Seyedeh Arta Razavi
CS 453
Project 1

Design

First you fire up both the client and the server in python.

Example:

Python UDPServer.py

Python UDPCLient.py

Python TCPServer.py

Python TCPCLient.py

Server side UDP

The server receives me message as a string in the format of

[number][operation code][number][space][decimal number]

The server splits the string into an array

If the length of that array is more than 2 then we know the user has put in an invalid input that is not in the format above so we return status code 300

The server gets the decimal probability of dropping a packet in the form of a string and turns it into a float. Then you multiply that float by 100 to get the percent. After obtaining percent you make an array that has:

#0s in the array = probability of dropping packets

#1s in the array = 100-probability of dropping packets

Then we randomly pick a number from that array to simulate dropping with that probability

If we pick 0 packet is dropped server does nothing

If we pick 1 we go on to calculate

We split the [number][operation code][number] section of the message and check to see if all parts of it are valid. We check if both numbers are digits and out operation code is +-/* and we are not dividing by 0

If all these checks pass we perform calculation and second back the result to the client along with status code 200

[result][space][200]

If one or more of those checks do not pass we send back -1 as the result and status code 300

[-1][space][300]

Go back to beginning and listen for more input

Seyedeh Arta Razavi
CS 453
Project 1

Client side UDP

User is prompted to input message in form of (note there are no spaces):

[number][operation code][number]

User is prompted again after that to input a decimal number representing the probability that the server will drop the packet. This number must be in decimal format for example if the user wanted packets dropped with the probability of 80% they would input 0.8

The client then appends these together with a space in between and sends them to the server

[number][operation code][number][space][decimal number]

On the client side a timeout is set to 0.1 seconds and the client will keep attempting to send packets and waits for a response from the server. If no response comes client doubles the timeout and tries again and it keeps doing this until it reaches time out of 2 seconds.

If no message is received and our time out has reached 2.0 seconds the client will tell user server is dead and abort

If a server response is recieved it comes in the format

[operation result][space][status code]

The client then unpacks this message by splitting the string

If the status code is 200 the client prints to the user the results of the calculation status code 200

If the status code is 300 the client tells the user something has gone wrong and prints status code 300

The client asks the user if they want to repeat and if yes it repeats

Server side TCP

The server receives me message as a string in the format of

[number][operation code][number]

The server splits the [number][operation code][number] section of the message

If the length of that is more than 3 it is in the wrong format send back status code 300 result -1

[-1][space][300]

Then the server checks to see if all parts of it are valid. We check if both numbers are digits and out operation code is +-/* and we are not dividing by 0

If all these checks pass we perform calculation and second back the result to the client along with status code 200

[result][space][200]

If one or more of those checks do not pass we send back -1 as the result and status code 300

[-1][space][300]

Go back to beginning and listen for more input

Client side TCP

User is prompted to input message in form of (note there are no spaces):

[number][operation code][number]

The client sends message to the server

[number][operation code][number]

Server response comes in the format

[operation result][space][status code]

The client then unpacks this message by splitting the string

If the status code is 200 the client prints to the user the results of the calculation status code 200

If the status code is 300 the client tells the user something has gone wrong and prints status code 300

The client asks the user if they want to repeat and if yes it repeats

Improvement

Maybe learn how to send many messages back to back instead of writing a string separated by spaces and then splitting it into an array on server side to get multiple messages.

Maybe implement a better way of dropping messages than creating an array of 1s and 0s

Maybe handle all of formatting checks with regex instead of simple parsing

Seyedeh Arta Razavi
CS 453
Project 1

Extensions
Make program calculate with floats as well as ints

Make program handle harder calculations like mod