PROMOND – Network File Access and Standby for the Files over the Network

by

Vinoth Pugazenthi

Problem report submitted to the College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Electrical Engineering

Approved by

Dr. Yenumula.V. Reddy, Committee Chairperson
Dr. Sumitra Reddy
Dr. James D Mooney

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2007

Keywords: network file access, network file standby, file auto save, TCP/IP

ABSTRACT

PROMOND – Network File Access and Standby for the Files over the Network

By

Vinoth Pugazenthi

The computer users of today work with a lot of files each day and to top it all, many of the users have more than one computer that they use. When we take the example of Professors in colleges and universities, the above proves more possible than not for then to use more than one computer.

In the absence of internet, the transportation of files within the set of computers they use was relatively difficult. But, with such an availability of file transfer over the network, we should be able to provide access of files over the network and avoid the transfer of files with removable devices. The removable devices help in file transfers, but we do have to find the files in the respective folders and keep track of the various versions of the many files, some of which may have the same names.

The project implies to enable a user maintain his/her files on a remote server and be able to access, modify and use the files from client systems, geographically separated, as long as they are connected on the network. This is a specific application developed keeping the professors and their kind of work as target. It enables a Professor that has to work from different computers, he/she may possess at different places.

ACKNOWLEDGEMENT

I would like to thank Dr. Yenumula V. Ramana Reddy for being more than just my advisor and guiding throughout the whole of the project. He also gave the interesting requirements for the projects and instigated the thinking in me as well as the rest of the members in the project to come up with new ideas. He also gave the liberty to implement the ideas that we came up with.

I would also like to thank Dr. Sumitra Reddy and Dr. James D. Mooney for being part of the committee and supporting the project throughout with their suggestions and encouragement.

I also thank my parents and sister, who were, though not involved technically in the project, very supportive and encouraging in doing the project, in spite of them being at a far distance.

I am grateful to Mr. Tim Mitchem and the College of Human Resources for funding me throughout my Master's while helping me learn and improve technically.

I am not complete in my acknowledgement, if I don't thank my friends that have helped me technically. They have also been a continuous support for me.

TABLE OF CONTENTS

List of Figures

Chapter 1 Introduction

1.1 Problem Definition

To develop a client that can facilitate a instructor to access the centrally located files from any of his/ her computers, edit, modify, auto-save the files and to provide a standby for the files being accessed for each session.

1.2 Network File Access

It is the world with the internet now. Everyone is running around doing multi tasks and handling many complications while getting work done. The concept of “PC Anywhere” is spreading among the computer users that travel a lot and get to work at many places. The Network File Access System is a way in which the user will be able to maintain the files in a server and access it from different computers. The user will be able to see, open, modify, read and write the files on the server.

1.3 Need for Network File Access

The computers are basically used to store the files. When we edit the files, we save them, close them, and open them later for editing them again. Nowadays, we have sleek laptops that can be carried around. But, there are a lot of cases where people are obliged to use more than one computer. For example, as this project is intended to address, the instructors in colleges are provided with more than one computer to use. They will be given an office computer, a computer dedicated for research, and apart from these, they may be having their own personal laptop and/or a desktop at home. In such cases, there may be a need to use the same files at all or many computers.

With much advancement in the computer and electronics field, the shrinkage in the size of memory storage devices has led to easy transportation of files, by using Floppies, CDs and USB Drives. The more common sight and action that a person does these days is send the files over email to oneself, download it from the other end and use it. However, the user has to hunt for the files, maintain the version, remember the files they were last editing and the situation is very complicated when there is more than one file with the same name, but in different folders.

The Network file access system is a necessity in the above mentioned situation, because of its capability in not only letting the users access the files from different computers in different locations, but also in keeping track of what files the user was using recently. The user need not worry about the version of the files that they edit.

1.4 Who does the Promond intend to serve?

The Promond is intended for serving instructors that use various different computers that are geographically separated. In using the Promond client, as long as the computer is connected to the network, it will be able to bring up the files that the instructorr has stored on the server. The environment that the professor sees on one computer will be the same as on the other computers. He/She will be able to see the private files and the public files on the server.

1.5 Private and Public Files

The private files will be the files that the user stores in the folder allocated to him/her on the server. The public files will be the general files in the folder that are viewable, accessible and editable by any user that has a private access to the server. It is for basic and general sharing of the files between various users.

Chapter 2 Technical Details

2.1 Sockets

According to Jim Kurose and Keith Ross, the sockets are defined as a *host-local, application-created, OS-controlled* interface into which application process can both send and receive messages to/from another application process.

The communication between the computers is facilitated by means of sockets. In this project, the user's computers are considered to be clients and the centralized system is considered to be the server. They are individually connected to the network through the sockets and connected with each other through their respective sockets.

The clients and the server understand each other with some kind of language. They have a protocol using which they communicate. In computer terms, a protocol is defined as a standard procedure for regulating data transmission between computers. In this project the inherent protocol used is TCP (Transmission Control Protocol).

2.2 Transmission Control Protocol:

TCP is a protocol that has the following features:

Connection-Based

Reliable

Has control over the flow of messages

Congestion control

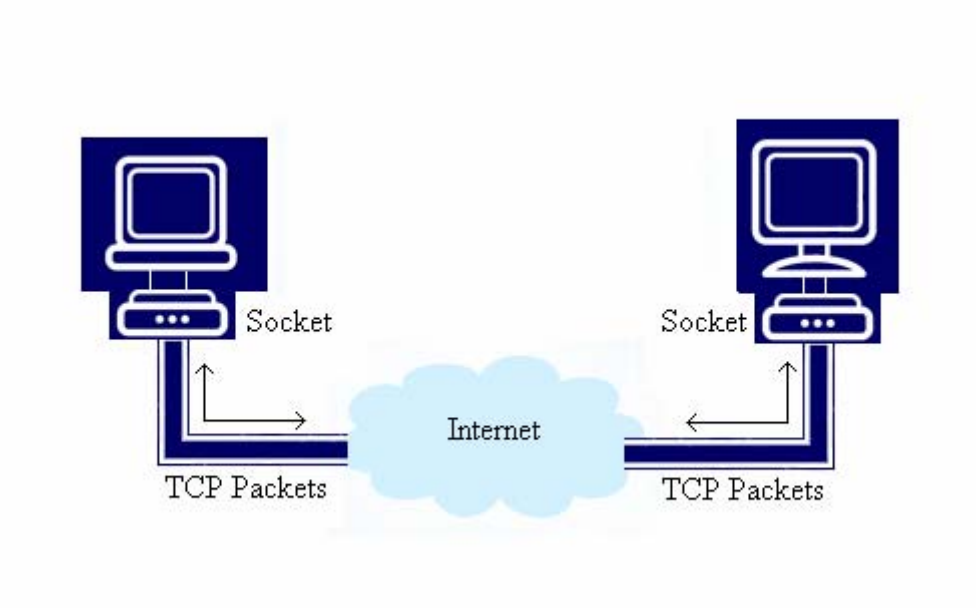


Figure 1: TCP Communication over the Sockets

2.2.1 Features of TCP

The Transmission Control Protocol has the following features:

i. Single sender and single receiver when sending each message

ii. Reliable, because of the Acknowledge messages

iii. Maintains flow control and congestion control

iv. Duplex data transfer (communication can happen in both directions simultaneously)

v. Handshaking signal enable connection confirmation

2.2.2 TCP Segment Structure

The Transmission Control Protocol packet consists of 32bits data that has the following information that it sends to the receiver.

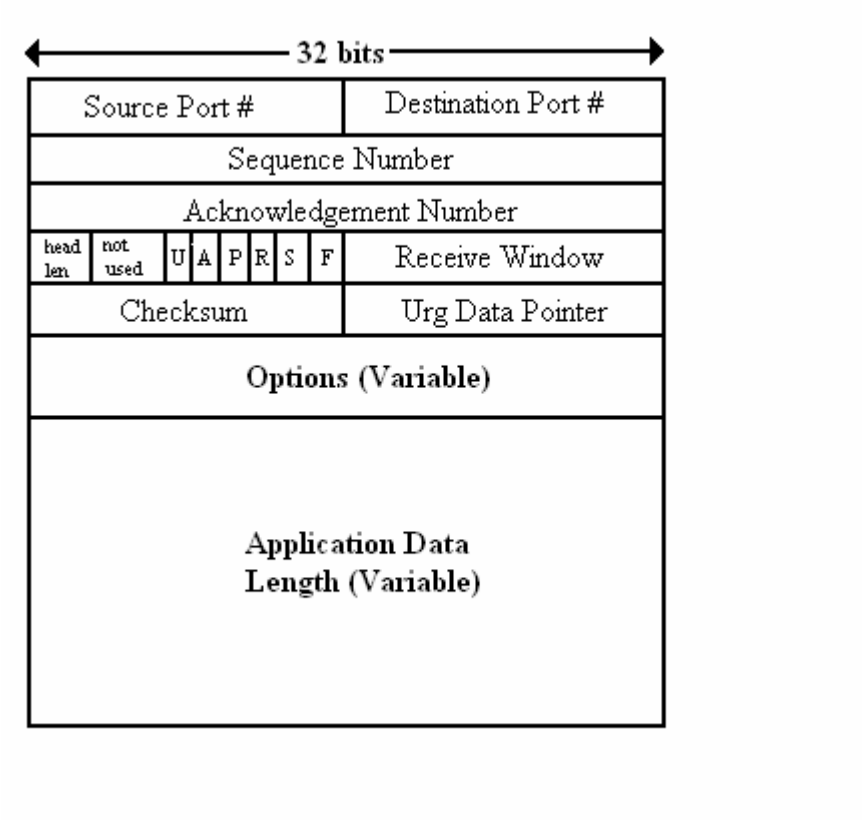


Figure 2: TCP Segment Structure

Figure adopted from Computer Networking: a Top down Approach Featuring the Internet

Source Port: The port number of the host that sends the message

Destination Port: The destination port of the host to which the sender is sending the message

Sequence Number: The sequence number allocated for this message

Acknowledgement Number: The acknowledgement number for the message it received

U: urgent data

A: Acknowledge Number valid

P: Push data

R, S and F: connection established (setup, teardown commands)

Checksum: Internet Checksum

Receive window: Maximum number of bytes receiver can accept

2.2.3 TCP Message Exchange

The exchange of messages between two hosts is shown below in the figure 3. In each message the sequence number of the message being sent is attached. If the host sending the message has received another message before, for which it needs to send the acknowledgement, the acknowledgement number for that sequence will be sent. Apart from these, the actual data will also be sent.

After receiving the message, the new message with the acknowledgement number for the received sequence will be sent by the other host.

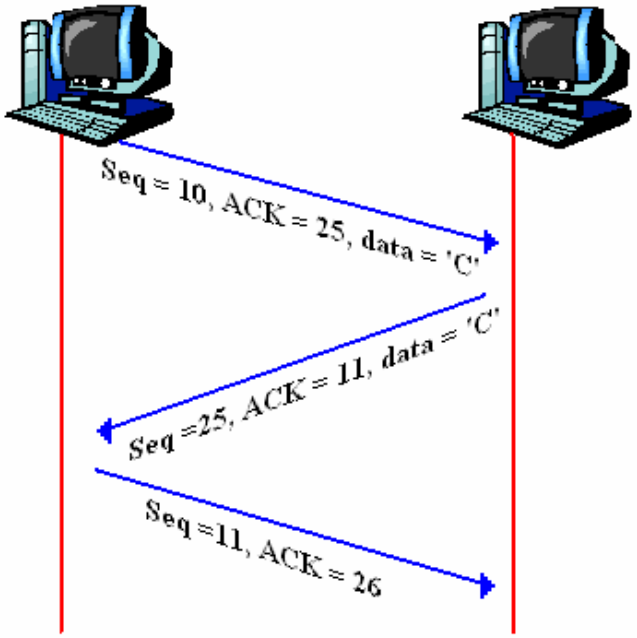


Figure 3: TCP Sequence and Acknowledgement

Figure adopted from Computer Networking: a Top down Approach Featuring the Internet

2.3 Byte Stream

The data is transferred in the form of byte streams. The messages are converted into bytes, written to the socket and are then sent to the destination port. When receiving the bytes, they are read as byte streams from the socket and are converted back to actual messages. These processes are accomplished by reading and writing through getInputStream() and getOutputStream() methods of the Java Socket class.

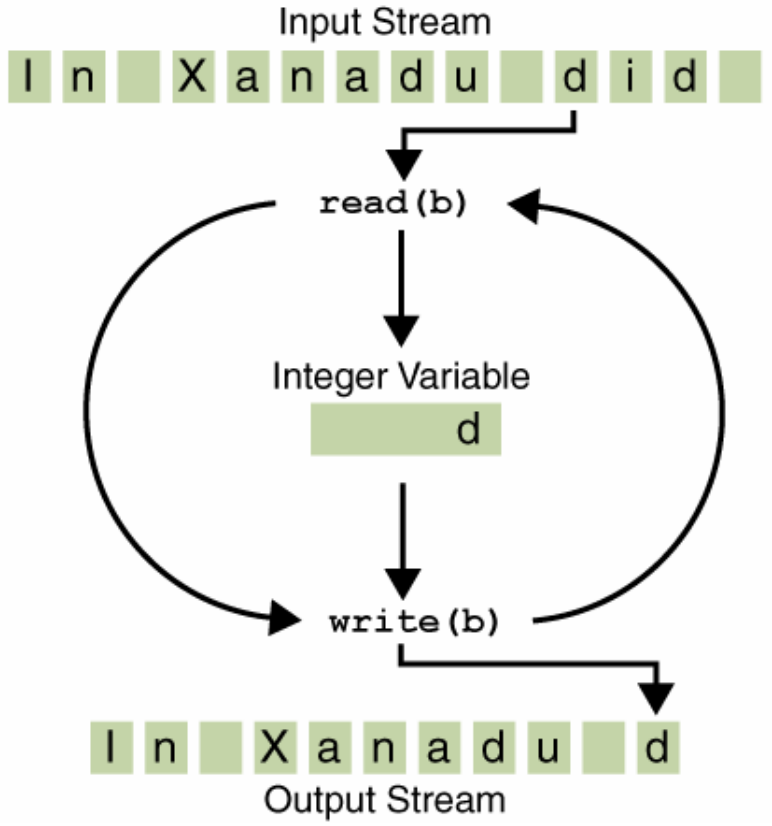


Figure 4: Byte Streams written and read from the sockets.

Source: http://java.sun.com/docs/books/tutorial/figures/essential/byteStream.png

File Streams:

In this project, the file transfer occurs in the form of streams of data read from files and transferred over the TCP network. File Input Streams and File Output Streams are used to read and write data in the sockets respectively.

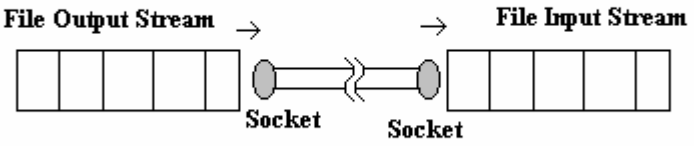


Figure 5: File Input and Output Streams.

Chapter 3 File Access and File Stand By

3.1 File Access System

In the file access panel, there are two different labels, one showing the Private files and the other showing the Public files. These files are actually a representation of the files physically present on the server. The user can browse through the folders shown on the client window. Whenever the user double-clicks on any of the files, the file will be opened in the local machine. The background details of the procedure to open the file in the local client is described in this section

The path of the file on the server is taken from the JTree that shows the files. The client then sends a request to the server to send the particular file. The server gets the path of the file sent by the client and opens a port for file transfer. The client gets the port through which the server will eventually send the file. It opens a port and accesses the port in the server that the server sent. Then the file is transferred using File Input/Output Stream. The file is stored in the temporary folder created by the client program in the local system. The client then opens the file from the temporary folder.

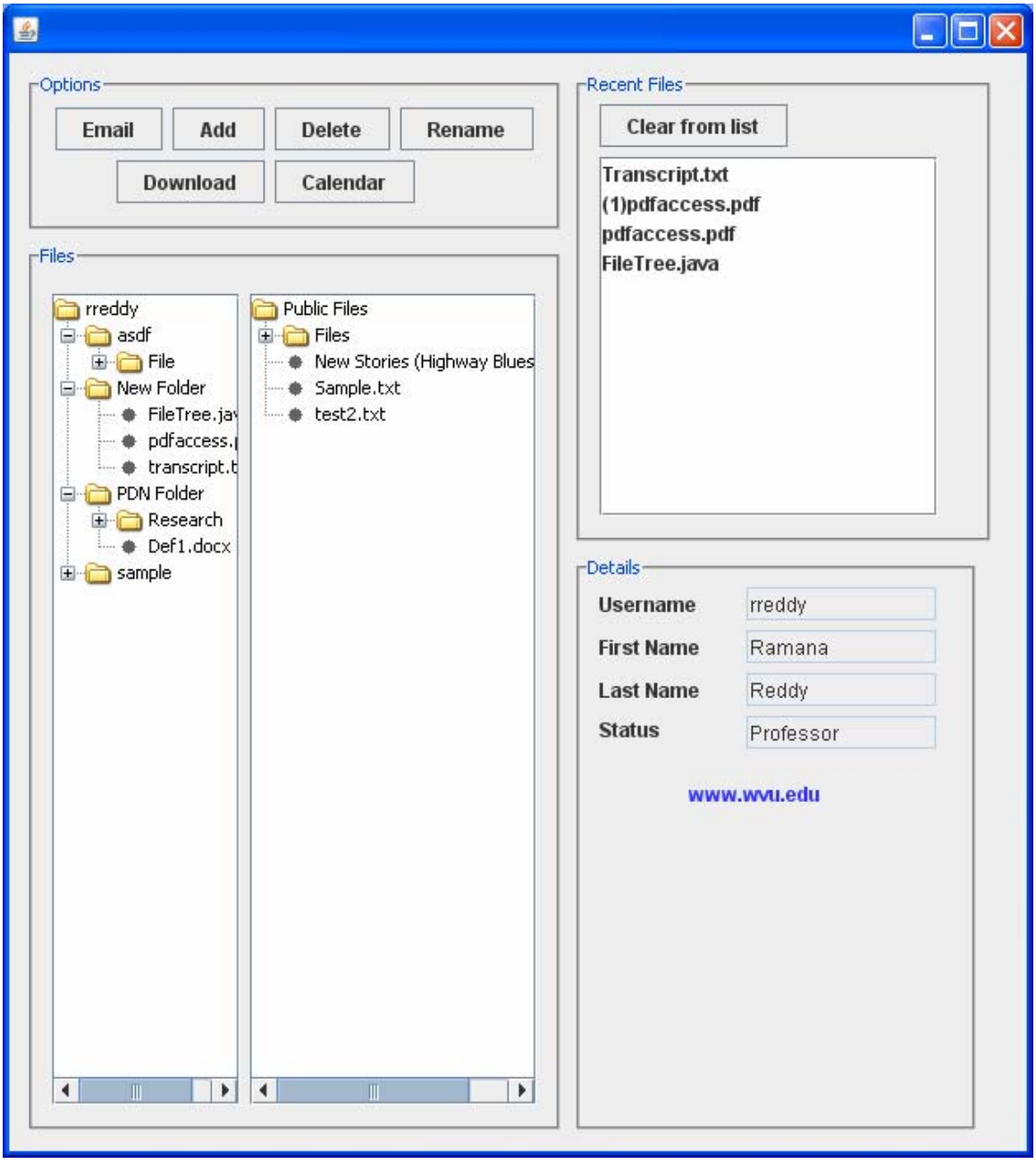


Figure 6: Client Window with all the panels.

3.2 Recent File List

Whenever the files are opened from the server, the client maintains a list of the files the user has opened in a static Hashtable. The hashtable contains both the file name and the path of that file on the server. It also shows on the File-Being-Used-List on the

right pane. When the session is ended the client sends the files back to the server. The server then gets those files and stores their respective path in the File Details database.

During the next login, regardless of what computer the user is using, the server retrieves the list of files used in the last login from the database, and sends them to the user's computer. The client then stores them in the temporary folder and opens it up for them. This creates a file restoration system where the recent files the user was editing are brought to the user, without the need for browsing through the folders to retrieve the recent files back.

However, the user can clear any number of files from the list, in which case, the files will be saved and sent back to the server, and they will not be automatically opened the next time, unless opened manually in the same session.

For every session, when the user logs out, the server updates the File Details database with the list of file paths sent by the client. So, the file paths are not appended rather it only stores the recent file list of the very last session and does not keep track of other historical sessions.

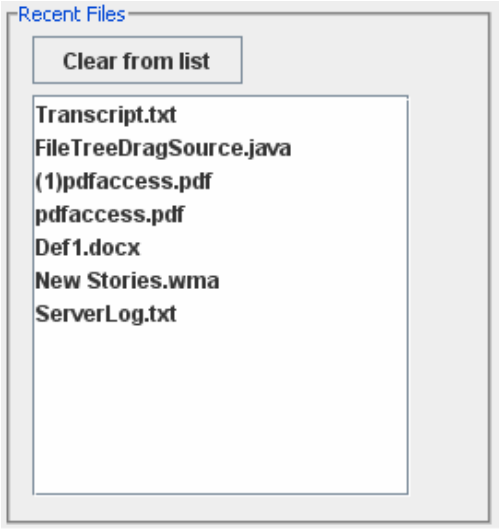


Figure 7: Recent Files or Files-Being-Used Panel

If the user opens a file from the JTree that already exists in the Recent File List (same file with the same path in the server), the client program will prompt a message box that tell the user that the file is already been opened. It will not request for a file transfer again. It will not open the file from the temporary folder. If the user wants to open the file, he/she can double-click on the file in the Recent File List and it will open the file for the user. By doing this, the user is not confused with the files that are already opened and files that show on the JTree with the same name, but in a different folder.

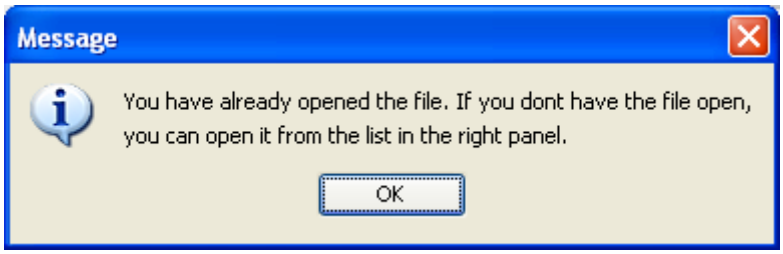


Figure 8: Opening an already open file prompt

3.3 File Names in the client

The client gets the files from the server and stores them all in one temporary folder. If the client opens a file, its name shows on the File-Being-Used List. However, if user browses through the folders in the server using the JTree and opens a different file with the same name as one of the previous file, the older file has a risk of being overwritten.

So, to handle the above mentioned problem, the client takes care of checking the file names in the list each time a new file is opened and if they have the same name, the new file name will be preceded by a "(n)", where 'n' represents the n^{th} file with the same name.

Although the filenames are stored in the client with a precedence of a number and brackets, the hastable still maintains the file path and the original name it has on the server. So, when restoring, sending or updating the file on the server, the server will still have the original name in the server, irrespective of which folder it is in and what name it showed to the user in File-Being-Used List on the client side.

The filename shown in the JTree on the client side will still reflect the original filename, and not the pseudo filename it shows on the File-Being-Used List.

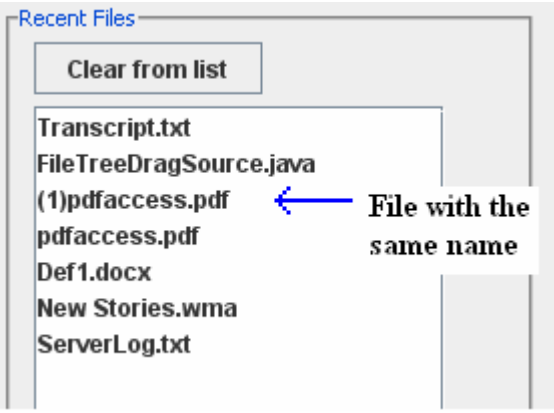


Figure 9: Filename with a pseudo "(n)" inserted before it.

3.3.1 Tooltip text of the file path in the List

The previous topic may have raised a question about how the user will know which file the filenames shown on the File-Being-Used List represents. This is solved by the following option that the user has.

The user can click on the filename on the File-Being-Used List and place the mouse on it. It shows a tool-tip text of the path of that specific file on the server. So, the user can actually know which files on the server the list represents. If the filename is a pseudo filename (the one that has numbers and brackets preceding it), the user can click on the file, place the mouse on it to know which file on the server it represents.

Figure 10: Tooltip text for the path of the file in the server

3.3.2 Maximum File Limit

The number of files the user can open and use at a particular instance is ten files. If the user clears one of more files from the list, he/she can open new files. But, if they don't, they will be prompted with a message box that tell the user that they have reached the maximum limit of files, he/she can use at one time.

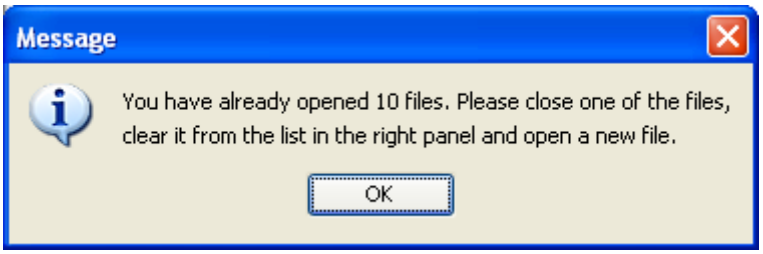


Figure 11: Maximum number of file limit reached prompt

The maximum limit is enforced to make sure the user is not jammed up with too many files at one time. If the recent file list capacity is unlimited, then the client might download as many files as it can and the user might be thrown into a situation the project is trying to solve – confused with lot of files from different folders. However, the maximum limit is debatable. It may vary from one person to other. If the number of files

limit is not appropriate, it has to be modified in the code, so the user can have more files open at a particular instance. Even though not implemented in this project, the solution for the problem would be to provide the user a functionality to change the maximum limit and make the client and the server compatible with the dynamic values that may be set by each user.

3.4 Auto-save of files

The user can open any number of files from the server. For each file that is opened, a thread is run by the client to monitor if any changes have been done to the file.

The Thread stores the last modified date and time. It checks it every ten seconds and verifies if it is the same. If it is greater than the previous last modified date and time, the file is modified within the last ten seconds. The File Monitor Thread requests the client program to update the modified file in the server. The client program requests for a file transfer with the server and it sends a copy of the file back to the server. The server replaces the file back in the specified location. At the back end, the file is sent to the server and the server replaces the file, but to the front end, the file is being updated with the modifications made to it. The client still has the file in the temporary folder and the user still has the file open. So, the user will not be able to see the file transfer. He/She is editing the most recent version of the file. Closing and opening the file back from the server will only send the updated version of the file.

This is the process that the client program goes through to auto-save the files that are on the server, when they are saved by the user on the client side.

Chapter 4 Server and Client

4.1 The Architecture

The Promond follows the Client-Server Architecture. The server accepts the client and when they are connected, the user can perform certain actions from the client side, which could initiate transfer of data between the client and the server. The figure below shows the prototype of the Promond Client and the Server.

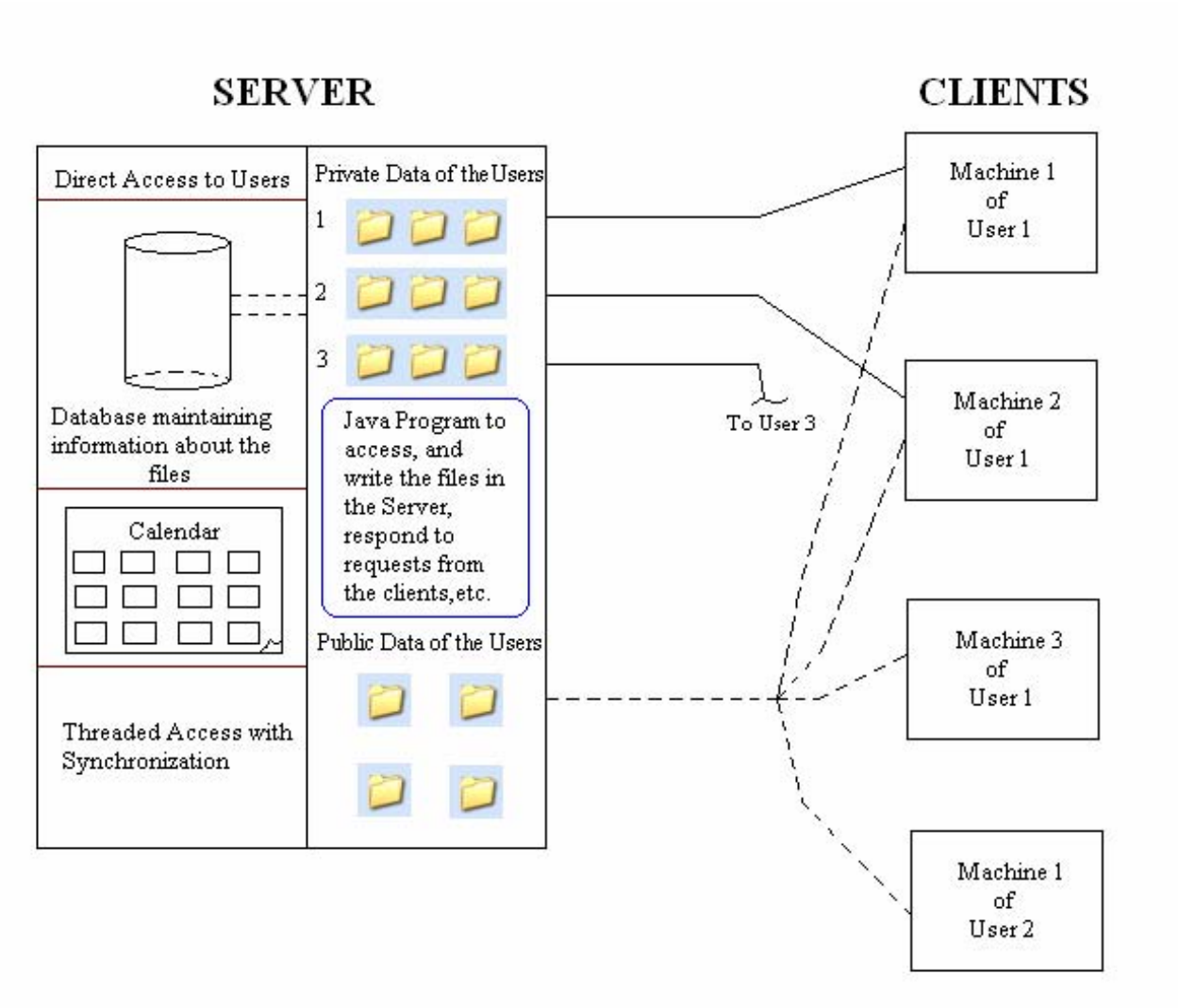


Figure 12: Architecture of the Promond Client-Server

4.2 Multi-Threading

The server programs runs in continuous loop to accept client connection requests. Each client connection is taken care of by a new Thread, which takes care of reading and processing the requests and sending responses back to the client. While this is happening, the server also waits for the next client connection request. In this way, the client connections are maintained differently in the same program but handled simultaneously. The requests and the responses do not interfere with each other.

4.3 Requests

The client sends many requests based on the action by the user in the client computer. These requests are identified by a common variable on both sides of the systems. For each request from the client, the server processes them and sends back the responses.

4.4 Responses

The responses are usually in the form of data sent back to the client depending on what the client requested for. Sometimes, it might even trigger a whole different method or class to handle the request, for example, a file transfer request would be processed by invoking the File Transfer class, which will directly send the file to the requested client.

The figure below shows the general flow diagram on both the client and the server side.

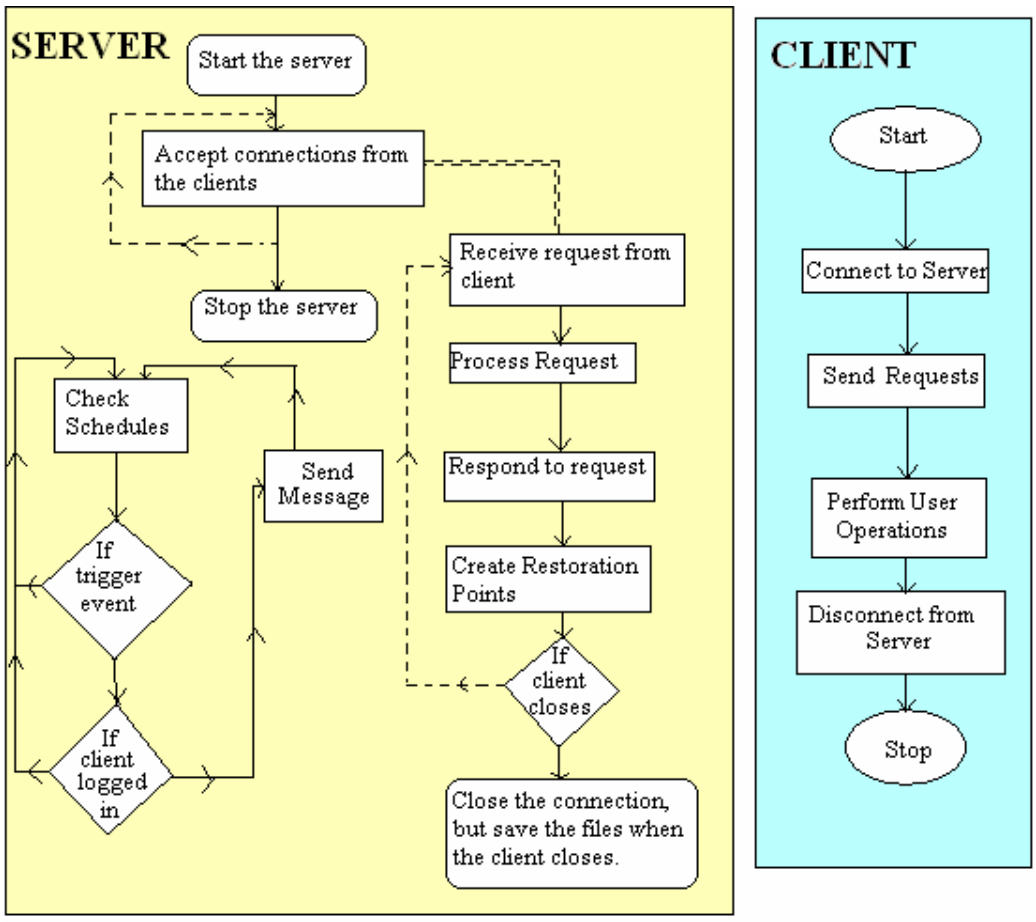


Figure 13: Flow Diagram for the Promond Server and Promond Client

4.5 Database

The server uses a MySQL database in which the following are stored:

User Details Table:

Fields: Username, Password, First Name, Last Name, Designation, Email ID

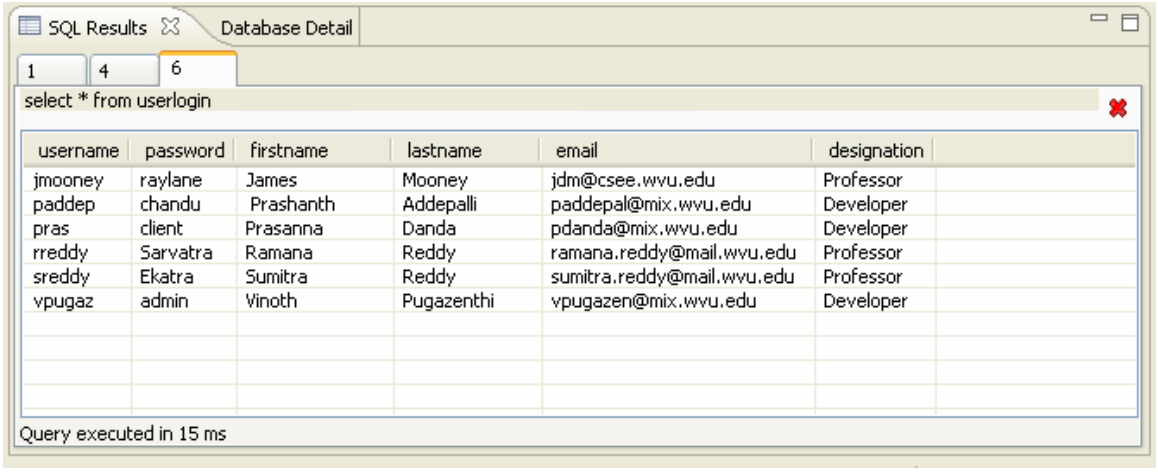


Figure 14: UserLogin database with sample records

File Details Table:

It contains the list of previously used files during the last exit from the client program.

The User Details database is used for the following:

i. When the login is requested by the client program, the server checks the username and password of the user in this table for authentication.

ii. The details such as the First Name, Last Name and Designation are sent to the client to display on the Details Panel in the client window.

iii. When the user sends the emails, the email id listed in this table is used as the "From-Address". The user need not type the "From-Address". It is taken from this table. However, the "To-Address" is retrieved from a different table altogether.

The File Details is used for only one reason and it is stated in the following:

i. Store and Retrieve the path on the server of the files used by the user, when he/she closes the client (The files listed on the Recent File List).

4.6 Client

The client program, when invoked will bring up a login window, successful login of which will get them to the client user window that has many components that help the user utilize the features and capabilities of Promond.

4.6.1 Login

The program uses a traditional login, where the user has to enter the username in the give username text box and the password in the password textbox. The password textbox is set with a password character "*", to hide the password when the user types it. The user is provided with an "Ok" and a "Cancel" button.

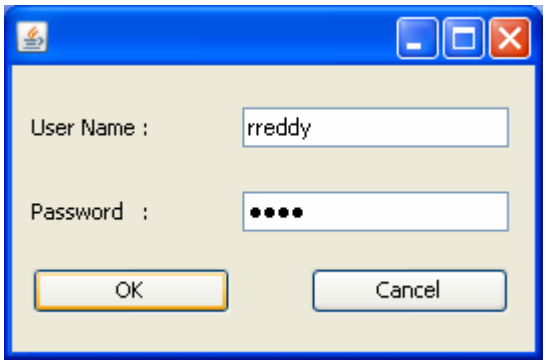


Figure 15: Login Screen

When he/she clicks "Ok" the username and the password are retrieved from the respective textboxes and sent to the server over the network. The server authenticates it and sends the response, whether the login succeeded or not. The password is case-sensitive. If the user login succeeds, the Client User Window will open and if the user login did not succeed, the client will prompt the user with a "Wrong Username/Password" message box. However, hitting the "Cancel" button immediately closes the connection with the server and exits out of the system.

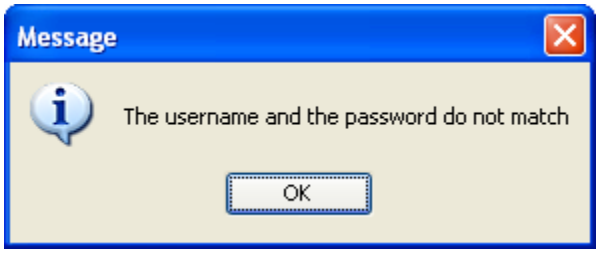


Figure 16: Wrong Username/Password prompt

4.6.2 User Interface

The client has a Graphical User Interface (GUI) for the user. The components in the user interface were built with Java Swing and Java AWT (Abstract Window Toolkit). The components use mouse-listeners to identify the user actions and generate the appropriate triggers.

The client window has the following:

i. An “Option” Panel

ii. A “File Access” Panel

iii. A “Recent Files” Panel and

iv. A “Details” Panel

Panels:

Option Panel:

The Option Panel basically consists of buttons that the user can click to work with the files selected in the file tree. The buttons are “Add”, “Rename”, “Delete” for the files and folders and “Email” for sending Email to the students that have taken the courses.

Files Panel:

The Files Panel has two JTrees in it, one for Private Files and the other for Public Files. These are representation of the files on the server.

Recent Files Panel:

The Recent Files Panel has a JList, which shows the recent files accessed by the user and the currently open files. It also has a “Clear from List” button to clear the selected file from the JList.

Details Panel:

For each user, a Details panel will display their username, first name, last name and the designation of the user in the Project or status of the user. It could be Developer, Administrator, Professor, etc. There is also a link provided to reach www.wvu.edu in the Details panel. It is a hyperlink to www.wvu.edu.

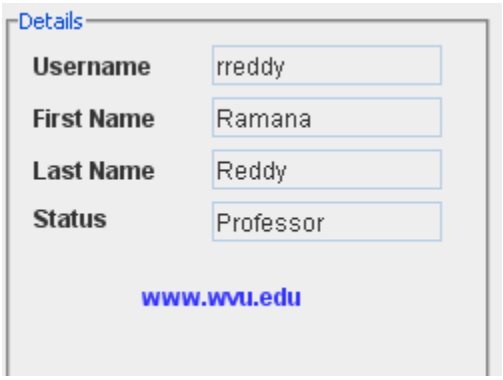


Figure 17: Details Panel showing the user information

4.7 File Transfer on User Actions

The file transfer happens through File Input and File Output Streams in the ports. There are two reasons for why file transfer should occur from the server to the client. They are:

i. The user double clicks on a file node in the JTree.

ii. The user logs in and the client needs the previously used files from the server.

For the first case, the client specifies the path of the file on the server, the user has double-clicked. And for the second case, the server sends the list of files it is going to send after the login.

There are three reasons why file transfer should occur from the client to the server. They are:

i. The user saves a file and it has to be updated in the server.

ii. The user clears the file from the File-Being-Used list and the file has to be sent from the temporary folder back to the server.

iii. The user exits from the client and the files in the temporary folder in the client have to be restored back in the server.

For all these cases, the client sends the file restoration request to the server.

Now that both the client and server are in knowledge of what the files are, the server sends random available port to the client. The client then invokes a File Transfer class, passing the file name and the port number the server has sent. The server also invokes a similar File Transfer class passing the port number it sent to the client. Inside the File Transfer class, two methods are present, one for upload and one for download. Based on which direction the transfer is going to take place, the appropriate method is invoked on each side. Then the file is sent over the network. It is retrieved and saved.

The server saves the file in the path specified by the client itself. The client keeps track of the file-paths on the server for each file it receives in a static Hashtable. So, when sending the file back to the server, it is easy for the client to just indicate the destination of the file it is sending and the server saves it back in the actual folder it sent the file

from. The client, however, does not need to get any information about where the file has to be saved, because it saves the files on a temporary folder created on the client computer and the server does not worry about it.

4.7.1 Port Availability

The server, for each request for a file transfer, has to send a port in the server, where the client can connect to. In order to do this, it has to first get an available (Open) port. It takes a random port and tries to create the server socket in it. If it returns an exception, it implicitly means that the port is being used. So, if the port can be created, it closed it immediately and send the port number to the server program, which in turn sends the port number to the client and then the File Transfer takes place in that port.

4.8 Signaling

On each event/ trigger, the client has to send the requests to the server. It may be one single request or a series of conversation between the client and the server. However, the client sends the messages to the server in the form of byte packets. The server in turn sends messages in the form of byte packets.

The system works in such a way that the client always initiates a conversation. After sending a message, based on the number of exchanges of messages for each request, the client may wait for the server to send a message. The client will not proceed unless it hears back from the server. The server operates in a similar way. After the end of the conversation, the client continues with the regular mouse-listeners, and does not send any requests unless triggered again. However, after the end of that conversation, the

server keeps checking for any messages sent from the client every 100 milliseconds and goes through a continuous loop checking infinitely until the client sends Exit/Logout message, in which case, it disposes the Thread for that client.

4.9 Events and Triggers

Each event is generated based on the action performed by the user. Each event that is generated triggers or invokes a method or a class that performs some action. Most of them communicate with the server to satisfy the request, either directly or indirectly, the implication of which is explained below.

The methods invoked may have been directly the cause of some action by the user or indirectly invoked by another method or a cause of a series of methods, the root cause of which will be the user.

There are two kinds of events that are generated in the client.

i. The mouse-listeners heard some click, double-click or click and drag by the users on the client user-interface.
ii. The monitoring threads could have checked for some changes made to files, invoking the client to update the file back in the server.

Chapter 5 Implementation of Promond

5.1 Development

The project involved various tools in the process of development, execution and testing.

5.1.1 NetBeans

The NetBeans 5.1 Integrated Development Environment was used to develop the Graphical User Interface for the client program in Swing and AWT.

5.1.2 Eclipse

The Eclipse 3.2 Integrated Development Environment was used to write, compile, run, test, debug and execute the project.

5.1.3 Environment (Windows)

The client program runs in the compatibility mode for JRE 5.0. The server should have MySQL 5.0 and the required JDBC drivers for the program to access it. Since the client sends only the requests and the server takes care of all accesses, and sends the responses back to the client, the client does not need any of the external drivers to access the database. The client can be installed in any of the computers installed with Windows Professional Version and can be used.

The client was initially started to be implemented for other Operating Systems also, addressing the portability issues and aptly modularizing and modifying the code. But, the client was finally coded for the Windows Environment as the target.

5.2 Installation

Server:

The server program should run on Windows XP Professional version. MySQL 5.0 should be installed on the server and the necessary JDBC drivers have to be installed. It should have Java Runtime Environment 6.0.

A folder with the name “Promond” has to be created on the root folder. The folders for each user with their username (Eg: Promond\rreddy) as the name of the folder have to be created by the Administrator under the Promond folder. Apart from these, there should be another folder called Public under the Promond folder.

The MySQL database should be inserted with a new record for each new user with values for the Username, Password, First Name, Last Name, Designation, Email ID fields. A new record has to be inserted with the username in the file details with the rest of the fields set to null (They need no be given any value). Since it contains the list of previously used files during the last exit from the client program, it will be populated with values after the very first logout of the user.

The Server.java is the main program. The class file of that program has to be run in order to accept clients and process their requests.

Client:

The client program has to be installed with the client program and can be run from any Windows XP Operating System (Home or Professional). The client has to have the Java Runtime Environment 6.0.

Both the server and the client have to be connected to the internet with acceptance in firewalls.

Chapter 6 Future Work and Conclusion

The project has been developed for Windows environment. In the future, I would like to address the Portability issues and overcome them, making the program suitable for as much environments as it could serve.

The program takes care of the auto saving of the files on the server when the user saves them on the client. It does not recognize if the files being used are closed or not. In the course of the project, I was able to monitor and detect if the file is being used or not for some file types. However, it could not be done for some of the file types (E.g.: ".txt" files). I would like to make the program detect it for all kinds of files, using which, other features, that this has been a hindrance to, can be added to the project.

The file limit of ten files for any user can be modified to provide an option for the user to actually specify the maximum limit that they may think would be appropriate for their kind of work. That will prompt for changes in the size of the table in the database that keeps track of the recently used files to support dynamicity.

This project has served the purpose of file access over the network in a specific sense and is intended for the professors. This project can easily be converted for the general users and can also be made commercially viable software. The concept of creating a standby for the files is more appropriate for users that work with lot of files and folders. The version control feature of the project replaces the use of any kind of removable devices and the action of mailing files to oneself to be able to access it in a different computer. This project can be a base for implementing more ideas and features.

References

1. TCP RFCs:

http://www.ietf.org/rfc/rfc793.txt

http://www.ietf.org/rfc/rfc2018.txt

http://www.ietf.org/rfc/rfc2581.txt

2. Computer Networking: A Top down Approach Featuring the Internet, 3rd edition. Jim Kurose, Keith Ross, Addison-Wesley, July 2004.

3. ., **"Eksarva: An Intelligent Collaboration Framework",** by Reddy R., Selliah S., Bharadwaj V., Yu J., Reddy S. and Kankanahalli S. in *Proceedings of the IEEE International Conference 'Intelligent Systems',* Varna, Bulgaria, June 22-24, 2004.

http://siplab.csee.wvu.edu/research/pubs/bulgaria_paper_final2.pdf

4. **"Eksarva: A Framework for Enabling Agent-Based Collaboration",** by Selliah, S., Reddy, R., Yu, J., Bharadwaj, V. and Reddy, S., in *Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE-2004) Agent-based Computing for Enterprise Collaboration (ACEC) Workshop,* Modena, Italy, June14-16, 2004.

http://siplab.csee.wvu.edu/research/pubs/ACEC-bharadwaj-eksarva-Revised.pdf

5. www.java.sun.com

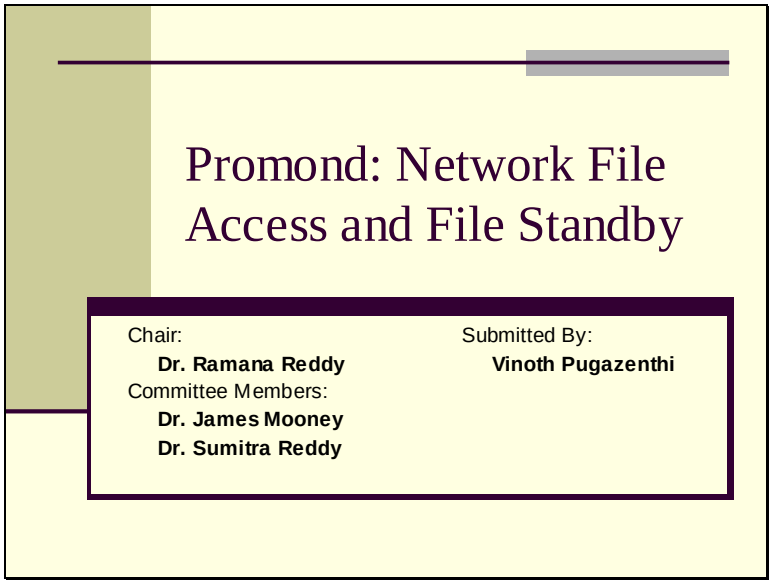
Promond References:

1. PROMOND – Email & Event Scheduler with Automated Triggered Messaging by Prashanth Addepalli, Nov 2007

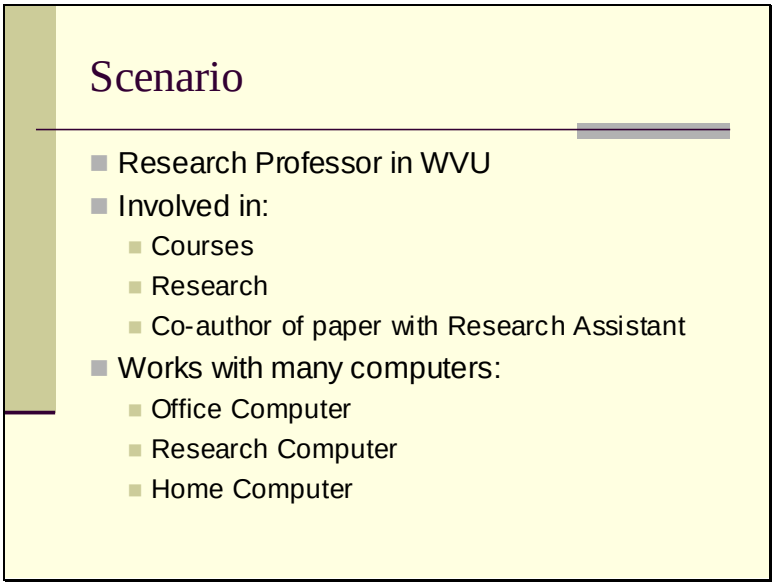
2. PROMOND – Network File Access and File Sharing by Prasanna Danda, Nov 2007

APPENDIX A

Slide 1



Slide 2



Slide 3

Scenario Description

- He goes to his office. Opens research files and course PPT files and edits them

Slide 4

Scenario Description

- Done with PPT, still working on research paper
- Class Time….
- Need to Close the files and go to the class

Slide 5

Scenario Description

- Go back to office, keep working on research files

Slide 6

Scenario Description

- Isn't it LUNCH TIME…. Where is the food? Got to go man….
- Close the reference papers and research files being used and go have food.

Slide 7

Scenario Description

- Wow….Strike an idea of writing a new paper with another Professor at lunch

Slide 8

Scenario Description

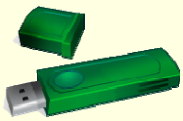
- Get back to work. Create common research paper to be edited with other Professor
- YYYupp… Need to share the file

Slide 9

Scenario Description

- Office hours over….
- Need to take copies of files in removable devices





Slide 10

Scenario Description

- Home sweet home

- Take some rest....But, WVU Professor are good in doing research even from home. Work again with the files

Slide 11

Scenario Description

- A good night's sleep

- Get back to work with the file (actually lot more files) the next day

Slide 12

Challenges faced each day

- May have to work with many files
- Should keep the last files used in mind
- Carry around in removable devices
- Care for and maintain the different versions of the files used
- Work with files with same names that belong in different folders
- Share and edit files simultaneously with co-authors

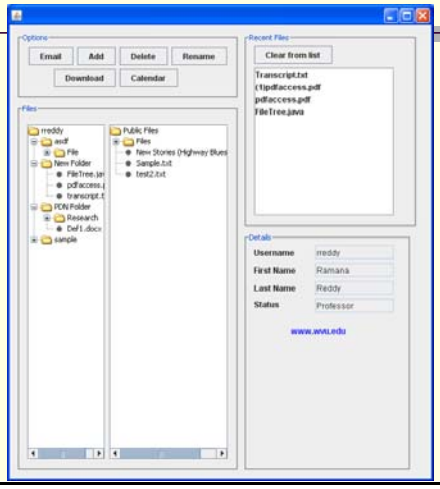
Slide 13

What Promond Does?

- Centralize the files on a server
- Create user account and login
- Open the files using the **Promond Client**
- Edit them on a **Temporary folder**
- Enable **Auto-Save** of files on server
- Keep track of **Recent Files** used for each user
- Open recent files from last login on each of the successive login

Slide 14

Promond Client



Slide 15

Logistics – Login

- Enter the Username/Password

- Client checks with server database for authentication

username	password	firstname	lastname
jmooney	raylane	James	Mooney
paddep	chandu	Prashanth	Addepalli
pras	client	Prasanna	Danda
rreddy	Sarvatra	Ramana	Reddy
sreddy	Ekatra	Sumitra	Reddy
vpugaz	admin	Vinoth	Pugazenthi

Slide 16

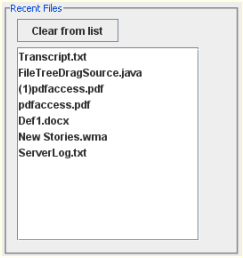
Logistics – File Access

- User can open private files and public files
- For each file open, they will be downloaded using FileInputStream to a ***TEMP*** folder
- The file will be opened from the temp folder
- The files are monitored for the last modified timestamp using Threads

Slide 17

Logistics – Recent Files

- Files used in last login are downloaded to the temporary folder in the client
- The files opened are maintained in a list visible to the user in a panel



Slide 18

Logistics – Recent Files (Cont….)

- He/She can open a closed file from the list by double-clicking on it
- Files with same name, but from different folders are shown with a number preceding it

Slide 19

Logistics – Recent Files (Cont….)

- How will the user know which of the two (or more files) are from which folder?
  - Answer: The user can click on the file and place the mouse on it. The tooltip text will show the path of the file on the server

Slide 20

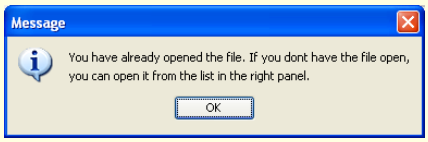
Logistics – Recent Files (Cont….)

- The user can clear any file from the list using the “Clear from list” button

Slide 21

Logistics – Recent Files (Cont…)

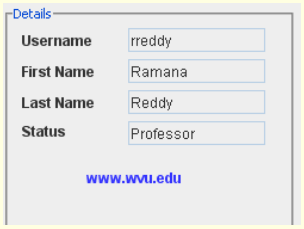
- An example condition:
  - File opened
  - Application using the file is closed
    - File still in Recent File List
  - Open the same file from the File Tree
- The client prompts the user to open the file from the Recent File List



Slide 22

Logistics – Details Panel

- The Details Panel show the details of the user (Professor) and also provides a link to www.wvu.edu



Slide 23

Logistics – File Monitor

- The monitor keeps checking for last-modified time stamp of the files that are open
- If they change, then that particular file is sent to the server for updating
- The file is still open for the user for further editing. The monitor thread will still be running

Slide 24

Logistics – Exit

- On each exit, the open files are sent back to the server
- The server path of the files in the list are saved in the database for opening in the next login
- The client program exits

Slide 25

Casting (Where do these fit in?)

- The user's recently used files are tracked and a standby is created at each exit.
  - He/She does not need to keep track of recently used files
  - Need not to hunt for last used files. It automatically opens it in the next login

Slide 26

Casting

- The files can be accessed from anywhere
  - The files are on the server. They can be seen and accessed from any computer connected on the internet
- The user does not need to carry the files around in removable devices
  - The files are saved on the server. So, it needs no transportation

Slide 27

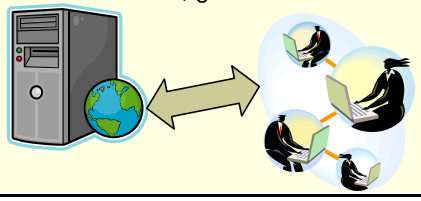
Casting

- The version control for the files are maintained
  - There are no duplicates
- The user need not upload the files on each save
  - The auto-save feature updates the changes to the file when saved on the local computer

Slide 28

Story (Revisited….)

- The Professor
  - Goes to the office – Works with files
  - Takes the class
  - Returns to the office – Works with files
  - Goes to lunch – Discusses about Research paper
  - Goes to the research lab – Works with files
  - Leaves offices, goes home – rest – back to files



Slide 29

Promond –Network File Standby

- A lot of use of files. But no problem about files such as:
  - Universal access of files
  - Recent Files used
  - Version Control
  - Auto-Save
  - No need for removable devices
- Promond takes care of it as an agent for the Professor (User)

Slide 30

Protocol and Applications Used

- Client and Server Programs: Java
- Database Used: MySQL
- Communication between the server and the client: TCP/IP
- File Transfer: File Input and File Output Streams
- Connections and Monitors: Threads

Slide 31

Exceptions and Problems Faced

- The program could not close the files after the user closes the client
- The program cannot check if the file is being used
- Solution tried:
  - Try to check if a lock is held on the file by any application
  - If yes, the file is being used
  - If no, the file is not being used

Slide 32

Exception

- Microsoft Word, Adobe Acrobat and other application hold a lock on the files opened in them
- Textpad, Notepad and other Text Editors do not hold a lock
- What is the problem?
  - The check for lock on files will not return correct values (boolean) in the latter point

Slide 33

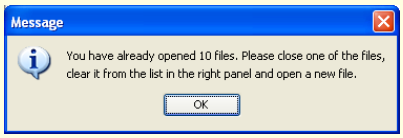
Exception – Closing the file

- How does it affect the program?
  - The trigger for monitoring the file is "***SAVING*** the File"
- Alternative if exception is overcome
  - The trigger can be "***CLOSING*** of the file" instead of "Saving of the File"

Slide 34

File Limits

- Virtually can be implemented with **NO FILE LIMITS** for the recent files
- Too many files opened at the same time on each login may be annoying and confusing
- Limit is enforced (Statically: 10 files)
- If more than 10 files are opened in one session, a prompt will appear



Slide 35

Future Work

- Make it portable over different environment
  - Many of modules and methods are Windows environment – specific
- Track and Close the files for the user when the client is closed
- Make the file limit dynamic and let the user decide the number of files to be accessed in each session
- Transfer the data over Secure Socket Layer to enhance security in the project

Slide 36

Conclusion

- Even though Promond is fine tuned for Professors, the broader concept can be used for any type of user
- It partially achieves replacement for carrying one's laptop or removable devices around to wherever the person goes

Slide 37

References

- TCP RFC:
  - http://www.ietf.org/rfc/rfc793.txt
- Computer Networking: A Top down Approach Featuring the Internet, 3rd edition. Jim Kurose, Keith Ross, Addison-Wesley, July 2004.
- Promond: File Exploring and Sharing over the Network, by Prasanna Danda, Nov 2007
- Promond: Event Scheduler and Email Client, by Prashanth Addepalli, Nov 2007
- www.java.sun.com

Slide 38

Acknowledgement

- Dr. Ramana Reddy
- Dr. Sumitra Reddy
- Dr. James Mooney

- Parents and Sister
- Friends
- Mr. Tim Mitchem
- Dr. John Oughton