G11 – Tuesday

Lab 4 Report:

1)

	B					
N		1	2	4	8	10
	20	0.000228	0.000233	0.000238	0.000238	0.000239
	40	0.000232	0.000232	0.000232	0.000247	0.000238
	80	0.000234	0.00024	0.000237	0.00023	0.000232
	160	0.000232	0.00023	0.000228	0.000232	0.000244
	320	0.000233	0.00023	0.000235	0.000238	0.00023

Table 1 - Average system initialization time

	B					
N		1	2	4	8	10
	20	0.000081	0.000087	0.000126	0.000116	0.000129
	40	0.000096	0.000115	0.000066	0.000153	0.000124
	80	0.000098	0.000107	0.000114	0.000075	0.000107
	160	0.000119	0.000074	0.000074	0.000096	0.000138
	320	0.000103	0.000097	0.000102	0.000162	0.000082

Table 2 - Standard deviation of the system initialization time

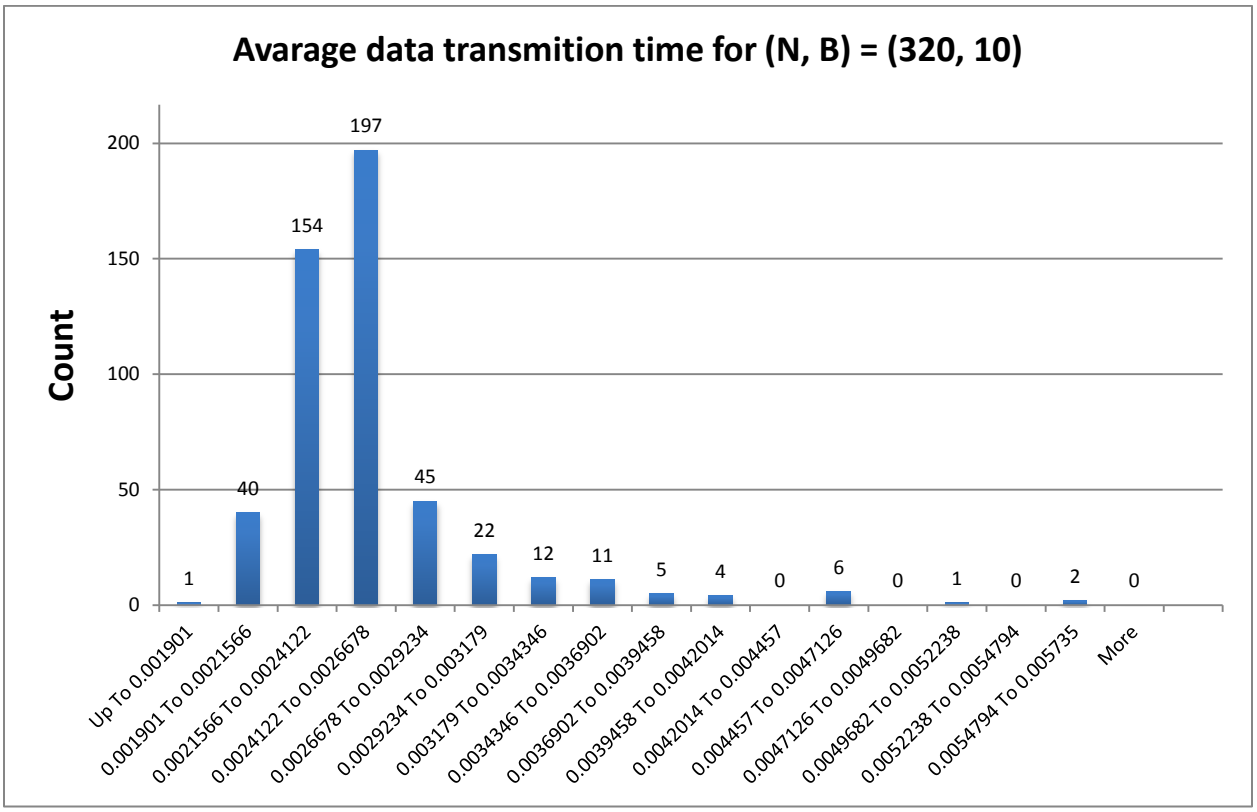
	B					
N		1	2	4	8	10
	20	0.001548	0.001547	0.00154	0.001507	0.001496
	40	0.001626	0.001579	0.0016	0.001572	0.001554
	80	0.001782	0.001714	0.001719	0.001703	0.001693
	160	0.00218	0.002064	0.001985	0.001943	0.001961
	320	0.003111	0.002847	0.002663	0.002587	0.002575

Table 3 - Average data transmission time

	B					
N		1	2	4	8	10
	20	0.000169	0.000196	0.000195	0.000171	0.0002
	40	0.000194	0.000228	0.000213	0.000203	0.000185
	80	0.000194	0.000243	0.000266	0.000199	0.000203
	160	0.000341	0.000189	0.000256	0.000195	0.000224
	320	0.000568	0.00054	0.000452	0.000593	0.000476

Table 4 - Standard deviation of the data transmission time

2) HISTOGRAM:



3)

A discussion of how the system initialization time and the data transmission time are related to the values of different (N,B) pairs on ecelinux :

The system initialization time remains almost the same for different values of N and B. Hence it does not depend on the values of N and B.

The data transmission time changes when the value of N and B are different.

As N increases and B remains constant, it is observed from the tables that the data transmission time also increases.

As B increases and N remains constant, it is observed from the tables that the data transmission time decreases.

Appendix:

Producer: (produce.c)

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <string.h>
#include <mqueue.h>
#include <sys/stat.h>
#include <time.h>
#include <sys/time.h>
#include <sys/wait.h>

//initializing the timing variables
double t1, t2, t3;
struct timeval tv;

int spawn (char* program, char* b, char* n)
{
  pid_t child_pid;

  /* Duplicate this process.  */
  gettimeofday(&tv, NULL);
  t1 = tv.tv_sec + tv.tv_usec/1000000.0; //Time before the fork
  child_pid = fork ();
  gettimeofday(&tv, NULL);
  t2 = tv.tv_sec + tv.tv_usec/1000000.0; //Time before the first integer is
generated

//arg_list contains a list of arguments that will be passed to the consumer
  char* arg_list[] = {
      "consumer", //argv[0], the name of the program
      b, //The size of the queue mailbox
      n, //The number of items to be produced by the producer
      NULL //the argument list MUST end with NULL
  };

  if (child_pid != 0) {
    /* This is the parent process.  */
    return child_pid;
 }
  else {
    /* Now execute PROGRAM, searching for it in the path.  */
    execvp (program, arg_list);
    /* The execvp function returns only if an error occurs.  */
    fprintf (stderr, "an error occurred in execvp\n");
    abort ();
  }
}

int main(int argc, char *argv[])
{

        int child_status;
        mqd_t qdes;
```

```
        char  qname[] = "/mailbox1_G11"; //queue name must start with '/'
        mode_t mode = S_IRUSR | S_IWUSR;
        struct mq_attr attr;
        int i;

        if ( argc !=3 ) {
           printf("You have to enter: ./produce <N> <B>");
                printf("<N> = number of integers the producer should produce\n");
           printf("<B> = number of integers the message queue can hold\n");
                exit(1);
        }
        spawn("./consumer.out", argv[2], argv[1]);


        //the maximum number of messages in the queue. <B>
        attr.mq_maxmsg  = atoi(argv[2]);
        attr.mq_msgsize = sizeof(int);
        attr.mq_flags   = 0;            /* a blocking queue  */

        qdes  = mq_open(qname, O_RDWR | O_CREAT, mode, &attr);
        if (qdes == -1 ) {
                perror("mq_open() failed");
                exit(1);
        }

        srand(time(NULL));
        for(i=0; i<atoi(argv[1]); i++) {
                int rand_num;
                rand_num = rand()%100; //Generating a random number
                //Sending a message to the queue
                if (mq_send(qdes, (char *)&rand_num , sizeof(int), 0) == -1) {
                        perror("mq_send() failed");
                }
        }

      //Waiting for the consumer to finish consuming
       wait(&child_status);

        gettimeofday(&tv, NULL);
       //Time after the last integer is consumed and displayed
        t3 = tv.tv_sec + tv.tv_usec/1000000.0;
       //Calculating the initialization time
        printf("Time to initialize system: %f seconds\n", t2-t1);
      //Calculating the data transmission time
        printf("Time to transmit data: %f seconds\n", t3-t2);
        if (mq_close(qdes) == -1) {
                perror("mq_close() failed");
                exit(2);
        }

        if (mq_unlink(qname) != 0) {

            perror("mq_unlink() failed");
                exit(3);
        }
        return 0;
}
```

Consumer (consumer.c):

```
#include <stdbool.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <mqueue.h>
#include <sys/stat.h>
#include <signal.h>
#include <unistd.h>
#include <sys/time.h>


#define _XOPEN_SOURCE 600

int i;

int main(int argc, char *argv[])
{
        mqd_t qdes;
        char  qname[] = "/mailbox1_G11";
        mode_t mode = S_IRUSR | S_IWUSR;
        struct mq_attr attr;

        attr.mq_maxmsg  = atoi(argv[1]); //Receiving "b" from producer
        attr.mq_msgsize = sizeof(int); // The size of each message
        attr.mq_flags   = 0;    /* a blocking queue  */

        //Opening the correct mailbox
        qdes  = mq_open(qname, O_RDONLY, mode, &attr);
        if (qdes == -1 ) {
                perror("mq_open()");
                exit(1);
        }

        //Consuming "N" integers that were produced by producer
        for(i=0 ; i<atoi(argv[2]) ; i++) {
                int rand_num;

                //Receiving the messages sent by the producer
                if (mq_receive(qdes, (char *)&rand_num, sizeof(int), 0) == -1) {
                } else {
                        printf("%d is consumed\n",rand_num);
                }
        }

       //Closing the mailbox and checking for errors in the process
        if (mq_close(qdes) == -1) {
                perror("mq_close() failed");
                exit(2);
        }

        return 0;
}
```