April 2006

Windows Based FTP Server with Encryption and other Advanced Features

Nicholas J. Byra
Worcester Polytechnic Institute

Project Number: MXC MQP 1096

Windows Based FTP Server with Encryption and other Advanced Features.

A Major Qualifying Project Report

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

in Computer Sciecne

by

Nick Byra

Date: April 27, 2006

Approved:

Prof. Michael Ciaraldi

Abstract

I created a fully functional, open source FTP server for the Windows platform, complete with encryption, bandwidth quotas, bandwidth throttling, remote administration, ratios and IP checking. My server uses threads to manage each connection. The configuration and logging files are done in XML and the C++ source code is fully documented.

Table of Contents

1. Introduction

This project created a Microsoft Windows FTP server. FTP (**file transfer protocol)** is used to move files from one network system to another, as well as all the ability to send and receive a file. FTP also includes additional features that most users also expect; such as bandwidth monitoring, credits, bandwidth throttling and encryption, among others.

This project was developed because many people share files with friends and usually the easiest way to do this is to set up an FTP server on their computer. The problem is the majority of users run Microsoft Windows as their primary operating system. While numerous open-source FTP servers exist for the *nix environment, few have been created for Windows platforms. Running a good FTP server does not usually motivate users to switch their operating system. They use Windows for their FTP or nothing. People running both a Windows and *nix computers often discover files that they want to share are located on the Windows machine, meaning that they need a Windows server. Some free open source Windows servers are available ("War FTP" and "Whitehorn FTP"), but have problems with stability, user friendliness, and they often lack features considered essential; for example, encryption. Also, many of the open source programs are poorly documented, making it very difficult to modify, or even understand them.

The big problem with available Windows FTP servers that are stable and have the desired features are that they cost money. Many users interested in this software are not willing to license it with prices over $80. This leads to people pirating the software, or using something that really does not suit their needs. There seem to be some types of applications that people feel they should have to pay for in order to use them and others they think should be available for free. FTP servers seem to fall on the side of the software that people think that they should not have to pay for, such as word processors or media players. There are already good free word processors and media players for Windows, Open

Office and Winamp respectively. There is now a need for a good FTP server with all of the desired features.

2. FTP Background

File Transfer Protocol is a relatively old protcol, the RFC for it dates back to 1985. This RFC is numbered 959 and deals with the commands:

```
USER <SP> <username> <CRLF>
PASS <SP> <password> <CRLF>
ACCT <SP> <account-information> <CRLF>
CWD  <SP> <pathname> <CRLF>
CDUP <CRLF>
SMNT <SP> <pathname> <CRLF>
QUIT <CRLF>
REIN <CRLF>
PORT <SP> <host-port> <CRLF>
PASV <CRLF>
TYPE <SP> <type-code> <CRLF>
STRU <SP> <structure-code> <CRLF>
MODE <SP> <mode-code> <CRLF>
RETR <SP> <pathname> <CRLF>
STOR <SP> <pathname> <CRLF>
STOU <CRLF>
APPE <SP> <pathname> <CRLF>
ALLO <SP> <decimal-integer>
     [<SP> R <SP> <decimal-integer>] <CRLF>
REST <SP> <marker> <CRLF>
RNFR <SP> <pathname> <CRLF>
RNTO <SP> <pathname> <CRLF>
ABOR <CRLF>
DELE <SP> <pathname> <CRLF>
RMD  <SP> <pathname> <CRLF>
MKD  <SP> <pathname> <CRLF>
PWD  <CRLF>
LIST [<SP> <pathname>] <CRLF>
NLST [<SP> <pathname>] <CRLF>
SITE <SP> <string> <CRLF>
SYST <CRLF>
STAT [<SP> <pathname>] <CRLF>
HELP [<SP> <string>] <CRLF>
NOOP <CRLF>
```

There have been several additional RFCs that have involved extentions to the ftp protcol.

The FTP RFCs include RFC 2228 which deals with security extentions for FTP. This includes the commands:

```
AUTH (Authentication/Security Mechanism),
ADAT (Authentication/Security Data),
PROT (Data Channel Protection Level),
PBSZ (Protection Buffer Size),
CCC (Clear Command Channel),
MIC (Integrity Protected Command),
```

```
CONF (Confidentiality Protected Command), and
ENC (Privacy Protected Command).
```

Also included is RFC 2389 which deals with the feature negotiation mechanism, it adds the commands:

```
FEAT
OPTS
```

3. Features

This server, named horseshoeFTPd, has all of the common features that a user would expect to find in any good FTP server. These features are the standard commands that allow downloading, uploading, and resuming, as well as some additional features that are included in a lot of the more advanced FTP servers.

Feature	Description
Upload	Allows the user to upload a file to the server.
Download	Allows the user to download a file off of the server.
Resume	Allows the user to resume a broken transfer where it left off.
Encryption	Allows for the traffic over the network to be encrypted with SSL.
Bandwidth throttling	Allows a max download and upload speed to be set for the users.
Bandwidth Quotas	Allows a limit to be set for the amount of bandwidth a user or the server can use during a set interval.
Ratios	Allows a ratio to be set that allows a user to download a set amount based on how much they upload.
SITE commands	Allows for remote administration of the server though commands on the server.
IP checking	Allows for checking of the IP that a user is connecting from to see if it is a authorized one.

FIGURE 1-1 Chart of features.

The server administrator has the option of deactivating a user's account. This is useful if you don't want that person to be able to access the server, but you don't want to have to delete the user's account.

There is also the option of turning FXP on or off; FXP is a server to server transfer protocol, it allows the server to send or receive files with another server, as opposed to a client. Another option for FXP is IP checking. With this set to true, checking will occur to see if the IP of the other server involved in the FXP operation is in the user's list of accepted IP addresses. If the IP is not, then the user will not be allowed to engage in a transfer with that server.

The administrator will have the option of setting an expiration date for a user account. This gives the administrator the ability of having the account automatically set itself to not active on a certain date. This feature will be useful if the administrator only wants to give a user access until a certain time; the administrator will not need to remember to deactivate the account.

Each user supports the ability to have a ratio; this is a ratio that determines the number of credits the user gets for uploading a file. These credits are then used by the user to download files. If the user has a ratio of 1/3 and they upload 100 megabytes, the user will receive 300 megabytes worth of credits. The user will then be able to download a file or multiple files up too that size. The ratio can be turned off if the administrator does not want to use it for a particular user, then that user will not have a credit system for the FTP. There is a free file size setting available, this will allow an administrator to specify the maximum file size that a user can download and have the download not count credits. If credits are not on for a user then the free file size does not matter for them.

There are also bandwidth quotas available. To use this, set a maximum amount of bandwidth that a user can use in a session. A session is the amount of time set by the bandwidth limit type; set in daily increments. The users bandwidth is reset to zero after the specified number of days. If a user uses more then their limit before the bandwidth resets, they will not be able to download or upload any more files until there is a reset.

In addition to the amount of bandwidth that a user can use, the administrator can also configure the maximum speed at which a user can download or upload files. There are separate settings to set the max allowed upload and download speed for a user. This can be set to any number of bytes per second, or can be turned off completely if the administrator does not want to limit the transfer speeds of the user.

The final configuration options that can be applied to users are IP rules. These are IP addresses that the user is allowed to connect from. This setting supports wild cards and ranges. The IP rules

specify only allowed IP's; there is no way to add an IP rule to dissalow an IP, an IP will be disallowed if it is not in the IP rules. So if an administrator wanted to not allow the IP 77.77.77.77 for a particular user but wanted to allow everything else they could just add the IP rule "*.*.*.000-076" and the rule "*.*.*.078-255" to accomplish this goal. There is no way to turn off IP checking so if the administrator does not want to use it they just add the rule "*.*.*.*" and everything will be allowed.

For configuration of global server settings there are also the same type of bandwidth quotas as available for the individual users. This is useful if the server administrator has a bandwidth limit for the day, week, or month and does not want to go over it, they can set the server to not transfer any more data once it gets to the limit that is set.

The administrator can also configure the range of ports that will be used for passive connections. They can use this setting to disable passive mode by setting the lower bounds of the range to a value greater then the upper bounds; this method is not officially supported, but it appears to work. Testing on this option hasn't been done so because of that it will not be officially supported.

The administrator can configure the max number of users allowed on the FTP at any given time. If any additional users attempt to connect, they will be disconnected with a message alerting them that the maximum number of users are connected.

The idle timeout for the server can be set; this is the amount of time that the server will wait for a command from a user before it assumes that they are not going to do anything else and disconnects them.

The size of the packet used to send data can also be configured. This option isn't expected to be used much; it just makes more sense to leave it as something that can be changed than to hard code it into the program.

The port that the FTP server will listen on can also be set by the administrator; the default is 21, but it can be set to use any valid port number.

The server has the option to use SSL encryption. In the current version, this option is only available as a global on or off feature, so either everyone has to use encryption or nobody can use encryption. In addition to encrypting the data being transferred, the password is also encrypted in the XML file that it is stored in using an SHA1 hash.

Finally, this server supports SITE commands. These are commands that you can issue to the FTP server to remotely administrate it. This accommodates operations such as add a user or group, delete a user, change a user's settings, change the global bandwidth limit, and other settings. Full documentation on these features is available in **APPENDIX A**.

4. The Modules

HorseshoeFTPd is broken down into several different modules. First is the FTP options module, it controls the global server settings and also deals with the individual session data for the users currently logged into the server. Then there is the TCP module that does the handling of all the network operations. This includes making both passive and active connections, as well as sending and receiving files. The final thing that the TCP module handles is reading and sending the telnet commands that the FTP server uses to operate. The next module is the XML module; this module deals with all of the XML configuration files. This includes the configuration file for the server itself, the users and the groups, and the directory structure that the server uses. The XML module handles reading in these files, creating their associated objects, and assigning the correct values in them. In addition to reading in the files, the XML module also handles taking pre-existing objects and updating the XML to reflect any changes that have been made to the data they contain. The user / group modules handle all of the individual user and group settings and statistics. The directory module maintains a tree of the directory structure that the server is using; this includes the virtual directories, the files and directories that were uploaded by users, as well as the users that have rights to these files and folders. The authorization module maintains a set of functions that are used to check a user's right to do something in any particular instance, like deleting a file. The parser module does the parsing of the commands that the FTP receives and checks the command for the correct syntax, checks the rights for the action, and then executes the correct action. Finally the logging module handles the logging of all the relevant actions on the server. The logging is done in XML format, but to avoid the overhead of the XML module, a simple append file is used instead that writes an XML formatted string to disk.

4.1 FTP Options

The FTP Options module is the most used module in the FTP server because it is responsible

for keeping track of all the local user variables that are necessary when any particular user is logged onto the server. This information is stored in an array which is allocated when the FTP Options object is created; the array is made to the size of the max number of users that are allowed on the FTP. Because of this, it is unwise to select a max user number that is really large, for example setting it to 999,999 in an attempt to have an almost infinite amount of users. The documentation for horseshoeFTPd states to use a realistic value for the setting of max users. When a user connects to the server, they are assigned a session id that is referred to as a "sid" in all of the documentation. This sid is an integer between 0 and the max users value. For each session, there is a pointer to the User object that belongs to the sid. If there is no user logged on yet in the session, the user variable is set to null. It should be noted that if the same user is logged on under more then one session, the pointer will all be to the same user object, not to different ones for each session. The data socket is stored for each session; this is the socket that was created for the data connection. This value is also used to tell if there is an active data connection for the session; if there is not an active data connection, then the value of the data socket will be null. In addition to the data socket, there is also a pointer to a ssl_cnx struct for each session. This struct is something that openssl uses for encryption. If encryption is on, the pointer will be to the struct, if encryption is disabled the pointer will be set to null. This is the same as it is with the ssl_cnx struct for the command connection. When it comes to the directory tree there are a couple items that need to be maintained for each session, the most important is the virtual folder that the user is currently in. A virtual directory or a virtual folder is something that represents an actual directory on the harddrive, the adminsitrator uses them to create a directory structure for the FTP server. The location of the current working directory is held by a pointer to that folder in the directory tree. The current subdirectory is also maintained, this is done with a string of the path of the current folder from the virtual directory. For example, if the user were in the directory "/test/redsox/horseshoe/" and the only virtual directory is the root which is "/" then the subdirectories

string would be “test/redsox/horseshoe”. If the directory that the user was in were “/vdir/blah/stonecoast/” and “vdir” was a virtual directory then what would be in their subdirectories string is “blah/stonecoast”. This string is reallocated every time it is set, so there is no limit to the length of the subdirectory. An incoming file variable is maintained for every session, in terms of memory management it is handled in the same way as the subdirectory variable is managed. The incoming file variable's main purpose is to hold the name of the file that the user is currently uploading; it also doubles as a holder when a file is being renamed, since the command to rename a file is done in two parts. The first part of the renaming command is giving the name of the file to be renamed, and the second part is giving the name that the file is going to be renamed to. The incoming file variable is used to hold the name of the file to be renamed in the time before the command of what to rename it to is issued. There is a type variable that is maintained to tell if a file should be transferred via a binary or an ASCII mode, this is almost not needed because most clients send the directory listing using ASCII mode and everything else using binary mode. The last thing that is maintained for each session in terms of dealing with files is a resume value. This value is used for resuming files both on uploads and downloads. When a user wants to resume a file they first issue the REST command, the argument in this command is the location in the file measured in bytes that they want the operation on the file to be started. This value is stored in the resume at value, when there is no value set, the variable is at zero. The last two session values maintained are the remote IP address that a user is connected from and a boolean value that, when set to true, tells that session to log off the user and shut down. The boolean value is used for both kicking a user from the server and shutting down the server.

Besides the session information, the FTP Options module holds global server settings and statistics. One of these is the name of the FTP server, for example if the administrator were to name the FTP “Serenity” when a user first connects to the server it sends the message “Welcome to Serenity”. This is the only application of the name of the server so it is not a critical setting. The next

thing that it maintains is the max number of users allowed on the server, as mentioned above this variable is used when making the arrays to hold the session id information. The server needs to be restarted if this variable is increased because there is no way to guarantee the pointers to critical session data won't be moved, there is no need to restart if the max users is decreased. Then there is the max connections allowed per IP. This is the max number of times any IP can connect to the server. The idle timeout is the time that a user must not issue a command over the telnet command session for them to be logged off for inactivity. This is not enforced if there is activity over a data connection; it only applies if there is no activity over a data connection. There is a port range that is used for the PASV command. When someone issues the PASV command the server establishes a listener on a port and then informs the client to connect to that port. The port that is used for listening is one that is in the range specified by this setting. It will be selected at random, if no free port that can be found in the range then an error will be returned and the PASV connection will not work. In the current version of the horseshoeFTPd the SSL is controlled by a global on/off boolean, this will probably be changed in the future when AUTH tls and AUTH ssl are implemented. The port that the server is listening on and the IP the server has binded to are also maintained in this module. Currently this server supports listening on only one IP address at a time, so if you are running horseshoeFTPd on a computer with more then one IP address you need to run multiple instances of the server in order to be able to have it listen on more then one of the IP addresses. The packet size variable is the size of the packets that the server is going to be sending in bytes. This packet size is used for both files and directory listings; the value is defaulted at 4096. The free file size variable holds the maximum size of a file that a user can download without having it require credits. What this means is that credits are not decreased when the file is download. This is one of the variables that are also maintained for each user. In this case whatever value is larger is used. If the user has a free file size of 5000 and the server has one of 4000 then that user has a free file size of 5000. If the server has it set at 1000 and the user has it at 50 then

the user has a free file size of 1000, all users on the server will have a free file size of at least the server free file size. The final thing kept track of is bandwidth statistics; there is the bandwidth-this-session, which is the amount of bandwidth that has been used since the value was reset to zero, and there is a total of both download and upload bandwidth. The bandwidth-this-session is reset when the current system time is greater then the reset-at time. This time is represented as an integer value of the number of seconds since January 1st 1970. What this reset at value is depends on the bandwidth limit type that has been set. If this value of the bandwidth limit type is set to 0, then there is no reset value; this means there is no limit, so the value does not need to be reset. If the value is greater then 0 it represents the number of days in between resets. If the administrator wanted to have the bandwidth reset every week, they would use a value of 7; if they wanted it to reset every month, then they would use a value of 30. Currently this will reset at the time of day that the setting is set, this will probobly be changed to reset at midnight in a later version. The bandwidth limit is the number of bytes than can be transfered in the time determined by the limit type. If users move more data than the limit in the allotted period, then all users attempting to download or upload anything will receive an over quota error and they will continue to get that until the reset at time is reached and the session variable is reset to zero.

4.2 TCP Manager

The TCP manager handles all of the network operations for the server; this starts off with establishing a listener. The listener is established on the port that is specified in the FTP options object. When a remote client makes a connection to the server then that connection is threaded off into another thread that handles that connection. Once a user has connected to the server the first thing that they are going to do is send data over the telnet connection that has been established. This operation is handled by the program's readline function. What this function does is read data on the telnet connection until the data being sent includes an end of line character. The sequence carriage return, line feed, “CR LF”

is expected but only the line feed is required in this implementation, the command will work if it ends with “CR LF” or “LF”, the line must end with LF. What every client should do is send a line to the server that ends with an end of line character, then wait for a response. Because of this there should never be a case where there is data after the end of line character, but if there is it will be ignored when sent to the parser. If there is no end of line character received in the timeout period the server will disconnect the user for a timeout and close the thread. Once a line has been received from the client the server needs to send a response telling the client how it responded to the command, this is done with the server's sendline function. The sendline function sends the response of the server to the client; the response consists of a positive 3 digit integer and then a text response. For example when a client connects to the server successfully the server sends it the code 220 with a text response of “Welcome to (name of server)”. The integer code is the only thing that is actually important to the client in most cases, the text is just for human readability, there are a few exceptions. These exceptions are the PASV command and the PWD command. In those cases important information is sent in the text, the IP and port that are to be connected to with the PASV command and the name of the current working directory with the PWD command.

The next most important thing that the TCP manager handles is establishing passive or active connections for data transfers. If the client sends the command PASV then a passive connection is going to be established. This means that the server is going to start listening on a port and wait for the client to connect. The port that will be listened on is picked at random from the range that is specified in the FTP options object, the server IP and port being listened on is sent back to the client telling it exactly where to connect. The accept function that waits for a connection will block for 60 seconds waiting for the connection to be established, if there is no connection in that time an error will be returned. This 60 second timeout is hard coded into the program. Originally there was no timeout and the server would just block until there was a connection made. This was changed because if the client

issued a PASV command and then did nothing after that, for example what would happen if the client's computer crashed during the operation, the thread would never unblock. In addition to a thread doing nothing there would be an instance of that user logged on that would never log off. This would be a problem if that user was only allowed to have one logon at a time; they would not be able to reconnect to the server until it was restarted. It would not even be possible to kick a user if their thread was constantly blocking. With the hard coded timeout value infinite blocking should no longer be a problem and there is no reason under normal circumstances that a connection should take anywhere near 60 seconds to be established after the PASV command. So there should never be a problem of this timeout occurring when there is nothing wrong.

When it comes to the active connection the client will send the PORT command to the server followed by the IP and the port that the server is to make a connection to. After either an active or a passive data connection has been established one of three things will happen, a directory listening will be sent to the client, a file will be sent to the client or the client will send a file to the server. If the operation is a file being sent to the server then the function saveFile will be called. This function will take the name of the file that is being sent and either create it or append it depending on the value of the resume at variable that would have been set with the REST command. The file will either be opened in binary or ASCII mode depending on the type that has been set. The function will then read in the data that the client is sending and write it out to disk. There is a hard coded 60 seconds timeout for this connection, if there is no data received for 60 seconds the connection will be closed and an error will be sent to the client saying that there was a timeout on the connection. The file will be considered complete when the client closed the data connection.

What happens when a file is sent from the server to the client is very similar to what happens when a file is sent from the client to the server. The only difference is that when the server is on the sending end it is reading the file form disk and sending it over the socket as opposed to reading it from

the socket and writing it to disk. The final activity that happens on a data connection is the sending of a directory listing. When this is called the contents of the directory that is represented by the current directory and the sub directories of the current directory are sent to the client. If there are any virtual folders in that directory they are included in the directory listing. Originally the directory listings where sent one line at a time, this turned out to have a lot of overhead when a large directory listing had to be sent, so now the directories are sent in packets of the packet size that is specified in the FTP Options object.

4.3 XML Manager

The XML manager module is what handles all of the configuration files. The server uses 4 different configuration files; these are the server options file(options.xml), the users file(user.xml), the groups file(group.xml) and the directory file(dir.xml). The XML manager is responsible for reading in these XML files from the disk, parsing them and assigning the correct values to the appropriate objects. All of the parsing is done using xerces which is an XML parser that was developed by the Apache group. For the act of reading in a file a XML Manager object is created and passed the name of the file that is to be opened. In the case of a user or a group there are 3 different operations that can be done. The first is creating a user object for one of the users stored in the XML file. The second thing is save changes to a user, this is done by passing the user object to the function and then all of the appropriate changes are made and written out to disk. The third operation is creating a new user from a user object that you pass it. The way this is dealt with is it clones one of the users already in the XML file and then changes the values in it to match those of the new user. If for some reason there were no users in the file when this operation was called then the function would not be able to create a new user. Since the system will not allow you to delete the last remaining group or user and there is a default user and group in the XML files when you start the server for the first time this should never be a problem

unless someone manually edits the XML file. For the server options the only thing available is the ability to load the settings into an object and to take an object and save the settings to disk. Since you do not need more then one set of server settings there is no reason to have he ability to create another instance of server settings.

For the directory listings the XML manager has the ability to recursively read in the directory structure from the XML file and create all the tree nodes for it. This includes all of the folders, groups and files. This function will only need to be called at startup and when it is called it does a check to make sure that all of the files listed in the XML file still exist. If the files do not exist, those file nodes are not added to the tree. For writing out the directory structure a pointer to the top node in the tree is taken and the XML document is recursively created from the in formations in the tree. This is a bit different than how it handles saving the other files because it creates this from scratch every time.

4.4 Directory Tree

The directory tree module is responsible for maintaining the directory structure of the server. This is made up of three different objects, there are the folder objects, the group objects and the file objects. The folder objects are what make up the virtual directories of the server. Each folder object has a location variable that represents the location on the hard rive of the folder, then it holds the name of the folder as it is seen in the virtual directory. The tree is structured with one root folder at the top then any number of folders on each level below that. Each folder has at least one group attached to it. The groups tell what permissions users have in that folder. By default every folder is created with a group of level 1000 with access of 1, what this means is that people in groups with a level of 1000 (the siteop group has this by default) have read privileges to that folder. The access levels are 1 for read, 2 for write and 3 for delete. Each folder must have at least one group but does not have a maximum number of groups; each additional group is appended onto a linked list. In addition to having a group,

each folder can contain files. The number of files that a folder contains is going to be anywhere from zero to infinity. The reason a file gets put in a folder is if a user uploads it, a file object is then created in that directory to take note of the upload. The file object consists of the name of the file. The path to the file from the root of the virtual directory, the name of the user that uploaded it and the access level of the user that uploaded it. File objects are also created when a user creates a directory. The file objects will stay in the folder as long as the files are still one the hard drive, they will be deleted from the folder either when they are deleted by the ftp server itself of when they are not detected when the directory tree is created by the XML parser on startup. There is a chart of how this tree is structured in **APPENDIX B**.

4.5 Parser

The parser module takes the commands that are issued by the client and parses them to determine the proper action that needs to be taken. If the command given is a valid one, then the correct actions will be taken for that command, if it is not a valid command or the syntax for that command is not valid then an error will be returned.

When the USER command is received the argument is the username that is trying to log on. The parser will check first to see if it is a valid user name. If the name is valid it will then check to see if that user has used all of their logins, this includes total logins and logins per IP. If the user is not over the total it will then send the response that it is waiting for a password. All users on this server require a password, anonymous or password less logins are not available. This is because this server is designed for people who have security in mind; anonymous login is not something they would need. If the user can not login it will return the appropriate error code.

When the PASS command is received the argument is going to be the password that the user is trying to log in with. The password will be hashed with the SHA1 algorithm and compared with the

hashed password that is stored with the user object that would have been created in the previous command of USER. If the password is a match the user will be logged in, if the password is not a match then the user will be disconnected and receive the error of password invalid.

The CWD command will change the current working directory to the one that is specified in the argument. If the argument begins with a "/"then it is to be assumed that the path being given is a complete path starting from the root of the directory tree. If the argument does not begin with a "/" then it is assumed that the path continues from whatever the current working directory is. If the path is a valid one and the user has rights to access it then the current working directory will be changed to that one that was passed in the argument. If it is not a valid directory then the users current directory will not change.

The CDUP command takes no arguments and simply moves you up one directory in the tree. If you are as high up in the tree as you can get either because you are at the top of the tree or because you do not have access to the directory above you then you will not move up any further and then you will get an access denied error or folder not found error depending on what the problem was.

The QUIT command takes no arguments and does exactly what it sounds like it does, it will log you off and close the connection.

The PORT command has the argument of the IP of the client and the port that it is listening on to make an active connection. When the server gets this command it will try to connect to that IP and port, if it fails it will return an error, if it is successful then a data connection will have been established.

The PASV command has no arguments, this command will set up a passive data connection, a more in depth description of how this is done is available under the network module section.

The TYPE command has the arguments of A,E,I,L. In this implementation on the options A and I are supported. They are for either an ASCII type transfer of a binary type transfer. Since E and L are

not used anymore they are not supported in this implementation.

The RETR command has the argument of the file that a user wants to download. If the argument starts with a "/" then it is assumed that a full path of the location of the file is being given. If the argument does not start with a "/" it is assumed that the location is from the current working directory. Privileges are checked and if the user has access to download the file. This includes making sure that they have read access to it, they have enough credits and they are not over a bandwidth quota. If they pass all of those things then the file is sent.

The STOR command is very similar to that of RETR, the difference is that the STOR command is for sending a file to the server as opposed to receiving one, the same rules for the path applies and the same access checking is done except that the checking is done for write access instead of read access. Checking is not needed to see if you have enough credits because you don't use credits when you upload, you gain credits.

The REST command is used to set the value in bytes that the next file transfer will start at. If it is a download then the file will start downloading at the location given. If it is an upload then the file will be resumed at the location specified in REST. Take note that a user must have delete privileges on that file to be able to resume it in an upload operation; you only need read rights to resume a file that you are downloading. Once a file operation has finished the rest value will be set to zero. Another call to rest before a file operation will simply reset the old resume value to the new resume value specified.

The RNFR command is used to specify a file that is going to be renamed; checking is done to make sure the file specified actually exists and that the user that is trying to rename it actually has rights to do so. In order to be able to rename a file or directory the user must have delete privileges to it. If they do not have the rights then an access denied message will be sent. If it is a valid file that can be renamed then the name of the file will be stored in the incoming file buffer.

The next command that should be received should be the RNTO command. This command

should be received right after the RNFR command, and the argument of it should be the new name of the file that was specified in the RNFR command. The name in the incoming file buffer will be checked again to ensure that you still have the rights to rename it and as long as the name given for the new name of the file is valid, the file will be renamed. If this is a file or a directory that a user has uploaded or a virtual directory, then the name in the directory tree will also be changed. It is not guaranteed that the command will work correctly if another command is issued in between the RNFR and RNTO commands.

The ABOR command has no arguments and it will abort whatever file operation is taking place at the time. If there is an upload or a download in progress then it will be stopped, if not then issuing this command will do nothing since there will be nothing to abort.

The DELE command has the argument of a file to be deleted. If the path of the file starts with a "/" then it will be assumed that it is a complete path to the file starting from the root of the directory tree. If it does not start with a "/" then it is assumed that the location of the file starts in the current working directory. The rights of the user are checked to see if they have the rights to delete the file. If they have the rights to delete the file then it will be deleted, if they do not then an access denied message will be sent. In order to have the rights to delete a file you must have an access level of 3 on it. If the file that is being deleted is a file that has been uploaded by a user then it will be deleted from the directory tree and in addition to that, the user that uploaded it will lose the credits that they gained from uploading it. This will only happen if that user has a ratio and they can not go into negative credits, so if they currently have less credits then they would lose then their credits total will go down to zero. If the Windows user that the server is running under does not have rights to delete the file an error will be returned that the operation could not complete.

The RMD command removes a directory, the argument is going to be the path of the directory, if the path begins with a "/" then it is assumed that it is a complete path to the file starting from the root

of the directory tree. If the path does not start with a "/" then it is assumed that the path starts from the current working directory. A directory can only be deleted if there are no files or sub directories in the directory, it must be empty in order to be deleted. If there is something in the folder then an error will be sent to the client. If it is a user created directory that node will be removed from the directory tree but no credits will be altered as they are when a user created file is deleted.

The MKD command makes a directory, the argument is the path of the directory to be created, if the path starts with a "/" the is assumed that the path starts at the root of the directory tree, if it does not start with a "/" then it is assumed that the path starts at the current working directory. In order for a directory to be created the user must have write access in that folder. Once the folder is created a file node will be created for it in the directory tree.

The PWD command prints the current working directory for the FTP; it does not take any arguments. It will print the complete path of where you are in the directory structure of the FTP starting with the root of the tree. So the response will be something like this "/test/bsg" it will always start with a "/" so if you are at the root of the directory tree then you will get a response of "/".

The LIST command takes no arguments and will send the listing of the current working directory to the client. This command used a data connection, so previous to this command either a PORT command or a PASV command has to have been issued to establish a data connection. The LIST command will only send you the listing in a directory that you have access to read. So if there is a virtual directory in your current working directory that you do not have access to read then you will not receive it in the directory listing, it will be as if it is not even there.

The NLST command is very similar to that of the LIST command the difference is that the list command gives you the listing of your current working directory, while the NLST command has an argument of the path of the directory that you want to get a listing of. If the directory you specify exists and you have access to it then you will receive a listing of it over a data connection that has to

have been previously established.

The STAT command takes no arguments and will return some statistics on the user that issued the command, these statistics include the amount that user has downloaded, the amount that they have uploaded and the number of credits that they currently have.

The HELP command sends a list of comma nds that the server supports and offers additional information on some of them with the command of HELP with the name of the command that you want more information on as the argument.

The NOOP command is a command that is designed to do nothing and simply prevent the user from timing out. Some servers give you the option of disabling that command, but it is still possible to issue any other invalid command to keep from timing out on most of those servers. And the way that this server handles timeouts made it not practical to have a way to disable this command. So if someone issues a NOOP command then the server will reply ok and nothing else will happen. The SIZE command has the argument of a file name. If the location of the file starts with a "/" then it is assumed that the file location is being given form the root of the directory tree, if it does not start with a "/" then it is assumed to be from the current working directory. If the file exists the size of it in bytes will be returned.

4.6 Authorization Module

The authorization module handles all of the checking to see if the user has rights to do certain things on the server. This module is responsible for checking to see if the server is full, that means that the max number of allowed users are connected. Checking if a user can log on again or if they have used all of their allotted sessions. This module also checks a user's IP rules to see if they are a match with the IP that the user is currently connecting from. Checking to see if a user has enough credits to download a file is done here, as well as the checking to see if the user is using a ratio, meaning are they

using credits or not. The access permissions for virtual folder are checked here, as well as if they have access to any of the virtual folders at any given level of the tree. Checking for read, write and delete privileges are done here as well.

4.7 Logging

The logging module handles all of the logging on the server. This includes everything from the transfer of files to the individual commands that are sent by users and the responses that are sent. The amount of logging that is actually written out to disk can be configured in order to have your desired level of logging.

5. Server Comparisons

I did some comparisons between horseshoeFTPd and some of the Windows FTP servers that are available for sale. Testing against some of the free open source servers out there for Windows does deserve to be done and will be when time permits. The servers that the comparisons are going to be done with are Raidenftpd, Gene6 FTP server and serv-u FTP server. These three were chosen because they appeared to be the three most popular Windows FTP servers.

The first thing I did was do a feature breakdown to see if horseshoeFTPd supported most of the features that these proprietary servers were supporting. The most important features I would consider to be encrypted connections, this is supported by all of the servers. As well is bandwidth throttling, and bandwidth quotas. The way that bandwidth quotas where implemented was different among all of the servers. Only the gene6 server supported being able to set a bandwidth quota to reset on a daily basis, all of the over servers had the smallest increment at one week.

When it comes to the availability of the SITE command, which is a command that allows remote administration of the FTP, things like adding users, and adding IP rules and other administrative function. Raiden and gene6 supports this feature while serv-u does not. And none of these servers are open source, giving horseshoeFTPd the clear advantage in that category. When it comes to the price horseshoeFTPd also has the clear advantage with a price $0.00. The price of a license for Gene6 comes in $119.95. The licensing fee for serv-u is the most expensive $249.95. Finally raidenftpd comes in with a licensing fee of $87.95.

Options	HorseshoeFTPd	Gene6	Serv-U	RaidenFTPD
SSL	Yes	Yes	Yes	Yes
Bandwidth limiting	Yes	Yes	Yes	Yes
SITE commands	Yes	Yes	No	Yes
Open Source	Yes	No	No	No
Price	Free	$119.95	$249.95	$87.95

Figure 5-1 Feature Breakdown

After comparing the features I decided to do some benchmarking to see how these servers stood up against horseshoeFTPd in terms of speed of transferring files, amount of CPU used and amount of memory used. I ran 2 different kinds of tests. The first one was moving 3 files that totaled 1 gigabyte and the second was moving 155 files that totaled in at 1 gigabyte. I did the transfers both using encryption and without using encryption. I ran each of these tests on all of the servers 3 times each and took the average numbers from all three runs. This was to eliminate any variation that could have been a result of something random either slowing down or speeding up the transfer. Serv-U was not included in this benchmarking because of the licensing agreement for it I was unable to use it for the testing. I had already used the trial that allows encryption on all of the computers I had available for testing.

First let's look at the test of 3 files of a total of one gigabyte without SSL on. All of the servers seem to be moving the files as fast as possible when encryption is not on. The memory usage is about the same, horseshoeFTPd has a slight advantage over the other servers but it is nothing that significant. When it comes to the CPU time both HorseshoeFTPd and gene6 have a CPU time of about the same, 23 for horseshoeFTPd and gene6 at 22 seconds, not really a significant difference. Raidenftpd is a good 5 seconds better coming in at 17 seconds giving it the advantage when it comes to transferring large files with no encryption.

	HorseshoeFTPd	Gene6	RaidenFTPD
Memory Usage	9732K	12584K	13260K
Virtual Memory	9940K	12112K	12840K
CPU	23 Sec	22 Sec	17 Sec
Time to complete	101 Sec	101 Sec	101 Sec

Figure 5-2 Three files no SSL breakdown

Now with encryption on more of a difference between the servers is starting to appear. Raidenftpd seems to be in the loosing category here using the most memory, still not by that significant of a margin, but still more then the other two servers. The difference in CPU time for raiden is significant being 36 seconds more then horseshoeFTPd at a total time to complete of almost 40 seconds more then the other two servers. With the time to complete gene6 just edges out horseshoeFTPd with about a 5 second advantage, in terms of the CPU time gene6 has about a 16 second advantage. So in terms of sending large file with encryption gene6 is the winner but horseshoeFTPd is not that far behind.

	HorseshoeFTPd	Gene6	RaidenFTPD
Memory Usage	10244K	12648K	15180K
Virtual Memory	10008K	12180K	12920K
CPU	83 Sec	67 Sec	119 Sec
Time to complete	108 Sec	103 Sec	145 Sec

Figure 5-3 Three files with SSL breakdown

Memory is pretty much the same for all of these with no real significant difference. Raidenftpd won the CPU race beating gene6 by 4 seconds and beating horseshoe by 6 seconds. So it appears that

when it comes to unencrypted transferred that raidenftpd uses the least amount of CPU power. But interestingly, even though it used the least amount of CPU it still took the longest total time to complete the operation taking a good 40 seconds more then horseshoeFTPd did to complete the job. Gene6 beat horseshoeFTPd by 10 seconds in total time to complete, making gene6 the best server when it comes to transferring a large number of smaller unencrypted files, but horseshoe was not that far off the pace. Even though raiden has the least amount of CPU the time it took complete I think makes it a clear loser in this matter.

	HorseshoeFTPd	Gene6	RaidenFTPD
Memory Usage	10548K	13268K	15044K
Virtual Mem Usage	10756K	13436K	12880K
CPU	24 Sec	22 Sec	18 Sec
Time to complete	119 Sec	109 Sec	162 Sec

Figure 5-4 One hundred fifty five files no SSL breakdown

It looks like again raidenftp has the worse numbers, it used the most memory, though not by a lot but its CPU time is 30 seconds more then that of horseshoeFTPd and the time to complete is almost 40 seconds more. Gene6 and HorseshoeFTPd are once again very close in terms of the numbers. Though gene6 did edge out horseshoeFTPd in all categories it is not by much, the margin of victory in the CPU time used is only 9 seconds and the total time to complete 19 seconds.

	HorseshoeFTPd	Gene6	RaidenFTPD
Memory Usage	10840K	13392K	15864K
Virtual Mem Usage	10580K	13580K	13212K
CPU	75 Sec	66 Sec	105 Sec
Time to complete	147 sec	128 Sec	185 Sec

Figure 5-5 One hundred fifty five files with SSL breakdown

All in all from all of these numbers compiled it seems that gene6 is the most efficient of all the FTP servers. HorseshoeFTPd is not as efficient as gene6, but it is more efficient then radienftpd. Even though it is not as efficient as gene6 is it is still in the same neighborhood, I would not say that there is enough of a difference in the numbers to say that gene6 is significantly more efficient. So in that respect horseshoeFTPd is a success in that is as efficient or better then the better of the proprietary FTP servers.

6. Stumbling Blocks

There were several stumbling blocks that I encountered durring this project. The largest involved the fact that the RFCs for FTP were not exactly clear in the way that everything should be implemented. The biggest issue this caused was with trying to send a directory listing. The RFC says how a directory listing should be sent but it does not say what format that the directory listing should be in. The RFC implies that it could be any format, this would have been OK back when all the clients would be text based. In that case the directory listing would just be dumped out to the screen. The problem is having a format that all of the GUI clients can parse and display. It turns out that using the unix ls format works with all GUI clients I have tested so that is the format that horseshoeFTPd uses.

There was also to problem of having alot of the commands and features that are mentioned in the original RFC are no longer relevant. It took much effort to determine which options were still required and which could be omitted because they were not needed. For example the STOU command that is used to create a unique file name on the server is not really needed any more. The same could be said for SMNT command that is used to mount a different file system.

7. Future Work

In this project there are still some things that need to be implemented, the biggest and most important being a GUI to allow adminsitrators to more easily change the configurations on the server. Currently the only way to do that is by means of the SITE commands. And while you can change every necessary server setting though the SITE commands it is not always the most user-friendly way to do. Especially considering that the administrator would need to issue a separate command to change all of the settings on a new user that they created. Because of this a GUI is important especially in the Windows environment to make this application more user-friendly and as a result an application that more people will want to use.

For features on the server that are not yet implemented but probably should be, there is global bandwidth throttling. There is already individual user bandwidth throttling but there is no way to limit the total output of the server. As it is currently, you could set all of the users to have a max download speed of 50 K/second but if there are 20 users logged on and they are all using all 50K then there would be 1 meg/second of traffic for the whole server. There should be a way to limit the total bandwidth output of the server. This feature has been excluded up to this point because the way the system was designed there was no obvious way to enforce this limit. Something will need to be made that will report the current speed of each user and then limit it accordingly if they are going to fast.

The next feature that still needs to be implemented is some type of anti-hammering and temporary banning system. Hammering is when someone repeatedly tries to connect to the server, usually because it is full and they are trying to login before anyone else. This can waste resources having to constantly accept a connection and then disconnect them. There needs to be a system in place when it can be set if someone tries to connect x number of times in y number of seconds then their IP will be banned and ignored when they try to connect for z number of seconds. This isn't expected to be overly

difficult to be able to implement, it's just something that has not been done yet.

IP version 6 is not supported by this server. It is something that should be added on the off chance that people are still using this program when the internet switches over to version 6. This is something that should have been implemented from the start but was a design oversight.

Auth tls and Auth ssl should be implemented. This is different then the encryption that is currently available because with the current encryption it is in an always on or always off state. With the AUTH command the user will have the options of when to use the encryption and when not to. The existing code was written with future support of this in mind, so very little code should have to be rewritten in order to support this feature.

As is almost always the case with any software project of this size future work will include fixing the bugs that will no doubt popup as more and more user testing is done. Hopefully there will not be too many of them since the design of the code seems sound.

8. Conclusion

Overall the project was a success, there are still some things in the server that need to implemented, these things are covered in the future work section. The main goal of creating an open source ftp server for the Windows platform that is completely useable has been accomplished. The code is well documented so it should be relatively easy for someone who is not that familiar with the project to be able to figure out what is going on. The system seems to be stable; it is not known to crash randomly and has no memory leaks. There are probably unknown bugs still in the program that have not been found yet.

APPENDIX A - SITE Commands

SITE adddir location name

This command will add a virtual directory to the ftps file system. In order to run this command you must be in the siteop group or have an access level of 1000. The first argument is the location of the directory on your harddrive that you wish to add, for example c:\my folder. Spaces are allowed in the folder path and the location should not end with \. The second argument is the name that you want the virual folder to be called, this name can not have any spaces in it. The virtual directory will be added in whatever your current directory on the ftp is, the default rights it will be given is a read access to people with level 1000. You will probobly need to add further rights to it with the use if the adddirgroup command.

SITE adddirgroup level permissions

This command will add a permissions group to the current virtual directory that you are in. In order to run this command you must be in the siteop group or have an access level of 1000. The first argument is the group level you want to give access too. This is represented by an integer value between 1 and 1000, all all uses have a group value that is in that range and every user thats group value is atleast as high as you set it here will have the rights you give it in the second argument. The second argument is the access rights that you want to give the group, this will be either the number 1,2,3. If you want only read permissions you give the value of one, if you want read and write permissions you give the value of two. If you want read, write and delete permissions you give the value of three, please not that in order to append files you need to have delete permissions.

SITE deldir directory

The this command will delete the virtual directory that you give it the name of. In order to run this command you must be in the siteop group or atleast in a group with an access level of 1000. The only argument is the name of the virual directory that you want to delete. You must be in the directory that this virtual directory is in.

SITE editusergroup

//add this later

SITE editdir location name

This command will edit the current virtual diirectory that you are in. In order to run this command you must be in the siteop group or have an access level of 1000. The first argument is the location of the directory on your harddrive that you wish to add, for example c:\my folder. Spaces are allowed in the folder path and the location should not end with \. The second argument is the name that you want the virual folder to be called, this name can not have any spaces in it. The virtual directory will be modified with the values given, all of the groups in it will remain the same.

SITE adduser name password group

This command will add a new user to the ftp. In order to run this command you must be in the siteop group or have an access level of 1000. The first argument is the name of the new user, this can not be the same as any user already in the database and must be no longer then 29 alpha numberic characters. The second argument is the password for that user, this passord much be no longer then 19 alpha numberic characters in legnth. The last argument is the name of the group that this user will become part of. This group name must be the name of a group that has been created, or the siteop group. An ip rule will be added to the user that is *.*.*.*, this means that by default the user will allow connection from any ip address, if you do not want this you will need to change it with the addip and delip commands in edituser. All of the other values in this user will be set to the default values, you will probobly need to change them with the edituser command, or lock the user to a group to make them use the group settings, also done with the edit user command.

SITE addgroup name level

This command will add a new group to the ftp. In order to run this command you must be in the siteop group or have an access level of 1000. The first argument is the name of the new group, this can not be the same as any group already in the database and must be no longer then 29 alpha numberic characters. The second argument is the access level for this new group, this must be an integer value between the values of 1 and 1000. This number will be used to determine the rights users have on the ftp, 1000 is the highest access level. All of the other values in this group will be set to the default values, you will probobly need to change them with the editgroup command.

SITE delgroup name

This command will delete the group you specify from the database. In order to run this command you must be in the siteop group or have an access level of 1000. You will not beable to delete a group that users are a part of, so before you delete a group you must assign all users that are part of it to another group or delete them.

SITE deluser name

This command will delete the user you specify from the database. In order to run this command you must be in the siteop group or have an access level of 1000. You can not delete a user that is currently connected to the ftp, if the user you are trying to delete is currently connected to the ftp then you must first disconnect them, you may use the kill command for that.

SITE editgroup

The editgroup command is used to edit the settings in a usergroup. The first argument you pass it is the name of the group that you want to edit. The next argument is the type of change that you want to make.In order to run this command you must be in the siteop group or have an access level of 1000.

SITE editgroup name allowfxp boolean

This command allowes you to change the group setting of wether to allow fxp or not. In order to run this command you must be in the siteop group or have an access level of 1000. The argument this takes is a boolean value represented by a 0 for false and a 1 for true.

SITE editgroup name bwlimit amount

This command lets you set the bandwidth limit for users in this group per sessions, the value is an integer value represented in bytes. So what you would pass it is the number of bytes that you will allow to be transferred in every sessions which is reprssented in number of days in bwlimittype. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE editgroup name bwlimittype days

This command lets you set the number of days that each session for this group lasts. In order to run this command you must be in the siteop group or have an access level of 1000. The argument that you pass it is the number of days that each bandwith seession last. This is used for the bandwidth quotas, you set the amount that a person is able to download in the bwlimit and they will be able to download that much every this number of days. If you want the user to have no quota just set the number of days to zero.

SITE editgroup name iponfxp boolean

This command allowes you to change the group setting of wether to do ip checking in an fxp transfer. What this means is that the server that you are fxping to must be in the users list of alloed IP addresses, if it is not then it will not be allowed. In order to run this command you must be in the siteop group or have an access level of 1000. The argument this takes is a boolean value represented by a 0 for false and a 1 for true.

SITE editgroup name freefilesize size

This command lets you set the maximum size of a file that a user can download with out having it count agaisnt their credits. If you want to not have any free files then set this value to zero. This is not a relevant setting if you do not have ratios on. In order to run this command you must be in the siteop group or have an access level of 1000. The argument that this takes is the size in bytes.

SITE editgroup name permissions level

This command allowed you to change the access level for this group, the argument must be an integer value between the values of 1 and 1000. This number will be used to determine the rights users have on the ftp, 1000 is the highest access level. This setting will effect all users that are in this group reguardless if those users have the lock to group value set. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE editgroup name maxdownloadspeed speed

This command allowes you to set the maximum speed that a user will beable to download a file from the server. The argument that it takes is the number of maximum bytes per second that they will be allowed to transfer. If you do not want their to be a max download speed then simply set this value to zero. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE editgroup name maxuploadnuspeed speed

This command allowes you to set the maximum speed that a user will beable to upload a file from the server. The argument that it takes is the number of maximum bytes per second that they will be allowed to transfer. If you do not want their to be a max upload speed then simply set this value to zero. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE editgroup name maxlogins number

This command will allow you to set the max number of times a user can be logged in at any given time. The value it is passed is an integer value between 0 and max int. There is not way to turn off max number of logins so if you want an unlimited number just set the value of maxlogins greater then the number of max users allowed on the server, then there will not be an issue. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE editgroup name ratiod number

This command lets you set the denominator of ratio that this group will use. The ratio repsents how mant credits that they get for every byte that they upload. The value must be a positive number large then 0, if it is not that will turn of ratios. If you want to turn off ratios it is recommended that you use the ratiooff command and not just set the numnerator and the denominator to -1. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE editgroup name ration number

This command lets you set the numerator of ratio that this group will use. The ratio repsents how mant credits that they get for every byte that they upload. The value must be a positive number large then 0, if it is not that will turn of ratios. If you want to turn off ratios it is recommended that you use the ratiooff command and not just set the numnerator and the denominator to -1. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE edituser

The edituser command is used to edit the settings in a user. The first argument you pass it is the name of the user that you want to edit. The next argument is the type of change that you want to make.In order to run this command you must be in the siteop group or have an access level of 1000.

SITE edituser name active boolean

This command allowes you to change wether or not that this users account is active. If they are not active then they will not be allwed to log on. In order to run this command you must be in the siteop group or have an access level of 1000. The argument this takes is a boolean value represented by a 0 for false and a 1 for true.

SITE edituser name addip iprule

This command will allow you to add an ip rule to a user. The ip rule must be one that is not already part ot the users. The format for the ip rule is either a * to reprsent that anythign is allowd, a 3 digit numner like 077 to specify that only that number is allowd or a seven digit sequence to symbolize a range like 077-100. So a valid IP rule would be "077.120-150.*.200" an invalid IP rule would be "*" or "77.*.*.77-100." An ip rule representes an ip that is allowed to connect and access will be granted if any ip rule in the list matches the ip that someone is connecting from. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE edituser name delip iprule

This command will allow you to delete an ip rile that is in a user. Each user must have atleast one ip rule so you will not be able to delete it if it is that last ip rule for that user. If an IP rule exists for the user that matches the string given here it will be deleted, given that it is not the last ip rule. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE edituser name ratiooff

This command allowes you turn the ratio off for a user. If you run this command the user's ratio will be trurned off and their credits will be set to zero. It takes no arguments. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE edituser name credits numcredtis

This command lets you set the number of credits that a user currently has. This must be a positive integer value. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE edituser name groupname gname

This command allows you to set the group that a user is in. The argument is the name of the group that you desire to make this user a member of. This group must be one that is in the groups database, either the siteop group or a group that you have created. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE edituser name locktogroup boolean

This command allowes you to set wether or not this user is locked to his group. What this means is if the settings for the group that this user is a part of are going to be the settings for this user. If a user is locked to his group and you change a group setting like max download speed then this user will get the new maxdownload speed value. If he is not locked to the group then he will not get the new value and he will keep what he had. In order to run this command you must be in the siteop group or have an access level of 1000. The argument this takes is a boolean value represented by a 0 for false and a 1 for true.

SITE edituser name password newpassword

This command allows you to set a new password for a user. The password must be no longer then 19 alphanumneric characters. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE edituser name allowfxp boolean

This command allowes you to change the group setting of wether to allow fxp or not. In order to run this command you must be in the siteop group or have an access level of 1000. The argument this takes is a boolean value represented by a 0 for false and a 1 for true.

SITE edituser name bwlimit amount

This command lets you set the bandwidth limit for users in this group per sessions, the value is an integer value represented in bytes. So what you would pass it is the number of bytes that you will allow to be transferred in every sessions which is reprssented in number of days in bwlimittype. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE edituserp name bwlimittype days

This command lets you set the number of days that each session for this group lasts. In order to run this command you must be in the siteop group or have an access level of 1000. The argument that you pass it is the number of days that each bandwith seession last. This is used for the bandwidth quotas, you set the amount that a person is able to download in the bwlimit and they will be able to download that much every this number of days. If you want the user to have no quota just set the number of days to zero.

SITE edituser name iponfxp boolean

This command allowes you to change the group setting of wether to do ip checking in an fxp transfer. What this means is that the server that you are fxping to must be in the users list of alloed IP addresses, if it is not then it will not be allowed. In order to run this command you must be in the siteop group or have an access level of 1000. The argument this takes is a boolean value represented by a 0 for false and a 1 for true.

SITE edituser name freefilesize size

This command lets you set the maximum size of a file that a user can download with out having it count agaisnt their credits. If you want to not have any free files then set this value to zero. This is not a relevant setting if you do not have ratios on. In order to run this command you must be in the siteop group or have an access level of 1000. The argument that this takes is the size in bytes.

SITE edituser name permissions level

This command allowed you to change the access level for this group, the argument must be an integer value between the values of 1 and 1000. This number will be used to determine the rights users have on the ftp, 1000 is the highest access level. This setting will effect all users that are in this group reguardless if those users have the lock to group value set. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE edituser name maxdownloadspeed speed

This command allowes you to set the maximum speed that a user will beable to download a file from the server. The argument that it takes is the number of maximum bytes per second that they will be allowed to transfer. If you do not want their to be a max download speed then simply set this value to zero. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE edituser name maxuploadnuspeed speed

This command allowes you to set the maximum speed that a user will beable to upload a file from the server. The argument that it takes is the number of maximum bytes per second that they will be allowed to transfer. If you do not want their to be a max upload speed then simply set this value to zero. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE edituser name maxlogins number

This command will allow you to set the max number of times a user can be logged in at any given time. The value it is passed is an integer value between 0 and max int. There is not way to turn off max number of logins so if you want an unlimited number just set the value of maxlogins greater then the number of max users allowed on the server, then there will not be an issue. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE edituser name ratiod number

This command lets you set the denominator of ratio that this group will use. The ratio repsents how mant credits that they get for every byte that they upload. The value must be a positive number large then 0, if it is not that will turn of ratios. If you want to turn off ratios it is recommended that you use the ratiooff command and not just set the numnerator and the denominator to -1. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE edituser name ration number

This command lets you set the numerator of ratio that this group will use. The ratio repsents how mant credits that they get for every byte that they upload. The value must be a positive number large then 0, if it is not that will turn of ratios. If you want to turn off ratios it is recommended that you use the ratiooff command and not just set the numnerator and the denominator to -1. In order to run this command you must be in the siteop group or have an access level of 1000.

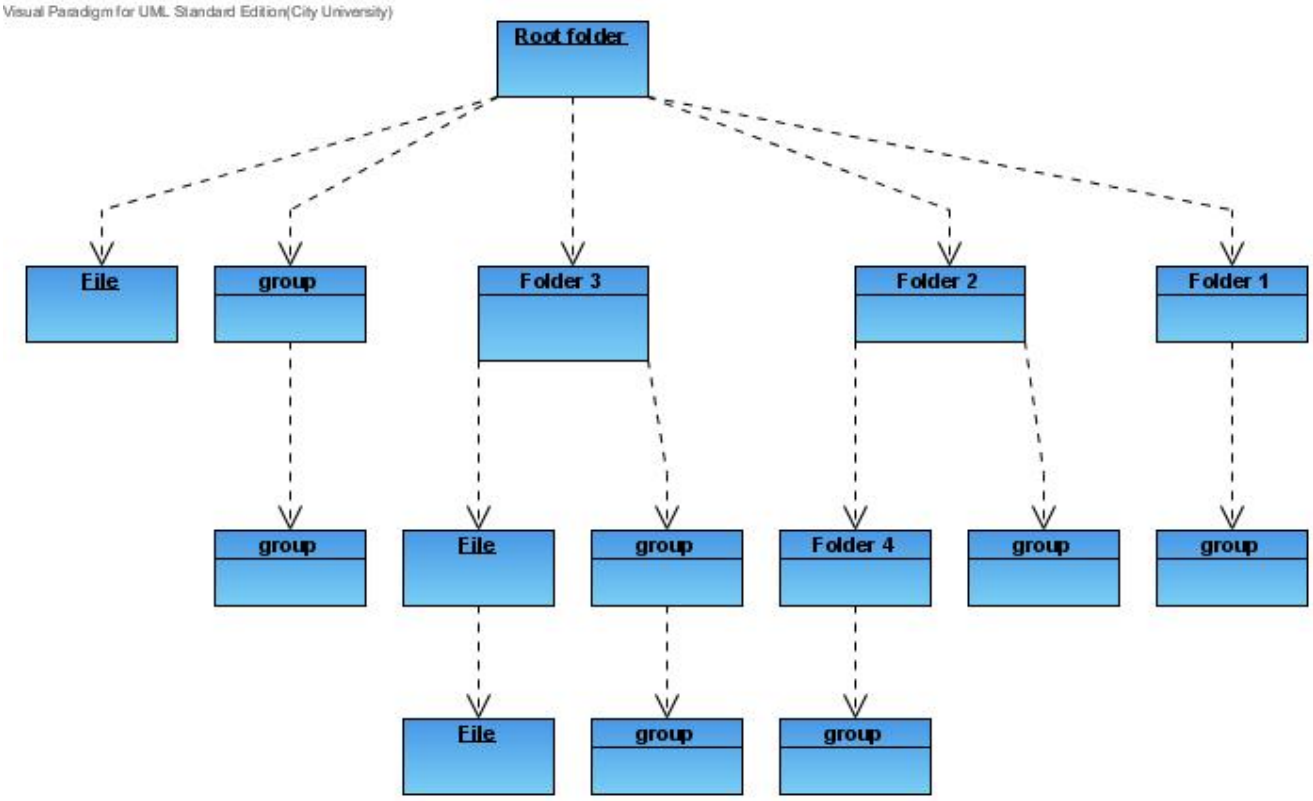
SITE kill name

This command allows you to foricibly disconnect a user from the ftp. The argument is the name of the user, if that user is currently connected to the ftp they will be disconnected. This includes all sessions of them logged in, so if they are logged in twice both connections will be closed. In order to run this command you must be in the siteop group or have an access level of 1000.

SITE shutdown

This command will disconnect all current users from the ftp, save all the settings and close the application. Be warned that once you run this command you will need to manualy restart the server before you will be able to do anything again.

APPENDIX B – Directory tree Diagram



APPENDIX C - XML Configuration Files

dir.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<Folder Location="J:\ftp.home\" Name="">
  <Group Name="1000" Permissions="1"/>
  <Group Name="999" Permissions="1"/>
  <Sub Location="H:\TV2\" Name="TV2">
    <Group Name="900" Permissions="1"/>
    <Group Name="500" Permissions="1"/>
  </Sub>
  <Sub Location="F:\TV\" Name="TV">
    <Group Name="500" Permissions="1"/>
  </Sub>
  <Sub Location="J:\Mp3z\" Name="Mp3z">
    <Group Name="899" Permissions="1"/>
  </Sub>
  <Sub Location="G:\tmp\" Name="~UPLOAD~">
    <Group Name="1" Permissions="3"/>
      <File Dir="false" Group="1000"
Name="Charmed.S02E22.Be.Careful.What.You.Witch.For.DVDRip.XViD-iTV.avi"
Owner="jade_dixon" Path=""/>
    <Sub Location="j:\MQP" Name="MQP">
      <Group Name="1000" Permissions="1"/>
    </Sub>
  </Sub>
  <Sub Location="H:\music.videos\" Name="H">
    <Group Name="1000" Permissions="1"/>
  </Sub>
  <Sub Location="j:\tv eps\" Name="testtv">
    <Group Name="1000" Permissions="1"/>
  </Sub>
</Folder>
```

group.xml

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<!--Sample XML file generated by XMLSpy v2006 sp2 U (http://www.altova.com)--><Groupfile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="E:\MQP\users.xsd">

<Group Allow_Fxp="true" Bandwidth_limit="0" Bandwidth_limit_type="0" Check_IP_for_FXP="false" Free_file_size="0" Group_Permissions="1000" Max_Download_Speed="0" Max_Logins="1" Max_Upload_Speed="0" Name="siteop" Ratio_D="-1" Ratio_N="-1"/>

</Groupfile>

user.xml

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<!--Sample XML file generated by XMLSpy v2006 sp2 U (http://www.altova.com)-->

<Userfile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="E:\MQP\users.xsd">

<User Active="true" Allow_Fxp="true" Bandwidth_limit="0" Bandwidth_limit_type="0"
Bandwidth_this_session="824192301" Bytes_downloaded="0" Bytes_uploaded="0"
Check_IP_for_FXP="false" Credits="-1" Experation_date="" Free_file_size="0"
Group_name="siteop" Group_permissions="1000" Lock_to_Group="false"
Max_Download_Speed="0" Max_Logins="1" Max_Upload_Speed="0" Name="testuser"
Password="a58157f57cc17ef55c3284ed157d20c5620be0fd" Ratio_D="-1" Ratio_N="-1"
Reset_At="1730179685">

<IP_rules Address="*.*.*.*" Allow="true"/>

</User>

<User Active="true" Allow_Fxp="true" Bandwidth_limit="2147483647" Bandwidth_limit_type="0"
Bandwidth_this_session="0" Bytes_downloaded="-111320636" Bytes_uploaded="510898580"
Check_IP_for_FXP="true" Credits="-1" Experation_date="" Free_file_size="0" Group_name="siteop"
Group_permissions="1000" Lock_to_Group="false" Max_Download_Speed="0" Max_Logins="100"
Max_Upload_Speed="0" Name="jade_dixon"
Password="daa179250729d02505666d42eb95415654e34605" Ratio_D="-1" Ratio_N="-1"
Reset_At="1145324759">

<IP_rules Address="*.*.*.*" Allow="true"/></User>

<User Active="true" Allow_Fxp="true" Bandwidth_limit="0" Bandwidth_limit_type="0"
Bandwidth_this_session="0" Bytes_downloaded="1700956749" Bytes_uploaded="49613756"
Check_IP_for_FXP="false" Credits="-1" Experation_date="" Free_file_size="0"
Group_name="siteop" Group_permissions="1000" Lock_to_Group="false"
Max_Download_Speed="0" Max_Logins="1" Max_Upload_Speed="0" Name="mjathree"
Password="89ab7f6c25a15970c79d2f0bac36b2d69568e583" Ratio_D="-1" Ratio_N="-1"
Reset_At="1145295638">

<IP_rules Address="*.*.*.*" Allow="true"/>

</User></Userfile>

options.xml

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<!--Sample XML file generated by XMLSpy v2006 sp2 U (http://www.altova.com)-->

<ServerOptions Bandwidth_limit="2147483647" Bandwidth_limit_type="0" Bandwidth_this_session="0" IP="130.215.238.156" Name="testFTPd" Reset_At="1145328062" UseSSL="false" dataportrangeH="3000" dataportrangeL="2000" freefilesize="0" idletimeout="60" maxconnectionsperip="5" maxusers="5" packetsize="4096" port="21" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="E:\MQP\serversettings.xsd"/>

log.xml

<Connection time="Wed Apr 12 02:54:03 2006" IP="130.215.168.228" />

<Login time="Wed Apr 12 02:54:04 2006" user="mjathree" status="Logged in" IP="130.215.168.228" />

<Transfer time="Wed Apr 12 02:54:07 2006" user="mjathree" action="Upload" file="G:\tmp\WarHammer.40000.Dawn.Of.War.Winter.Assault[CD1][PC][PDX.TEAM][www.Patron divx.net].BWI" status="Started" IP="130.215.168.228" />

<Transfer time="Wed Apr 12 02:54:07 2006" user="mjathree" action="Upload" file="G:\tmp\WarHammer.40000.Dawn.Of.War.Winter.Assault[CD1][PC][PDX.TEAM][www.Patron divx.net].BWI" status="Resuming" IP="130.215.168.228" />

<Transfer time="Wed Apr 12 02:55:17 2006" user="mjathree" action="Upload" file="G:\tmp\WarHammer.40000.Dawn.Of.War.Winter.Assault[CD1][PC][PDX.TEAM][www.Patron divx.net].BWI" status="Timeout" IP="130.215.168.228" />

<Logout time="Wed Apr 12 02:55:55 2006" user="mjathree" status="Complete" IP="130.215.168.228" />

<Connection time="Wed Apr 12 02:58:22 2006" IP="130.215.168.228" />

<Login time="Wed Apr 12 02:58:22 2006" user="mjathree" status="Logged in" IP="130.215.168.228" />

<Logout time="Wed Apr 12 02:59:29 2006" user="mjathree" status="Complete" IP="130.215.168.228" />

<Connection time="Wed Apr 12 03:16:15 2006" IP="130.215.168.228" />

<Login time="Wed Apr 12 03:16:15 2006" user="mjathree" status="Logged in" IP="130.215.168.228" />

<Connection time="Wed Apr 12 03:17:10 2006" IP="130.215.168.228" />

<Login time="Wed Apr 12 03:17:11 2006" user="mjathree" status="Logged in" IP="130.215.168.228" />

<Connection time="Wed Apr 12 03:19:27 2006" IP="130.215.168.228" />

<Login time="Wed Apr 12 03:19:27 2006" user="mjathree" status="Logged in" IP="130.215.168.228" />

<Connection time="Wed Apr 12 03:21:01 2006" IP="130.215.168.228" />

<Login time="Wed Apr 12 03:21:01 2006" user="mjathree" status="Logged in" IP="130.215.168.228" />

<Connection time="Wed Apr 12 03:23:52 2006" IP="130.215.168.228" />

<Login time="Wed Apr 12 03:23:53 2006" user="mjathree" status="Logged in" IP="130.215.168.228" />

<Connection time="Wed Apr 12 03:27:04 2006" IP="130.215.168.228" />

<Login time="Wed Apr 12 03:27:04 2006" user="mjathree" status="Logged in" IP="130.215.168.228" />

<Transfer time="Wed Apr 12 03:27:07 2006" user="mjathree" action="Upload" file="G:\tmp\WarHammer.40000.Dawn.Of.War.Winter.Assault[CD1][PC][PDX.TEAM][www.Patron divx.net].BWI" status="Started" IP="130.215.168.228" />

<Transfer time="Wed Apr 12 03:27:07 2006" user="mjathree" action="Upload" file="G:\tmp\WarHammer.40000.Dawn.Of.War.Winter.Assault[CD1][PC][PDX.TEAM][www.Patron divx.net].BWI" status="Resuming" IP="130.215.168.228" />

<Connection time="Wed Apr 12 03:27:21 2006" IP="130.215.238.156" />

<Login time="Wed Apr 12 03:27:21 2006" user="jade_dixon" status="Logged in" IP="130.215.238.156" />

<Logout time="Wed Apr 12 03:28:22 2006" user="jade_dixon" status="Complete" IP="130.215.238.156" />

<Connection time="Wed Apr 12 03:29:57 2006" IP="130.215.168.228" />

<Login time="Wed Apr 12 03:29:57 2006" user="mjathree" status="Logged in" IP="130.215.168.228" />

<Transfer time="Wed Apr 12 03:29:59 2006" user="mjathree" action="Upload" file="G:\tmp\WarHammer.40000.Dawn.Of.War.Winter.Assault[CD1][PC][PDX.TEAM][www.Patron divx.net].BWI" status="Started" IP="130.215.168.228" />

<Transfer time="Wed Apr 12 03:29:59 2006" user="mjathree" action="Upload" file="G:\tmp\WarHammer.40000.Dawn.Of.War.Winter.Assault[CD1][PC][PDX.TEAM][www.Patron divx.net].BWI" status="Resuming" IP="130.215.168.228" />

<Transfer time="Wed Apr 12 03:29:59 2006" user="mjathree" action="Upload" file="G:\tmp\WarHammer.40000.Dawn.Of.War.Winter.Assault[CD1][PC][PDX.TEAM][www.Patron divx.net].BWI" status="Over user bandwidth quota" IP="130.215.168.228" />

<Transfer time="Wed Apr 12 03:30:27 2006" user="mjathree" action="Upload" file="G:\tmp\WarHammer.40000.Dawn.Of.War.Winter.Assault[CD1][PC][PDX.TEAM][www.Patron divx.net].BWI" status="Started" IP="130.215.168.228" />

<Transfer time="Wed Apr 12 03:30:27 2006" user="mjathree" action="Upload" file="G:\tmp\WarHammer.40000.Dawn.Of.War.Winter.Assault[CD1][PC][PDX.TEAM][www.Patron divx.net].BWI" status="Resuming" IP="130.215.168.228" />

<Transfer time="Wed Apr 12 03:30:27 2006" user="mjathree" action="Upload" file="G:\tmp\WarHammer.40000.Dawn.Of.War.Winter.Assault[CD1][PC][PDX.TEAM][www.Patron divx.net].BWI" status="Over user bandwidth quota" IP="130.215.168.228" />

<Logout time="Wed Apr 12 03:31:28 2006" user="mjathree" status="Complete" IP="130.215.168.228" />

<Connection time="Wed Apr 12 03:31:30 2006" IP="130.215.168.228" />

<Login time="Wed Apr 12 03:31:30 2006" user="mjathree" status="Logged in" IP="130.215.168.228" />

<Transfer time="Wed Apr 12 03:31:42 2006" user="mjathree" action="Upload" file="G:\tmp\WarHammer.40000.Dawn.Of.War.Winter.Assault[CD1][PC][PDX.TEAM][www.Patron divx.net].BWI" status="Started" IP="130.215.168.228" />

<Transfer time="Wed Apr 12 03:31:42 2006" user="mjathree" action="Upload" file="G:\tmp\WarHammer.40000.Dawn.Of.War.Winter.Assault[CD1][PC][PDX.TEAM][www.Patron divx.net].BWI" status="Over user bandwidth quota" IP="130.215.168.228" />

<Logout time="Wed Apr 12 03:32:44 2006" user="mjathree" status="Complete" IP="130.215.168.228" />

APPENDIX D - Acronyms

FTP – File Transfer Protocol

SSL – Secure Socket Layer

FXP – Server to Server Transfer Protocol

CPU – Central Processing Unit

RFC – Request for Comments

APPENDIX F - Files on CD

SOURCE CODE:
(in sourcefiles.zip)

auth.cpp
auth.h
consts.h
crypt.cpp
crypt.h
filefolders.cpp
filefolders.h
ftpd.cpp
ftpd.h
ftp_options.cpp
ftp_options.h
helpfunctions.cpp
helpfunctions.h
logging.cpp
logging.h
ls.cpp
ls.h
mem_manage.cpp
mem_manage.h
parser.cpp
parser.h
site.cpp
site.h
stdafx.h
tcp_manager.cpp
tcp_manager.h
user.cpp
user.h
ws-util.cpp
ws-util.h
xml_manager.cpp
xml_manager.h
horseshoeftpd.vcproj

CONFIG:

user.xml
group.xml
options.xml
dir.xml

DLLS:

xerces-c_2_7.dll

EXECUTABLES:

horseshoeFTPd.exe

OTHER:

README
COMPILEREADME
SITECMDS