Scalability Issues in Large Peer-to-Peer Networks - A Case Study of Gnutella University of Cincinnati Technical Report 2001

Mihajlo A. Jovanovic, Fred S. Annexstein, Kenneth A. Berman
ECECS Department, University of Cincinnati, Cincinnati, OH 45221
mjovanov@ececs.uc.edu, fred@cayley.ececs.uc.edu, ken.berman@uc.edu

Abstract:

With the peer-to-peer model quickly emerging as a computing paradigm of the future, there is an ever-increasing need for distributed algorithm that would allow peer-to-peer applications to scale to a large community of users. The main difficulty in designing such algorithms is that currently, very little is knows about the nature of the network topology on which these algorithms would be operating. The end result is that even simple protocols, as in the case of Gnutella, result in complex interactions that directly affect the overall system's performance. In this paper we present a case study of Gnutella- a worldwide, distributed information sharing system. To study the Gnutella network topology, we implemented a distributed computing application that allows topology discovery to be performed in constant time - an important feature considering Gnutella's highly dynamic nature. Upon analyzing the obtained topology data, we discovered that it exhibits strong small-world properties. In addition, we observed a power-law distribution with regards to node degrees, a property previously reported in other technological networks such as the Internet and the WWW. We believe these properties of the network topology are an important step toward an accurate mathematical model and could serve as an aid in both analyzing the performance of existing algorithms and designing new, more scalable, solutions.

Introduction

The sudden emergence of new network applications such as Napster [Na00], Gnutella [Gn00], and Freenet [Fr00] [Cl00] has re-introduces a style of computing known as peer-to-peer in an attempt to provide worldwide access to information. The main obstacle to this goal of a fully distributed information sharing system seems to be the design of scalable algorithmic solutions. Algorithms such as those for routing and searching peer-to-peer networks are typically implemented in a form of an application-level protocol. The inadequacy of the existing protocol became painfully clear to Gnutella developers in the summer of 2000, when the size of the user community rapidly increased. The problem is that many peer-to-peer protocols were designed without any knowledge of the network topology on which they would be operating. In applications such as Gnutella and Freenet, much like in many social networks, this topology is determined by collective phenomena, as users connect to each other in a seemingly random manner forming a high level network. We believe understand the nature of the network topology is an important step in analyzing the performance of algorithms operating on such network. Furthermore, new algorithms can benefit by taking advantage of specific structural properties of the topology. Here we present a case study of Gnutella as a model of a pure peer-to-peer system. To discover the Gnutella network topology, we have developed a distributed network crawler based on the Gnutella protocol. Our study of the Gnutella network topology reveals two important structural properties, mainly the small-world property as defined by Watts and Strogatz, and the power-law distribution of node degrees. In the subsequent sections of this report we introduce the basic architecture of Gnutella and provide a brief survey of the related work. We then describe the implementation of our distributed computing application for discovering Gnutella network topology. Finally, we present our results consisting of several visualizations of the obtained topology data, along with analysis of some important structural properties.

Gnutella

Gnutella is a fully distributed, information sharing technology based on a peer-to-peer model. This means that users connect to each other directly through a piece of client software forming a high-level network. An instance of this client software running on a particular machine is called a host. Gnutella uses IP as its underlying network service, while the communication between hosts is specified in a form of an application-level protocol. Gnutella protocol supports four basic types of messages as shown in table 1.

Table 1: Gnutella protocol messages

Type	Description	Contains
Ping	Request for a host to announce itself	No body
Pong	Reply to Ping	IP and port of responding host; number and size of files shared
Query	Search request	Minimum speed requirement of the responding host; search string
Query Hits	Reply to Query	IP and port, speed of responding host; number of matching files and their indexed result set

The routing technique specified by the protocol is based on a constrained broadcast, where messages are passed recursively between hosts. In other words, each Gnutella hosts forwards the received ping and query messages to all of its neighbors. The response messages are routed along the same path along which the original request arrived by means of dynamically updated routing tables maintained by each host. To limit exponential spread of messages through the network, each message header contains a time-to-live (TTL) field. TTL is used in the same fashion as in the IP protocol: at each hop its value is decrement until it reaches zero, at which point the message is dropped. The remaining details related to the Gnutella protocol are not relevant to understanding this report and can be found at [Gn00].

Related Work

A lot of work has been done on modeling small-world networks in the last several years. The term small-world originated with a famous social experiment conducted by Stanley Milgram in the late 1960s. Watts and Strogatz in [Wa98] showed that other networks, such as those occurring in nature and technology, also exhibit "small-world" behavior. Without providing a strict mathematical definition, the two define small-world behavior in terms of two properties, mainly the characteristic path length and clustering. In order to quantify these properties for various networks, Watts and Strogatz defined characteristic path length L and clustering coefficient C. L is a global property of the graph, defined as the "number of edges in the shortest path between two vertices, averaged over all pairs of vertices." C, a local (node) property measuring "cliquishness" of that particular node, is calculated by taking all the neighbors of a vertex, counting the edges between them, and then dividing by the maximum number of edges that could possibly be drawn between those neighbors. To demonstrate small-world behavior of several well-known networks, Watts and Strogatz present their respective values for L and C benchmarked against a random graph of the same size. To generate graphs with small-world topology, Watts and Strogatz proposed a model based on interpolating between a highly regular ring lattice and a random graph. This model was latter generalized by Kleinberg in [Wa98], who introduced an additional parameter to the model thereby defining an entire family of random networks. Kleinberg showed that the performance of decentralized algorithm varies within this family of network models, proving the existence of a unique model within the family for which decentralized algorithms are effective. The idea most relevant to our work is that the small-world property of network topology is closely related to the design and analysis of algorithms operating on those networks. Several research groups have also reported evidence of various power law distributions in small-world network topologies, particularly in large technological networks. The most frequently cited of those power laws relates to distribution of node degrees. Faloutsos *et al* [Fa99] discovered three power laws in Internet topology at both inter-domain and router level. Examining node degree distribution, they found power law exponents to be remarkably consistent among various instances of topology data, ranging from 2.15 to 2.2. This observation led them to suggest the use of power law exponents as a way to characterize graphs. Power laws governing both in-degree and out-degree node distribution have also been observed in the case of the WWW graph, where nodes are web sites and edges are hyperlinks []. Barabasi and Albert confirmed this phenomenon for other, non-technological, small-world networks. The existence of such power laws is significant because it contradicts previously proposed models of small-world networks. These models fail to explain the existence of highly-connected nodes, a likely consequence of a power-law distribution that has been verified in practice. As a remedy, Barabasi and Albert propose an alternate model based on the concept of growth and preferential attachment. They go a step further to suggest that the scale-free nature of the degree distributions, as indicated by the remarkable consistency in the values of power law exponents, demonstrates the importance of the collective phenomena in development of many small-world networks.

Design of Our Gnutella Crawler

To discover topology of the Gnutella network, we have developed a Gnutella network crawler that we call gnutcrawl. Our strategy for discovering the Gnutella network topology takes advantage of the particular intricacy of the Gnutella protocol and is based on the following algorithm:

Pseudo-code for the topology discovery algorithm:

```
procedure buildTopoMap(List hosts, Graph map)
  for (all elements h in hosts)
    connect to h
    if (connection established) then
      send Ping message with TTL=2
      for (all Pong messages received)
        if (Pong host h2 != h) then
          add edge between h and h2 to map
          if (h2 is not in hosts) then
            add h2 to the end of hosts
          endif
        endif
      endfor
    endif
  endfor
end procedure
```

Notice that the term crawler should be taken loosely, since our algorithm does not perform the traditional BFS or DFS of the network. Instead, it requires as input a reasonably accurate and complete list of Gnutella hosts. The main reason for this is efficiency, as it will become apparent later. This strategy in effect partitions the problem of discovering network topology into two steps: discovering nodes (active hosts) and discovering edges (host connections). Since the existing Gnutella clients already handle the first step, our algorithm focuses on the second step of the topology discovery process. As already mentioned, our algorithm does not require the input list of Gnutella hosts to be neither comprehensive nor completely accurate. The reason for this is that newly joined hosts will be dynamically discovered through their connections with existing hosts, and hosts that have left the network will be ignored as the connection attempt will fail. The ability of our algorithm to work with incomplete input data is particularly important considering highly dynamic nature of the Gnutella network. In such a setting, where nodes could join and leave the network at any moment, one could argue that no host list is ever fully complete or accurate. However it is not difficult to see that the performance of our algorithm is directly related to the nature of its input list. Ideally, if the input host list was complete, our algorithm would be able to obtain topology of the network in a single step, provided that the machine on which the algorithm was running is capable of maintaining up to n simultaneous network connections, where n <= input list size. Such performance is optimal for any topology discovery algorithm based on Gnutella's application-level protocol. Unfortunately, with the size of the Gnutella network typically exceeding 1000 hosts, it becomes impossible to satisfy practical requirements for achieving such performance using only a single workstation. This is why we have modified our original algorithm to operate in a distributed fashion. The distributed version of our algorithm can be written as follows:

Pseudo-code for the distributed topology discovery algorithm:

```
procedure buildTopoMap(List hosts, Graph map)
  startIndex = (size of hosts / numberOfProcs) * procID
  endIndex = startIndex + (size of hosts / numberOfProcs) - 1
  List subset = hosts[startIndex..endIndex]

  for (all elements h in hosts)
    connect to h
    if (connection established) then
      send Ping message with TTL=2
      for (all Pong messages received)
        if (Pong host h2 != h) then
          add edge between h and h2 to map
          if (h2 is not in hosts) then
            add h2 to the end of hosts
          endif
        endif
      endfor
    endif
  endfor
end procedure
```

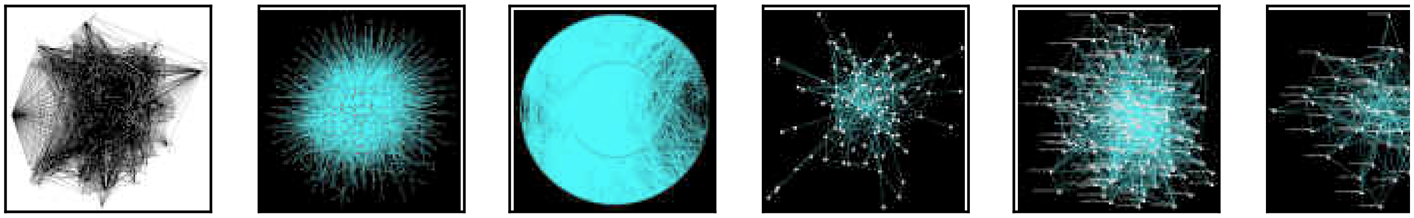
Each processor participating in the computation receives a complete host list and then runs a simple distributed algorithm to determine the sublist for which it is responsible, based on its processor number and the total number of processors. Having each processor receive a copy of the entire host list is necessary to prevent them from duplicating the work done by other processors without requiring additional inter-processor communication. In fact this algorithm requires no communication between the processors except at the beginning of the computation (to distribute the input list) and at the end (to assemble the complete topology graph). The remarkable efficiency of our algorithm stems from highly asynchronous nature of the task. As long as the size of the input list is less than or equal to pb, where p is the number of processors and b is the maximum number of connections each processor can maintain simultaneously, the algorithm will operate in constant time. With the current size of the largest connected public segment of the Gnutella network typically around 1000 hosts, we were able to satisfy this requirement using only a handful of workstations. Additional overhead could be incurred in the input host list is not exhaustive, since according to our algorithm, each processor would discover new hosts independently. However it is easy to show that, as long as the number of newly discovered hosts is within b, performance of our algorithm will be within a factor of two of optimal. This is because only a single additional step will be introduced by each processor. The problem of redundant work could be completely alleviated through inter-processor communication, but this would inevitably introduce additional overhead. An important issue in any distributed computation is load balancing. In our case, this refers to the actual number of connections each processor is required to make. Recall that the input host list may also contain some hosts that have recently left the network. Therefore, even though each processor will receive an equal number of potential hosts to connect to, the number of actual live hosts in the list is likely to be smaller and will vary between processors. However our empirical results indicate this was not a problem. Intuitively this could also be explained because, even though the actual number of connections made by each processor could vary, they are still handled simultaneously by each processor in a single step.

Implementation of Our Gnutella Crawler

We have implemented the distributed algorithm presented in the previous section in Java using remote method invocation (RMI). RMI is Java's version of remote procedure calls (RPC). It is distributed as a standard Java library providing a mechanism for distributed object communication. We chose Java for its portability across platforms because it allows our crawler to be deployed on a heterogeneous network of workstations. The portion of our crawler that implements Gnutella's application-level protocol is based on furi [Fu00] - an open-source Gnutella client written by William Wong. In our implementation, crawling a subset of the Gnutella network is provided as a service residing on various remote locations throughout the network. Using RMI terminology, our algorithm described in the previous section is implemented as a distributed object residing on remote machines. In addition our distributed computing system includes an object serving as the "brain" of the entire computation. This central object is responsible for bootstrapping the entire topology discovery by distributing the initial list of Gnutella hosts to other remote objects. Upon receiving the input, each remote object performs topology discovery of its portion of the network and subsequently returns a graph object representing their (local) network topology. The central object is responsible for merging all the output graphs into a single one representing topology of the entire Gnutella network. The main limitation of our crawler is related to the notion of private networks. Since a significant portion of Gnutella users reside behind a firewall that prevents anyone from establishing direct connection to them, our crawler will not be able to accurately discover topology between such hosts. Notice that these hosts will still appear in the final topology graph, through their connections with hosts outside the firewall. In this sense, the topology obtained by our crawler can be viewed as a subgraph of the actual Gnutella network topology. In addition, even though running time of our algorithm is optimal for any topology discovery algorithm based on the Gnutella protocol, the actual execution time is still bounded by the RTT time of messages in the Gnutella network and can take up to several minutes. One could therefore argue the integrity of our topology data, based on the fact that the network structure has significantly changed over the course of several minutes. Despite these limitations, we believe the topology obtained by our crawler still captures important properties of the actual Gnutella network. These properties are discussed in the following section of our report.

Results

Using our crawler, we obtained several instances of the Gnutella network topology between November 13 and December 28, 2000. Several images of the topology graph from December 27 at around 7PM are shown right below. This particular topology instance consists of 1026 nodes and 3752 edges. The diameter of the graph is 8. The visualizations were done using Otter - a network visualization tool developed by Caida, and LEDA's graph drawing software.



Gnutella As A Small-World

To show that the Gnutella network topology exhibits strong small-world characteristics, we calculated the clustering coefficient and the characteristic path length for several topology instances obtained during the months of November and December of 2000. The values for each one are benchmarked against a random graph G(n, p) and the 2D mesh of the same size (in terms of the number of nodes) as the original topology graph. For random graphs, average values out of 100 trials are shown.

Table 2: Clustering coefficient *C*

	Count source vertex			**Do not count source vertex**		
	Gnutella	**G(n,p)**	**2D mesh**	**Gnutella**	**G(n,p)**	**2D mesh**
11/13/2000	0.643587	0.389914	0.413181	0.035122	0.007789	0
11/16/2000	0.701287	0.492788	0.41276	0.010896	0.005636	0
12/20/2000	0.539189	0.268877	0.412366	0.065172	0.009371	0
12/27/2000	0.514996	0.278801	0.41276	0.063023	0.010213	0
12/28/2000	0.521659	0.27966	0.411995	0.054443	0.009013	0

Because it is not clear from their definition whether Watts and Strogatz consider each vertex to be a neighbor of itself, we have calculated the results using both methods. Based on the results in table 2, we believe the two were not counting the source vertex. However the results obtained on a 2D mesh, typically regarded as a highly clustered topology, highlight a potential inconsistency with this definition.

Table 3: Characteristic path length *L*

	Gnutella	**G(n,p)**	**2D mesh**
11/13/2000	3.72299	4.48727	20.6667
11/16/2000	4.42593	5.5372	21.3333
12/20/2000	3.3065	3.6649	22
12/27/2000	3.30361	3.70995	21.3333
12/28/2000	3.32817	3.7688	22.6667

As you can see, all of the Gnutella topology instances show the small-world phenomenon: characteristic path length is comparable to that of a random graph while the clustering coefficient is considerably higher. The results presented in table 2 and 3 clearly indicate strong small-world properties of the Gnutella network topology. It is our thesis that this is an important issue to consider when designing distributed algorithms, such as those for routing and searching. For example, Gnutella's current broadcast routing strategy is clearly not likely to work well on a clustered topology of a small-world network, as it would generate large amounts of duplicate messages. This would result in poor utilization of network bandwidth and hinder scaling - a phenomenon recently observed in practice.

Power Laws in Gnutella

Upon further analysis of the obtained topology data, we discovered a power-law distribution of node degrees. Figure 1 shows node degree distribution for the Gnutella topology obtained on December 28:

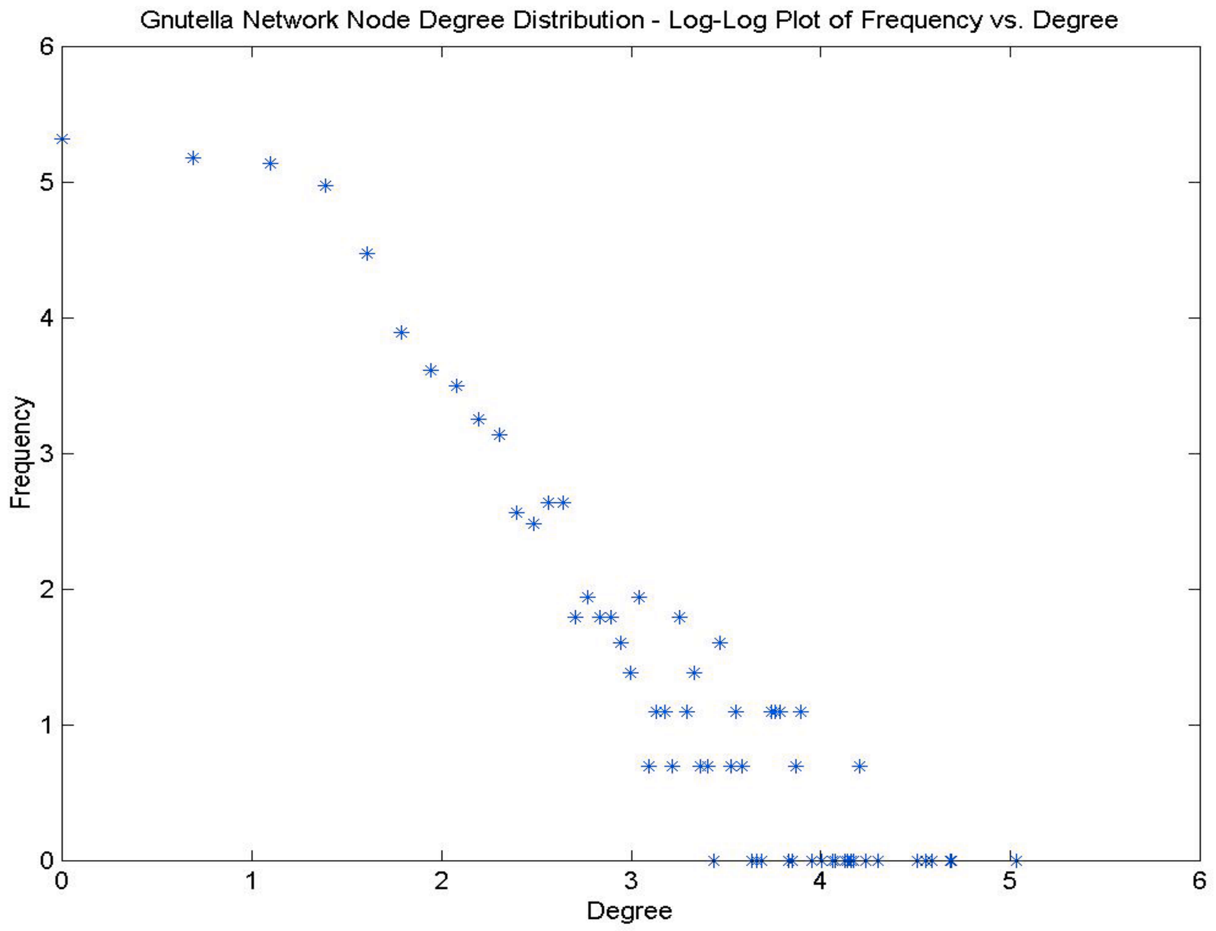


Figure 1: Power-law degree distribution of the Gnutella network topology

As you can see, on a log-log scale data appears to be very linear. Using the least-squares method, we calculated power law exponent to be -1.4. Interestingly, this value was consistent for all the Gnutella network crawls performed in December of 2000, in correlation with previous results discussed above. Linear correlation coefficient for this crawl was 94

Conclusion and future work

We presented a case study of Gnutella - a model of a pure peer-to-peer system. Upon analyzing the Gnutella network topology, we observed strong small-world characteristics and a power-law distribution of node degrees. We hope these results will prove useful to developers of other similar peer-to-peer network applications is designing future algorithms that take advantage of those properties. In the future, we hope to use these results as a guide in formulating a strict mathematical model of the system. We also plan to continue work in the following directions:

1. Verify the existence of power-laws and small-world properties in other peer-to-peer networks.
2. Experiment with new graph layout algorithms for visualizing topologies of peer-to-peer networks.
3. Apply graph theoretic methods to topologies of peer-to-peer networks in search for scalable algorithmic solutions.

Bibliography

Fa99
: M. Faloutsos, P. Faloutsos, and C. Faloutsos. On Power-Law Relationships of the Internet Topology. SIGCOMM 1999.

Cl00
: Ian Clarke, Oskar Sandberg, Brandon Wiley, Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. July 2000.

Wa98
: Duncan J. Watts, Steven H. Strogatz. Collective dynamics of small-world networks. Nature, VOL 393, pp 440-442, June 1998.

Wa98
: Jon Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. Cornell Computer Science Technical Report 99-1776, October 1999.

Br00
: Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, Janet Wiener. Graph structures in the web. 2000.

Kl00
: Jon M. Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, Andrew Tomkins. The Web a a graph: measurements, models, and methods.

Gn00
: The Gnutella Homepage. http://gnutella.wego.com/.

Na00
: The Napster Homepage. http://www.napster.com/.

Fu00
: The Furi Homepage. http://www.jps.net/williamw/furi/.

Fr00
: The FreeNet Homepage. http://www.freenet.sourceforge.net/.

Mihajlo Jovanovic 2001-01-21