Multimedia Systems manuscript No.
(will be inserted by the editor)

Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts

Stefan Saroiu, P. Krishna Gummadi, Steven D. Gribble

Department of Computer Science and Engineering, University of Washington, Seattle, WA, 98195-2350

Received: March 31st, 2002 / Revised version: October 14th, 2002

Abstract The popularity of peer-to-peer multimedia file sharing applications such as Gnutella and Napster has created a flurry of recent research activity into peer-to-peer architectures. We believe that the proper evaluation of a peer-to-peer system must take into account the characteristics of the peers that choose to participate in it. Surprisingly, however, few of the peer-to-peer architectures currently being developed are evaluated with respect to such considerations. In this paper, we remedy this situation by performing a detailed measurement study of the two popular peer-to-peer file sharing systems, namely Napster and Gnutella. In particular, our measurement study seeks to characterize the population of end-user hosts that participate in these two systems. This characterization includes the bottleneck bandwidths between these hosts and the Internet at large, IP-level latencies to send packets to these hosts, how often hosts connect and disconnect from the system, how many files hosts share and download, the degree of cooperation between the hosts, and several correlations between these characteristics. Our measurements show that there is significant heterogeneity and lack of cooperation across peers participating in these systems.

1 Introduction

The popularity of peer-to-peer file sharing applications such as Gnutella and Napster has created a flurry of recent research activity into peer-to-peer architectures [7,11,20,25–27]. Although the exact definition of "peer-to-peer" is debatable, these systems typically lack dedicated, centralized infrastructure, but rather depend on the voluntary participation of peers to contribute the resources from which the infrastructure is constructed. Membership in a peer-to-peer system is ad-hoc and dynamic: as such, the challenge of such systems is to figure out a mechanism and architecture for organizing the peers in a way that they can cooperate to provide a useful service to the community of users. For example, in a file sharing application, one challenge is organizing peers into a cooperative, global index so that all content can be quickly and efficiently located by any peer in the system [11,20,25,27].

In order to evaluate a proposed peer-to-peer system, the characteristics of the peers that choose to participate in the system must be understood and taken into account. For example, if some peers in a file-sharing system have low-bandwidth, high-latency network connections to the Internet, the system must be careful to avoid delegating large or popular portions of the distributed index to those peers, for fear of overwhelming them and making that portion of the index unavailable to other peers. Similarly, the typical duration that peers choose to remain connected to the infrastructure has implications for the degree of redundancy necessary to keep data or index metadata highly available. In short, the system must take into account the suitability of a given peer for a specific task before explicitly or implicitly delegating that task to the peer.

Surprisingly, however, few of the architectures currently being developed are evaluated with respect to such considerations. We believe that this is, in part, due to a lack of information about the characteristics of hosts that choose to participate in peer-to-peer systems. We are aware of a single previous study [2] that measures only one such characteristic, namely the number of files peers share. In this paper, we remedy this situation by performing a detailed measurement study of the two most popular peer-to-peer file sharing systems, namely Napster and Gnutella. The hosts that choose to participate in these systems are typically end-users' home or office machines, located at the "edge" of the Internet.

Our measurement study seeks to precisely characterize the population of end-user hosts that participate in these two systems. This characterization includes the bottleneck bandwidths between these hosts and the Internet at large, IP-level latencies to send packets to these hosts, how often hosts connect and disconnect from the system, how many files hosts share and download, and correlations between these characteristics. Our measurements consist of detailed traces of these two systems gathered over long periods of time — four days for Napster and eight days for Gnutella respectively.

There are two main lessons to be learned from our measurement results. First, there is a significant amount of heterogeneity in both Gnutella and Napster; bandwidth, latency, availability, and the degree of sharing vary between three and five orders of magnitude across the peers in the system. This implies that any similar peer-to-peer system must be very careful about delegating responsibilities across peers. Second, peers tend to deliberately misreport information if there is an incentive to do so. Because effective delegation of responsibility depends on accurate information, this implies that future systems must have built-in incentives for peers to tell the truth, or systems must be able to directly measure or verify reported information.

The rest of the paper is structured as follows. Section 2 deals with our measurement technology. Section 2.1 describes the architectures of Napster and Gnutella. Section 2.2 presents the techniques used to crawl these systems. Section 2.3 discusses the active measurement tools used to probe the characteristics of the peers discovered. Our measurement results are described in Section 3. Section 4 contains a brief discussion of our results and several recommendations for future file sharing peer-to-peer system designs. Finally, our conclusions are presented in Section 5.

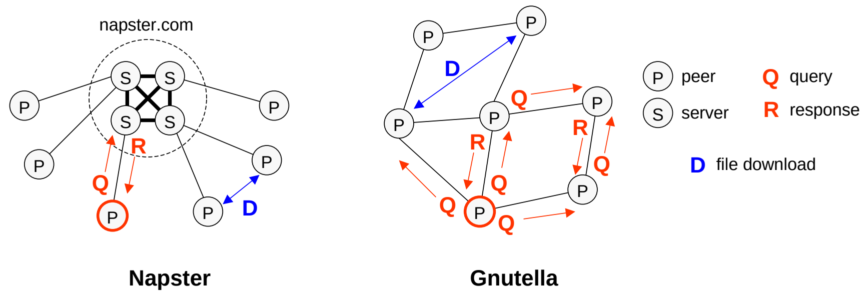


Fig. 1 Locating files in Napster and Gnutella. In Napster, each peer directly queries a central server to discover the location of files. In Gnutella, peers form an overlay network over which file queries are broadcast. In both systems, once a peer discovers the location of a file, that peer downloads the file using a direct TCP connection to the peer that hosts the file.

2 Measurement Methodology

To collect our measurements of Napster and Gnutella, we periodically *crawled* each system in order to gather snapshots the systems' populations. The information collected in these snapshots includes the IP address and port number of each peer in the population, as well as some information about the peers as reported by their software. In this section on the paper, we describe the architectures of Napster and Gnutella, the techniques used to crawl these systems and the tools we used to actively probe the characteristics of the peers. We also reflect on the limitations of our methodology.

2.1 The Napster and Gnutella Architectures

Both Napster and Gnutella have similar goals: to facilitate the location and exchange of files (typically images, audio, or video) among a large group of independent users connected through the Internet. In these systems, files are stored on the computers of the individual users (or *peers*), and exchanged using an HTTP-style protocol over a direct connection between the downloading and uploading peers. All peers in these systems are symmetric: they all have the ability to function both as a client and a server. This symmetry distinguishes peer-to-peer systems from many conventional distributed system architectures. Although the process of exchanging files is similar in both systems, Napster and Gnutella differ substantially in how peers locate files (Figure 1).

In Napster, a large cluster of dedicated central servers maintains an index of the files that are currently being shared by active peers. Each peer maintains a persistent connection to one of the central servers, through which the file location queries are sent. The servers then cooperate to process the query and return a list of matching files and their locations to the peer. After receiving the results, the peer may then select one or more files and locations from this list and initiate file exchanges directly from other peers. In addition to maintaining an index of shared files, the centralized servers also monitor the state of each peer in the system, keeping track of metadata such as the peers' reported connection bandwidth and the duration that the peer has remained connected to the system. This metadata is returned with the results of a query, so that the initiating peer has some information to distinguish possible download sites.

There are no centralized servers in Gnutella, however. Instead, Gnutella peers form an *overlay network* by forging point-to-point connections with a set of neighbors. To locate a file, a peer initiates a controlled flood of the network by sending a query packet to all of its neighbors. Upon receiving a query packet, a peer checks if any locally stored files match the query. If so, the peer sends a query response packet back towards the query originator through the overlay. Whether or not a file match is found, the peer continues to flood the query through the overlay.

To help maintain the overlay as the users enter and leave the system, the Gnutella protocol includes *ping* and *pong* messages that help peers to discover other nodes. Pings and pongs behave similarly to query/query-response packets: any peer that sees a ping message sends a pong back towards the originator, and forwards the ping onwards to its own set of neighbors. Ping and query packets thus flood through the network; the scope of flooding is controlled with a time-to-live (TTL) field that is decremented on each hop. Peers occasionally forge new neighbor connections with other peers discovered through the ping/pong mechanism. Note that it is possible to have several disjoint Gnutella overlays of Gnutella simultaneously coexisting in the Internet; this contrasts with Napster, in which peers are always connected to the same cluster of central servers.

2.2 Crawling the Peer-to-Peer Systems

In this section of the paper, we describe the design and implementation of our Napster and Gnutella crawlers. The goal of these crawlers is to produce "snapshots" of these systems, by collecting large sets of participating peers.

2.2.1 The Napster Crawler Because we did not have direct access to indexes maintained by the central Napster servers, the only way we could discover the set of peers participating in the system at any given time was by issuing queries for files, and keeping a list of peers referenced in the queries' responses. To discover the largest possible set of peers, we issued queries with the names of popular song artists drawn from a long list downloaded from the web.

Based on our experience and observations, the Napster server cluster consists of approximately 160 servers; each peer establishes a connection with only one server. When a peer issues a query, the server the peer is connected to first reports files shared by "local users" on the same server, and later reports matching files shared by "remote users" on other servers in the cluster. For each crawl, we established a large number of connections to a single server, and issued many queries in parallel; this reduced the amount of time taken to gather data to 3-4 minutes per crawl, giving us a nearly instantaneous snapshot of peers connected to that server. For each peer that we discovered during the crawl, we then queried the Napster server to gather the following metadata: (1) the bandwidth of the peer's connection as reported by the peer herself, (2) the number of files currently being shared by the peer, (3) the current number of uploads and the number of downloads in progress by the peer, (4) the names and sizes of all the files being shared by the peer, and (5) the IP address of the peer.

The Napster protocol indicates which peers connect to the same server as our crawler (local peers) and which peers connect to other Napster servers (remote peers). To get an estimate of the fraction of the total user population we captured, we separated the local and remote peers returned in our queries' responses, and compared them to statistics periodically broadcast by the particular Napster server that we queried. From these statistics, we verified that each crawl typically captured between 40% and 60% of the local peers on the crawled server. Furthermore, this 40-60% of the peers that we captured contributed between 80-95% of the total (local) files reported to the server.

Our crawler did not capture any peers that do not share any of the popular content in our queries. This introduces a bias in our results, particularly in our measurements that report the number of files being shared by users. However, the statistics reported by Napster revealed that the distributions of number of uploads, number of downloads, number of files shared, and bandwidths reported for all remote users were quite similar to those that we observed from our captured local users. These statistics are reported by regular Napster file-sharing software.

2.2.2 The Gnutella Crawler The goal of our Gnutella crawler is the same as our Napster crawler: to gather nearly instantaneous snapshots of a significant subset of the Gnutella population, as well as metadata about peers in captured subset as reported by the Gnutella system itself. Our crawler exploits the ping/pong messages in the protocol to discover hosts. First, the crawler connects to several well-known, popular peers (such as `gnutellahosts.com` or `router.limewire.com`). Then, it begins an iterative process of sending ping messages with large TTLs to known peers, adding newly discovered peers to its list of known peers based on the contents of received pong messages. In addition to the IP address of a peer, each pong message contains metadata about the peer, including the number and total size of files being shared.

We allowed our crawler to continue iterating for approximately two minutes, after which it would typically gather between 8,000 and 10,000 unique peers (Figure 2). According to measurements reported by Clip2 [8] at the time that we gathered our results, this corresponds to at least 25% to 50% of the total population of peers in the system at any time. After two minutes, we would terminate the crawler, save the crawling results to a file and begin another crawl iteration to gather our next snapshot of the Gnutella population.

Unlike our Napster measurements, in which we were more likely to capture hosts sharing popular songs, we have no reason to suspect any bias in our measurements of the Gnutella user population. Furthermore,

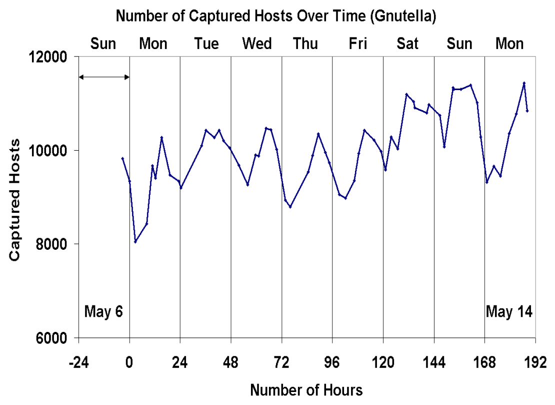


Fig. 2 The number of Gnutella hosts captured by our crawler over time. A diurnal cycle is obviously present; the number of captured hosts peaked between 3pm PST to 7pm PST during weekdays. Our crawler typically observed 10% more hosts during the weekends.

to ensure that the crawling process does not alter the behavior of the system in any way, our crawler neither forwarded any Gnutella protocol messages nor answered any queries.

2.2.3 Crawler Statistics Both the Napster and Gnutella crawlers were written in Java, and ran using the IBM Java 1.18 JRE on Linux 2.2.16. The crawlers ran in parallel on a small number of dual-processor Pentium III 700 MHz computers with 2GB RAM, and four 40GB SCSI disks. Our Napster trace captured four days of activity, from Sunday May 6th, 2001 through Wednesday May 9th, 2001[1]. We recorded a total of 509,538 Napster peers on 546,401 unique IP addresses. Our Gnutella trace spanned eight days (Sunday May 6th, 2001 through Monday May 14th, 2001) and captured 1,239,487 Gnutella peers on 1,180,205 unique IP-addresses.

2.3 Active Measurements

For each gathered snapshot of Napster and Gnutella, we also performed various direct measurements of additional properties of the peers. Our goal was to capture data that would enable us to reason about the fundamental characteristics of the peers (both as individuals and as a population) participating in any peer-to-peer file sharing system. The data collected includes the distributions of bottleneck bandwidths and latencies between peers and our measurement infrastructure, the number of shared files per peer, the distribution of peers across DNS domains, and the "lifetime" of the peers in the system (i.e., how frequently peers connect to the systems and how long they remain connected).

Unfortunately, in some cases, we could not reuse current network measurement tools due to their unscalability and slow speeds. Instead, we developed ways to incorporate existing measurement techniques into

[1] During this time, Napster was at the peak of its popularity.

new tools that are more appropriate to the scale of our project. In this section, we describe our techniques and tools used to probe peers in order to measure their bandwidths, latencies and availabilities. In order to distinguish the direction of probing traffic sent to a remote host, we will use “upstream” to refer to traffic from the remote host to the local host, and “downstream” to traffic from the local host to the remote host.

2.3.1 Bottleneck Bandwidth Measurements One characteristic that we wanted to gather was the speed of peers' connections to the Internet. This is not a precisely defined concept: the rate at which content can be transferred between two peers depends on the bottleneck bandwidth on the path between the two peers, the available bandwidth along the path, and the latency between the peers.

The central Napster servers can provide the connection bandwidth of any peer as reported by the peer itself. However, as we will show later, a substantial percentage of the Napster peers (as high as 25%) choose not to report their bandwidths. Furthermore, there is a clear incentive for a peer to discourage other peers from downloading files by falsely reporting a low bandwidth. The same incentive to lie exists in Gnutella; in addition to this, in Gnutella, bandwidth is reported only as part of a successful response to a query, so peers that share no data or whose content does not match any queries never report their bandwidths.

Because of this, we decided to actively probe the bandwidths of peers. There are two difficult problems with measuring the bandwidth to and from a large number of hosts: first, even though available bandwidth is the most accurate indicator of the speed at which downloads will proceed, available bandwidth can significantly fluctuate over short periods of time. Second, available bandwidth is determined by measuring the loss rate of an open TCP connection, which is expensive to measure in practice. Instead, we decided to use the bottleneck link bandwidth as a first-order approximation to the available bandwidth; because our workstations are connected by a gigabit link to the Abilene network, it is likely that the bottleneck link between our workstations and any peer in these systems is last-hop link to the peer itself. This is particularly likely since, as we will show later, most peers are connected to the system use low-speed modems or broadband connections such as cable modems or DSL. Thus, if we could characterize the bottleneck bandwidth between our measurement infrastructure and the peers, we would have a fairly accurate upper bound on the rate at which information could be downloaded from these peers.

Bottleneck link bandwidth between two different hosts equals the capacity of the *slowest* hop along the path between the two hosts. Thus, by definition, bottleneck link bandwidth is a physical property of the network that remains constant over time for an individual path.

Estimating bottleneck bandwidth is not a new area of research [12,14,4,6,19,16,10]. Broadly speaking, two estimation techniques exist: (1) “one-packet” – performing a controlled flood of the measured network path or (2) “packet-pair” – use the difference in the arrival times of a large packet pair traversing the measured network path. Although various bottleneck link bandwidth measurement tools are available [13, 18,5,10,15], for a number of reasons, all of these tools were unsatisfactory for our purposes.

To be useful given the scale and goal of our measurements, a bottleneck bandwidth estimation tool must have several properties: it must be accurate, it must produce estimates quickly, it must have the ability to work in uncooperative environments, and it must be able to scale to measuring tens of thousands of hosts over a relatively short period of time. Fast measurements are essential when dealing with multiple network paths and peers; spending a minute on a single measurement can make a tool too slow to be valuable in practice.[2] In measurements dealing with tens of thousands of hosts, deploying software on all endhosts is impossible in practice; even a low degree of cooperation, such as having enough data to transfer between the endhosts, is sometimes an infeasible assumption to make. A tool should be able to cope with tens of thousands of network paths and Internet hosts, implying that tools that need to flood the network to make a single measurement simply will not scale. None of the existing tools possesses all of these properties; in the next section of this paper, we present the design of SProbe [22], our packet-pair bottleneck bandwidth measurement tool that does achieve all of these properties.

2.3.2 Measuring Bottleneck Bandwidth in the Downstream Direction To measure downstream bottleneck bandwidth, a large packet pair needs to traverse the path from the local to the remote host. The difference in the packets' arrival times at the remote host can be used directly to estimate the downstream bottleneck bandwidth of the network path [16]. In general, the larger the difference, the smaller the bottleneck bandwidth. Because we cannot force peers in either system to cooperate explicitly with our measurements, in practice it is impossible for us to discover the time dispersion of the packet pair when it arrives at the remote host. However, in certain cases, we can cause the remote host to respond to the probe packets. Assuming that the packets sent as responses do not queue at the bottleneck bandwidth on the reverse path, their time dispersion can be used as an approximation to the initial, large packet pair time dispersion.

SProbe [22] relies on an exploitation of the TCP protocol, similar to the ones described in [24], to perform the sequence of packet exchanges described above. In the TCP protocol, a SYN packet pair sent to an inactive port of the remote machine is answered by a RST packet pair. Regular SYN packets are 40-byte packets, having no payload data. SProbe appends a large payload (1460 bytes) to each sent SYN packet. Since the answered RST packets are small packets (40 bytes), they are unlikely to queue at the bottleneck link on the reverse path, and, therefore, their time dispersion can be used as an approximation to the remote host time dispersion of the large SYNs packet pair. Note that this packet exchange only relies on a correct TCP implementation at the remote host and is sufficient to measure the downstream bottleneck bandwidth. Figure 3 illustrates the packet exchange initiated by SProbe.

On certain network links (such as modem lines), packets are compressed before traversing the link. This effect can alter packet size and impact the accuracy of the packet pair measurement. Thus, SProbe appends a pre-compressed payload to the SYN packets to minimize the compression effects on the packet size. We

[2] A minute per measurement translates to spending a week for about 10,000 sequential measurements.

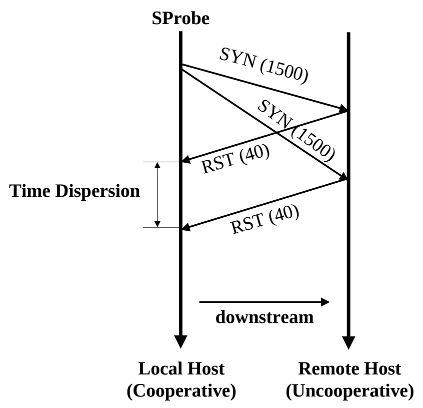


Fig. 3 The packet exchange initiated by SProbe to estimate the bottleneck bandwidth in the downstream direction. SProbe uses the interarrival latency of the RST packets as an estimate of the dispersion between the SYN packets upon arrival at the uncooperative host.

are not aware of any other tool that explicitly deals with packet compression. Finally, although the current implementation of SProbe sends SYN packets, the same measurement sequence can be achieved using FIN, ACK or DATA packets.

Our technique, however, has several limitations. Because firewalls silently drop SYN packets to inactive ports, SProbe cannot distinguish hosts behind firewalls from offline hosts. When packet loss occurs, SProbe will timeout after five seconds and terminate with an *unanswered* message. It is up to the user or the application invoking SProbe to decide whether to retry the measurement.

2.3.3 Dealing with Cross Traffic, Reordering and Multi-Channel Links In general, the accuracy of packet pair measurements can degrade due to cross traffic interference. A rogue packet queued at the bottleneck link between the packet pair invalidates the measurement. Current packet pair tools [13,18,5,10] deal with cross traffic by sending a large amount of probe packet pairs and gathering a large number of measurements. Assuming that cross traffic interference does not dominate the measurements, statistical algorithms are used to extract the average, free-of-cross-traffic case. Different statistical algorithms have been proposed to extract accurate estimates from sets of gathered data points [6,17,10].

Unfortunately, these approaches have two inherent drawbacks: the tools are *difficult to scale*, and the tools are *slow*. Instead, SProbe relies on a small amount of extra probe traffic to remain resilient in the face of these adverse effects. SProbe shapes its probe traffic to reveal information about the conditions of the measured network path. It sends a train of small, 40-byte SYN packets as an “envelope” surrounding the large packet pair (1500-byte). The remote endhost responds with a same size train of RST packets. SProbe analyzes the time dispersion of the packets in the train and uses two heuristics that determine the presence of cross traffic, reordering or multi-channel links. SProbe currently uses a train of six SYN packets, two of which constitute the large packet pair.

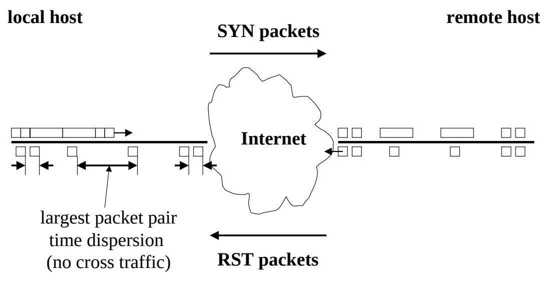


Fig. 4 The consistent arrival times heuristic test. The time dispersion of the RST packets corresponding to the large SYN packet pair (the middle packets) should be the largest dispersion among packet pairs when no cross traffic is present.

The Shuffle Heuristic Test: SProbe uses the sequence number of the received RST packets to determine whether the ordering of the received packets has been preserved relative to the ordering of the sent SYN packets. Packet reordering can be an indication that rogues packets interfered with the interarrival times of the probing packet pair. When packets were reordered, SProbe discards the measurement and returns an *unknown* estimate. Note that a multi-channel link is also likely to reorder the train packets.

The Consistent Arrival Times Heuristic Test: When the received RSTs are not reordered, the time dispersion of the two RSTs in the middle of the train should be larger than the dispersion of any of the smaller 40-byte packet pairs. If not, it is likely that a rogue packet was placed between 40-byte packet pair. This indicates cross-traffic presence during probing; SProbe discards the measurement and returns an *unknown* estimate. Figure 4 illustrates the packet train exchange and the consistent arrival times heuristic test.

2.3.4 Measuring Bottleneck Bandwidth in the Upstream Direction Unlike the downstream direction, where virtually no cooperation is required from the remote host, the upstream direction requires a low degree of cooperation, although no measurement software needs to be deployed. SProbe uses a TCP connection to produce its estimate; during slow-start, the TCP protocol consecutively sends a number of packets equal to its congestion window. Whenever these packets are large, they can serve as probing packets to estimate the upstream bottleneck bandwidth.

In the case of a Gnutella peer, SProbe initiates a TCP connection, and performs the Gnutella protocol's initial handshake. Once the handshake is complete, the corresponding peer will start forwarding all packets carrying Gnutella traffic to the SProbe host. Unfortunately, most of these packets are small in size, and therefore cannot be used to obtain a reliable estimate of bottleneck bandwidth. Instead, immediately after completing the Gnutella handshake, SProbe will signal the other host not to send any data for a while, by advertising a TCP window size of 0 bytes. Under normal TCP semantics, advertising a 0 byte window

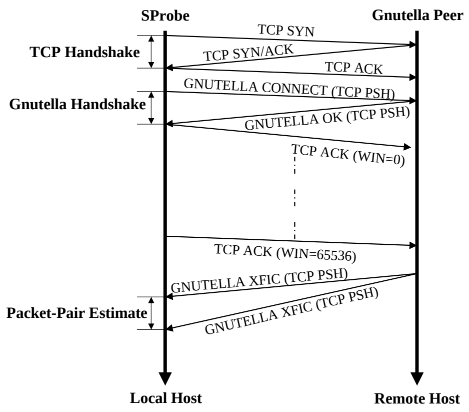


Fig. 5 The packet exchange initiated by SProbe to estimate the bottleneck bandwidth in the upstream direction. SProbe closes its receiver-advertised TCP window as a mechanism for ensuring that the remote host has enough data to fill two large, back-to-back TCP packets.

signals the unavailability of any buffers to the TCP stack and causes the remote hosts to postpone sending any data. In the meantime, the remote host accumulates an increasing amount of Gnutella traffic in its TCP buffers, waiting for SProbe to advertise a non-null window size. After waiting for about ten round-trip times (RTTs), SProbe advertises a large TCP window, causing the remote host to immediately send at least two back-to-back large TCP packets, containing several Gnutella protocol messages that were accumulated in the meantime. SProbe uses the first two of these large packets to produce an estimate of the upstream bottleneck bandwidth.

Using this technique, SProbe is able to produce *fast* estimates while maintaining normal Gnutella protocol semantics. With this approach, we were able to measure several tens of thousands of Gnutella host in a transparent way, without the need for their cooperation. Figure 5 shows the typical packet exchange sequence used by SProbe. In general, twelve round-trip times (RTTs) and exchanging less than 4KB are sufficient for SProbe to produce an estimate.

2.3.5 Latency Measurements Given the list of peers' IP-addresses obtained by the crawlers, we measured the round-trip latency between the peers and our measurement machines. For this, we used a simple tool that measures the RTT of a 40-byte TCP packet exchanged between a peer and our measurement host. Our interest in latencies of the peers is due to the well known feature of TCP congestion control which discriminates against flows with large round-trip times. This, coupled with the fact that the average size of files exchanged is in the order of 2-4 MB, makes latency a very important consideration when selecting amongst multiple peers sharing the same file. Although we realize that the latency to any particular peer is dependent on the location of the host from which it is measured, we feel the distribution of latencies over the entire population of peers from a given host might be similar (but not identical) from different hosts, and hence, is of general interest.

2.3.6 Lifetime Measurements To gather measurements of the availability (or "lifetime") characteristics of peers, we needed a tool that would periodically probe a large set of peers from both systems to detect when they were participating in the system. Every peer in both Napster and Gnutella connects to the system using a unique IP-address/port-number pair; to download a file, peers connect to each other using these pairs. There are therefore three possible states for any participating peer in either Napster or Gnutella:

1. **offline:** the peer is either not connected to the Internet or is not responding to TCP SYN packets because it is behind a firewall or NAT proxy.
2. **inactive:** the peer is connected to the Internet and is responding to TCP SYN packets, but it is disconnected from the peer-to-peer system and hence responds with TCP RST's.
3. **active:** the peer is actively participating in the peer-to-peer system.

We developed a simple tool (which we call *LF*) using Savage's "Sting" platform [23]. To detect the state of a host, *LF* sends a TCP SYN-packet to the peer and then waits for up to twenty seconds to receive any packets from it. If no packet arrives, we mark the peer as offline. If we receive a TCP RST packet, we mark the peer as inactive. If we receive a TCP SYN/ACK, we label the host as active, and send back a RST packet to terminate the connection. We chose to manipulate TCP packets directly rather than use OS socket calls to achieve greater scalability; this enabled us to monitor the lifetimes of tens of thousands of hosts per workstation. Because we identify a host by its IP address, one limitation in the lifetime characterization of peers our inability of distinguishing hosts sharing dynamic IP addresses (e.g., DHCP).

2.3.7 A Summary of the Active Measurements For the lifetime measurements, we monitored 17,125 Gnutella peers over a period of 60 hours and 7,000 Napster peers over a period of 25 hours. These peers were randomly selected from the set of all captured hosts. For each Gnutella peer, we determined its status (offline, inactive or active) once every seven minutes, and for each Napster peer, once every two minutes.

For Gnutella, we attempted to measure bottleneck bandwidths and latencies to a random set of 595,974 unique peers (i.e., unique IP-address/port-number pairs). We were successful in gathering downstream bottleneck bandwidth measurements to 223,552 of these peers, the remainder of which were either offline or had significant cross-traffic. We measured upstream bottleneck bandwidths from 16,252 of the peers (as Section 2.3.4 describes, upstream bottleneck bandwidth measurements require open TCP connections to measured hosts and, therefore, are harder to obtain than downstream measurements). Finally, we were able to measure latency to 339,502 peers. For Napster, we attempted to measure downstream bottleneck bandwidths to 4,079 unique peers. We successfully measured 2,049 peers.

In several cases, our active measurements were regarded as intrusive by several monitored systems. Unfortunately, e-mail complaints received by the computing staff at the University of Washington forced us to prematurely terminate our crawls, hence the lower number of monitored Napster hosts.

2.4 Limitations of the Methodology

Even though they are relatively new, peer-to-peer file-sharing systems are one of the most popular Internet applications. The wide deployment of home Internet connections (modems, cable modems and DSL) has contributed to the success of peer-to-peer as a file-sharing medium. The ability to completely characterize the participants in current peer-to-peer systems is highly important to system designers and network administrators. Although our methodology enables us to capture and understand some of the basic properties of participants in Napster and Gnutella, it falls short of providing a full and complete description of peers in the Internet.

An ideal characterization of peers should include a description of the workload they pose to peer-to-peer systems, to enable designers to tune these systems for high performance. In addition to their lifetime, knowing the rate of peers' births lets us understand the extent to which these systems must scale and lets designers simulate peer-to-peer models. Ideally, we would like to characterize participating hosts, even when they share or re-use IP addresses. Unfortunately, our crawlers only discover pairs of IP addresses and port numbers, and therefore we make each unique pair represent a single participant. As a result, our findings are susceptible to IP aliasing and IP reuse problems, such as NAT and DHCP.

Finally, as this paper will show, a large fraction of the peers participating in these systems are home users using broadband connections to the Internet. Very little is known about network characteristics of the broadband medium (cable modem and DSL): packet loss rates, where congestion occurs and whether routing can avoid it, last hop latencies, how high the variability of network latencies is.

3 Measurement Results

Our measurement results are organized according to a number of basic questions addressing the capabilities and behavior of peers. In particular, we attempt to address how many peers are capable of being servers, how many behave like clients, how many are willing to cooperate, and also how well the Gnutella network behaves in the face of random or malicious failures.

3.1 How Many Peers Fit the High-Bandwidth, Low-Latency Profile of a Server?

One particularly relevant characteristic of peer-to-peer file sharing systems is the percentage of peers in the system having *server*-like characteristics. More specifically, we are interested in understanding what

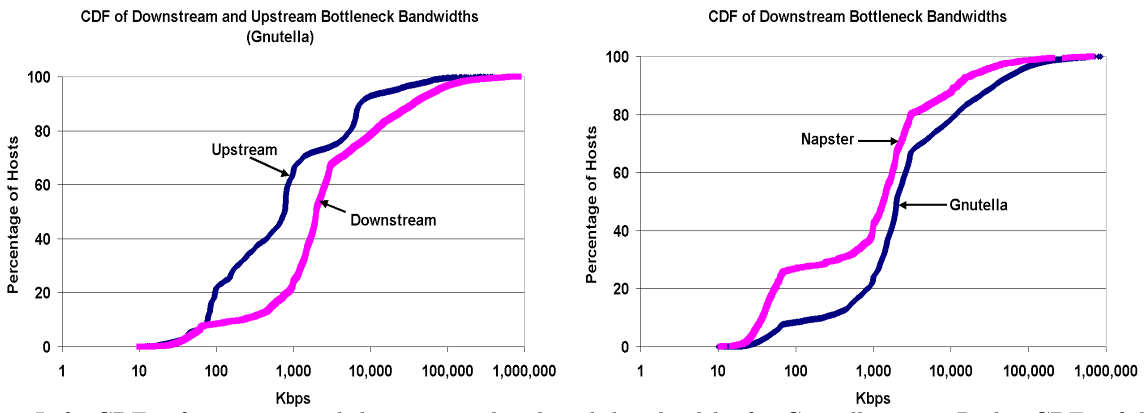


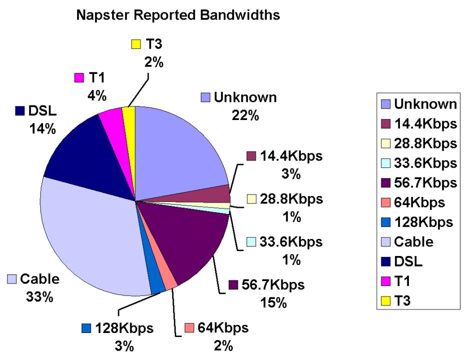
Fig. 6 Left: CDFs of upstream and downstream bottleneck bandwidths for Gnutella peers; Right: CDFs of downstream bottleneck bandwidths for Napster and Gnutella peers.

percentage of the participating peers exhibit the server-like characteristics with respect to their bandwidths and latencies. Peers worthy of being servers must have high-bandwidth Internet connections, they should remain highly available, and the latency of access to the peers should generally be low. If there is a high degree of heterogeneity amongst the peers, a well-designed system should pay careful attention to delegating routing and content-serving responsibilities, favoring server-like peers.

3.1.1 Downstream and Upstream Measured Bottleneck Link Bandwidths To fit the profile of a high-bandwidth server, a participating peer must have a high upstream bottleneck link bandwidth, since this value determines the rate at which a server can serve content. On the left, Figure 6 presents cumulative distribution functions (CDFs) of upstream and downstream bottleneck bandwidths for Gnutella peers.[3] From this graph, we see that while 92% of the participating peers have downstream bottleneck bandwidths of at least 100Kbps, only 8% of the peers have upstream bottleneck bandwidths of at least 10Mbps. Moreover, 22% of the participating peers have upstream bottleneck bandwidths of 100Kbps or less. Not only are these peers unsuitable to provide content and data, they are particularly susceptible to being swamped by a relatively small number of connections.

The left graph in Figure 6 reveals asymmetry in the upstream and downstream bottleneck bandwidths of Gnutella peers. On average, a peer tends to have higher downstream than upstream bottleneck bandwidth; this is not surprising, because a large fraction of peers depend on asymmetric links such as ADSL, cable modems or regular modems using the V.90 protocol [1]. Although this asymmetry is beneficial to peers that download content, it is both undesirable and detrimental to peers that serve content: in theory, the download capacity of the system exceeds its upload capacity. We observed a similar asymmetry in the Napster network.

[3] "Upstream" denotes traffic from the peer to the measurement node; "downstream" denotes traffic from the measurement node to the peer.



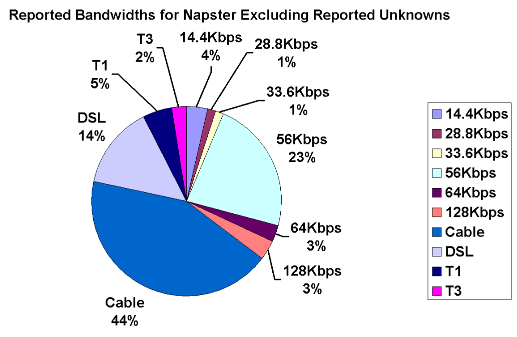


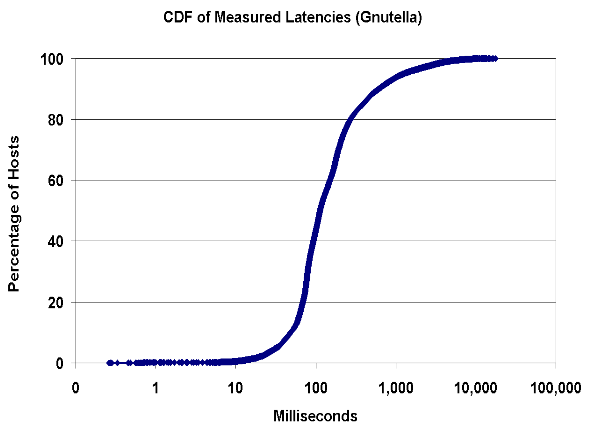
Fig. 7 Left: Reported bandwidths For Napster peers; Right: Reported bandwidths for Napster peers, excluding peers that reported "unknown".

The right graph in Figure 6 presents CDFs of downstream bottleneck bandwidths for Napster and Gnutella peers. As this graph illustrates, the percentage of Napster users connected with modems (of 64Kbps or less) is about 25%, while the percentage of Gnutella users with similar connectivity is as low as 8%.

At the same time, 50% of the users in Napster and 60% of the users in Gnutella use broadband connections (Cable, DSL, T1 or T3). Furthermore, only about 20% of the users in Napster and 30% of the users in Gnutella have very high bandwidth connections (at least 3Mbps). Overall, Gnutella users on average tend to have higher downstream bottleneck bandwidths than Napster users. Based on our experience, we attribute this difference to two factors: (1) the current flooding-based Gnutella protocol is too high of a burden on low bandwidth connections, discouraging them from participating, and (2) although unverifiable, there is a widespread belief that Gnutella is more popular to technically-savvy users, who tend to have faster Internet connections.

3.1.2 Reported Bandwidths for Napster Peers In contrast to Figure 6 that reports *measured* peer bandwidths, Figure 7 illustrates the breakdown of Napster peers with respect to their voluntarily *reported* bandwidths; the bandwidth that is reported is selected by the user during the installation of the Napster client software. (Peers that report "Unknown" bandwidth have been excluded in the right graph.)

As Figure 7 shows, a significant percent of the Napster users (22%) report "Unknown". These users are either unaware of their connection bandwidths, or they have no incentive to accurately report their true bandwidth. Indeed, knowing a peer's connection speed is more valuable to others rather than to the peer itself; a peer that reports high bandwidth is more likely to receive download requests from other peers, consuming network resources. Thus, users have an incentive to misreport their Internet connection speeds. A well-designed system therefore must either directly measure the bandwidths rather than relying on a user's input, or create the right incentives for the users to report accurate information to the system.



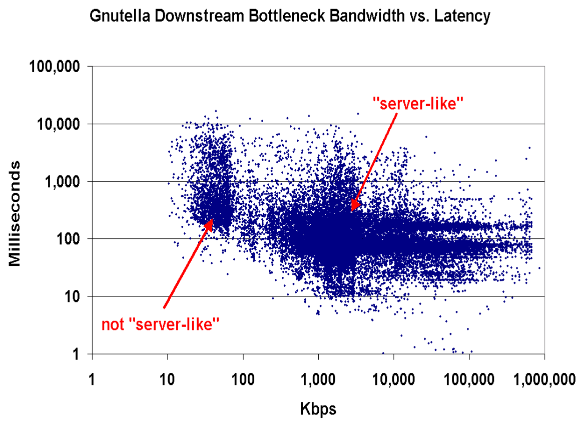


Fig. 8 Left: Measured latencies to Gnutella peers; Right: Correlation between Gnutella peers' downstream bottleneck bandwidth and latency.

Finally both Figures 6 and 7 confirm that the most popular forms of Internet access for Napster and Gnutella peers are cable modems and DSLs (bottleneck bandwidths between 1Mbps and 3.5Mbps).

3.1.3 Measured Latencies for Gnutella Peers Figure 8 (left) shows a CDF of the measured latencies from our measurement nodes to those Gnutella peers that form several consecutive snapshots of the Gnutella overlay . Approximately 20% of the peers have latencies of at least 280ms, whereas another 20% have latencies of at most 70ms: the closest 20% of the peers are four times *closer* than the furthest 20%. From this, we can deduce that in a peer-to-peer system where peers' connections are forged in an unstructured, ad-hoc way, a substantial fraction of the connections will suffer from high-latency.

On the right, Figure 8 shows the correlation between downstream bottleneck bandwidth and the latency of individual Gnutella peers (on a log-log scale). This graph illustrates the presence of two clusters; 10% of all peers form a smaller one situated at (20-60Kbps, 100-1,000ms) and 70% of all peers form a larger one at over (1,000Kbps, 60-300ms). These clusters correspond to the set of modems and broadband connections, respectively. The negatively sloped lower-bound evident in the low-bandwidth region of the graph corresponds to the non-negligible transmission delay of our measurement packets through the low-bandwidth links.

An interesting artifact evident in this graph is the presence of two pronounced horizontal bands. These bands correspond to peers situated on the North American East Coast and in Europe, respectively. Although the latencies presented in this graph are relative to our location (Seattle, WA, USA), these results can be extended to conclude that there are three large classes of latencies that a peer interacts with: (1) latencies to peers on the same part of the continent, (2) latencies to peers on the opposite part of a continent and (3) latencies to trans-oceanic peers. As Figure 8 shows, the bandwidths of the peers fluctuate significantly within each of these three latency classes.

3.2 How Many Peers Fit the High-Availability Profile of a Server?

Server worthiness is characterized not only by high-bandwidth and low-latency network connectivity, but also by the availability of the server. If, peers tend to be unavailable frequently, this will have significant implications about the degree of replication necessary to ensure that content is consistently accessible on this system.

On the left, Figure 9 shows the distribution of *uptimes* of peers for both Gnutella and Napster. Uptime is measured as the percentage of time that the peer is available and responding to traffic. The “Internet host uptime” curves represent the uptime as measured at the IP-level, i.e., peers that are in the inactive **or** active states, as defined in Section 2.3.6. The “Gnutella/Napster host uptime” curves represent the uptime of peers in the active state, and therefore responding to application-level requests. For all curves, we have eliminated peers that had 0% uptime (peers that were never up throughout our lifetime experiment).

The IP-level uptime characteristics of peers are quite similar for both systems; this implies that the set of peers participating in either Napster or Gnutella are homogeneous with respect to their IP-level uptime. In addition, only 20% of the peers in each system had an IP-level uptime of 93% or more.

In contrast, the application-level uptime characteristics of peers differ noticeably between Gnutella and Napster. On average, Napster peers tend to participate in the system more often than Gnutella peers. One might hastily conclude that since more users participate in Napster, more content is available and therefore peers have, on average, longer uptimes. However, this data can also be used to draw an opposite conclusion: more content means that users can find the files of interest faster, which results in shorter uptimes. We believe that this difference is primarily a factor of the design of the client software; Napster’s software has several features (such as a built-in chat client and an MP3 player) that cause users to run it for longer periods of time.

Another significant difference can be observed in the tail of the application-level distributions: the best 20% of Napster peers have an uptime of 83% or more, while the best 20% of Gnutella peers have an uptime of 45% or more. Our (unproven) hypothesis is that Napster is, in general, a higher quality and more useful service, and that this has a large influence on the uptime of its peers relative to Gnutella.

On the right, Figure 9 presents the CDF of Napster and Gnutella session durations that are *less than twelve hours*. The graph is limited to twelve hours because of the nature of our analysis method; we used the create-based method [21], in which we divided the captured traces into two halves. The reported durations are only for sessions that started in the first half, and finished in either the first or second half. This method provides accurate information about the distribution of session durations for session that are shorter than half of our trace, but it cannot provide any information at all about sessions that are longer than half our trace.[4] As Figure 9 illustrates, 50% of the peers never remain online for more than one hour. Since we

[4] This method is necessary, since sessions may be active (or inactive) for periods that are far longer than our trace duration; these long sessions, if unaccounted, would skew the results.

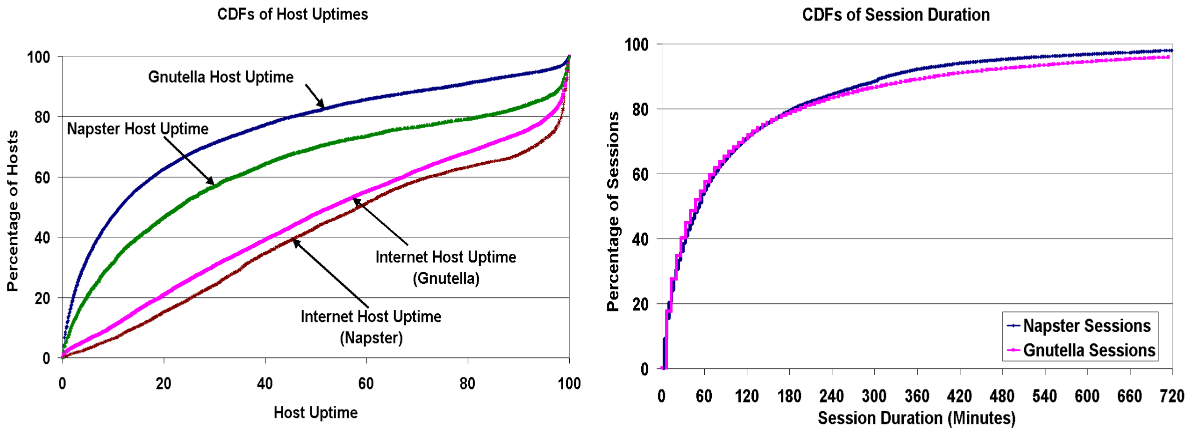


Fig. 9 Left: IP-level uptime of peers ("Internet Host Uptime"), and application-level uptime of peers ("Gnutella/Napster Host Uptime") in both Napster and Gnutella, as measured by the percentage of time the peers are reachable; Right: The distribution of Napster/Gnutella session durations.

observed a roughly constant number of peer participating in Napster and Gnutella, we conclude that over the course of one hour, half of the participants leave these systems and are replaced by another half.

There is an obvious similarity between Napster and Gnutella; for both, most sessions are quite short—the median session duration is approximately 60 minutes. This is not surprising, as it corresponds to the time it typically takes for a user to download a small number of music files from the service.

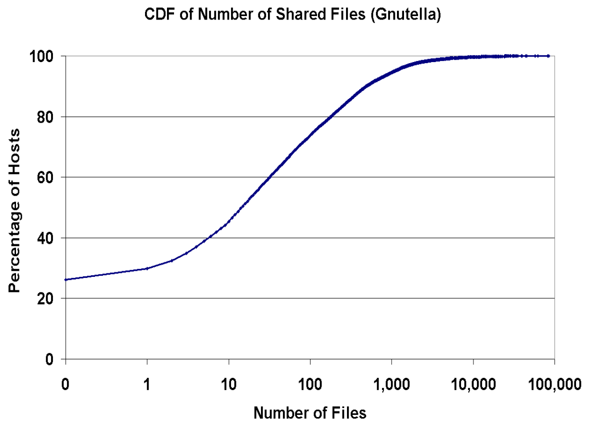
3.3 How Many Peers Fit the No-Files-to-Share, Always-Downloading Profile of a Client?

In addition to understanding the percentage of server-like Napster and Gnutella peers, it is equally important to determine the number of client-like peers. One aspect of a client-like behavior is that little or no data is shared in the system. Previous studies refer to these peers as *free-riders* [2] in the system.

Another variable of interest is the number of downloads and uploads a participating peer is performing at any given time. A peer with a high number of downloads fits the profile of a client, whereas a peer with a high number of uploads fits the profile of a server. In addition, correlating the number of downloads with a peer's bandwidth should depict a clear picture as to how many of the participating peers bring no benefits to the system, i.e., they have no files to share, they have low bandwidths, and they always download files.

Although we believe that any peer-to-peer system will have its *free-riders*, the system should not treat its peers equally, but instead, it should create incentives and rewards for peers to provide and exchange data.

3.3.1 Number of Shared Files in Napster and Gnutella In Figure 10, the left graph shows the distribution of shared files across Gnutella peers, and the right graph shows this distribution for both Napster and Gnutella, but with peers sharing no files eliminated from the graph. (As previously mentioned, we could not capture any information about peers with no files from Napster.)



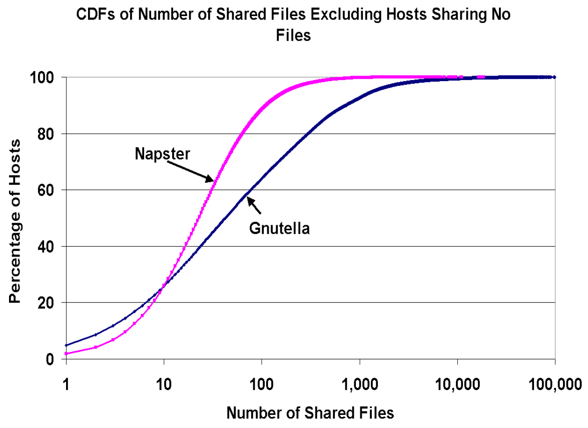


Fig. 10 Left: The number of shared files for Gnutella peers; Right: The number of shared files for Napster and Gnutella peers (peers with no files to share are excluded).

From the left graph, we see that as high as 25% of the Gnutella clients do not share any files. Furthermore, about 75% of the clients share 100 files or less, whereas only 7% of the clients share more than 1000 files. A simple calculation reveals that these 7% of users together offer more files than all of the other users combined. This fact illustrates that in spite of claims that *every peer is both a server and a client*, Gnutella has an inherently large percentage of *free-riders* [2].

The right graph shows that Napster peers are slightly more consistent and offer less variation in the number of shared files than Gnutella peers. Nonetheless, about 40-60% of the peers share only 5-20% of the shared files, which indicates that there is a large amount of free-riding in Napster as well.

3.3.2 Number of Downloads and Uploads in Napster In Figure 11, the left graph shows the distribution of concurrent downloads by Napster peers classified by the peer's reported bandwidth, and the right graph shows a similar curve for the number of concurrent uploads. Because these graphs were obtained by capturing snapshots of the download and upload activity using our crawler, these distributions are biased towards capturing low-bandwidth peers, since downloads take longer through low-bandwidth connections.

Nonetheless, this graph shows interesting correlations between peers' reported bandwidths and their concurrent downloads and uploads. First, there are 20% more zero-download high-speed peers than zero-download low-speed peers. We see two possible explanations: either higher-bandwidth peers tend to download less often, or they spend less time downloading because they have higher connection speeds. Second, the correlation between bandwidths and the downloads is reversed relative to bandwidths and uploads (the percentage of zero-upload peers is higher for modems than for cable modems).

3.3.3 Correlation between the Number of Downloads, Uploads, and Shared Files On the left, Figure 12 shows the percentage of downloads, the percentage of the peer population, the percentage of uploads and the percentage of shared files, grouped according to the reported bandwidth from Napster peers. The number

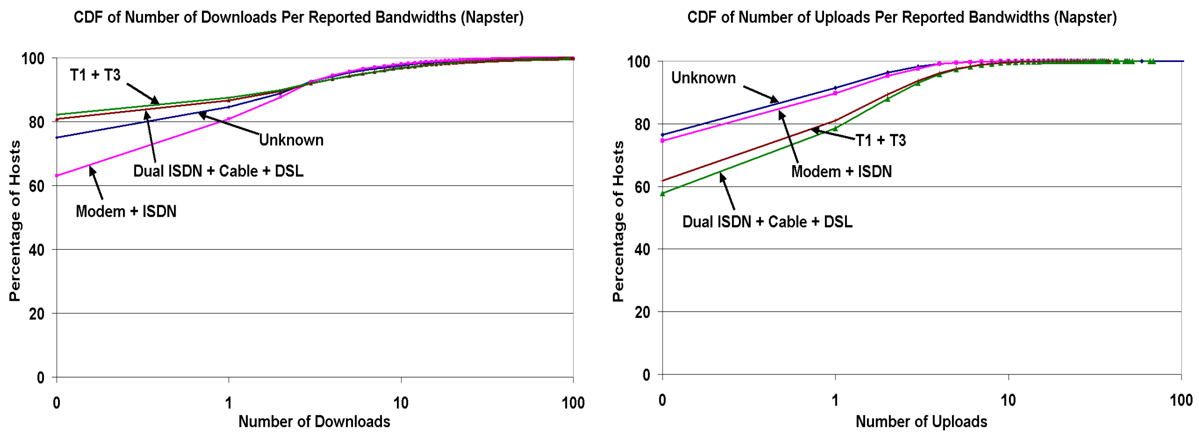


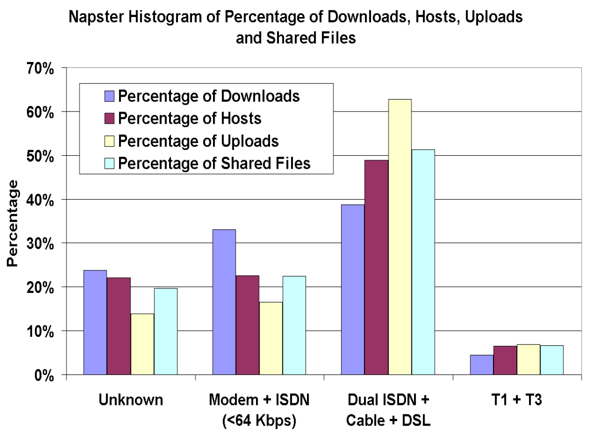
Fig. 11 Left: The number of downloads by Napster users, grouped by their reported bandwidths; Right: The number of uploads by Napster users, grouped by their reported bandwidths.

of shared files seems to be uniformly distributed across the population: the percentage of peers in each bandwidth class is roughly the same as the percentage of files shared by that bandwidth class.

However, the relative number of downloads and uploads varies significantly across the bandwidth classes. For example, although 56Kbps modems constitute only 15% of the Napster peers, they account for 24% of the downloads. Similarly, cable modems constitute 32% of the peers, but they account for 46% of the uploads. The skew in the number of uploads is attributed by users selecting high-bandwidth peers from which to download content. The skew in the number of downloads, however, seems to be more representative of the natural tendency of low-bandwidth peers to be free-riders.

On the right, Figure 12 shows the distribution of the number of shared files by Napster peers classified by the peer's reported bandwidth. A peer's bandwidth has little effect on the number of shared files. In Napster, half of the modem participants share 18 files or less, whereas half of the users with higher Internet connection speeds share 28 files or less. Unfortunately, since our methodology cannot capture peers sharing no files, it is possible that a different degree of correlation between a peer's bandwidth and its number of shared files might exist.

Figure 13 shows the distribution of concurrent downloads (on the left) and uploads (on the right) by Napster users classified by their number of shared files. On average, peers with fewer shared files perform fewer downloads and uploads. Having little data to share directly impacts the ability of a peer to contribute a large number of uploads to the system. However, as Figure 13 illustrates, peers with less shared data seem to be less interested, on average, to download from the system. Since, on the right, Figure 12 does not indicate any substantial lack of bandwidth available to these peers, we conclude that, although they are able to download as much data as everyone else, these participants prefer to rather download fewer files. Finally, the contrast in the slopes of the downloads and uploads distribution curves for the peers sharing



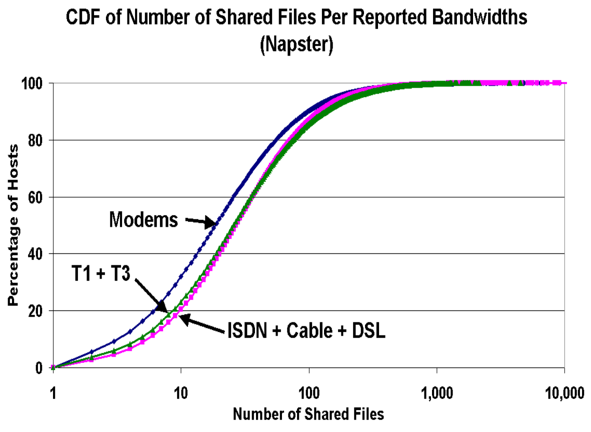
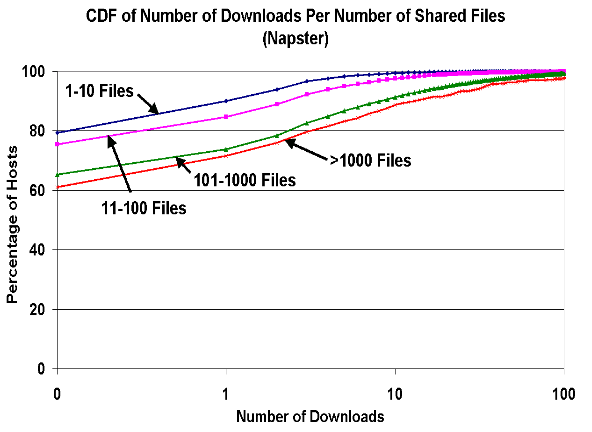


Fig. 12 Left: Percentage of downloads, peers, uploads and shared files, grouped by reported bandwidths (in Napster); Right: The number of shared files by Napster users, grouped by their reported bandwidths.



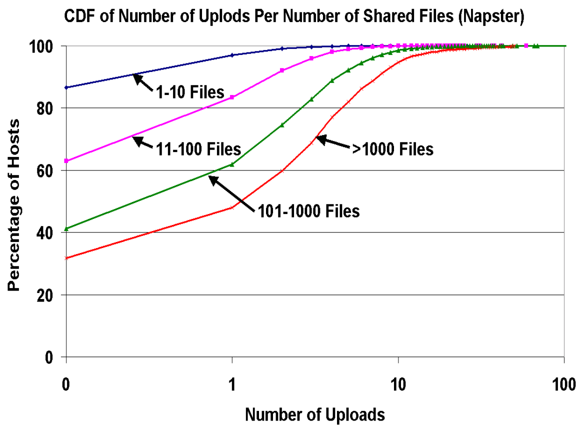


Fig. 13 Left: The number of downloads by Napster users, grouped by their number of shared files; Right: The number of uploads by Napster users, grouped by their number of shared files.

similar numbers of files demonstrates that peers contributing more data have a higher number of uploads, on average.

3.4 The Nature of Shared Files

Another aspect of interest deals with the characteristics of the shared files in the two systems. In Napster, all shared files must be in MP3 format, whereas any file type can be exchanged in Gnutella. Each point in Figure 14 corresponds to the number of files and the number of MB a Napster and Gnutella peer reports as shared (plotted on a log-log scale). The obvious lines in both graphs imply that there is a strong correlation between the numbers of files shared and the number of shared MB of data. The slopes of the lines in both graphs are virtually identical at 3.7MB, corresponding to the size of a shared typical MP3 audio file. Another

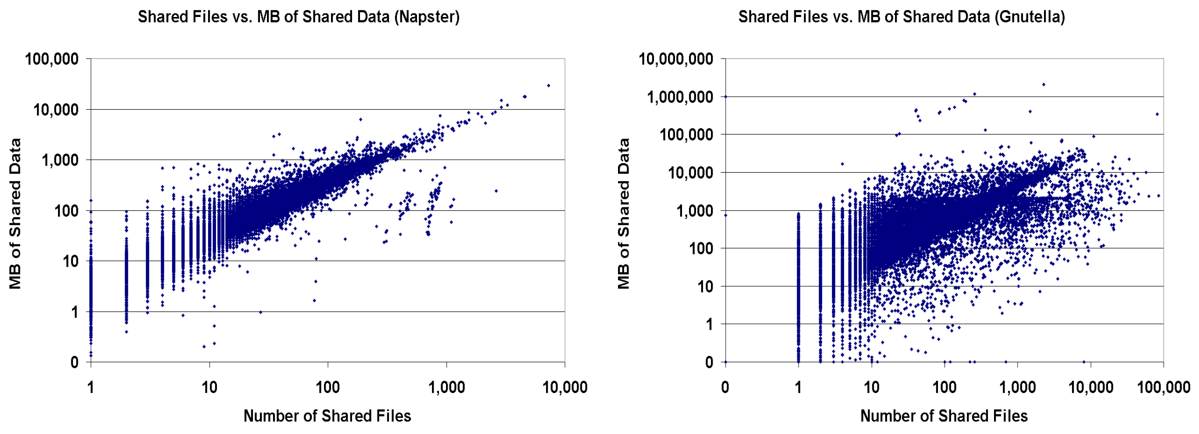


Fig. 14 Shared files vs. shared MB of data in Napster and Gnutella.

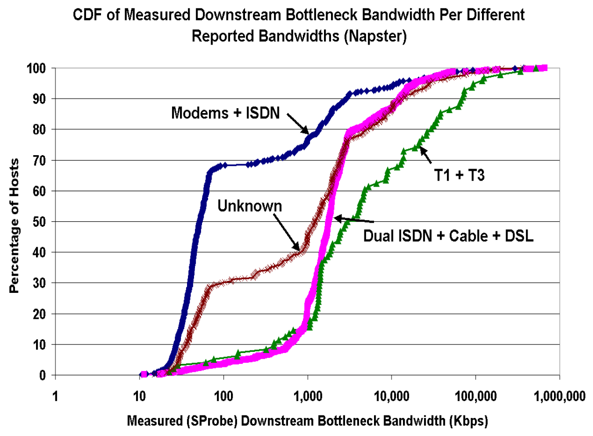
interesting point is the presence of a Gnutella peer that apparently shares 0 files but 730 MB of data; clearly a bug in the software or a case of malicious peers misreporting the amount of data they have to share.

3.5 How Much Are Peers Willing to Cooperate in a P2P File-Sharing System?

The peer-to-peer model fundamentally depends on the concept of cooperation. How willing peers are to cooperate is of vital importance to the viability of these systems. Devising an experiment to quantify a peer's willingness to cooperate is of course very difficult; as a first-order approximation, we measured the extent to which peers deliberately misreport their bandwidths.

The user interfaces for file querying are quite similar in Napster and Gnutella. When peers holding a requested file are discovered, their information, such as IP address, DNS name, network latency and reported bandwidth (including the value "Unknown"), are returned and presented to the user requesting the file. This user then selects one of these target peers and, as a result, it initiates a direct download of the requested file. The user downloading the file is likely to select a peer that has high bandwidth and low latency. In consequence, participating peers have an incentive to deliberately misreport lower bandwidths to the system, in order to discourage others from initiating downloads from them.

On the left, Figure 15 shows the distribution of measured bandwidths for Napster peers, classified by their reported bandwidth. Note that as high as 30% of the users that report their bandwidth as 64 Kbps or less actually have a significantly greater bandwidth. In Napster (and any similar system), a peer has an incentive to report a smaller bandwidth than the real value, in order to discourage others from initiating downloads and consuming the peer's available bandwidth. Similarly, we expect most users with high bandwidths to rarely misreport their actual bandwidths. Indeed, Figure 15 confirms that only than 10% of the users reporting high bandwidth (T1 and T3) in reality have significantly lower bandwidth. Because more high bandwidth peers



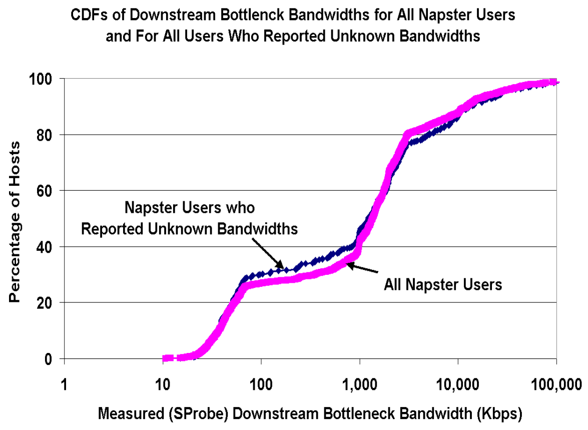


Fig. 15 Left: Measured downstream bottleneck bandwidths for peers, grouped by their reported bandwidths; Right: CDFs of measured downstream bottleneck bandwidths for those peers reporting unknown bandwidths along with all Napster users.

"misreport" their bandwidth than low bandwidth peers, it is unlikely that these are the result of ignorance or misconfiguration.

In addition to showing that many peers are uncooperative in Napster, this graph serves to validate the accuracy of our bottleneck bandwidth estimation technique. There is an extremely strong correlation between measured bandwidth and reported bandwidth, across all reported classes.

The right graph in Figure 15 shows the distribution of measured downstream bottleneck bandwidth of Napster peers reporting unknown bandwidths. Overlain on top of this distribution, we have shown the distribution of measured bandwidths of all Napster peers, regardless of their reported bandwidth. The similarity between the two curves implies that peers reporting unknown bandwidths are uniformly distributed across the population.

3.6 Resilience of the Gnutella Overlay in the Face of Attacks

In Gnutella, peers form an overlay network by each maintaining a number of point-to-point TCP connections, over which various protocol messages are routed. The Gnutella overlay presents a great opportunity to understand the challenges of creating effective overlay topologies. In particular, we were interested in the resilience of the Gnutella overlay in the face of failures or attacks.

In Gnutella, the fact that peers are connecting and disconnecting from the network has implications about the nature of the overlay topology. In practice, because peers tend to discover highly available and high-outdegree nodes in the overlay, connections tend to be formed preferentially. As Barabási and Albert show [3], vertex connectivities in networks that continuously expand by the addition of new vertices and in which nodes express preferential connectivity toward high-degree nodes follow a power-law distribution.

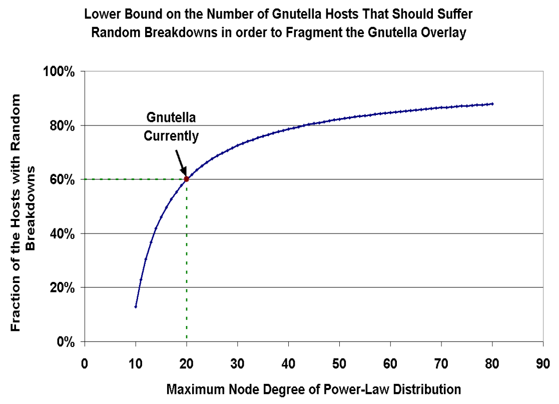


Fig. 16 Lower bound on the number of Gnutella peers that must suffer random breakdowns in order to fragment the Gnutella network.

Indeed, previous studies have confirmed the presence of a vertex connectivity power-law distribution for the Gnutella overlay [8] with an index of $\alpha = 2.3$.

Cohen et. al [9] have analytically derived that networks in which the vertex connectivity follows a power-law distribution with an index of at most ($\alpha < 3$) are very robust in the face of random node breakdowns. More concretely, in such networks, a connected cluster of peers that spans the entire network survives even in the presence of a large percentage p of random peer breakdowns, where p can be as large as:

$$p \leq 1 + \left(1 - m^{\alpha-2} K^{3-\alpha} \frac{\alpha - 2}{3 - \alpha}\right)^{-1}$$

where m is the minimum node degree and K is the maximum node degree. For Gnutella, Figure 16 shows a graph of this equation as a function of the maximum degree observed in the system, where the power-law index α was set to 2.3 and the minimum node degree m was set to 1.

As this graph shows, Gnutella presents a highly robust overlay in the face of random breakdowns; for a maximum node degree of 20 (which is approximately what we observed in the real Gnutella overlay), the overlay fragments only when more than 60% of the nodes shutdown. While overlay robustness is a highly desirable property, the assumption of random failures breaks down in the face of an orchestrated attack. A malicious attack would perhaps be directed against the best connected, popular, high degree nodes in the overlay.

The left graph in Figure 17 depicts the topology of 1771 peers forming a connected segment of the Gnutella network captured on February 16th, 2001. The middle graph shows a portion of the topology after 30% of the nodes are randomly removed. After this removal, the largest connected component in the topology consists of 1106 of the remaining 1300 nodes. However, in the right graph, we show the original topology after removing the 63 (less than 4%) best connected Gnutella peers. This removal has effectively “shattered” the overlay into a large number of disconnected components. As we see, although highly resilient in the face

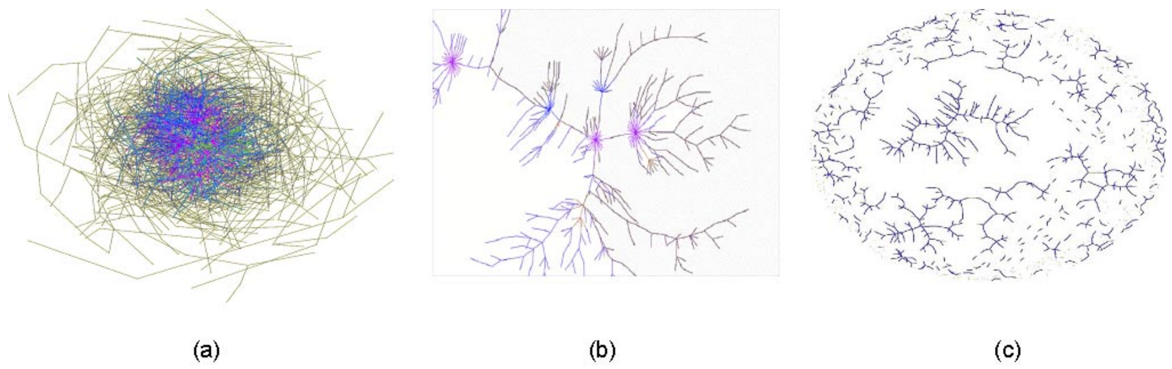


Fig. 17 Left: Topology of the Gnutella network as of February 16, 2001 (1771 peers); Middle: Topology of the Gnutella network after a random 30% of the nodes are removed; Right: Topology of the Gnutella network after the highest-degree 4% of the nodes are removed.

of random breakdowns, Gnutella is nevertheless highly vulnerable in the face of well-orchestrated, targeted attacks.

4 Recommendations to Peer-To-Peer System Designers

There has been a flurry of proposed distributed algorithms for routing and location in a P2P system. Most of these protocols and proposals make the implicit assumption that the delegation of responsibility across nodes the overlay should be uniform, and hence that all nodes will tend to participate and contribute equally in information exchange and routing. In contrast, our measurements indicate that the set of hosts participating in the Napster and Gnutella systems is heterogeneous with respect to many characteristics: Internet connection speeds, latencies, lifetimes, shared data. In fact, the magnitude of these characteristics vary between three and five orders of magnitude across the peers! Therefore, P2P systems should delegate different degrees of responsibility to different hosts, based on the hosts' physical characteristics and the degree of trust or reliability.

Another frequent implicit assumption in these systems is that peers tend to be willing to cooperate. By definition, to participate in a P2P system, a peer must obey the protocol associated with the system. In addition, most users tend to download pre-created software clients to participate in these systems (as opposed to authoring their own). These software packages typically ask users to specify configuration parameters (such as Internet connection speed) that will be reported to other peers. As we have shown, many of these parameters are in practice either left unspecified or deliberately misreported. Instead of relying on reported characteristics, we believe that a robust system should attempt to directly measure the characteristics of peers in the system.

Another myth in P2P file-sharing systems is that all peers behave equally, both contributing resources and consuming them. Our measurements indicate that this is not true: client-like and server-like behavior can clearly be identified in the population. As we have shown, approximately 26% of Gnutella users shared no data; these users are clearly participating to download data and *not* to share. Similarly, in Napster we observed that on average 60-80% of the users share 80-100% of the files, implying that 20-40% of users share little or no files.

The experiments and the data presented in this paper indicate that many of the characteristics that Napster and Gnutella P2P systems in practice match the characteristics of the classic server-client model. Thus, we believe that future robust P2P protocols should account for the hosts heterogeneity, relying on self-inspection and adaptation to exploit the differences in the hosts' characteristics, behavior, and incentives.

5 Conclusions

In this paper, we presented a measurement study performed over the population of peers that choose to participate in the Gnutella and Napster peer-to-peer file sharing systems. Our measurements captured the bottleneck bandwidth, latency, availability, and file sharing patterns of these peers.

Several lessons emerged from the results of our measurements. First, there is a significant amount of heterogeneity in both Gnutella and Napster; bandwidth, latency, availability, and the degree of sharing vary between three and five orders of magnitude across the peers in the system. This implies that any similar peer-to-peer system must be very deliberate and careful about delegating responsibilities across peers. Second, even though these systems were designed with a symmetry of responsibilities in mind, there is clear evidence of client-like or server-like behavior in a significant fraction of systems' populations. Third, peers tend to deliberately misreport information if there is an incentive to do so. Because effective delegation of responsibility depends on accurate information, this implies that future systems must either have built-in incentives for peers to tell the truth or systems must be able to directly measure and verify reported information.

References

1. 3Com. 3com v.90 technology, 1998. `http://www.3com.com/technology/tech_net/white_papers/pdf/50065901.pdf`.
2. E. Adar and B. Huberman. Free riding on gnutella. *First Monday*, 5(10), October 2000.
3. A. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
4. Jean-Chrysostome Bolot. End-to-End Packet Delay and Loss Behavior in the Internet. In *Proceedings of the ACM SIGCOMM 1993 Technical Conference*, pages 289–298, August 1993.
5. R. Carter. Cprobe and bprobe tools. `http://cs-people.bu.edu/carter/tools/Tools.html`, 1996.
6. Robert L. Carter and Mark E. Crovella. Measuring Bottleneck Link Speed in Packet-Switched Networks. Technical Report BU-CS-96-006, Computer Science Department, Boston University, March 1996.

7. I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, Berkeley, CA, 2000.
8. Clip2. Gnutella measurement project, May 2001. `http://www.clip2.com/`.
9. Reuven Cohen, Keren Erez, Daniel ben Avraham, and Shlomo Havlin. Resilience of the internet to random breakdowns. *Physical Review Letters*, 85(21), November 2000.
10. Constantinos Dovrolis, Parmesh Ramanathan, and David Moore. What Do Packet Dispersion Techniques Measure? In *Proceedings of the IEEE INFOCOM 2001*, Anchorage, AK, USA, April 2001.
11. Peter Druschel and Antony Rowstron. Past: A large-scale, persistent peer-to-peer storage utility. In *Proceedings of the Eighth IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, Schoss Elmau, Germany, May 2001.
12. V. Jacobson. Congestion Avoidance and Control. In *Proceedings of the ACM SIGCOMM '88 Conference*, Stanford, CA, USA, August 1988.
13. Van Jacobson. pathchar. `http://www.caida.org/tools/utilities/others/pathchar/`, 1997.
14. Srinivasan Keshav. A Control-Theoretic Approach to Flow Control. In *Proceedings of the ACM SIGCOMM 1991 Technical Conference*, September 1991.
15. Kevin Lai. Nettimer. `http://mosquitonet.stanford.edu/~laik/projects/nettimer/`, 2001.
16. Kevin Lai and Mary Baker. Measuring Link Bandwidths Using a Deterministic Model of Packet Delay. In *Proceedings of the ACM SIGCOMM 2000 Technical Conference*, Stockholm, Sweden, August 2000.
17. Kevin Lai and Mary Baker. Nettimer: A Tool for Measuring Bottleneck Link Bandwidth. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems*, March 2001.
18. Bruce A. Mah. pchar: A tool for measuring internet path characteristics. `http://www.employees.org/~bmah/Software/pchar/`, 2001.
19. Vern Paxson. *Measurements and Analysis of End-to-End Internet Dynamics.* PhD thesis, University of California, Berkeley, April 1997.
20. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the ACM SIGCOMM 2001 Technical Conference*, San Diego, CA, USA, August 2001.
21. Drew Roselli, Jacob Lorch, and Thomas Anderson. A comparison of file system workloads. In *Proceedings of the 2000 USENIX Annual Technical Conference*, San Diego, CA, USA, June 2000.
22. Stefan Saroiu. SProbe: A Fast Tool for Measuring Bottleneck Bandwidth in Uncooperative Environments. `http://sprobe.cs.washington.edu`, 2001.
23. Stefan Savage. Sting: a TCP-based Network Measurement Tool. In *Proceedings of the 1999 USENIX Symposium on Internet Technologies and Systems (USITS '99)*, October 1999.
24. Stefan Savage, Neal Cardwell, David Wetherall, and Tom Anderson. TCP Congestion Control with a Misbehaving Receiver. In *ACM Computer Communications Review*, volume 29, pages 71–78, October 1999.
25. Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable content-addressable network. In *Proceedings of the ACM SIGCOMM 2001 Technical Conference*, San Diego, CA, USA, August 2001.

26. M. Waldman, A.D. Rubin, and L.F. Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proceedings of the 9th USENIX Security Symposium*, August 2000.

27. B.Y. Zhao, K.D. Kubiatowicz, and A.D. Joseph. Tapestry: An infrastructure for fault-resilient wide-area location and routing. Technical Report UCB//CSD-01-1141, University of California at Berkeley Technical Report, April 2001.