Crawling Gnutella: Lessons Learned*

David G. Deschenes, Scott D. Weber, and Brian D. Davison
Department of Computer Science and Engineering, Lehigh University
19 Memorial Drive West, Bethlehem, PA 18015

January 2004

Abstract

Since its inception in early 2000, the Gnutella network has been the subject of numerous topology measurement studies. Although those studies vastly improved our understanding of the network, and of unstructured peer-to-peer networks in general, they did not address the problem of obtaining accurate measurements efficiently. In this paper we discuss how to adapt existing measurement approaches to features provided by version 0.6 of the Gnutella protocol, present the results of several measurement experiments, and analyze collected data with an eye towards identifying efficient and accurate measurement techniques.

1 Introduction

The most popular approach to measuring the topology of the Gnutella network [5] is through a process that is remarkably similar to that used by a Web crawler[7]. Proceeding from hosts about which it has offline knowledge, a Gnutella crawler treats the network as a graph and typically performs a breadth-first exploration by identifying and following connections between hosts. In general, a crawler has completed its exploration when it has visited all hosts to which connections were identified.

While this approach seems intuitive, it is difficult to use in practice. Of primary concern is the fact that the topology of the Gnutella network is

*Technical Report LU-CSE-04-005, Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA, 2003. Deschenes and Davison were supported in part by the National Science Foundation under grant ANI-9903052. Information about this project may be retrieved from http://wume.cse.lehigh.edu/ or by emailing the last author at davison@cse.lehigh.edu.

constantly changing as new hosts join and leave, requiring that a crawler complete its exploration quickly in order to ensure a reasonable approximation of the topology at a moment in time. Furthermore, the crawler has no notion of the size of the network that it is measuring, and therefore cannot judge whether or not the entire network has been measured. And finally, throughout this process a crawler must maintain knowledge of hosts that are good entry-points into the network, weed through hosts that are incorrectly identified as participating in the network, and struggle with participating hosts that are uncooperative.

It is because of the aforementioned problems that previous topology measurements have not asserted any amount of accuracy. In this paper we discuss how to adapt existing measurement approaches to features provided by version 0.6 of the Gnutella protocol, and explain how characteristics of the network may be used to improve measurements.

2 Previous Work

Over a period of five months during the summer of 2000, Clip2 DSS performed the first well-known study of the Gnutella network [3]. The study encompassed a wide range of topics, including but not limited to: network connectivity, message classification, file and file type distribution, as well as geographical classification of hosts. It estimated that the network consisted of no less than 10,000 hosts and at times approached 30,000 hosts. Unfortunately, no details were provided regarding the methods used to collect topology data.

In his master's thesis [4], Jovanovic describes the design of both serial and parallel Gnutella crawling algorithms. Jovanovic identified problems that might lead to the collection of questionable topology data, including: high average RTT of messages and an inability to discover network connections within private networks. Finally, Jovanovic analyzed the topology of the Gnutella network using data collected during the winter of 2000. He found that the network exhibited "small world" properties of clustering and small diameter and that node degrees followed a power-law distribution.

In [10], Saroiu, Gummadi and Gribble attempted to characterize the end-user hosts participating in both the Napster and Gnutella networks. Although the authors ran many short crawls of the Gnutella network over a period of eight days during May of 2001, and gathered information about 1,239,487 hosts, they were only able to discover subsets of the topology.

Finally, in [8], Ripeanu, Foster and Iamnitchi analyzed Gnutella network

topology data and message traffic gathered over a six month period starting in late 2000. The authors observed significant growth in the network's size, measuring a maximum of approximately 48,000 hosts in May 2001. Using successive crawls, they also found that 40% of the hosts depart the network in less than four hours, while only 25% are active for more than a day.

While we only directly address the task of crawling the Gnutella network in this paper, we were able to draw ideas from a number of studies [1, 6, 9, 11] that focused on analyzing the network's traffic. In general, those studies attempted to characterize the network's users and their content or to understand the scalability of the network.

3 Gnutella Protocol Overview

Two versions of the Gnutella protocol are in common use today, and specifications for both are publicly available.[1] In early 2000, the original version of the protocol, 0.4, was reverse engineered from closed source software being developed at Nullsoft.[2] That version of the protocol defines how hosts establish connections, the messages that they may use to communicate, and how those messages are propagated through the network. Most importantly, it outlines how hosts are able to discover each other, the process which enables one to crawl the network. According to the 0.4 specification, if a host propagates a ping message identifying the host to the network, that host should receive a pong message from each and every other host within the horizon of the ping message. Furthermore, because every message is encoded with its distance traveled through the network, we can determine who is directly connected to the peer, and it is from that information that a Gnutella crawler is able to build a model of the network's topology.

While version 0.4 (v0.4) of the Gnutella protocol laid the foundation for the network, it did not provide many of the features sought by Gnutella software vendors. So, late in 2001 the community of developers supporting the Gnutella protocol introduced version 0.6 (v0.6). The most significant change brought by v0.6 was the introduction of handshaking during connection establishment. The handshaking process was designed to allow for the exchange of headers, in a method strikingly similar to that employed in HTTP, that identify the extended features of the hosts establishing the connection. Version 0.6 was quickly integrated by Gnutella software vendors and prompted the development of numerous extensions to the protocol.

[1] http://rfc-gnutella.sourceforge.net/

[2] http://www.nullsoft.com/

One of the headers in common use is the "X-Try" header. Hosts send this header followed by host addresses identifying other hosts on the network. This way, even if a host must reject an incoming connection, it can still provide enough information to the connecting host so that it may continue its attempts to join the network.

For additional details of the Gnutella protocols, the reader is referred to the Gnutella specifications.[1]

4 Crawler Architecture

Our Gnutella crawler is a multi-threaded application written in Java and makes use of non-blocking, asynchronous I/O to ensure good performance. After the crawler has been initialized and a start set has been provided, the crawler proceeds in exploring the network. Because the crawler is capable of maintaining connections to hundreds of hosts simultaneously, it is able to explore the network from many different points.

In order track its progress, the crawler maintains a list of hosts that it has visited, a list of hosts that it is visiting, and a queue of those that it has not yet visited. Hosts that are discovered during the crawl are added to the queue.

When a crawl is started, the crawler may optionally listen for and accept incoming Gnutella connections. When the listener is used, the crawlers advertise the listener's IP address and port pair to the hosts with which they connect through X-Try headers and pong messages as appropriate. Each connection to the listener is treated as if it were the only connection being handled. The listener does not forward and does not broadcast any messages, nor does it respond to queries. It only sends and responds to ping messages.

In addition to providing snapshots of the Gnutella network, our crawler keeps a history of its experiences with each and every host ever identified to it. Following the completion of each network measurement, the crawler updates a host database which summarizes each host experience. In addition, separate logs are kept for the listener and the crawling threads with all crawling threads logging to the same pair of logs.

Connection logs contain information about each connection attempt that is made. For each connection, the remote address and port is logged along with the remote host's user agent name when available, the protocol version used, the state in which the connection terminated and the time spent in each of the three major connection phases (TCP connection, Gnutella con-

nection, Gnutella session). The TCP connection time is the time to establish a TCP connection with the remote host and is always zero for incoming connections. The Gnutella connection time is measured as the time between the establishment of a TCP connection and the establishment of a Gnutella session. The Gnutella session time is measured from the establishment of a Gnutella session until the connection is dropped (typically by the remote host).

Pong logs contain information about each pong message that is received. For each pong, the remote address is logged along with the address contained within the pong data. The number of hops taken, number of files shared and number of kilobytes shared are also logged. Finally, a measure of the time between the establishment of a session and the arrival of the pong are stored to allow effective analysis of pong response time distributions.

5 Comparing Protocol Versions

As was mentioned earlier in section 3, two versions of the Gnutella protocol are in common use today. The key contribution of, and main motivation for, v0.6 was support for the development of protocol extensions. Numerous developers took advantage of that support and created extensions designed to improve the performance of user searches, reduce network load, and stabilize the topology of the network. Regrettably, no one has yet measured the extent to which this new protocol has been deployed or its impact on the topology of the network. In order to shed light on the situation we performed simultaneous crawls of the network using both versions 0.4 and 0.6 of the Gnutella protocol. The pairs of crawls were started every three hours from midnight April 10, 2003 until 9 PM April 16, and all used an identical start set containing slightly more than 50,000 host addresses.

5.1 Peer Cooperation

In order to obtain accurate topology data, it is essential that active hosts cooperate with a crawler. A carefully constructed start set can provide a number of useful entry points into the network, but a crawler depends on the identification of other active hosts to ensure reasonable coverage.In v0.4 the only host identification mechanism is the pong message, while v0.6 makes use of both pong messages and X-Try headers.

Figure 1 shows that when using v0.6, more than seven times as many hosts are identified than when using v0.4. Moreover, we can see that the difference in number of identifications is due to the use of X-try headers,

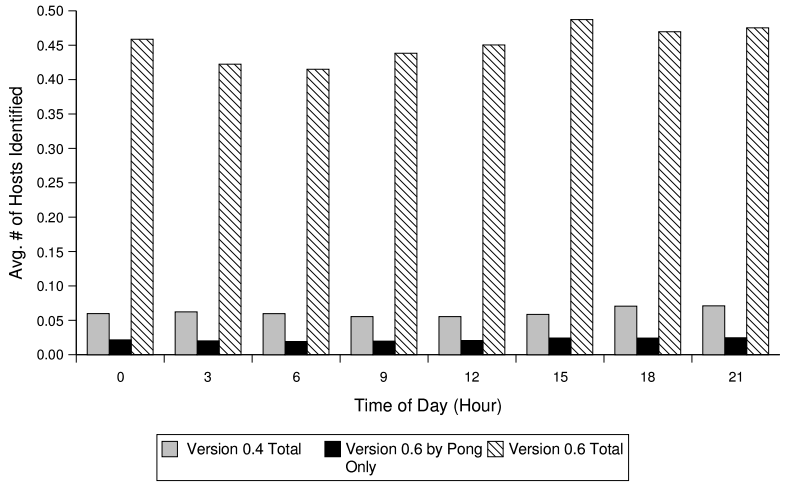


Figure 1: Comparison of host identifications per connection attempt.

as fewer pongs per session are recorded when using v0.6. In general, these results may be explained by the fact that X-Try identifications may be collected even from those hosts refusing a connection attempt from a crawler, and they suggest that hosts using v0.6 offers an increased level of cooperation over those using v0.4.

5.2 Measured Network Sizes

Given the levels of peer cooperation identified above it should come as no surprise that crawls using protocol v0.6 contained, on average, nearly five times as many hosts as those using v0.4. Figure 2 compares the average measured network sizes for a given time of day. Also, while pairs of simultaneous crawls identified a common set of peers, that common set always made up more than 40% of the network measured by v0.4, but it never made up more than six percent of the network measured by v0.6. Considering the restricted nature of the crawls performed we cannot conclusively say that protocol v0.4 should be avoided when developing a Gnutella network crawler. However, the data presented suggests that there has been a significant shift among Gnutella vendors to v0.6, and that the new version is substantially better with respect to the measurement of a complete network topology. Accordingly, all remaining discussion focuses on our measurements using protocol v0.6.

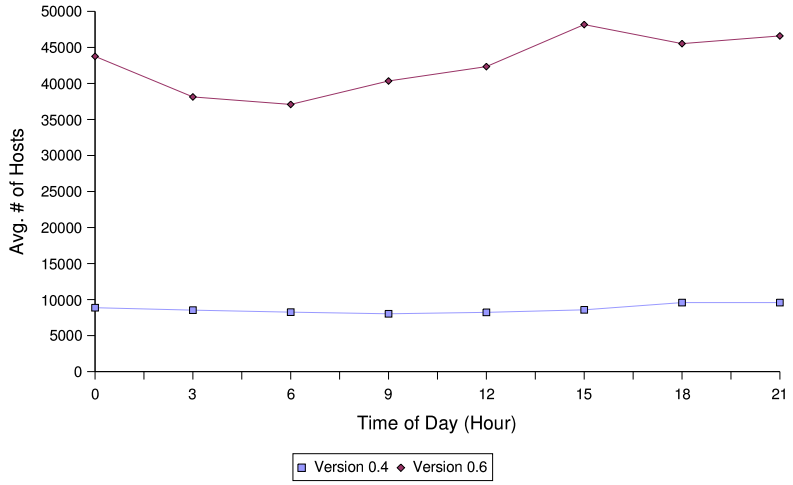
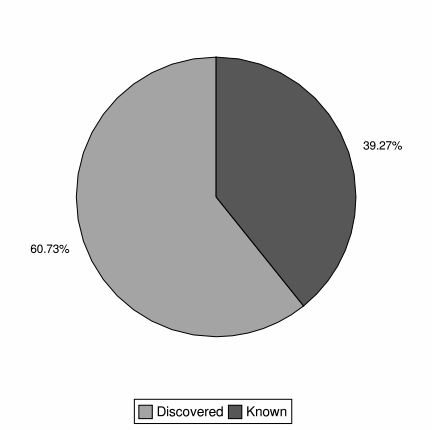


Figure 2: Comparison of measured network sizes.

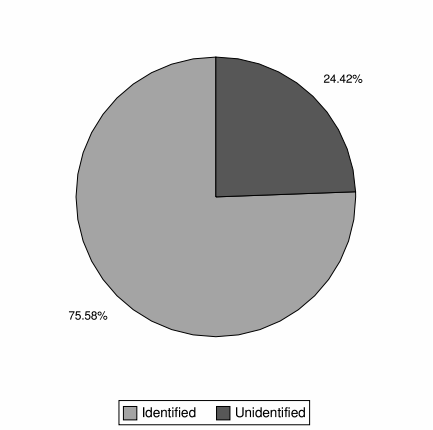
6 Characterizing Valuable Peers

Although crawls using v0.6 generate considerable amounts of data, they are horribly inefficient. An average crawl in the experiment described in Section 5 required visits to more than 87,000 hosts, acquiring TCP connections to only 20% of them and Gnutella sessions with only 6.5%. As a result, a disproportionate amount of time is spent attempting to acquire data from hosts that are either no longer connected to the Internet, no longer active in the Gnutella network, or too busy to handle another incoming connection. Both the start set and set of hosts discovered during a crawl are sources of this wasted effort. On average, only about 10% of discovered hosts provide sessions, while a meager 5% of hosts in the start set provide sessions.

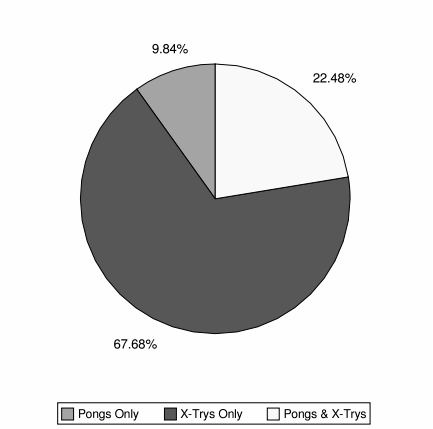
Although discovered hosts provide a majority (60%) of all sessions as shown in Figure 3 (a)), it is not so overwhelming that we can ignore the contributions of hosts in the start set. Instead, we focus our analysis on all hosts *identified* (those hosts which are named in a pong message or X-Try header), which may include hosts in our start set. In Figure 3 (b) we can see that more than 75% of all sessions are provided by hosts that have been identified, making it an excellent discriminating factor. Furthermore, in Figure 3 (c) we can see that hosts identified only by X-Try headers account for nearly 68% of all identified hosts providing sessions. So, it is essential that, in addition to collecting and processing pong messages, a crawler make use of X-Try headers.



(a) Breakdown of Sessions (Discovered vs. Known)



(b) Breakdown of Sessions (Identified vs. Unidentified)



(c) Breakdown of Identified Hosts Providing Sessions

Figure 3: Characterizing valuable peers.

Though hosts identified only by X-Try headers make up most of the identified hosts that provide sessions, this does not mean that X-Try headers are more useful than pongs with respect to picking out valuable hosts. If we consider the total number of hosts only identified by X-Try headers we

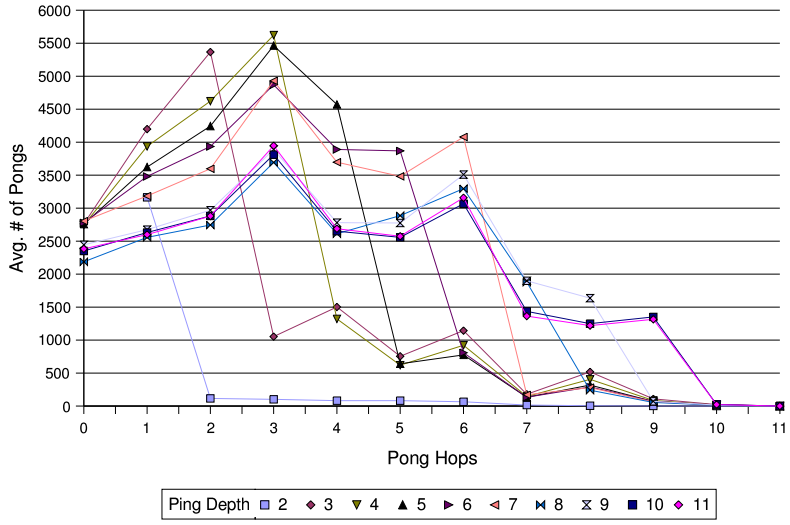


Figure 4: Comparison of pong responses for various ping depths.

see that just 8.5% of those hosts provide sessions. That is in contrast to the 22% of hosts only identified by pongs and 27% of hosts identified by both pongs and X-Try headers that provide sessions. Therefore, when ordering hosts for the crawl, hosts that have been identified should be placed before those that have not; and more importantly, those that have been identified by both pongs and X-Try headers should be considered before those only identified by pongs, which in turn should be considered before those only identified by X-Try headers.

7 Exploring Ping Depths

By varying the TTL of the ping messages sent out by our crawler, we are able to obtain data about how the network handles messages. The varying TTL of the pings allows us to see how the properties and scope of the network change with size. The ping depth experiment ran ten times every day, from April 9 to 23, 2003, each crawler instance randomly choosing from one of ten TTL values between 2 and 11 inclusive. The TTL values were selected such that each would be used exactly once in each set of ten runs.

7.1 Pong Caching

In an attempt to reduce the amount of pong traffic in the Gnutella network, many v0.6 clients support a "pong caching" scheme. As pong messages are received, they are stored in a local cache. Upon receiving a ping message, the caching client responds with pongs from its cache and may choose to drop the ping message from the network, regardless of its TTL. These schemes can considerably reduce the amount of ping and pong traffic on the network while still providing ping sources with information about potentially active hosts.

Disregarding the ping depth used, each run of the crawler received at least one pong message from every depth, 0 through 10. As can be seen in Figure 4, there is a large drop in the number of pong messages received beyond the expected horizon for any given ping depth. However, the number of pongs does not drop to zero. A ping message entering the network with a TTL of 3 should not trigger the responses from hosts 5 hops away. However, if pong caching is in use, a host 2 hops away may respond with pong messages that it had earlier received from hosts further away. The fact that the pong message count is not zero for hosts outside the horizon of the ping message to which they are responding suggests that pong caching is taking place. Moreover, the steep drop-off in the pong message count suggests that only a fraction of hosts implement pong caching or that most hosts implement "smart" pong caching schemes, which only respond with pong messages identifying hosts previously measured within the horizon of the ping message.

7.2 Crawling Ping

A ping message with a TTL of 2 is often referred to as a crawling ping. This ping is essentially a request to learn about hosts connected to a neighbor. A recipient of a crawling ping may choose to respond to the message directly rather than forwarding it to all its neighbors. When responding in this way, the protocol asks that the pong messages are delivered as if they originated at the responder by setting the hop count to zero instead of one. It appears that some clients take the notion of a crawling ping one step further. These clients assume that a ping with TTL of 2 is an attempt by the sender to learn about other hosts with which it might connect. To aid the ping source, the recipient replies with a large number of pong messages with a hop counts of zero and the addresses of hosts likely on the network, but not necessarily neighbors.

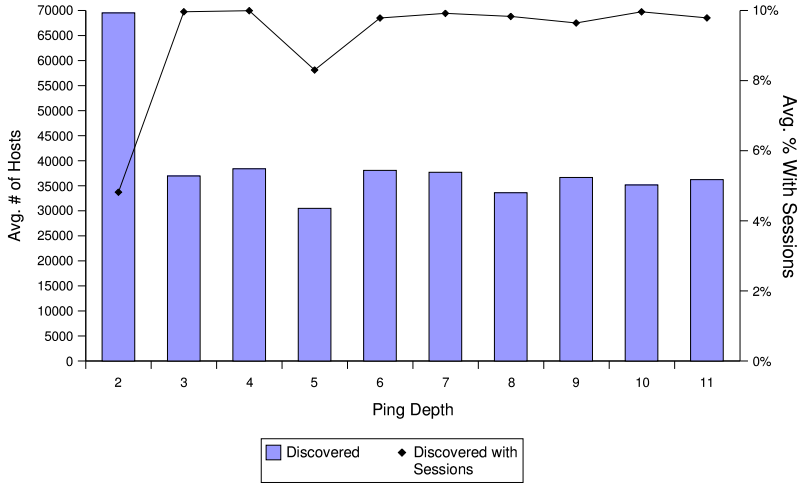


Figure 5: Hosts discovered and the percentage of those hosts with which the crawler was able to obtain a session.

Crawls using a ping depth of 2 received many more pongs than crawls at any other depth: an average of about 68,000 for depth 2 versus a maximum average of about 26,000 for any other depth. In crawls with ping depth of 2, 95% of the pong messages received had hop counts of zero. In crawls of all other ping depths, the portion of pong messages received with hop count zero never exceeded 16%. To avoid distortion of the figure, pongs with hop count zero resulting from a ping depth 2 crawl are omitted from Figure 4.

7.3 Host Discovery

A *host discovery* occurs when the crawler learns about a host address that it has not yet seen. As might be expected from the discussion in the previous section, crawls using a ping depth of two yielded roughly twice that discovered in a run of any other depth. Figure 5 shows the average number of hosts discovered by a crawl for each of the ping depths tested.

7.4 The Value of a Discovery

Also shown in Figure 5 is the ratio of the discovered hosts with which our crawler was actually able to establish a Gnutella session (that is, when both sides have exchanged protocol headers and responded with "200 OK"). The crawls with ping depth two provide twice as many host discoveries, but half

as many hosts with which sessions can be established. This suggests that while ping depth two retrieves a larger quantity of data, the quality of that data may not be as good. The dip in the ratio at ping depth 5 is due to two anomalous crawls in the data set.

7.5 Choosing a Ping Depth

The ping depth experiments allowed us to determine what useful data can be obtained from the network given varying input. By considering the information from the preceding sections, we can make some conclusions about what depths to choose for different intended results.

If the goal is to get a large list of hosts participating in the network as quickly as possible, a depth of 2 is probably most appropriate. This will yield many more discoveries in the same amount of time as any other depth. However, since many of the discoveries will come from pong messages with (inaccurate) hop counts of zero, it is impossible to determine the network topology from these messages.

To obtain the best topology estimate, we should examine pongs with hop counts greater than zero. The topology can be most easily extracted from pongs with a hop count of one. These pongs indicate a direct connection between a neighbor of the crawler and a neighbor of that neighbor. Pongs with larger hop counts may be combined with other information to determine links, but only provide inexact connection information. So to get the most accurate and largest topology estimate possible, we should look to maximize the number of pongs we receive with hop count of one. Looking at Figure 4, we see that this occurs with a ping depth of 3.

To get the most non-zero pongs possible, we should choose ping depth 7. On average, this depth yields more data than any other (excluding ping depth 2).

8 Time Effects

The Gnutella network is highly available. At any given time, tens of thousands of hosts are actively participating. By collecting data throughout the day, we can see how diurnal patterns affect the network.

8.1 Time of Day

We observe two, mostly inversely related patterns with respect to time of day (summarized in Figure 6). While most of the indicators of network activity,

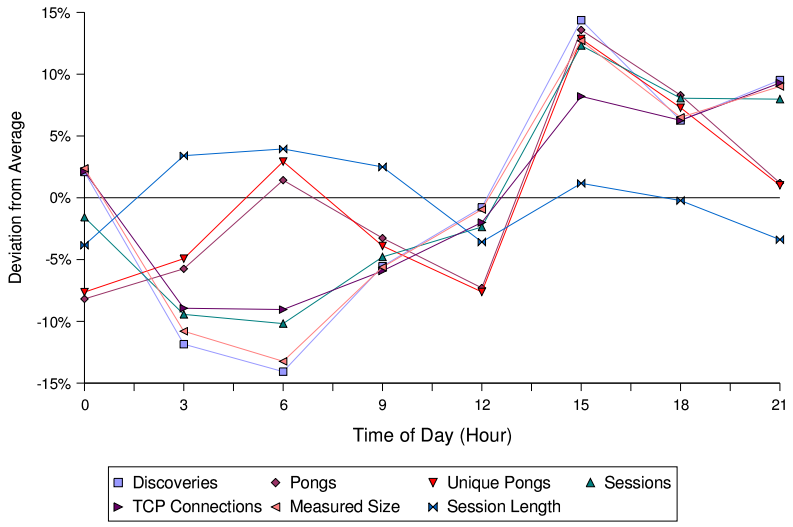


Figure 6: Daily fluctuation in network activity.

including measured size and number of discoveries, follow one pattern, the number of pongs received follows another. Surprisingly, the diurnal pattern for pongs is often inversely related to that for number of connections and sessions. At first glance this seems counter-intuitive. The average length of a session also follows a similar pattern as that of pongs and may provide an explanation.

Pongs are received only during sessions with other hosts, and in general longer sessions yield more pongs. This assumption is supported by the evidence in Figure 6. We also note that sessions tend to last longer when the network is smaller and shorter as the network grows. One possibility is that when the network shrinks, hosts spend less time changing their connections since there are fewer other hosts with which to connect. So, in order to keep itself connected to the Gnutella network, a host will maintain its connections for lengths of time roughly inversely proportional to the size and activity of the network.

The Gnutella network is busier in the late afternoon and early evening. This time period can be correlated with the end of the typical work or school day. The Gnutella network does not have any explicit function to find hosts nearby geographically, but a client may choose to favor lower-latency connections, which would impose a rough geographic constraint on the network. This may explain why our results show the network changing

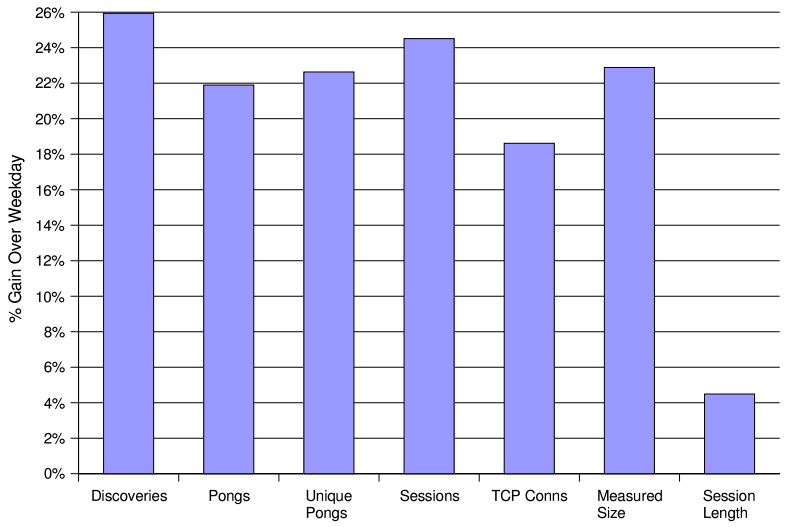


Figure 7: Gains in weekend traffic over weekday traffic.

roughly in-step with a typical work or school day in the Eastern time zone, where the measurements were taken.

8.2 Day of Week

Just as the Gnutella network changes size with respect to the hour of the day, it also changes with respect to the day of the week. The network grows larger on the weekend just as it does during the evening hours when people have left work. Figure 7 shows the increase in each of the network indicators on a weekend day over the same value on a weekday.[3] These results suggest that the weekend cannot be treated the same as weekdays when examining network structure.

9 Summary

We have examined the current state of the public Gnutella network. Several conclusions can be drawn that should guide future research involving the network:

[3]For the purposes of this calculation, we consider Monday through Thursday as weekdays and Saturday and Sunday as weekend days. During our data collection, Friday fell on a religious holiday, and so it is omitted as it is difficult to tell in which category it belongs.

- Version 0.4 of the Gnutella protocol is still in active use on the network, but its popularity is waning. Crawling the network with v0.6 of the protocol yields significantly more data than with v0.4 and finds more than 40% of the hosts found using v0.4. Since v0.4 only finds approximately 6% of the hosts found by v0.6, crawling the network using only v0.4 is clearly not effective for obtaining a full picture of the network. Since a significant chunk of the data found by v0.4 is unique to v0.4, it may be necessary to crawl the network with both protocols to obtain the best picture possible.

- Just as in WWW crawling [2], it is possible to optimize the order in which nodes are visited to obtain higher quality data in a shorter amount of time.

- The data obtained from the use of different ping depths varies substantially. A depth of 2 is useful for finding many hosts quickly, while a depth of 7 yields the most non-zero pongs.

- Time of day and day of week changes indicate that the network is not stable on short or long timescales. Therefore, any network crawl must be repeated many times in a day and for multiple days to accommodate the effects of the fluctuating activity.

References

[1] K. Anderson. Analysis of the traffic on the Gnutella network. Available from http://www.cs.ucsd.edu/classes/wi01/cse222/projects/reports/p2p-2.pdf, 2001.

[2] J. Cho, H. Garcia-Molina, and L. Page. Efficient crawling through URL ordering. In *Proc. 7th World Wide Web Conf.*, 1998.

[3] C. D. Group. Gnutella: To the bandwidth barrier and beyond. Originally found at http://dss.clip2.com/gnutella.html, Nov. 2000.

[4] M. Jovanovic. Modeling large-scale peer-to-peer newtorks and a case study of Gnutella. Master's thesis, Univ. of Cincinnati, April 2001.

[5] G. Kan. Gnutella. In A. Oram, editor, *Peer to Peer: Harnessing the Benefits of Disruptive Technologies*, chapter 8, pages 94–122. O'Reilly, Sebastopol, CA, Mar. 2001.

[6] E. P. Markatos. Tracing a large-scale peer-to-peer system: an hour in the life of Gnutella. In *Proc. of 2nd IEEE/ACM Int'l Symposium on Cluster Computing and the Grid*, 2002.

[7] B. Pinkerton. Finding what people want: Experiences with the WebCrawler. In *Proc. 2nd World Wide Web Conf.*, 1994.

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

[9] J. Ritter. Why Gnutella can't scale. No, really. Available from http://www.darkridge.com/~jpr5/doc/gnutella.html, Feb. 2001.

[10] S. Saroiu, P. K. Gummadi, and S. Gribble. Measuring and analyzing the characteristics of Napster and Gnutella hosts. *Multimedia Systems*, 9:170–184, 2003.

[11] D. Zeinalipour-Yazti and T. Folias. A quantitative analysis of the Gnutella network traffic. Available from http://www.cs.ucr.edu/~csyiazti/cs204.html, June 2002.