A Quantitative Analysis of the Gnutella Network Traffic

Demetris Zeinalipour-Yazti Theodoros Folias
Dept. of Computer Science
University of California
Riverside, CA 92507, U.S.A.
{csyiazti,folias}@cs.ucr.edu

Abstract. Peer-to-Peer (P2P) file-sharing systems such as Gnutella, Morpheus and Freenet have recently attracted a lot of interest from the internet community because they realized a distributed infrastructure for sharing files. Such systems have shifted the Web's Client-Server model paradigm into a Client-Client model. The tremendous success of such systems has proven that purely distributed search systems are feasible and that they may change the way we interact on the Internet. P2P systems uncover many new exciting features such as robustness, scalability and high fault tolerance but with a price. Most research concentrates on optimizing the communication and data model of such systems but inadequate work has been done in area of analyzing such systems. Most approaches tend to use as their basis simulation models which can lead to wrong observations and solutions.

In this project we investigate the behavior of the Gnutella system by analyzing large log traces that we have obtained with gnuDC, our Distributed Gnutella Crawler. We describe gnuDC design and implementation choices and we then describe its architecture. We make an analysis of 56 million messages that we obtained with 17 workstations in a 5 hour interval. We have also done an extensive analysis on IP addresses observed in the gnutella network. We believe that our study will facilitate the design of new more efficient communication algorithms between peers.

1 Motivation

In this project, we want to investigate the behavior of the Gnutella fully distributed P2P system by analyzing large log traces. Similar studies which were performed in the past based their results on a small set of log traces. This makes it difficult to generalize their conclusions because it is not clear whether these result present general trends or if they only present a temporary behavior of the network. Jovanovic et al. [4] studied in 2000 the Gnutella Network by obtaining five snapshots of it. Each snapshot included on average 1000 nodes and 3000 edges. These numbers are obviously not indicative any more since the Gnutella community has grown at extremely high rates ever since. Clip2 [27] shows that the typical number of peers found in the Gnutella [30] network during a weekday is $43,546$ peers sharing $1,843,549$ files. Recently another huge network of peers, namely Morpheus [24], which was previously operating over the Kazaa [29] protocol, joined the Gnutella Network. Cnet.com has logged $78,629,070$ downloads of the popular Morpheus Gnutella beta client in a period of 12 days making it the most popular download on

the internet. Moreover, more sophisticated Gnutella peers, such as Limewire [26], were developed in the meanwhile and their activity might have changed the dynamics of the network.

All this factors makes it difficult to say how the network looks like and what kind of traffic is traversing the network. We aim to develop an Online Network Traffic Analyzer which will help us to extract network statistics on regular intervals. In this way we may be able to extract some long-term properties which do not change over time.

We need to mention that most research in P2P systems concentrates on optimizing the communication and data model of P2P systems [5][6], without taking into regards the real model of such systems. Most approaches tend to use as their basis simulation models which can lead to wrong observations and solutions. There are many reasons for obtaining an accurate network model with properties that do not change over time. Some of the main reasons, which are also summarized in [2], are the following:

1. Provide an insight into the nature of the underlying system.
2. Enables analytical analysis of algorithms that are designed to perform on such topologies.
3. Allows generation of realistic topologies for simulation purposes that accurately capture important structural characteristics present in the original networks.
4. Facilitates design of new scalable algorithms that can take advantage of particular structural properties.
5. Allows prediction of future trends, thereby allowing developers to address potential problems in advance.

Our Contribution It this paper we make a quantitative analysis of the Gnutella Network Traffic at a large-scale (i.e. with 17 workstations-crawlers), which to our knowledge was not presented in any publication. We base our experimental results on a large number of log traces, i.e. 700MB, in contrast with other publications which use only a small number of log traces. We further more describe design and implementation issues of a large-scale distributed Gnutella Crawler.

2 The Gnutella Protocol at a Glance

Gnutella's distributed search protocol [30] allows a set of peers, servents or clients, to perform filename searches over other clients without the need of an intermediate Index Server. The Gnutella network topology is a pure Ad-Hoc topology where Clients may join or leave the network at any time without affecting significantly the overall operation of the network. The Protocol hence, is designed in a highly fault-tolerant fashion with a quite big overhead of synchronization messages traveling through its network. All searches are performed over the Gnutella network while all file downloads are done offline. In this way every servent that needs to serve a file launches a mini HTTP 1.1 web server and communicates with the interested servents with `HTTP` commands.

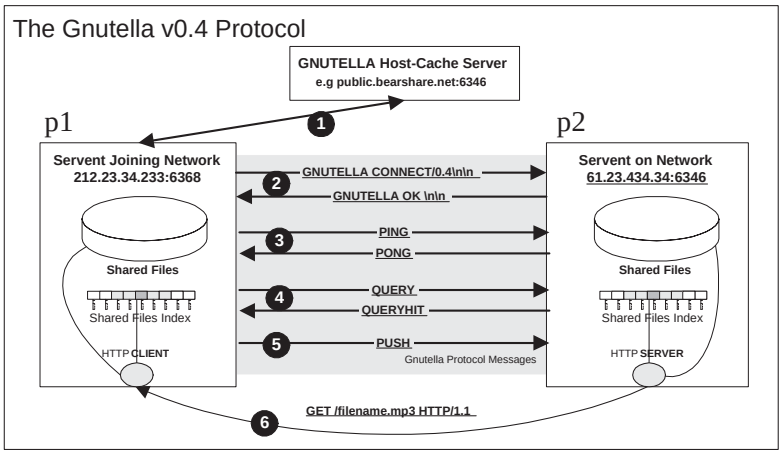

Fig. 1. Gnutella v0.4 Protocol.

The core of the protocol is comprised of a set of descriptors which are used for communication between servents and also sets rules for the inter-servents descriptors exchange. These descriptors are *Ping, Pong, Query, QueryHit and Push*. *Ping* messages are sent by a servent that needs to discover hosts that are currently active on the network. *Pong* messages on the other hand are responses to *Ping* requests every time a host wants to come in communication with the peer that requests to join the network (Figure 1, step 3). Although the protocol doesn't set any bound on the amount of incoming or outcoming connections from a particular host, most Gnutella clients come with a default value for these parameters. A client joining the Gnutella network for the first time may of course not have any clue regarding the current topology or his neighboring peers making the connection to the network impossible. For this reason most client vendors have setup *Host-Caches* Servers which serve clients with IP addresses of servents currently connected to the network (step 1). Another mechanism which is widely deployed, but is not part of the protocol, is the use of local caches of IP addresses from previous connections. In this way a servent doesn't need to connect to an IP acquisition server (i.e. a host-cache) but can rather try to establish a connection with one of its past peers.

Right after a peer has obtained a valid *IP address* and *socket port* of another servent it may perform several queries by sending *Query* descriptors and receive asynchronously results in *QueryHit* descriptors (step 4). Step 6 shows how a servent p_1 may download a file from another servent p_2. This procedure is done offline with the HTTP protocol. If the servent p_2 is firewalled, then p_1 may request from p_2, with a *Push* descriptor (step 5), to "push" the file.

3 Related Work.

There are several efforts of analyzing the traffic of large-scale P2P systems. Although most of them come from the academic environment, recently companies such as Limewire have also shown an increased interest in analyzing Gnutella's Network Traffic.
In this section we will describe some of the most important developments in the area of Analyzing P2P Network Traffic and Modeling P2P systems.

3.1 Peer-to-Peer Computing Concepts

A thorough analysis of Peer-to-Peer Computing is done in HP's technical report [7]. This report summarizes the key concepts of P2P Computing and gives an overview of the most important systems such as SETI@Home [8], Groove [32], Magi [25], Freenet [31] and Gnutella [30]. Although the report analyzes characteristics of P2P systems from the perspective of scalability, anonymity, self-organization, performance and others, it does not answer questions such as:

Question 1. How do these systems really look like?
Question 2. What kind of traffic are these networks carrying?
Question 3. What is the communication overhead for each particular P2P system?

These questions are probably difficult to answer given that many of the systems analyzed in this report are proprietary (e.g. SETI@Home) or hold certain properties (e.g. Magi works with X.509 certificates) which makes their analysis difficult. This is probably also the reason why most P2P new ideas are implemented on top of Gnutella which is an open protocol; it is popular and extremely interesting from the technical point of view.

3.2 Simulating Peer-to-Peer Systems

Simulating a Peer-to-Peer system can usually provide a researcher with some useful insights in the nature of the system. However simulations in most cases don't capture the big picture since initial assumptions and configuration settings may change final results significantly. The Anthill Project [10] developed at the University of Bologna uses Jtella [9] Java API as a basis for building a fully customizable API for the Gnutella network. The aim of the project is to create a simulation framework which will allow researchers to develop and validate new P2P algorithms. The system itself is inspired from the biological metaphor of Ant colonies. Although the project doesn't emphasize particularly on P2P case studies, it is worth it to mention that they are currently using their framework to investigate the properties of the Freenet [31] algorithm by modifying its protocol and comparing the performances of different implementations.

Their framework intends to obtain:

1. Information about the queries performed by users and their distribution. More specifically they aim to find popular queries or keywords that may be exploited to implement intelligent caching algorithms.
2. Information about the files stored in the Gnutella network, which might be obtained by logging the Gnutella QUERYHIT messages.
3. Information about the shape of the network, which might be obtained by actively probing Gnutella PING and PONG messages. They also intend to take advantage of the Gnutella PUSH messages in order to partially investigate which files are downloaded by users.

Although Anthill uses the notion of *scenarios*, which is composed of a collection of interconnected nodes and a scheduling of requests to be performed, there is no documentation on that. Nevertheless we mention that typical Simulation Test beds usually have of two components: (1) The *Network Graph Generator* and (2) the *Network Nodes*. A P2P Simulator is also described in the appendix of [13].

The Network Graph Generator, usually takes as an input the following parameters:

1. Number of Nodes in the Network
2. Topology of the P2P network (e.g. tree, tree-with added cycles, random or power-law)
3. Branching factor for tree topology, if a tree is used.
4. Constant Outdegree of a node, in the case a random graph is used.
5. Outdegree exponent for power low, if a power low graph is used.

Other Simulators, such as ours, can also generate outputs which can be piped into Visualization Tools, such as GraphViz [14], and generate graphical representations of the network topology.

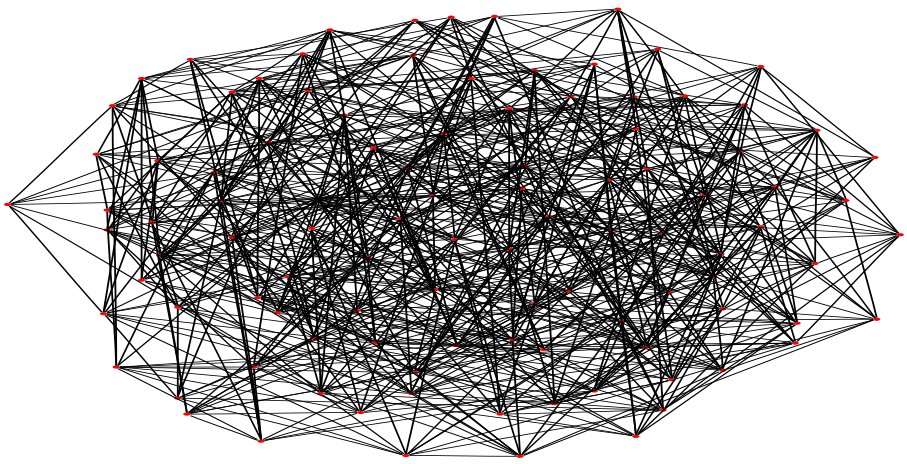

Fig. 2. Visualization of a random graph of 100 peers and outdegree=4 w/ our Graph Generator.

Figure 2, presents a random graph generated with our Network Graph Generator using Graphviz's dot 2D undirected graph layout. It is important to mention that generating visualizations for huge graphs can take a considerably large amount of time and may finally not provide the adequate help in understanding how the network looks like. Visualizing P2P network graphs is described in some extend in [2]. They try to visualize the Gnutella backbone (i.e. inter-connected nodes with $degree > 10$) rather than the whole network in order to obtain some more understandable results. We believe that Visualizations provide "nice" figures but contribute very little in understanding Large-scale P2P networks.

The Network Nodes, in simulation networks can be either part of a real network, where a node opens and maintains real network connections (i.e. sockets) or part of a program (e.g. a c program). We denote the first type of nodes as *hard-nodes* and the second type *soft-nodes* respectively. Both type of nodes will read initialization data upon initialization and behave according to the settings that where generated by the Graph Generator (i.e. open connections to the appropriate hosts). Although Hard-Nodes are making simulations much more difficult they are introducing many desirable properties, such as network failures, which are found in real settings and which might provide more accurate simulation results. Anthill uses hard-nodes while simulation efforts such as [12] use soft-nodes.

3.3 Modeling Large-Scale Peer-to-Peer Networks

Jovanovic et al. study [3], of Modeling Large-scale Peer-to-Peer Networks, is to our knowledge the only comprehensive work done in the area of modeling Peer-to-Peer systems.

Their study reveals that Gnutella has some important structural properties, such as *small-world* properties and several power-law distributions of certain graph metrics. They mention the famous Milgram's experiment [21] which was conducted in the early 1960's, and in which a number of letters, addressing a person in the Boston area, were posted to a randomly selected group of people in Nebraska. Each person who received the letter forwarded it to someone that they knew, on a first name basis. As many of the letters finally reached the designated person, the average number of hops observed was between five and six. Their study reveals that a similar, small world property existed in the Gnutella Network. More specifically in 5 different snapshots of the Gnutella Network they found that the diameter of the network ranged from 8-12.

In their work they have also discovered that the Gnutella Network obeys all four of the power-laws described in Faloutsos et al. work. [1]. More specifically they found, on a Gnutella snapshot gathered on the 28^{th} of December 2000, that the Rank Exponent R (Power-Law 1) holds with $R = -0.98$ and a correlation coefficient of $|r| = 0.94$. It is important to mention that similar results which were obtained one month earlier by an independent group at U. of Chicago [22]. The same group claims that this power

law faded-out in repeated experiments in the March-June 2001 period.
Jovanovic's study on the same snapshot of data also revealed that the Outdegree Exponent O (Power-Law 2) also holds with $O = -1.4$, although this comes in disagreement with the $O = -2.3$ exponent found in the 6 month earlier study of the DSS [27] group.
Their study finally shows that the Hop-Plot Exponent H (Power-Law 3) and Eigen Exponent E (Power-Law 4) hold for four different snapshots with very high coefficients of $|h| = 0.99$, $H = 3.5$ and $|i| = 0.94$, $E = 2.83$ respectively.

The general belief is that earlier versions of the Gnutella Network were Power-Law but as the network has grown this property doesn't hold any more. One important fact is that as the Gnutella network became more mature, more intelligent clients were added to the network. Intelligent clients can affect dramatically the way a Network Crawler operates. The latter relies on the fact that clients will respond to its requests but in the case the clients do not comply with this requirement, the Network crawler will generate inaccurate data. We believe that it would be interesting to re-examine the pre-mentioned results since the Gnutella network has undergone significant changes in terms of structure and size.

They are also presenting interesting visualizations of their gathered data, which were visualized with LEDA [23], which is not publicly available anymore. The main disadvantage of their study is that their experiments were performed on a small set of peers (1000), which is not representative of the today's picture of the network. Additionally, their Gnutella Crawler Implementation is in some sense static since it starts from a pre-specified seed file of peers and relies on the fact that it will discover new nodes on runtime.

3.4 Obtaining Real Network Data

Crawling is usually the most effective approach in order to obtain real Gnutella Traffic data. The role of Crawlers is usually to attach to a set of K Gnutella entry-points (i.e. hosts) and harvest various massages that are routed through these nodes. For example a Crawler might extract from a *pong* descriptor (Figure 3) the IP address of a particular peer as well as its path distance (i.e. hop count) from the crawler.

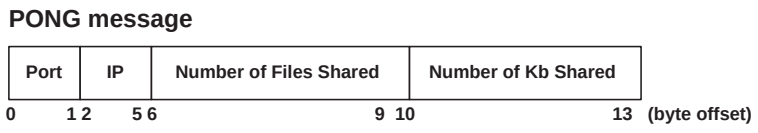

Fig. 3. A Gnutella Pong Descriptor.

Crawlers may then try to resemble all the gathered data in order to create snapshots or maps of the

network. Crawlers are actually affecting the operation of other P2P clients since they decrease their out-degree (and subsequently their horizon), without offering them any exchange (i.e. results on queries). We need to mention that crawling results are not completely accurate since a crawler may not capture a set of peers which belong to a Gnutella sub-graph.

Figure 4, which was obtained from [26], presents the number of Gnutella Peers and the number of hosts accepting new connections. Although we can't say exactly how these data were obtained, since Limewire refused to tell us, we mention that they were most probably obtained by a combination of Crawling and by utilizing their Host Caches.

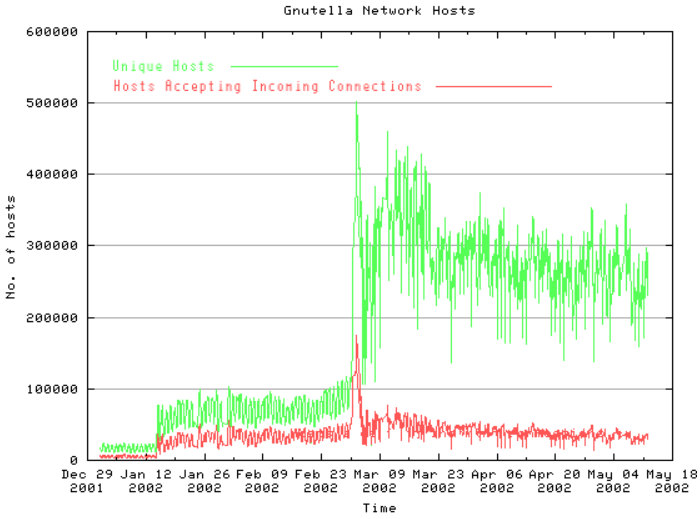

Fig. 4. LimeWire's Gnutella snapshot from 29^{th} Dec. 2001 - 5^{th} May 2002.

GnutellaMeter [34] is another system which monitors the traffic patterns of the Gnutella network by attaching itself as a passive observer to well-positioned peers in the network and by analyzing search queries and identification messages passing through the Gnutella network. GnutellaMeter presents the top 300 queries performed on the Gnutella Network. Their proprietary infrastructure makes it difficult to conclude how results are gathered. Also, It is not possible to say whether their results are real-time or not.

Clip2's DSS Group discovered in [20] several interesting features of the Gnutella network. The study which was conducted in July 2000 showed that the number of unique Gnutella users per day is no less than 10,000 and may range as high as 30,000. They also show that the *diameter* of the network was 22, indicating that some regions of the network were not in communication with others. The *diameter* of a

network is defined as the longest path between two hosts on the network. The diameter is the opposite to the notion of the shortest path between two hosts. Older results of their experiments, which were obtained two months earlier, show that the diameter was found to be smaller, typically 8 or 9. Some other fact is that a network with few hosts can have a large diameter and vice versa, since it is a function of how hosts are inter-connected and the out-degree, rather than the number of hosts participating in a given topology. This result is particularly interesting to us since it shows that the dynamics of such networks may change significantly in a short period.

Another observation that they have done is that a host was most likely to have a single connection, and hosts with higher numbers of connections were increasingly uncommon which refers to power-law degree distributions. We need to recall, from the previous sub-section, that similar behavior was obtained on about the same time from both Jovanovic et al. and Ripeanu et al.

One final question that their study tried to answer is *"What are users searching for?"*. The DSS Group presents in [20] the set of queries that contained specific types of extensions. The result indicates that most users are interested in audio/video files and far less users are interested in images (which can be obtained more efficiently by Image Search Engines).

The DSS group also verified that Gnutella is a truly international phenomenon, since one of three hosts was found to be located on a non US-centric domain. Their study analyzed 3.3 million addresses, of which 1.3 million (39%) were resolvable to non-numeric hostnames. On this subset of addresses they found that the ratio of domination was $19 : 8 : 2 : 1$ for the following domains COM, NET, and EDU and combined {ORG, US, GOV, and MIL} respectively. It is interesting to repeat such experiments to see how this ratio might have changed since the Gnutella network has changed significantly in size in the meanwhile.

4 *gnuDC* - Gnutella Distributed Crawler.

In this section we will describe design and implementation issues of gnuDC, a large-scale distributed Gnutella Crawler. gnuDC allows us to obtain various large-scale Gnutella network traffic metrics. Crawlers, spiders or bots are well known in the WWW world. These WWW Crawlers [15] typically traverse the hypertext structure of the Web automatically, starting from an initial hyper-document and recursively retrieving all documents accessible from that document. P2P Crawlers on the other hand need to connect to some pre-specified *Host-Cache* server, obtain a set of peers active in the network, try to connect to these peers and finally discover more and more peers while the operate. A determinant factor between WWW Crawlers and P2P Crawlers is that the second ones try to discover a topology which is highly dynamic. For this reason P2P Crawlers need to be able to discover the entire topology in a relatively small interval. WWW Crawlers on the other hand can operate for weeks or months in order to accomplish their task.

4.1 Design issues of a Large Distributed P2P Crawler.

A Large Distributed P2P Crawler needs to hold several properties in order to make the process of discovering a network topology efficient. Jovanovic et al. study in [2] the design issues of a sequential and parallel P2P crawler. The main drawback of their design is that it is based on the assumption that an initial set of all peers in the network is available. This was probably a reasonable assumption, on the time of their study since the largest connected public segment was rarely exceeding $2,000$ hosts. This is obviously not true any more since the Gnutella Network has grown dramatically. In the experimentation section we show that the number of unique IPs discovered in a 5 hours interval were $300,000$. Below we describe the main factors that guided our design:

1. **Obtain all Network Statistics in a small Interval** K. Since the Gnutella Network is typically very large, discovery of resources can't be done sequentially (i.e. with one node). *gnuDC* tries to address the issue of obtaining network data in a small interval K by parallelizing the process of a single Gnutella Node. A typical way of achieving that is by partitioning the number of hosts to be discovered among the several Gnutella Nodes that will participate in the Crawl and to harvest data with a large set of nodes.

2. **Scale with the Network Size** A Large-Scale Crawler must be able to scale as the size of the real Network increases. Centralized approaches such as SETI@Home [8] or Napster [28] have proven that it is feasible to scale with many millions of users while maintaining centrally index structures. The main target of distributed systems with centralized indexes is to reduce interference and move only questions and answers through the network rather than the actual data. gnuDC is implemented based on a *share-nothing* [16] scheme were each of our Gnutella client runs in its own memory space and logs information on local disks (e.g. /tmp). This model provides our crawler great flexibility and scalability. Log Traces which are dispersed on disks of several remote disks can be collected both at runtime and at the end.

3. **Maintain Network Health**. Since many hundreds of Nodes will attach to Gnutella Network their operation shouldn't affect the regular operation of the network. Typically every time a message traverses a Gnutella Node the *Hop Count* of the message is increased and its *TTL value* is decreased. In this way a Gnutella message will eventually terminate after some hops (typically 7). If each of our crawlers modified messages as they were traversing them then we would affect negatively users performing searches on the Gnutella Network since their *horizon* would decrease. For this reason we decided not to change the status of any message traversing one of our nodes.

4. **Platform-Independence**. Networks of workstations are usually consisted of a large number of workstations each of which might run a different Operating System (e.g. Linux, SunOS, Unix). This makes the deployment of such systems a difficult task since different executables must be build for each of these operating systems. Java on the other hand provides an elegant way to overcome the platform-independence problem since it is based on the "write once, run everywhere" philosophy. It furthermore provides support for networking (i.e. sockets), threads, RMI and many others. For these reason we chose to implement our system in JAVA since it made it an ideal choice for our application.

5 The Architecture of gnuDC

gnuDC is a Gnutella Distributed Crawler which obtains Gnutella network traffic measurements while addressing the four design issues described in the previous section. Figure 5 illustrates the four basic components of gnuDC, the *IP Index Server*, *gnuDC bricks* which are standalone Gnutella Clients, an online *Log Aggregator* and *Log Analyzer*. These components operate independently and asynchronously. The whole system can easily be deployed on a cluster or network of workstations since it can be bootstrapped by the execution of a single *bash* script.

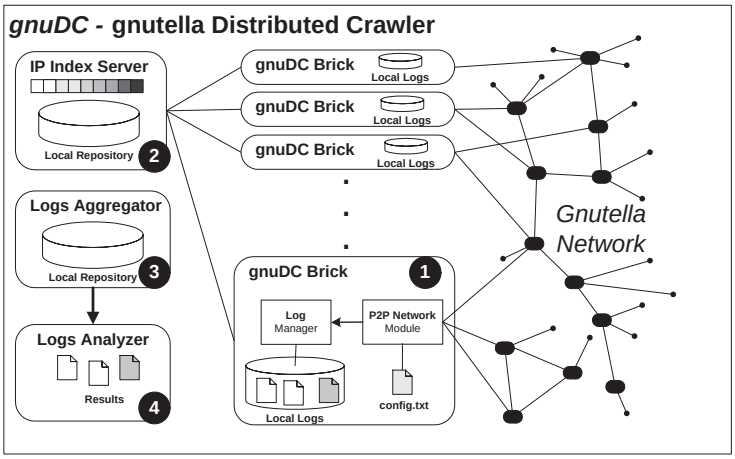

Fig. 5. gnuDC - Gnutella Distributed Crawler.

5.1 IP Index Server.

IP Index Server is the component responsible for maintaining at runtime an updated list of hosts currently in the network. Every gnuDC Brick, i.e. one of our Gnutella clients participating in the crawling,

will establish a permanent socket link upon initialization with the Index Server and feed it with hosts' information as they are discovered. The index server is the only centralized component and hence we have paid close attention to its design and implementation. It is implemented as a Multithreaded Server and can easily be configured to accept one hundred connections. Although the Server maintains an in-memory index structure for the state of each IP currently in the network it also flushes its state to secondary storage with *double buffering* techniques. In this way it minimizes the I/O cost while providing on the same time persistency.

We have conducted several performance experiments with automated clients in order to benchmark the performance limitations of the IP Index Server. We ran this experiment with 40, 80 and 120 automated clients respectively. Each of the automated clients was generating random IPs and was submitting them, through the permanent socket link, to IP Index Server. The Index Server was indexing the IPs in an in-memory data structure while it was also storing, in chunks of 1000 IPs, the new IPs as they arrived in the system.

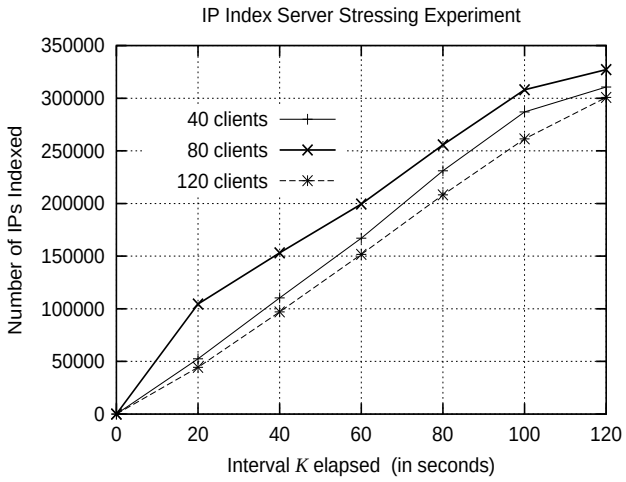

Fig. 6. Benchmarking the IP Index Server.

Figure 6 illustrates that the Index Server can sustain considerably high amounts of load. Our measurement shows that the IP Index Server was accepting averagely $2,500$ *IPs/sec* with a peak of $5,000$ *IPs/sec*. The cost for maintaining the in-memory data structure was averagely $300MB$ for $240,000$ IPs. We mention that snapshots of the Gnutella Network, maintained by Limewire [26], suggest that there are averagely $250,000$ peers at any given moment. Given that we found it satisfactory to deploy our system since we had averagely $500MB$ in our availability. In the same figure we can also see that although the optimum number of clients would be 80, using 120 clients has no important performance penalty.

5.2 gnuDC Bricks.

A *gnuDC Brick* is a Gnutella Client which will connect to the Gnutella Network and Log various messages as they traverse that node. For the implementation of a gnuDC Brick we have used as a basis the JTella[9] API, which is a Java API for Gnutella. Each gnuDC Brick configures itself by reading the `conf.txt` file which contains information such as *Maximum Incoming & Outcoming Connections*, *Socket Timeouts*, *Port Number*, *Log Directory* and others. Although the initial port number is identical for all gnuBricks (i.e. port=11000), a gnuBrick will seek for a different port number if 11000 is occupied. In this way a gnuBrick can start without any particular problem on any machine it is launched.
Each gnuBrick maintains a *Peer List* which contains the IP addresses of known peers in the network. Every time a new IP is discovered or lost the IP Index Server is notified through the permanent socket link which is a Gnutella servent API.

5.3 Log Aggregator.

The *Log Aggregator* is the component responsible for collecting the harvested data from the local disks of all gnuBricks and copying them to a centralized folder. This procedure can be done while the rest system is active and hence allows us to obtain at real-time network traffic metrics. The *Log Aggregator* is implemented as a set of *Unix Bash scripts* and uses *ssh* to connect to the set of machines that are participating in a given Crawl. These hosts are identified in a file named *crawler_hosts.txt*. In this way we can easily adapt the crawler to any set of remote hosts.

5.4 Log Analyzer.

The *Log Analyzer* is responsible for analyzing the harvested data remove unnecessary information from the log traces, count occurrences of messages, find the top queries and many other statistics. For its implementation we have used a combination of *bash scripts*, *C++ routines* as well as *Java Programs* where the need of each particular language method was more appropriate. We mention that runtime statistics and graphs can be obtained while the system is running. The process for Aggregating and Analyzing the collected data takes between $5 - 10$ minutes for $700MB$ of log traces.

6 Experiments.

In our experiments we deployed gnuDC on 85 nodes running on 17 workstations. All workstations were AMD Athlons 4, 1.4 GHz with 1GB RAM running Mandrake Linux 8.0 (kernel 2.4.3-20) interconnected with a 10/100 LAN connection. Before launching a massive crawl we performed several test experiments to ensure the stability of our system. On the 1st of June 2002, we performed our first "long" crawl which

lasted five hours. After that we performed several other small scale experiments to gather data on specific issues.

6.1 Technical Difficulties.

Unfortunately, we were not able to perform long lasting crawls since our system was running on a set of Lab Machines which are used by students during weekdays. Hence we adopted the policy to perform experiments only during early morning hours (i.e. 1:30 a.m. - 6:30 a.m.). Another problem was that our system was collecting huge amounts of log traces. As an example we mention that the system created 700 MB of log traces in the relatively small interval of 5 hours. It was consequently infeasible to store all this data on our accounts due to quota limitations. Our last and most important problem was that the Department's Administrators have blocked any remote access (i.e. establishing a TCP connection on any port number of a lab machine). In this way we were not able to accept any *incoming connections*. This was apparently important since it reduced dramatically the degree of a gnuBrick from 100 connections (which was our default setting) to $10-30$ connections. Limewire shows that most Gnutella users are not accepting any new incoming connections. We believe that for the same reason our cluster was not able to establish too many outgoing connections. On the other hand if we were able to accept incoming connections then we would most probably satisfy many "thirsty for outgoing connections" peers. Nevertheless we mention that a different internal network setting would automatically resolve this issue.

6.2 Analysis of Gnutella Messages.

In this subsection we describe some interesting results regarding the messages that were routed through our cluster. We mention that the sample that we analyze exceeds 56 million messages.

Figure 7 presents the message breakdown by Message Type. We may see that only a very few Push messages (i.e. $3,000$) were routed through our cluster and hence they are not presented in the pie chart. Figure 7 confirms that Gnutella has actually a huge communication overhead (i.e. ping/pong messages), since it is averagely 63% of all the messages. We believe that is due to the fact that Gnutella connections are relatively unstable which leads peers in a endless effort of discovering new peers rather than to the fact that peers are joining and leaving the network at fast paces. *Pong Stealing* might be a reason why a peer is attracting more network load than the load it can really handle. We define as *Pong Stealing* when an intermediate node B is obtaining the ip address, from a Pong message which is routed through him. In this situation a host C which replied positively to a ping request of a host A might get a connection request from both A and B although its initial intention was to accept only 1 connection. This phenomenon might destabilize $C's$ current network connections since C is forced to; at least, abort the one of the two requested connections. Another reason for network instability might be that some "powerful" host A is

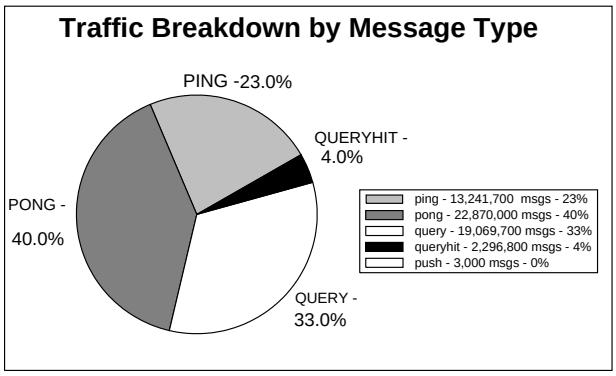

Fig. 7. Messages Breakdown by Message Type.

routing many messages to another "weak" host "B" with a result that B is kept so busy that it is not able to handle its rest connections. if B is implemented appropriately (i.e. keep different queues for different connections) then of course such a thing won't occur but our point here is that this really depends on the implementation of each Gnutella client.

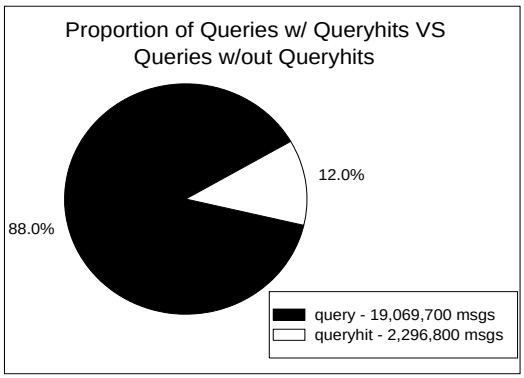

Fig. 8. Proportion of Queries w/ QueryHits VS Queries w/out Queryhits.

Figure 8 presents the amount of queries routed through our cluster versus the amount of queries that were routed with queryhits. A similar measurement was obtained in [18] where the percentage of Queries with Queryhits varied between 10-12%.

6.3 Analysis of Queries.

In this subsection we discuss our observations after analyzing 15,153,524 unique queries. Here we found various interesting patterns such as high locality of specific queries and we were also able to distinguish Gnutella Users into three classes. Table 1 presents the ranking of the top 20 queries. We can clearly see that most queries are submitted in large numbers and hence there exist a high locality of specific queries. This observation might lead to better caching policies at peers which might cache Queryhits that are posted in response to popular queries. Another interesting observation is that Gnutella Users can be classified into

#	Query	Occurrence	%	#	Query	Occurrence	%
1	divx avi	588,146	3,88%	**11**	divx	24,363	0,16%
2	spiderman avi	50,175	0,33%	**12**	spiderman	23,274	0,15%
3	p__ mpg	39,168	0,25%	**13**	xxx avi	22,408	0,14%
4	star wars avi	38,473	0,25%	**14**	capture the light	21,651	0,14%
5	avi	29,911	0,19%	**15**	buffy mpg	20,365	0,13%
6	s_ mpg	27,895	0,18%	**16**	g__ mpg	20,251	0,13%
7	Eminem	27,440	0,18%	**17**	buffy avi	19,874	0,13%
8	eminem mp3	25,693	0,16%	**18**	t__ mpg	19,492	0,12%
9	dvd avi	25,105	0,16%	**19**	seinfeld vivid	18,809	0,12%
10	b____	24,753	0,16%	**20**	xxx mpg	18,686	0,12%

Table 1. Top 20 Queries on Gnutella. (inappropriate queries marked with '_')

three main categories *Seasonal-Content Searchers*, *Adult-Content Searchers* and *File Extension Searchers*.

Seasonal-Content Searchers are those who are seeking for content that is currently popular, such as new movies or chart songs. *Spiderman* for example was a popular term since its new movie was released around the time we conducted the experiments. Finally popular TV shows, such as "Seinfeld" were also among the top searches in this category.

Adult-Content Searchers are users which are looking for mature content. These searchers seem not very selective since they are looking for content that contains a general Adult-Content term rather than a specific movie. We have observed from other informal experiments that these users are constant over time and that the text patterns they are searching for are not changing very much.

Finally *File Extension Searchers* are those which are not looking for something particular but which are rather interested to download anything that may seem interesting to them but which is of a specific file-type (e.g. avi, mp3). In this category we may find for instance a user who is searching for the term "mp3". It is clear that this user has not something specific in its mind but that he is rather interested to

"browse" the mp3 files shared by his fellow file-sharers. Our informal experiments, which were conducted in May 2002, show that this category is also showing some constant search patterns over time.

These three categories might actually have a large overlap. For example a user searching for seasonal content might as well perform file extension searches.

Gnutella Users are mainly interested in video media rather than audio media. This differentiates Gnutella users from other File-sharing applications such as Napster where audio was the only available media. The trend that most users are seeking for multimedia content might also reveal that they are "bandwidth-capable" of downloading such media. Finally Gnutella users seem not interested in other type of content such as software, images or text since their aggregate percentage is not very large.

Table 2 presents the demand in specific filetype extensions. Our analysis indicates that the trends have not changed significantly from the DSS's study which was performed in October 2000. The table validates that the users in Gnutella are seeking for Multimedia Content (audio/video) since their aggregate exceeds 65% of all filetypes searched. Although the trend for demand of multimedia in filesharing applications is not new we mention that the trend nowadays is constantly towards video instead of audio, which was the main media exchanged in the napster [28] community.

#	Filetype	Occurrence	%	#	Filetype	Occurrence	%
1	avi	2,837,002	18,72%	**11**	mov	144,193	0,95%
2	mp3	2,703,551	17,84%	**12**	pdf	76,914	0,51%
3	mpg	1,985,354	13,10%	**13**	rar	66,644	0,44%
4	ra	1,287,578	8,50%	**14**	exe	60,176	0,40%
5	rm	422,047	2,79%	**15**	wav	37,690	0,25%
6	zip	400,057	2,64%	**16**	doc	31,740	0,21%
7	mpeg	398,739	2,63%	**17**	txt	11,287	0,07%
8	jpg	288,427	1,90%	**18**	gz	11,070	0,07%
9	asf	168,531	1,11%	**19**	html	3,755	0,02%
10	ps	146,288	0,97%	**20**	jpeg	2,326	0,02%

Table 2. Top 20 Filetypes Requested on Gnutella (in queries).

Figure 9 shows a 4 minute snapshot of ping/pong and query/queryhit messages routed through our cluster. There is clearly some analogy between ping/pong and query/queryhit pairs. This is because a ping for instance; will generate many pong messages. (averagely four times as many) while a query will only generate queryhits the one eight of the times. We can also see that there is some relation between the two

graphs. This is attributed to the fact that if we have more ping/pong messages (i.e. more hosts) then we will observe also more query/queryhit messages. This implies that users are actually actively searching the network for content. In other words this observation tells us that there are only few or no *pathetic peers* (i.e. peers that are connected to the network without performing queries).

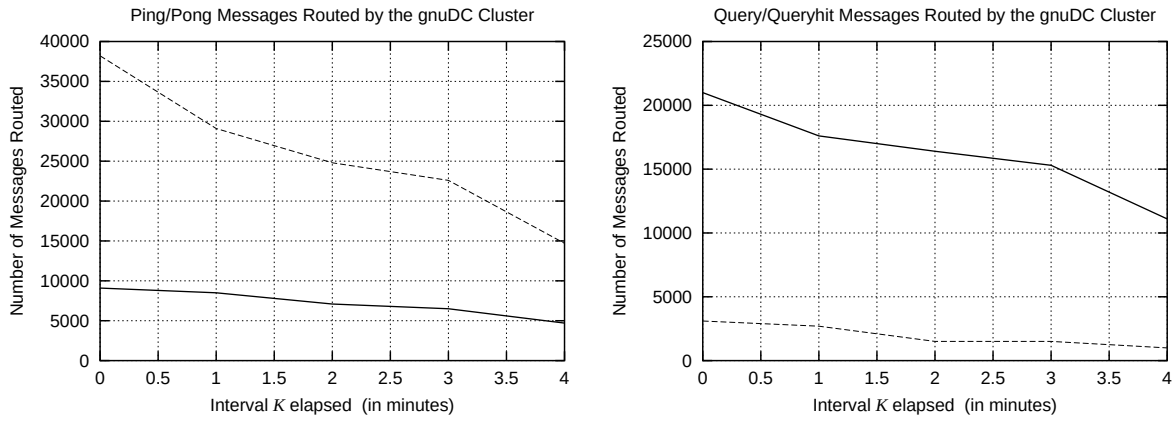

Fig. 9. a) Ping/Pong Messages Routed Trace **b)** Query/QueryHit Messages Routed Trace.

6.4 Analysis of IP Addresses.

In this subsection we discuss our analysis of 294,000 Unique IP Addresses that were gathered from *pong* messages (see figure 3). that were routed through our system on the experiment of the 1^{st} of June 2002. Every unique IP Address that is discovered by a gnuBrick is posted, through a permanent socket link to the IP Index Server. Figure 10 present the pace at which the IP Index Server indexes IP addresses as they were posted by the gnuBricks.

Each gnuBrick maintains a local Hashtable of all IPs that it has seen before in order to avoid sending duplicate IP addresses to the Index Server. Of course since there is no central coordination of the gnuBricks it is possible that two gnuBricks send the same IP Addresses to the Index Server. In this case the Index Server is responsible to filter out duplicates since it maintains a global view of all IP addresses observed by the system.

For the analysis of the IP Addresses we have written a *Multi-Threaded Reverse DNS Lookup (MRDL)* engine which finds the DNS entries of IP addresses obtained by the system. *MRDL* takes as an input a list of N unique IP addresses , partitions them into k buckets, where k is the number of threads that will work towards resolving the IP addresses. After the partition phase the system launches k threads which

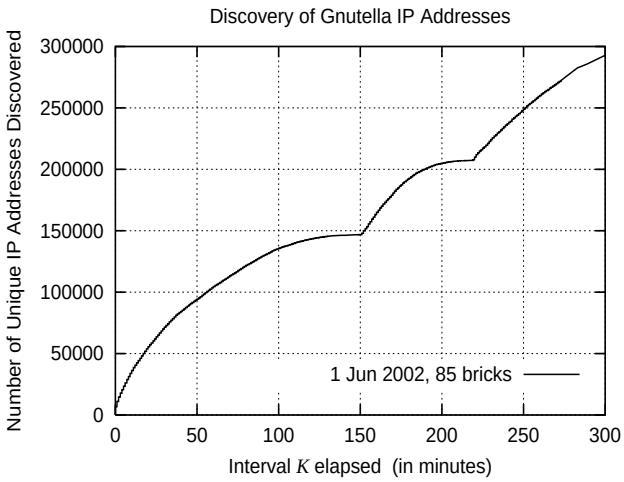

Fig. 10. Discovery of IP Addresses.

obtain the DNS entries of the IP addresses they where assigned to. The MRDL engine was operating with 100 threads and we were able to obtain all DNS entries in a period of two hours.

The MRDL engine ended up with a set of $244,522$ resolved IP addresses. An aggregate of $49,478$ or $16,92\%$ were not resolvable. We mention that the non-resolvable set of IPs contain both hosts which where not reachable at the time of the resolution as well as IP addresses which are allocated for private networks [19] (i.e. 192.x.x.x, 172.16.x.x and 10.x.x.x).

From which domains are Gnutella Users coming from? In this subsection we wanted to figure out from where the Gnutella users are actually coming from. Clip2 [27] reported in 2000 that Gnutella was a truly international phenomenon. Our measurements indicate similar behavior with a main distinction. We observed that although Gnutella users are coming from around the globe, most of them come from only a few countries (U.S.A., Germany, Canada, France and England).

We believe that this observation is apparently important since if these countries are inter-connected with high-speed networks then it probably it would not matter that the underlying Gnutella topology doesn't match the real topology (topology mismatch).

Table 3 presents the top 20 domains from which Gnutella users are coming from (see table 5). Although it was expected that both .net and .com domains will dominate in this measurement, since these domains are globally used by ISPs, we also found that the number of Gnutella users from various domains is more a function of how advanced the networks of the various ISPs in these countries are rather than the actual number of Internet users in these countries. For example we expect that the Australian domain must have far larger number of Internet users than the English domain but the number of Gnutella users are larger

#	Country	Dom.	IPs	%	#	Country	Dom.	IPs	%
1	Network	.net	94,456	38,88%	**11**	Belgium	.be	2,527	1,04%
2	US Commercial	.com	81,943	33,73%	**12**	Italy	.it	2,038	0,84%
3	Canada	.ca	8,039	3,31%	**13**	Sweden	.se	1,532	0,63%
4	France	.fra	5,565	2,29%	**14**	Spain	.es	1,495	0,62%
5	US Educational	.edu	5,102	2,10%	**15**	Singapore	.sg	1,333	0,55%
6	England	.uk	4,118	1,69%	**16**	Switzerland	.ch	1,256	0,52%
7	Germany	.de	3,693	1,52%	**17**	Japan	.jp	1,089	0,45%
8	Australia	.au	3,663	1,51%	**18**	Norway	.no	1,010	0,42%
9	Austria	.at	2,962	1,22%	**19**	Brazil	.br	775	0,32%
10	Netherlands	.nl	2,625	1,08%	**20**	New Zealand	.nz	651	0,27%

Table 3. Distribution of Gnutella IP Addresses to Domains.

for the English domain.

The next table 4 concentrates on only ISPs of the .net and .com domains and tries to determine which of these ISPs are paying the price of the Gnutella Network. In this table we can see that the German ISP (T-Online) contributes with the largest number of IP addresses if we compare it with its rest .net "competitors". In the .com column we can see that the first three ranks are occupied by USA's ISPs.

.NET Companies					**.COM Companies**				
#	ISP	Dom.	IPs	%	#	Country	Dom.	IPs	%
1	T-Online	t-dialin.net	14,998	15,88%	**1**	Road Runner	rr.com	21,834	26,65%
2	Comcast	comcast.net	9,221	9,76%	**2**	American Online	aol.com	17,343	21,16%
3	Cox Comm.	cox.net	7,757	8,21%	**3**	AT & T	attbi.com	13,939	17,01%
4	Shaw	shawcable.net	5,321	5,63%	**4**	Rogers Comm.	rogers.com	3,758	4,59%
5	CSC Holdings.	optonline.net	4,830	5,11%	**5**	ntl Group	ntl.com	3,109	3,79%

Table 4. *a)* IPs contributed by .net organizations **b)** IPs contributed by .com organizations.

Table 5 presents the overall ranking of ISPs based on their share of Gnutella Hosts they are contributing to the Network. We can see clearly that US, Canadian, German, French and English organizations are dominating the Gnutella network. This table shows that the largest part of the Gnutella network is occupied by only a few countries. The table also reveals that Asian countries that have advanced networks, such as Japan, are not particularly active in this community although their popular Napster-like *File Rogue* [33]

service was suspended.

Overall Ranking of Organizations (ISPs)									
#	ISP	Domain	Country	%	#	ISP	Domain	Country	%
1	Road Runner	rr.com	US.	9, 43%	**11**	Adelphia Comm.	adelphia.net	US.	1, 73%
2	American Online	aol.com	US.	7, 49%	**12**	Wanadoo	wanadoo.fr	France	1, 67%
3	T-Online	t-dialin.net	Germany	6, 48%	**13**	Rogers Comm.	rogers.com	Canada	1, 62%
4	AT & T	attbi.com	US.	6, 02%	**14**	Woolworths Gr.	co.uk	England	1, 58%
5	Comcast	comcast.net	US.	3, 98%	**15**	ntl Group LTD	ntl.com	England	1, 34%
6	Cox Comm.	cox.net	US.	3, 35%	**16**	Verizon	verizon.net	US.	1, 27%
7	Shaw	shawcable.net	Canada	2, 30%	**17**	SBC Pacific Bell	pacbell.net	US.	1, 26%
8	Sympatico Lycos.	sympatico.ca	Canada	2, 15%	**18**	Verizon (DSL)	dsl-verizon.net	US.	1, 03%
9	CSC Holdings	optonline.net	US.	2, 09%	**19**	British Telec..	btopenworld.com	England	1, 00%
10	BellSouth Telec.	bellsouth.net	US.	2, 00%	**20**	SBC Internet	swbell.net	US.	0, 94%

Table 5. Overall ranking of domains based on the number of hosts they contribute to the Gnutella Network.

6.5 Analysis of Hop Count found in Query Messages.

In this subsection we describe the analysis we have performed on the hop count of the 15 million queries sample. The hop count reveals how many hops have query messages travelled before reaching one of our crwalers. Figure 11 presents the distribution of queries to hop count $1-14$ while the largest observed hop count of a query was 16. The figure is particularly interesting since it reveals that the Gnutella network started conforming to its specifications. More specifically we mention that from the $15,153,524$ unique queries that the gnuDC cluster logged, only $22,423$ queries had traveled more than 7 hops before traversing one of the gnuDC Bricks. Recall that the Gnutella protocol advises that packets shouldn't travel more than 7 hops and that packets that have a greater value than that should be discarded. We can see that some of the Gnutella clients $22,423$ are still sending queries with TTLs more than 7. The obvious reason for doing such a thing is to get as many answers as possible from the network. Such a behavior is destabilizing the network health since it leads to huge amounts of message forwards for 1 query. Fortunately it seems that the Gnutella peers became conscious and that they thwart such a behavior. The next interesting point of the graph is its bimodal distribution with two peaks both at 1 and 7. A similar distribution was also reported in [17] although their query sample was very small (i.e. averagely 8000 queries). The number of queries that travelled 7 hops is larger than the number of queries traveled 3-6 hops since as the hop count increases we are also accepting queries from more hosts. The reason that so many queries have traveled

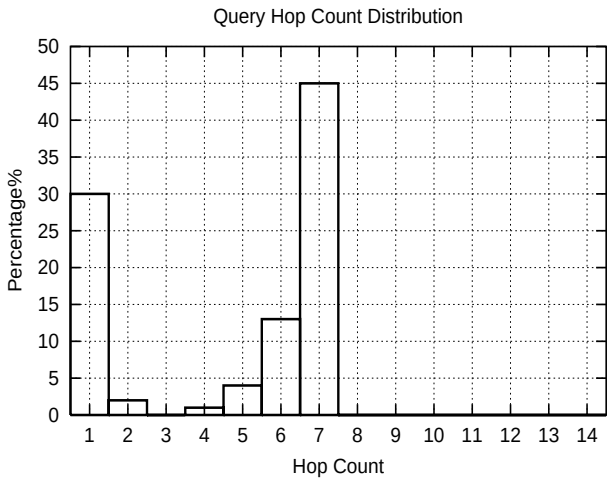

Fig. 11. Distribution of Hops that Query Messages have traveled before reaching our gnuDC cluster.

only 1 hop is probably because network connection intervals between peers can be sometimes very short. For this reason as soon as a connection between 2 peers is formed both peers first promote their own messages and if the connection is still active then they also route messages which are coming from other peers (i.e. $hopcount > 2$).

7 Conclusions.

In this paper we have performed a quantitative analysis of the Gnutella network traffic. We have designed and implemented *gnuDC*, a distributed Gnutella Crawler which was used to harvest network data with 85 nodes which were running on parallel on 17 workstations. gnuDC enabled us to gather large amount of network data which were analyzed both offline and online.
Our analysis on the network traffic revealed and/or confirmed the following things:

1. The Gnutella communication overhead is huge. More specifically we found that *ping/pong* messages occupy 63% percentage of all messages while the *useful utilization* of the network (i.e. query/queryhit) was only 37% percentage.

2. Gnutella Users seem to belong to three main categories *Season-Content Searchers*, *Adult-Content Searchers* and *File Extension Searchers*. *Season-Content Searchers* are those who are seeking for content that is currently popular, such as new movies or chart songs. *Adult Content Searchers* are users which are looking for mature content and *File Extension Searchers* are those which are not looking for something particular but which are rather interested to download anything that may seem interesting

to them but which is of a specific filetype (e.g. avi, mp3).

3. Gnutella Users are mainly interested in video media rather than audio media. This differentiates Gnutella users from other File-sharing applications such as Napster where audio was the only available media. The trend that most users are seeking for multimedia content might also reveal that they are "bandwidth-capable" of downloading such media. Gnutella users seem not interested in other type of content such as software, images or text since their aggregate percentage is not very large.

4. Most clients seem to have conformed to the specifications of the Gnutella protocol and that efforts of over-allocating networking resources, with large TTL values in messages, are thwarted.

5. Finally although Gnutella is a truly international phenomenon its largest segment is dominated by only service providers which belong to a few countries (i.e. US, Canada, France, Germany and England).

Further metrics will be obtained in future experiments in order to see how these trends might change over time. We are also interested in examining more carefully other data that we have obtained but which we couldn't analyze due to time shortage. We finally mention that such metrics might facilitate the development of more advanced P2P protocols which might take into consideration various bottlenecks of the current Gnutella Protocol.

References

1. Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In SIGCOMM, pages 251-262, 1999.
2. M. Jovanovic, "Modeling Large-scale Peer-to-Peer Networks and a case study of Gnutella", Master's Thesis, University of Cincinati, April 2001.
3. M. Jovanovic, F.S. Annexstein, and K.A. Berman. Modeling Peer-to-Peer Network Topologies through "Small-World" Models and Power Laws. In TELFOR, Belgrade, Yugoslavia, November, 2001
4. F.S. Annexstein, K.A. Berman, and M. Jovanovic. Latency Effects on Reachability in Large-scale Peer-to-Peer Networks. In ACM Symposium on Parallel Algorithms and Architectures, Crete Island, Greece, 2001.
5. H. Stoica, I. Morris, R. Karger, D. Frans Kaashoek, M. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications", SIGCOMM 2001.
6. S. Gribble, A. Halevy, Z. Ives, M. Rodrig, D. Suciu. "What Can Databases do for Peer-to-Peer? WebDB Workshop on Databases and the Web, June 2001.
7. D. Milojicic, V Kalogeraki, R. Lukose, K Nagaraja, J Pruyne, B. Richard, S Rollins, Z. Xu, "Peer-to-Peer Computing", Technical Report HPL-2002-57, HP Labs. 2002
8. "The SETI@home (Search for Extraterrestrial Intelligence at Home) Project", UC Berkeley, `http://setiathome.ssl.berkeley.edu/`.

9. Ken Mccrary, "The JTella Java API for the Gnutella network", October 2000, http://www.kenmccrary.com/jtella/.

10. The Anthill Project, Gnutella Monitoring, Department of Computer Science, University of Bologna.

11. G. Pandurangan, P. Raghavan, E. Upfal, "Building P2P Networks with Good Topological Properies", Brown University, 2001

12. Ramanathan, K. Murali, V. Kalogeraki, J. Pruyne, "Finding Good Peers in Peer-to-Peer Networks", Technical Report HPL-2001-271, HP Labs, 2001.

13. Arturo Crespo and Hector Garcia-Molina, "Routing Indices for Peer-to-peer Systems", In ICDCS, 2002.

14. Stephen North, Emden Gansner, John Ellson, "Graphviz - open source graph drawing software", http://www.research.att.com/sw/tools/graphviz/

15. D. Zeinalipour-Yazti, M. Dikaiakos, "Design and Implementation of a Distributed Crawler and Filtering Processor," The Fifth Workshop on Next Generation Information Technologies and Systems (NGITS'2002) , Caesarea, Israel, June 25-27, 2002.

16. David DeWitt and Jim Gray,"Parallel database systems: the future of high performance database systems", Communications of the ACM, vol. 35, no 6, pages 85–98, 1992.

17. Kelsey Anderson, "Analysis of the Traffic on the Gnutella Network", University of California, San Diego CSE222 Final Project, March 2001.

18. Evangelos P. Markatos: Tracing a large-scale Peer to Peer System: an hour in the life of Gnutella. In the Proceedings of the CCGrid 2002: the second IEEE International Symposium on Cluster Computing and the Grid, May 2002, pages 65-74.

19. Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear, "RFC1918 - Address Allocation for Private Internets", February 1996.

20. Clip2, "Gnutella: To the Bandwidth Barrier and Beyond", November 6, 2000, http://www.clip2.com/gnutella.html

21. Stanley Milgram experiment description, available at http://smallworld.sociology.columbia.edu/description.html.

22. M.Ripeanu, "Peer-to-peer Architecture Case Study: Gnutella Network", Technical Report, University of Chicago, 2001.

23. Leda, Algorithmic Solutions Software GmbH, http://www.mpi-sb.mpg.de/LEDA/leda.html.

24. Morpheus. , MusicCity.com, http://www.musiccity.com/.

25. Magi Enterprise, Endeavors Technology, *http://www.endeavors.com/*.

26. LimeWire. , LimeWire.com, http://www.limewire.com/.

27. Clip2. , Clip2.com, http://www.clip2.com/.

28. Napster, Napster.com, http://www.napster.com/.

29. Kazaa, Kazaa.com, http://www.kazaa.com/.

30. Gnutelliums, Gnutella, http://www.gnutelliums.com/.

31. Freenet, FreenetProject.org, http://freenet.sourceforge.net/.

32. Groove, Groove Networks Inc. http://www.groove.net/.

33. File Rogue, File Rogue Inc. http://www.filerogue.com/.

34. The GnutellaMeter, http://www.gnutellameter.com/gnutella-hosts.html.