Trivial File Transfer Protocol Report

TFTP-UDP-Server

The Server for this assignment is set to a specific port, 9000, when first run, and runs in the command line. The port 9000 was a random port selected to be the servers. The server waits for a packet to be received from a client, and given the opcode: 01 for writing to file; 02 for reading from file; 03 for being sent data from client; 04 for being sent an acknowledgement from the client, it then executes the correct code whether this be creating a file, finding a file and sending data back, sending back read data, or writing to a file. If the client wants to read from a file that doesn't exist the server then sends back an error packet with opcode 05, as mentioned in the brief, other than this there is no error detection. The packets are created and sent back and are all in bytes and therefore satisfy being in octet mode.

In order for the server to handle multiple clients the clients each have a unique port number, but the server also stores them under an IP address as well, because if they were connecting from a different machine two clients could be using the same port numbers. In order to keep track of block numbers and filenames these are stored in separate HashMaps, with the Key being a class called AddressAndPort and is stored in the package, 'server', this has the fields Address and port, and the data being state or block number for one HashMap, and the filename for the other.

The server therefore handles multiple clients as stated in the brief, as it receives the packet and works out the port number and address from the packet, which allows it to get back to that clients current stage of transmission.

The server saves all files created in the Files folder in its directory, this is because it was found that if the files were created in the main directory, and the file was trying to be called 'test' for example this file wouldn't be created due to the folder 'test' existing in that directory.

TFTP-UDP-Client

When the client is first run in the console, a random unused port on the clients system is used as the port number. The client is asked if they'd like to store or receive a file, and can do so with the input into the console, after which they input the filename they are looking to use. If they are writing to file, the contents of the file is transferred to the server. If they are reading the user is given the contents of the file they were looking for if it exists. After which they are asked if they'd like to repeat the process, as the program runs in a while loop in the run() subroutine. This is to allow the user to repeat the process if the program throws an error, for example; if the user is trying to read from a file that doesn't exist, the only error that is required by the specification.

The first packet sent to the server by the client has the opcode 01/02 dependant on whether the client is reading/writing to a file. Once the server sends back: an ACK opcode 04 for writing, or data opcode 03 for reading, the client then sends the writing data over opcode 03, or an ACK packet back to the server, until the transmission is over when the length of the data packets are less than 516 in bytes.

When the program is ran again, a new random port is selected, and this only so that the server knows it is a new connection. The server would cope if it was the same port, but it allows the server to show the history of transactions that happened.

When writing to file which uses the write() subroutine, using the inputted filename it gets all the bytes in the file, which is then converted to a byte array to send. This is done for simplicity and also it is easy to print out the inputted data if there was a problem when developing. When the data is sent it is repeated for every 512 bytes in the input string, as this is the maximum to be transmitted in the brief.

When reading from a file which uses the read() subroutine, a block number is used as mentioned in the brief, this is attached to every ACK packet sent back to the server and the block number is then increased. The client determines whether or not the data packet sent from the server is also less than 516 bytes in length in order to know when to stop looking to receive and end the transmission.

The client saves all files created in the Files folder in its directory, this is because it was found that if the files were created in the main directory, and the file was trying to be called 'test' for example this file wouldn't be created due to the folder 'test' existing in that directory.