Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computer Networks 50 (2006) 842–858

Computer Networks

www.elsevier.com/locate/comnet

The FastTrack overlay: A measurement study

Jian Liang [a,*], Rakesh Kumar [b], Keith W. Ross [a]

[a] *Department of Computer and Information Science, Polytechnic University, Brooklyn, NY 11201, United States*
[b] *Department of Electrical and Computer Engineering, Polytechnic University, Brooklyn, NY 11201, United States*

Available online 31 August 2005

Abstract

Both in terms of number of participating users and in traffic volume, FastTrack is one of the most important applications in the Internet today. Nevertheless, because FastTrack is proprietary and uses encryption, little is understood about FastTrack's overlay structure and dynamics, its messaging protocol, and its index management. We have built two measurement apparatus—the FastTrack Sniffing Platform and the FastTrack Probing Tool—to unravel many of the mysteries behind FastTrack. We deploy the apparatus to study FastTrack's overlay structure and dynamics, its neighbor selection, its use of dynamic port numbers to circumvent firewalls, and its index management. Although this study does not fully solve the FastTrack puzzle, it nevertheless leads to a coherent description of FastTrack and its overlay. Furthermore, we leverage the measurement results to set forth a number of key principles for the design of a successful unstructured P2P overlay. The measurement results and resulting design principles in this paper should be useful for future architects of P2P overlay networks as well as for engineers managing ISPs.
© 2005 Elsevier B.V. All rights reserved.

Keywords: P2P; FastTrack; Overlay; Network measurement

1. Introduction

In Fall 2003, when this study was performed, FastTrack (which includes KaZaA, Grokster and Imesh) had more than 3 million active users sharing over 5000 terabytes of content. On the University of Washington campus network in June 2002, FastTrack consumed approximately 37% of all TCP traffic, which was more than twice the Web traffic on the same campus at the same time [6]. With over 3 million users, FastTrack is significantly more popular than Napster or Gnutella ever was. Sandvine estimates that in the US 76% of P2P file sharing traffic is FastTrack traffic and only 8% is Gnutella traffic [19]. Clearly, both in terms of number of participating users and in traffic volume, FastTrack is one of the most important

[*] Corresponding author.
E-mail addresses: jliang@cis.poly.edu (J. Liang), rkumar04@utopia.poly.edu (R. Kumar), ross@poly.edu (K.W. Ross).

1389-1286/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2005.07.014

applications ever carried by the Internet. In fact, it can be argued that FastTrack has been so successful that any new proposal for a P2P file sharing system should be compared with the FastTrack benchmark. However, largely because FastTrack is a proprietary protocol which encrypts its signalling messages, little has been known to date about the specifics of FastTrack's overlay, the maintenance of the overlay, and the FastTrack signalling protocol.

In this paper we undertake a comprehensive measurement study of FastTrack's overlay structure and dynamics, its neighbor selection, its use of dynamic port numbers to circumvent firewalls, and its index management. Although this study does not fully solve the FastTrack puzzle, it nevertheless leads to a coherent description of FastTrack and its overlay, while providing many new insights about the details of FastTrack.

To unravel the mysteries of the FastTrack overlay, we developed two sets of measurement apparatus: the FastTrack Sniffing Platform and the FastTrack Probing Tool. The *FastTrack Sniffing Platform* is a set of FastTrack nodes that are forced to interconnect in a controlled manner with one another, while one node is also connected to hundreds of platform-external FastTrack nodes. The FastTrack Sniffing Platform collects FastTrack signalling traffic, from which we can draw conclusions about the structure and dynamics of the FastTrack overlay. The *FastTrack Probing Tool* establishes a TCP connection with any supplied FastTrack node, handshakes with that node, and sends and receives arbitrary encrypted FastTrack messages with the node. It is used for analyzing node availabilities and FastTrack neighbor selection. Both of these apparatus consume limited resources. One of the contributions of this paper is to show how it is possible to obtain extensive overlay information of a large-scale overlay application with a low-cost measurement infrastructure.

We use these tools to obtain insight into the following questions:

- It is well known that the FastTrack overlay is organized in a two-tier hierarchy consisting of supernodes (SNs) in the upper tier and ordinary nodes (ONs) in the lower tier. But how many children ONs does a typical SN support? What fraction of the peers in FastTrack are SNs? Are the SNs densely interconnected or sparsely interconnected?
- How long are ON-to-SN connections in the overlay? How long are SN-to-SN connections in the overlay? What is the typical lifetime of a SN?
- How does an ON discover candidate SNs for parenting? Once it has a set of candidate SNs, how does it choose a particular parent among them? In choosing the parent, does it take locality or SN workload into account?
- By allowing peers (ONs and SNs) to select their own server port numbers, FastTrack is more difficult to block with firewalls and NATs (Network Address Translations). How does FastTrack manage the server port numbers? What fraction of FastTrack nodes are behind NATs?
- What are the characteristics of the protocol that peers use to establish overlay links among themselves?
- How is the file index (relating each file copy to an IP address and port number) organized among the SNs?

In addition to providing novel insights into a remarkably successful P2P system, we leverage our measurement results to set forth a number of key principles for the design of an unstructured P2P overlay. As we discuss in Section 5 these principles, include distributed design, exploiting heterogeneity, load balancing, locality, connection shuffling, and firewall/NAT circumvention.

This paper should not only be of interest to P2P designers, but also to engineers at upper- and lower-tier ISPs, who are interested in acquiring a thorough understanding of P2P overlays and traffic. Because P2P file sharing systems can generate vast quantities of traffic, networking engineers, who dimension the network and introduce content distribution devices such as caches, need a basic understanding of how major P2P file sharing systems operate. Although there has been recent work in analyzing the file-sharing workload in FastTrack [6,14], to our knowledge we are the first to undertake a comprehensive study of a hierarchical unstructured overlay for a P2P system.

The paper focuses on the FastTrack overlay network and index management. It addresses neither FastTrack's downloading protocol (for example, FastTrack's parallel downloading and request queuing) nor its incentive scheme for encouraging uploading. The paper is complementary to [6,14], which focus on FastTrack file-sharing traffic. It is also complementary to a recent measurement study on pollution in P2P file sharing systems [15].

This paper is organized as follows. Section 2 provides an overview of FastTrack. Section 3 describes are measurement apparatus. Section 4 presents our measurement results. Section 5 sets forth basic design principles for unstructured P2P file sharing applications. Section 6 surveys related work. Finally, Section 7 summarizes our findings and concludes.

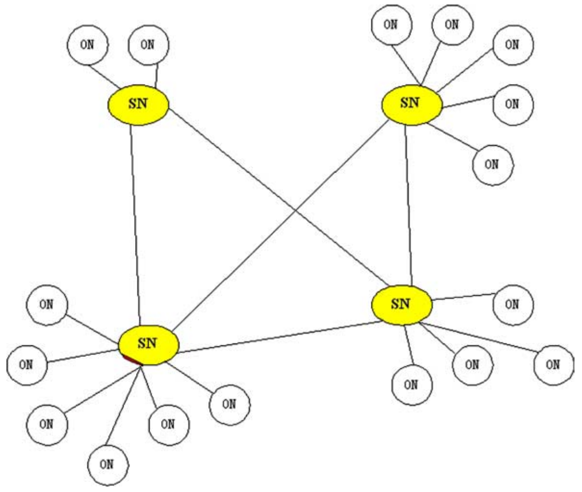


Fig. 1. FastTrack's two-tier overlay network.

2. Overview of the FastTrack

The FastTrack Web site [10] provides a rudimentary description of how FastTrack works. Moreover, various (and often obscure) articles, Web sites, and message boards provide additional scraps of information. In this section we collect and unify this publicly available information. The goal of this section is to (i) organize this obscure information in a digestable form for the P2P research community and (ii) present a broad-brush picture of FastTrack and its overlay. In the subsequent sections we describe our own measurement contributions.

FastTrack peers differ in availability, bandwidth connectivity, CPU power, and NATed access. FastTrack was one of the first P2P systems to exploit this heterogeneity by organizing the peers into two classes, supernodes (SNs) and ordinary nodes (ONs). SNs are generally more powerful in terms of connectivity, bandwidth, processing, and non-NATed accessibility. As we shortly describe, SNs also have greater responsibilities. As shown in Fig. 1, each ON has a parent SN. When an ON launches the FastTrack application, the ON chooses a parent SN, maintains a semi-permanent TCP connection with its parent SN, and uploads to this SN the metadata for the files it is sharing.

As with most other P2P file sharing systems, FastTrack maintains a file index that maps file identifiers to the IP addresses. This file index is distributed across the SNs. In particular, each SN maintains a local index for all of its children ONs, so that each SN is similar to a (mini) Napster hub. But in contrast with Napster, a SN is not a dedicated server; instead, it is typically a peer belonging to an individual user.

We know from [12] that for each file an ON is sharing, the metadata that the ON uploads to its parent SN includes: the *file name*, the *file size*, the *ContentHash*, and the *file descriptors* (for example, artist name, album name, and text entered by users). The file descriptors are used for keyword matches during querying. The ContentHash plays an important role in the FastTrack architecture. FastTrack hashes every file to a hash signature, which becomes the ContentHash of the file. The ContentHash is the only identifier used to identify a file in an HTTP download request. If a download from a specific peer fails, the ContentHash enables the FastTrack client to automatically search for the specific file, without issuing a new keyword query.

When a user wants to locate files, the user's ON sends a query with keywords over the TCP connection to its parent SN. For each match in its database, the SN returns the IP address, server port number, and metadata corresponding to the match. As shown in Fig. 1, each SN also maintains long-lived TCP connections with other SNs, creating an overlay network among the SNs. When a SN receives a query, it may forward the query to one or more of the SNs to which it is connected. A given query will in general visit a small subset of the SNs, and hence will obtain the metadata information of a small subset of all the ONs.

The FastTrack File Format project [12] has determined the syntax and semantics of FastTrack system files. From this project, we know that a FastTrack peer has the following software components:

1. The FastTrack Media Desktop (KMD).
2. Software environment information stored in the Windows Registry. Included in this environment information is a list of up to 200 SNs, which we refer to as the *SN list cache*. For each the 200 SNs in cache, the list includes a number of attributes including the SN IP address and port number.
3. DBB files, with each DBB file containing metadata for the files that the peer is willing to share. An active FastTrack process permanently monitors the local folders that are shared; file add, delete, are reflected in the DBB file.
4. DAT files, with each file containing a partially downloaded file. A DAT file grows in size as more data is retrieved. Once all the file data is retrieved, the DAT file is renamed to the original file which was intended to be downloaded.

Each FastTrack peer exchanges four different types of TCP traffic with other peers in the network:

1. Signaling traffic, which includes handshaking traffic for connection establishment between peers; metadata extracted from the DBB files, uploaded from ONs to SNs; supernode lists; and queries and replies. All signaling traffic is encrypted.
2. File transfer traffic (e.g., MP3s, videos, etc.) transferred directly among the peers without passing through intermediate SNs. File transfers are not encrypted and are sent within HTTP messages.
3. Commercial advertisements, sent over HTTP.
4. Instant messaging traffic, encoded as Base64.

The FastTrack ON–SN and SN–SN signalling messages are encrypted. The impressive giFT project [1] has reverse-engineered FastTrack's encryption algorithms, so that users of giFT-FastTrack can search and download files from the FastTrack network. Some of our own measurement tools incorporate the encryption/decryption code provided through the giFT project. The Sig2dat tool project [25] makes available a tool for obtaining the FastTrack ContentHash of any file. This tool is increasingly being used by FastTrack users, who post file names and corresponding ContentHash values on Web sites and message boards. This helps in countering pollution attacks, wherein bogus files are intentionally placed in the network by competing interests [18,15].

Many users today use KaZaA-Lite [11], an unofficial copy of KMD, rather than the KaZaA client (KMD) distributed by Sharman. Each KaZaA-Lite client emulates Sharman's KMD and participates in the FastTrack network. During the search process, a KaZaA-Lite ON first sends its query to the SN to which it is connected. We have learned from our own measurement work that after receiving all the replies from its parent SN, the ON disconnects and connects with a new SN, and re-sends the query to the new SN. During a specific search, the ON may connect to many SNs. The ordinary node typically maintains the TCP connection with the last SN in the sequence of connections, until another search is performed. Our measurement work has determined that during each hop, the ON re-sends its metadata to the new SN, and the previous SN removes the ONs metadata.

3. Measurement apparatus

This section describes the measuring apparatus we use to conduct our measurements on FastTrack's overlay.

3.1. The FastTrack sniffing platform

As shown in Fig. 2, we built a sniffing platform consisting of three workstations, each with a KMD version 2.0 client installed. We patiently

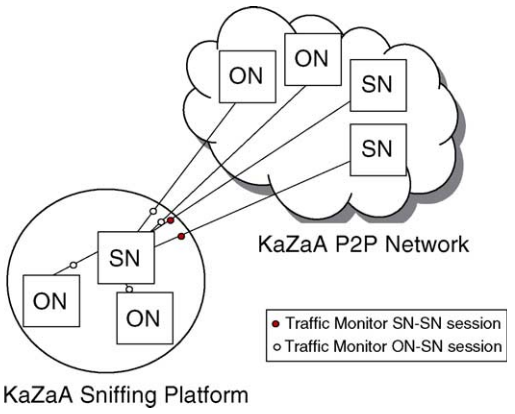

ON
ON
SN
SN
SN
ON
ON
KaZaA P2P Network
● Traffic Monitor SN-SN session
○ Traffic Monitor ON-SN session
KaZaA Sniffing Platform

Fig. 2. *The FastTrack Sniffing Platform*. The SN typically has hundreds of connections to external peers.

waited until FastTrack promoted one of the three nodes to a SN. At startup all three workstations functioned as ONs in the FastTrack network. These workstations enjoyed high bandwidth connectivity with sufficient hardware resources. When one of the workstations was promoted to a SN, we manipulated the Windows Registries in the other two Platform ONs in a manner that forced them to adopt the Platform SN as their parent. (How we did this will soon become clear.)

As shown in Fig. 2, the Platform SN also connects to platform-external FastTrack ONs and SNs. We deploy software traffic monitors around the Platform SN to capture all of the inbound and outbound signalling traffic. We then do an off-line traffic analysis based on our understanding of the FastTrack signalling protocol.

The FastTrack Sniffing Platform was installed in two different subnets: one connected to the Polytechnic University campus network; the other connected to a residential cable access network. In this manner we can take snapshots of the FastTrack network from two entirely different types of network access. These locations were chosen because they are representative of the type of subnets that the majority of FastTrack peers connect through, i.e., campus networks and residential access.

By deploying the FastTrack Sniffing Platform in conjunction with the encryption/decryption tools from the giFT project [1], we have been able to determine that FastTrack nodes frequently exchange with each other lists of SNs. In particular, when an ON connects with a parent SN, the SN immediately pushes to the ON a *SN refresh list*, which consists of the IP addresses, port numbers and workload values of up to 200 SNs. The first entry in the SN refresh list is the parent SN that is sending the list. When an ON receives a SN refresh list from its parent SN, the ON will typically purge some of the entries from its SN list cache and add entries sent by the parent SN. Neighboring SNs in the overlay also exchange SN refresh lists. By frequently exchanging SN refresh lists, nodes maintain up-to-date lists of active SNs.

3.2. Overlay probing tool

By deploying the FastTrack Sniffing Platform in conjunction with the encryption/decryption tools from the giFT project [1], we have been able to determine the sequence and semantics of many of the FastTrack messages, as well as the sequence of events that ensue during overlay link establishment between ON and SN and between SN and SN.

When a peer launches the FastTrack client, the first task of the client is to choose a parent SN and establish an overlay link (i.e., TCP connection) with it. To this end, we have discovered that the following steps are taken:

- As described in Section 2, the ON has a SN list cache. The ON chooses several (typically 5) candidate SNs from the list and probes the candidates by sending one UDP packet to each candidate. The ON then receives UDP responses from a subset of these candidates.
- To each SN from which it receives a UDP response, the ON attempts to establish a TCP connection. For each such connection, the SN and ON will exchange encryption key material, the ON will send peer information, and the SN will send a SN refresh list. Included in the peer information is the local IP address, service port number and username. The ON may be behind a NAT in which case the local IP address is a private address and different from the NAT's address.

- The ON will then select one of the SNs and disconnect from the other SNs. The one remaining SN becomes the ON's parent SN.
- The ON can then send query messages to its chosen SN.

The timing diagram for both the ON–SN and SN–SN overlay connection establishment is shown in Fig. 3. Note that the procedure for establishing a SN–SN overlay link differs from that of establishing ON–SN overlay link.

We also determined the structure of signalling messages exchanged between FastTrack peers. This structure is presented in Fig. 4. Each message begins with the identifier "K", which is then followed by a message-type field (two bytes), a payload-length field (two bytes), and the payload itself.

Based on our understanding of the FastTrack signaling protocol and message structure, we have developed the FastTrack Overlay Probing Tool. This tool can fully emulate the behavior of the

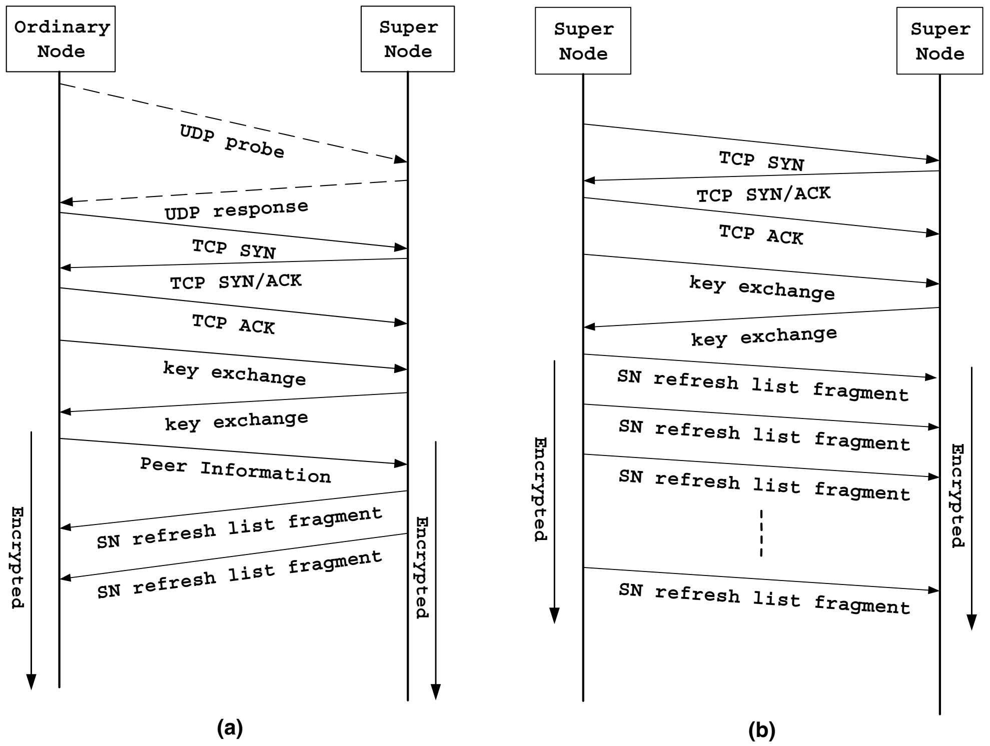


Fig. 3. Connection establishment protocol between peers in the FastTrack overlay. (a) ON–SN connection establishment and (b) SN–SN connection establishment.

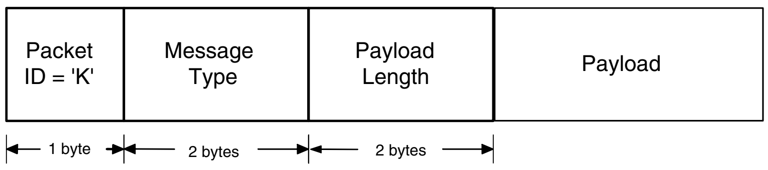


Fig. 4. *FastTrack's signalling message format.* Five bytes of header with variable length payload.

KMD client for initiating new connections and exchanging signalling messages with other FastTrack nodes. We use this tool to (1) probe whether any arbitrarily specified SN in the overlay is alive and (2) to retrieve the SN refresh list sent from probed SN, and (3) obtain the workload of the probed SN. As discussed in Section 4, we use this list to test our hypothesis of locality awareness as a selection criterion in forming new SN–ON overlay links in FastTrack.

4. Measurement results

4.1. Overlay structure and dynamics

4.1.1. Structural properties of the overlay

We first explored the degree of connectivity of a typical SN. Specifically, from the SN in the FastTrack Sniffing Platform, we study the number of simultaneous connections to platform-external ONs and SNs. The Platform SN does not offer any files for sharing. From the moment when a Platform node is promoted to a SN, we monitor the number TCP connections emanating from the SN. We did this experiment in the two environments—Polytechnic campus and broadband residential access network—over different time periods. Fig. 5 shows that in every case, the number of connections begins at one and climbs to a threshold, around which it subsequently vacillates. For the number of simultaneous SN–SN connections, this threshold is almost always in the 40–50 range. For the number of simultaneous SN–ON connections, depending on the day, this threshold is in the 100–160 connection range for the Platform SN on the Polytechnic campus and in the 55–70 range for the Platform SN in the residential access network. With the use of the status message in the FastTrack protocol we estimated

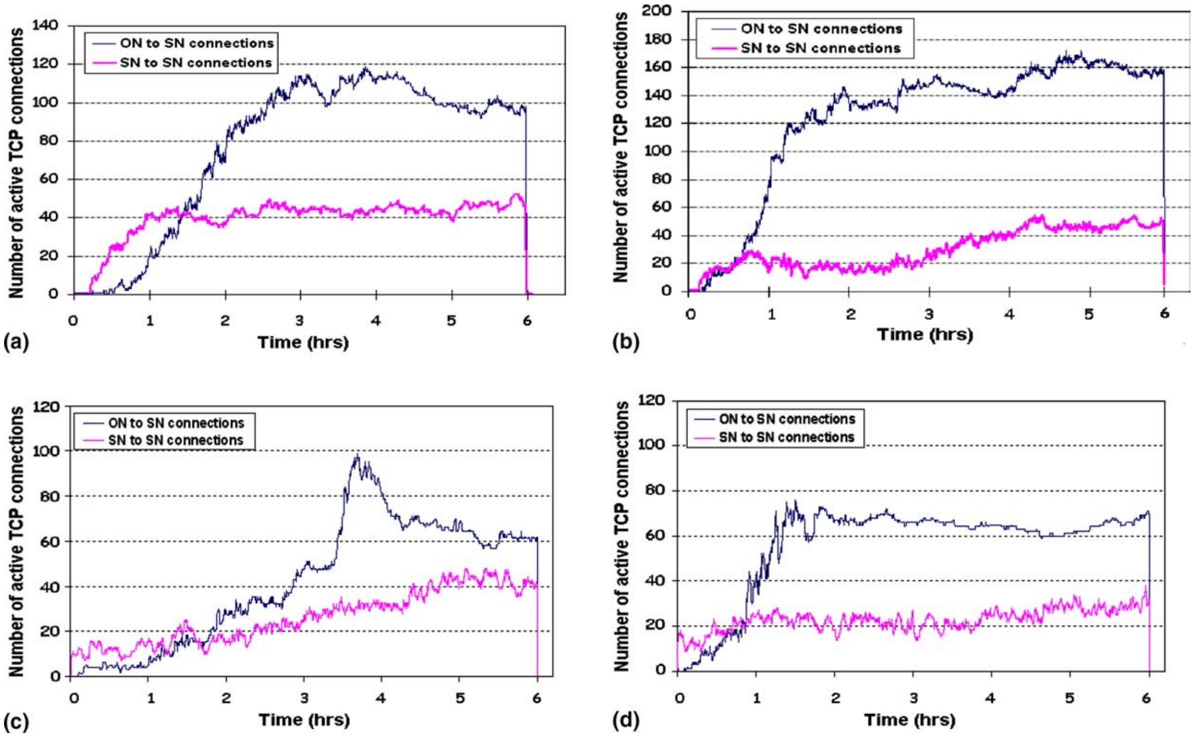


Fig. 5. Evolution of SN–SN and ON–SN connections with time. (a) Polytechnic campus—session evoluiton, Aug. 22, 2003. (b) Polytechnic campus—session evoluiton, Oct. 24, 2003. (c) Residential Cable Modem—session evoluiton, Aug. 23, 2003. (d) Residential Cable Modem—session evoluiton, Oct. 25, 2003.

the size of FastTrack network as roughly 3 million peers. We therefore speculate that there are on the order of 25,000–40,000 SNs in the FastTrack overlay, with the number varying with the time of day. This claim has also been corroborated by a complimentary measurement study reported in [15]. Combining these estimates, we conclude that the SN–SN overlay network is very sparsely connected, with each SN connected to about 0.1% of other SNs in the overlay.

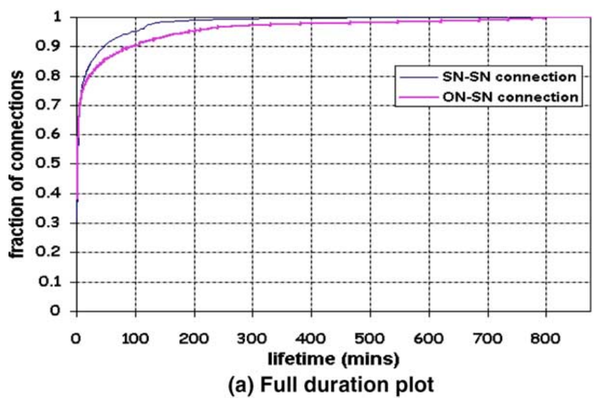
4.1.2. Overlay dynamics

Our measurement study has determined the FastTrack overlay is highly dynamic. Although the number of simultaneous connections vacillates around a threshold, as observed in Fig. 5, the individual connections change frequently.

Using the FastTrack Sniffing Platform we performed measurements on the duration of ON–SN TCP connections and SN–SN TCP connections spread over seven days. We show one such representative measurement done on October 24, 2003 in Fig. 6 at Polytechnic University. Here we monitored over a period of 12 h a total of 5206 ON connections and 3850 SN with our Platform SN. We plot the distribution of connection lifetime for these two types of TCP connections in Fig. 6. The average durations of ON–SN connections and SN–SN connections are 34 min and 11 min, respectively. We also observe that a remarkable 32% of the SN–SN connections and 38% of the ON–SN connections lasted for less than 30 s. Among connections that last for at least 30 s, the average durations of ON–SN connections and SN–SN connections are 57 min and 23 min, respectively.

We attribute the large number of short lifetime ON–SN connections to two factors. First, as discussed in Section 3, at startup an ON probes candidate SNs listed in its SN refresh list with UDP packets for possible connections. The ON then initiates simultaneous TCP connections with the available SNs in its SN refresh list. Out of these successful connections, the ON selects one SN as the final choice and it disconnects from other SNs. Hence the ON–SN connection establishment process generates many short-lived ON–SN connections. A second reason for short-lived ON–SN connections is that many ONs are KaZaA-Lite clients. As described in the Introduction, KaZaA-Lite clients hop supernodes during the query process. Each such hop generates a short-lived connection.

We conjecture the short lifetime of SN–SN connections is due to (1) SNs searching for other SNs with currently small workloads, (2) long-term connection shuffling, to allow users to query a large set of SNs over long time scales and (3) at times, SNs in the overlay connect to each other just for the purpose of exchanging SN lists. The shuffling of neighbor SN–SN connections allows a larger range of the network to be explored, for example, when searching takes place over hours or days for



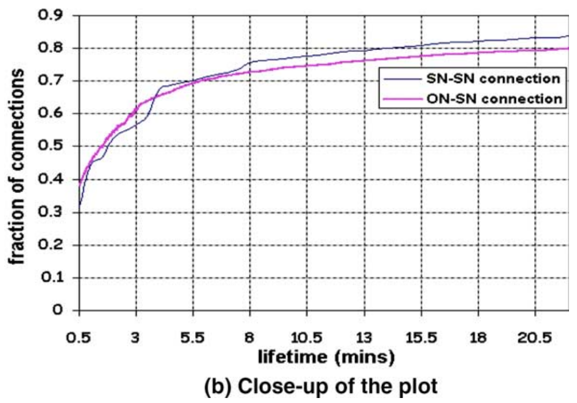


Fig. 6. *Connection lifetime distributions*. The plots on the left are for the full duration of the trace. The plots on the right are the corresponding close-ups for shorter duration which show the distribution more clearly for connections of lower lifetimes.

download lists and fragments of large files (such as movies).

4.2. Parent selection

One crucial characteristic of a two-tier overlay is the criteria that an ON employs to select a parent SN. In this section we describe results from our experiments on determining the prominent factors influencing the choice of a parent SN.

As discussed in Section 3, when an ON establishes an overlay link with a parent SN, the ON receives a list of 200 SNs from the parent SN. As already discussed, this list is a subset of all the SNs that the parent maintains in a cache. The specific SNs included in this list restricts the ON's future choices about which SNs to connect to; this in turn affects the overlay topology. Based on our measurements, we hypothesize that FastTrack peers mainly use two criteria for ON-to-SN and SN-to-SN neighbor selection, namely, workload and Locality.

4.2.1. Workload

Recall that each ON maintains a SN list Cache in the Windows Registry. For each SN in the list, this list includes four attributes: SN IP address, SN port number, SN workload, and timestamp. We now investigate the how the SN workload values in the list influence parent selection. The exact definition of SN workload is unknown, but as shown in Fig. 7 there is a clear correlation between the workload and number of connections that the SN is supporting. For our Platform SN, Fig. 7 plots the evolution of the number of on-going TCP connections as well as the evolution of the Platform SN workload. The similarity of these evolution plots is remarkable.

Recall that during startup, the ON chooses a subset of SNs (usually five SNs) from the SN cache

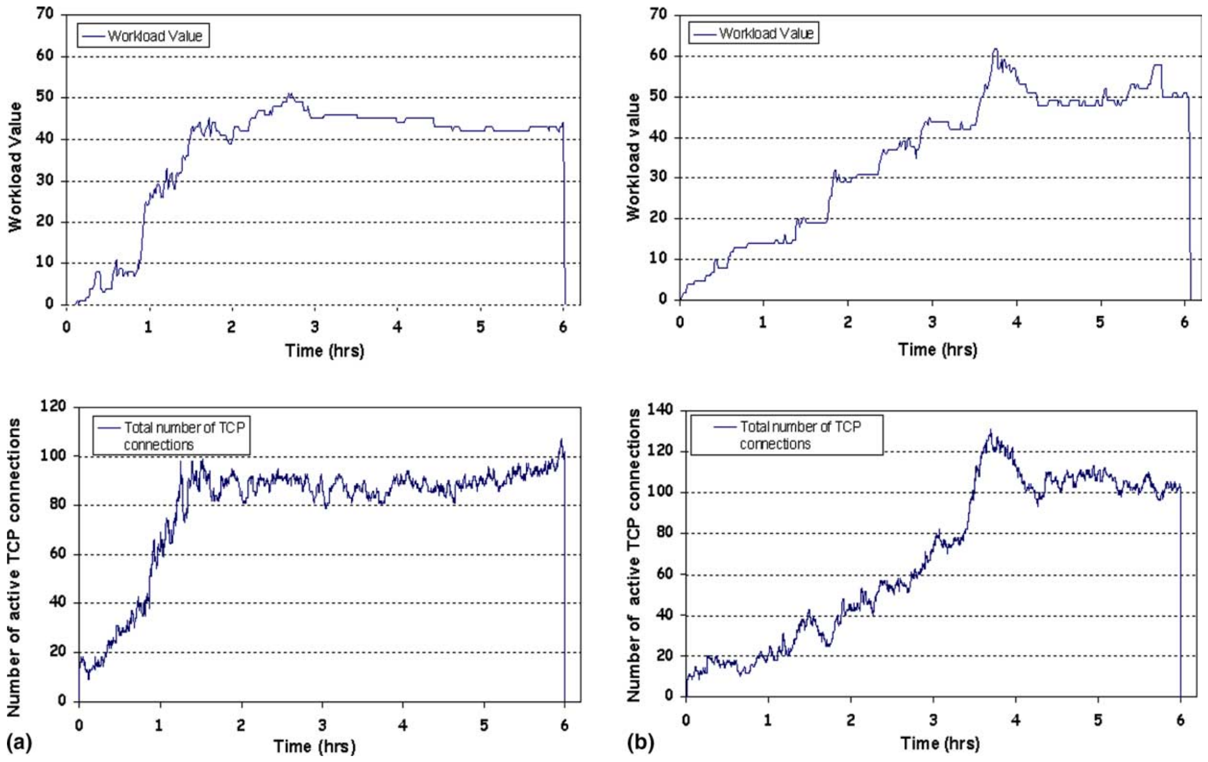


Fig. 7. Correlation between the workload value for SN and the number of TCP connections to the SN in consideration. (a) Campus connection (b) residential connection.

list as candidates for a parent SN. We hypothesize that an ON takes SN workload into account when choosing the candidates. To test this hypothesis, we force an ON in the Platform to connect to and then disconnect from the overlay. Every time the ON attempts to connect, it chooses a subset of SNs from its SN list cache in the Windows Registry and attempts connections as discussed in Section 3. We sniff this signalling traffic and determine this SN subset; we also extract the 200 SNs in the SN cache list in the ON from the Windows Registry. We then calculate the average workload of the chosen subset of SNs and the average workload of the SNs in the SN cache list. We repeat this measurement every half hour. Fig. 8 presents the results. Clearly, the FastTrack client displays a marked preference for SNs with low values for the workload.

4.2.2. Locality

We hypothesize that an ON takes locality into account when selecting a parent SN, and that SNs take locality into account when selecting neighboring SNs in the overlay. We have performed two experiments to investigate locality. The first experiment uses Ping to measure the round-trip time (RTT) from the Platform SN to the platform-external ONs and SNs to which it connects. Fig. 9 shows the distribution of these RTTs. We observe that about 60% of the SN–SN connections have RTTs less than 50 ms. It is instructive to compare these values with some typical RTT values for IP datagrams in the Internet. Transatlantic traffic between US East Coast and Europe experiences a RTT of about 100 ms, while the RTT for traffic between North America and Asia is approximately 180 ms [5]. Also it can be observed that almost 40% of the ON–SN connections have RTTs less than 5 ms, with the other 60% having RTTs more or less uniformly distributed over hundreds of milliseconds.

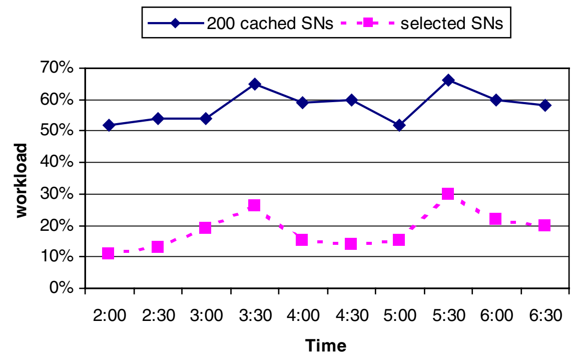


Fig. 8. *Preference for Supernodes with less Workload.* The top curve shows the average of the workloads listed for each entry in the SN list cache. The bottom curve shows the average of the workloads of the SN chosen by the FastTrack client to be probed.

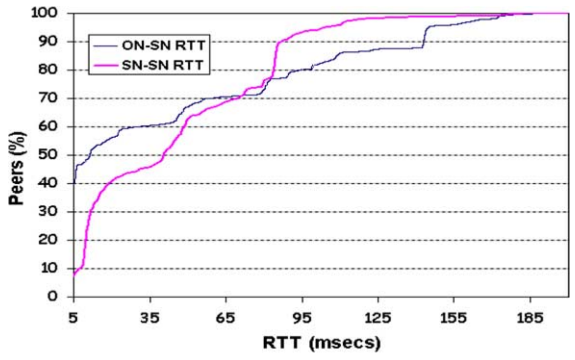


Fig. 9. *Round-Trip Time measurement.* CDF of RTTs between supernode neighbors. The *X*-axis is the RTT values and the *Y*-axis is the corresponding percentage of neighbors that have a RTT less than that value.

The second locality experiment is based on IP prefixes. This measurement is made possible with the use of the FastTrack Probing Tool, discussed in Section 3, which can connect to a pre-specified SN and retrieve its SN refresh list. In this experiment we install the FastTrack probing tool in two nodes in the US, one with a 128/8 prefix and the other with a 24/8 prefix. From each of these nodes we connect to five SNs with prefix 128/8, to another five SNs with prefix 24/8 and again to another five SNs with prefix 213/8. The first two groups of these SNs are in the US and the third group is in Sweden.

We can see from figure that a high percentage of SNs in the SN lists have similar IP prefixes as the child ON. Thus, when a SN prepares a SN list for an ON, it appears that the SN includes in the list SNs that are topologically close to the ON. The percentage is slightly less in Fig. 10(c). This is likely because the SNs based in European countries tend to have less knowledge of SNs based in the US and thus are not able to include as many

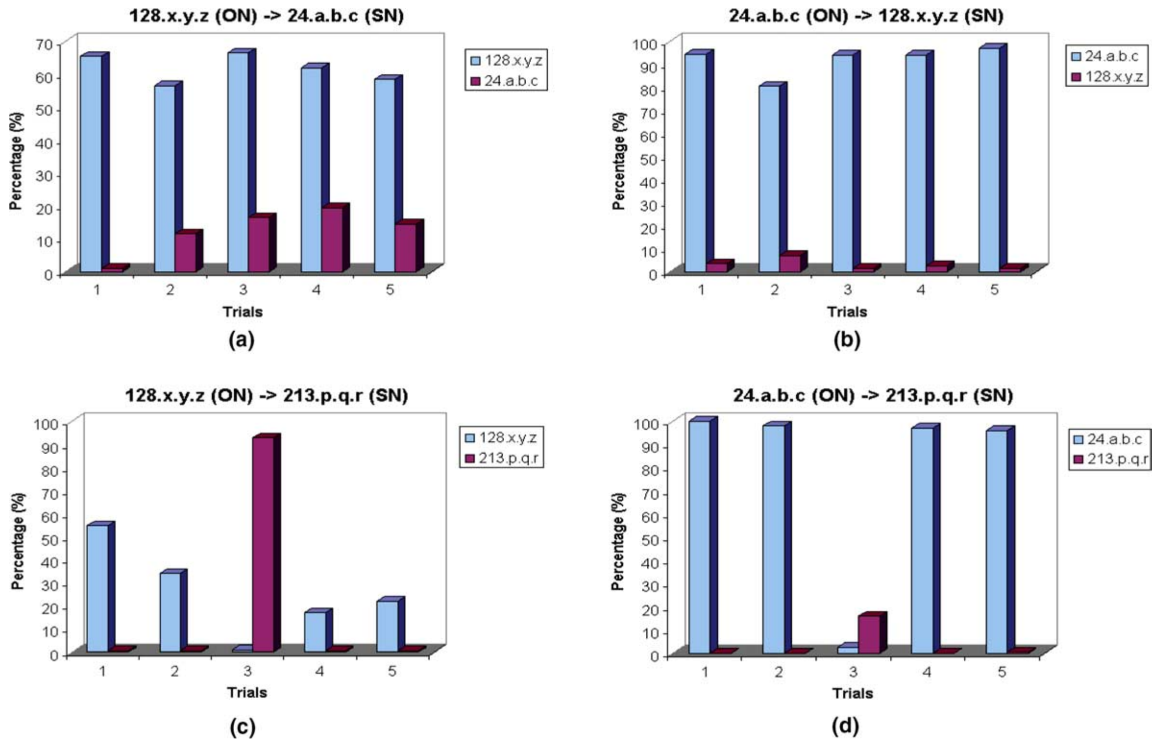


Fig. 10. *IP prefix locality*. Shows the percentage of SNs in the SN list having common IP prefix with the child ON and the parent SN. (a) The child ON IP address is from 128/8 and of parent SN is from 24/8, (b) the child ON IP address is from 24/8 and of parent SN is from 128/8, (c) the child ON IP address is from 128/8 and of parent SNs are from 213/8 and (d) the child ON IP address is from 24/8 and of parent SNs are from 213/8.

SNs matching the IP prefix of the connecting child ON. We have also discovered from other experiments that the 24/8 prefix has a very high density of SNs. This is the reason we see even the Swedish SN supplying a high percentage of SNs matching the IP prefix connecting ON.

Thus, from our two locality experiments, involving RTT and prefix matching, it appears that FastTrack takes locality into account as it dynamically constructs the overlay network. However, it is not clear whether FastTrack uses RTTs, prefixes, or some combination of the two. Although locality helps to confine the FastTrack traffic within nearby ASes, it also means that the search results tend to be localized.

4.3. Supernode lifetime

Fig. 11 shows distribution of lifetime for 965 unique SNs, monitored over a period of 65 h. The data for this figure was obtained with the FastTrack Probing Tool (at Polytechnic University), which repeatedly sent probing packets to each of the 965 SNs every 5 min. From our experiment we determined the average lifetime of a supernode in the FastTrack overlay to be 149 min ($\simeq$2.5 h).

4.4. Firewall evasion and NAT circumvention

Earlier KMD clients used the default port number 1214. Thus, with the earlier versions, whenever Peer A wanted to connect with Peer B, it connected to port 1214. With this fixed port number, administrators of campus and corporate networks could easily configure their firewalls to prevent internal FastTrack peers from connecting to external FastTrack peers. Later versions (KMD v2.0+ and KaZaA-Lite) employ dynamic port numbers to evade firewalls. Here, each peer chooses its

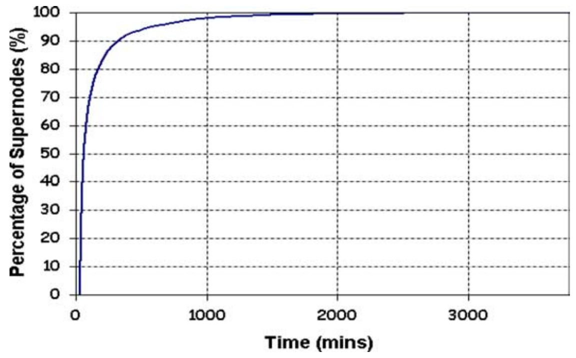


Fig. 11. *CDF as percentage of lifetime of supernodes.* 965 supernodes were monitored over 65 h for their lifetime. The average lifetime was found to be around 149 min (2.5 h).

own random port number and advertises it to other peers in the overlay. Specifically, when an ON establishes a link with a parent SN, it informs the parent SN of its port number. Furthermore, the SN refresh lists sent among the peers also advertise the port numbers of the SNs. Based on the 19,637 SN addresses we collected, we found only 707 SNs (3.6%) use the default 1214 port. 18,887 SNs (96.3%) use the non-default port numbers from 1024 to 65535. Ten SNs even use the 80 port number. Since the FastTrack port numbers are dynamic, it is very difficult to block FastTrack connections, unless a very rigid filtering policy is employed at the firewall.

The reality of today's Internet is that a large fraction of peers reside behind NATs. In fact our measurements indicate that roughly 30% of the FastTrack peers are behind NATs. This is problematic for P2P networks, where in principle, every peer should be capable of serving (i.e., uploading) files. In particular, if peer A wants to download a file from a NATed peer B, peer A cannot initiate a direct TCP connection to peer B and send a request for a file. FastTrack's two-tier hierarchy provides a mechanism to partially solve this problem. In FastTrack, when peer A sees that peer B has a private NAT address, instead of sending a request directly to peer B, it sends the request to peer B's parent SN. The parent SN then sends a message to peer B, indicating that it should initiate a connection directly back to peer A. With this connection in place, the file can be sent over the connection from peer B to peer A. This technique of using an intermediate peer which already has a TCP connection in place to the NATed peer is called *connection reversal* [13]. Our measurement studies have shown that FastTrack implements connection reversal.

4.5. Index management

As described in Section 2, on joining the FastTrack network, an ON uploads metadata information contained in the DBB file to its parent SN. We did experiments to measure the distribution of the amount of this metadata uploaded to the Platform SN (at Polytechnic University) from the connected ON sessions. The trace combines experimental data from a total of 894 ON–SN sessions. Fig. 12 shows the cumulative distribution function of the metadata uploaded to the Platform SN with respect to the percentage of ON–SN connections responsible for it. It can be observed from the plot that 13% of the ON peers are responsible for over 80% of the metadata uploaded. It is interesting to compare this data with results reported in [21], wherein on University of Washington campus, 8.6% of FastTrack peers were serving 80% of the requests.

Another important index management strategy FastTrack SNs employ is to purge metadata of any child ON as soon as it disconnects from the parent. We verified this by having a child connect to a SN, upload meta data to the SN, and then later disconnect from the SN. When querying the SN, from a second ON, for the meta data, the second ON would get a response as long as the first ON remained connected to the SN. As soon as

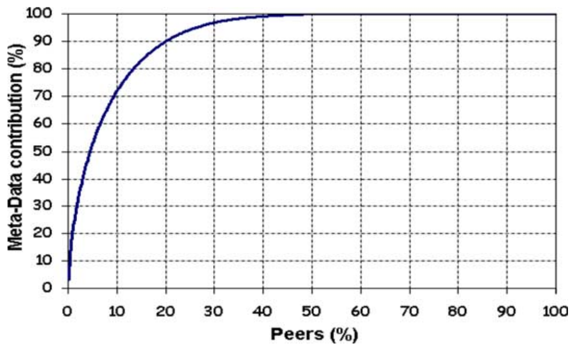


Fig. 12. CDF of amount of metadata uploaded to a SN.

the first ON disconnected, the SN ceased to receive responses for its query.

Independently, we also attempted to observe whether SN–SN traffic between neighbors in FastTrack involves exchange of metadata information collected from their respective child ONs. We then sniffed traffic between our Platform SN and other neighboring SNs of the FastTrack overlay, and no index information exchange was observed. The advantage of exchanging metadata information in the SN overlay is that it will decrease the amount of query traffic required to be forwarded among the SNs. Also it will possibly make the system more robust to temporary parent SN failures in the sense that content transfer requests can still be directed to child ONs in absence of their parent SN. This is because other SNs may have the metadata information about these child ONs available. On the other hand, the biggest disadvantage of exchanging metadata information among SNs is the complexity and bandwidth cost of implementing a distributed gossiping algorithm which will be needed to maintain the consistency of the metadata information.

5. Basic design principles for unstructured P2P file sharing systems

FastTrack is one of the most successful large-scale P2P applications to date. Its success is partially due to many of the design decisions that were made for its overlay. We now leverage our measurement results to set forth a number of key principles for the design of an unstructured P2P overlay. These principles can serve as broad guidelines to advance the state of the art in design of P2P systems.

1. *Distributed design:* Unlike Napster, FastTrack does not rely on infrastructure servers; essentially all of its nodes run on user peers. A distributed design has a number of advantages: no need for infrastructure equipment and maintenance; resilience to faults; and resilience to legal attacks (that is, FastTrack cannot be brought down as Napster was by simply unplugging a central server).
2. *Exploiting heterogeneity:* Peers differ in availability, bandwidth connectivity, CPU power, and NATed access. FastTrack was one of the first P2P systems to exploit this heterogeneity by organizing the peers into two classes, super nodes (SNs) and ordinary nodes (ONs). SNs are generally more powerful in terms of availability, bandwidth, processing, and non-NATed accessibility. The powerful peers should naturally bear more of the workload due to signaling and querying traffic. FastTrack assigns more responsibilities to the SNs. In particular, the SNs process, distribute, and respond to query traffic; and the SNs process index-maintenance traffic and overlay maintenance traffic.
3. *Load balancing:* In a two-layer hierarchical design such as FastTrack, the upper-tier nodes (i.e., the SNs) process the large majority of the signaling traffic (overlay-maintenance traffic and index-maintenance traffic) as well as the query traffic. In order to not overwhelm any subset of SNs, care should be taken to balance the load of this traffic across all the upper-tier nodes. However, it is difficult to equally balance this traffic as peer behavior is unpredictable. To achieve approximate balance, the overlay can be designed so that each SN has roughly the same degree in the overlay (that is, has roughly the same number of TCP connections to ON and SN neighbors). From our measurement results in Section 4 we found that an ON selects a parent SN with a relatively small value of "workload". Since we also showed there is a strong correlation between workload and the degree of the SN, FastTrack attempts to distribute the signaling and querying load across the SNs.
4. *Locality in neighbor selection:* Ideally, the neighbors in an overlay network should be close in terms of latency and network topology [4]. Constructing overlay links with short RTTs helps to reduce query/response delays. Constructing overlays so that neighboring peers (e.g., a SN and its children) are topological close helps to confine query and download traffic within an AS (or within nearby ASes) We saw in Section 4.2.2 that locality in the form of a common IP prefix and short RTTs play in determining an

ON's parent SN as well as in the selection of SN–SN links. Although locality considerations help to confine query and download traffic to regions, we also note it may have an adverse impact on the content availability in the file-sharing system; for example, if a query from the US only reaches US SNs, which in turn only have US peers for children, then the query may not be able to locate obscure African content. Thus the advantages of locality have to be weighed against to the need for high content availability.

5. *Connection shuffling:* In a P2P file sharing system, a user may repeatedly query for an obscure file over a period of days. By shuffling the links in the overlay, a larger set of SNs can be visited for an extended search period. Similarly, during a download, a peer server providing the download may disconnect in the middle of the download. Many P2P file sharing systems, including FastTrack, automatically and repeatedly search for a new copy of the file. Shuffling the links in the overlay helps the P2P system to find a replacement copy to complete the download. We discussed in Section 4.1.2 that SNs engage in shuffling of neighbor connections. Thus, a design principle in building unstructured P2P file sharing systems with limited scope queries (such as FastTrack) is that of overlay shuffling, which broadens the scope of a query when repeatedly querying over long time periods.
6. *Efficient gossiping algorithms:* In a two-tier distributed P2P system, it is critical that the SNs learn about the other SNs in the network, so that they can shuffle connections as well as find new SNs when existing connections leave. Thus the SNs need to *gossip* SN lists to each other. On one hand, it is desirable that the SNs frequently exchange gossiping information, so that the SNs have reasonably accurate knowledge about the SNs that are currently operating. On the other hand, it is desirable to minimize the amount of overlay maintenance traffic, as this traffic can be a burden on the SNs. Our experiments have enabled us to determine that FastTrack's signalling protocol makes explicit provisions to this end. Specifically, one of the fields in the SN refresh list is the "freshness" field. This value enables the peers (ONs and SNs) to estimate the freshness of the SN availability information. We have observed that a newly connecting ON completely ignores SNs that have a low value of "freshness" regardless of the proximity or workload values of the SN.
7. *Firewall avoidance and NAT circumvention:* Although not typically addressed in the P2P research literature, a P2P file sharing will not thrive unless it takes explicit measures to deal with firewalls and NATs, which are prevalent throughout the Internet. As discussed in Section 4, techniques like port blocking will not succeed as FastTrack uses dynamic port numbers along with its hierarchical design to avoid firewall blocking. Furthermore, it uses connection reversal to allow NATed peers to share files. However, an application level signature based approach as discussed in [23] can prove to be an effective network administration tool for ISPs.

6. Related work

Recently there have been a number of P2P measurement studies, although to our knowledge none has carefully examined FastTrack. The identification of P2P specific traffic is considered in [23,9]. The accurate signature-based techniques discussed in [23] could be deployed by an ISP to identify and filter illicit P2P traffic. An analysis of P2P traffic by measuring flow-level information collected at multiple border routers across a large ISP-network is done in [24]. By measuring FastTrack traffic in the University of Washington campus, the paper [6] studies file-sharing workloads and develops models for multimedia workload. A recent paper that characterizes P2P traffic is [8]. This last measurement of P2P traffic was done at the link level by reverse engineering the protocols of P2P applications and identifying characteristic strings in the payload.

A crawling system was previously developed for the Gnutella P2P network [20]. See also [7] for some additional work on crawling Gnutella and Napster. In [15], a FastTrack overlay crawler is developed, which is used to study the extent of polluted content in FastTrack.

There has also been some recent measurement work on the spread of spyware in P2P systems. In [22] the authors develop signatures for popular spyware and obtain traces of network activity within the University of Washington campus to quantify the spreading of spyware.

Scalability issues in the Gnutella network are studied in [17,2]. The latter presents a detailed analysis of how scalability is improved with flow control, dynamic topology adaptation, one-hop replication, and node heterogeneity. The paper [16] studies the general problem of search and replication strategies in unstructured P2P networks. It proposes a query algorithm based on multiple random walks which is shown to be as fast as Gnutella's query flooding method but reduces the network traffic by up-to two orders of magnitude. The paper [3] evaluates and compares different replication strategies in unstructured P2P networks. It identifies that uniform and proportional replication strategies yield sub-optimal performance and then proposes an optimal replication strategy based on square root replication.

The recent paper [27] studies the advantages of design of unstructured P2P systems based on super-peers (SNs in FastTrack). The paper explores a number of important questions such as the potential drawbacks of super-peers, ways of improving reliability of super-peers, and the maximum number of children a super-peer should entertain to optimize efficiency. The paper [26] argues for a hybrid architecture for P2P systems, whereby structured search techniques are used to index and locate rare items, and flooding-based techniques are used for locating highly replicated content.

7. Summary

We conclude this paper with a short summary of our findings. The supernodes form the backbone of the FastTrack network. There are roughly 25,000–40,000 supernodes; the average supernode lifetime is about 2.5 h, although these lifetimes greatly vary across supernodes. Each supernode maintains a list of SNs it believes to be up. The SNs frequently exchange subsets of these lists with each other. Thus, the FastTrack backbone is self-organizing and is managed with a distributed, but proprietary, gossip algorithm. SNs establish both short-lived and long-lived TCP connections with each other. The SNs shuffle the long-lived connections (with average duration of 23 min), which improves search performance when search is carried out over long periods (hours) as is often done in P2P file sharing. Each SN has about 40–60 connections to other SNs at any given time. Each SN has about 60–150 children ONs at any given time. Each SN maintains an index, storing the metadata of the files its children are sharing. SNs do not exchange metadata with each other.

When a user first acquires a FastTrack client, the client comes pre-installed with a list of candidate SNs. When the client is executed, the client connects with one or more the SNs in this list and obtains refresh lists. It appears that the entries in these refresh lists are biased with locality; the provided SNs are close to the ON with respect to various locality metrics. When an ON obtains a new list of SNs, it modifies its own cached list. Thus the ON-to-SN connections are formed in a decentralized, distributed manner and appear to take locality into account.

FastTrack has a life of its own, without requiring any intervention from a centralized authority. Unlike Napster, FastTrack cannot be shut down by simply pulling the plug on a centralized server farm. Thus, FastTrack will likely persist for the foreseeable future. Finally, we conjecture that there is significant room for improving the search performance in two-tier unstructured P2P file sharing systems.

Acknowledgment

We gratefully acknowledge help of ZhongQiang Chen for valuable feedback and discussions.

References

[1] A GiFT plugin for FastTrack. Available from: <http://developer.berlios.de/projects/gift-fasttrack/>.

[2] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, S. Shenker, Making Gnutella-like P2P systems scalable, in: ACM SIGCOMM, Karlsruhe, Germany, August 2003.

[3] E. Cohen, S. Shenker, Replication strategies in unstructured peer-to-peer networks, in: Proceedings of the ACM SIGCOMM Conference, Pittsburgh, PA, USA, August 2002.

[4] L. Garces-Erce, K.W. Ross, E. Biersack, P. Felber, G. Urvoy-Keller, TOPLUS: topology centric lookup service, in: Fifth International Workshop on Networked Group Communications (NGC'03), Munich, Germany, September 2003.

[5] Global Crossing's IP Network Performance. Available from: <http://ipstats.globalcrossing.net/>.

[6] K.P. Gummadi, R.J. Dunn, S. Saroiu, S.D. Gribble, H.M. Levy, J. Zahorjan, Measurement, modeling, and analysis of a peer-to-peer file-sharing workload, in: Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP-19), New York, USA, October 2003.

[7] K.P. Gummadi, S. Saroiu, S.D. Gribble, A measurement study of peer-to-peer file sharing systems, in: Proceedings of Multimedia Computing and Networking, January 2002 (MMCN'02), San Jose, CA, USA.

[8] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, M. Faloutsos, Is P2P dying or just hiding? IEEE Globecom, Dallas, Texas, November 2004.

[9] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, M. Faloutsos, File-sharing in the Internet: a characterization of P2P traffic in the backbone, Technical Report, November 2003.

[10] FastTrack Homepage. Available from: <http://www.kazaa.com>.

[11] FastTrack Lite 2.10. Available from: <http://www.k-lite.tk/>.

[12] FastTrack P2P FastTrack File Formats. Available from: <http://home.hetnet.nl/~frejon55/>.

[13] J. Kurose, K.W. Ross, Computer Networking: A Top-Down Approach Featuring the Internet, Addison-Wesley, 2005.

[14] N. Leibowitz, M. Ripeanu, A. Wierzbicki, Deconstructing the Kazaa network, in: 3rd IEEE Workshop on Internet Applications (WIAPP'03), Santa Clara, CA, USA, June 2003.

[15] J. Liang, R. Kumar, Y. Xi, K.W. Ross, Pollution in P2P file sharing systems. IEEE INFOCOM, Miami, FL, March 2005.

[16] Q. Lv, P. Cao, E. Cohen, K. Li, S. Shenker, Search and replication in unstructured peer-to-peer networks, in: 16th Annual ACM International Conference on Supercomputing, New York City, NY, USA, June 2002.

[17] Q. Lv, S. Ratnasamy, S. Shenker, Can heterogeneity make Gnutella scalable? in: The 1st International Workshop on Peer-to-Peer Systems (IPTPS), Cambridge, MA, USA, March 2002.

[18] Overpeer Inc. Available from: <http://www.overpeer.com>.

[19] Regional characteristics of P2P. Available from: <http://www.sandvine.com>.

[20] M. Ripeanu, I. Foster, A. Iamnitchi, Mapping the Gnutella network: properties of large-scale peer-to-peer systems and implications for system design, IEEE Internet Computing Journal 6 (1) (2002) 50–57.

[21] S. Saroiu, K.P. Gummadi, R.J. Dunn, S.D. Gribble, H.M. Levy, An analysis of Internet content delivery systems, in: Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002), Boston, MA, USA, December 2002.

[22] S. Saroiu, S.D. Gribble, Henry M. Levy, Measurement and analysis of Spyware in a University environment, in: Proceedings of the First Symposium on Networked Systems Design and Implementation (NSDI '04), San Francisco, CA, March 2004.

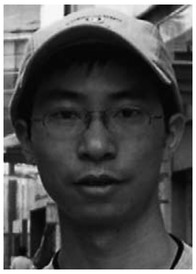
[23] S. Sen, O. Spatcheck, D. Wang, Accurate, scalable in-network identification of P2P traffic using application signatures, in: Proceedings International WWW Conference, New York, USA, May 2004.

[24] S. Sen, J. Wang, Analyzing peer-to-peer traffic across large networks, ACM/IEEE Transactions on Networking 12 (2) (2004) 219–232.

[25] Sig2dat tool for FastTrack network. Available from: <http://www.geocities.com/vlaibb/tools.html>.

[26] B. Thau Loo, R. Huebsch, I. Stoica, J. Hellerstein, The case for a hybrid P2P search infrastructure, in: The 3rd International Workshop on Peer-to-Peer Systems (IPTPS), San Diego, CA, USA, February 2004.

[27] B. Yang, H. Garcia-Molina, Designing a super-peer network, in: 19th International Conference on Data Engineering, Bangalore, India, March 2003.



Jian Liang received his B.S. from Hunan University, China, in 1995. He is pursuing Ph.D. at the Polytechnic University, New York, U.S.A. since 2001. His research interests include overlay networking, measurement and security issues of P2P systems.

Rakesh Kumar received the B.Tech degree from the Indian Institute of Technology Guwahati, India, in 2002. He is currently working towards the Ph.D. degree at the Polytechnic University, New york, U.S.A where is a research assistant in the Center for Advanced Technology in Telecommunications. His research interests include overlay networks, stochastic modeling of P2P systems and economics of P2P systems.

Keith W. Ross joined Polytechnic University as the Leonard J. Shustek Professor in Computer Science in January 2003. Before joining Polytechnic University, he was a professor for five years in the Multimedia Communications Department at Eurecom Institute in Sophia Antipolis, France. From 1985 through 1997, he was a professor in the Department of Systems Engineering at the University of Pennsylvania.

Professor Ross has worked in peer-to-peer networking, stochastic modeling, video streaming, multi-service loss networks, content distribution networks, voice over IP, optimization, queuing theory, optimal control of queues, and Markov decision processes. He is currently associate editor for IEEE/ACM Transactions on Networking and has served on numerous journal editorial boards and conference program committees.

Professor Ross is co-author (with James F. Kurose) of the popular textbook, Computer Networking: A Top-Down Approach Featuring the Internet, published by Addison-Wesley (preliminary edition in 1999, first edition in 2000, second edition in 2002, third edition in 2004). The text is used by over 200 US universities each academic year, is widely used internationally, and has been translated into ten languages. Professor Ross is also the author of the research monograph, Multiservice Loss Models for Broadband Communication Networks, published by Springer in 1995.

From July 1999 to July 2001, Professor Ross took a leave of absence to found and lead Wimba, an Internet technology start-up. Wimba develops and markets Java-based asynchronous and synchronous voice-over-IP technologies, primarily for the on-line education and language learning markets. In April 2004, Wimba merged with Horizon-Live, a multimedia e-learning company based in New York City.