# Understanding KaZaA

Jian Liang
Department of Computer and Information Science
Polytechnic University
Brooklyn, NY 11201
Email: jliang@cis.poly.edu
Phone: 1-646-552-6788

Rakesh Kumar
Department of Electrical and Computer Engineering
Polytechnic University
Brooklyn, NY 11201
Email: rkumar04@utopia.poly.edu
Phone: 1-347-244-5078

Keith W. Ross
Department of Computer and Information Science
Polytechnic University
Brooklyn, NY 11201
Email: ross@poly.edu
Phone : 1-718-260-3859

***Abstract*— Both in terms of number of participating users and in traffic volume, KaZaA is one of the most important applications in the Internet today. Nevertheless, because KaZaA is proprietary and uses encryption, little is understood about KaZaA's protocol, architecture and signaling traffic. We have built a measurement platform for collecting and measuring KaZaA's signaling traffic. These measurements provide insight on KaZaA's architecture, protocol, and overlay behavior. The reader should take away from this paper a deep understanding of one of the largest distributed systems ever to be deployed in the Internet.**

**Keywords: P2P Networks, File Sharing Systems, KaZaA, Topology, Traffic Measurement.**

## I. Introduction

On a typical day, KaZaA has more than 3 million active users sharing over 5,000 terabytes of content. On the University of Washington campus network in June 2002, KaZaA consumed approximately 37% of all TCP traffic, which was more than twice the Web traffic on the same campus at the same time [4]. Thus, both in terms of number of participating users and in traffic volume, KaZaA is one of the most important application in the Internet today.

An understanding of the KaZaA's protocol, architecture and signaling traffic is of critical importance for the P2P research community. With over 3 million satisfied users, KaZaA is significantly more popular than Napster or Gnutella ever was. Sandvine estimates that in the US 76% of P2P file sharing traffic is KaZaA/FastTrack traffic and only 8% is Gnutella traffic [16]. Thus, any new proposal for a P2P file sharing system should be compared with the KaZaA benchmark. To date, little has been known about the specifics of the KaZaA design. For example, the SIGCOMM 2003 paper [8] proposes a new P2P file sharing architecture, but does not provide a rigorous comparison of the design with that of KaZaA: "while KaZaA's appears to offer better scaling than Gnutella, its design has neither been documented nor analyzed".

It is also important for upper- and lower-tier ISPs to acquire a thorough understanding KaZaA traffic. Because KaZaA generates vast quantities of traffic, networking engineers, who dimension the network and introduce content distribution devices such as caches, need a basic understanding of how KaZaA operates. Although there has been recent work in analyzing the file-sharing workload in KaZaA [4] and [7], little is published about KaZaA's protocol or software architecture.

The goal of this paper is to provide the research community with a deep understanding of how KaZaA operates. Because KaZaA uses a proprietary protocol with encryption, this is a daunting task. We have built a measurement platform that has enabled us to gain significant insights into KaZaA. The paper focuses on the KaZaA overlay network, search mechanism, index management system, signaling traffic, and software architecture. The paper addresses neither KaZaA's downloading protocol (for example, the parallel downloading and queuing) nor its incentive scheme for encouraging uploaders. The paper is complementary to [4] and [7], which focus on KaZaA file-sharing traffic. The reader should obtain from this paper a deep understanding of one of the largest distributed systems ever to be deployed in the Internet.

## II. Overview of KaZaA Design

KaZaA Web site [1] provides a rudimentary description of how KaZaA works. Moreover, various articles, Web sites, and message boards provide additional scraps of information. We begin with an overview of KaZaA overlay network and search mechanism. This overview combines publicly available information with some of our own investigations, which are described in more detail in Section III.

KaZaA resembles Gnutella in that it does not use a dedicated server for tracking and locating content. However, unlike Gnutella, not all peers are equal. KaZaA has two classes of peers, Ordinary Nodes (ONs) and Super Nodes (SNs). The more powerful peers are SNs and have greater responsibilities. As shown in Fig. 1, each ON is assigned to a SN. When an ON launches the KaZaA application, the ON establishes a TCP connection with a SN. The ON then uploads to its SN metadata for the files it is sharing. This allows the SN to maintain a database which includes the identifiers of all the files its children are sharing, metadata about the files, and the corresponding IP addresses of the ONs holding the files. In this way, each SN becomes a (mini) Napster-like hub. But in contrast with Napster, a SN is not a dedicated server (or server farm); instead, it is typically a peer belonging to an individual user.

One important design lesson learned from the KaZaA experiment is that that large-scale P2P systems should exploit the heterogeneity of the peers. Peers differ in up times, bandwidth connectivity, and CPU power. Moreover, some peers are behind NATs and therefore have restricted file sharing and and database capabilities. To exploit the heterogeneity, the peers should be organized in a hierarchy of two or more peers, with the peers in the higher tiers being more powerful in terms of connectivity, bandwidth, processing, and non-NATed accessibility.

One important design decision for a two-tier hierarchical system (such as KaZaA) is whether a SN should only track the content of its children, or whether it should track the content its children *and* the content of the children its of neighboring supernodes. In the latter case, SNs would exchange with each other the metadata from their children. *Our measurement work has determined that each SN database only keeps records for files located in its direct children; it doesn't track the files that are in ONs under other SNs.* However, to our knowledge, there has been no evaluation of this design choice to date.

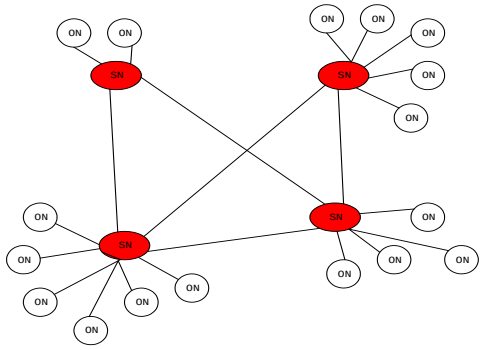



Fig. 1. *Supernode and Ordinary nodes in KaZaA network*

For each file that it is sharing, the metadata that an ON uploads to its 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) [12]. The file descriptors are used for keyword matches during querying. The ContentHash plays an important role in the KaZaA architecture. KaZaA hashes every file to a hash signature, which becomes the ContentHash of the file. In the most recent version of KaZaA, ContentHash is the only tag used to identify a file in an HTTP download request. If a download from a specific peer fails, the ContentHash enables the KaZaA client to search for the specific file automatically, without issuing a new keyword query.

When a user wants to find files, the user's ON sends a query with keywords over the TCP connection to its SN. For each match in its database, the SN returns the IP address and metadata corresponding to the match. 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. *Our measurement work has determined that SNs often change their SN-to-SN connections on a time scales of tens of minutes.* This shuffling allows a larger range of the network to be explored, for example, when searching takes place over hours or days for download lists and fragments of large files (such as movies).

A KaZaA peer has the following software components:

1) The KaZaA Media Desktop (KMD).
2) Software environment information stored in the Windows Registry.
3) DBB files, with each DBB file containing metadata for the files that the peer is willing to share. An active KaZaA process permanently monitors the local folders that are shared; file add, delete, is reflected in the DBB file [12].
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.

*Our measurement work has determined that each KaZaA 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.

*We have determined that, as part of the signalling traffic, KaZaA nodes frequently exchange with each other lists of supernodes. ONs keep a list of up 200 SNs whereas SNs appear to maintain lists of thousand of SNs. When a peer A (ON or SN) receives a supernode list from another peer B, peer A will typically purge some of the entries from its local list and add entries sent by peer B.* By frequently exchanging supernode lists, nodes maintain up-to-date lists of active SNs. Moreover, as we shall see in Section 4, these lists are used for building locality-influenced overlays.

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 KaZaA 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 measurement work that after receiving all the replies from its parent SN, the ON often disconnects and connects with a new SN, and resends the query to the new SN.* During a specific search, the ON may hop to many SNs. The ordinary node typically maintains the TCP connection with the last SN in the sequence of hops, until another search is performed. *We have learned that during each hop, the ON resends its metadata to the new SN, and the previous SN removes the ONs metadata.*

One could imagine the following three-step search process: (1) the system first searches for blocks of metadata that match the keywords (a file has an associated "block" of metadata); (2) after receiving the metadata blocks, the user selects for downloading the file that interests the user the most; (3) the system uses the ContentHash of the file to locate peers that contain the file. Note that in this three-step procedure, the search for ContentHash of the desired file and search for the location of the desired file are decoupled. In the context of this of three-step procedure, an important design decision for index management is whether a SN should continue to cache (for up to some timed period) the metadata of an ON after an ON disconnects from it. Caching metadata is particularly compelling in the context of KaZaA-lite, in which ONs are frequently changing SNs. *However, our investigations have determined that SNs do not cache metadata when ONs disconnect from them.* Furthermore, in the search process, KaZaA does not use the above three-step procedure, but instead directly returns nodes which contain files whose metadata matches the keywords. However, as stated earlier, when a peer requests a file from another peer, the requesting peer identifies the file with the ContentHash; and when a download from a specific peer fails, the ContentHash enables the KaZaA client to search for the specific file automatically, without issuing a new keyword query.

We will describe our measurement Testbed and additional measurement insights in the next section. We conclude this section with a brief descriptoin of some other ongoing KaZaA projects. The FastTrack File Format project [12] has determined the syntax and semantics of KaZaA system files, including the DBB file, the DAT file, and the Supernode List Cache. The project [12] has also investigated the KaZaA information that is stored in the Windows Registry at HKLM\Software\KaZaA\ConnectInfo\KazaaNet. The Sig2dat tool project [13] makes available a tool for obtaining the KaZaA ContentHash of any file. This tool is increasingly being used by KaZaA users, who post file names and corresponding

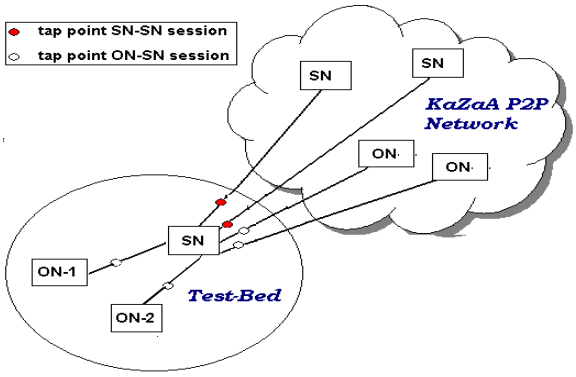



Fig. 2. *Test-Bed Configuration.* We show few of the many connections that test-bed SN has to the rest of the KaZaA P2P network.

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 [15]. Finally, the impressive giFT project [3] has reverse engineered KaZaA's encryption algorithms, so that users of giFT-FastTrack can search and download files form the KaZaA network.

## III. Measurement Work

As shown in Fig. 2, we built a test-bed consisting of three workstations, each with a KMD version 2.0 client installed. We patiently waited until KaZaA promoted one of the three nodes to a SN. These workstations are connected to the Polytechnic University campus network. At startup all three workstations functioned as ONs in the KaZaA 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 ONs so that each of the two registries listed only the promoted SN. In this manner, we forced both ONs to become children of the SN. The test-bed also connects to the larger KaZaA P2P network through connections from the test-bed SN to other SNs and other ONs in the KaZaA network. We then deployed software traffic monitors to capture *all of the traffic inbound to and outbound from the SN for each of its ON and SN connections*. We then did an offline traffic analysis based on our understanding of the KaZaA signalling protocol.

We have also developed our own version of a KaZaA client, which emulates the behavior of the official KMD client for signalling traffic. We use it to do relevant experiments with the 200-node SN lists sent from a SN to its children. Our client can fully participate in the FastTrack network; specifically it has the ability to connect to an arbitrarily specified SN and then retrieve a SN list from the specified SN.

Admittedly our results may be partially biased since the measurements were taken from one location. However, we believe that this location (a university campus in the US) is a representative location for a KaZaA SN; furthermore, we performed the experiments on several different days. Thus we do not expect the bias to be significant.

### A. Topology Structure

We first explored the degree of connectivity of a SN. Specifically, from our campus SN node, we studied the number of simultaneous connections to ONs and to other SNs. Fig. 3 presents measurement results taken on four different days for different durations. Each graph in Fig. 3 shows the evolution of the number of simultaneous TCP connections from ONs to our test SN and from SNs to our test SN. For both the ON and SN connections, the number of connections begins at one and climbs to a threshold, around which it subsequently vacillates. For the number of simultaneous SN-ON connections, depending on the day, this threshold is in the 100-160 connection range. Since on a typical day there are roughly 3 million peers, we therefore speculate that there are on the order of 30,000 supernodes in the KaZaA network at any given moment. We also observe that for the number of simultaneous SN-SN connections, depending on the day, the threshold is in the 30-50 connection range. Thus, at any given moment, each supernode is roughly connected to 0.1% of the total number of supernodes.

### B. Topology Dynamics

Our measurement study has determined the KaZaA overlay is highly dynamic. Although, as observed in Fig. 3, the number of simultaneous connections vacillates around a threshold, the individual connections change frequently.

We performed measurements on the duration of ON-SN TCP connections and SN-SN TCP connections on Oct. 24, 2003. We monitored over a period of 12 hours a total of 5206 ON connections and 3850 SN TCP connections to our test-bed supernode. We plot the distribution of connection lifetime for these two types of TCP connections in Fig. 4. The average duration of a ON-SN connection and of a SN-SN connection are 34.3 mins and 11 mins, respectively. We also observe that a remarkable 35% of the SN-SN connections lasted for less than 30 seconds; the percentage is slightly more for the ON-SN connections. Among connections that last for at least 30 seconds, the average duration of a ON-SN connection and of a SN-SN connection are 56.6mins and 23.3mins, respectively.

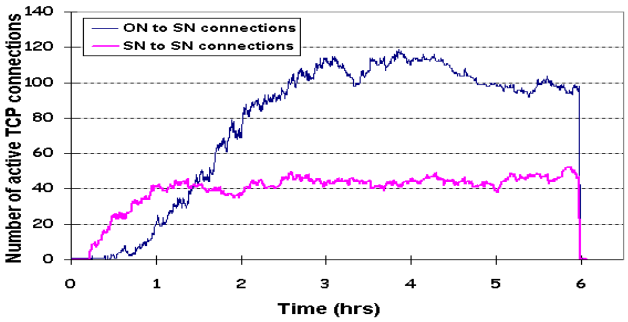
We attribute the large number of short lifetime ON-SN connections to two factors. First, we have observed that initially at startup, an ON probes candidate SNs listed in its Supernode List Cache with UDP packets for possible connections. The ON then initiates simultaneous TCP connections with the available SNs in its SN list. Out of these successful connections, the ON selects one SN as the final choice and it disconnects from other SNs. Hence, *some* ON-SN connections are short-lived. 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, and (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 list caches.

### C. Neighbor selection

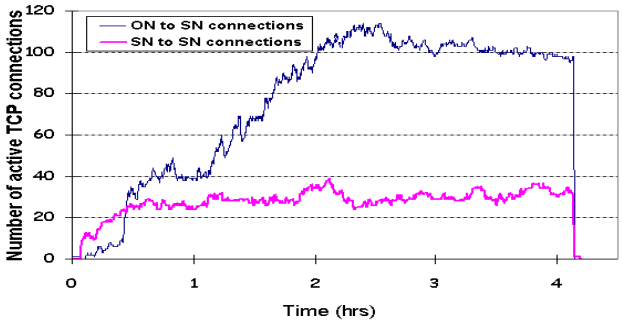
One crucial characteristic of the topology is the criteria that nodes (SNs and ONs) employ to select neighbors. In this section we describe results from our experiments on determining the prominent factors influencing neighbor selection in KaZaA P2P network.

We know that a newly connecting ON receives a list of 200 SNs from its parent SN. As already discussed, this list is a subset of all the SNs in the parent SN cache. The contents of this list sent from the SN to the ON influence the ON's future decisions about which SNs to connect to; this in turn affects the overlay topology. As discussed at the beginning of this section, we use our own version of the KaZaA client to obtain and analyze these lists.

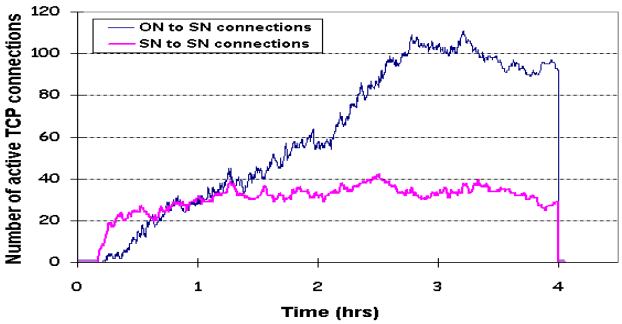
Based on our measurements, we hypothesize that KaZaA peers mainly use two criteria for ON-to-SN and SN-to-SN



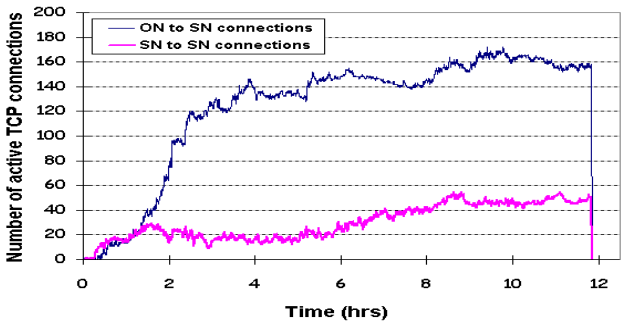

(a) session evolution, Aug. 22, 2003




(b) session evolution, Aug. 25, 2003

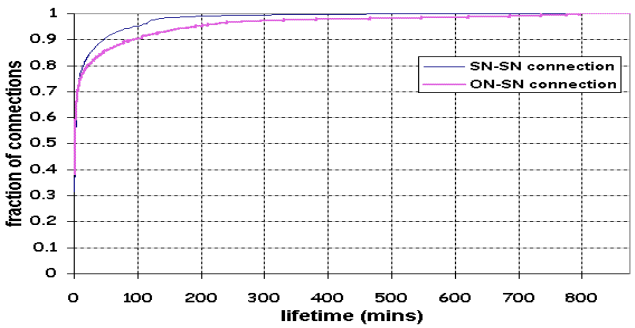



(c) session evolution, Aug 27

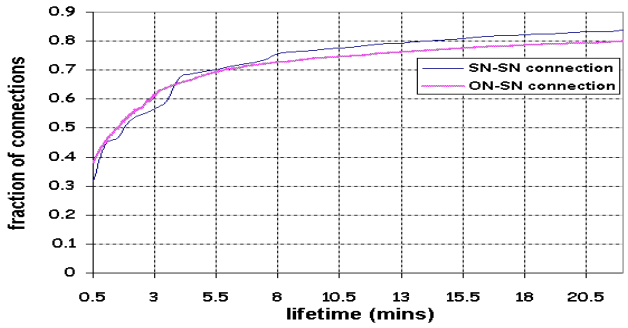



(d) session evolution, Oct 24

Fig. 3. *Evolution of SN-SN and ON-SN connections with time.*




(a) full duration plot

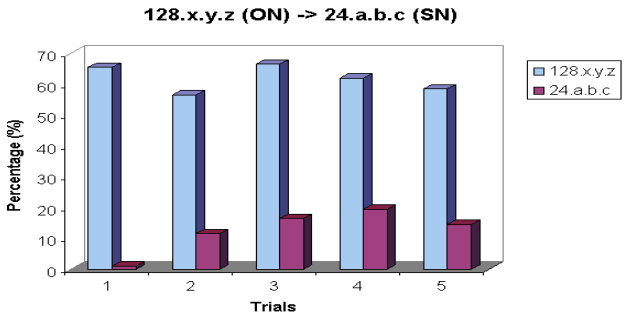



(b) close-up of the plot

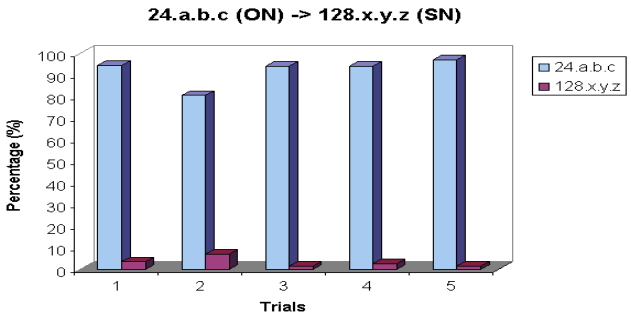
Fig. 4. *Connection lifetime distributions.* The graphs on the left are for full duration of trace. The graphs on the right are the corresponding close-ups for shorter duration which show the distribution more clearly for connections of lower lifetimes.

neighbor selection. One of these criteria is *supernode workload*. Each KaZaA ON chooses a parent SN from the local Supernode List Cache in the Windows Registry. One of the information fields in this list is the average workload of the supernode [12]. It is unclear how the value in this field is calculated. Nevertheless, in our experiments, the KaZaA client displayed a marked preference for SNs with low value for the workload field. Fig. 7 illustrates this preference.

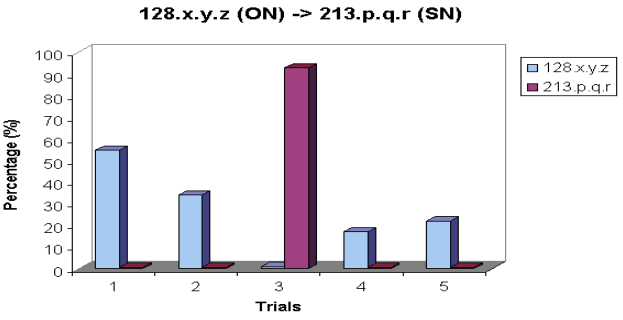
The second criteria is based on locality, that is, nodes (both ONs and SNs) appear to choose overlay neighbors that are in some sense close. We have performed two experiments to investigate locality. The first experiment uses Ping to measure the round-trip time (RTT) from the SN in our testbed to its ON and SN neighbors. Fig. 6 shows the distribution of the RTT with respect to the percentage of neighboring SN peers. We observe that about 60% of the connections between neighboring SNs have RTT less than 50 msec. It is instructive to compare these values with some typical RTT values for IP datagrams on the Internet. Transatlantic traffic between U.S East Coast and Europe experiences a latency of 100 ms, while the RTT for traffic between North America and Asia is approximately 180ms [5]. Also it can be observed that almost 40% of the



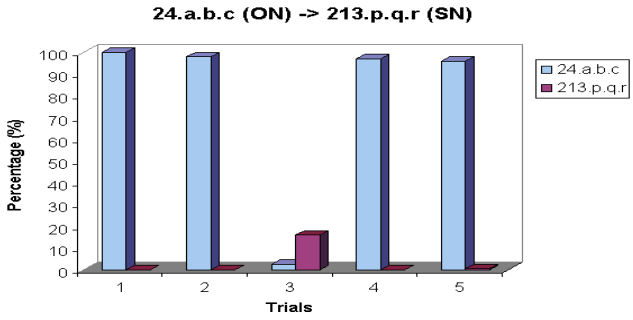

(a) The child ON IP address is from 128.x.y.z and of parent SN is from 24.a.b.c




(b) The child ON IP address is from 24.a.b.c and of parent SN is from 128.x.y.z




(c) The child ON IP address is from 128.x.y.z and parent SNs are from 213.p.q.r




(d) The child ON IP address is from 24.a.b.c and parent SNs are from 213.p.q.r

Fig. 5. *Measuring the characteristics of the IP prefixes on contents of SN lists.* The figure shows the percentage of SNs in the SN list received by the child ON having common IP prefix with the child ON and the parent SN.

ON-SN connections have a RTT less than 5 msec, with the the other 60% having RTTs more or less uniformly distributed over hundreds of milliseconds.

The second locality experiment is based on IP prefixes. Recall that when an ON connects to a parent SN, it receives a SN list. Our experiments indicate that the SNs in the list have IP prefixes that tend to correlate with the prefix of the ON. In Fig. 5 we provide the percentage of SNs in the list having common IP prefixes with the parent SN and with the connecting child ON for three different cases. In Fig. 5(a) we connect as an ON from a 128.x.y.z IP address to a 24.a.b.c SN; then we do the opposite in Fig. 5(b), connecting as an ON from a 24.a.b.c IP address to a 128.x.y.z SN. Both the child ON and the parent SN in these cases are based in United States. In Fig. 5(c) and 5(d) we connect from the 128.x.y.z and 24.a.b.c ONs respectively to a parent SN in Sweden with 213.p.q.r IP addresses.

We can see from Fig. 5(a) and Fig. 5(b) that a high percentage of SNs in the lists have similar IP prefixes as the connecting ON while this percentage drops slightly in Fig. 5(c). This is because the SNs based in European countries tend to have less knowledge of SNs based in the U.S. and thus are not able to include as large of a percentage of SNs matching the IP prefix of the connecting child ON. We have also discovered from other experiments that 24.a.b.c subnet hosts a very high density of SNs. This is the reason we see SNs even in Sweden reporting a high percentage of SNs matching the IP prefix of the connecting ON from 24.a.b.c. We conjecture that the parent SN is going to include in the SN list as many SNs as it is aware of which have a matching IP prefix with the connecting ON.

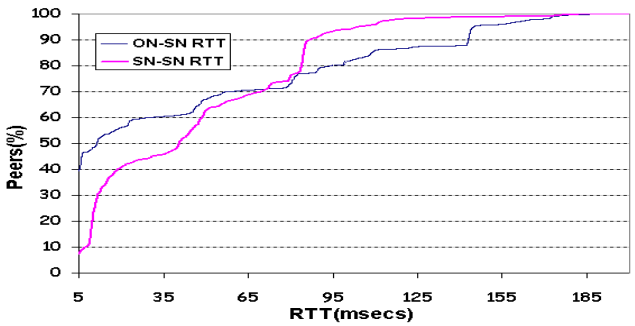



Fig. 6. *Round-Trip Time measurement.* CDF of RTTs between supernode neighbors. On X-axis we have RTT values and on Y-axis we have the corresponding percentage of neighbors who have a RTT value equal to or less than that value.

Thus it appears that KaZaA takes locality into account when dynamically creating the overlay network. Although this helps to confine the KaZaA traffic within nearby ASes, it also means that the search results tend to be local.

### *D. Supernode Lifetime*

Figure 8 shows distribution of lifetime for 965 unique supernodes, monitored over a period of 65 hours. We use UDP probe packets to track availability of SNs for the experimental period. From our experiment we determined the average lifetime of a supernode in the KaZaA overlay to be 149 mins ($\simeq$2.5hours).

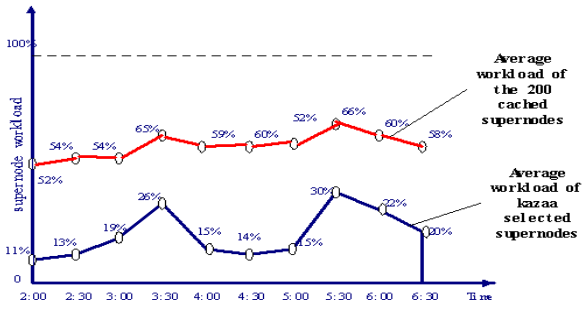



Fig. 7. *Preference for Supernodes with less Workload.* The top curve shows the average of the workloads listed for each entry in the supernode cache. The curve at the bottom shows the average of the workloads of the supernode chosen by the KaZaA client to be probed.

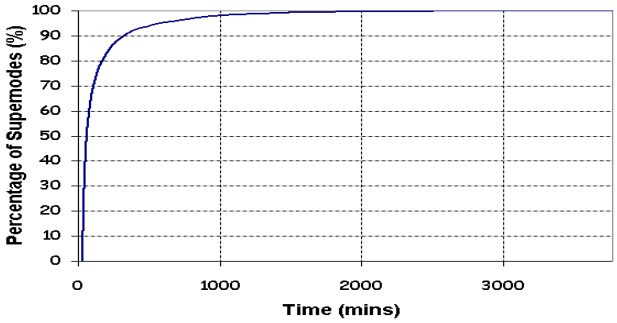



Fig. 8. *CDF as percentage of lifetime of supernodes.* 965 supernodes were monitored over 65hrs for their lifetime. The average lifetime was found to be around 149mins (2.5hrs)

### E. Magnitude of Signaling Traffic

Fig. 9 shows the evolution of bit-rate of signaling traffic with time. We show in the figure the combined bit-rate of upstream and downstream traffic at the supernode. We deduce from the collected trace data that the average upstream and downstream bandwidth consumption for signaling traffic is 161 kbps and 191 kbps, respectively. This gives insights into the amount of resources needed so that an ON can be promoted to a SN. This also explains why the majority of supernodes in the KaZaA overlay belong to the university campus networks or cable home users but very few from the DSL home users who typically have less than 128 kbps of upstream bandwidth.

As described in Section II, on joining the KaZaA 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 test-bed SN from the connected ON sessions. The trace combines experimental data from a total of 894 ON-SN sessions. Fig. 10 shows the cumulative distribution function of the meta-data uploaded onto our test-bed 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 meta-data uploaded. It is interesting to compare this data with results reported in [6], wherein on University of Washington campus, 8.6% of KaZaA peers were serving 80% of the requests.

## IV. Summary

To conclude this paper with a short summary of our findings. The supernodes form the backbone of the KaZaA network. There are roughly 30,000 supernodes; the average supernode lifetime is about 2.5 hours, although these lifetimes greatly vary

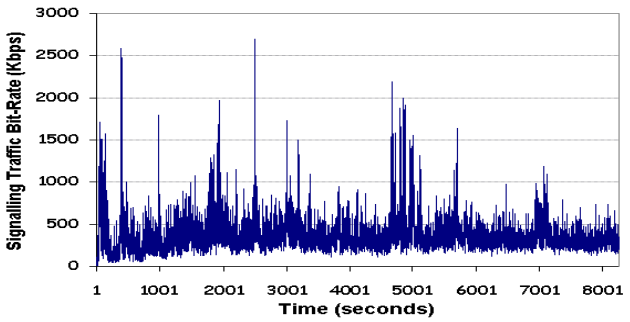



Fig. 9. *Bit-Rate Evolution of Signalling (combined up-stream and downstream) Traffic.* First 140 mins of the measurement shown in the plot. Samples placed one second apart. The bursty nature of the traffic is clearly seen.

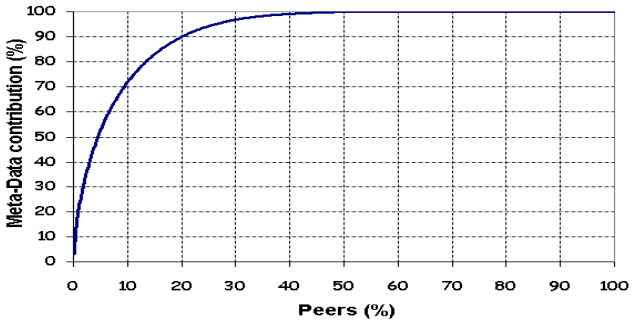



Fig. 10. *CDF of amount of Meta-Data uploaded to a supernode.* The amount of Meta-Data present at supernodes has direct correspondence with the content available for download in the P2P file-sharing network. Ideally the curve should be same as of distribution of a uniformly distributed random variable

across supernodes. Each supernode maintains a list of SNs it believes to be up. The SNs frequently exchange (possibly subsets) of these lists with each other. Thus, the KaZaA 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 minutes), which improves search performance when search is carried out over long periods (hours) as it 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 100 to 200 children ONs at any given time. Each SN maintains a database, storing the metadata of the files its children are sharing. SNs do not exchange metadata with each other.

When a user first acquires a KaZaA or KaZaA-lite client, the client comes pre-installed with a cache of candidate SNs. When the client is executed, the client connects with one or more the SNs in this list and obtains new lists. It appears that the entries in these 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.

KaZaA has a life of its own, without requiring any intervention from a centralized authority. Unlike Napster, KaZaA cannot be shut down by simply pulling the plug on a centralized server farm. Thus, KaZaA will likely persist for the foreseeable future. Many design decisions taken by the creators of KaZaA (and KaZaA-lite) seem to be have been done without careful consideration. 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] KaZaA Homepage, http://www.kazaa.com
[2] The Gnutella protocol specification v4.0. http://dss.clip2.com/GnutellaProtocol04.pdf/
[3] A GiFT plugin for FastTrack, http://developer.berlios.de/projects/gift-fasttrack/
[4] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan, "*Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload*," Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP-19), October 2003.
[5] http://ipstats.globalcrossing.net/
[6] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "*An Analysis of Internet Content Delivery Systems*," Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002), December 2002.
[7] N. Leibowitz, M. Ripeanu, and A. Wierzbicki, "*Deconstructing the Kazaa Network*," 3rd IEEE Workshop on Internet Applications (WIAPP'03). 2003. Santa Clara, CA.
[8] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, S. Shenker, "*Making Gnutella-like P2P Systems Scalable*," SIGCOMM 2003.
[9] K. Gummadi, S. Saroiu, S. Gribble, "*King estimating Latency between Arbitrary Internet End Hosts*," Proceedings of the SIGCOMM Internet Measurement Workshop (IMW 2002).
[10] KaZaA Hack 2.5, http://www.kazaahack.net/home.html
[11] KaZaA Lite 2.10, http://www.k-lite.tk/
[12] KaZaA P2P FastTrack File Formats http://home.hetnet.nl/∼frejon55/
[13] Sig2dat tool for FastTrack network, http://www.geocities.com/vlaibb/tools.html
[14] giFT-FastTrack plugin for giFT, http://giftproject.org
[15] Overpeer Inc, http://www.overpeer.com
[16] Regional characteristics of P2P, http://www.sandvine.com