NWEN303

Project one

Liam De Grey

300294561

Part A)

For this part I have followed my own procedure with how the program runs.   
A set number of clients and providers are first made, these are then combined into 1 big list being added one by one (Client, Provider, Client, Provider etc.). Each Client and provider is its own thread. From here the threads are started one by one, straight after each other. Each will thread will run a set number of times as set by ‘count’. Each cycle starts by finding a random number between 1 and 2; this is its service Id. If the thread is a client, it will first check all existing provided services. If a service provided id matches the threads service id then a match is found. Once the match is found, the thread sleeps for a random amount of time between 1ms and 500ms. Once this completes, the thread attempts to lock this section of code, so that no other threads can remove a service provided at the same time. In here, it will check that the service id still exists. If yes, then the removal of the service will commence. Else the client will simply return from the thread and post its own service needed. After finishing with the lock, whether the removal was successful or not, the thread releases its lock on the code for other threads. Once the post is made, the thread will sleep for random intervals, checking after each interval if the service posted has been met yet. If the service does not get accepted by a provider, then the thread will timeout, remove the posted service, and run through the next cycle. This procedure is the same for providers, however, providers post the services provided, and look for services needed. A member is a term I will use to describe both Clients and Providers at a higher level.

I have decided, after much consideration, to remove more of the randomness involved in this program. By altering the randomly assigned sleep times to set times of 500ms or a value set by the user, I am able to view the behaviors of the program better. This means I can trial different amounts of threads, loops etc. and be able to compare the results and how they all run. I have also removed the randomness around assigning service Ids by using a seeded Random number. This means that each time the same sequence of random numbers will be generated.

I will be changing the following Pthrough program arguments

* Number of clients/providers: These values are simply the number of clients created and the number of providers created
* Sleep time: The time taken to wait after finding a service that fits the member’s service request.
* Time out: This is the time taken before a service is timed out and the member removes the service and creates a new one. This value is calculated by the users 4th input of maxWaitingLoops. This number is multiplied by the above sleep time to get the total timeout duration.

The values I will be checking upon finishing a test:

* Provided/Needed services added: The number of each type of service added to the bulletin board.
* Services fulfilled: A combined total of the number of services that are successfully fulfilled
* Services taken while waiting: The number of services that are taking while a thread is currently sleeping after having found the required service.
* Services timed out: The number of services that don’t find a match and therefore must be removed by its member.
* Total time: The total time taken, starting from starting each thread to the moment each thread finishes.

Before running the tests I would expect to see a few things.

* As sleep time increases, I believe the amount of services taken while waiting should be increased as this allows for more threads to enter into the same wait cycle waiting on a specific service, also I believe, the total execution time will increase.
* Another thing I would expect to see is as time out duration increases I would expect more services to be fulfilled as the services are waiting round longer allowing other services to come in and take the given service. This will also, however, greatly slow down the execution as when a service does in fact have no match, then it will need to wait longer before discarding the service and creating a new one.
* I would expect to observe the amount of services taken while waiting to increase as number of clients and number of providers increases because this means there are more members competing for the same service.

The following results are averaged from three trials (discarding outliers)

(Home PC, fast 4 cores)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Number clients** | **Number providers** | **Sleep time (ms)** | **Time out (ms)** | **Provided services added** | **Needed services added** | **Services fulfilled** | **Services taken while waiting** | **Services timed out** | **Total time (ms)** |
| 5 | 5 | 500 | 5000 | 15 | 14 | 20 | 1 | 8 | 12566 |
| 2 | 2 | 500 | 5000 | 7 | 5 | 8 | 0 | 4 | 13081 |
| 2 | 2 | 50 | 5000 | 5 | 7 | 8 | 1 | 4 | 1513 |
| 2 | 2 | 5000 | 5000 | 5 | 9 | 6 | 1 | 8 | 180067 |
| 5 | 5 | 1000 | 10000 | 11 | 16 | 25 | 0 | 4 | 31060 |
| 5 | 5 | 10000 | 10000 | 12 | 15 | 23 | 2 | 4 | 300063 |
| 10 | 10 | 1000 | 5000 | 29 | 22 | 49 | 9 | 2 | 17059 |
| 20 | 20 | 1000 | 5000 | 59 | 46 | 95 | 25 | 10 | 26070 |
| 40 | 40 | 1000 | 5000 | 159 | 51 | 190 | 52 | 19 | 27075 |
| 20 | 20 | 10000 | 5000 | 53 | 50 | 97 | 15 | 6 | 270075 |
| 20 | 20 | 1000 | 15000 | 51 | 54 | 95 | 24 | 10 | 57072 |

These results don't exactly back up my expectations above. I believed that as sleep time increased, then services taken while waiting would increase, but this didn’t seem to be the case. The only difference I observed upon increasing sleep time was the amount of time taken to execute the program grew dramatically. I also thought that as time out duration increases, so will services fulfilled, however again, this was not the case. As time out duration increased the only major difference noticed was the great increase in program execution time. This was disappointing, however, my last guess did line up correctly with my results. I believed that the greater the number of clients and providers, the greater the number of services taken while waiting. This theory did in fact stack up as seen in the table above.

Part B)