2019-3-6

Lab 08 - Due at end of class

As usual, make a new branch in your assignments repo called ‘lab08’ and do your work in there. commit your changes to github and make a pull request assigned to me.

Be careful not to submit any files other than the ones meant for me to see. No .DS\_Store, or similar non assignment related files.

For the first problems, you will working with two files, network\_queue.py and network\_pri\_queue.py

Both use 3 threads to read a file and sent it to network router using one queue. The router then processes packets on to the network.

Q1 - 33pts

a) To get the network\_queue.py file safe to run it needs to handle the IOError. Implement the try/except for file opening and reading in each thread. In it’s current state if the file doesn’t exist or there is an error in opening or reading, the thread will exit and never set the ‘done’ flag. The key to make sure the thread sets its ‘done’ flag to true before exiting so the main thread can know to quit.

Q1b) Create files with 50 lines of text each with filenames to match what each thread is attempting to open. I recommend having the contents be different in each file such that you can easily observe which file specific data is from. For instance, I user 50 lines of the emma.txt, the words.txt and then 50 lines of random characters in my lab.

Also have each thread prepend some text to the line before it send it to the queue. Using a format operator to write some text and a packet index before the line that is sent.

For instance, if the thread KidsInternet is meant to send the line “12345” for its 25th packet, it should change the line to “kid25:12345”

Q1c) Run the script and observe the order packets are printed or “processed” by the router and the “fullness” of the queue plotted in the graph.

Comment on what is happening in each portion of the graph. Can you tell when a specific thread has ended?

What do the slopes of the plot indicate?

Q1d) The processing\_rate is the speed the router can process packets. Change this value to observe and the effects it has. What happens when it’s less that the lowest datarate? What about when it’s 100x the highest datarate?

Q2) 33pts

network\_pri\_queue.py is a very similar script but uses a priority queue with 3 priority levels.

1. Make the same file IO exception handing changes and prepending text to sent packets as you did in network\_queue.py. Run the file and comment on it’s order of processing packets and plot of queue fullness.
2. Can you say which plot line is which priority? Why or why not?
3. If you change the processing\_rate what kind of effect does it have on the order of processed packets?

Q3) The Monty Hall Challenge - 33pts

This is a pretty famous problem that can even get experienced statisticians arguing with each other. It comes from the famous game show of the 70’s called Let’s make a deal.

On the show there is a “big deal of the day” prize. The big deal of the day prize works like this. There is usually one big prize, and two booby prizes, such as a goat. You can choose to get the whatever is behind door number 1, door number 2, or door number 3. Once you have picked your door then Monty will reveal what is behind one of the other doors. He won’t reveal the grand prize but he might show you the goat. Then Monty will give you the opportunity to switch your door. The question is, should you switch? Do you increase your odds of getting the grand prize by switching doors? Does it stay the same? Or, do your odds of winning go down if you switch? If you have already heard this keep it to yourself and let everyone make their best guess before find out the truth.

The math behind this is pretty complicated, but now that you are an experienced Monte Carlo simulator you can solve this one. With some Python code. Using the random number generator you can put a prize behind a door. Then using the random number generator you can let the ‘contestant’ pick a door. You will then remove a door using the rule that you cannot reveal the grand prize. Then you can simulate what happens if the contestant switches doors or not. You can do this 10,000 times to determine the odds of winning if you switch and the odds of winning of you stick.

The general idea is to write a function that takes a switch as an input, uses randint to select the door the prize is in, and again to select the guess. It returns true or false based on if the guess was correct or not.

def monty\_guess(switch):

prize = random.randint(1,3)

guess = random.randint(1,3)

If switch:

Change the guess

Return guess == prize

Then run 10000 simulation of each scenario, one simulation where the guess is changed and another where it is not. The result Save the results in a list (I would recommend a numpy array) and calculate the mean of how often each method is correct.

My guess is that once you go through the effort to code it, you will have a very good idea of what the results will be, even if it contradicts what you initially thought.