, but under pathological node removal quickly becomes very fragmented (after removing just 4% of nodes).

Fig. 11(c) depicts the fraction of remaining nodes in the topology which remain connected, in both the random and pathological node removal. *This figure clearly shows the Gnutella overlay is not only extremely robust to random peer removals, but it also exhibits high resilience to pathological node removal.* Even after removing 85% of peers randomly, 90% of the remaining nodes are still connected. For the pathological case, after removing the 50% of peers with the highest-degree, 75% of the remaining nodes remain connected. There are two possible factors contributing to this difference with earlier results [13]: (i) the higher median node degree of most nodes in modern Gnutella, and (ii) a non-negligible number of missing nodes and edges in the partial snapshot of the earlier study. Our result implies that complex overlay construction algorithms (e.g., [26]) are not a necessary prerequisite for ensuring resilience in unstructured overlays.

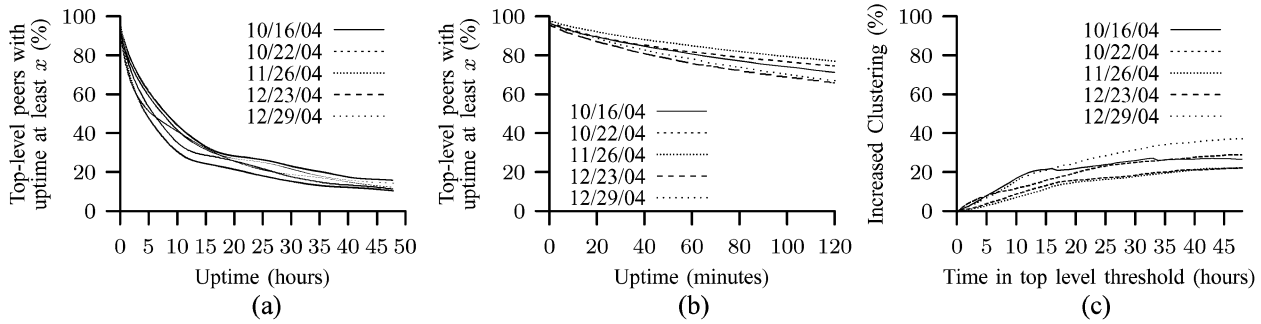


Fig. 12. Number of stable peers and their external connectivity for different τ. (a) Percentage of top-level peers with uptime at least x; (b) percentage of top-level peers with uptime at least x (zoomed in); (c) percentage of increased clustering among stable nodes, relative to a randomized topology for five different snapshots.

IV. Overlay Dynamics

In Section III, we characterized the graph-related properties of individual snapshots of the overlay topology. However, in practice the overlay topology is inherently dynamic since connections (i.e., edges) are constantly changing. These dynamics can significantly affect the main functionality of the overlay which is to provide connectivity and efficiently route the messages (e.g., queries, responses) among participating peers. Characterizing overlay dynamics enables us to examine their impact on performance of P2P applications. For example, a query or response message can be routed differently or even dropped as a result of changes in the edges of the overlay. To our knowledge, aggregate dynamics of unstructured P2P overlay have not been studied and thus these dynamics can not be incorporated in meaningful simulation-based evaluations of P2P protocols.

There are two basic causes for dynamics in the overlay topology as follows:

- Dynamics of Neighbor Selection: Two existing peers in the overlay may establish a new (or tear down an existing) connection between them. Such a change in edges is not triggered by users and thus *protocol-driven*.
- Dynamics of Peer Participation: When a peer joins (or leaves) the network, it establishes (or tears down) its connections to other participating peers in the overlay. Therefore, these changes in overlay edges are *user-driven*.[10]

Note that the user-driven dynamics of peer participation are likely to exhibit similar distributions in different P2P applications [27], [28]. Therefore, identifying the effect of user-driven dynamics on the overlay provides useful insights for the design and evaluation of other unstructured P2P overlays.

To characterize the dynamics of the Gnutella network, we investigate (i) whether a subset of participating peers form a relatively stable core for the overlay; (ii) what properties (such as size, diameter, degree of connectivity, and clustering) this stable core exhibits; and (iii) what underlying factors contribute to the formation and properties of such a stable core.

Methodology: By definition, if the overlay has a stable core, it must be composed of the long-lived ultrapeers. Short-lived peers are not stable, and leaf peers are not part of the core since they do not provide connectivity. Therefore, to identify the stable core of the overlay at any point of time, we select the subset of top-level peers who have been part of the overlay for at least τ min, i.e., whose uptime is longer than a threshold τ. We call this subset of peers the *stable peers*, or $SP(\tau)$, and only focus on this subset in our analysis. By changing τ, we can control the minimum uptime of selected peers and thus the relative stability and size of $SP(\tau)$.

To conduct this analysis, we use several slices of our dataset where each slice represents a period of 48 hours of continuous back-to-back snapshots of the overlay topology, with hundreds of snapshots per slice. We treat the last captured snapshot over each 48 hour period as a reference snapshot. Any peer in the reference snapshot must have joined the overlay either before or during our measurement period. By looking back through the snapshots, we can determine (with accuracy of a few minutes) the arrival time of all peers that joined during the measurement period. For those peers that were present for the entire measurement period, we can conclude that their uptime is at least 48 hours. Having this information, we can annotate all peers in the reference snapshot with their uptime information. Fig. 12(a) depicts the CCDF of uptime among existing peers in the reference snapshot for several slices (Fig. 12(b) presents the initial part of the same graph). In essence, this figure presents the distribution of uptime among participating peers in steady state, implying that the size of $SP(\tau)$ exponentially decreases with τ. This behavior is more visible over longer time scales. Furthermore, this also implies that the total number of possible connections within $SP(\tau)$ dramatically decreases with τ.

Internal Connectivity Within the Stable Core: To study different angles of connectivity among ultrapeers within $SP(\tau)$, we focus only on the connections of the overlay where both end points are inside $SP(\tau)$, i.e., we remove all edges to peers outside $SP(\tau)$. We call this the stable core overlay or $SC(\tau)$. The first question is: *how much connectivity is there between the peers in $SC(\tau)$?* Fig. 13(a) depicts the percentage of ultrapeers within $SC(\tau)$ that are in the largest connected component, as a function of τ. This figure demonstrates that while the fraction of connected peers slightly decreases with τ over long time scales, a significant majority (86%–94%) of peers within $SC(\tau)$ remain connected in one large component. The minor drop in the percentage of connected peers is due to the exponential decrease in number of peers within $SC(\tau)$, which in turn reduces

[10]Note that Gnutella does not run as a daemon. Therefore, peer arrival/departure is a moderately reliable indication of user action. We are mindful that dynamic IP addresses could force some peers to leave and rejoin the network with a new address. Nevertheless, we group such changes as user-driven since they are beyond the control of the P2P protocol.

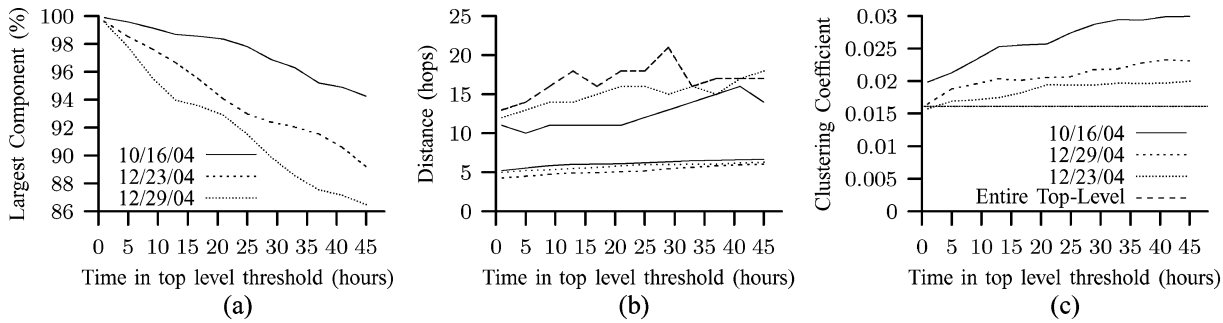


Fig. 13. Different angles of connectivity with the stable core. (a) Percentage of peers in the stable core that are part of the core's largest connect component; (b) diameter (top) and characteristic path length (bottom) of the largest connected component of the stable core; (c) clustering coefficient within the largest connected component of the stable core.

the number of edges among peers, and thus affects the opportunity for pairwise connectivity.

The second question is: *how clustered and dense is the connected portion of the core overlay?* Fig. 13(b) shows the diameter and characteristic (mean) path length among fully connected peers in the largest component of the stable core overlay. Interestingly, both the mean path length and the diameter of the stable core overlay remain relatively stable as τ increases, despite the dramatic drop in number of edges. Furthermore, the mean path length for the stable core overlay, even when it has a very small population (only 10% of top-level peers for $\tau = 45$ h), is around 5 hops, very close to the mean path length for the entire top-level overlay (4.17–4.23 from Table III). Finally, Fig. 13(c) depicts the evolution of the clustering coefficient for the stable core overlay as τ increases, along with the clustering coefficient for the entire top-level overlay in the reference snapshot. This figure shows two important points: (i) peers within the stable core overlay are more clustered together than the entire top-level overlay on average, and, more importantly, (ii) connectivity among peers within the stable core overlay becomes increasingly more clustered with τ.

External Connectivity to/from the Stable Core: To quantify the connectivity between $SC(\tau)$ and the rest of the overlay, we examined whether peers within $SC(\tau)$ have a higher tendency to connect to each other rather than peers outside the core. To quantify any potential tendency, we generate a control graph by randomizing the connections between peers. That is, given a snapshot, $G(V, E)$, we randomly generate a graph $G'(V, R)$ using the same set of peers (V) such that the degree of each peer is unchanged, i.e., $(|R_{ij}\forall j| = |E_{ij}\forall j|)\forall i$. The randomized version gives us a control for the number of edges internal to $SC(\tau)$ that arise purely as a result of the degree distribution of the graph. We can then compare the number of edges internal to $SC(\tau)$ in the snapshot with the number in the randomized version as follows:

$$\frac{|E_{ij}\forall i, j \in SC| - |R_{ij}\forall i, j \in SC|}{|R_{ij}\forall i, j \in SC|}$$

This captures the percentage increase in internal edges compared to the expected value, and is plotted as a function of τ in Fig. 12(c). The figure demonstrates that the longer a peer remains in the network, the more biased its connectivity becomes towards peers with the same or higher uptime. The characteristics of internal and external connectivities for $SC(\tau)$ imply that the longer a peer remains in the overlay, the more likely it establishes connections to peers with equal or higher uptimes, i.e., the more biased its connectivity becomes toward peers with higher uptime. Since connections for all participating peers exhibit the same behavior, connectivity of the overlay exhibits a biased "onion-like" layering where peers with similar uptime (a layer) have a tendency to be connected to peers with the same or higher uptime (internal layers of the onion). Since the size of $SP(\tau)$ decreases with τ, this means that internal layers are both smaller and more clustered.

Implications of Stable and Layered Core Overlay: The onion-like connectivity of the unstructured overlay implies that all peers within the core do not depend on peers outside the core for reachability. In other words, the core overlay provides a stable and efficient backbone for the entire top-level overlay that ensures connectivity among all participating peers despite the high rate of dynamics among peers outside the core.

A. Examining Underlying Causes

A key question is: *how does this onion-like layered connectivity form?* To address this issue, we quantify the contribution of user-driven and protocol-driven changes to the edges of the overlay. We can distinguish protocol-driven versus user-driven changes in edges between two snapshots of the overlay as follows: if at least one of the endpoints for a changing edge has arrived (or departed) between two snapshots, that change is user-driven. Otherwise, a changing edge is protocol-driven. To answer the above question, we examine a 48-hour slice of back-to-back snapshots from 10/14/2004 to 10/16/2004, using the first snapshot as a reference. Given a slice, we can detect new or missing edges in any snapshot compared to the reference snapshot, for peers in both snapshots. Let δ_{p-} and δ_{u-} (δ_{p+} and δ_{u+}) denote the percentage of missing (and new) edges in a snapshot due to protocol-driven (p) and user-driven (u) causes, relative to the reference snapshot. Note that δ_p and δ_u are by definition cumulative since the reference snapshot does not change. Fig. 14(a) and (b) depicts $\delta_- = \delta_{p-} + \delta_{u-}$ and $\delta_+ = \delta_{p+} + \delta_{u+}$. The left graph ($\delta_-$) shows that around 20% and 30% of edges in the overlay are removed due to protocol-driven and user-driven factors during the

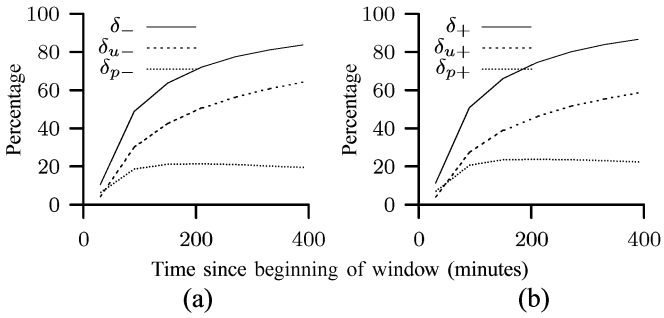


Fig. 14. Contribution of user- and protocol-driven dynamics in variations of edges in the overlay. (a) Removed edges; (b) added edges.

first 100 min, respectively. After this period, almost all removed edges are due to departing peers. Similarly, from the right graph (δ_+), many edges are added during the first 100 min due to both protocol-driven factors and the arrival of new peers. After this period, almost all new edges involve a newly arriving peer. These results show two important points: First, each peer may establish and tear down many connections to other peers during the initial 100 min of its uptime. But peers with higher uptime (i.e., peers inside $SC(\tau)$ for $\tau \geq 100$ min) maintain their connections to their remaining long-lived neighbors, and only add (or drop) connections to arriving (or departing) peers. This behavior appears to explain the formation of the biased onion-like layering in connectivity within the overlay. Second, user-driven dynamics are the dominant factor in long-term changes of the overlay. Since dynamics of peer participations exhibit rather similar characteristics in different P2P systems [27], other Gnutella-like overlays are likely to show similar behavior. We plan to conduct further investigations to better understand the underlying dynamics that contribute to this behavior.

V. Related Work

In an extension of the work of this paper, in [29] we explore long-term fluctuations in the structure of the Gnutella overlay topology over a 1.5 year period, as well as examine correlations between the overlay structure and the geographic location of peers. In [30], we extend Cruiser to capture the list of files shared by each peer and characterize the static, topological, and dynamic properties of available files in Gnutella.

As listed throughout this paper, there are a handful of prior studies which also perform a case study of Gnutella to characterize peer-to-peer overlay topologies in file-sharing applications [10], [12], [20], [24]. These studies are more than three years old, do not verify the accuracy of their captured snapshots, and conduct only limited analysis. A recent study [14] uses both passive measurement and active probing of 900 super nodes to study behavior of the Kaaza overlay. It mostly focuses on the number of observed connections (within the top-level overlay and from the top-level overlay to leaf nodes) and their evolution with time. However it does not examine detailed graph-related properties of the overlay, or the collective dynamics of the entire overlay topology, both of which are investigated in this paper.

There are a wealth of measurement studies on other properties of peer-to-peer systems. These studies cover several topics: (i) file characteristics [31]–[34]; (ii) transfer characteristics [32], [35]; (iii) peer characteristics [13], [23]; (vi) query characteristics [33], [36]–[38]; and (v) comparisons of different implementations [39], [40]. Since they explore different aspects of peer-to-peer networks, these studies complement our work. There are also several modeling and simulation-based studies on the improvement of search in Gnutella-like P2P networks [6]–[9]. Our characterizations can be directly used by these studies as a reference for comparison of suggested topology models, and our captured overlay snapshots can be used for trace-driven simulation of their proposed search mechanisms.

Finally, the research studies on characterization of the Internet topology (e.g., [41]) and network topology generators (e.g., [42]) are closely related to our work. However, these studies focus on the Internet topology rather than an overlay topology. We plan to conduct further characterization of the Gnutella topology by applying some of the suggested graph analysis in these studies to the Gnutella overlay topology.

VI. Conclusion

In this paper, we presented Cruiser, a crawler for rapidly capturing accurate snapshots of P2P overlay topologies. We showed how a crawler that is too slow may introduce significant measurement error, introduce techniques for measuring the accuracy of a crawler, and found that inadequate crawler speed may have been responsible for some conclusions in prior work such as a power-law degree distribution.

Using Gnutella, we presented the first detailed characterization of an unstructured two-tier overlay topology that is typical of modern popular P2P systems, based on accurate and complete snapshots captured with Cruiser. We characterized the graph-related properties of individual snapshots, the dynamics of the overlay topology across different time scales, and investigated the underlying causes and implications. Our main findings are summarized in Section I-A.

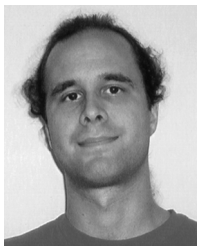
This study developed essential insights into the behavior of overlay topologies which are necessary to improve the design and evaluation of peer-to-peer file-sharing applications. The existence of a stable well-connected core of long-lived peers suggests that there may be benefits in terms of increasing search resilience in the face of overlay dynamics, by biasing/directing the search towards longer lived peers and therefore towards this core. It may also be useful to cache indexes or content at long-lived peers in order to reduce load on the stable core, especially if the biased forwarding of queries is adopted. For example, the idea of one-hop replication [43], intended for power-law topologies, can be changed to a probabilistic one-hop replication biased towards peers with longer uptime.

We are continuing this work in a number of directions. To complement Cruiser's approach of capturing the entire network, we are also developing sampling techniques for peer-to-peer networks that may be used when capturing the entire network is infeasible. We are also gathering data from Gnutella, Kad, and BitTorrent to conduct a detailed characterization of the dynamics of peer participation (or churn) to develop useful models for use in simulation and analysis.

References

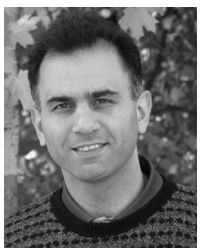
[1] P2P Networks [Online]. Available: http://www.slyck.com/ 2006
[2] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos, "Is P2P dying or just hiding?," in *Proc. IEEE Globecom*, Dallas, TX, Nov. 2004.

[3] T. Karagiannis, A. Broido, M. Faloutsos, and K. C. Claffy, "Transport layer identification of P2P traffic," in *Proc. Internet Measurement Conf.*, Taormina, Italy, Oct. 2004.
[4] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup protocol for Internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, Feb. 2003.
[5] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proc. ACM SIGCOMM*, 2001.
[6] Y. Chawathe, S. Ratnasamy, and L. Breslau, "Making Gnutella-like P2P systems scalable," in *Proc. ACM SIGCOMM*, 2003.
[7] B. Yang, P. Vinograd, and H. Garcia-Molina, "Evaluating GUESS and non-forwarding peer-to-peer search," in *IEEE Int. Conf. Distributed Systems*, 2004.
[8] B. Yang and H. Garcia-Molina, "Designing a super-peer network," in *Int. Conf. Data Engineering*, Mar. 2003.
[9] K. Sripanidkulchai, B. Maggs, and H. Zhang, "Efficient content location using interest-based locality in peer-to-peer systems," in *Proc. IEEE INFOCOM*, 2003.
[10] M. Ripeanu, I. Foster, and A. Iamnitchi, "Mapping the Gnutella network: Properties of large-scale peer-to-peer systems and implications for system design," *IEEE Internet Computing J.*, vol. 6, no. 1, 2002.
[11] Gnutella: To the Bandwidth Barrier and Beyond. clip2.com, Nov. 2000.
[12] M. Jovanovic, F. Annexstein, and K. Berman, "Modeling peer-to-peer network topologies through "Small-World" models and power laws," in *Proc. TELFOR*, Nov. 2001.
[13] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "Measuring and analyzing the characteristics of napster and Gnutella hosts," *Multimedia Syst. J.*, vol. 9, no. 2, pp. 170–184, Aug. 2003.
[14] J. Liang, R. Kumar, and K. W. Ross, "The KaZaA overlay: A measurement study," *Comput. Netw. J. (Elsevier)*, 2005.
[15] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "Sampling techniques for large, dynamic graphs," in *Proc. Global Internet Symp.*, Barcelona, Spain, Apr. 2006.
[16] D. Stutzbach and R. Rejaie, "Capturing accurate snapshots of the Gnutella network," in *Proc. Global Internet Symp.*, Miami, FL, Mar. 2005, pp. 127–132.
[17] D. Stutzbach and R. Rejaie, "Characterizing the two-tier Gnutella topology," in *ACM SIGMETRICS*, Banff, AB, Canada, Jun. 2005, Extended Abstract.
[18] D. Stutzbach, R. Rejaie, and S. Sen, "Characterizing unstructured overlay topologies in modern P2P file-sharing systems," in *Proc. Internet Measurement Conf.*, Berkeley, CA, Oct. 2005, pp. 49–62.
[19] D. Stutzbach and R. Rejaie, "Evaluating the accuracy of captured snapshots by peer-to-peer crawlers," in *Passive and Active Measurement Workshop*, Boston, MA, Mar. 2005, pp. 353–357, Extended Abstract.
[20] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Int. Conf. Supercomputing*, 2002.
[21] A. Singla and C. Rohrs, "Ultrapeers: Another step towards Gnutella scalability," in *Gnutella Developer's Forum*, Nov. 2002.
[22] A. Fisk, "Gnutella dynamic query protocol v0.1," in *Gnutella Developer's Forum*, May 2003.
[23] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 2, pp. 219–232, Apr. 2004.
[24] L. A. Adamic, R. M. Lukose, B. Huberman, and A. R. Puniyani, "Search in power-law networks," *Phys. Rev. E*, vol. 64, no. 46135, 2001.
[25] D. J. Watts, "Six degrees," in *The Essence of a Connected Edge*. New York: ACM Press, 2003.
[26] R. H. Wouhaybi and A. T. Campbell, "Phenix: Supporting resilient low-diameter peer-to-peer topologies," in *Proc. IEEE INFOCOM*, 2004.
[27] D. Stutzbach and R. Rejaie, "Characterizing Churn in peer-to-peer networks," Univ. of Oregon, Eugene, OR, Tech. Rep. CIS-TR-2005-03, May 2005.
[28] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the Internet," in *Internet Measurement Conf.*, Taormina, Italy, Oct. 2004.
[29] A. Rasti, D. Stutzbach, and R. Rejaie, "On the long-term evolution of the two-tier Gnutella overlay," in *Proc. Global Internet Conf.*, Barcelona, Spain, Apr. 2006.
[30] S. Zhao, D. Stutzbach, and R. Rejaie, "Characterizing files in the modern Gnutella network: A measurement study," in *Proc. Multimedia Computing and Networking Conf.*, San Jose, CA, Jan. 2006.
[31] J. Chu, K. Labonte, and B. N. Levine, "Availability and locality measurements of peer-to-peer file systems," in *ITCom: Scalability and Traffic Control in IP Networks II Conf.*, Jul. 2002.
[32] N. Leibowitz, M. Ripeanu, and A. Wierzbicki, "Deconstructing the Kazaa network," in *Proc. WIAPP*, 2003.
[33] E. Adar and B. A. Huberman, "Free riding on Gnutella," *First Monday*, vol. 5, no. 10, Oct. 2000.
[34] J. Liang, R. Kumar, Y. Xi, and K. W. Ross, "Pollution in P2P file sharing systems," in *Proc. IEEE INFOCOM*, Miami, FL, Mar. 2005.
[35] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," in *Proc. SOSP*, 2003.
[36] K. Sripanidkulchai, "The popularity of Gnutella queries and its implications on scalability," Jan. 2001 [Online]. Available: http://www-2.cs.cmu.edu/~kunwadee/research/p2p/paper.html
[37] A. Klemm, C. Lindemann, M. Vernon, and O. P. Waldhorst, "Characterizing the query behavior in peer-to-peer file sharing systems," in *Internet Measurement Conf.*, Taormina, Italy, Oct. 2004.
[38] F. S. Annexstein, K. A. Berman, and M. A. Jovanovic, "Latency effects on reachability in large-scale peer-to-peer networks," in *Proc. Symp. Parallel Algorithms and Architectures*, Crete, Greece, 2001, pp. 84–92.
[39] P. Karbhari, M. Ammar, A. Dhamdhere, H. Raj, G. Riley, and E. Zegura, "Bootstrapping in Gnutella: A measurement study," in *Proc. PAM*, Apr. 2004.
[40] Q. He and M. Ammar, "Congestion control and message loss in Gnutella networks," in *Multimedia Computing and Networking Conf.*, Santa Clara, CA, Jan. 2004.
[41] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology," in *ACM SIGCOMM*, 1999.
[42] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, "Network topology generators: Degree-based versus structural," in *ACM SIGCOMM*, 2002.
[43] Q. Lv, S. Ratnasamy, and S. Shenker, "Can heterogeneity make Gnutella scalable?," in *IPTPS*, 2002.



Daniel Stutzbach (M'06) received the B.S. degree in electrical engineering from Worcester Polytechnic Institute in 1998 and the Ph.D. degree in CIS from the University of Oregon in 2006.

He recently founded Stutzbach Enterprises, LLC, which offers consulting services related to his doctoral research. His interests include peer-to-peer networking and network measurement.



Reza Rejaie (SM'06) received the Ph.D. and M.S. degrees from the University of Southern California, Los Angeles, in 1999 and 1996, and the B.S. degree from the Sharif University of Technology, Tehran, Iran, in 1991.

He is currently an Assistant Professor at the University of Oregon, Eugene, OR. From 1999 to 2002, he was a Senior Technical Staff member at AT&T Labs–Research, Menlo Park, CA. His research interests include peer-to-peer streaming, network measurement, congestion control and multimedia networking.

Dr. Rejaie received an NSF CAREER Award for his work on P2P streaming in 2005. He is also a senior member of the ACM.

Subhabrata Sen (M'01) received the B.Eng. degree in computer science in 1992 from Jadavpur University, India, and the M.S. and Ph.D. degrees in computer science from the University of Massachusetts, Amherst, in 1997 and 2001, respectively.

He is currently a Principal Technical Staff Member at the Internet and Networking Systems Research Center at AT&T Labs–Research, Florham Park, NJ. His research interests include network traffic measurement, characterization and classification, network data mining, security and anomaly detection, peer-to-peer systems, and end-to-end support for streaming multimedia.