April 11, 2016

Project 4 Technical Report

EEEE-663 Real-Time and Embedded Systems

Author(s)	**:**	**Vyoma Sharma, Siddharth Ramkrishnan**
Email ID	**:**	**sg5232@rit.edu, sxr4316@rit.edu**
UID	**:**	**835003392 , 311004697**

Table of Contents

1. Project Requirements

Design and implement an embedded, stand-alone QNX Neutrino program to simulate the workflow in a typical banking environment -- single queue with queuing to a multi-threaded server or three separate server processes.

Problem Statement:

- Customers enter the bank to transact business on a regular basis. Each new customer arrives every one to four minutes, based on a uniform random distribution. Each new customer enters a single queue of all customers.
- Three tellers are available to service customers in the queue. As tellers become available, customers leave the queue, approach the teller and conduct their business. Each customer requires between30 seconds and 8 minutes for their transaction with the teller. The time required for each transaction is based on a uniform random distribution.
- The bank is open for business between the hours of 9:00am and 4:00pm. Customers begin entering when the bank opens in the morning, and stop entering when the bank closes in the afternoon. Customers in the queue at closing time remain in the queue until tellers are available to complete their transactions.

Graduate Extension

- Inclusion random breaks for each of the tellers.
- Each teller will take a break every 30 to 60 minutes for 1 to 4 minutes. If a teller break time occurs while serving a customer they will go on break as soon as they finish the current customer.
- The next break for a teller occurs from 30 to 60 minutes from when they started their previous break.
- A break can only occur after the completion of the current customer transaction. • The break timing and duration is based on a random uniform distribution.

Design Constraints

- You are free to use any QNX supported concurrency mechanism to implement this lab. Be sure to describe your selected architecture in your lab report.
- You may want to create a semaphore to communicate the number of available tellers but other options include QNX message sending and reply blocking.
- Each teller is modelled as an independent thread in the Teller Server or as three separate processes.
- The simulation parameters as described in the Problem Statement can be “hard-coded” as internal constants. However, only define those values in one location – a header file is strongly recommended.
- The simulation time is scaled such that 100 milliseconds of absolute clock time represents 1 minute of simulation clock. Therefore, this program will run about 7 hours x 60 minutes /hour * 0.1 seconds / minute.
- The output must be presented in simulation clock time (9 AM through closing time around 4 PM).
- In order to get the timing right I strongly recommend using timers and then waiting for the timer to complete. Using CPU consuming library functions like usleep are discouraged. Substantially inaccurate simulations will be penalized.

2. Abstract

A simulation of bank operation is performed using *multi-threaded* concurrency implementation. The customer queue is simulated by a thread operating a random delay to introduce a new customer to the queue. Three teller threads operating independently accept customers waiting in the queue, while determining a random transaction time for each customer in queue. An overall timing mechanism limits the operation of the customer queue creation system to the bank working hours. Teller threads would operate even after bank closing hours until the customer queue is empty.

3. Introduction

The stand-alone program simulates bank operation for 7 hours (9:00 AM to 4:00 PM), in which customers are added to a queue every few minutes of the simulated bank clock. Three tellers are independently accepting customers currently in the queue in a First In First Out (FIFO) priority, i.e. the customer who arrives at the earliest is attended first by whichever teller is available. Each customer is assigned a random transaction time, after which the teller is available to accept the next customer. The tellers operate even after bank operation hours if there are customers still waiting in the queue.

3.1. Graduate Extension

The tellers, at random intervals, independent of each other take breaks when they are not actively serving a customer for a random duration. The teller, on a break would be unable to accept a new customer when he is on a break, thereby increasing the overall waiting time of the customers on the queue

4. Area of Focus

The primary areas of focus of this project were the overall bank operation algorithm and the implementation of suitable concurrency operation and its associated data protection. The overall Bank Operational Algorithm was largely handled by Siddharth Ramkrishnan, with Vyoma Sharma focussing on the implementation of the multi-threading concepts and data protection, which was implemented using Mutexes.

5. System Block Diagram (Software Architecture)

5.1. Main Function

The main function initializes the timer which simulates the bank simulation time by 1 second for every 1.6ms of real time clock. Thereby speeding up the simulation by a factor of 625. This results in the overall bank simulation of 7hrs * 60mins * 60 seconds completing in around 40 seconds of real time clock.

Customer Queue Thread is initiated, and its operation is limited to the simulation clock for 25200 seconds, after which no new customers are included in the customer queue.
Multiple Teller threads are initiated and are terminated when bank closes or customer queue is empty, whichever condition is longer.

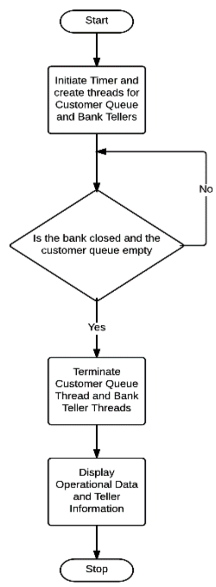

Figure 1. Control Flow Diagram of Main() function

5.2. Customer Queue Updation

The customer updating thread when initiated, adds new customers to the queue when a random delay time is elapsed. The operation of the thread is controlled by a higher priority clock timer, which determines if the thread is allowed to add new customers and also if the determined delay time has elapsed, and a new customer could be created and added to the queue.

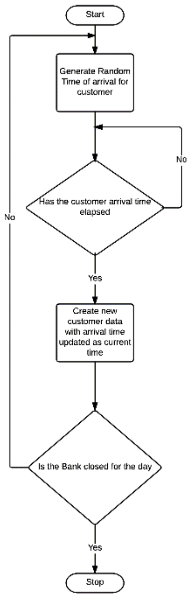

Figure 2. Customer Queue Updation Thread

5.3. Bank Teller Operation

Figure 3. Bank Teller Operation Thread

The bank teller operation is the most important function in terms of operational complexity. The priority of operation is determined as ongoing break, current customer, next break check, next customer, idle. This priority is implemented as a series of nested control and branching statements. The timer functions which monitor the duration of each of the states operate independent of the state determination and exit the state, once the required wait time for each teller action has elapsed.

6. System Operation and Behaviour

6.1. No Teller Break - Bank Simulation Log

```
Bank is now Open for Business. Current Time: 7: 00 AM

Bank is Closed. Viewing Operational Information

Teller Breaks are Disabled

Total Number of Customers Serviced: 153
Average Customer Wait Time         : 181
Average Customer Service Time      : 197
Average Teller Idle Time            : 276
Maximum Customer Wait Time         : 300
Maximum Teller Idle Time           : 5800
Maximum Depth of Customer Queue    : 1

Teller A Operation Information

Number of Customers Served: 53
Maximum Service Time       : 359
Total Service Time         : 10402
Total Idle Time            : 13720
Total Break Time           : 0

Teller B Operation Information

Number of Customers Served: 41
Maximum Service Time       : 353
Total Service Time         : 7626
Total Idle Time            : 17080
Total Break Time           : 0

Teller C Operation Information

Number of Customers Served: 58
Maximum Service Time       : 354
Total Service Time         : 11846
Total Idle Time            : 11440
Total Break Time           : 0
```

6.2. Teller Break - Bank Simulation Log

```
Bank is now Open for Business. Current Time: 7: 00 AM

Bank is Closed. Viewing Operational Information

Teller Breaks are Enabled

Total Number of Customers Serviced: 154
Average Customer Wait Time         : 178
Average Customer Service Time      : 190
Average Teller Idle Time            : 254
Maximum Customer Wait Time         : 280
Maximum Teller Idle Time           : 1980
Maximum Depth of Customer Queue    : 1


Teller A Operation Information
Number of Customers Served: 42
Maximum Service Time        : 351
Total Service Time          : 8436
Total Idle Time             : 14860
Total Break Time            : 1426


Teller B Operation Information
Number of Customers Served: 60
Maximum Service Time        : 337
Total Service Time          : 10132
Total Idle Time             : 12060
Total Break Time            : 1705


Teller C Operation Information
Number of Customers Served: 51
Maximum Service Time        : 353
Total Service Time          : 10728
Total Idle Time             : 12300
Total Break Time            : 1361
```

7. Conclusion

The project was in interesting introduction to the concepts of concurrent operations on a single core system. The impact of parallel threads on standard IOs was an interesting take away from the assignment. The impact of the delay introduced and the data loss associated with the Momentics Environment and the QNX target environment was an unanticipated effect, whose impact was identified and negated for efficient simulation.