TFTP Report

((#) Refers to the line in the class being referenced.)

Both client packages accept two arguments:
- Args[0] is the request to either read or write, with “1” representing read and ”2” representing write.
- Args[1] is the pathname of the file to read from on the server side, or write to from the client side. An example, textfile.txt, with pathname “src/textfile.txt” has been included in the project(s) to show an example of this working.

TFTP-UDP-Client:

Args[0] is checked to see what request the user requires (90). If it is a read, a file output stream is created (94), used to save the data coming in. If it is a write, a file input stream is created (102), using the given file from Args[1]. If this file cannot be found, an error packet (108) is sent to the server, and the client closes its socket. (The client waits to close its socket though to make sure the error packet has been received (233)). If the file is found, an RRQ/WRQ packet is sent to the server (122). The client then enters a blocking call (140). If a timeout occurs due to not receiving a packet for 10,000 time units (87), the last packet sent by the client will be retransmitted (150), and another receive() blocking call will be entered (151/152).
If the packet received was the last received packet (142), it means the client needs to retransmit its last sent packet (143) and enter another receive() blocking call. Otherwise, once a packet is received, its opcode is checked (160).
If an ACK is received (because the user requested a write), its block number is checked to see if it is the expected block number (169). If it isn’t, do nothing and enter another receive() blocking call (140). If it is, then read up to 512 bytes from the input stream (170). If <512 bytes are read (i.e. all bytes have been read from the input stream after this read), the data packet to be sent to the server will be the last one (177). If no bytes are read, an empty data packet (173/180) will be sent to the server. Otherwise, a data packet storing data of 512 bytes will be sent to the server (182).
If a DATA packet is received (because the user requested a read), its block number is checked to see if it is the expected block number (196). If it isn’t, do nothing and enter another receive() blocking call (140). If it is, then firstly check the data packet contains some data (198). If it doesn’t contain data, close the socket immediately because all data has already been received and read (199/200). If it does contain data, write the data to the file (204/205), then send an ACK back that has the same block number as the received DATA packet (212-215). If there were <512 bytes of data in the received packet (208), close the socket after sending the final ACK (209) as all data has now been read.
If an ERROR packet is received (because the file could not be found from the server in a read request), close the output stream (223) and delete the file on the client that the file on the server was going to be read into (225), then close the socket.

TFTP-UDP-Server:

Begins by waiting for a packet from the client side (97). Once one is received, its opcode is checked (123).
If a RRQ packet is received, socket timeouts are enabled (127) and a file input stream is created (136), using the given file from Args[1] on the client side. If this file cannot be found, an error packet (140) is sent to the client, which leads to the client closing its socket. If the file is found, the first data packet is sent to the client (167-169). If the file is empty, the server will not expect an ACK back from the client (155), and the timeout can be disabled as the server no longer expects client packets (157). If the file contains <512 bytes, the server only expects one more ACK from the client (160). In all cases, the server will go back to a blocking call (97).
If a WRQ packet is received, socket timeouts are enabled (175) and a file output stream is created (182), used to save the data coming in (the filename is received from the WRQ packet (177-180)). An ACK with block number 0 must be sent to the client to acknowledge the write request was received (183-188).
If a DATA packet is received (because the user requested a write), its block number is checked to see if it is the expected block number (199). If it isn't, do nothing and enter another receive() blocking call (97). If it is, then firstly check the data packet contains some data (205). If it doesn't contain data, enter another receive() blocking call because all data has already been received and read (206). If it does contain data, write the data to the file (209/210), then send an ACK back that has the same block number as the received DATA packet (212-215). If there were <512 bytes of data in the received packet (201), disable timeouts on the socket (202) as no more packets are expected from the client.
If an ACK is received (because the user requested a read), it is firstly checked to see if it is the last ACK the server will receive (227). If it is, timeouts can be disabled (228) as the server expects no more packets from this client. If it isn't the last ACK, its block number is checked to see if it is the expected block number (230). If it isn't, do nothing and enter another receive() blocking call (97). If it is, then read up to 512 bytes from the input stream (231). If <512 bytes are read (i.e. all bytes have been read from the input stream after this read), the data packet to be sent to the server will be the last one, therefore only one more ACK is expected from the client (239). If no bytes are read, a final empty data packet (235/242) will be sent to the server, and timeouts are disabled as the server does not expect an ACK for the empty data packet. Otherwise, a data packet storing data of 512 bytes will be sent to the server (244).
If an ERROR packet is received (because the file could not be found from the client in a write request), disable timeouts (254) and enter another receive() blocking call.

TFTP-TCP-Client:

Firstly, a byte is received from the server (48) to make sure the server thread object for the client object has had time to be created. Then Args[0] is checked to see what request the user requires (54).
If it is a read, a file output stream is created (62), used to save the data coming in, and the input stream of the client/slave socket (59) is used to receive messages from the server. The RRQ packet is sent to the server (64/65), and the client waits for a DATA or ERROR packet to be accepted into its input stream (72).
If a DATA packet is received, check the data packet contains some data (78). If it doesn't contain data, close the socket immediately because all data has already been received and read (79/80). If it does contain data, write the data to the file (84/85). If there were <512 bytes of data in the received packet (88), close the socket after sending the final ACK (89) as all data has now been read.
If an ERROR packet is received (because the file could not be found from the server in a read request), close the output stream (96) and delete the file on the client that the file on the server was going to be read into (98), then close the socket.
If the request in Args[0] is a write, a file input stream is created (108), using the given file from Args[1]. If this file cannot be found, an error packet (114) is sent to the server, and the client closes its socket. (The client waits to close its socket though to make sure the error packet has been received (150)). If the file is found, an WRQ packet is sent to the server (120). Then client then goes through a loop of reading bytes from the input stream (129) and sending that data to the server (139-142), until the final sequence of bytes has been read (131/135). The client socket is then closed.

TFTP-TCP-Server:

A byte is sent to the client to tell it the server thread has been created and is ready to start accepting data from the client (46). The server then waits for a packet from the client side (49). Once one is received, its opcode is checked (51).
If it is an RRQ packet, a file input stream is created (54), using the filename from the RRQ packet (60-63). If this file cannot be found, an error packet (70) is sent to the client, and the client socket connection is closed (141). If the file is found, the server goes through a loop of reading bytes from the input stream (77) and sending that data to the client (87-90), until the final sequence of bytes has been read (79/83). The client socket connection is then closed (141).
If it is a WRQ packet, a file output stream is created (103), used to save the data coming in (the filename is received from the WRQ packet (98-101)). The server then waits for a DATA or ERROR packet to be accepted into its input stream (108).
If a DATA packet is received, check the data packet contains some data (112). If it doesn't contain data, close the socket connection immediately because all data has already been received and read (113/114). If it does contain data, write the data to the file (118/119). If there were <512 bytes of data in the received packet (122), close the socket connection (141) as all data has now been read.
If an ERROR packet is received (because the file could not be found from the client in a write request), close the socket (141).