Architecture and Optimization for a Peer-to-Peer Content Management System

by

Dion M. Edge

MBA
MIT Sloan School of Management, 2003

B.S., Systems Engineering
United States Military Academy, 1996

SUBMITTED TO THE DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN CIVIL AND ENVIRONMENTAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

FEBRUARY 2004

Signature of Author: _______

Department of Civil and Environmental Engineering
January 15, 2004

Certified by: ____________________

John R. Williams
Associate Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by: ______

Heidi Nepf
Associate Professor of Civil and Environmental Engineering
Chairman, Department Committee on Graduate Students

Architecture and Optimization for a Peer-to-Peer Content Management System

by

Dion M. Edge

Submitted to the Department of Civil and Environmental Engineering
on January 15, 2004 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Civil and Environmental Engineering

ABSTRACT

This thesis will explore the design and optimization of a peer-to-peer network application as a solution to complex content management problems. Currently, most content management systems are expensive, cumbersome and inflexible custom solutions that require knowledge workers to change their work habits. Peer-to-peer offers a uniquely decentralized and, potentially, scalable solution for knowledge workers by providing a simple and visual tool for file management, meta-data description and collaboration. This thesis will reference a client beta designed and developed by the author.

Additionally, this thesis will address the need for content management solutions, the state of current solutions and a requirements document for a solution. Subsequently, the thesis will explore the design aspects of a peer-to-peer content management solution. As well as designing and developing a P2P client as proof of concept, this thesis will mathematically explore the implications of scaling the client to many users and methods to optimize performance. The last few chapters will cover the implementation of the client, proposed next steps for development and analysis of alternative architectures.

Thesis Supervisor: John R. Williams

Title: Associate Professor of Civil and Environmental Engineering

Acknowledgements

Sincerest appreciation is extended to Dr. John R. Williams, the author's thesis advisor and major engineering professor, for his guidance and encouragement in this endeavor. The author is also grateful to Dr. Williams for providing guidance on network optimization techniques and validating the network optimization model. Special gratitude is due to Neeraj Jain, the author's partner in developing the BrainFist client, for sharing a vision of an open source content management solution. The author is deeply indebted to Brooke, wife; Isabella, daughter; and Atticus, son; all of whom have exhibited infinite patience, understanding and support of the author's dedication to this thesis and entire graduate matriculation. Finally, recognition is due to Rodney and Fran Edge, the author's parents, for encouragement to stay and finish the fight.

Contents

Chapter 1

Introduction

This thesis originally started out with a simple question, "How can a grad student at MIT manage the countless digital documents accumulated over the past two years in order to make use of said documents in the foreseeable future?" or, more generally, "How can knowledge workers effectively manage and use the ever increasing supply of information?" As a grad student, the author is primarily concerned with static documents such as .doc, .pdf, .ppt, .ps, .xls, .dbm, etc. – and not with e-mails, web pages or web logs. This decision is made for pragmatic reasons and to narrow the scope of the investigation. Consequently, the purpose of this investigation is to analyze the viability of using Peer-to-Peer technology (P2P) to address the subset of content management dealing with document management through metadata.

This requires an understanding of Knowledge Management (KM) systems, Document Databases (DD), Collaboration Applications (CA) and, finally, Content Management (CM) systems. Content Management is the newest and best defined term in the information technology lexicon to address the problems facing knowledge workers. This thesis considers KM solutions, web-based archives, complex proprietary database solutions and, more pointedly, Peer-to-Peer technology. The thesis concludes that it is necessary to pursue a solution tailored to the user's problems instead of an immediately profitable enterprise solution. Consequently, open source platforms are an attractive resource for developing a prototype CM solution. After deciding on the appropriate technology platform, in this case the P2P open-source development environment Gnutella, the author set out to design, build and analyze a working prototype.

1.1 Problem Statement

This thesis deals with the problem of content management. More specifically, the thesis addresses the aspect of document management through metadata. These issues will be explored at greater length in Chapter 3. Additionally, for the sake of perspective, the thesis will consider these content management issues for an organization having between 150 and 50,000 users, using the same type of static documents mentioned above. The author attempts to address these narrowly defined aspects of CM with a P2P solution.

There are many components of CM not addressed in the solution but briefly discussed in Chapter 3, such as Structured Content and Automated Workflow. These issues become critically important when assigning responsibility and authorization for editing, creating and accessing documents. Additionally, the author acknowledges significant issues concerning the security of P2P networks in the Chapter Six. Network security, reputation mechanisms and unique identifiers are addressed in the last chapter, but only briefly.

1.2 Scenario

In order to present the purpose of this thesis, it is necessary to paint a scenario, or case study, of the problem in real terms. Currently, management consultants operate the way many knowledge workers do – solving problems by using previous work experiences (and products), applying proven solutions and presenting the recommendation to clients, often in the form of a

PowerPoint presentation. This involves cataloguing and maintaining scores of documents of various formats (Word and PDF documents, Excel spreadsheets, PowerPoint presentations, Access databases, etc.). Currently, most knowledge workers depend on the standardized File System of Folders and Subfolders. This is an inflexible solution, at best, since documents can only exist in one folder, unless through wasteful replication. Metadata is the most promising way of describing documents, unconstrained by the singular Folder description or the file's physical location. Although metadata is covered extensively in Chapter Three, it deserves attention here to clearly define the problem. Metadata, a file's descriptors, allows for improved searching, cataloguing and reuse of content.

For example, the consultant may have a PDF of a white paper on "Real Options for Product Development in Biotechnology." This paper could exist in several locations such as <Finance>, <Project Management>, <Biotechnology> or <Product Development>. Most likely, however, in the File System methodology, the consultant will only have the file in one folder. A another consultant working on a different project may have use for such a paper, but, even with access to the owner's files, only has a one in four chance of discovering the document in a single search. This is part of the problem in what is commonly known as the 'information silo.' Searching metadata parameters such as Keywords, Subject, Author, etc. will increase the probability of the white paper's reuse.

Metadata, by circumventing the hierarchy of the current File System, avoids the information silo problem and creates transparent methods for searching, cataloguing and reusing content. Unless the consultant has access to an expensive, proprietary Knowledge or Content Management System, documents are lost in plain site. The consultant may have several documents relevant to a project but without a method of classifying and tracking those documents, the relevant content is lost in a sea of information. BrainFist proposes a solution to this specific problem of document management by using metadata to maximize the potential use of content. Furthermore, BrainFist is based on a distributed architecture that promotes the searching and sharing of content among users in an organizational setting.

1.3 Thesis Overview

The thesis will present how the author defines the content management problem, evaluates the P2P landscape of potential solutions and develops a potential solution, complete with feature and scaling requirements. Furthermore, the author will present evidence and proof of concept to the reader which presents the BrainFist client as a file management tool and a viable/scalable P2P solution.

Chapter Two surveys the related work and contributions of others necessary to the creation of the BrainFist client. More than a listing of previous work, the author presents criteria for comparison of different P2P technologies. Chapter Three discusses the problems commonly associated with Content Management needs as well as translated feature requirements. Chapter Four presents the BrainFist design overview as well screen shots of the client itself. In Chapter Five, a network optimization model is presented. Specifically, the model illustrates how the client scales on the Gnutella network for an increasing number of users and network traffic. Chapter Six explores the two most critical issues facing a P2P Content Management System: Security and Roles and Authorization. Chapter Seven presents an analysis of the end product as well as future steps needed to reach a fully utilized solution and a conclusion with salient learning points from this thesis and project.

1.4 Contribution

This thesis presents a P2P client, based on the LimeWire protocol, which specifically addresses a limited number of clearly defined content management issues. BrainFist is the name of the customizable P2P client and will be available for download from an informational web site [1]. BrainFist is a free P2P client that uses the scalable P2P architecture to build a customized network for a given organization, providing an interface for file/document management through the manipulation of metadata. Furthermore, the focus of this thesis is not on computer science but on information technology. This thesis presents an applied solution that uses an emerging technology to address the problems created by other emerging technologies.

Chapter 2

Related Work

Peer-to-Peer technology is a loosely defined term that encompasses several definitions. Although, for the purpose of this paper, P2P refers to the application-level environment, it is necessary to identify what definitions of P2P are not relevant. Specifically, Peer-to-Peer can be generally defined by the following:

- Peer-to-Peer Communications are the logical direct and indirect connections between physical nodes on a given network layer.
- Peer-to-Peer Network is the physical relationship between nodes on a given network.
- Peer-to-Peer Computing defines the end-user application-level environment. This is the relevant definition for the purpose of this thesis.

Furthermore, Intel's David Barkai formally defines P2P computing as "a network-based computing model for applications where computers share resources via direct exchanges between the participating computers." A generalized illustration presents a P2P network where clients connect to each other through the internet to exchange files, communicate with instant messaging or distribute computational functions.

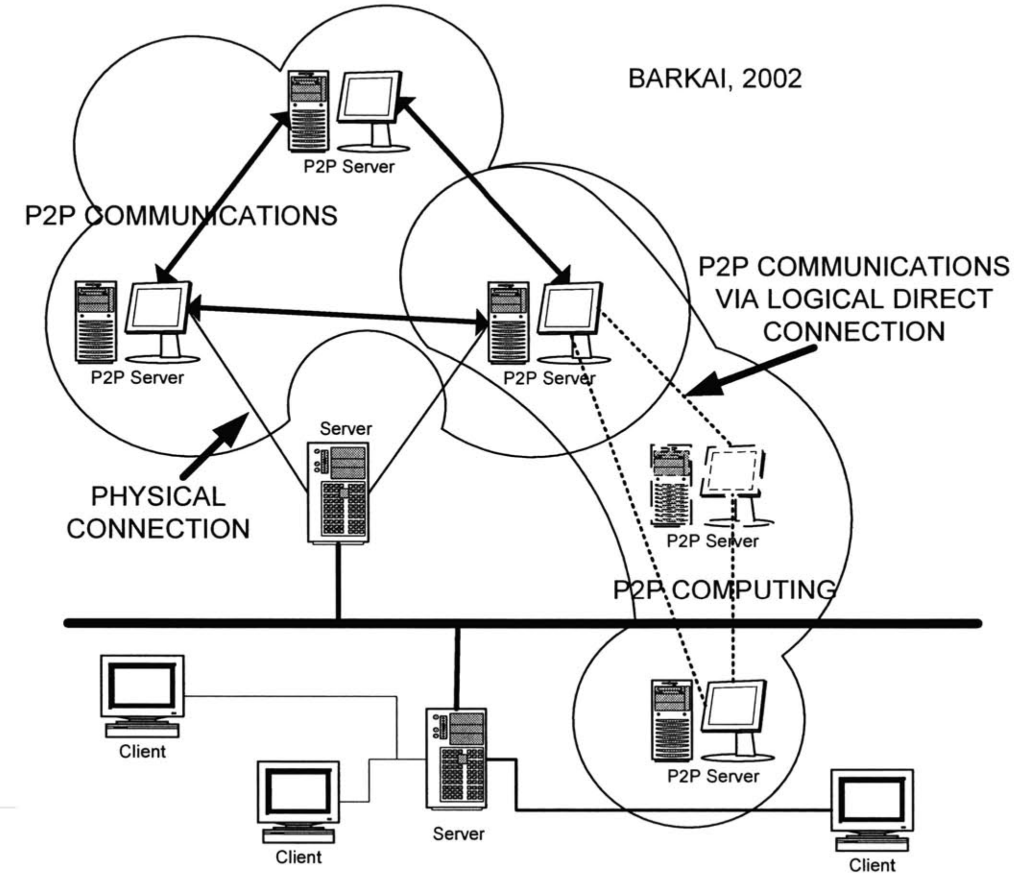


Figure 1

P2P, despite its heterogeneous nature and open source origins, does possess several distinct and recurring characteristics. Through simplification, it is easy to describe P2P computing in terms of advantages and disadvantages. After describing these common characteristics, it is easy to transition to a set of criteria to evaluate the major P2P architectures or topologies. Finally, the author will cite several major P2P applications to illustrate the evaluated architectures.

2.1 Common Characteristics

The primary advantage of most P2P applications is that the application harnesses the computing power (storage capacity, processing power, file archival, etc.) of many nodes. Often P2P applications do not have a central point of failure but through redundancy are extremely resilient to failure (e.g., the Internet does not have a central point of failure and, concurrently, neither do P2P systems). Finally, scalability is a key attribute. Since every node is a peer, it is possible to add more peers to the system and scale to larger networks. The primary disadvantages stems from decentralized coordination. P2P applications struggle with the trade off between global state consistent and redundancy. There is a definitive need for distributed coherency protocols but all nodes are not created equal. This adds a level of complexity in identifying available resources among nodes that often conflicts with the desired anonymity of P2P. One of the most significant problems arising from coordination is identifying the presence of active nodes. Users will drop off and join the network constantly, requiring constant system monitoring and coordination. Finally, the most significant exogenous disadvantage to P2P applications is the intersection of intellectual property, litigation and the 'illegal' behavior of copyright infringement. The Recording Industry Association of America (RIAA) is currently suing its own customers to prevent file swapping of copyrighted music [3].

2.2 Topologies

After identifying the major advantages and disadvantages of the generalized P2P network it is easy to evaluate the major P2P topologies (Centralized, Ring, Hierarchical, Decentralized and Hybrid) by looking at commercial examples.

2.2.1 Centralized – Napster

Napster is the most well known example of a P2P application as well a Centralized network topology. Users search a server or "broker" for the location of an MP3 file, and then download the file directly from the host node. This centralized structure of file management possesses both significant advantages and disadvantages. From the user perspective, Napster is a P2P application since the media files are located and retrieved from another Napster user. From an indexing perspective, as well as a legal standpoint, Napster is a centralized topology with a single reference point. This obviously allows for greater positive control over the network but also lacks redundancy and resistance to network failures, both systemic and intentional. The following diagram is a simple illustration of the Napster architecture [4].

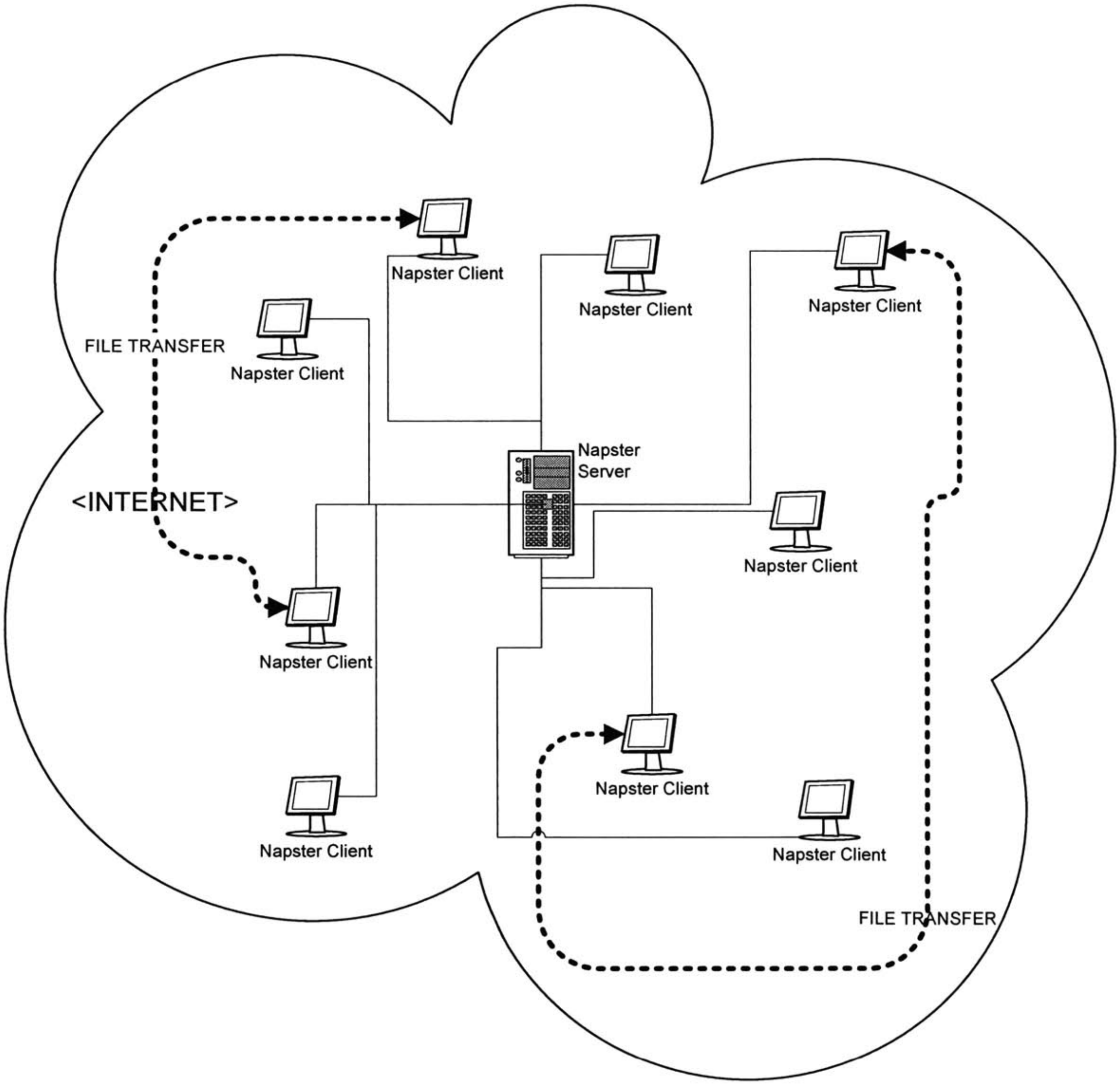


Figure 2

2.2.2 Ring – Chord

The most prominent example of a Ring topology is the MIT native project Chord. Described as a scalable, robust, peer-to-peer service, the Chord distributed lookup protocol, or Chord primitive, addresses the fundamental problem of locating the node that stores a particular data item. Chord does this through its primitive and provides support for a "single, flexible operation: given a key, it maps the key onto a node." The key and node identifiers are chosen from the same 160-bit space and all computation is performed modulo 2160. The Chord distributed hash function calculates the successor of a given key or the node with the smallest identifier greater than the keys. The primitive does this by maintaining a fixed amount of state (logN entries where N is the number of nodes in the system) to enable the success of any key to be determined by exchanging logN messages with other nodes of high probability. The primitive allows for many applications since all data items are matched against keys and storing key/data item pairs becomes a function of the redundant hash. This allows for unreliable nodes to enter and exit the network with little impact on the probabilistic integrity of the system. Communication cost and the state maintained by each node, theoretically, scales logarithmically with the number of nodes.

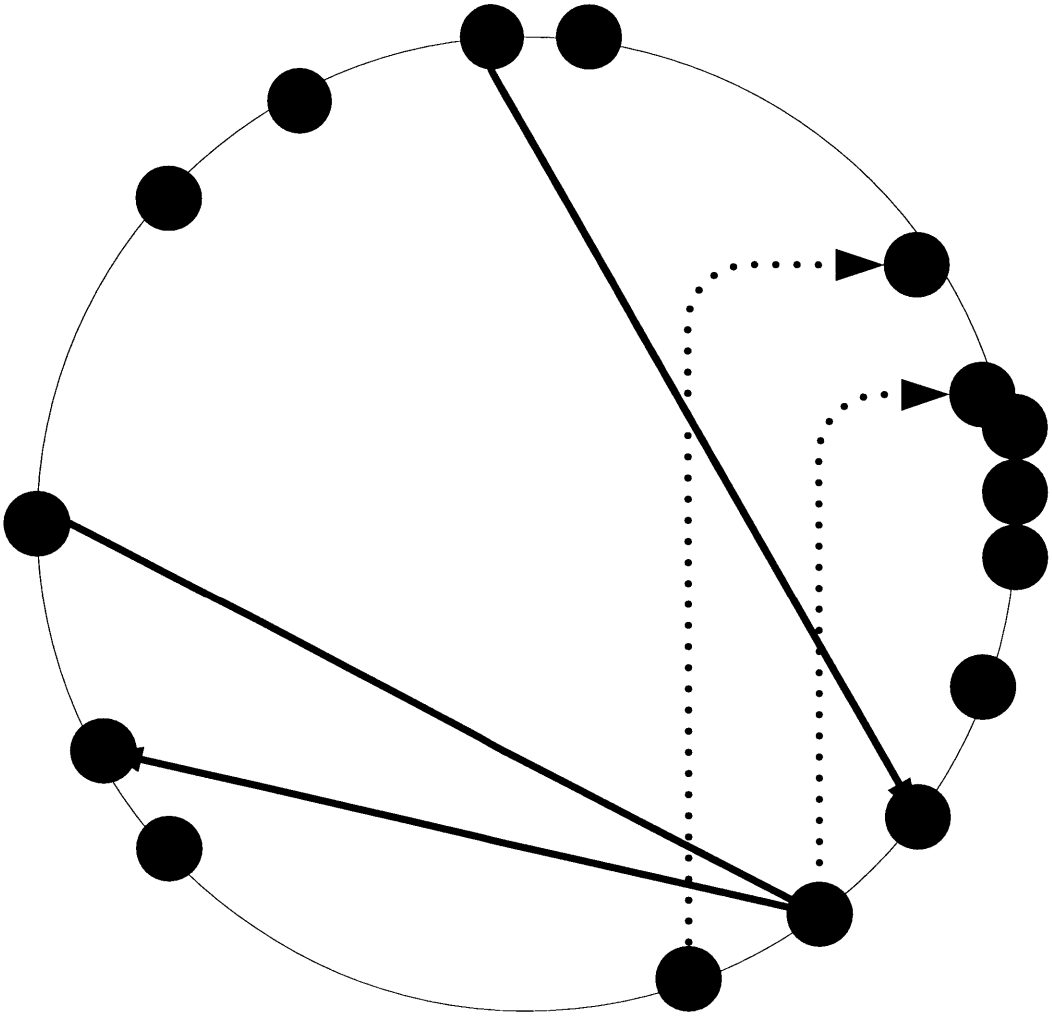


Figure 3

The above figure is taken from the MIT LCS Chord paper and represents a network of 50 nodes. The two arrows represent a single node's routing table entries while the curved lines represent entries to the nodes immediate successors. Additionally, straight lines are finger table entries which allow lookups to be performed in logN time. According to recent LCS papers, this layered architecture has tested with positive and promising results but is still experimental for the purposes of this thesis [5].

2.2.3 Hierarchical – JXTA

The JXTA Project is an open source P2P infrastructure consisting of core protocols for node discovery and messaging; a service layer that facilitates interaction such as searching and sharing between disparate P2P applications; and, finally, the application layer itself. The technologies allow any device connected on the network to communicate and exchange resources in a peer to peer fashion. More relevant, is the Content Management Service (JCMS) proposed by the JXTA Project. The JCMS allows JXTA applications to share and retrieve content within a peer group. This is accomplished through a protocol based on JXTA "pipes" for transferring content between peers where each item of shared content is represented by a unique content identifier (id). Each id is generated from a 128-bit MD5 checksum generated from the data. This creates a favorable environment for searching content by unique MD5 id's instead of arbitrary content descriptors. This increases the probability of locating a 'closer' peer as well as higher quality content. Additionally, each content item has an associated content advertisement which provides meta-information describing the content (name, length, id, etc.). What is quite helpful is that the content advertisement is stored as an XML document. For example, an advertisement may take on the form:

```
<?xml version="1.0">
   <!doctype jxta:contentAdvertisement>
   <jxta:contentAdvertisement>
        <name>index.html</name>
        <cid>md5:1a8baf7ab82c8fee8fe2a2d9e7ecb7a83</cid>
        <type>text/html</type>
        <length>23983</length>
        <description>Web site index</description>
   </jxta:contentAdvertisement>
```

Figure 4

JXTA follows a hierarchical topology because the infrastructure possesses few search hubs containing many id's. The hubs possess higher bandwidth and processing power and help moderate individual peers. Conceptually, the JXTA infrastructure takes on the following:

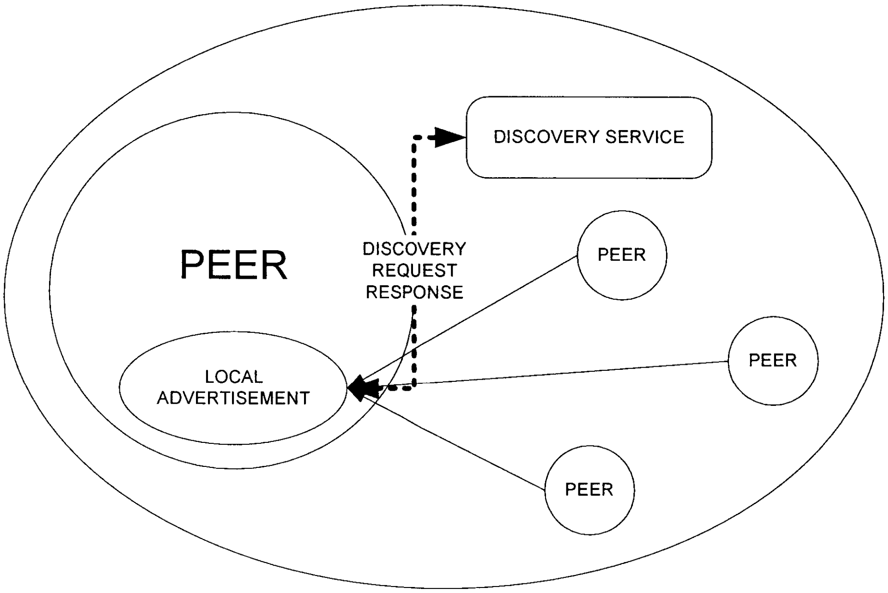


Figure 5

This creates a more structured transaction environment without losing significant benefits from a decentralized topology. Obviously more rigid, JXTA developers are making improvements in distributed searching and caching advertisements for quicker returns. Although JXTA possesses many of the desired attributes, the network protocols are still in the development stage and not optimal for projects with imminent deadlines [6].

2.2.4 Decentralized – Gnutella

Gnutella is a protocol for distributed search and retrieval of inconsistent data. It is the most decentralized model discussed in this thesis. Gnutella views servers and clients equally as servents. Servents provide client applications so users can issue queries, view search results, accept queries from other servents, check for matches against their local data set, and respond appropriately. As a decentralized topology, the Gnutella protocol is highly fault-tolerant but difficult to optimize with a large number of servents. The following set of descriptors are used for communicating data between servents:

- Ping – Used to actively discover hosts on the network. A servent receiving a Ping descriptor is expected to respond with one or more Pong descriptors.
- Pong – The response to a Ping. Includes the address of a connected Gnutella servent and information regarding the amount of data it is making available to the network.
- Query – The primary mechanism for searching the distributed network. A servent receiving a Query descriptor will respond with a QueryHit if a match is found against its local data set.
- QueryHit – The response to a Query. This descriptor provides the recipient with enough information to acquire the data matching the corresponding Query.
- Push – A mechanism that allows a fire walled servent to contribute file-based data to the network.

Additionally, Gnutella uses a Breadth-First-Search (BFS) and Time-To-Live (TTL) algorithm to search adjacent servents for a given query. The client initiating the query pings adjacent servents, the number of which are designated in the protocol, and propagates the search for TTL iterations. Consequently, this increases the network traffic geometrically. Chapter Five will discuss these scaling and optimization issues in greater detail. Gnutella developers have made significant headway in adding protocols that alleviate network tension. The MetaData initiative allows metadata tagging for queries that improves response quality. The Ultrapeer protocol allows caching of file location by servent. This takes on the attractive attributes of a more centralized topology without giving up the benefits of network redundancy. The following diagram gives a conceptual illustration of how individual servents operate on the network.

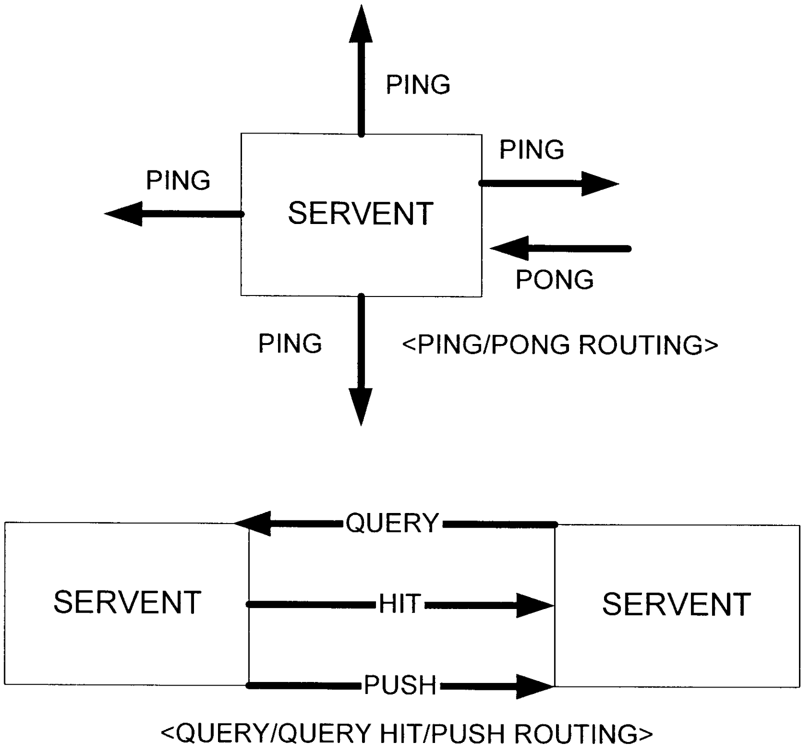


Figure 6

LimeWire is a software package, based on the Gnutella protocol, which enables individuals to search for and share computer files with anyone on the internet. A product of Lime Wire, LLC, LimeWire is compatible with the Gnutella file-sharing protocol and can connect with anyone else running Gnutella-compatible software. The LimeWire program will connect at startup, via the internet, to the LimeWire Gateway, a specialized intelligent Gnutella router, to maximize the user's viewable network space. LimeWire is written in Java, and will run on Windows, Macintosh, Linux, Sun, and other computing platforms. LimeWire takes advantage of the Gnutella protocols and is also an open source initiative. This is the open source protocol used for the BrainFist client [8].

2.2.5 Hybrid – FastTrack

The FastTrack Stack is a hybrid topology that combines centralized super nodes with decentralized leaf nodes and supports a greater variety of file formats such as movies and software applications. Similar to the other file-swapping service like Gnutella, FastTrack has a decentralized base and does not rely on any central servers. However, the network works slightly differently from others applications like Napster. Napster-hosted servers contain master lists of files on the PCs of users on its network. With FastTrack, a user logs onto the network and is directed to a SuperNode by the FastTrack servers. These SuperNodes contain lists of user files for search and download. Additionally, these SuperNodes are hosted directly on user's PCs so it becomes virtually impossible to trace and remove files from this network. Also, the number of SuperNodes increases as per the demand essentially forming a cluster of thousands of machines that intelligently handle search requests and file routing [9].

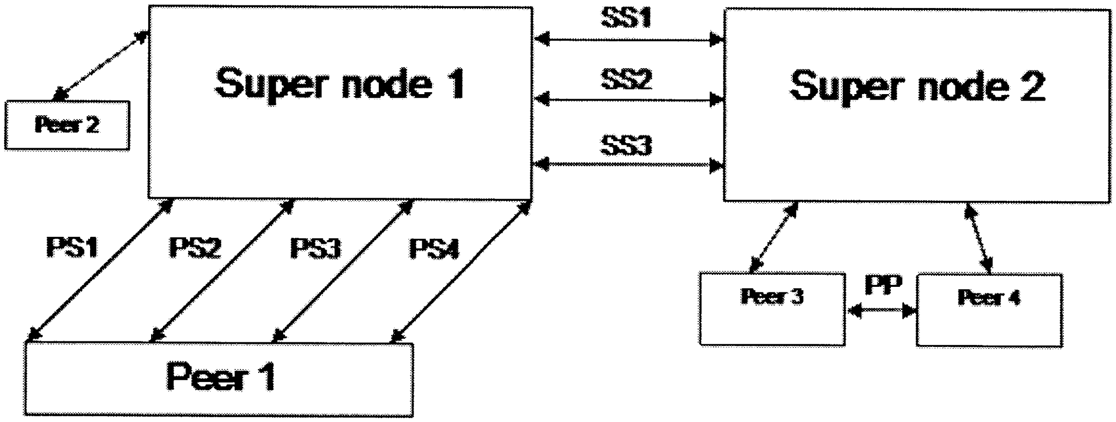


Figure 7

PS: Peer to Super node communication

- User login and authentication using unique user name and password
- Search queries regarding resources required based on metadata or free text
- Search results providing matching resources and their location

- Resource information sharing

SS: Super node to Super node communication

- Replication of configuration information
- Search queries regarding resources required based on metadata or free text if resource information is not available locally on the super node
- Search results providing matching resources and their location

PP: Peer to Peer communication

- Resource and intellectual capital sharing among the peers
- Instant messaging between the peers

FastTrack's network topology is a distributed, self-organizing network. Neither search requests nor actual downloads pass through any central server. The network is multi-layered, so that more powerful computers become search hubs or SuperNodes. Any client may become a SuperNode if it meets the criteria of processing power, bandwidth and latency. Network management is 100% automatic since SuperNodes are generated according to network demand. This is currently the most popular P2P network for file swapping over the Internet although it is facing legal issues since the super nodes are easy to identify on the network, hence easy to shut down by participating ISPs [9].

Although file sharing itself is not a violation of most ISP's acceptable use policy (AUP), it is the SuperNode feature of these applications that is a violation of AUP. This SuperNode feature violates AUP in two ways: First, the SuperNode functionality makes these programs operate as a power server. Servers are generally not allowed on a residential cable modem. Second, the SuperNode potentially talks to millions of other computers on the Internet all at the same time, abusing bandwidth, which is another AUP violation. As long as these applications do not exhibit signs of abuse, ISPs will allow them to function. If abuse is detected, all programs using this technology will be severely limited to keep their bandwidth within AUP limits [9].

2.2 Adaptive Architecture

The intermittent nature of a P2P network presents unique problems as well as great promise. A fundamental paradigm in P2P is that of a large community of intermittently-connected nodes that cooperate to share files. Because nodes are intermittently connected, the P2P community must replicate and replace files as a function of their popularity to achieve satisfactory performance. Given the nature of the content management problem, which is discussed in great detail in the next chapter, the BrainFist client is based on the open source, Gnutella-based, LimeWire protocol. The decentralized and adaptive nature of the open source Gnutella is highly attractive for the content management problem since most applications will only require service for less than 50,000 users.

Chapter 3

The Problem

In this chapter, the problem of content management is analyzed. Understanding the parameters of the problem for the purpose of this thesis is crucial in limiting the scope of the solution. The requirements and design of the solution is covered in the next chapter. This chapter is mainly concerned with the general understanding of content management, specific and applicable aspects and relevant examples of commercial products that will help in illustrating the theory. This chapter is organized as follows. First, the definition of content management is specified. Second, the relevant attributes are defined. Finally, commercial examples of content management products and market projections are used to flesh out the preceding concepts.

3.1 Summary

There are three key trends driving the need for, and emergence of, distributed content management solutions: explosion of unstructured data; the critical need to formally manage content; and internetworking and collaboration within and between enterprises. These trends are converging to produce two key requirements—the need to create superior online user experiences and the need to work collaboratively. Distributed content management systems address the need to access content wherever it resides, produce content while maintaining control over it, and collaborate efficiently by sharing data real-time within a distributed network of stakeholders. These systems create virtual content repositories that reduce the need for structured storage.

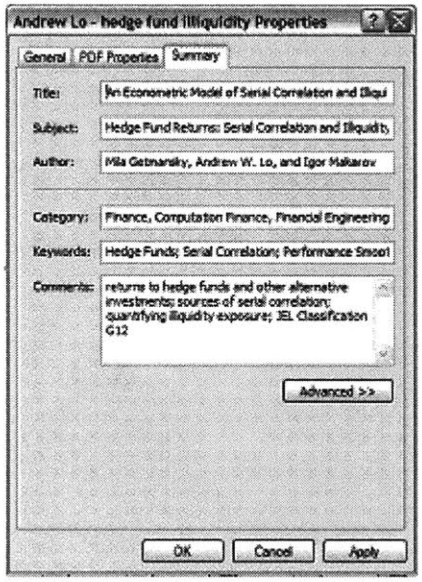
To clarify, the phrase "content management" has various definitions depending on what an organization may need or a vendor may offer. There is no general purpose or standard content management system that can satisfy today's diverse business needs across the board. Therefore, factors for successfully implementing a content management system vary depending on individual business needs. These include strong repository management for storing meta-data (such as indexes and fields) and managing users' interactions with the stored content through library services, workflows and delegated administrative capabilities for distributing and managing roles and responsibilities across business units [10].

3.1.1 Meta Data

Meta-data is often described as "data about data". David Marco presents a more complete definition as

> ...all physical data (contained in software and other media) and knowledge (contained in employees and various media) from inside and outside an organization, including information about the physical data, technical and business processes, rules and constraints of the data, and structures of the data used by a corporation [11].

Metadata enables content to be retrieved, tracked, and assembled automatically, and makes content accessible. The primary benefits are reduction of ‘useless’ content, proliferation of ‘useful’ content through search and replication, improved workflow, lower maintenance efforts and reduced costs. Metadata for reuse can be particularly useful in eliminating redundancies in recreating content. Authors can search for elements before beginning to create content that is probably not original anyway. This saves time and hours that are crucial in any business. Furthermore, through constant reuse, metadata is applied automatically, based upon the document definition, (e.g., type of content), while other metadata is added by the author (e.g., keywords), making the content more valuable for future users. Metadata also allows content to be retrieved through searching, either in an authoring tool, or in the retrieval application itself. Common metadata includes but is not limited to the following:



Title: An Econometric Model of Serial Correlation and Illiqui

Subject: Hedge Fund Returns: Serial Correlation and Illiquidit

Author: Mila Getmansky, Andrew W. Lo, and Igor Makarov

Category: Finance, Computation Finance, Financial Engineering

Keywords: Hedge Funds; Serial Correlation; Performance Smoot

Comments: returns to hedge funds and other alternative investments; sources of serial correlation; quantifying illiquidity exposure; JEL Classification G12

Figure 8

Metadata for retrieval enables users to specifically define which content elements they want to view. This metadata can also be used to dynamically populate content for users, based on specific profiling information. Status changes based on metadata are initiated through workflow automation, as well as by end users. Properly defining the metadata you need helps to make sure that the right information is delivered to the right person, for the right reason, at the right time [10].

3.1.2 Structuring Content

Information models determine the necessary components needed for desired information products. However, information models don't tell you how to write the content effectively. Structured content principles provide these guidelines. Effective structured content guidelines used in conjunction with models produce clear, understandable, and very consistent materials. Information can be structured such that each time someone views an information product (e.g., brochure, user guide, etc.); it contains the same types of information, in the same order, with the same presentation style. Structured content guidelines help to govern how content is written. Guidelines are based on how the information is going to be used and what type of information it is. Organizations normally establish structured content guidelines for each type of information product produced. There may be structured content guidelines for all the information elements contained in: policy documents, procedure documents, annual report, employee newsletters, and marketing collateral. All updates must be made at the source so that wherever the content is used, it is updated consistently [10].

For example, within an enterprise, information often suffers from inconsistencies in presentation, structure, and organization. Standards for producing information may vary from department to department, and often one department will not know that another department is working on a similar information product or, that they could use information your department is producing. Inconsistencies can cause frustration, lost time and money while users try to find and interpret information, as well as the costs from having to rewrite information for multiple uses. Structured content adheres to principles of cognitive psychology and is based on how people read and comprehend information. Structured writing also assumes that "not all information is created equally." In other words, information differs according to its type and should be consistently structured in a way best suited to its type [10].

For example, a procedure is different than a process, or a concept, and should use a structure best suited to procedural information. It is necessary to define the content structure through audiences/product lines/platforms; to identify how content will be reused; to decide how best the content should be structured, based on its type and potential audiences, based on principles of clear communication; and to formalize and publish structured content guidelines for all authors to follow, showing how all the pieces fit together to form a complete information product. Structured content relies on content standards rather than format standards. Content standards refer to the type of content in each element, and how it must be structured in order to be reused. While format is critical in helping users to read and comprehend information, it is addressed separately from content. Format standards refer to how the information must look, in the published outputs. This allows writers to focus on the content—ensuring the content is accurate and contains the necessary elements for comprehension and for reuse. Format is addressed through information design, and is normally attached to content elements through stylesheets (e.g., XSL or cascading style sheets) [10].

3.1.3 Dynamic Content

Dynamic content is information that is assembled only when it is requested. It does not exist as a document; rather, it exists as a series of information objects that are assembled in response to the user's requests or requirements. The majority of dynamic content is delivered dynamically through the web or can be dynamically assembled as a PDF document. Dynamic content provides corporations with the ability to provide exactly the right information at the right time to their customers. It provides the ultimate flexibility in information reuse. Dynamic content can create multiple information products on demand; specifically address customer needs; and reduce the cost of creation of multiple information products [10].

Dynamic content uses an information management methodology known as personalization. Personalization means providing specific, relevant information to defined users or user groups. Customers are assigned logins. Associated with each login is a customer profile that identifies the customer's information needs. Using a combination of user profiles and user selections, the system learns the user's information patterns and determines what additional information may be relevant. Personalization is supported by profiling – the process of describing a customer's needs, requirements, and interests, based on a user profile. To create a customer profile, it is necessary to conduct a thorough audience and information analysis, develop information models, and assign metadata [10].

3.1.4 Automated Workflow

Workflow tasks typically have multiple steps, involve more than one person, and have a well-defined objective or end result. Automated workflow can be defined as a system that is designed to control the movement of data from one process to another, triggering appropriate actions and/or generating process control messages as required. As part of content development, automated workflow can help move content elements through the steps in the authoring and publishing processes, to be able to output content using the content management system. A well-designed workflow solution combines automation, business rules, metadata and a willingness to change on the part of content producers. Workflow can assist the collaboration, automating much of the process and enabling continuous content status tracking. One of the important benefits to be realized from the use of workflow tools is the automation of maintenance tasks for version archiving and audit trails. In industries where certain regulatory compliance is mandatory, automated workflow provides a practical solution to keep track of the types of details necessary for compliance [10].

There should be a separate workflow for each main content type to be produced. Typically, business analysts perform this analysis and produce the workflow diagrams. Workflows should be defined independently of the systems used to help facilitate the tasks, to make sure that the system is designed to support people's work processes, not the other way around. Once the workflows have been designed, information technologists, systems engineers and/or systems integrators create the workflows [10].

3.1.5 eXtensible Markup Language

XML is a standard for the development of markup languages for web-based information. Unlike HTML, which has a fixed set of tags, XML lets the user define his or her own markup language. Based on Standard Generalized Markup Language (SGML), XML has been optimized for web-based delivery of all kinds of information. A family of standards, rather than a single standard, XML defines all aspects of information presentation, including markup, linking, style, structure, and metadata. While it is possible to develop a content strategy without XML, using certain traditional authoring tools, XML supports the chunking of information into elements down to the paragraph or even sentence level. This chunking, along with efficient use of metadata, enables more efficient search and retrieval of content elements when used in conjunction with a content management system [10].

Using XML has a number of benefits that directly support a content strategy, including reuse of existing content, reducing redundancy and costs; dynamic content delivery of personalized content; separation of content from presentation to allow multiple output formats; better-managed content, resulting in reduced costs; and improved search and retrieval capabilities through the use of metadata. XML requires a DTD (Document Type Definition) to support the development and management of content. The DTD is like a structural template: it explicitly defines the structure of the content. This explicit structure ensures that authors can only enter content which follows the structure. All of the required pieces of information are in place and in the correct order. This will assist authors in writing rapidly and eliminating validation errors. XML tags define the content, the "look and feel," applied to the content, depending upon the desired output [10].

For example, the content can look one way on paper, another in HTML, and many other ways if used in an article, presentation, or poster. The "look and feel" is defined by the appropriate stylesheet selected in the authoring/review cycle. XML allows the creation of semantic tags, rather than generic tags. In XML, tags could be called "Abstract," "Keywords" or "Author." The semantic tags automatically provide metadata about the content they enclose, and can be interpreted for display in many ways. For explicit metadata, XML can define attributes for the elements in a document. Similar in syntax to HTML attributes (/color/ = 'red'), XML attributes are defined by the user, to provide whatever additional information is required to identify the use of the information. Attributes can be used to identify information that is specific to format, product, user, or use [10].

The first usage of XML has been application-oriented, relying heavily on databases for fast access to information. This means that the XML-based content can easily be "chunked" for storage as elements rather than large sections and files, for fast access to individual elements of information. XSL, eXtensible Stylesheet Language, is the piece of the XML family of standards that defines formatting. But unlike a traditional stylesheet, which manages the look of a document, XSL is used to convert XML documents to other formats. These include HTML for web output, other markup languages like Wireless Markup Language (WML), and PDF. XSL stylesheets can be used to manipulate information, including sorting, filtering, moving and repeating information, and generating new information, such as tables of content and lists of figures. XML builds documents out of individual content files on the fly. Individual pieces can be assembled upon demand, in response to user requests or to meet the needs of a specific output format [10].

3.2 Applications

Vendors have consistently underestimated the difficulty of building robust document management applications. P2P start-ups have had a difficult time reaching the companies that would most benefit, possibly due to certain platform monopolies that prevent outside vendor product integration. Established Independent Software Vendors (ISVs) have been notoriously slow in adopting the P2P technology since, by its very nature, P2P gives away most of its value to its users. Additionally, P2P start-ups have become entangled in legal battles that, incidentally, frighten the larger ISVs. Finally, ISVs have just not been able to develop effective P2P business models. There are some exceptions to P2P solutions for content management. NextPage and hotComm offer such solutions while Microsoft, the largest software vendor of all, offers its content management solution, InfoPath, which is integrated with Microsoft's XML web services.

3.2.1 Baker & McKenzie

The Chicago, IL-based Baker & McKenzie is one of the nation's largest law firms, boasting 62 offices worldwide, $940 million per year in revenue and 2800 attorneys. It is a gigantic task to keep track of the documents produced by these globally-dispersed lawyers – and squeezing the value out of that content even more so. While the firm's document management technology was keeping track of the documents as they flowed in and out, it wasn't well-equipped to extract higher value from a far-flung content network. The Chicago firm is using a peer-to-peer application to make information about contract negotiations, financial strategy and legal discussions accessible to teams of lawyers and clients around the world. Baker & McKenzie is using NextPage Inc.'s Matrix peer-to-peer (P2P) knowledge management software. The law firm was an early customer of NextPage's NXT platform, a P2P content management system that connected its 2,800 attorneys in 35 countries. The price for NextPage's distributed networking infrastructure is $85,000 for a platform that supports 250 users (about $340 per user) [12].

3.2.2 Morningstar

Morningstar Systems, a Collaborative Systems consulting company, delivers enhanced Knowledge Management solutions to Government agencies and 2000 organizations in North America and Europe. Morningstar Systems implemented hotComm's dynamic peer connectivity technology, into the Knowledge Management solutions it supplies to clients seeking to capture, distribute and leverage corporate knowledge. Specifically, the hotComm technology was integrated into Morningstar System's Web Based Collaborative Knowledge Management solutions, a framework which helps companies quickly identify, prioritize and communicate their knowledge management initiatives. All peers on the network have server capability with Instant Messaging (IM) Live peer to peer technology. This is a powerful, scalable and secure peer to peer architecture that provides fast, efficient, private interactive access or exchange of text, speech or files between registered peers across the Internet. hotComm includes IM–Live, which delivers immediate, secure and server-less, Internet-based Messaging for any online collaboration opportunity. When any user clicks on an IM–Live link, placed on a Web page or in the signature of an email message, IM–Live launches a local Java applet and automatically creates a direct and secure real time connection to the associated hotComm peer [13].

3.2.3 Microsoft InfoPath

InfoPath 2003 enables the creation of rich, dynamic forms that teams and organizations can use to gather, share, reuse, and manage information – improving collaboration and decision-making within an organization. Microsoft Office InfoPath 2003 is a new program in the Microsoft Office System that provides a flexible and efficient way to collect the information needed to make better-informed decisions. InfoPath enables organizations or teams to easily reuse and repurpose the information they collect. The information can then be shared, reused or repurposed across business processes and organizations support for XML Web services. InfoPath includes powerful features like data validation in forms without writing code; forms creation solutions using existing customer-defined schemas; publication of forms to a shared network folder, a Web server, SharePoint Services form library; notification and dissemination by e-mail; and maintenance of the most up-to-date form automatic downloads [14].

3.2.4 Market Projections

Sales of content management software will total $3 billion by 2004, up from $900 million in sales during 2000. This shows a projected compounded annual growth rate of 35%. In 2001, the Yankee Group Corporations paid more than half a billion dollars for software to manage Web content alone while separate products now exist that address Web content, internal documents, digital rights, and other areas, there will be a push to integrate all these functions under the rubric Content Management [15].

Chapter 4

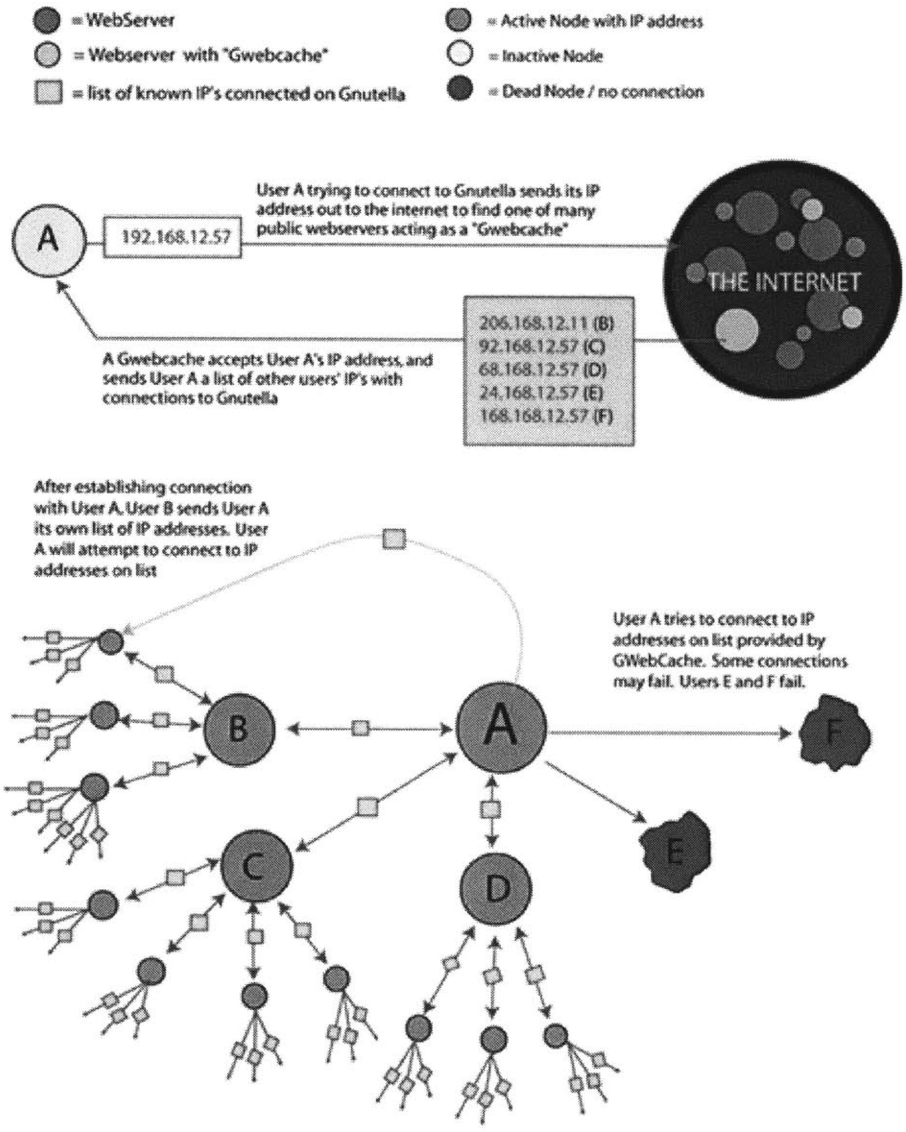
Design

The goal of the BrainFist client is to create a P2P enabled Content Management Solution that facilitates the sharing and management of files while integrating into the individual user's workflow habits. Specifically, BrainFist, in this initial beta version, is a desktop client for strategy and management consultants – a tool that manages files through virtual file classification, populates metadata and provides a P2P interface for searching desired content on fellow BrainFist clients. Accenture is the largest consultancy in the world with 22,000 users and this is the number used as a cap for the target user group [16]. In Chapter 5, the optimization of the client on the Gnutella network is discussed in greater detail, addressing optimal number of users, TTL parameters, and ultapeer configuration. Furthermore, the basic features of the developed beta are discussed here with references to future versions and significant feature additions. Chapter 6 will address future product developments as well the implications for BrainFist users.

BrainFist is a client developed using the LimeWire open source application code. Written in Java and compatible with the major operating systems, the LimeWire client code has evolved into both a sophisticated graphical user interface and a core Gnutella protocol and sharing engine. Therefore, this chapter is laid out in two primary segments. First, the LimeWire protocol is addressed in great detail. Second, the BrainFist requirements are detailed with supporting examples from developed code and screen shots. Finally, the chapter addresses major issues in future product versions.

4.1 LimeWire

This section will lay out the important class structures throughout the LimeWire code, discuss operational issues and generally assist in the understanding of the LimeWire code at the design level. The Java source code of LimeWire is well documented, allowing developers to readily build Javadocs from the source code. Reviewing the class structures and LimeWire terminology will give the reader a fairly crisp understanding of the BrainFist architecture.

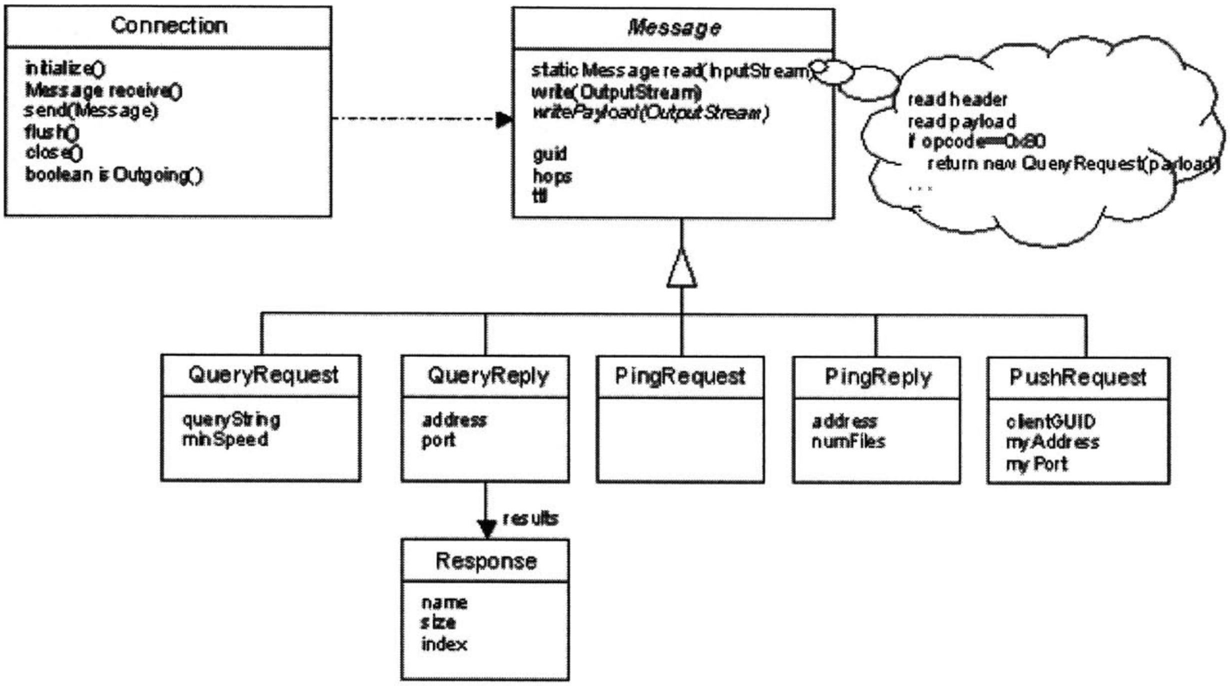

= WebServer
= Webserver with "Gwebcache"
= list of known IP's connected on Gnutella
= Active Node with IP address
= Inactive Node
= Dead Node / no connection
A
192.168.12.57
User A trying to connect to Gnutella sends its IP address out to the internet to find one of many public webservers acting as a "Gwebcache"
THE INTERNET
206.168.12.11 (B)
92.168.12.57 (C)
68.168.12.57 (D)
24.168.12.57 (E)
168.168.12.57 (F)
A Gwebcache accepts User A's IP address, and sends User A a list of other users' IP's with connections to Gnutella
After establishing connection with User A, User B sends User A its own list of IP addresses. User A will attempt to connect to IP addresses on list
User A tries to connect to IP addresses on list provided by GWebCache. Some connections may fail. Users E and F fail.
B
A
F
C
D
E

© LIMEWIRE

Figure 9

4.1.1 Messages and Connections

Message is an abstract class representing all types of Gnutella messages. Message stores data common to all Gnutella messages, like GUIDs and has the following subclasses for each Gnutella message: *QueryRequest*, *QueryReply*, *PingRequest*, *PingReply*, and *PushRequest*. Incoming and outgoing have separate constructors. Connection class implements a messaging connection with two constructors for incoming and outgoing connections and provides methods for reading and writing messages. Incoming takes an address and port and outgoing takes a Socket created by a *ServerSocket* [17].

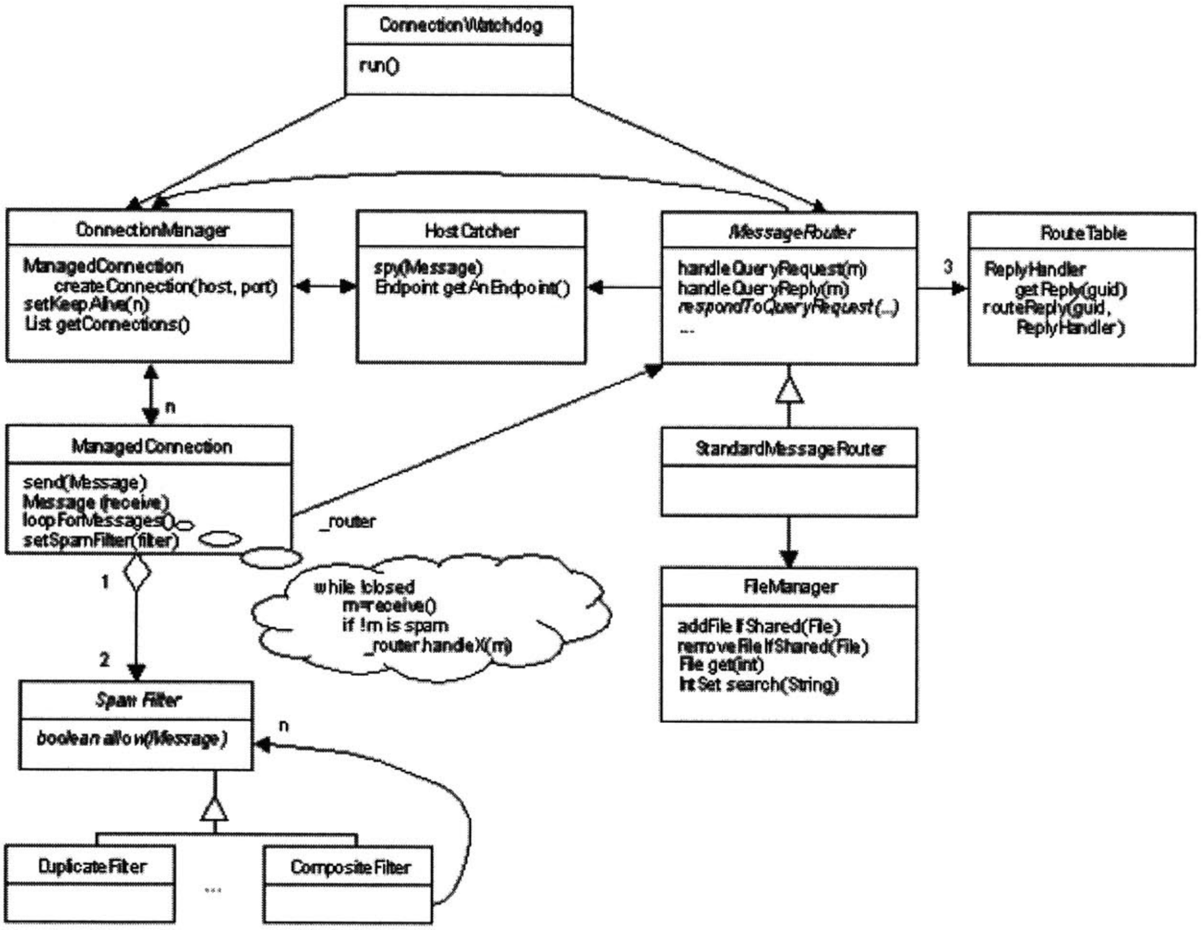


Christopher Rorhs, 2001

Figure 10

4.1.2 Fetching and Routing

Connection is subclassed by *ManagedConnection*, which allows automatic buffering and flushing of outgoing messages and drops messages in the following order: *PingRequest*, *PingReply*, *QueryRequest*, *QueryReply*, *PushRequest*. Furthermore, the *loopForMessage* method automatically reads messages. Finally, various statistics are automatically generated throughout operation. *ConnectionManager* maintains the list of all *ManagedConnection* instances and is responsible for fetching outgoing connections as needed. *ConnectionWatchdog* looks for connections that have not sent or received any data in the last few seconds and kills those connections that don't respond to a ping with TTL=1. *MessageRouter* broadcasts queries and pings to all of *ConnectionManager's* connections and sets up reply routing tables. Query replies are then routed to the appropriate connection. Two abstract methods are defined to respond to queries: *respondToQueryRequest*(..) and *respondToQueryReply* [17].

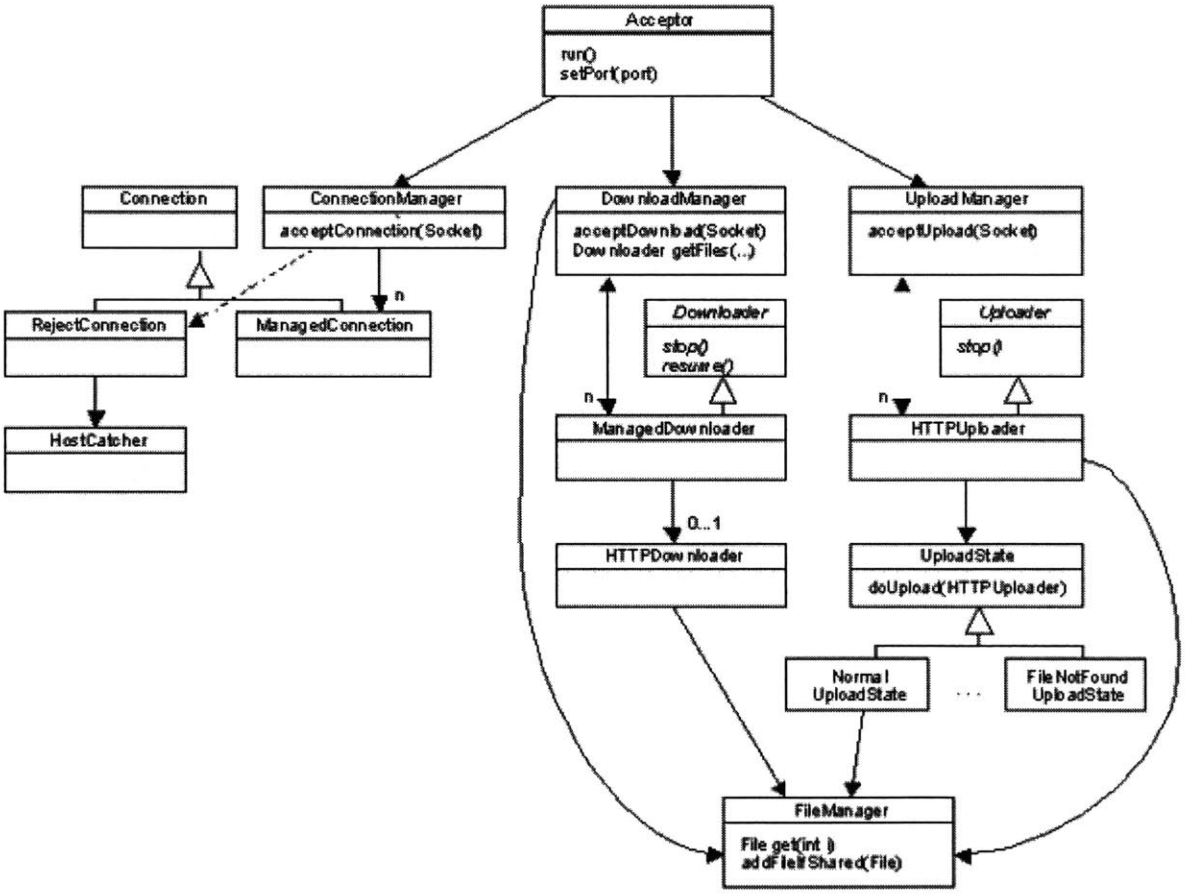


Christopher Rorhs , 2001

Figure 11

4.1.3 Uploads and Downloads

Acceptor is the class responsible for creating a server socket, accepting incoming TCP connections, and dispatching queries. Sockets for uploads and downloads are passed to *UploadManager* and *DownloadManager*. *ManagedDownloader* provides the logic for "smart downloading" from a list of locations. Each attempt from a single location is implemented with a single *HTTPDownloader* instance. *DownloadManager* uses *FileManager* to determine if you have already downloaded the file, and *HTTPDownloader* uses *FileManager* to share the file after downloading. Each upload is implemented with a single *HTTPUploader* instance [17].



Christopher Rorhs , 2001

Figure 12

4.1.4 Threading

LimeWire is a thread-intensive program, using a two-threads-per-connection model. Acceptor thread accepts incoming connections. *ManagedConnection* has two threads: one to read data from the connection and one to write queued packets. *ConnectionManager* creates several *ConnectionFetcher* threads to fetch outgoing connections. *ConnectionWatchdog* thread looks for and replaces 'dead' connections. Each Downloader and Uploader thread transfers a single file. The Main/GUI thread manages the details of the application interface. Worker threads keep the GUI responsive when calling potentially blocking operations [17].

4.1.5 Keyword Matching Algorithm

Keyword Substring Matching (KSstringM) is the current policy. It is implemented in a straightforward manner, by searching each file name in a list. The problem is that the time complexity is proportional to the number of files and the length of file names.

> A query Q matches a file name F if all strings in WORDS(Q) are a substring of F.
> For example, "ello eatle" will match "beatles yellow+submarine" since both "ello" and "eatle" are substrings of "beatles yellow+submarine" [17].

Keyword Subset Matching (KSsetM) is the weakest policy. It is implemented efficiently by indexing each file name and storing the keywords in a hash table. Then matching can be done in constant time. The space required is only proportional to the sum of the lengths of all filenames.

> A query Q matches a file name F if WORDS(Q) is a subset of WORDS(F). For example "submarine beatles" will match the above file name, but "sub beatles" will not [17].

Keyword Prefix Matching (KPM) is best implemented by indexing each file name and storing the keywords in a Trie, a tree-like data structure that takes advantage of words with common prefixes.

> A query Q matches a file name F if all strings in WORDS(Q) are a prefix of some element of WORDS(F). For example, "sub beatle" will match the above file name, but "sub eatles" will not [17].

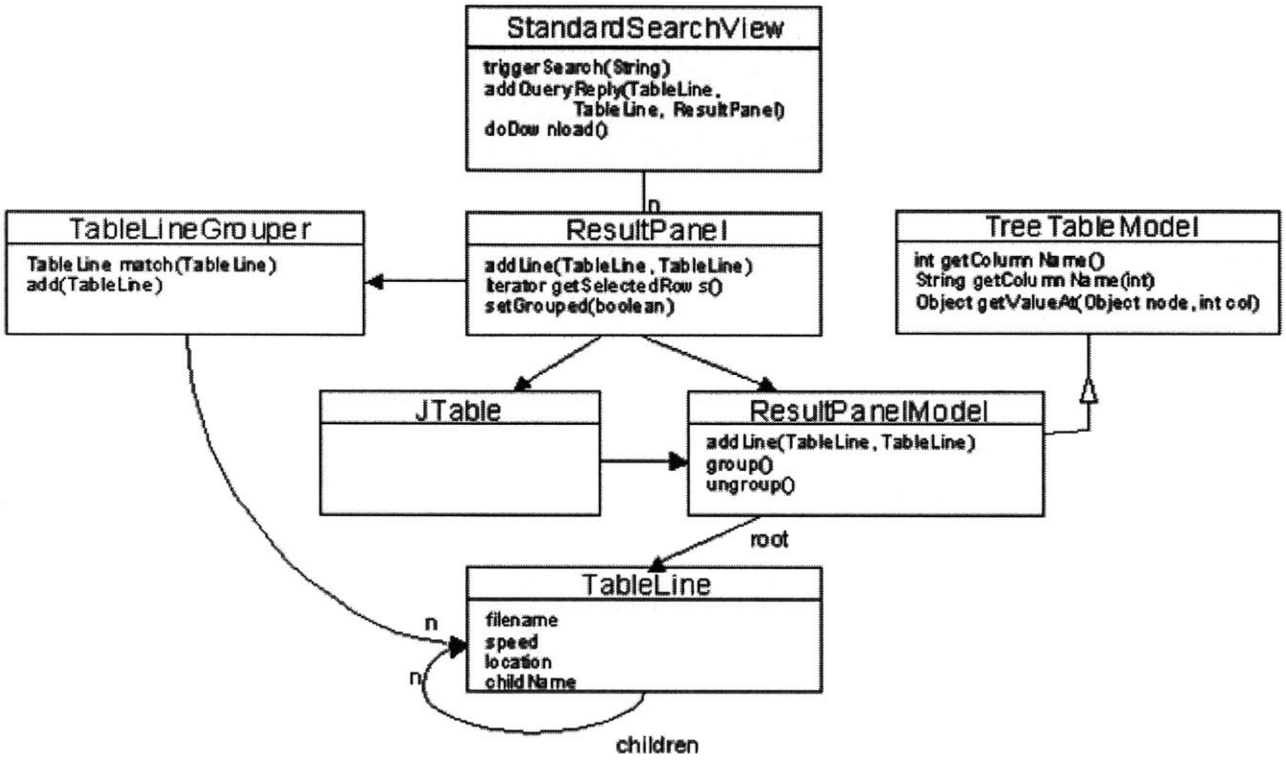
Faster Keyword Substring Matching (FKSM) is the same policy described in KSstringM and can be implemented in the same time as KPM by using a Suffix Tree, but at the cost of more space.

> Now let W be an element of WORDS(Q). Note that W is a substring of F iff W is a prefix of some substring of W. Using this idea, we simply match W against the Trie above. This takes time proportional to the length of W [17].

4.1.6 Search Results Grouping Algorithm

LimeWire defines two results to be similar if they have the same file extension, their file names differ by just a few characters, and their sizes differ by no more than 60k. More formally, file names are considered similar if they have an edit distance of no more than 4 characters or 5% of the file name, whichever is smaller. Calculating the edit distance of two strings can be done with a classic dynamic programming algorithm in $O^{(M2)}$ time, where M is the length of a file name. Each search result is ultimately stored in the GUI as a single *TableLine* instance. The result grouping algorithm is implemented in the *TableLineGrouper* class. The backend delivers search results to the GUI via *ActivityCallback.handleSearchResult*. This method adds the *QueryReply* to a queue in *SearchView* and returns immediately. The *ResultGrouperThread* then consumes each *QueryReply* from the queue. The search view table is implemented with a single

StandardSearchView instance, which holds a reference to multiple *ResultPanel's*, each representing a single search result tab [17].



Christopher Rorhs , 2001

Figure 13

4.1.7 Download Algorithm

The download method of *DownloadManager* takes an array of *RemoteFileDesc* (*RFD*) as an argument and returns a single *ManagedDownloader*. These *RFDs* do not necessarily need be identical or even similar. This means that a *ManagedDownloader* must manage multiple buckets internally. *ManagedDownloader* will manage one bucket at a time until it completion. It is important to make a distinction between groups and buckets. A group is set of search results that are approximately the same. Each search result is represented as an instance of *RFD*, which is basically a host/file pair. *RFD's* are grouped by *TableLineGrouper* when they are passed to the GUI. A bucket is a set of identical *RFD's*. Two *RFDs* are considered identical if they have the same hash. Furthermore, two *RFDs* are considered identical if they have the same name and size and don't have conflicting hashes (i.e., both hashes are null or at least one is null) [17].

When the user selects a number of search results and presses the download button, the GUI constructs a new downloader for each *RFD* using *DownloadManager.download*(..). However, any two *RFDs* in the same group will be passed to a single downloader. Hence a single *ManagedDownloader* will be created for each selected group, with multiple buckets within each group. Incidentally, grouping and bucketing are independent processes. Grouping is in the GUI and bucketing is in the backend. The basic idea behind swarmed (multi-source) downloads is to download one file in parallel from multiple servers. This increases throughput if the downstream capacity of the downloader is greater than the upstream capacity of the fastest uploader [17].

LimeWire uses a "split/steal" swarming algorithm. The controller thread of *ManagedDownloader* spawns a number of worker threads, each of which working in parallel to download identical data from multiple sources. LimeWire also uses two techniques to detect corruption when downloading. The first is to simply check the hash when the download is complete. This only works if the initial search result had a hash. Secondly, LimeWire checks overlapping regions of each download chunk. This requires verifying that the bytes being written match the area already on disk using *VerifyingFile* and *IncompleteFileManager* [17].

ManagedDownloader uses three template methods to control re-queries: *nextRequeryTime*, *newRequery*, and *allowAddition*. Three subclasses of *ManagedDownloader* exist primarily to control this re-query behavior each providing slightly different constructors.

RequeryDownloader (“wish list downloader”) will continue to issue re-queries using the original search keywords with the first results matching the search keywords are downloaded. *ResumeDownloader* (“incomplete file downloader”) will issue re-queries with the name and hash of the selected incomplete file and will only accept search results that match this incomplete file. *MagnetDownloader* re-queries by hash or keyword [17].

4.1.8 Critical Definitions

- **Broadcasting**: Messages on the Gnutella Network are sent in one of two ways. First, they may be broadcast, or sent to all hosts on your network horizon. Pings and search requests are broadcast. Second, messages may be routed, or sent only to particular location [8].
- **GUID**: Short for Global Unique Identifier, the GUID is a randomized string that is used to uniquely identify a host or message on the Gnutella Network. This prevents duplicate messages from being sent on the network [8].
- **GWebCache**: A distributed system for helping servents connect to the Gnutella network, thus solving the "bootstrapping" problem. Servents query any of several hundred GWebCache servers to find the addresses of other servents. GWebCache servers are typically web servers running a special module [8].
- **Horizon**: Your horizon is the group of Gnutella servents that you are capable of communicating with at a particular time [8].
- **Host Catcher**: The host catcher keeps track of the Gnutella hosts that sent these pongs, maintaining a list of active Gnutella hosts [8].

- **Message**: All information sent over the Gnutella Network is in one of five message types. It is either a ping, a pong, a search request, a search reply, or a push request [8].
- **Peer**: Two computers are considered peers if they are communicating with each other and playing similar roles. Gnutella's peer-to-peer model uses no servers, so the network is composed entirely of peers. Computers connected to the Gnutella Network are also referred to as "nodes" or "hosts" [8].
- **Ping**: When a new user joins the Gnutella Network, he broadcasts a message called a "ping request" to the network, announcing his presence on the network. Nodes which receive this ping, send a pong back to the pinging user to acknowledge that they have received this message [8].
- **Pong**: When a node on the Gnutella Network receives a ping request, it replies with a pong (also sometimes referred to as a "ping response"). This pong contains the responding host's IP address and port, as well as number of files the responding host is sharing and their total size [8].
- **Port**: On most servents, the default port for Gnutella is 6346. This means that a servent running Gnutella software is listening on port 6346. However, the user can change the port that is assigned to Gnutella on his computer, and he will still be able to communicate with other servents that are listening on port 6346 [8].
- **Push request**: When a servent is behind a firewall, other hosts are not able to connect to that servent directly to download a file. When this happens, the host trying to download the file sends a push request, asking the servent behind the firewall to connect out and upload the file to him [8].

- **Servent**: In the decentralized Gnutella model, each computer on the network is both a client and a server and is thus called a "servent" [8].
- **Time to Live**: Abbreviated "TTL," the Time to Live is the number of hops that a message will make on the Gnutella Network before being discarded. Each servent that views a message will decrement its TTL by 1, and will discard that message when the TTL reaches 0. This prevents messages from being sent back and forth across the Gnutella Network indefinitely. The default configuration is TTL = 7 [8].
- **Ultrapeer**: A non-fire-walled, high-capacity servent that shields "leaf nodes" from the majority of messaging traffic. A leaf node is a node shielded traffic by an Ultrapeer. The default configuration are six Ultrapeers and 75 leaf connections [8].

4.2 Requirements

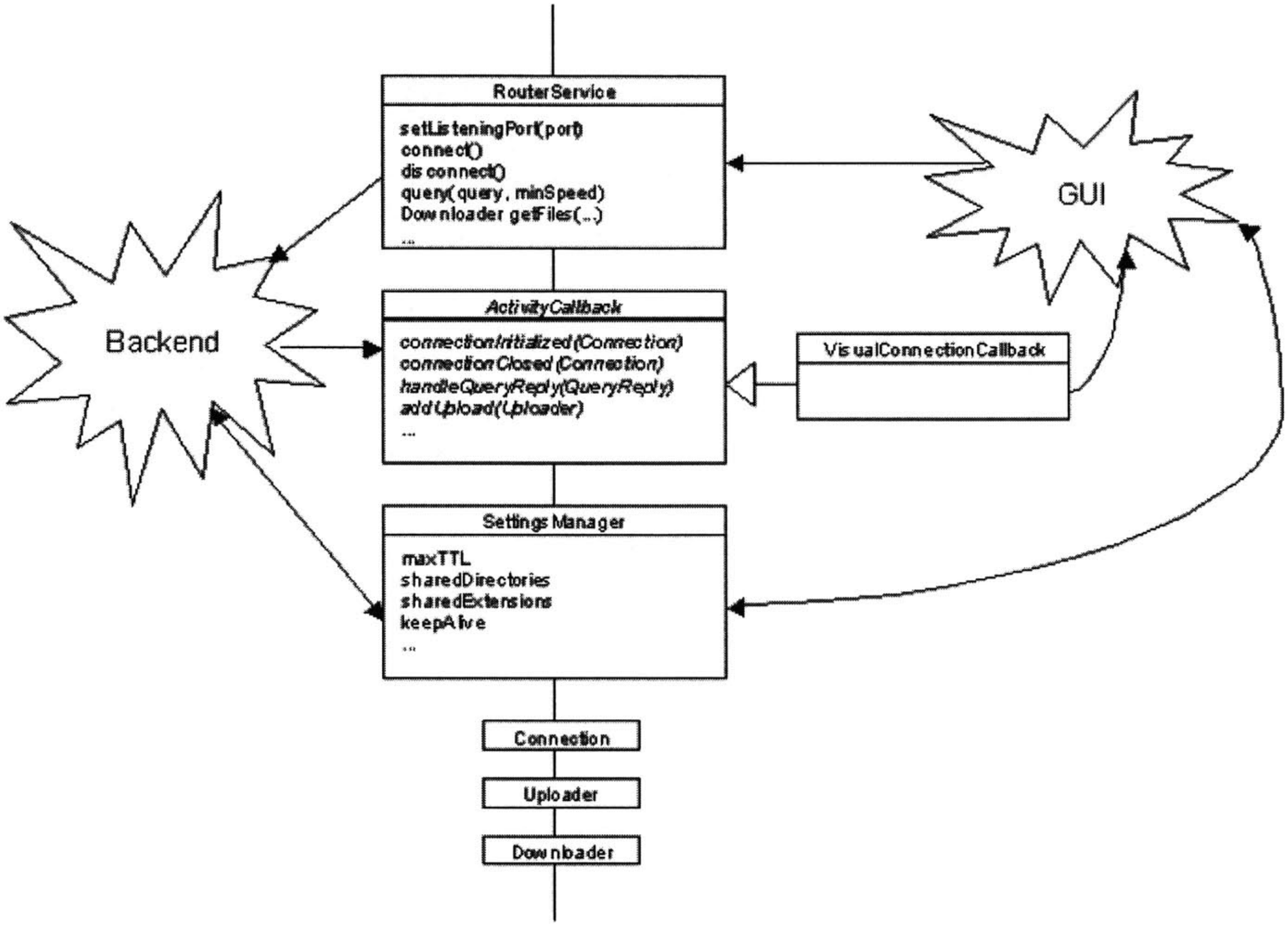
In beginning of this chapter, the solution was defined for a Strategic/Management Consultancy (SMC) of no more than 50,000 users. This is done for innocuous reason. First, an SMC organization is homogenous from a content management perspective. The content created, or recycled, is fairly simple from a computing perspective: PDFs, Word documents, Excel files, PowerPoint presentations, and other 'processed documents.' There will most likely not be a need to search out large SQL databases or stand alone applications from peers. These types of items will most likely be directed and centralized. However, the number of processed documents will be very large and exist in informational silos, as discussed in the previous chapter. Hence, the SMC becomes an attractive customer for the first beta of the BrainFist client. The goal of the BrainFist client is to provide a versatile content management tool based on the LimeWire protocol with the following initial feature requirements:

- Easily manage existing client files through a 'drag and drop' user interface which allows for instant metadata updates and/or changes.
- Allow the user to specify metadata categories through the customization of metadata parameters such as 'Subject' and 'Keywords.'
- Provide a P2P based search tool that allows the user to share with other client holders as well search remote repositories.
- Create a customized P2P network tool that facilitates the creation, management and proliferation of useful content.
- From the workflow process perspective, the tool should integrate and amplify the user's work habits.
- From a computational perspective, the tool should scale reasonably and not become a burden on the local network (this will be addressed in the next chapter).

The key to file management features is that the tool itself is so useful that its constant use ensures a consistent P2P network and helps to reduce 'free rider' problems intrinsic to the nature of P2P. Future feature requirements and product versioning will be discussed in the sixth and final chapter.

4.2.1 GUI

The interface between the backend and the GUI is well-defined in the LimeWire design protocol. The GUI triggers backend operations through a *RouterService* instance. The backend communicates with the GUI through the *ActivityCallback* interface. *SettingsManager* provides a list of properties shared between the backend and GUI, such as the list of shared directories and is notoriously difficult to modify. *RouterService*, *ActivityCallback*, and *SettingsManager* have a large number of methods and any significant feature requires a modification to one of them. Methods in *RouterService* and *ActivityCallback* should generally be non-blocking [17].



Christopher Rorhs , 2001

Figure 14

4.2.2 Views

BrainFist searches with meta-data information provided within the document files. The BrainFist metadata index will be a virtual repository, listing all metadata about files selected by the user to be shared on the BrainFist network. The file can sit in a default folder created by installing the BrainFist client and can remain in their respective locations on the user's hard drive. The user will be allowed to select which physical folders and/or files to share within the BrainFist options tab. The incentive to share files is built in the client itself because to use the file management features, the user, by default, must select the file and/or folders to be shared in order to use the file management features of BrainFist. There are options to not share files and folders and still use the file management features but the default is to share. Furthermore, as long as the user is connected to internet the BrainFist client, by default, is active. In order to use the file management features, the BrainFist client must be activated. There is an option to use the file management features and not be connected to the BrainFist network if the user has an internet connection but the default is to be on the network when activated. The initial window that appears on the user's desktop when the BrainFist client is activated is shown below.

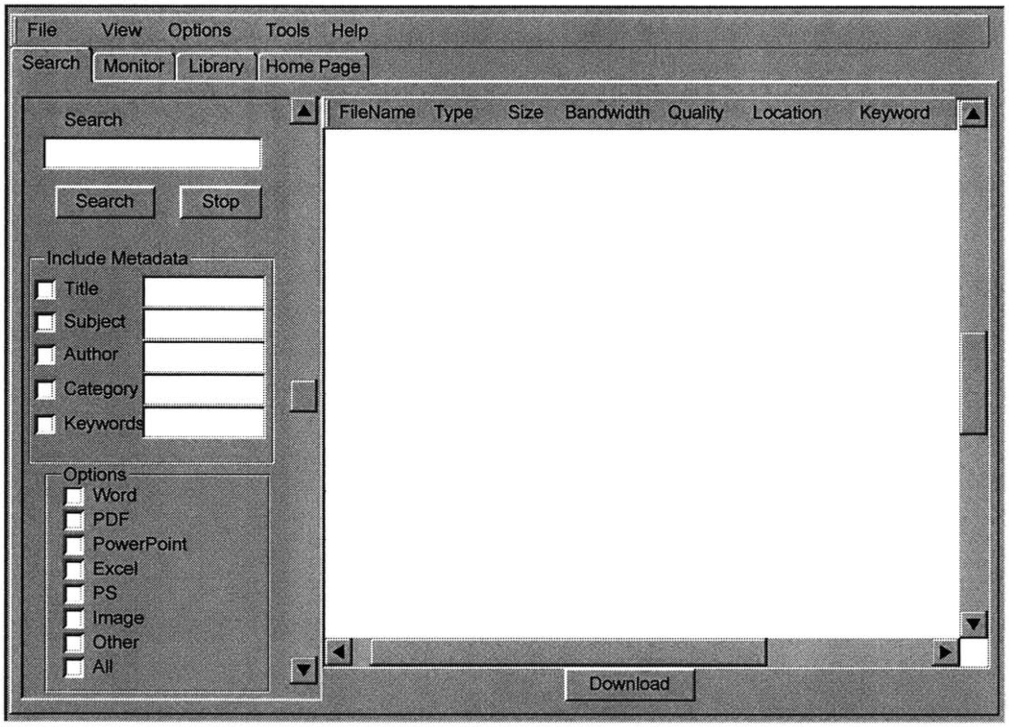
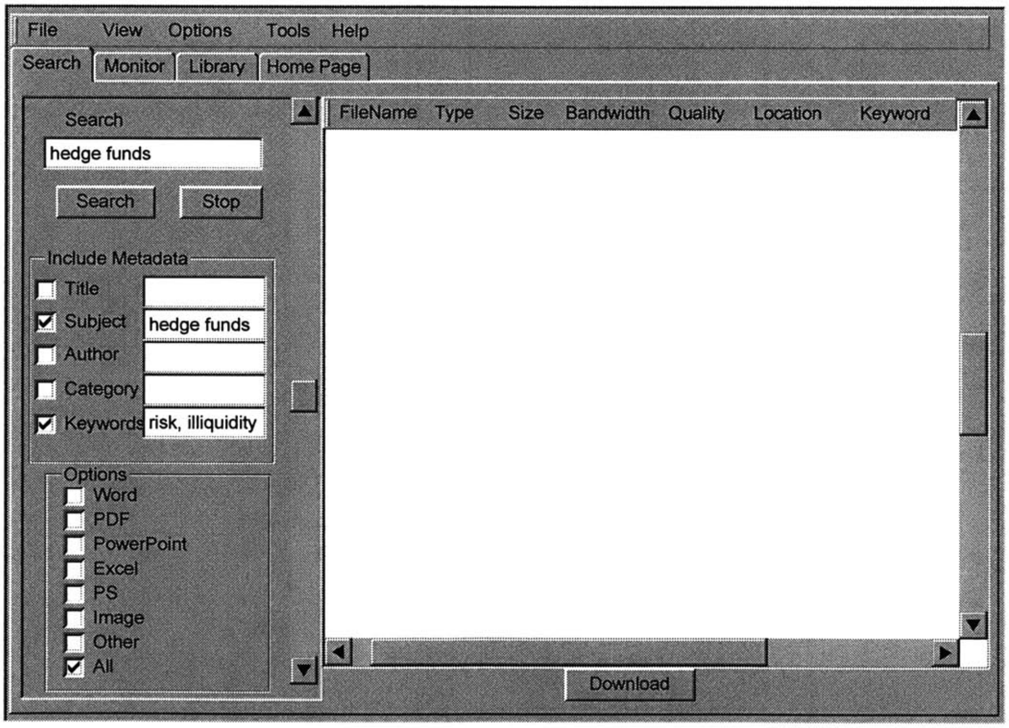


Figure 15

Search View. The Search view is relatively straightforward. The user can use this view to search for relevant files by populating the metadata fields and file options. Of course, the fewer the user selects, the more results are included. Using an earlier example, the user needs to find a document on hedge funds for a client to determine viable investment options. The user needs to have a general understanding of the risks involved so a search of relevant documents becomes priority. Instead of using Google or some other search engine that will display results of greatly varying relevance, the user decides to the BrainFist client on his or her firm issued laptop. This way the user can search the documents of fellow consultants, peers, for relevant and useful documents. The user fills out 'hedge funds' which automatically populates the Subject metadata box. Additionally, the user populates the Keyword metadata box. Finally, the user selects All in the Options frame because he or she is looking for any media at this point. The screen below is what he would see.



Search
hedge funds

Include Metadata

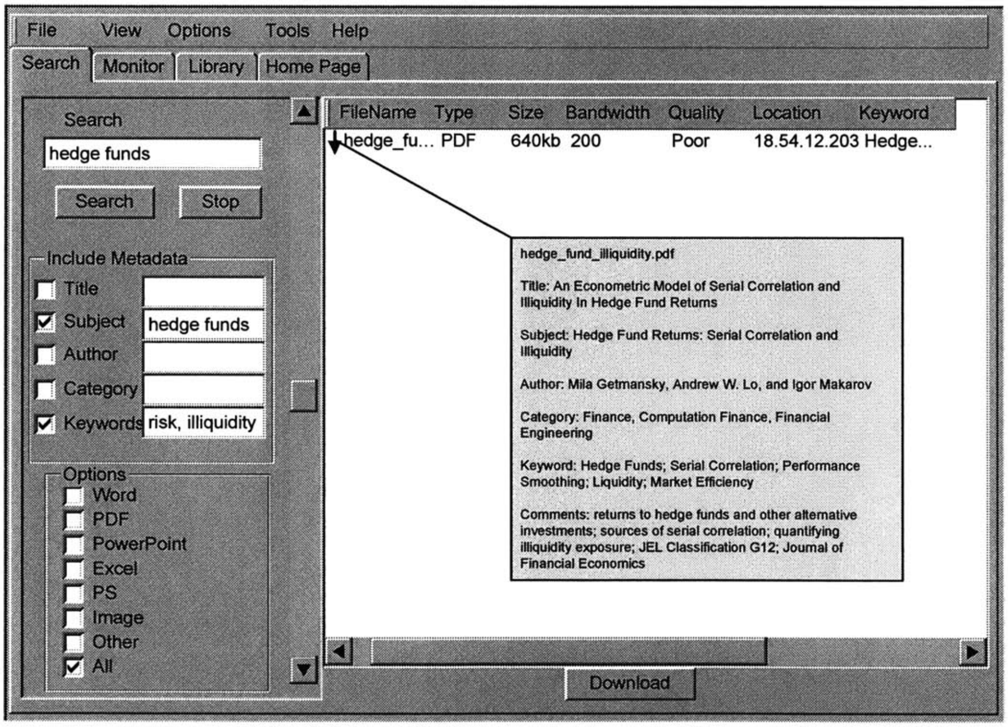
☐ Title
☑ Subject hedge funds
☐ Author
☐ Category
☑ Keywords risk, illiquidity

Options

☐ Word
☐ PDF
☐ PowerPoint
☐ Excel
☐ PS
☐ Image
☐ Other
☑ All

Figure 16

This simple search will result in numerous returns since the parameter are intentionally not clearly defined. Since this application is currently running on the author's laptop, the results are selective and illustrative. The Search Results will appear in the same screen, specific to the file and search parameters, and will be determined by the core API code. The results will include FileName, Type (of media), Size (of the file), Bandwidth (of the remote host), Quality (determined by the number of replications on the network), Location (IP of remote host), and Keyword (the most useful piece of metadata at this point). The user can then select the file and click on the Download button to download the file. Once the file has started downloading, a small 'down arrow' will appear to the left of the FileName to distinguish it from other files. It will also appear in Monitor view. Running the cursor over any of the column fields will display the full metadata and allow the user to determine if the file meets his or her search requirements. The screenshot below is what the user might see.



Search: hedge funds

Include Metadata

□ Title

☑ Subject: hedge funds

□ Author

□ Category

☑ Keywords: risk, illiquidity

Options

□ Word
□ PDF
□ PowerPoint
□ Excel
□ PS
□ Image
□ Other
☑ All

FileName	Type	Size	Bandwidth	Quality	Location	Keyword
hedge_fu...	PDF	640kb	200	Poor	18.54.12.203	Hedge...

hedge_fund_illiquidity.pdf

Title: An Econometric Model of Serial Correlation and Illiquidity In Hedge Fund Returns

Subject: Hedge Fund Returns: Serial Correlation and Illiquidity

Author: Mila Getmansky, Andrew W. Lo, and Igor Makarov

Category: Finance, Computation Finance, Financial Engineering

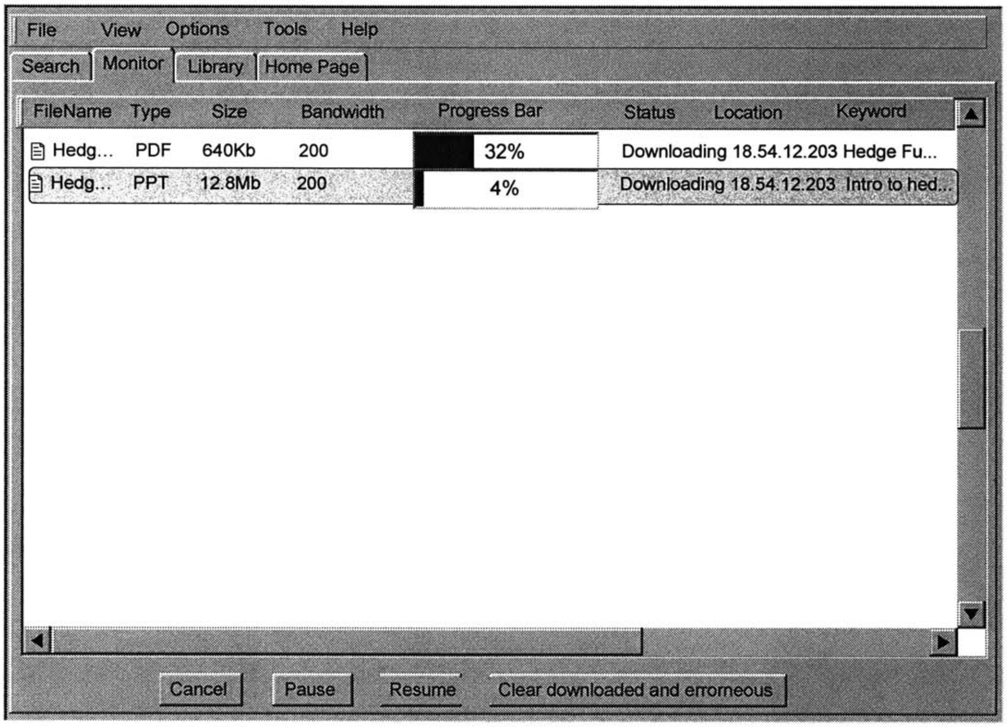
Keyword: Hedge Funds; Serial Correlation; Performance Smoothing; Liquidity; Market Efficiency

Comments: returns to hedge funds and other alternative investments; sources of serial correlation; quantifying illiquidity exposure; JEL Classification G12; Journal of Financial Economics

Figure 17

Monitor View. The Monitor view shows the progress of downloading files. This includes parameters such as Filename, Type, Size, Bandwidth, Progress Bar, Status, Location and

Keywords. Please note that fields “Progress Bar” and “Status” are additional to the parameters listed in the Search view. The Progress Bar shows the percentage of the file already downloaded. Status shows the current status of the file (like “downloading”, “downloading finished”, “Paused”). This view has buttons at the bottom of the frame for “Cancel”, “Pause”, “Resume”, “Clear downloaded and erroneous.” These buttons are fairly self-explanatory and KaZaA and Gnutella users will find this view familiar. Below is what the user will see when downloading files from the “Hedge Fund” example.

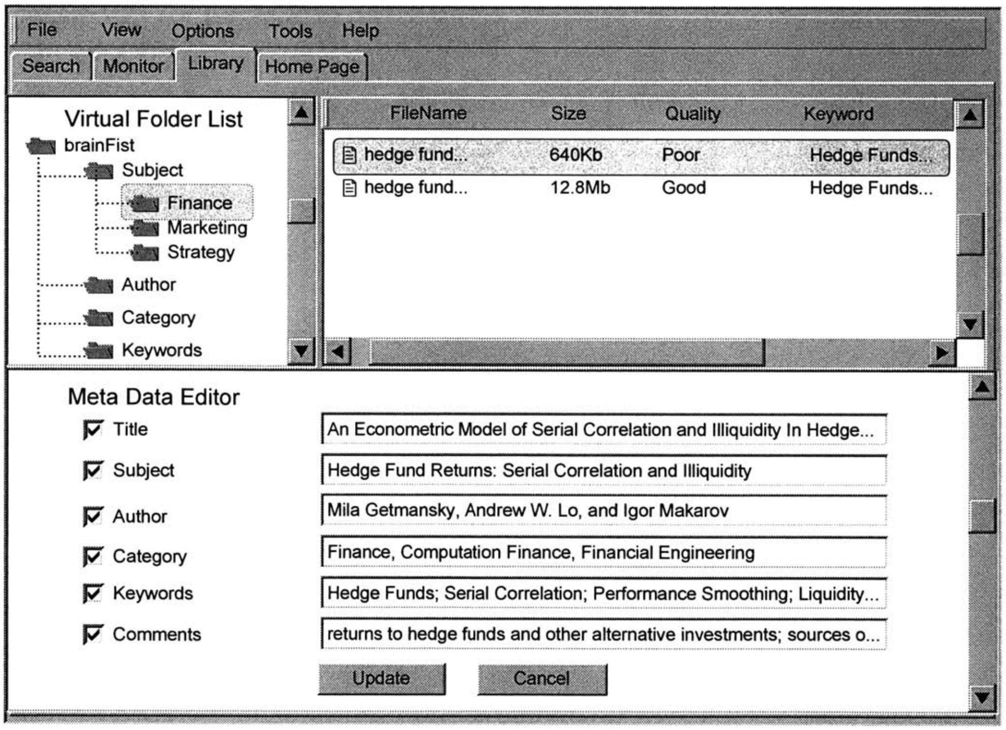


FileName	Type	Size	Bandwidth	Progress Bar	Status	Location	Keyword
Hedg...	PDF	640Kb	200	32%	Downloading	18.54.12.203	Hedge Fu...
Hedg...	PPT	12.8Mb	200	4%	Downloading	18.54.12.203	Intro to hed...

Figure 18

Library View. The Library view has three main frames. The top left frame shows the “Virtual Folder List.” The Virtual Folders and semantic representations of the metadata contained in the files the user has designated to be managed by the BrainFist client. Virtual folders are not physical locations on the user’s hard drive but rather metadata categories that help translate document into content. For example, in the ‘Hedge Fund’ case, the user has already created

Virtual Folders to represent the Subjects: Finance, Marketing and Strategy. The user can then 'drag and drop' the file into a respective virtual subject folder – changing its metadata. Additionally, the user can manually change the metadata for a given file by using the 'Meta Data Editor' in the lower frame. The user can also customize virtual folders for the major metadata categories – Subject, Author, Category and Keywords. This process is covered more closely in the Menus section under Options. This provides an expedient way for the user to manage the metadata of his or her files, increasing reuse and decreasing unnecessary content creation. Furthermore, the user does not have to change the physical addresses of his or her files.



FileName	Size	Quality	Keyword
hedge fund...	640Kb	Poor	Hedge Funds...
hedge fund...	12.8Mb	Good	Hedge Funds...

Meta Data Editor

☑ Title: An Econometric Model of Serial Correlation and Illiquidity In Hedge...

☑ Subject: Hedge Fund Returns: Serial Correlation and Illiquidity

☑ Author: Mila Getmansky, Andrew W. Lo, and Igor Makarov

☑ Category: Finance, Computation Finance, Financial Engineering

☑ Keywords: Hedge Funds; Serial Correlation; Performance Smoothing; Liquidity...

☑ Comments: returns to hedge funds and other alternative investments; sources o...

Figure 19

Home Page View. This view is merely a direct link to the BrainFist web site, which is currently under construction and will not be released until the beta is ready for distribution. Every user who downloads the BrainFist client will be given a username and password to contribute to the

BrainFist blog site. Furthermore, the user can submit questions and comments to the author concerning the BrainFist client, its features and any grievances.

4.2.3 Menus

There are five popup menus: File, Views, Options, Tools and Help. Although these are not fully developed yet, these menus are the placeholders for future product features. Currently, the features in development are the Tools that allow the user to customize the metadata categories.

File Menu. The File menu has three items: Connect, Disconnect and Exit. Connect initiates a connection to the Gnutella network. Disconnect breaks the connection. Exit closes the application's main window but the client will still run in background with the BrainFist Icon in the task bar. Right clicking on the status bar will initiate a popup menu with the option of closing or maximizing the client.

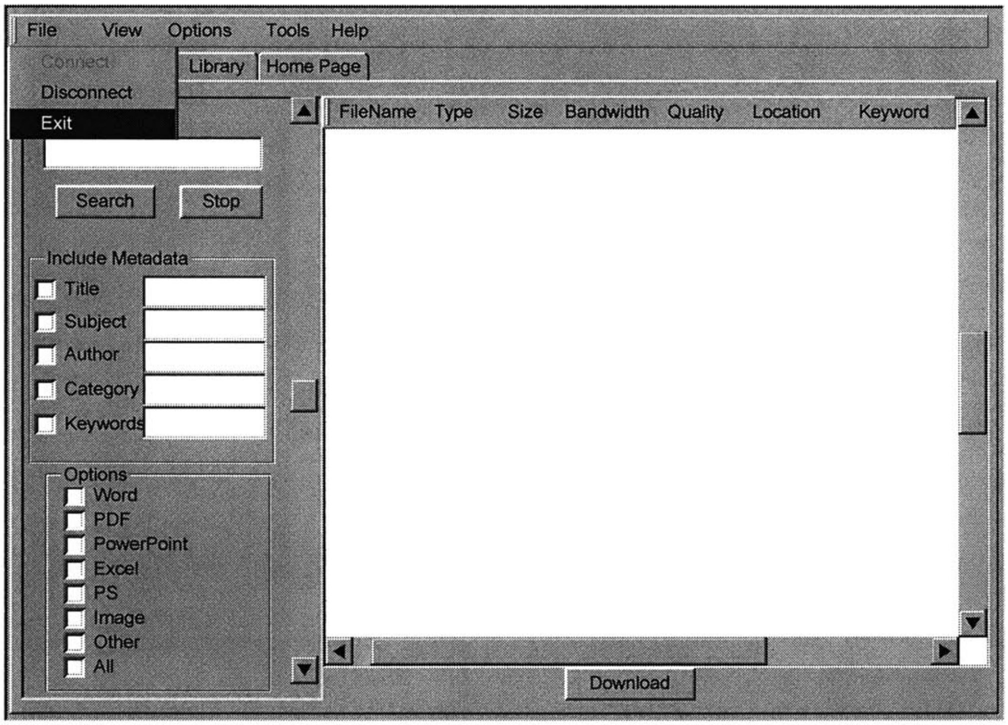


Figure 20

Views. Views has four menu items: Search, Monitor, Library and Home. Each one activates the corresponding View.

Options. Currently this menu only has two items: Share and Settings. The first allows the user to select which physical folders to share by using a Windows Explorer menu and check boxes. More items will be added as features are added to future versions. Settings allow the user to self designate as a Superpeer. This item will be discussed more in depth in the next chapter. Additionally, Settings allow the user to select different 'Skins' or application looks as they are developed.

Tools. This menu currently has the Virtual Folder creation tool as well as placeholders for known future features: Security, Plug-ins, and Buddy List. The Virtual Folder item currently contains Metadata parameters: Subject, Author, Category and Keywords. The user can create up to 99 of each parameter. The future features will be discussed in the last chapter.

Help. This menu contains, initially, two item: About, which discusses the version and copyright and FAQ which will cover a continuously updated list of questions and answers.

Chapter 5

Optimization

Superpeers are generally considered to be a promising approach to search optimization in distributed networks. This chapter analyzes the performance of several superpeer architectures, comparing the different architectures and presents a model for the BrainFist superpeer architecture. Finally, the author illustrates how a customized P2P content management solution can be modified depending on the number of users in a given organization. Originally, it was possible to query every host in the Gnutella network in search of some special file. Then a query was able to reach about half of all hosts – then less than half. Now the same query would be lucky to reach a single-digit percentage of all network hosts, which makes a search for the rare file a improbable exercise. The most recent approach to this problem is to use the superpeers – the host that acts as the concentrator, holding the content indices and routing the queries on behalf of the 'leaf hosts'. In this scenario, the ad hoc peer-to-peer network uses a default configuration of 6 superpeer connections, 75 leaf node connections, and a TTL setting of three, creating a semi-structured, two-level topology [18].

5.1 Superpeer Architecture

Every superpeer has an index of all the content stored on its leaf nodes, so instead of delivering the query message to all the network nodes, it has to be delivered only to the superpeers, which can perform the local searches in their index databases without actually querying their leaf nodes. It has been noted long ago that the traffic in the Gnutella network tends to saturate the dial-up modem links, thus limiting the search request reach in the network. So the natural countermeasure was to isolate the modem hosts from the query routing, selecting a subset of well-connected hosts called 'superpeers' and making them bear the routing load on behalf of their 'leaf nodes' [18].

5.1.1 Redundant Superpeer Cluster

Another approach to the search improvement in the superpeer network is to introduce the redundancy into the superpeer architecture. Osokine introduced the 'k-redundancy' as a special variant of superpeer design and showed that this design can dramatically increase search effectiveness. The idea behind the k-redundancy is to group k superpeers together and upload the same leaf indices on all of them. A leaf node sends a query to all the superpeers it's connected to, or to a random one. These superpeers or superpeer retransmit this query to other superpeers. In the Gnutella network case it means that every superpeer randomly chooses its connection points into the network, regardless of the choices made by other superpeers - including the superpeers in the same k-redundant cluster. For the finite network assumption, such as BrainFist, the probability of the same query arriving from the different superpeers to the same superpeer or even into the same k-redundant superpeer cluster is significant [18].

5.1.2 Superpeer Network Problem

Although the superpeer architecture has the potential to deliver a significant search performance increase, the obvious drawback is in coordinating the resources of an efficient superpeer architecture. If the network is fully decentralized, every host has to make its own decisions about becoming a superpeer or not, which can significantly affect the network performance. If, for example, too few hosts decide to become the superpeers, the leaf nodes might not have enough redundant connections. Contrarily, if there are too many superpeers, they won't be able to fully utilize their storage capacity. The answer to this dilemma is the customization of solutions on an organizational basis where resources are coordinated by peers, not necessarily by a centralized system. This requires a solution with specific scope, not a universal and infinite network [18].

5.2 Mutual Index Caching

Another approach to the peer-to-peer search optimization is the mutual index caching. In this approach the peer-to-peer network hosts store not only the indices of their own data, but also the indices of the data located on other hosts. Doing so requires every peer-to-peer host to set aside some space for these indices and to somehow obtain them from the other network hosts, effectively becoming a superpeer. Specifically, the indices cache not the content but the content's metadata. The caching strategy enables the widest possible search query reach [18].

5.3 Model Parameters

To illustrate the potential and fault tolerance of a Gnutella based client, this report uses Jordan Ritter's model [19], which predicted the Gnutella network crash. Although the author attempted to replicate Ritter's results exactly, some changes had to be made to maintain the integrity of the model. Still, the author feels confident that Ritter's model, for the purpose of this thesis, provides a sufficient analysis of the network effects of the BrainFist client.

Network Optimization Objective: Maximize the number of connected users – P – and minimize the bandwidth consumption { h (n, t, s) & i (n, t, s) }.

- P – P is the number of users connected to the GnutellaNet. For the initial model, we will assume a range of 150 to 50,000 users.
- N – N is the number of connections held open to other servents (client/server nodes) in the network. The default configuration of the original Gnutella client is 4 connections.
- T – T is TTL, or Time To Live, on packets. TTL's are used to age a packet and ensure that it is relayed a finite number of times before being discarded.
- B – B is the amount of available bandwidth, or alternatively, the maximum capacity of the network transport. It is determined by the two functions:
- h (n, t, s) – h (n, t, s) is a function describing the maximum amount of bandwidth generated by relaying a transmission of s bytes given any n and t. Generation is defined as the formulation and outbound delivery of data.

  $$h(n, t, s) = n * s + f(n, 1, t-1) * (n-1) * s$$

- i (n, t, s) – i (n, t, s) is a function describing the maximum amount of bandwidth incurred by relaying transmission of s bytes given any n and t. Incurrence is defined as the reception or transmission of data across a unique connection to a network.

  $$i(n, t, s) = (1 + f(n, 1, t-1)) * n * s + f(n, t, t) * s$$

- f (n, x, y) – f (n, x, y) is function describing the maximum number of reachable users that are at least x hops away, but no more than y hops away.

  $$f(n, x, y) = \mathrm{SUM}\,[\,((n-1)\,\hat{}\,(t-1)) * n\,,\ t = x \rightarrow y\,]$$

- g (n, t) – g (n, t) is a function describing the maximum number of reachable users for any given n and t.

  $$g(n, t) = f(n, 1, t)$$

- a – a is mean percentage of users who typically share content. For our client we will assume a normal distribution with $\mu = 50\%$ and $\sigma = 25\%$.
- b – b is a mean percentage of users who typically have responses to search queries. For our client we will assume a normal distribution with $\mu = 25\%$ and $\sigma = 12.5\%$.

- r – r is a mean number of search responses the typical respondent offers. For our client we will assume a normal distribution with $\mu = 5$ and $\sigma = 2.5$.
- l – l is a mean length of search responses the typical respondent offers. For our client we will assume a normal distribution with $\mu = 60$ and $\sigma = 20$.
- R – R is a function representing the Response Factor, a constant value that describes the product of the percentage of users responding and the amount of data generated by each user.

$$R = (a * b) * (88 + r * (10 + l)) \sim \text{initial value} = 55$$

- $j (n, T, R)$ – $j (n, T, R)$ is a function describing the amount of data generated in response to a search query by tier T, given any n and Response Factor R.

$$j (n, T, R) = f (n, T, T) * R$$

- $k (n, t, R)$ – $k (n, t, R)$ is a function describing the maximum amount of bandwidth generated in response to a search query, including relayed data, given any n and t and Response Factor R.

$$k (n, t, R) = \text{SUM} [j (n, T, R) * T, T = 1 \rightarrow t]$$

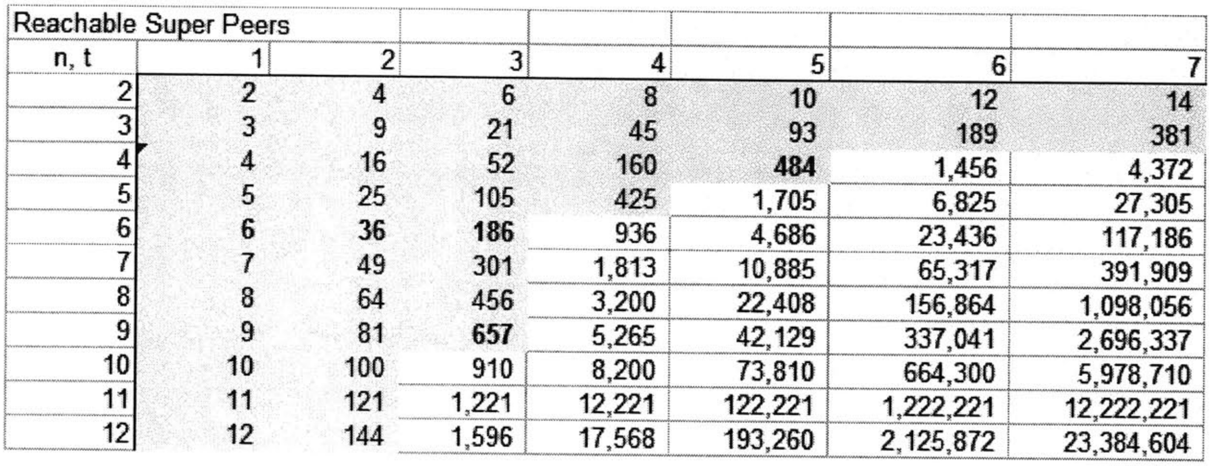
5.3.1 Assumptions

The following are a list of assumption necessary to start the model. First, the packet sizes for Search and Search Response are based on Ritter's model [19].

	Search Query Packet Makeup	Search Response Packet Makeup
IP header	20 bytes	20 bytes
TCP header	20 bytes	20 bytes
Gnutella header	23 bytes	23 bytes
Number of hits	N/A	1 byte
Port	N/A	1 byte
IP Address	N/A	4 bytes
Speed	1 byte	3 bytes
Result Set	N/A	r * (8 + l + 2) bytes
Search String	18 + 1 bytes (trailing null)	N/A
Servent Identifier	N/A	16 bytes
Total	83 bytes	88 + r * (10 + l) bytes

Table 1

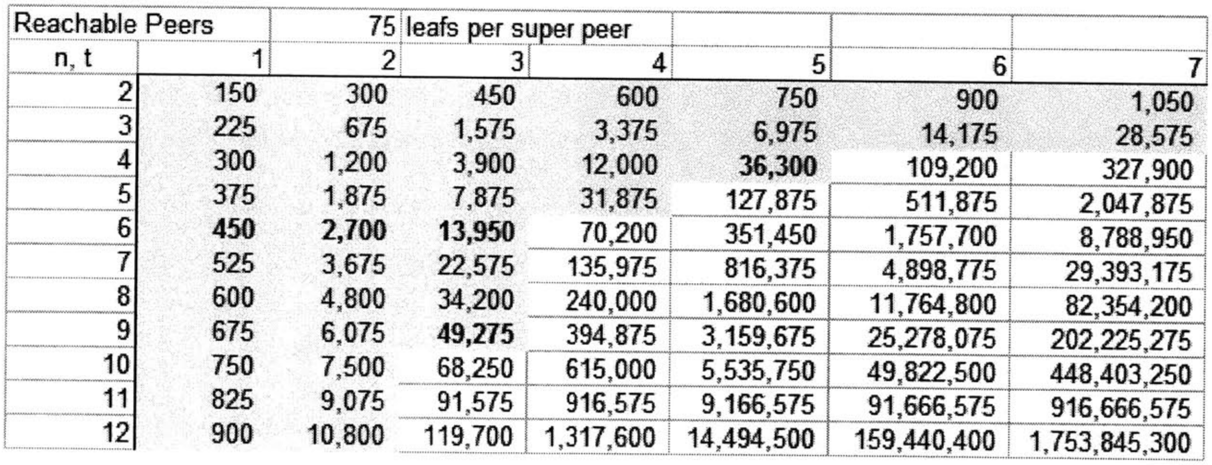
Second, a geometric model is developed to determine, first, the number of reachable Super Peers based on the configuration of nodes (n) and TTL (t).



Reachable Super Peers							
n, t	1	2	3	4	5	6	7
2	2	4	6	8	10	12	14
3	3	9	21	45	93	189	381
4	4	16	52	160	**484**	1,456	4,372
5	5	25	105	425	1,705	6,825	27,305
6	6	**36**	**186**	936	4,686	23,436	117,186
7	7	49	301	1,813	10,885	65,317	391,909
8	8	64	456	3,200	22,408	156,864	1,098,056
9	9	81	**657**	5,265	42,129	337,041	2,696,337
10	10	100	910	8,200	73,810	664,300	5,978,710
11	11	121	1,221	12,221	122,221	1,222,221	12,222,221
12	12	144	1,596	17,568	193,260	2,125,872	23,384,604

Table 2

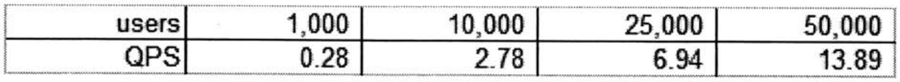
Next, the number of reachable peers is determined using the default value of 75 'leafs' or individual Nodes per Super Peer. The highlighted areas are the relevant parameters for the objective number of users (150 to 50,000).



Reachable Peers		75	leafs per super peer				
n, t	1	2	3	4	5	6	7
2	150	300	450	600	750	900	1,050
3	225	675	1,575	3,375	6,975	14,175	28,575
4	300	1,200	3,900	12,000	**36,300**	109,200	327,900
5	375	1,875	7,875	31,875	127,875	511,875	2,047,875
6	**450**	**2,700**	**13,950**	70,200	351,450	1,757,700	8,788,950
7	525	3,675	22,575	135,975	816,375	4,898,775	29,393,175
8	600	4,800	34,200	240,000	1,680,600	11,764,800	82,354,200
9	675	6,075	**49,275**	394,875	3,159,675	25,278,075	202,225,275
10	750	7,500	68,250	615,000	5,535,750	49,822,500	448,403,250
11	825	9,075	91,575	916,575	9,166,575	91,666,575	916,666,575
12	900	10,800	119,700	1,317,600	14,494,500	159,440,400	1,753,845,300

Table 3

Additionally, assume 1 query per user per hour, a high estimate given BrainFist will primarily be a work augmenting tool, which is 0.00028 queries per second (QPS). Using this ratio, it is easy to calculate QPS for any given number or nodes or users on the network.

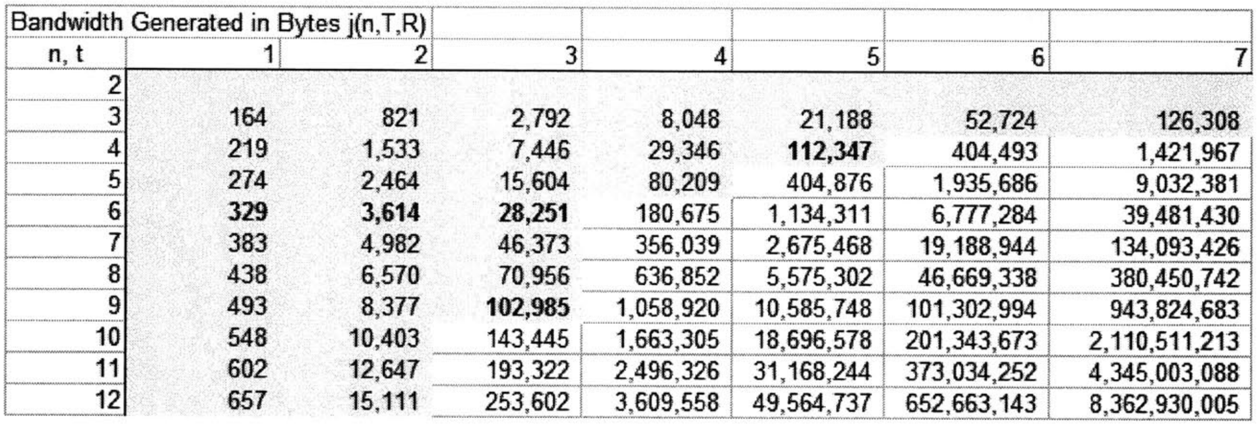


users	1,000	10,000	25,000	50,000
QPS	0.28	2.78	6.94	13.89

Table 4

5.3.2 Dynamic Model

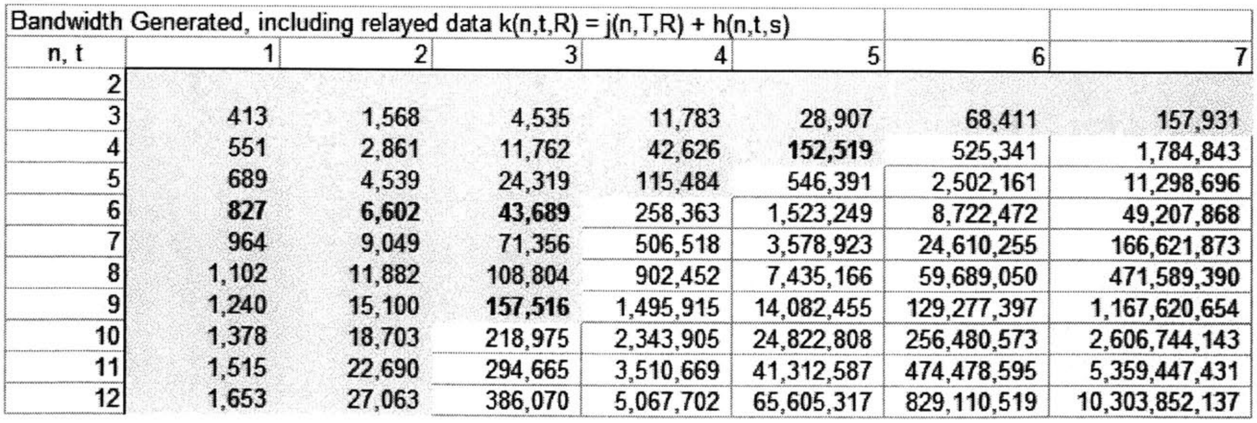
Based on the assumptions above, it becomes clear that for the objective number of users (150 to 50,0000) the bandwidth generated is not as high as predicted in Ritter's model [19]. This is because of the Super-Peer architecture enabled in the Gnutella network. Fist, examine the data generated in response to a search query.



Bandwidth Generated in Bytes j(n,T,R)							
n, t	1	2	3	4	5	6	7
2							
3	164	821	2,792	8,048	21,188	52,724	126,308
4	219	1,533	7,446	29,346	**112,347**	404,493	1,421,967
5	274	2,464	15,604	80,209	404,876	1,935,686	9,032,381
6	**329**	**3,614**	**28,251**	180,675	1,134,311	6,777,284	39,481,430
7	383	4,982	46,373	356,039	2,675,468	19,188,944	134,093,426
8	438	6,570	70,956	636,852	5,575,302	46,669,338	380,450,742
9	493	8,377	**102,985**	1,058,920	10,585,748	101,302,994	943,824,683
10	548	10,403	143,445	1,663,305	18,696,578	201,343,673	2,110,511,213
11	602	12,647	193,322	2,496,326	31,168,244	373,034,252	4,345,003,088
12	657	15,111	253,602	3,609,558	49,564,737	652,663,143	8,362,930,005

Table 5

Next, examine the bandwidth generated in response to a search query.



Bandwidth Generated, including relayed data k(n,t,R) = j(n,T,R) + h(n,t,s)							
n, t	1	2	3	4	5	6	7
2							
3	413	1,568	4,535	11,783	28,907	68,411	157,931
4	551	2,861	11,762	42,626	**152,519**	525,341	1,784,843
5	689	4,539	24,319	115,484	546,391	2,502,161	11,298,696
6	**827**	**6,602**	**43,689**	258,363	1,523,249	8,722,472	49,207,868
7	964	9,049	71,356	506,518	3,578,923	24,610,255	166,621,873
8	1,102	11,882	108,804	902,452	7,435,166	59,689,050	471,589,390
9	1,240	15,100	**157,516**	1,495,915	14,082,455	129,277,397	1,167,620,654
10	1,378	18,703	218,975	2,343,905	24,822,808	256,480,573	2,606,744,143
11	1,515	22,690	294,665	3,510,669	41,312,587	474,478,595	5,359,447,431
12	1,653	27,063	386,070	5,067,702	65,605,317	829,110,519	10,303,852,137

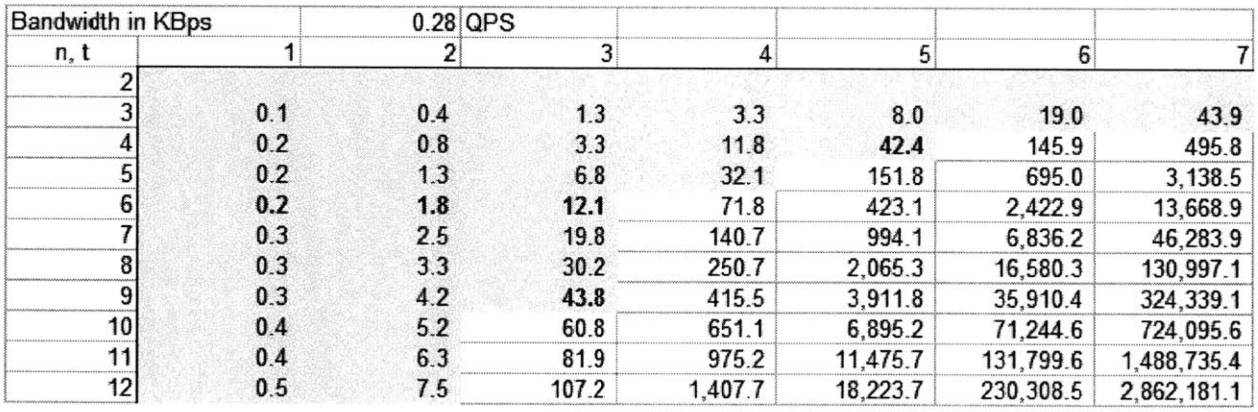
Table 6

5.3.3 Optimal Parameters

Based on a simple set of assumptions, the model can determine QPS for an expected number of users. Furthermore, it becomes possible to anticipate network usage and customize the BrainFist client. For example, the client for a given organization can be set with the following parameters:

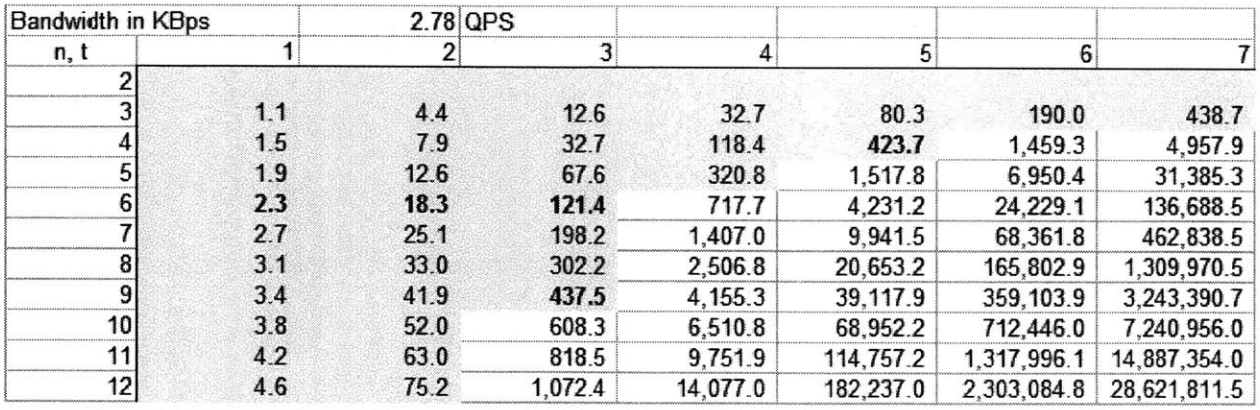
- Supernode – 6
- Leafs – 75
- Nodes – 6
- TTL – 3

For 1,000 Users the model predicts a network rate of 12.1 KBps; for 10,000 Users – 121.4 KBps; and, in this particular scenario, for 50,000 Users – 606.8 KBps. All of these rates are reasonable for a network of high speed internet connections. Below, are the three examples mentioned. If network usage appears high in some cases then the setting for the client (Supernodes, Leafs, Nodes, TTL) can be adjusted to compensate.



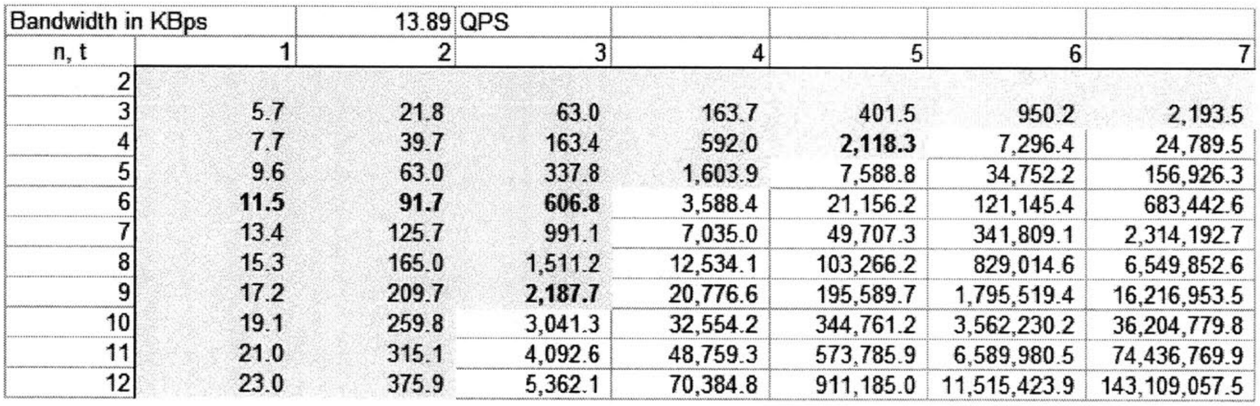
Bandwidth in KBps		0.28	QPS				
n, t	1	2	3	4	5	6	7
2							
3	0.1	0.4	1.3	3.3	8.0	19.0	43.9
4	0.2	0.8	3.3	11.8	**42.4**	145.9	495.8
5	0.2	1.3	6.8	32.1	151.8	695.0	3,138.5
6	**0.2**	**1.8**	**12.1**	71.8	423.1	2,422.9	13,668.9
7	0.3	2.5	19.8	140.7	994.1	6,836.2	46,283.9
8	0.3	3.3	30.2	250.7	2,065.3	16,580.3	130,997.1
9	0.3	4.2	**43.8**	415.5	3,911.8	35,910.4	324,339.1
10	0.4	5.2	60.8	651.1	6,895.2	71,244.6	724,095.6
11	0.4	6.3	81.9	975.2	11,475.7	131,799.6	1,488,735.4
12	0.5	7.5	107.2	1,407.7	18,223.7	230,308.5	2,862,181.1

Table 7



Bandwidth in KBps		2.78	QPS				
n, t	1	2	3	4	5	6	7
2							
3	1.1	4.4	12.6	32.7	80.3	190.0	438.7
4	1.5	7.9	32.7	118.4	**423.7**	1,459.3	4,957.9
5	1.9	12.6	67.6	320.8	1,517.8	6,950.4	31,385.3
6	**2.3**	**18.3**	**121.4**	717.7	4,231.2	24,229.1	136,688.5
7	2.7	25.1	198.2	1,407.0	9,941.5	68,361.8	462,838.5
8	3.1	33.0	302.2	2,506.8	20,653.2	165,802.9	1,309,970.5
9	3.4	41.9	**437.5**	4,155.3	39,117.9	359,103.9	3,243,390.7
10	3.8	52.0	608.3	6,510.8	68,952.2	712,446.0	7,240,956.0
11	4.2	63.0	818.5	9,751.9	114,757.2	1,317,996.1	14,887,354.0
12	4.6	75.2	1,072.4	14,077.0	182,237.0	2,303,084.8	28,621,811.5

Table 8



Bandwidth in KBps		13.89	QPS				
n, t	1	2	3	4	5	6	7
2							
3	5.7	21.8	63.0	163.7	401.5	950.2	2,193.5
4	7.7	39.7	163.4	592.0	**2,118.3**	7,296.4	24,789.5
5	9.6	63.0	337.8	1,603.9	7,588.8	34,752.2	156,926.3
6	**11.5**	**91.7**	**606.8**	3,588.4	21,156.2	121,145.4	683,442.6
7	13.4	125.7	991.1	7,035.0	49,707.3	341,809.1	2,314,192.7
8	15.3	165.0	1,511.2	12,534.1	103,266.2	829,014.6	6,549,852.6
9	17.2	209.7	**2,187.7**	20,776.6	195,589.7	1,795,519.4	16,216,953.5
10	19.1	259.8	3,041.3	32,554.2	344,761.2	3,562,230.2	36,204,779.8
11	21.0	315.1	4,092.6	48,759.3	573,785.9	6,589,980.5	74,436,769.9
12	23.0	375.9	5,362.1	70,384.8	911,185.0	11,515,423.9	143,109,057.5

Table 9

5.4 Implications

In the past three years the Gnutella network grew by several orders of magnitude with tens of millions of users and counting. Ritter predicted the 'Napster meltdown' but this failed to materialize. The key reason for near meltdown in 2000 was that the early Gnutella network had no built-in flow control in the message transfer mechanisms, and the only two mechanisms to stop the query propagation were the request TTL and the loop detection. Once the network size exceeded 5,000 hosts, which happened in a few hours, the network size was still too low for TTL limit to have any noticeable effect, but high enough for the traffic sent to every host to exceed the average client's modem incoming bandwidth. These early Gnutella servents could do several things when the send() calls failed: Some servents just dropped the connection. There was actually nothing wrong with this connection, it was just overloaded, but when it was closed, all the query replies due to return over this link had no path back to the query source and were dropped, too. In the meantime both hosts established the new connections to some other hosts to replace the closed ones. These connections were promptly overloaded, and the process was repeated. This caused a Gnutella-wide network 'flapping', dramatically reducing the number of returned results - not that the queries did not reach the remote hosts, but the results could not find their way back to the requestor [18].

It was clear that unless some adaptive flow control mechanism would be found that would allow every host to carry its share of network load without overloading its physical link, the network would stay unusable. But it took the Gnutella community several months to figure out how exactly to achieve that. The first flow-controlled servents started to appear on the network around December, 2000, and by the end of January, 2001 most of the big vendors had the flow control implemented. It was called "rudimentary flow control" and mostly involved dropping queries when the link was getting overloaded. That simple measure helped to bring the Gnutella network back from the July-December 2000 'Napster meltdown.' This whole subject received almost no attention outside the tight circle of Gnutella developers. Surprisingly, few people know that there is a flow control in Gnutella, and that it is the flow control, not the TTL that stops the query propagation and avoids the physical link overload. Sophisticated flow control proposals limit the query transmission when the host links are saturated, doing this adaptively and in real time according to the host bandwidth and current load [18].

Chapter 6

Critical Issues

Two issues are critically important in the continued development of the BrainFist client. A Content Management System must address the need for a network to assign Roles to its users, each Role granting a different level of Authorization for each subset of users. Secondly, and most critically, is the need for Security. The latter issue becomes especially difficult given that the BrainFist client is based on P2P architecture. P2P presents its own security concerns, not the least of which is network integrity. This chapter will address both of these issues and propose some measures that will become very relevant in future product development. The author further acknowledges that the BrainFist client is not a tenable and realistic solution if these issues are not addressed in the next developmental stage.

6.1 Roles and Authorization

In order to keep users from impeding the productivity of fellow users, it becomes absolutely necessary to assign authorizations. Organizations create new roles for employees: system administrators, editors, etc. Additionally, the production of knowledge (content) requires different users to assume different responsibilities. Organizations need to develop access authorizations and process workflows for managing these responsibilities. Employees should be able to change only the information that is specific to their content or responsibilities. Finally, organizations must protect confidential data from unauthorized access. The administrator must be able to map the company's reporting structure to the workflows in their content management system to ensure content accuracy and security.

BrainFist can use XML tags to assign different level of authorization for specified roles. These roles coordinate management of content within the system. There are four primary roles: Subscriber, Author, Editor, and Administrator. Each of these roles has certain rights that enable users to complete content updates. An example would be a managing partner (Administrator) in a consulting firm initiating a white paper on a given topic. While certain users, associate consultants, have authority to create content (Author), other users, such as project managers, may have authority to edit and revise existing content (Editor). Users not directly involved in the project may view but not alter the content with respect to the project (Subscriber). The managing partner (Administrator) has final authority for publication.

6.1.1 Principle Responsibilities

The user roles in the BrainFist system can be customized but the default configuration will allow four levels of authorization by designating four distinct roles. These roles differ according to their principle responsibilities: administrative and editorial. This is the default configuration but Roles should be modeled after organizational duties and structure. The following roles are listed from least to greatest level of authorization with each subsequent level inclusive of the previous level (e.g. Author has all the rights of Subscriber while Editor has all the rights of Subscriber and Author).

- **Level 1: Subscriber.** Subscribers can be internal (employees) or external (guest) users, able to browse folders and view unrestricted content. This is the most basic role in the authorization hierarchy. Subscribers can browse but not edit content.
- **Level 2: Author.** Authors can create, edit and submit content for approvals as well determine metadata for documents. Authors create and submit content for approval to an editor. Although authors set metadata properties themselves, Editor and/or Administrators can override these settings.
- **Level 3: Editor.** Editors approve pages for publication, verify metadata and provide resource material for Authors. Editors ensure content is accurate and can approve or decline submissions. Additionally, Editors are responsible for 'stocking' folders with resources such as media files, documents (doc, ppt, xls, pdf, etc.). This is one of the most critical roles in the initial phase of deploying the BrainFist client within an organization.
- **Level 4: Administrator.** Administrators are solely responsible for creating folder hierarchies (subject, keywords, authors, etc.). Furthermore, Administrators are responsible for creating and managing user roles. Finally, Administrators have access to everything in the BrainFist network.

6.1.2 Access Control

How these roles are reliably assigned is the subject of various research projects. Most notably is Guillermo Navarro's work at the Combinatronics and Digital Communication at the Depart of the Computer Science at the Universitat Autonoma de Barcelona [21]. Significant headway has been made in standardizing access control in distributed systems. Most relevant to a system like

BrainFist, are efforts in Trust Management, X.509 Attribute Certificates, the Secure Assertion Markup Language (SAML), the eXtensible Access Control Marup Language (XACML) and the eXtensible rights Markup Language (XrML). The last three are specifically designed to authenticate XML tags and facilitate the assignment of roles. The following diagram illustrates logical assignment and verification of users.

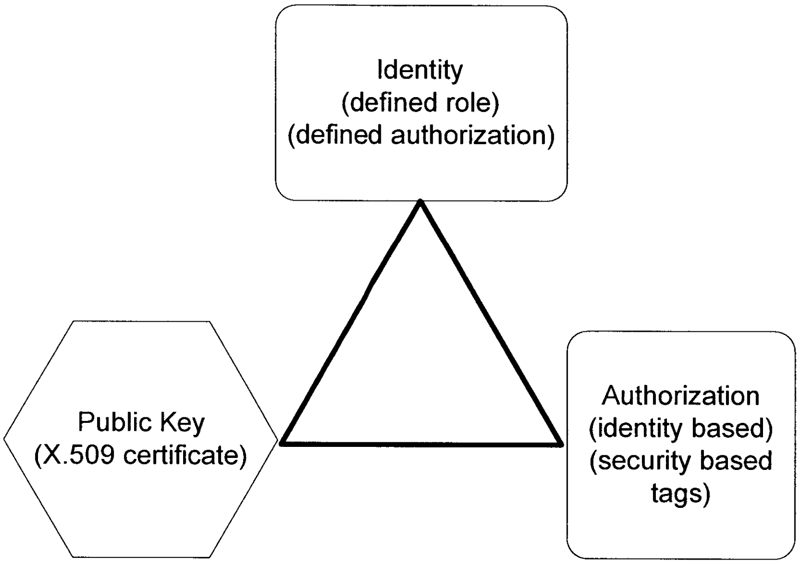


Figure 21

6.2 Security

Security is an essential component of any computer system, and it is especially relevant for P2P systems. Security, in the P2P landscape, is the technological and procedural methods for establishing trust, trust in other users and trust with software vendors. Many proposals are already being studied and people are acknowledging that security is an area P2P must address, if it is to be accepted by consumers. Most recently, the U.S. House of Representatives passed the broad Government Network Security Act of 2003 [22]. This legislation requires federal agencies to protect their networks from P2P file sharing. This includes, but is not limited to, clearly defining the security measures in P2P technology development for federal government use. The security measures are not specified except to state that measures taken can be either technological or procedural in nature. The salient point here is the P2P security is not an internal mandate of programmers and technologists but an organizational issue for all potential knowledge workers.

6.2.1 Threats

P2P networking is open to various forms of attack, break-in, espionage, and malicious mischief. P2P, specifically, does not bring any novel threats to the network, but does bring familiar ones such as worms and virus attacks. Unfortunately, P2P networking circumvents enterprise security by providing decentralized security administration, decentralized shared data storage, and a way to circumvent critical perimeter defenses such as firewalls and NAT devices. Additionally, P2P software users can easily configure their application to expose confidential information for personal gain. P2P file-sharing applications can result in a loss of control over what data is shared outside the organization [23].

P2P applications get around most security architectures in the same way that a Trojan horse does. The P2P application is installed on a “trusted device” that is allowed to communicate through the corporate firewall with other P2P users. Once the connection is made from the trusted device to the external Internet attackers can gain remote access to the trusted device for the purpose of stealing confidential corporate data, launching a Denial of Service attack or simply gaining control of network resources. Internally, there must be a feasible method to add/delete users to/from the network without increasing vulnerability. The system is under the most threat from users and former users who know the ins and outs of the system.

Furthermore, there will always be malicious users who are intent on gaining clandestine access to corporate networks [23].

6.2.2 Mechanisms and Protocols

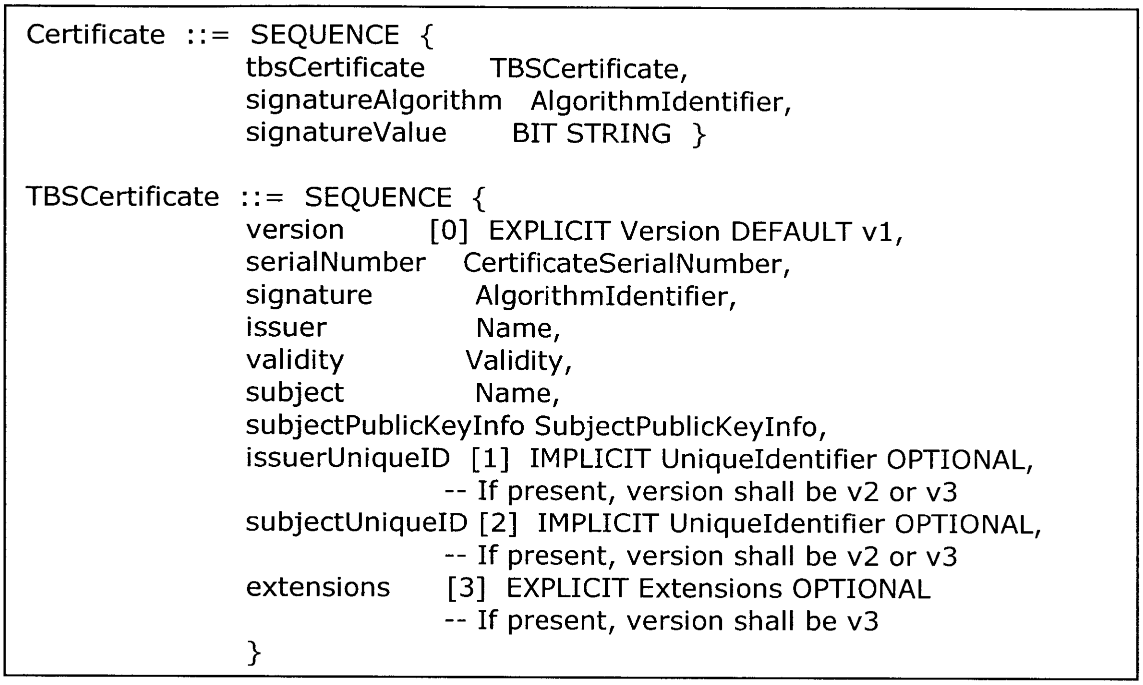
All security mechanisms deployed today are based on either symmetric/secret key or asymmetric/public key cryptography, or sometimes a combination of the two. Secret key techniques are based on the fact that the sender and recipient share a secret, which is used for various cryptographic operations, such as encryption and decryption of messages and the creation and verification of message authentication data. This secret key must be exchanged in a separate out of bound procedure prior to the intended communication. Public Key Techniques are based on the use of asymmetric key pairs. Usually each user is in possession of just one key pair. One of the pair is made publicly available, while the other is kept private. Because one is available there is no need for an out of band key exchange, however there is a need for an infrastructure to distribute the public key authentically. Besides being one way functions, cryptographic public keys are also trapdoor functions and the inverse can be computed if the private key is known. Mechanisms for establishing strong, cryptographically verifiable identities are very important [23].

For protection of information transmitted over a P2P network, some P2P's employ the industry standard Secure Sockets Layer (SSL) protocol. This guarantees that files and events sent will arrive unmodified, and unseen, by anyone other than the intended recipient. Since both peers use SSL both sides automatically prove who they are to each other before any information is transferred over the network. Some Virtual Private Networks (VPNs) use IPSec technologies, an evolving framework of protocols that has become the standard for most vendors. IPSec is useful because it is compatible with most different VPN hardware and software, and is the most popular for networks with remote access clients. Security comes from the workstation's IP address or its X.509 certificate, establishing the user's identity and ensuring the integrity of the network and acting as the network layer protecting all the processed data packets. Public Key Infrastructure (PKI) is another industry standard but the full-featured X.509 PKI over SSL network backbone (authentication and transport encryption) is the established cryptographic standard for Internet e-commerce [23].

6.2.3 Cryptography

SSL, Pretty Good Privacy (PGP), and X.509 provide a ground level of existing standards that, with some modification, can be used with a decentralized P2P network. There are many implementations of standards such as SSLeay, OpenSSL, JSSE and Cryptix, but these were developed for client server architectures. For example, most SSL implementations do not support dynamic switching of keys and downloading of Control Runtime Languages (CRLs). Furthermore, SSL is point to point and will not stand alone in a dynamic and inconsistent P2P environment [24].

Although the thesis has touched on proposed encryption methods in **6.1.2**, the standard language of Internet cryptography is the Abstract Syntax Notation One (ASN.1), a formal language for abstractly describing messages to be exchanged between distributed computer systems. The following figure is an example of an ASN.1 markup.



```
Certificate ::= SEQUENCE {
                tbsCertificate      TBSCertificate,
                signatureAlgorithm  AlgorithmIdentifier,
                signatureValue      BIT STRING }

TBSCertificate ::= SEQUENCE {
                version        [0]  EXPLICIT Version DEFAULT v1,
                serialNumber   CertificateSerialNumber,
                signature      AlgorithmIdentifier,
                issuer         Name,
                validity       Validity,
                subject        Name,
                subjectPublicKeyInfo SubjectPublicKeyInfo,
                issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                               -- If present, version shall be v2 or v3
                subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                               -- If present, version shall be v2 or v3
                extensions      [3]  EXPLICIT Extensions OPTIONAL
                               -- If present, version shall be v3
                }
```

Figure 22

Common in many industries (financial services, shipping, etc.), the ASN.1 uses its own standard protocols such as the Distinguished Encoding Rules (DER). Along with X.509 certificates, it is possible to establish secure and trustworthy P2P networks with established standards. ASN.1 defines the structure of a certificate while DER defines the external representation [24].

Chapter 7

Conclusion

This thesis presents the design and optimization of BrainFist, a peer-to-peer content management system. Far from an end-to-end solution, BrainFist addresses the aspect of content management concerned with metadata. Through a simple and versatile interface, the user is allowed to manage files and the files' metadata without having to severely alter his or her workflow routine. Furthermore, BrainFist addresses the problem of the network strain created by P2P systems. BrainFist is a simple tool for one simple reason: BrainFist fosters innovation not in the hands of the developer but in the hands of the end user. By creating a P2P based CM solution that only addresses file management issues, the user or organization is left to decide how to best implement and utilize such a tool. This does not mean that BrainFist is inclusive and complete. This initial beta addresses the issue of file management through metadata. There are three other crucial aspects addressed in future development: security, collaboration, and workflow.

First, and definitely the most immediate, is the issue of security. BrainFist, as a custom solution for small organizations (150 to 50,000 users) must address the issue of network security and protection from intellectual property theft or infringement. Furthermore, there is an inherent capacity for the rapid spread of trouble and decentralization which can make auditing extremely difficult. The first tool available for addressing such issues is a reputation mechanism. When the user downloads and installs the BrainFist client, he or she is given a unique identifier created with time-date stamp and random number generator. This will serve to authenticate the user as a trustworthy node on the network. Additionally, it helps to identify the user as a reliable and productive contributor to the network. Additionally, Roles and Authorization protocols become critical in the assignment of responsibilities and powers. By developing a clearly defined system of user protocols, BrainFist users not only develop trust in the network but in their own ability to contribute and benefit from the network.

Collaboration is the buzz word that inspires IT consultants and venture capitalists alike – and also the most elusive when it comes to proof of concept. While there are some examples of mild successes, such as Groove, BrainFist does not propose a collaboration solution, it does propose tools for innovation that, hopefully, will be adopted by users [20]. First, with the unique identifier assigned to each BrainFist client, Instant Messaging (IM) becomes the next logical feature. WASTE is a tool for Internet Relay Chat (IRC) and chat that can run independently of the host P2P file-sharing network [25]. Simple additions to the BrainFist client will allow users to work together on documents without having to enable an alternate application such as MS Outlook or AOL IM.

Finally, the workflow process is the key to early adoption. The BrainFist client has, thus far, presented a CMS tool that does not require the user to significantly change his or her work habits. Development is already under way to create BrainFist plug-ins for most of the Microsoft Office products (Word, Outlook, Excel, PowerPoint, Visio, Project) as well Adobe Acrobat and LaTeX. This is an involved process but doable. The goal is to have a BrainFist icon in every major application so documents can be tagged and peers notified of content creation or updates.

BrainFist is a customizable tool based on an open source P2P network. Any organization (university, consultancy, law firm) can develop a custom solution for its content management endeavors without having to invest heavily in server based products or other costly systems. Most importantly, the scalable nature of BrainFist allow for continued development not envisioned by the author.

Bibliography

[1] BrainFist. http://brainfist.com/, 2004.

[2] David Barkai. *Peer-to-Peer Computing: Technologies for Sharing and Collaborating on the Net.* Hillsboro, 2002.

[3] *Entertainment Law Digest.* Volume 8, Issue 4, October 2003.

[4] Napster. http://www.napster.com/, 2003.

[5] Hari Balakrishnan, et al. *Chord: Scalable, Robust Peer-to-Peer Services.* MIT Laboratory for Computer Science, March 2002.

[6] Brendon J. Wilson. *JXTA.* Indianapolis, June 15, 2002.

[7] Gnutella. http://www.gnutella.com/, 2003.

[8] LimeWire. http://www.limewire.com/, 2003

[9] FastTrack. http://fasttrack.nu/, 2004.

[10] Ann Rockley, et al. *Managing Enterprise Content: A Unified Content Strategy.* Indianapolis, 2003.

[11] David Marco. *Building and Managing the Meta Data Repository: A Full Lifecycle Guide.* New York City, July 17, 2000.

[12] Mary Mosquera. "Law Firm Collaborates Globally," *InternetWeek.* October 22, 2001.

[13] 1ST Works Corporation. Press Release, August 14, 2001.

[14] MS Office InfoPath. http://office.microsoft.com/home/, 2004.

[15] Doug Bartholomew. "The Summer of Our Content," *CFO.* June 01, 2003.

[16] Accenture. http://accenture.com/, 2004.

[17] Christopher Rorhs. "LimeWire Design," August 20, 2001.

[18] S. Osokine. "Search Optimization in the Distributed Networks," October 15, 2002.

[19] Jordan Ritter. "Why Gnutella Can't Scale. No, Really," February 2001.

[20] Groove Networks. http://groove.net/, 2003.

[21] Guillermo Navarro. "Access Control and Mobile Agents," Combinatronics and Digital Communication, Department of the Computer Science, Universitat Autonoma de Barcelona, September, 2003.

[22] "Government Network Security Act of 2003," H.R. 3159, September 25, 2003.

[23] Declan Murphy, et al. "P2P Security," P2P Networks TCD 4BA2 Project, 2003.

[24] Jim Lowrey, "Secure Peer to Peer Communication," November, 2001.

[25] Lucas Gonze. "WASTE is P2P IM," http://www.oreillynet.com/pub/wlg/3253, May 31, 2003.