

**Implementing Interest-Based Shortcuts over Gnutella:  
An Investigation into P2P algorithms**

**CSC 569**

**1 December 2003**

Dave Gridley

Ryan Retting

Scott Thomas

Brett Tsudama

## ABSTRACT

The popularity of peer-to-peer (P2P) multimedia file sharing applications such as Gnutella and Napster have created a flurry of research into this area. Many novel improvements to these distributed algorithms have been proposed, and studies have been done to characterize the complex networks that form out of this P2P paradigm. In this paper, we will present our findings regarding current P2P algorithm research. In addition, we will also detail our implementation of a P2P algorithm improvement called *interest-based shortcuts*.

## I. INTRODUCTION

In early 1999, the peer-to-peer (P2P) phenomenon was “born” with the introduction of Napster [1]. Napster was developed by an undergraduate student at Northeastern University named Shawn Fanning. He envisioned a system where users would be able to share MP3-encoded music files with each other easily and efficiently. The popularity of Napster exploded, and at one point it attracted over 30 million users with over 800,000 of them accessing the network simultaneously [1]. Unfortunately, a number of legal restrictions on the distribution of copyrighted material eventually led to Napster’s downfall in July of 2001, but the internet community was now acutely aware of the power of the P2P paradigm.

Since then, a number of P2P architectures have been developed and deployed on the internet, thanks to the keen desire of internet users to distribute, share, and obtain multimedia content. Gnutella [2], Freenet [3], BitTorrent [4], and Kazaa [7] are all examples of content distribution schemes that take advantage of the P2P paradigm.

None of these P2P file sharing applications are perfect, and naturally a great deal of research has been done to try and improve them. Modifications to many of the protocols have been proposed, and there has even been in-depth studies on the characteristics of the user population within a P2P system [5]. We have researched a number of enhancements to the P2P paradigm and have implemented the Gnutella protocol along with one of these improvements called *interest-based shortcuts* [6]. Simulation of our improved Gnutella protocol shows that interest-based shortcuts can have a positive effect on performance within a Gnutella network.

The paper is organized as follows. Section II gives an overview of P2P architectures and applications in general. Section III talks specifically about the Gnutella protocol. Section IV presents some proposed improvements to a P2P system, and section V details the improvement (interest-based shortcuts) that we have implemented. Section VI is a discussion of our implementation and results. Finally, group contributions are specified in section VII and the paper concludes in section VIII.

## II. WHAT IS PEER-TO-PEER NETWORKING?

The exact definition of “peer-to-peer” is debatable, but it generally refers to a system that lacks a dedicated, centralized infrastructure. In a P2P system, all parties have the same communication capabilities and all parties can initiate a session with any other party. The success of a P2P system depends on the voluntary participation of peers to contribute resources (content and/or bandwidth) to make the system more powerful (increased content) or to improve the infrastructure (bandwidth). Membership in a P2P system is ad-hoc and dynamic: as such, the challenge of such systems is to figure out a mechanism and architecture for organizing the peers in a way that they can cooperate to provide a useful service to the community of users [5].

There are two distinct models a P2P system can adhere to: centralized and decentralized. The centralized model (used by Napster), maintains a centralized control over the content distribution network. The central server(s) maintain a database of all available content and references to the users that host the content. Upon querying the central server(s), a user can then establish a direct connection to the appropriate user which is hosting the desired content.

The decentralized model (used by Gnutella) is one where every user on the network is a “servent” (acts as both SERVER and cliENT). Each user only knows about a small number of other users (“neighbors”). When a user begins a search, he queries his immediate neighbors. These neighbors will then forward the query to their neighbors, who will do the same. This will continue recursively until some reasonable limit, usually implemented using the time-to-live (TTL) field in a query packet. If at any time a user matches a query, he will notify the user that originated the query. In this way, a direct connection between the querying user and the query-match user can be established and a file transfer can commence.

Obviously, there are advantages and disadvantages to both models. The big advantage to the centralized model is its simplicity. Assuming the central system can support the load, users can quickly and efficiently find the desired files. In contrast, searching and locating is much more complicated in a distributed model. Also, in the centralized model, users are guaranteed to find a file if another user is hosting it (since all users must register with the central system). This guarantee is infeasible in a very large distributed system due to bandwidth and computing resource constraints.

By the same token, there are disadvantages to the centralized model along with corresponding advantages in the decentralized model. In a centralized system there is a single point of failure at the central server(s), while a decentralized system is much more robust to link and node failures. Also, a centralized system can provide less privacy and anonymity than a decentralized system, an important consideration for many P2P users.

### III. WHAT IS GNUTELLA?

Gnutella is a decentralized P2P file-sharing model that was developed in early 2000 by a company called Nullsoft [8]. It was only available to the public for a very brief time before development was halted. However, the protocol was able to be reverse engineered from the source code that was temporarily available [12]. It is important to note that Gnutella is simply a protocol, not an application. Many individuals and companies have developed clients which support the Gnutella protocol, the most popular being Morpheus [9], LimeWire [10], and BearShare [11]. Since all of these clients use the same protocol to communicate, all Gnutella clients are interoperable. Typically, users will refer to the entire network of different clients as the “Gnutella network”.

To share files on the Gnutella network, a user (A) starts with one of the Gnutella clients and connects to another computer (B) with Gnutella enabled. A will announce its existence to B, which then announces A’s existence to all of its (B) neighbors. This will continue until some reasonable threshold is reached, at which point A will be able to send out queries to find peers who have some desired content.

There are four basic messages that are passed throughout the Gnutella network. A *Ping* message is used to actively discover hosts (such as when A joins the network in the above example). A *Pong* message is a reply to the *Ping* message, and contains addressing and available content information. A *Query* message is sent out with desired file information and is used to search for files in the network. A *QueryHit* message is sent out by a user when he matches a *Query* message and contains information so that the querying user can obtain the desired file.

It is important to note that all response messages (*Pong* and *QueryHit*) must travel down the same path the carried the matching request-for-response message (*Ping* and *Query*). This has implications for our later discussion on improvements to the Gnutella protocol. Also, the protocol does not specify the actual file transfer protocol. Instead, users use the HTTP protocol for point-to-point file transfers.

### IV. IMPROVEMENTS TO THE P2P ARCHITECTURE

#### **P2P Assumptions**

Before discussing improvements to a P2P system, we must first understand the problems and limitations inherent in a current P2P system. An in-depth study was done on the user populations in the Gnutella and Napster networks in [5], and the results were quite interesting. It proved that many implicit assumptions made by P2P system designers were false. One of these assumptions was that all nodes in the system will participate and contribute equally to information exchange and routing; therefore, the delegation of responsibility across nodes should be uniform. The study showed that the set of users participating in a P2P network are very heterogeneous with respect to a number of important characteristics such as internet connection speeds, latencies, lifetimes, and amount of shared data. Some of the differences between these characteristics were

between three and five orders of magnitude across peers [5]! As a result, a “good” P2P system should delegate different degrees of responsibility to different hosts, based on some set of host characteristics (trust, reliability, speed, etc).

Another frequent implicit assumption in P2P systems is that peers are willing to cooperate. Users are usually asked to report their internet connection speed when configuring their client, and the study found that as high as 40% of all users either deliberately misreport their bandwidth or leave it undefined. When there is a benefit to misreport characteristics, users will do so. Instead of relying on reported characteristics, a P2P system should attempt to directly measure the characteristics of peers in the system.

Yet another myth in P2P file-sharing systems is that all peers will behave equally in terms of consuming and contributing resources. The study clearly showed this assumption to be false, as server-like and client-like behavior was evident in the population. It was found that approximately 26% of Gnutella users and 20-40% of Napster users share little to no files, indicating that these users are freeloading and consuming resources. A P2P system should take this fact into consideration and encourage users to share data, perhaps through some sort of reward or rating mechanism.

By thoroughly understanding and characterizing the user populations of common P2P systems, we can make better design decisions when developing a new P2P system or when proposing improvements to a current P2P system. The following subsections will detail some of these proposed improvements to the P2P paradigm.

### **User Registration and User Rating System [12]**

Download failures can significantly degrade the quality of service provided by the network. Users expect to obtain files with a certain level of reliability. In [12], the authors used Gnutella and did 100 random searches with a TTL value of 10 based on some frequently used keywords of non-adult content. They found that over 95% of searches failed to provide results that exhibited:

- Reliability (would not become inaccessible during download)
- Appropriate downloading speed (over 30 KB/sec)
- Complete data (an entire song, file, etc)
- Easily comprehensible results (results returned have consistent naming schemes, etc)

Based on these searches, the authors conclude that the two problems that degrade the network’s effectiveness are download failures and the lack of cooperation (content sharing) between peers. In order to resolve these problems, the authors propose a user registration and rating scheme.

Basically, this scheme requires all users to register with the system, providing information such as username, password, geographical location, and email address. Each user is then given a unique ID which is utilized when performing queries such that other

users can obtain more detailed information. This is a novel idea, but since Gnutella follows the decentralized model, there is no central server where this information can be stored. Instead, the authors propose a system whereby a number of registration servers are distributed throughout the network. These servers run a distributed database that contains all the user registration information. Synchronization of the data is done overnight when network usage is low. Load balancing becomes a problem, as does server synchronization.

The user rating system attempts to resolve the lack of cooperation between peers. By implementing a rating system, users would be encouraged to contribute and share files instead of freeloading. It would also be used to allocate bandwidth when multiple peers are competing to download from a single host.

Both user registration and user rating are good ideas, but this paper does not detail any implementation details. A similar system has been implemented on eBay with good results, but the applicability of this scheme to a P2P system has not yet been completely evaluated.

### **Gnutella Private Networks**

The authors of [15] also propose another Gnutella improvement called Gnutella private networks. This would require the use of special registration servers which would be responsible for a certain sub-network within the Gnutella network. This sub-network could be a file-sharing network for a company or an academic institution. In this way, these organizations could utilize the Gnutella network for their private file-sharing needs.

In order to implement this, a modified version of the above registration scheme would be utilized along with high-level encryption to ensure data security. While the authors do not present any implementation details, this is a promising improvement to the Gnutella network.

### **Chord [13]**

A fundamental problem with P2P systems is the efficient location of a node that stores the desired data item. Gnutella handles this with simple queries, but this can be disorganized and inefficient. Chord presents a scalable lookup system that can be used in the highly dynamic and decentralized environment of a P2P system.

Chord is not an improvement to Gnutella, rather, it is a protocol that can be used for speedy lookups in a P2P system. Chord is utilized when building a P2P application, much as Gnutella is used as the base protocol for file-sharing applications. Chord does one thing: it maps keys to nodes. In this way, nodes are given responsibility to host the certain key, and a mechanism is defined which allows for easy searching and retrieval of keys.

This scheme can be applied to a Gnutella-like network quite easily. If we map keys to individual files, then Chord can be utilized to distribute the keys uniformly throughout the network. The keys contain routing information which can allow a user to find the

corresponding file within the network. Chord gives us the ability to efficiently search through the keys, giving us improved performance compared to the simple query method of Gnutella. For further details on Chord, including the actual algorithm and simulation results, see [13].

Another similar consistent hashing scheme is proposed in [15], where the authors propose a system which builds an optimal topology using a combination of hashing and indexing. Nodes are responsible for storing key information for all content within a certain zone, thereby allowing a querying node to quickly find content by searching a small number of zones. This is an improvement over the original Gnutella query mechanism via flooding. For a detailed explanation, see [15].

### **Topology Management**

Building P2P networks with desirable topological properties is a fundamental problem in P2P computing. If the network is fragmented or has a high diameter (nodes are separated by many hops), flooding based queries are ineffectual since many queries will not reach a large set of hosts. The authors of [16] present a simple heuristic that servers can follow when joining and leaving a Gnutella network. Using stochastic analysis, the authors show that if all servers follow the same heuristic, certain guarantees can be made regarding the network diameter and other desirable properties.

The details of the heuristic and implementation are not included here, but one thing of note is the use of a single host server which maintains a cache of containing the list of all nodes in the system. The authors show that the load on the server can be minimal (and therefore feasible in a huge network such as Gnutella), but this still represents a single point of failure. Guaranteeing the availability of a single server introduces more distributed computing issues, and the authors do not discuss how this would be accomplished.

### **Query Expansion [14]**

Search algorithms in P2P systems are incredibly inefficient when compared to information retrieval (IR) algorithms. This is because it is impossible for hosts in a P2P system to have useful global knowledge such as popularity of data items and relationships between keywords and data items. P2P systems use naïve text-match searching, where data items are only returned when they exactly match the keyword in the query.

An IR technique that has been used for decades is query expansion. Query expansion means adding relevant terms to the original query. The purpose of query expansion is to cope with the mismatch between the term used by the searcher and the desired content. Think of the function of a thesaurus and you have the basic concept of query expansion.

The authors of [14] propose a mechanism whereby query expansion is implemented in a P2P system using keyword relationships and the distributed mechanisms of updating a thesaurus. A thesaurus Keyword Relation Database (KRDB) is kept at every node, and

keyword relationships are kept for data items at the node itself. This enables semantic searches within a P2P system.

The authors have implemented a prototype system by modifying an open-source client to the Gnutella protocol, LimeWire. Results were not provided, but it is expected that query expansion will increase the effectiveness of queries within a Gnutella network.

## V. INTEREST-BASED SHORTCUTS

Gnutella is a file sharing protocol that is based off of flooding queries to all peers. This simple approach allows Gnutella to be easily understandable and robust, but a problem with flooding is that it is often not scalable. The simplistic nature of Gnutella, however, is fundamental to its success. Here we will discuss the desired scaling improvements without changing the overall feel of Gnutella. The basis behind our improvement is a method called interest-based locality which makes the assumption that if a certain peer has a piece of content one is interested in than they are more likely to have other items of interest as well. This simple algorithm can greatly reduce the amount of flooding and make Gnutella more efficient.

The design philosophy behind this work is that peers that share common interests create shortcuts to each other. Peers will use these shortcuts to locate content. In the case where interest-based shortcuts lack the desired content the algorithm resorts to the underlying Gnutella framework. These shortcuts are more of a loose performance enhancing structure that sits on top of Gnutella's simple, robust, and unstructured overlay. One of the more valuable aspects of this approach is that it is not necessarily an integral part of Gnutella, but instead can be applied to many other peer-to-peer content location paradigms making it very extensible. The other aspect of interest-based shortcuts that makes them so attractive is their simple design relative to the resulting improvements in content location and scalability.

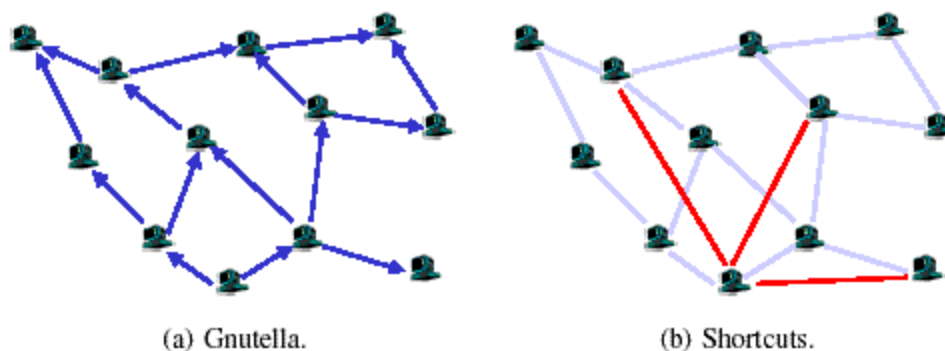


Figure 1

Shortcuts do not change the underlying architecture at all, instead, they serve as performance-enhancing hints. To better understand the improvements made by interest-based shortcuts we can take a look at two figures. Figure 1 (a) shows how the underlying Gnutella search for content. Figure 1 (b) depicts the interest-based method where we see the requesting node searched only its shortcuts for the desired content rather than

flooding everything. To accomplish improvements on a large scale scalable peer-to-peer system the requirements are that the algorithm be adaptive and self-improving. To meet both of these we can investigate the two parts of shortcut discovery and shortcut selection.

For discovery our first attempt is done through the normal means of flooding. The returned peers having the desired content are all potential candidates for the shortcut list. The flexibility here is that peers can passively monitor their own responses and decide on who they want in their own personal shortcuts. We will look into this more in shortcut selection below. Shortcuts are based off of their perceived utility to the individual peer. After each new search the list can be updated to have the peers with the highest utility in the list. There are even propositions for discovery where shortcuts share their lists out of band with other peers. The overall goal for discovery is to always use a shortcut and not have to resort to flooding.

For shortcut selection there are again different approaches to fine tune the shortcut list. The flexibility of this approach allows for peers to create their lists using algorithms concentrating on different metrics. Rankings and metrics can include; probability of providing content, latency of the path to shortcut, available bandwidth of path, amount of content at the shortcut, and load of the shortcut. All of these metrics have minute differences based off of the desired content and relative size of the network. Again we are concentrating on the highest utility for the individual peer. As one might assume there are different algorithms for different scaled networks and content.

In this research paper the experimental results showed that having the interest-based layer above the Gnutella groundwork greatly optimized performance. The benefits and overhead of the overlay were measured based on a number of factors: success rate (how often shortcuts were utilized or used), load characteristics (is the load on the network reduced), query scope (what fraction of the peers were involved), minimum reply path lengths (how long does it take for replies), and additional state (how much overhead is involved in the additional state.) The load at each node was cut to a quarter of that with the normal flooding approach. The number of hops to locate content on a normal Gnutella network was somewhere around 4 hops where as with the interest-based shortcuts approach the shortest path was reduced to around 1.5 hops. This is just with the generalized approach of adding the shortcut layer. There are many specialized algorithms and search methodologies that can be applied when specific uses are known. Distributed web pages or web objects have interesting properties in that the content may or may not need the latest update from the publisher. Compare this type of system to, say, a file sharing application where the content is mirrored across multiple sites with no need to check the source for updates or changes.

## **VI. IMPLEMENTATION AND RESULTS**

In our implementation we took the above idea of interest-based shortcuts and tried out query hits and a few of our own ideas for the performance metrics. There were several complicated metrics mentioned in the paper that would rank and select different shortcuts

based off of their utility. Our goals were to still improve on the number of hops necessary for content and also to reduce the amount of flooded queries on the network. The difference for us was the size of our test bed network. For our demo we were limited to about 12 machines and wanted to concentrate on only simple file sharing so the need for additional metrics was not needed and would have appeared uninteresting. We decided to show how even the one metric approach could reveal great improvements over standardized Gnutella.

To start off we needed to develop the Gnutella protocol and have it fully functional. This had several unforeseen difficulties in and of itself before even getting to the interest-based improvements. The basic nature of Gnutella is that it will flood to everyone in the network. In our lab situation flooding like this would have resulted in everyone being exactly one hop from each other. This of course is uninteresting and not realistic of real world peer-to-peer systems. To combat this situation we created a Host Cache Server which keeps track of peers as they sign on. Each new peer is given the last two peers as neighbors creating a topology similar to Figure 2. With the Host Cache Server creating this topology we can simulate a multi-hop network and really show the differences between normal flooding and the improvements that an interest-based shortcut overlay will allow.

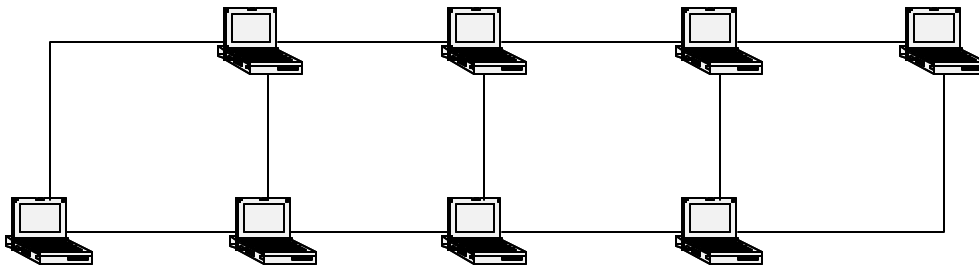


Figure 2:

Our implementation was done in java and had a GUI that would display all of the shared files as well as neighbors, query results and other interesting peer-to-peer information. In Figure 3 we can see the connection interface. The connect button will communicate with the Host Cache Server which returns peers to populate the Neighbors table. In Figure 4 we have the main text based search which at first will flood the network looking for the desired files. The returned query hits populate the Search Results table which in turn would allow a user to download any desired file. The results are also added to the Shortcuts table. For demonstration of the Shortcuts we put a check box that would allow users to either use them or simply stick with the underlying Gnutella flooding.

Our methodology with the shortcuts is that they should not be too stagnant not measured on more than their usefulness. We populate our shortcuts with all of the returned query hits. For future searches we go through the Shortcuts first and see if they return any of our additionally desired files. If they fail to we fall through to the underlying Gnutella method of flooding and purge our Shortcuts. The next set of returned results from the flood now becomes our new Shortcuts.



Figure 3

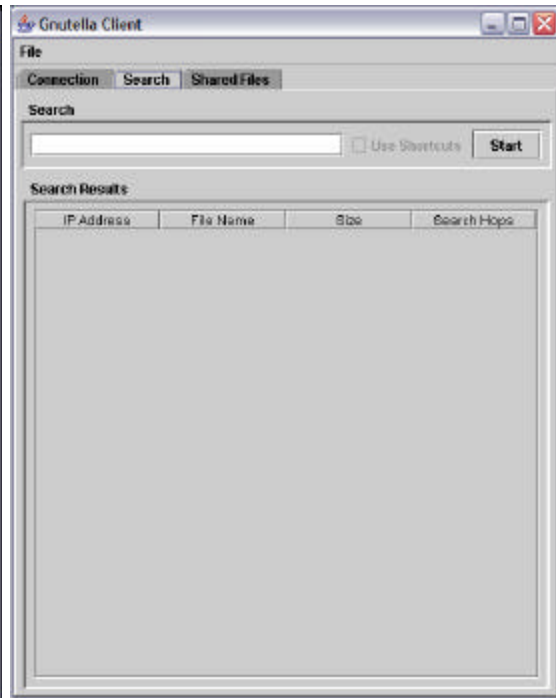


Figure 4

To demonstrate if the Interest-based shortcut method was successful or not we displayed the number of hops used for the search. In the case where flooding was needed the results were often up to three. In our implementation we set the TTL to be three so that the flooding would not reach beyond an acceptable level for our small test bed. When the Shortcuts were used and successful the hop count was always 1 being that there was a stored direct connection. Due to this fact we set the TTL of all shortcuts to 1. The reduced hop count helps with both the latency of queries and also reducing unnecessary network traffic.

## VII. GROUP CONTRIBUTIONS

All group members contributed equally to this project. Scott was the team leader and handled the tricky parts of the implementation. Brett focused on the development of the project idea, much of the research findings, and the organization of the write-up. Dave was instrumental in our in-depth study into interest-based shortcuts, and Ryan handled most of the GUI and the protocol implementation.

All team members were responsible for researching at least one paper that proposed an improvement to the Gnutella protocol. Papers that did not deal specifically with Gnutella but with P2P algorithms in general were also acceptable. During group meetings we would report our findings to the entire group, and midway through our project we made the decision to implement interest-based shortcuts.

At this point, implementation responsibilities were split between team members, with Scott handling overall integration, Ryan focusing on GUI development, and Brett and Dave working on protocol implementation.

## **VIII. CONCLUSION**

In this paper, we have shown the current state of research in the field of peer-to-peer computing for content sharing. Specifically, proposed improvements to current P2P protocols and architectures were presented and discussed. We have implemented *interest-based shortcuts* over the Gnutella protocol and have shown a marked improvement over the basic Gnutella query mechanisms. To conclude, *Interest-based shortcuts* are a relatively simple improvement that can be applied to any P2P protocol with promising results.

## REFERENCES

- [1] Napster, <http://www.napster.com>
- [2] Gnutella, <http://www.gnutella.com>
- [3] Freenet, <http://freenet.sourceforge.net>
- [4] BitTorrent, <http://bitconjurer.org/BitTorrent/>
- [5] S. Saroiu, K.P. Gummadi, S.D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. of Multimedia Computing and Networking 2002 (MMCN '02)*.
- [6] K. Sripanidkulchai, B. Maggs, H. Zhang, "Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems," *IEEE INFOCOM 2003*.
- [7] Kazaa, <http://www.kazaa.com>
- [8] Nullsoft, <http://www.nullsoft.com>
- [9] Morpheus, <http://www.morpheus.com>
- [10] LimeWire, <http://www.limewire.com>
- [11] BearShare, <http://www.bearshare.com>
- [12] I. Ivkovic, "Improving Gnutella Protocol: Protocol Analysis and Research Proposals," *Prize-Winning Paper For LimeWire Gnutella Research Contest*, September 2001.
- [13] I. Stoica et al, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Transactions on Networking*, Vol. 11, No. 1, February 2003.
- [14] K. Nakauchi et. al., "Peer-to-Peer Keyword Search Using Keyword Relationship," in *Proc. GP2PC 2003*, Tokyo, Japan, May 2003.
- [15] J. Considine and T. Florio, "Scalable Peer-to-Peer Indexing with Constant State," *Boston University Tech Report 2002-026*, September 2002.
- [16] G. Pandurangan et. al., "Building P2P Networks with Good Topological Properties," 2001.