Event Simulator

This assignment is essentially an event-driven simulation of a bank. For a bank with a certain fixed number of tellers, two settings are considered,

- When there are multiple queues (one for each teller) and,
- When there is a single common queue among the tellers.

The variables of interest which we are trying to measure are,

- Average amount of time spent by the customer in the bank
- Maximum waiting time of a particular customer.
- Total amount of service time and amount of teller idle time.

Execution instructions

In the Top directory, executing 'make' will lead to the compilation of the source codes and the respective object files, executables are then stored in the appropriate locations. Running 'make execute' will execute for different test cases as mentioned shown in the terminal. This will also produce a plot between average waiting time vs number of sellers for each case which was run.

Implementation Details

A struct node is created which represents the general node and contains various attributes such as type, id, status etc according to the description in the code. There are several functions such as:

1. delete() : Pops out the first element of the linked list and returns the size of the new linked list,
2. insertEvent() : Inserts an event into the list event and maintains the sorted order of the list.
3. insertCustomer() : Inserts a customers in the list whose head is specified as one of the argument.

All of these are accessed by *Function Pointers*. For the case of multiple queues, firstly an array of node pointers *tellerQueues* is created whose size equals the number of tellers. Then in another loop, all the customers are instantiated along with the customer arrival events which (events) are pushed in the event queue. The events have a pointer to the instantiated customers. The arrival times are randomly generated between 0 and 60.

Afterward, the simulation code begins, and depending on the event popped out there are three branches as follows:

- If the event has type "CA" (Customer Arrival) then, the customer referred by the event is added to the shortest queue (in case of multiple queues) or the common queue (in case of a single queue). This is followed by the deletion of the event from the event queue.
- If the event has type "CS" (Customer Serviced), the statistics about the customer's service time is collected and the particular teller's count for the number of customers served is incremented. This is followed by the deletion of the event from the event queue.
- If the event has type "TI" (Teller Idle), For the case of a single queue, if there are customers waiting then we set the status of the teller to 0.

Note:
A teller status : 0 indicates availability of the teller to serve the customer and '1' indicates unavailability.
If not then we set the status to 1 and insert another teller idle event. For the case of multiple queues, if there are customers in that particular queue, then we set the status to 0 otherwise 1 and insertion of an idle event.

After these conditions, waiting for customers in the queue are popped from the common queue (in case of a single queue) and tellers individual queues (in case of multiple queues) subject to the availability of tellers. The corresponding CS and TI events are also inserted.

Inference

As per the plot, the average waiting time of the customers (in a single queue scenario) decreases (non-uniformly) with the increase in the number of tellers. That should indeed be the case as well. When tried different realistic combinations, in most cases it turned out that the average time spent by the customer is less for multiple queues, and the maximum waiting time is lesser for the case of the single queue.

Param Khakhar
2018CS10362