Peer-to-Peer Architecture Case Study: Gnutella Network

Matei Ripeanu
Computer Science Department,
The University of Chicago
matei@cs.uchicago.edu

Abstract

Despite recent excitement generated by the P2P paradigm and despite surprisingly fast deployment of some P2P applications, there are few quantitative evaluations of P2P systems behavior. Due to its open architecture and achieved scale, Gnutella is an interesting P2P architecture case study. Gnutella, like most other P2P applications, builds at the application level a virtual network with its own routing mechanisms. The topology of this virtual network and the routing mechanisms used have a significant influence on application properties such as performance, reliability, and scalability. We built a 'crawler' to extract the topology of Gnutella's application level network. In this paper we analyze the topology graph and evaluate generated network traffic. We find that although Gnutella is not a pure power-law network, its current configuration has the benefits and drawbacks of a power-law structure. These findings lead us to propose changes to Gnutella protocol and implementations that bring significant performance and scalability improvements.

1. Introduction

Peer-to-peer systems (P2P) have emerged as a significant social and technical phenomenon over the last year. P2P systems provide infrastructure for communities that share CPU cycles (e.g., Entropia [21], SETI@Home[22]) and/or, storage space (e.g., FreeNet [23], Gnutella [19]), or that support interpersonal collaborative environments (Groove [20]). Two factors have fostered the recent explosive growth of such systems: first, the low cost and high availability of huge computing and storage resources, and second, increased network connectivity. As these trends continue, the P2P paradigm can be expected to become more popular.

Peer-to-peer networks allow individual computers to communicate directly with each other and to share information and resources without using specialized 'servers'. Early P2P applications often do not fully fit this definition. Napster, for example, relies on centralized databases to index and locate files and respond to searches; only file transfers occur directly between peers. Newer applications (e.g., Gnutella, FreeNet) are completely decentralized: each member of the network performs the same set of tasks and "serves" all other network members.

A common characteristic of this new breed of applications is that they build, at the application level, a virtual network with its own routing mechanisms. The topology of this virtual network and the routing mechanisms used have a significant influence on application properties such as performance, reliability, and, in some cases, anonymity. The virtual topology also determines the communication costs incurred when running the P2P application.

These considerations have motivated us to conduct a detailed study of the topology and protocol of a popular P2P system, Gnutella. In this study, we benefited from Gnutella's large existing user base and open architecture, and, in effect, use the public Gnutella network as a large-scale, if uncontrolled, testbed. We capture the network topology, the generated traffic, and the resources' dynamic behavior. We use our findings to evaluate costs and benefits of the P2P approach and to investigate possible improvements to the routing protocol that would allow for better scaling and increased reliability.

Recent research [1,7,8,13] shows that networks ranging from natural networks such as molecules in a cell or people in a social group to the Internet organize themselves so that most nodes have few links and a small number of nodes have many links. In [14], Albert finds that networks following this organizational pattern (power-law networks) display an unexpected degree of robustness when facing random node failures: the ability of the network to communicate is unaffected by high failure rates. However, error tolerance comes at a high price: these networks are extremely vulnerable to attacks, i.e., to the selection and removal of a few nodes that play

the most important role in assuring the network's connectivity. We show that Gnutella is similar to a power-law network, thus being able to operate in highly dynamic environments.

Related work: Distributed Search Solutions (DSS) group publishes partial results of their Gnutella research [4,5]. A number of other research projects build on DSS's data: [2] analyzes Gnutella user behavior while [6] focuses on efficient search protocols in power-law networks. T. Hong ([18]) uses medium scale simulations (up to 1000 nodes) to explore Gnutella network properties. However, our network crawling and analysis technology (developed independently of this work) goes significantly further in terms of scale (both spatial and temporal) and sophistication. The present article significantly extends previously published results: while DSS presents only raw facts about the network, here we analyze the generated network traffic and find underlying patterns in network organization.

The paper is structured as follows: the next section provides a brief enumeration of P2P design goals and Section 3 succinctly describes Gnutella's protocol. Section 4 introduces the *crawler* we developed to discover Gnutella's virtual network topology. In section 5 we thoroughly analyze the network. We conclude in Section 6 with proposed changes to the protocol and its implementations that could bring significant scalability improvements.

2. Design Goals of P2P File Sharing Systems

Most P2P file sharing applications attempt to fulfill the following design goals:

- *Ability to operate in a dynamic environment* – P2P applications operate in dynamic environments where hosts are likely to often join or leave the network. Applications must achieve flexibility to keep operating transparently despite a constantly changing set of resources.
- *Performance and Scalability* (aggregate storage size, response time, availability, query throughput) – Ideally, when increasing the number of nodes, storage space and file availability should grow linearly, response time should remain constant, while search throughput should remain high or grow.
  We believe P2P shows its full potential only on large-scale deployments where the limits of the client/server paradigm become obvious. Moreover, scalability is important as P2P applications exhibit what economists call "network effect" [10]: the value of a network to an individual user increases with the total number of users participating in the network.
- *Reliability* – External attacks should not cause significant data or performance loss.
- *Anonymity* - Anonymity is valued as a means of protecting the privacy of people seeking or providing unpopular information.

3. Gnutella Protocol Description

Gnutella nodes, called *servents* by developers, perform tasks normally associated with both SERVers and cliENTS. They provide client-side interfaces through which users can issue queries and view search results, accept queries from other servents, check for matches against their local data set, and respond with corresponding results. These nodes are also responsible for managing the background traffic that spreads the information used to maintain network integrity.

In order to join the system a new node/servent initially connects to one of several known hosts that are almost always available (e.g., gnutellahosts.com). Once attached to the network (having one or more open connections with nodes already in the network), nodes send messages to interact with each other. Messages can be *broadcasted* (i.e., sent to all nodes with which the sender has open TCP connections) or simply *back-propagated* (i.e., sent on a specific connection on the reverse of the path taken by an initial, broadcasted, message). Several features of the protocol facilitate this broadcast/back-propagation mechanism. First, each message has a randomly generated identifier. Second, each node keeps a short memory of the recently routed messages, used to prevent re-broadcasting and implement back-propagation. Third, messages are flagged with time-to-live (TTL) and "hops passed" fields.

The messages allowed in the network are:

- *Group Membership* (PING and PONG) *Messages.* A node joining the network initiates a broadcasted PING message to announce its presence. When a node receives a PING message it forwards it to its neighbors and initiates a back-propagated PONG message. The PONG message contains information about the node such as its IP address and the number and size of shared files.
- *Search* (QUERY and QUERY RESPONSE) *Messages*. QUERY messages contain a user specified search string, each receiving node matches against locally stored file names. QUERY messages are broadcasted. QUERY RESPONSES are back-propagated replies to QUERY messages and include information necessary to download a file.
- *File Transfer* (GET and PUSH) *Messages.* File downloads are done directly between two peers using GET/PUSH messages.

To summarize: to become a member of the network, a *servent* (node) has to open one or many connections with nodes that are already in the network. In the dynamic

environment where Gnutella operates, nodes often join and leave and network connections are unreliable. To cope with this environment, after joining the network, a node periodically PINGs its neighbors to discover other participating nodes. Using this information, a disconnected node can always reconnect to the network. Nodes decide where to connect in the network based only on local information, and thus form a dynamic, self-organizing network of independent entities. This virtual, application-level network has Gnutella servents at its nodes and open TCP connections as its links. In the following sections we describe how we discover this network topology and analyze its characteristics.

4. Data Collection: The Crawler

We have developed a *crawler* that joins the network as a servent and uses the membership protocol (the PING-PONG mechanism) to collect topology information. In this section we briefly describe the crawler and discuss other issues related to data collection.

The crawler starts with a list of nodes, initiates a TCP connection to each node in the list, sends a generic join-in message (PING), and discovers the neighbors of the node it contacted based on the replies it gets back (PONG messages). Newly discovered neighbors are added to the list. For each discovered node the crawler stores its IP address, port, the number of files and the total space shared. We started with a short, publicly available list of initial nodes, but in time we have incrementally built our own list with more than 400,000 nodes that have been active at one time or another.

We first developed a sequential version of the crawler. Using empirically determined optimal values for connection establishment timeout as well as for connection listening timeout (the time interval the crawler waits to receive PONGs after it has sent a PING), a sequential crawl of the network proved slow: about 50 hours even for a small network (4000 nodes). This slow search speed has two disadvantages: not only it is not scalable, but the dynamic network behavior means that the result is far from a network topology snapshot.

In order to reduce the crawling time, we next developed a distributed crawling strategy. Our distributed crawler has a client/server architecture: the server is responsible with managing the list of nodes to be contacted, assembling the final graph, and assigning work to clients. Clients receive a small list of initial points and discover the network topology around these points. Although we could use a large number of clients (easily in the order of hundreds), we decided to use only up to 50 clients in order to reduce the invasiveness of our search. These techniques have allowed us to reduce the crawling time to a couple of hours even for a large list of starting points and a discovered topology graph with more than 30,000 active nodes.

Note that in the following we use a conservative definition of network membership: we exclude the nodes that, although were reported as part of the network, our crawler could not connect to. This situation might occur when the local servent is configured to allow only a limited number of TCP connections or when the node leaves the network before the crawler contacts it.

5. Gnutella Network Analysis

Figure 1 presents the growth of the Gnutella network in the past 6 months. We ran our crawler during November 2000, February/March 2001, and May 2001. While in November 2000 the largest connected component of the network found had 2,063 hosts, this grew to 14,949 hosts in March, and 48,195 hosts in May 2001. Although Gnutella's failure to scale has been predicted time and again, the number of nodes in the largest network component grew about 25 times (admittedly from a low base) in the past 6 months.

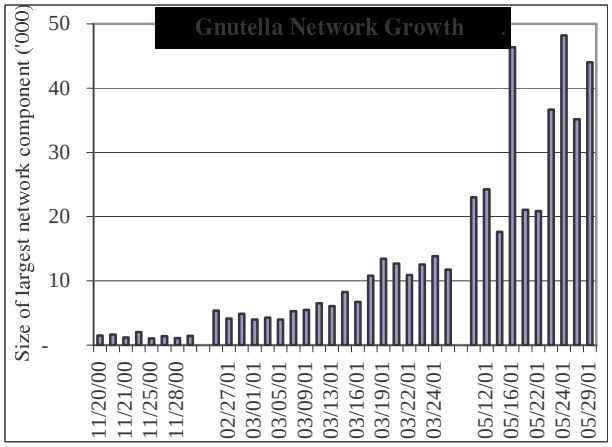

Figure 1: Gnutella network growth. The plot presents the number of nodes in the largest connected component in the network. Data collected during Nov. 2000, March 2001 and May 2001.

We identify three factors that allowed the network this exceptional growth in response to user pressure. First, as we argue in Section 5.1, careful engineering led to significant overhead traffic decreases over the last six months. Second, the network connectivity of Gnutella participating machines improved significantly. Our rough estimate (based on tracing DNS host names) is that the number of DSL- or cable-connected machines grew twice as fast as the overall network size. While in November 2000 about 24% of the nodes were DSL or cable modem enabled, this number grew to about 41% six months later. Finally, the efforts made to better use available networking resources by sending nodes with

low available bandwidth at the edges of the network eventually paid off.

It is worth mentioning that the number of connected components is relatively small: the largest connected component always includes more than 95% of the active nodes discovered, while the second biggest connected component usually has less than 10 nodes.

Using records of successive crawls, we investigated the dynamic graph structure over time. We discovered that about 40% of the nodes leave the network in less than 4 hours, while only 25% of the nodes are alive for more than 24 hours. Given this dynamic behavior, it is important to find the appropriate tradeoff between discovery time and invasiveness of our crawler. Increasing the number of parallel crawling tasks reduces discovery time but increases the burden on the application. Obviously, the Gnutella map our crawler produces is not an exact 'snapshot' of the network. However, we argue that the network graph we obtain is close to a snapshot in a statistical sense: all properties of the network: size, diameter, average connectivity, and connectivity distribution are preserved.

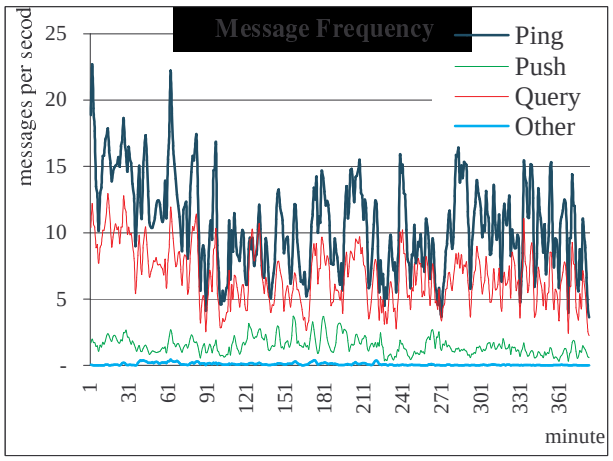

Figure 2: Gnutella generated traffic classified by message type over a 376 minute period in November 2000. Note that overhead traffic (PING messages serve only to maintain network connectivity) forms more than 50% of the traffic. For backward compatibility, flooding is also used to deliver some file download requests (PUSH messages). The only 'true' user traffic is QUERY messages.

5.1. Gnutella Generated Traffic

We also used the crawler to eavesdrop the traffic generated by the network. In Figure 2 we classified Gnutella generated traffic in November 2000, according to message type. We detected that the volume of generated traffic is the main obstacle for better scaling and wider deployment. After adjusting for message size, we determined that, on average, only 36% of the total traffic was user-generated traffic (QUERY messages). 55% of the traffic was pure overhead (PING and PONG messages) while 9% of the traffic contained either bogus messages (1%) or PUSH messages that were broadcasted by servents that were not fully compliant with the latest version of the protocol.

By June 2001 (presumably due to the arrival of newer Gnutella implementations), unnecessary overhead traffic was cut down: generated traffic contained 92% QUERY messages, 8% PING messages and insignificant levels of other message types. A large part of the network's ability to grow can be attributed to this significant reduction of overhead traffic.

5.2. Shared Data Distribution

Figure 3 presents the correlation between the amount of data shared and the number of links a node maintains. While most nodes share few files and maintain only a couple of connections, a small group provides more than half of the information shared, while another, distinct, group provides most of the essential connectivity (this characteristic of the network is also mentioned in [2]).

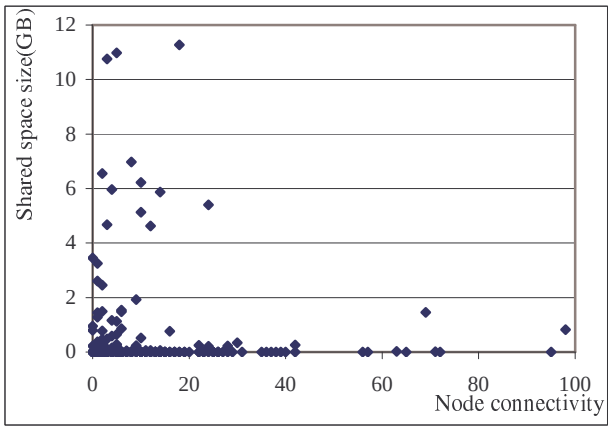

Figure 3: Correlation between the amount of data shared by a node and the number of links it maintains. Gnutella network on 12/03/2000 with 1876 nodes and 0.93TB of shared disk space. The picture is similar for other network snapshots.

In Figure 4 we eliminate the 'free-riders' (nodes that do not share any files) and plot the distribution of the number of files shared by each node. We discover a power-law distribution for the number of nodes N that share files: the number of nodes sharing k files is $N=k^{-C}$, where c is a constant. This distribution holds for the majority of our Gnutella network measurements, regardless of the time when they were taken, with a constant c in the range 0.8 to 0.95. This result is only partially surprising as the power law is manifest in a large

number of instances in the WWW and Internet world [1,9,12]: the number of pages served by an HTTP server, the number of hits received by an HTTP server, the number of in/out-links from a Web page, and the number of network connections to an Internet host are all distributed according to a power-law. Moreover, there are strong similarities between power-law and Zipf's distributions ([15] shows how one of these distribution laws can be expressed in the other's terms). A large number of man-made and naturally occurring phenomena including incomes, word frequencies, earthquake magnitudes, web-cache object popularity and even Gnutella queries [17] are distributed according to a Zipf's law distribution.

The power-law distribution observed makes the network extremely dependent on the information provided by the largest. Thus, the largest 1% of all nodes provide 30% of the files available in the network, while the largest 10% of nodes provide 71% of the files available. Free riding (i.e., participating in the network without sharing files) can only exacerbate this dependency.

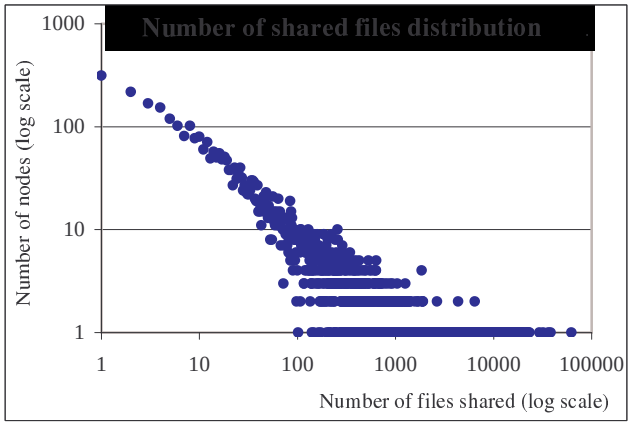

Figure 4: Power-law data distribution: The number of nodes that share a particular number of files is roughly proportional to the number of files raised to a negative constant power.

5.3. Node Connectivity and Network Topology Analysis

One interesting feature of the network is that, over a seven-month period, with the network scaling up almost two orders of magnitude, the average number of connections per node remained constant (Figure 5). Assuming this invariant holds, it is possible to estimate the number of connections a larger network will create and find scalability limits based on available bandwidth (a detailed analysis is presented in [16]).

When analyzing global connectivity and reliability patterns in the Gnutella network, it is important to keep in mind the self-organized network behavior: users decide only the maximum number of connections a node should support, while nodes decide to whom to connect or when to drop/add a connection based only on local information.

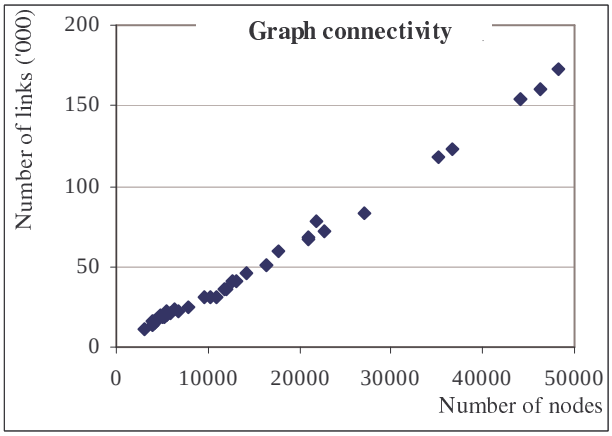

Figure 5: Average node connectivity. Each point represents one Gnutella network. Note that, as the network grows, the average number of connections per node remains constant (average node connectivity is 3.4).

Recent research [1,7,8,13] shows that many natural networks such as molecules in a cell, species in an ecosystem, and people in a social group organize themselves as so called *power-law networks*. In these networks most nodes have few links and a tiny number of hubs have a large number of links. More specifically, in a power-law network node connectivity follows a power law distribution (i.e. the fraction of nodes with L links is proportional to L^{-k}, where k is a network dependent constant as introduced in Section 5.2.).

This structure helps explaining why networks ranging from metabolisms to ecosystems to the Internet are generally highly stable and resilient, yet prone to occasional catastrophic collapse [14]. Since most nodes (molecules, Internet routers, Gnutella servents) are sparsely connected, little depends on them: a large fraction can be taken away and the network stays connected. But, if just a few highly connected nodes are eliminated, the whole system could crash. One implication is that these networks are extremely robust when facing random node failures, but vulnerable to well-planned attacks.

Given the diversity of networks that exhibit power-law structure and their properties, we were interested to determine whether Gnutella falls into the same category. Figure 6 presents the connectivity distribution in November 2000. Although data are noisy (due to the small size of the networks), we can easily recognize the signature of a power-law distribution: the connectivity distribution appears as a line on a log-log plot. [6,4]

confirm that early Gnutella networks were power-law. Later measurements (Figure 7), however, show that more recent networks move away from this organization: there are too few nodes with low connectivity to form a pure power-law network. In these networks the power-law distribution is preserved for nodes with more than 10 links while nodes with fewer links follow an almost constant distribution.

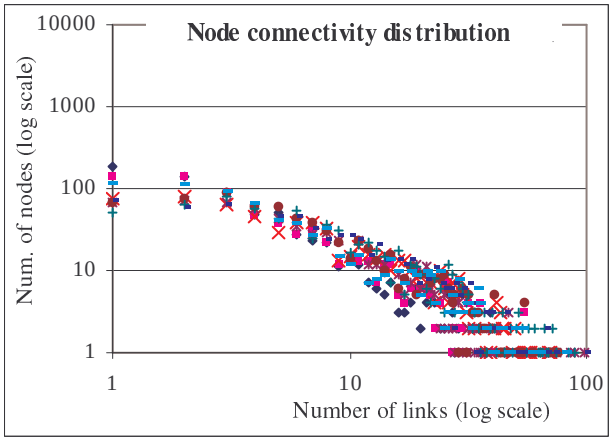

Figure 6: Connectivity distribution during November 2000. Each series of points represents one Gnutella network topology we discovered at different times during that month. Note the log scale on both axes. Gnutella nodes organized themselves into a power-law network.

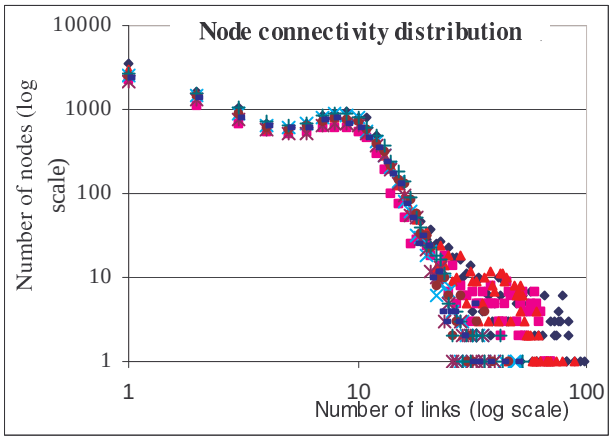

Figure 7: **Connectivity distribution** during March 2001. Each series of points represents one Gnutella network topology discovered during March 2001. Note the log scale on both axes. Networks crawled during May/June 2001 show a similar pattern.

An interesting issue is the impact of this new, multi-modal distribution on network reliability. We believe that the more uniform connectivity distribution preserves the network capability to deal with random node failures while reducing the network dependence on highly connected, easy to single out (and attack) nodes.

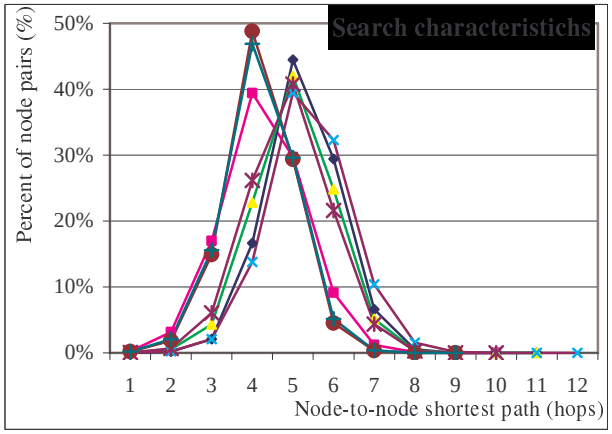

Figure 8: Distribution of node-to-node shortest paths. Each curve represents one network measurement. Note that, although the largest network diameter (the longest node-to-node path) is 12, more than 95% of node pairs are at most 7 hops away.

Another interesting issue is the combined impact of network growth and node connectivity distribution on node-to-node distance in the network. Remarkably, while the network scaled up in size about 50 times, the average node-to-node distance grew only 26% from an average of 4.24 in November to an average of 5.35 in May 2001. Figure 8 shows the distribution on node-to-node distances and confirms the average one hop increase over the past 6 months: the three left-most curves represent network measurements from November 2000 while the other represent network measurements from May 2001. It is easy to see that the curves that represent more recent measurements have uniformly shifted right about one hop.

6. Summary and Potential Improvements

Gnutella is an open, decentralized, P2P search protocol that is mainly used to find and share files. Computers running Gnutella protocol-compatible software form an application-level network. We have developed tools to discover and analyze this network. Our analysis shows that Gnutella node connectivity follows a multi-modal distribution, combining a power law and a quasi-constant distribution. This property keeps the network as reliable as a pure power-law network when assuming random node failures, and makes it harder to attack by a malicious adversary. Gnutella takes few precautions to ward off potential attacks. For example, the network topology information that we have obtained here is easy to obtain and would permit highly efficient denial-of-service attacks. Some form of security mechanisms that would prevent an intruder from gathering topology information appears essential for the long-term survival of the network

(although it would make global network monitoring more difficult if not impossible).

We see two directions for improvement. First, we observe that the application-level topology determines the volume of generated traffic, the search success rate, and application reliability. We could design an agent that constantly monitors the network and intervenes by asking servents to drop or add links as necessary to keep the network topology optimal. Agents (or nodes) could embed some information about the underlying physical network and build accordingly the virtual application topology. Note that implementing this idea requires only minor protocol modifications.

The second direction is to replace flooding with a smarter (less expensive in terms of communication costs) routing and group communication mechanism. We have collected a large amount of data on the environment in which Gnutella operates and shall use it in simulations to investigate the appropriateness of these and various other alternatives.

Acknowledgements

I am grateful to Ian Foster, Adriana Iamnitchi, Larry Lidz, Conor McGrath, Dustin Mitchell, and Alain Roy for their insightful comments and generous support. This work started as a joint class project with Yugo Nakai and Xuehai Zhang. This research was supported by the National Science Foundation under contract ITR-0086044.

References

[1] M. Faloutsos, P. Faloutsos, and C. Faloutsos, *On Power-Law Relationships of the Internet Topology*, SIGCOMM 1999.

[2] E. Adar and B Huberman, *Free riding on Gnutella*, First Monday Vol. 5, Number 10, - Oct. 2nd 2000.

[3] Gnutella protocol specification v4.0. Available at: http://dss.clip2.com/GnutellaProtocol04.pdf

[4] DSS group, *Gnutella: To the Bandwidth Barrier and Beyond*, available at: http://dss.clip2.com, Nov. 6, 2000.

[5] DSS group, Bandwidth Barriers to Gnutella Network Scalability, available at: http://dss.clip2.com, Sept. 8, 2000.

[6] L. Adamic, R. M. Lukose, A R. Puniyani, and B. A. Huberman, *Search in Power-Law Networks*

[7] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins and J. Wiener, *Graph structure in the web*, 8th International WWW Conference, May 15-19 Amsterdam.

[8] A. Barabasi and R. Albert. *Emergence of scaling in random networks,* Science, 286(509), 1999.

[9] B. Huberman and L. Adamic, *Growth dynamics of the World-Wide Web,* in Nature, 399 (1999) p130.

[10] M. Katz and C. Shapiro, Systems Competition and Network Effects, *Journal of Economic Perspectives*, vol. 8, no. 2, pp. 93-115, 1994.

[11] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley, 1991.

[12] N. Shiode and M. Batty, *Power Law Distributions in Real and Virtual Worlds,* INET'2000, Yokohama, July 2000.

[13] A. Barabási, R. Albert, H. Jeong, and G. Bianconi *Power-law distribution of the World Wide Web*, Science 287, (2000).

[14] R. Albert, H. Jeong, and A. Barabási, *Attak and tolerance in complex networks,* Nature 406 378 (2000).

[15] L. Adamic, *Zipf, Power-laws, and Pareto - a ranking tutorial,* online tutorial: http://www.hpl.hp.com/shl/

[16] M. Ripeanu, *Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design*, University of Chicago Technical Report TR-2001-26, July 2001.

[17] K. Sripanidkulchai, The popularity of Gnutella queries and its implications on scalability, February 2001, available at: www.cs.cmu.edu/~kunwadee/research/p2p/gnutella.html

[18] T. Hong, *Performance,* In *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, ed. by A. Oram. O'Reilly and Associates: Sebastopol, CA, 2001.

[19] http://gnutella.wego.com/

[20] http://www.groove.net/

[21] http://www.entropia.com

[22] http://setiathome.ssl.berkeley.edu/

[23] http://freenet.sourceforge.net/