FSS
File Sharing System

UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

School of Professional Advancement

File Sharing System
(FSS)

Aleem Akhtar 101927-006
FINAL PROJECT

FSS
File Sharing System

University of Management and Technology
School of Professional Advancement

File Sharing System
(FSS)

Project Submitted to the faculty of Computer Science in Information Technology,
University of Management and Technology, Lahore,
Pakistan.

In partial fulfillment of the requirements for the degree of

Master of Computer Science

Approved by

_______________ **Supervisor**

_______________ **Head Project Evaluation Committee**

FSS
File Sharing System

Table of Contents

FSS
File Sharing System

ACKNOWLEGMENTS

Thanks to **ALLAH ALMIGHTY**, The Most Beneficent and the Most Merciful, Who has bestowed His blessings upon us and made us able to complete out project and this report, although it is not an easy task, but with the devoting and inspiring attitude of our respected teacher I have at last succeeded in the completion of my project.
I would like to my profound gratitude to **Khurram khan**, **Slaeem Ata** and **Imran Saleem** their inspiring attitude, patronage and personal interest in the completion of my project. They provided us an atmosphere that is highly Convenient & appreciative.
I am also very grateful to all teachers for their expert's advice and friendly concerns.

BASIC TERMINOLOGY

PEERS

All of the computers in the network are considered equal and are therefore referred to as peers.

"A PC connected to the Internet, that can communicated directly with other PCs, using some communication protocol".

PEER-TO-PEER NETWORKING

Peer-to-peer is a style of networking in which a group of computers communicate directly with each other, rather than through a central server (such as hard drives and processing cycles).
Peer-to-peer computing is sharing of computer resources and services by direct exchange between the computers. It is a type of network where computers connected to each other pool their data and resources like disk drives, computing cycles etc. and these collective resources are accessible from every PC connected in that network. This is unlike the Client/Server architecture, where some computers are dedicated to serving the others, i.e. the servers serving the requests of the clients.

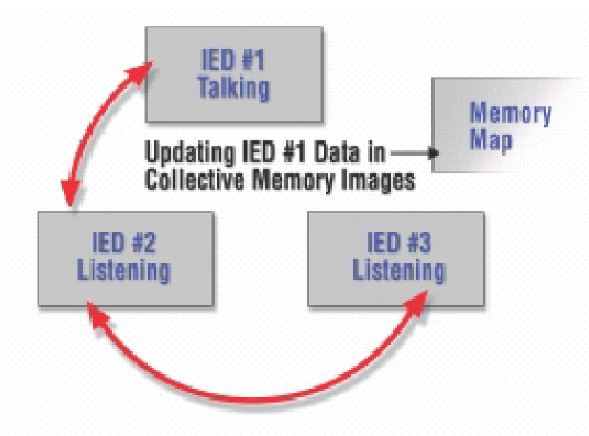


Figure 2 Data sharing for PeerComm network

Types of Peer-to-Peer

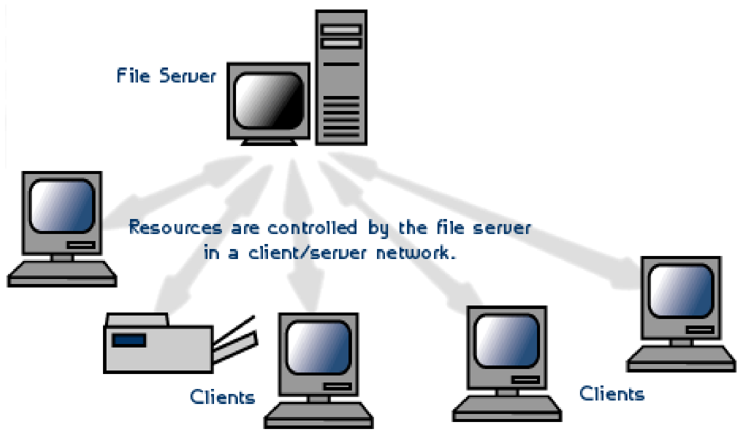
In pure P2P networks, peers have the same capability and responsibility.
Hybrid:
In Hybrid P2P scenarios, there are intermediate servers to facilitate the interaction between peers. For e.g. Napster, the largest Internet mp3 songs sharing site, is a hybrid

P2P type of a network. Napster has a central server, which coordinates the file sharing among the millions of peers, who are the members of Napster.

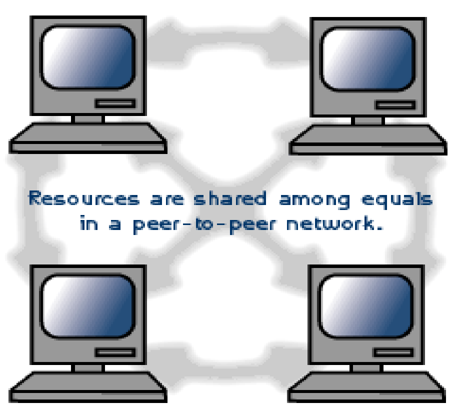
Pure:

Gnutella, a file sharing service on the Internet, is a pure P2P service where there is no presence of a central server for facilitating peer-to-peer files sharing. In pure peer-to-peer all participating nodes are peers. That means all have the same rights and duties although they may rely on different hardware basis. There is no central institution, to manage, to control and coordinate.

Client Server



Peer-to-Peer Networking



PEER-TO-PEER AND NAPSTER

Peer- to- Peer Networking is mostly known under the brand of Napster. Within this application the Peer-to-Peer networking concept is used to share files, i.e. the exchange of MPEG Layer3 (mp3) compressed audio files. However, Peer-to-Peer is not only about file sharing, it is also about establishing multimedia communication networks based on

Peer-to-Peer concepts or re-sources sharing. A basic problem often encountered, is the multi-faceted and confusing situation, concerning the terms related to Peer-to-Peer networking in publications and discussions.

The cases where P2P will do well are:
When the same data exists on different PCs. In case where only 1 machine has the data required, access to it could be unreliable. When reliability or constant availability is important and the data is stored as unique content on each PC then it will not be available when required.
The files are static; the information being downloaded is never changed. Content that constantly changes makes the recently downloaded content obsolete and therefore causes a lot of versions of the same file, thereby reducing reliability.
Data, which is not very quality sensitive. It really does not bother the user much as to how good the data is so the user doesn't mind trusting the person sharing it. If a music file for personal use was converted from CD to MP3 poorly, many people don't care. If the file being downloaded was destined for broadcast or other commercial purposes having an appropriate trust relationship with the source may be important and that complicates things and may not be practical.

- Reliable connection speed. If the data required is on a low-speed link then any high-speed connection on the other side will not help. The data will be transferred at the low rate only.

Some limitations of P2P:

Devise a way to enable devices to find each other in a computing model that lacks a central server.
P2P requires peers to access decentralized resources that often don’t have permanent IP addresses because they are not always connected to the Internet.
Bandwidth sharing also is a cause of major concern. P2P requires a lot of bandwidth. If a file is required by a thousand users at the same time, is residing on a single machine, then there will be hardly any bandwidth left for the user of that machine to do any other activities on the Internet.
Corporate Firewalls pose the final problem. The tendency of firewall administrators to keep their users from getting fixed IP addresses makes it much harder for the end user’s system to host a service. Even if a user gets a fixed IP address, the Domain Name System is a barrier to end-to-end addressing.
Data is very difficult to find in a P2P network because of lack of standardized ways to categorize it.
Some of the things that still need to be worked upon, when designing a P2P networks are:
Ubiquitous: P2P should be capable of encompassing the billions of computers that form the Internet today.

Standards:

Interoperability: look for ways to achieve interoperability. In the Internet there are millions of computers with thousands of configurations, which may or may not be compatible with each other, so P2P networks should work in such a way that compatibility issues get resolved. Therefore standards are critical for interoperability.
Security: There are two issues in this domain, one is with secure sharing in groups where there should be a way to guarantee trust within a secure group and prevent any unauthorized access. The other security issue is the protection against attack like viruses, worms, man-in-the-middle attacks etc.
Robustness: In order to make a P2P network robust, there should be some kind of control using backup/restore alternatives, guaranteed identity of content and efficient storage and network management. There should also be provision for intermittent connectivity and resilience against software and hardware failures.
Performance: A P2P network should understand the performance implications of a billion connected computers and figure out ways to coordinate the traffic so that least congestion occurs in the network. It should also understand the inherent scalability limits of certain networks and the performance limits of the present protocols on the Internet.
Community: P2P development should strive towards building up dedicated community of users from all diverse parties of interest like IT Departments, enterprise Community users, peer-to-peer application developers and the open-source community. P2P should allow spontaFSSus, secure communication between small groups that can be created on-the-fly.

Example:

DNS:

The Domain Name System (DNS) is an example of a system that blends peer-to-peer networking with a hierarchical model of information ownership. The remarkable thing about DNS is how well it has scaled, from the few thousand hosts it was originally designed to support in 1983 to the hundreds of millions of hosts currently on the Internet. The lessons from DNS are directly applicable to contemporary peer-to-peer data sharing applications.
DNS was established as a solution to a file-sharing problem. In the early days of the Internet, the way to map a human-friendly name like bbn to an IP address like 4.2.49.2 was through a single flat file, hosts.txt, which was copied around the Internet periodically. As the Net grew to thousands of hosts and managing that file became impossible, DNS was developed as a way to distribute the data sharing across the peer-to-peer Internet.
The namespace of DNS names is naturally hierarchical. For example, O'Reilly & Associates, Inc. owns the namespace oreilly.com: they are the sole authority for all names in their domain, such as www.oreilly.com. This built-in hierarchy yields a simple, natural

way to delegate responsibility for serving part of the DNS database. Each domain has an authority, the name server of record for hosts in that domain. When a host on the Internet wants to know the address of a given name, it queries its nearest name server to ask for the address. If that server does not know the name, it delegates the query to the authority for that namespace. That query, in turn, may be delegated to a higher authority, all the way up to the root name servers for the Internet as a whole. As the answer propagates back down to the requestor, the result is cached along the way to the name servers so the next fetch can be more efficient. Name servers operate both as clients and as servers.
DNS as a whole works amazingly well, having scaled to 10,000 times its original size. There are several key design elements in DNS that are replicated in many distributed systems today. One element is that hosts can operate both as clients and as servers, propagating requests when need be. These hosts help make the network scale well by caching replies. The second element is a natural method of propagating data requests across the network. Any DNS server can query any other, but in normal operation there is a standard path up the chain of authority. The load is naturally distributed across the DNS network, so that any individual name server needs to serve only the needs of its clients and the namespace it individually manages.
So from its earliest stages, the Internet was built out of peer-to-peer communication patterns. One advantage of this history is that we have experience to draw from in how to design new peer-to-peer systems. The problems faced today by new peer-to-peer applications systems such as file sharing are quite similar to the problems that Usenet and DNS addressed 10 or 15 years ago.

Advantages:

Locates files quickly and efficiently.
Searches are as comprehensive as possible.
All users must be registered to be on the network.

Disadvantages

Vulnerable to censorship and technical failure.
Popular data becomes less available because of server load.
Central index data may be out of date as it is not updated real time.

IP ADDRESSES

Every machine on the Internet has a unique identifying number, called an IP Address. A typical IP address looks like this:
216.27.61.137
To make it easier for us humans to remember, IP addresses are normally expressed in decimal format as a "dotted decimal number" like the one above. But computers communicate in binary form. Look at the same IP address in binary:
11011000.00011011.00111101.10001001

The four numbers in an IP address are called octets, because they each have eight positions when viewed in binary form. If you add all the positions together, you get 32, which is why IP addresses are considered 32-bit numbers. Since each of the eight positions can have two different states (1 or 0) the total number of possible combinations per octet is 2^8 or 256. So each octet can contain any value between 0 and 255. Combine the four octets and you get 2^{32} or a possible 4,294,967,296 unique values!

Out of the almost 4.3 billion possible combinations, certain values are restricted from use as typical IP addresses. For example, the IP address 0.0.0.0 is reserved for the default network and the address 255.255.255.255 is used for broadcasts.

The octets serve a purpose other than simply separating the numbers. They are used to create classes of IP addresses that can be assigned to a particular business, government or other entity based on size and need. The octets are split into two sections: Net and Host. The Net section always contains the first octet. It is used to identify the network that a computer belongs to. Host (sometimes referred to as Node) identifies the actual computer on the network. The Host section always contains the last octet. There are five IP classes plus certain special addresses:

Default Network - The IP address of 0.0.0.0 is used for the default network.

Class A - This class is for very large networks, such as a major international company might have. IP addresses with a first octet from 1 to 126 are part of this class. The other three octets are used to identify each host. This means that there are 126 Class A networks each with 16,777,214 (2^{24} -2) possible hosts for a total of 2,147,483,648 (2^{31}) unique IP addresses. Class A networks account for half of the total available IP addresses. In Class A networks, the high order bit value (the very first binary number) in the first octet is always 0.

Net	Host or Node
115.	24.53.107

Loopback - The IP address 127.0.0.1 is used as the loopback address. This means that it is used by the host computer to send a message back to itself. It is commonly used for troubleshooting and network testing.

Class B - Class B is used for medium-sized networks. A good example is a large college campus. IP addresses with a first octet from 128 to 191 are part of this class. Class B addresses also include the second octet as part of the Net identifier. The other two octets are used to identify each host. This means that there are 16,384 (2^{14}) Class B networks each with 65,534 (2^{16} -2) possible hosts for a total of 1,073,741,824 (2^{30}) unique IP addresses. Class B networks make up a quarter of the total available IP addresses. Class B networks have a first bit value of 1 and a second bit value of 0 in the first octet.

Net	Host or Node
145.24.	53.107

Class C - Class C addresses are commonly used for small to mid-size businesses. IP addresses with a first octet from 192 to 223 are part of this class. Class C addresses also include the second and third octets as part of the Net identifier. The last octet is used to identify each host. This means that there are 2,097,152 (2^{21}) Class C networks each with 254 (2^8 -2) possible hosts for a total of 536,870,912 (2^{29}) unique IP addresses. Class C networks make up an eighth of the total available IP addresses. Class C networks have a first bit value of 1, second bit value of 1 and a third bit value of 0 in the first octet.

Net	Host or Node
195.24.53.	107

Class D - Used for multicasts, Class D is slightly different from the first three classes. It has a first bit value of 1, second bit value of 1, third bit value of 1 and fourth bit value of 0. The other 28 bits are used to identify the group of computers the multicast message is intended for. Class D accounts for 1/16th (268,435,456 or 2^{28}) of the available IP addresses.

Net	Host or Node
224.	24.53.107

Class E - Class E is used for experimental purposes only. Like Class D, it is different from the first three classes. It has a first bit value of 1, second bit value of 1, third bit value of 1 and fourth bit value of 1. The other 28 bits are used to identify the group of computers the multicast message is intended for. Class E accounts for 1/16th (268,435,456 or 2^{28}) of the available IP addresses.

Net	Host or Node
240.	24.53.107

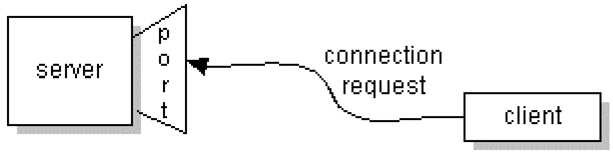
Broadcast - Messages that are intended for all computers on a network are sent as broadcasts. These messages always use the IP address 255.255.255.255.

SOCKETS

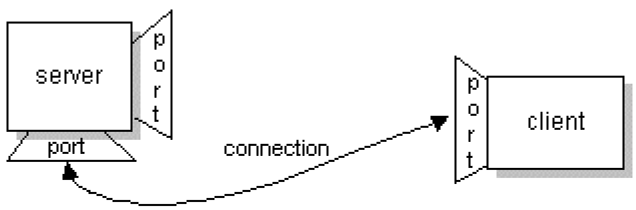
A socket is one end-point of a two-way communication link between two programs running on the network. Socket classes are used to represent the connection between a client program and a server program. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent.

Normally, a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

On the client-side: The client knows the hostname of the machine on which the server is running and the port number to which the server is connected. To make a connection request, the client tries to rendezvous with the server on the server's machine and port.



If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to a different port. It needs a new socket (and consequently a different port number) so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.



On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server. Note that the socket on the client side is not bound to the port number used to rendezvous with the server. Rather, the client is assigned a port number local to the machine on which the client is running.
The client and server can now communicate by writing to or reading from their sockets.

PROJECT MANAGEMENT PLAN

OBJECTS & SCOPES
PEER SIDE SOFTWARE

Objectives:

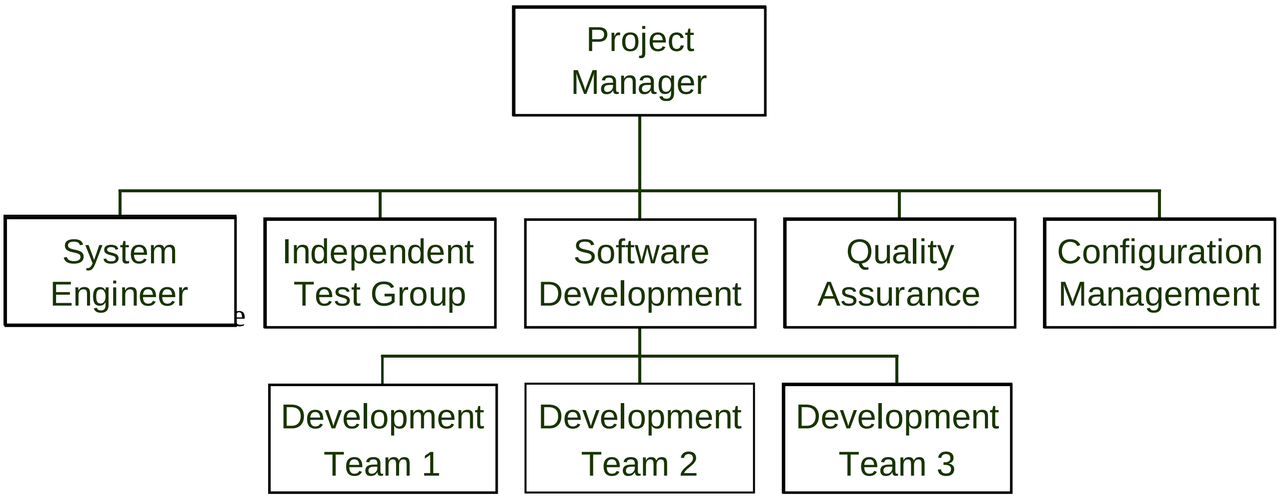
Peer side software will allow Peer to configure Peer side software at the start of the session.
Peer side software will allow Peer to register at the server if the software is being used for the first time.
Peer side software will allow Peer to configure the desired server at the start of the session with which the peer wants to be connected.
Search for the files from all the peers that are connected to the server.
Receive and manage the file as received from Peers.
Peer side software should be able to send the files to other peers that request for the specific files.

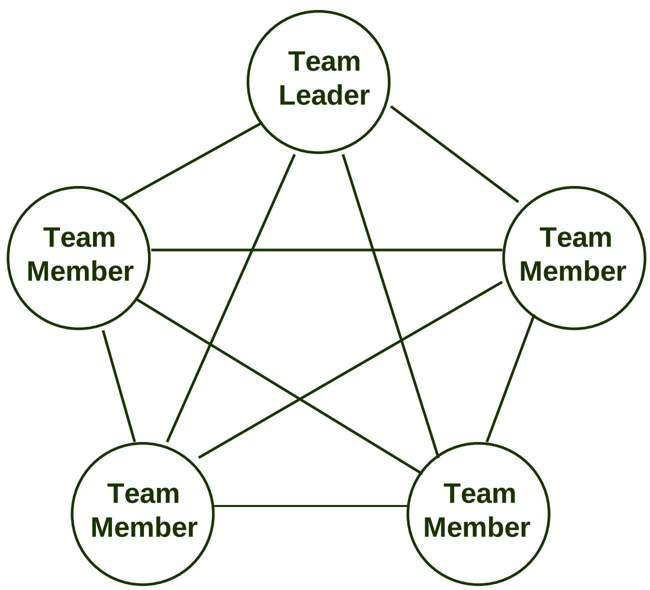
SERVER SIDE SOFTWARE

Objectives:

Server Side Software should be able to manage the database of all the registered peers.
Server side software should allow searching facility between the peers connected in that particular session.
In case a peer is interested in a file transfer, then the server side software should allow the direct connection between the to peers.

ORGANIZATIONAL STRUCTURE





Democratic Team

Team lead will communicate with the team members on behalf of the project manager and present the team to the project manager. The team lead will coordinate activities within the team. Responsible for handling the administrative tasks. The whole team in periodic meetings walk troughs will make technical decisions.

REQUIREMENTS

For details about requirement please refer to Requirement Specification document of FSS (P2P File Sharing System).

RESOURCES

Software

The following software resources are required for development.

Microsoft Windows 7/Windows2003 Server
Visual Studio.NET
MSDN 2002
Microsoft Office XP
MS Project
Rational Rose
Macromedia Fireworks
Adobe Photoshop
Microsoft Windows

Hardware

Server

Three Core 2 Duo Machines with 2GB RAM
(Development & Production)

Peer

Core 2 Duo Machines
2GB RAM
Library
UML
Visual C++
Win32 API
Ms Project

Project Manager 1 (Full-time)

Development 1+
(It may vary and training will required)

QA/Testing

1+
(It may vary as per available of resources and training requirement)

EXIT CRITERIA

This project will be deemed complete when all the functionality mentioned in the Functional Specification FSS_1.0 has been implemented.

Acceptance Criteria

Functional Specification will act as the acceptance criteria.

DELIVERABLES

Requirement Specification
Project Plan
Date of Deliver 07-10-2013
Location UMT, Lahore
Functional Specification
Shipment Package
Tested Source Code
Installation Guide (Readme.doc)

TRAINING

UML
VISUAL C++
Microsoft Project
Rational Rose
PROJECT SCHEDULE
Project timelines may vary frequently due to the nature of the project and resources.

Please refer to the latest version Project_FSS_1.5.mpp

PEER INVOLVEMENT

Joint Reviews
The RS will be review jointly.

INFORMAL MEETINGS

The informal meeting will be schedules at different intervals as per the requirement.

Reporting

Modification and changes

Modifications will have to be agreed by the Project Advisor

Approval

Formal approval will be required from Project Advisor.

Acceptance

As per FS

Access to facilities

All the Books which are related to the Project are always available to the development team.

Peer Deliverables

Tools
Microsoft Windows 7/Windows2003 Server
Visual Studio
MSDN
Microsoft Office
MS Project
Macromedia Fireworks
Adobe Photoshop