CS 3413

Assignment 5

Due Date: October 16th, 2019 at 9:30 am

ASSIGNMENT IS TO BE COMPLETED INDIVIDUALLY BY ALL STUDENTS!

Your solution to the problem, written in C, should be submitted via D2L.

A network device has only one physical link between the computer and the network switch. Simple network devices are only capable of transmitting messages in one direction. The device has to be able to handle requests, both outgoing and incoming. To further complicate its function, it receives outgoing messages into an outgoing buffer and requests for reception to an incoming buffer from multiple applications. To make this work, the network device has to switch between receiving and sending messages. When sending or receiving it has to limit the number of messages so as to not overload the physical link and drop messages.

Write a pthreads program that mimics the network device so that it takes actions (message to send from outgoing buffer or receive request from incoming buffer) and places at most *n (given as first command line argument)* messages on the physical link at any given time, and all of them are either sending or receiving. It takes *s (given as second command line argument)* seconds for a message to be transferred. Only one message can be initiated for transfer at any given time because the device spends one second to start the transfer of each message. The device blocks to save resources if it doesn't send or receive. Use sleep() to simulate transfer and placing work on the device. Implement device functionality in a thread separate from application threads. Applications and their actions are specified by input that your program is to take through stdin, header line is to be ignored. The input table has following format:

Application	Action	Production time
Jimatron	Send	1
Marybot	Send	2
Marybot	Receive	1
Sueborg	Send	3

Exactly one thread has to be created for each application. Any thread can do multiple operations of both kinds. Applications generate actions one at a time in order of input table (Marybot generates an outgoing message first), each action takes a number of seconds to generate (specified in third column) and then place generated action into the matching buffer of the device. The device switches processing of actions from one buffer to another if another buffer is full and *n* actions from the current buffer have been processed since the last switch.

The device reports changes of action and send or receipt of messages. If the value of *s* was 2 and *n* was 2, we would have the following output:

```
Device is idle:         1
Device is sending:      2
5:      Jimatron has sent
6:      Marybot has sent
8:      Sueborg has sent
Device is receiving:    8
11:     Marybot has received
Device is idle:         11
```

You can refer to the below timeline for clarifications:

Action		Time										
application	function	1	2	3	4	5	6	7	8	9	10	11
Jimatron	Send	G	P	T	T							
Marybot	Send	G	G	P	T	T						
Marybot	Receive			G					P	T	T	
Sueborg	Send	G	G	G		P	T	T				

G – being generated by application. P – being placed by device for transferring. T – being transferred.

To assist your programming here are some notes:

- The values of *n* and *s* are provided as command line args. Therefore the command line for your program is: ./a.out *n s*
- The network device is a thread.
- Outgoing messages are stored in a FIFO buffer. Incoming requests are stored in a separate FIFO buffer. The capacity of each buffer is 4 messages.
- You **cannot** lose messages! So, applications will block if they cannot put their messages in the buffer.