```c
#include "uart.h"
#include <cmsis_os2.h>
#include <lpc17xx.h>
#include <stdio.h>
#include "random.h"

//defining global constants
#define K 10
#define N 2
#define SERVER_FREQ 10
#define CLIENT_FREQ 9

typedef struct {

  osMessageQueueId_t q_id;
  double sleep_time;
  double avg_loss_ratio;
  double avg_arrival_rate;
  double avg_service_rate;
  double elapsed_time;
  uint32_t dropped;
  uint32_t total_sent;
  uint32_t total_recieved;

} MSGQUE_t;

//This will just set the LEDs to the binary representation of a given unsigned char.
//Quite useful for debugging.
void charToBinLED(unsigned char c)
{
    if(c&1)
      LPC_GPIO1->FIOSET |= 1<<28;
    else
      LPC_GPIO1->FIOCLR |= 1<<28;
    if(c&2)
      LPC_GPIO1->FIOSET |= 1<<29;
    else
      LPC_GPIO1->FIOCLR |= 1<<29;
    if(c&4)
      LPC_GPIO1->FIOSET |= 1U<<31;
    else
      LPC_GPIO1->FIOCLR |= 1U<<31;
    if(c&8)
      LPC_GPIO2->FIOSET |= 1<<2;
    else
      LPC_GPIO2->FIOCLR |= 1<<2;
    if(c&16)
      LPC_GPIO2->FIOSET |= 1<<3;
    else
      LPC_GPIO2->FIOCLR |= 1<<3;
    if(c&32)
      LPC_GPIO2->FIOSET |= 1<<4;
    else
      LPC_GPIO2->FIOCLR |= 1<<4;
    if(c&64)
      LPC_GPIO2->FIOSET |= 1<<5;
    else
      LPC_GPIO2->FIOCLR |= 1<<5;
    if(c&128)
      LPC_GPIO2->FIOSET |= 1<<6;
    else
      LPC_GPIO2->FIOCLR |= 1<<6;
}

//set the LED pins to be outputs
void initLEDPins()
{
  //set the LEDs to be outputs. You may or may not care about this
  LPC_GPIO1->FIODIR |= 1<<28; //LED on pin 28
  LPC_GPIO1->FIODIR |= 1<<29;
  LPC_GPIO1->FIODIR |= 1U<<31;
  LPC_GPIO2->FIODIR |= 1<<2;
```

```
73      LPC_GPIO2->FIODIR |= 1<<3;
74      LPC_GPIO2->FIODIR |= 1<<4;
75      LPC_GPIO2->FIODIR |= 1<<5;
76      LPC_GPIO2->FIODIR |= 1<<6;
77    }
78
79    //create a single message queue for testing purposes only. It will be initialized in main
80    osMessageQueueId_t q_id;
81
82    /*
83
84      Our client will send a message once every second
85
86    */
87
88    //main array of queues
89    MSGQUE_t msgqueue[N];
90
91    //client function
92    void client(void* args)
93    {
94      int msg = 0;
95      MSGQUE_t *newque = (MSGQUE_t*) args;
96
97      while(1)
98      {
99        double delay,tickfreq;
100
101       //counting total number of messages successfully sent
102       newque->total_sent++;
103
104       //sending a message in client thread
105       osStatus_t stat = osMessageQueuePut(newque->q_id,&msg,0,0);
106
107       //checking if message was successfully put into the queue
108       if(stat==osOK)
109       {
110         newque->total_recieved++;
111       }
112       //the message queue is full and if the message was lost
113       else
114       {
115         newque->dropped++;
116       }
117
118       //delay for tick frequency number of ticks. This means 1 second.
119       //We do not need to yield because of this delay
120       tickfreq = osKernelGetTickFreq();
121       delay = get_random_delay_seconds(CLIENT_FREQ, tickfreq);
122       osDelay(delay);
123
124     }
125
126   }
127
128   //server function
129   void server(void* args)
130   {
131     int receivedMessage=0;
132     MSGQUE_t *newque = (MSGQUE_t*) args;
133
134     while(1)
135     {
136
137       //receiving a message in server thread
138       osStatus_t stat = osMessageQueueGet(newque->q_id,&receivedMessage,NULL,osWaitForever);
139
140       //delay for tick frequency number of ticks. This means 1 second.
141       //We do not need to yield because of this delay
142       double delay,tickfreq;
143       tickfreq = osKernelGetTickFreq();
144       delay = get_random_delay_seconds(SERVER_FREQ, tickfreq);
```

```
145       osDelay(delay);
146
147       //adding to sleep time
148       newque->sleep_time += (delay/tickfreq);
149       //adding to elapsed time
150       newque->elapsed_time += (delay/tickfreq);
151
152       //we're just going to print the message to the LEDs, mod 256:
153       charToBinLED((unsigned char)(receivedMessage % 256));
154
155       //We need to yield because it is possible that this thread wakes and sees
156       //a message right away and we aren't using priority in this course
157       osThreadYield();
158
159     }
160
161   }
162
163   //monitor function
164   void monitor(void *args)
165   {
166     uint32_t count=0;
167
168     while (1)
169     {
170       count++;
171       //Once per Second
172       osDelay(osKernelGetTickFreq());
173
174       for(int i = 0; i < N; i++)
175       {
176         //Average Message Loss ratio (number of overflows divided by total messages sent)
177         msgqueue[i].avg_loss_ratio = ((double)(msgqueue[i].dropped)/(double)(msgqueue[i].total_sent));
178
179         //Average Message Arrival Rate(total number of messages sent divided by the elapsed time)
180         msgqueue[i].avg_arrival_rate = ((double)(msgqueue[i].total_sent)/(msgqueue[i].elapsed_time));
181
182         //Average Service Rate (total number of received messages divided by the sum of server random
    sleep time)
183         msgqueue[i].avg_service_rate = ((double)(msgqueue[i].total_recieved)/(msgqueue[i].sleep_time));
184
185         //printing all values to putty
186         printf("%d,%d,%d,%d,%f,%f,%f,%f\n",count,msgqueue[i].total_sent,msgqueue[i].total_recieved,
187         msgqueue[i].dropped, msgqueue[i].elapsed_time, msgqueue[i].avg_loss_ratio,
188         msgqueue[i].avg_arrival_rate, msgqueue[i].avg_service_rate);
189
190       }
191
192     }
193
194   }
195
196   //main function
197   int main(void)
198   {
199     //always call this function first
200     SystemInit();
201     initLEDPins();
202
203     //we need to initialize printf outside of any threads
204     printf("Project 4 ready\n");
205
206     //initialize the kernel
207     osKernelInitialize();
208
209     //initializing client and server while making new queue.
210     for (int i = 0; i < N; i++)
211     {
212       //initialize our message queue: 10 messages that are integers with default parameters (the "NULL"
    part means "Let the OS figure out the configuration")
213       msgqueue[i].q_id = osMessageQueueNew(K,sizeof(int),NULL);
214
```

```
215       //set up the threads
216       osThreadNew(client,&msgqueue[i],NULL);
217       osThreadNew(server,&msgqueue[i],NULL);
218
219     }
220
221     //initializing montior thread.
222     osThreadNew(monitor, NULL, NULL);
223
224     //starting the kernal
225     osKernelStart();
226
227     while(1);
228   }
229
```