IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4685

Peer-to-Peer Corporate Resource Sharing and Distribution with Mesh

Ramesh Subramanian
Quinnipiac University – School of Business
275 Mount Carmel Avenue, Hamden, CT 06518
P: 203-582-5276, F: 203-582-8664
ramesh.subramanian@quinnipiac.edu

Brian Goodman
IBM – Advanced Internet Technology
150 Kettletown Road, Mail Drop 121
Southbury, CT 06488
bgoodman@us.ibm.com

INTRODUCTION

Currently, much of the corporate data and content within "global" organizations are distributed by replicating and distributing such data and content using centralized content repositories. That is, the data is globally distributed, but made available within a location or geographical area by using a "central" server that is responsible for serving the content to clients located within the area.

The advent of peer-to-peer (P2P) computing has changed this approach. The term "P2P computing" emphasizes the shift away from centralized and client/server models of computing to a fully decentralized, distributed model of computing.

During the last couple of years, the term "P2P," or "peer-to-peer" has aggressively moved to the center-stage of the computing field. According to Clay Shirky, "P2P is a class of applications that takes advantage of resources – storage, cycles, content, human presence – available at the edges of the Internet…" **(Shirky, 2000)**. A report on P2P technology by Gartner Consulting **(Gartner Consulting Report, 2001)** states that "half of the current server-based content management vendors will add Data Centered P2P functionality to their product offerings by 2005 (0.7 probability)."

With P2P computing, the accent has shifted from storing content in, and serving from, centralized servers to storing and serving (at least some of) the content from the client-side. In this model, the content provider manages his/her content in a local client, and shares the content with anyone who is allowed to access the content. Responsibility for content creation, storage and security dwells on the client side. This has a lot of ramifications for the way in which corporate data is distributed.

There are several advantages to using the P2P approach to resource sharing in organizations. By shifting the responsibility for content to the client side, server-side management of diverse resources can be vastly reduced. Server managers need not be responsible for the integrity of the content. Problems arising from centralized distribution of content could possibly be averted.

The disadvantages include factors such as reduced security and reduced integrity of content arising from client-side mismanagement.

In this paper we discuss the architecture and implementation details of Mesh, a P2P system for corporate resource sharing. In section 2 we discuss current P2P architectures, their strengths and weaknesses. In section 3 we discuss some prime issues in P2P computing. In section 4 we present the primary design considerations underlying Mesh. In section 5 we present the Mesh architecture and discuss how Mesh works. In this section, we also specify certain generic characteristics for P2P systems. Section 6 contains implementation notes, and section 7 presents our conclusions.

P2P ARCHITECTURES: CURRENT ART

Several P2P implementations have been proposed in recent months. Most of the P2P applications are *file-centric*, and facilitate either synchronous or asynchronous file sharing. Newer applications also provide access to resources other than files. Almost all of the P2P systems that have been implemented use either a central-server based approach (e.g., Napster) or a "pure" P2P approach (e.g., Gnutella-based implementations. Gnutella is discussed in a later section).

We discuss below, the main architectural approaches to peer-to-peer resource sharing systems: P2P with centralized control, *pure* P2P, and a hybrid approach that incorporates aspects of the former two approaches.

Napster

A good example of P2P with centralized control is *Napster. (Note: In July 2000, Ninth Circuit U.S. District Judge Marilyn Hall Patel, issued the first of two injunctions that closed down the Napster service (from King, 2002). The Napster company and web site (http://www.napster.com), which in late 1999 boasted of 80 million registered users, is now defunct).* The Napster system uses a central server to maintain a list of connected clients. Every client connects to the central server, which scans the clients' disks for shared resources, and maintains directories and indexes of the resources. The central server responds to search queries from connected clients by sending back information on which of the clients hold the resources. Once a client knows where to find the resources that it is seeking (i.e., which client has the files it is searching for), it makes a direct connection to the appropriate client and transfers the resources.

Napster is not web-based, and does not run in a browser. It is a stand-alone application that runs on each individual client, and uses TCP/IP for its data-communication and data transfers. Since Napster depends on a central server that acts as a collector and regulator of information, the clients are not guaranteed anonymity. The Napster system is also vulnerable if the central server fails.

Gnutella

A good example of *pure* P2P is *Gnutella* (http://gnutella.wego.com). Gnutella is a generic term used to identify those P2P systems that use the gnutella protocol **(Kan, 2001)**. This is a reverse-engineered version of a P2P protocol that briefly appeared in AOL's system around March 2000, as a means for sharing recipes. Thus, there is no single interpretation of what the protocol is, actually. However, there are certain common elements that manifest in Gnutella-based systems. Chief among those is that Gnutella does away with the central server. In this system, each client continuously keeps track of other clients by pinging known clients in the system. Searches are propagated from one client to its immediate neighbors in ever-increasing circles until answers are found, or the search times out. Search responses are propagated back to the searcher in the same manner.

Like Napster, Gnutella-based systems are also not web-based, and run as stand-alone applications in client environments.

Gnutella is a truly anonymous resource sharing system. The searcher does not know the identity of the responder, and vice-versa. Trust is implicitly assumed.

A serious problem of Gnutella-based systems is their reputation for being unreliable. Lacking a central server that keeps track of which client is connected, and which is not, there is no way for a particular client to know if all its neighbors are alive and connected. This leads to less than reliable performance.

Web Mk

The *third* approach to P2P systems is what is referred to as *Web Mk.* This is more of an approach than an actual product, and is described in a Gartner Group Report **(Gartner Consulting Report, 2001)** on the emergence of P2P computing.

This is a web-based approach that uses web servers and web browsers. The web browsers would be configurable by users and would integrate resource-sharing features. The servers will maintain multiple indexes and allow

access to different forms of data. This type of system would use software agents or Bots to provide services such as extraction and consolidation of multiple resources, chat facilities, and notifications of changes. Search requests could be stored in the server and set to run in real-time or as a batch process, and alert the appropriate clients of the results.

The Gartner Report article does not mention any specific product that uses this approach. It is likely that a few products that incorporate some of the features described in the report will be released in time.

JXTA

A fourth approach to P2P computing is JXTA. Sun Microsystems has recently offered a new P2P application framework called JXTA (pronounced 'juxta') **(Gong, 2001)**. The framework offers a set of protocols, each of which is defined by a message. Each message has a predefined format and includes various data fields. The protocols offered are:

- Peer Discovery Protocol: Enables a peer to find another peer, peer group or advertisement.
- Peer Resolver Protocol: Enables peers to send and receive generic queries.
- Peer Information Protocol: Enables peers to learn more about other peers.
- Peer Membership Protocol: Enables a peer to join other peer groups, get information about or membership into groups, etc.
- Pipe Binding Protocol: Allows a peer to bind a pipe advertisement to a pipe end-point, thus indicating where messages actually go over the pipe.
- Endpoint Routing Protocol: Enables a peer to get routing information to route messages.

JXTA is transport-independent and can utilize TCP/IP as well as other transport standards. It is meant to be a conceptual framework that provides some protocols and mechanisms using which one can implement either a centralized or decentralized P2P system. In our opinion, JXTA offers a robust and flexible P2P solution framework. However, it must be noted that the protocols it offers are not standard.

ISSUES IN P2P FRAMEWORKS

Issues about the performance of Gnutella have been widely studied and discussed in available P2P literature **(Hong, 2001)**. Since JXTA is a relatively new, with a limited number of P2P implementations based on it, we not have much by way of prior studies. However, we believe that both Gnutella and JXTA, at their core, share certain basic traits. We discuss these traits, with the associated issues related to performance.

- **Effort expended:** Both approaches primarily offer frameworks for a "pure" (or almost pure) P2P architecture, which disposes of the need for a "central" server. Since there is no central server to maintain a master index, it takes more effort to query the system. Since each search requires a hop, this adds to the total bandwidth load and increases the search time. The lack of a central server also requires that a peer be aware of at least one other connected peer. Several recent Gnutella-based systems have tried to solve this problem by including a central server, which plays a limited role, such as providing some seed-IP numbers to a peer joining the network – thereby moving away from a "pure" P2P implementation.
- **Participation base:** Both approaches to building P2P communities are synchronous, and thus depend on the presence of a sufficient base of connected clients in order to function successfully.
- **Connection speed:** Connection speed dominates processor and I/O speed as the bottleneck. Due to the highly parallel nature of P2P, a connection fast enough to talk to one remote peer quickly becomes less so for ten of them trying to connect simultaneously. This problem will affect both Gnutella and JXTA systems, as they are inherently synchronous systems.
- **Free rider:** According to a recent analysis by E. Adar and B.A. Huberman at Xerox PARC **(Edar and Huberman, 2000)**, nearly 70 percent of current Gnutella users may be sharing no files at all. This may not be a big issue with central-server systems like Napster, since all the files and directories are indexed in the central server. But in a Gnutella system, where searches are propagated to each peer, this may consume more bandwidth and contribute to performance decline. (This problem may have been solved in JXTA, through the introduction of the "Endpoint Routing Protocol).

OUR APPROACH TO P2P RESOURCE SHARING: PRIMARY DESIGN CONSIDERATIONS

We propose a hybrid approach to P2P resource sharing within a corporate environment. The approach is code-named *Mesh.* Mesh is a hybrid system for P2P resource sharing. It consists of both a server and a client component. It supports the features of the three existing P2P architectures described above, as well as certain some additional characteristics.

The Mesh client is an application running in the client computer, and will be a modified Gnutella client. While building upon the base protocol we integrate a reliable IP repository, security integration through enterprise systems, an enhanced client side database for better search results and some basic network activity reduction. Together these qualities provide better P2P services for the corporate environment.

Specific characteristics of Mesh:

- **Reliable IP Repository –** Unlike Gnutella, each client first "announces itself" to a Mesh server, and requests a list of IP addresses of connected clients. The Mesh server sends a seed list of the connected clients. (We call this a "seed" list because each client needs only a limited number of other connected clients to get started). The Mesh server maintains a "current" list of connected clients by maintaining a list of clients, and pinging each client periodically.
- **Metadata –** Most of the currently available P2P systems do not provide the facility for client-side metadata description. Mesh will provide for client-side metadata description. Client-side metadata description allows an easy way to search for different types of content based on types (i.e., spreadsheet, audio, video), content subject (i.e., annual report, benefits plan, etc), content keywords, etc.
- **Authentication and Authorization –** The metadata could also consist of simple file descriptions as well as security and access control information. The security checks can be local account based or enterprise level security such as the corporate LDAP directory.
- **Enhanced Client Database –** Each Mesh client maintains a database of resources that it shares. The database will not only contain names and characteristics of the files, but also user-defined metadata describing the files.
- **Reduced Network Activity –** Unlike Gnutella, each client does NOT ping the other clients continuously. Instead, a client maintains awareness of other connected clients by downloading the list of IP addresses from the Mesh server periodically. Each client sends a Gnutella handshake to each of the clients in the list received. If a Gnutella acknowledgement is received from another client, that client is added to the original client's list.
- **Gnutella Protocol Based –** Like most of the packet communications, search and search response is accomplished among clients using the Gnutella protocol **(CapnBry, 2002)**.

This approach enables us to use the central server concept within a P2P environment that results in a highly enhanced P2P resource sharing system. This system thus builds upon existing P2P approaches and provides additional functionality.

MESH ARCHITECTURE

The Mesh architecture is illustrated in Figure 1.

Explanation of the Architecture

- Like Gnutella clients/servers, Mesh receives three types of messages, GET, SEARCH and PING. In the figure these low level packets are represented as components (boxes). The code in a Mesh client that handles "message routing" is labeled "Event Dispatcher."
- The Event Dispatcher takes the incoming packet and routes it to the correct message handler (i.e., PING, GET or SEARCH handler). Items in the figure marked in gray represent the components that are Mesh specific. These are the Reliable IP components, and the Authorization and Authentication components.
- In Figure 1, it should be noted that GET and SEARCH cannot be reached directly. A user may choose to share files without any required security checks. However, if a security check has been specified, if there is a security violation, then the remote client's request is never handed off to

the GET and SEARCH handlers. Basically, this means that the GET and SEARCH are hidden behind a custom firewall.

- The PING handler uses the "Reliable IP subsystem" (see figure above), and is not behind any security wall. It thus functions like a handler in any typical Gnutella client/server. In fact, the GET, PING and SEARCH handlers appear to a remote client as "available." The remote client does not know if a particular action is possible on a Mesh client or not. If a remote client initiates an action (i.e., PING, SEARCH or GET), and does not get any response, it simply means that the Mesh client's resources were protected against unauthorized searches and GETs. When there is a failed authorization or failed authentication, the remote client initiating the action does not receive any response from the search (i.e., it does not receive a response such as: "No items found"). Thus the remote client has not way of knowing if the items requested were protected or if they were non-existent.

Generic Architectural Attributes of P2P Resource Sharing Systems

Based on the prior work and our own work described above, we list below some attributes that we believe should be generic to P2P resource sharing systems. It is important to note that these attributes are not all available in current P2P systems. Some of the attributes have been incorporated into our Mesh system implementation.

- P2P systems should provide mechanisms for providing a reliable set of IPs to each connected peer. For example, each peer could first "announce itself" to a central server, and request a list of IP addresses of other connected peers. The server would send a seed list of the connected peers' IPs. (We call this a "seed" list because each peer needs only a limited number of other connected peers to get started). The central server maintains a "current" list of connected clients by pinging each client periodically – thus providing a reliable IP repository of connected peers.
- Each peer should maintain an enhanced database of resources that it shares. The database should not only contain names and characteristics of the files, but also user-defined metadata describing the files – thus providing an enhanced client database that is easier to search
- The metadata could consist of simple file descriptions as well as security and access control information. The security checks can be local account based or enterprise level security such as the corporate LDAP directory – thus providing enhanced authentication and authorization in the P2P resource sharing environment.
- Every peer need NOT ping the other peers continuously. Instead, a peer can maintain awareness of other connected peers by downloading the list of IP addresses from the central server. Each peers will then send a handshake to each of the peers in the list received. If an acknowledgement is received from another peer, that peer is added to the original peer's list. This "handshake" is also repeated while initiating a search, and defunct IPs are discovered and removed – a process that would lead to reduced network activity and decreased bandwidth usage.
- The central server can be enhanced to maintain a list of previous searches done by a specific peer, so that when the peer that actually has the resource comes online, it can be informed about the search.

IMPLEMENTATION

The Mesh implementation consists of two parts: the Mesh server and the Mesh client. The Mesh server was implemented in Java™. The Mesh client was implemented in Sash™. Sash Weblications for Windows is a dynamically configurable programming environment for rapidly building and deploying platform-integrated desktop applications using JavaScript and DHTML. This programming environment enables Web programming beyond the browser, and the resulting applications are integrated seamlessly into the common desktop environment and take advantage of the latest standards in Web services (http://www.alphaworks.ibm.com/tech/sash).

Figures 2 through 4 present some representative screenshots of the Mesh implementation.

CONCLUSION

In this paper, we have described a P2P system that is built using a hybrid approach, combining the features of pure P2P and central-server-oriented P2P systems. This approach enables us to use the central server concept within a P2P environment that results in a highly enhanced P2P resource sharing system. This system thus builds upon existing P2P approaches and provides additional functionality.

We have also provided a generic set of attributes for P2P systems, such as providing reliable IPs for P2P interaction, providing extensions to enhanced searches, providing enterprise level security and reducing network activity.

Preliminary tests indicate that the prototype and architecture is viable, and future work will involve enhanced architectures and testing for scalability.

REFERENCES

CapnBry 2002. "The Gnutella protocol," available online at http://capnbry.net/gnutella/protocol.php.

Edar, E. and Huberman, B.A. 2000. *"Free Riding on Gnutella,"* First Monday, Issue 5_10, October 2000, available online at http://firstmonday.org/issues/issue5_10/adar/index.html.

Gartner Consulting 2001. "The Emergence of Distributed Content Management and Peer-to-Peer Content Networks," GartnerGroup Report # 010022501, January 2001.

Gong, Li 2001. "JXTA: A Network Programming Environment," IEEE Internet Computing, Vol .5, No. 3, May/June 2001, available online at http://computer.org/internet/v5n3/w3jxta.html.

Hong, Theodore 2001. *"Performance,"* in Peer-To-Peer: Harnessing the Benefits of a Disruptive Technology, pp 203-241, Edited by Andy Oram, O'Reilly & Associates, California, 2001.

Kan, Gene 2001. *"Gnutella,"* in Peer-To-Peer: Harnessing the Benefits of a Disruptive Technology, pp 94-122, Edited by Andy Oram, O'Reilly & Associates, California, 2001.

King, Brad 2002. *The Day the Napster Died,"* Wired News, May 15, 2002. Available at http://www.wired.com/news/mp3/0,1285,52540,00.html.

Shirky, Clay 2000. *"What is P2P..And What Isn't,"* November 2000; available online at http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html.

URLS CITED

The Napster home page, http://www.napster.com

The Gnutella home page

Figure 1. Mesh architecture

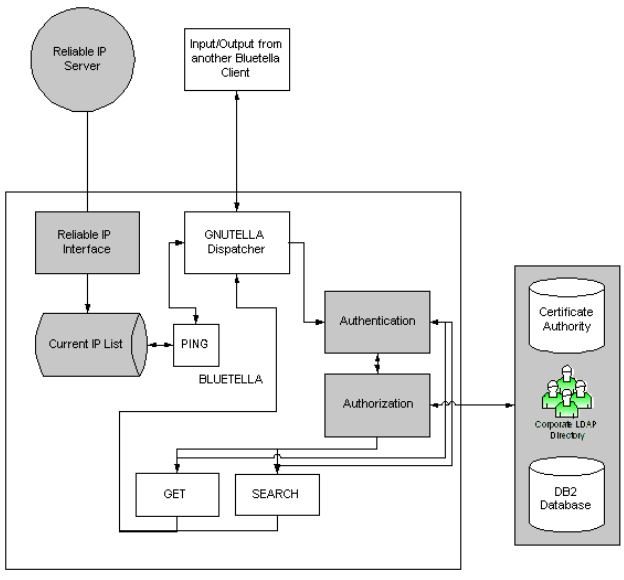

Figure 2. Mesh main screen

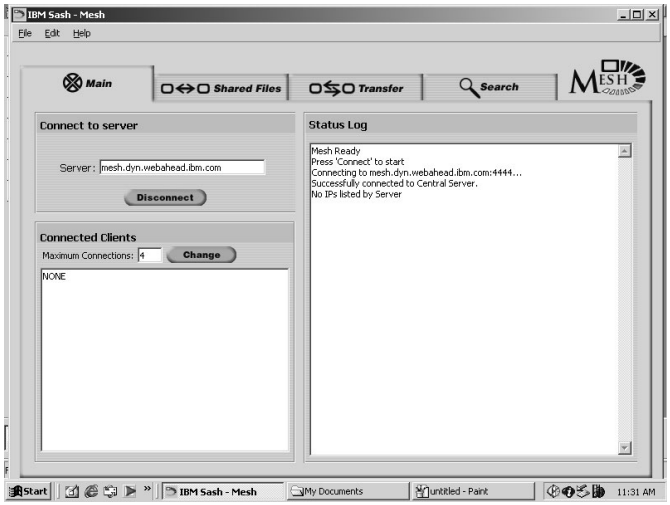

Figure 3. Mesh shared files screen

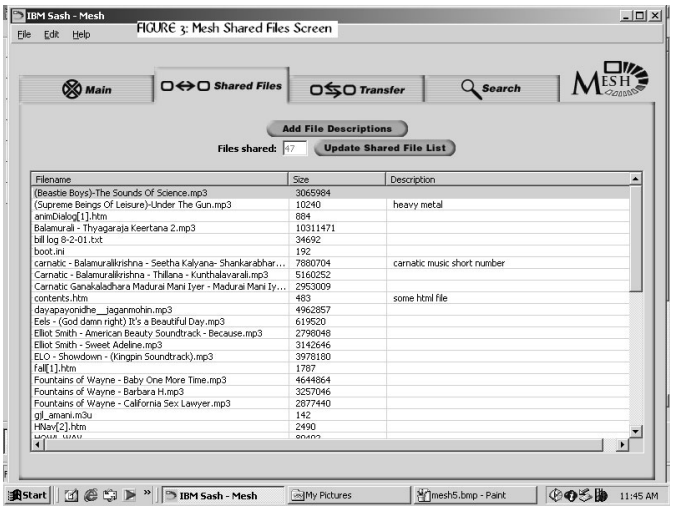

Figure 4. Mesh search results screen