|  |  |  |  |
| --- | --- | --- | --- |
| BN | Section | Student ID | اسم الطالب |
| **39** | **4** | **9220982** | **يوسف خالد عبد الرؤف محمد زهره** |
| **40** | **4** | **9220984** | **يوسف خالد عمرمحمود** |

# System Design

## Flow Chart

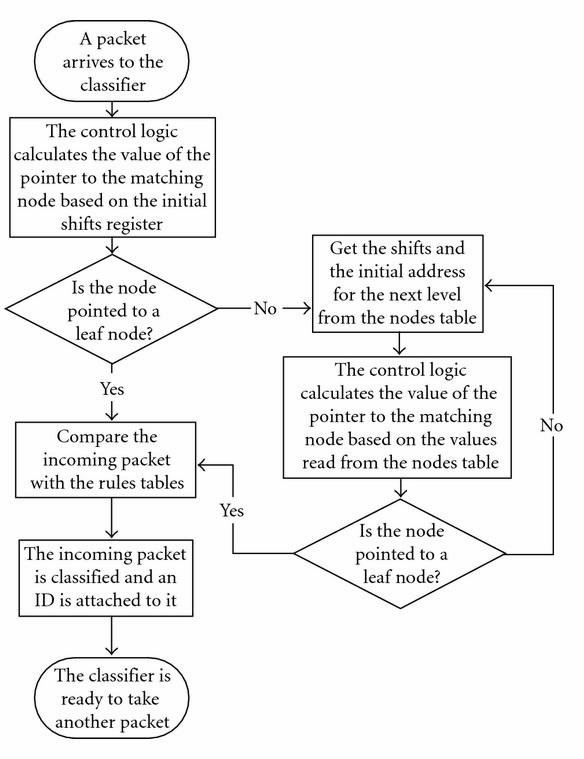
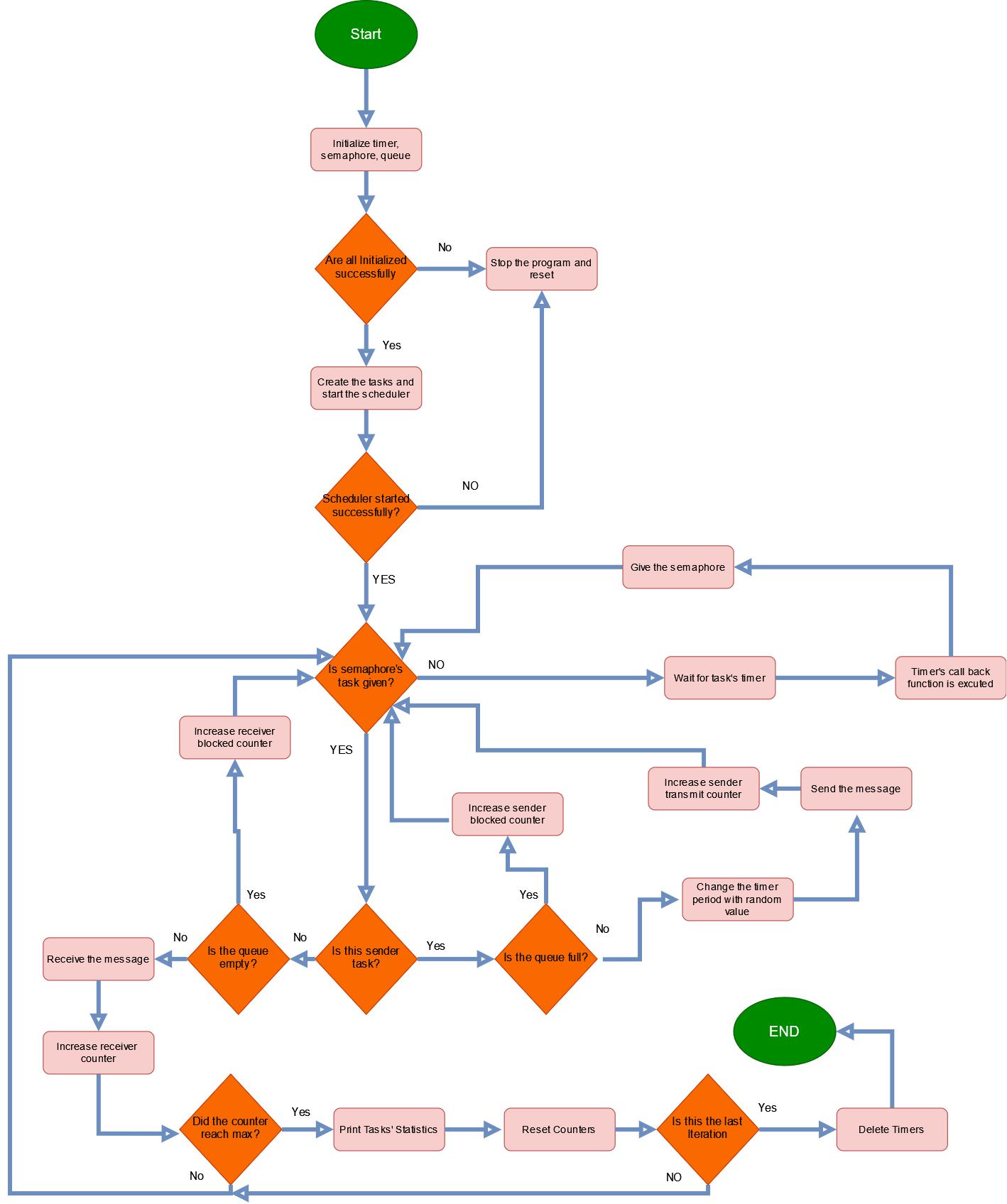


Figure 1: System Flow Chart

## Main Structure:

As shown in figure [2], The main data structure is queue.[2]

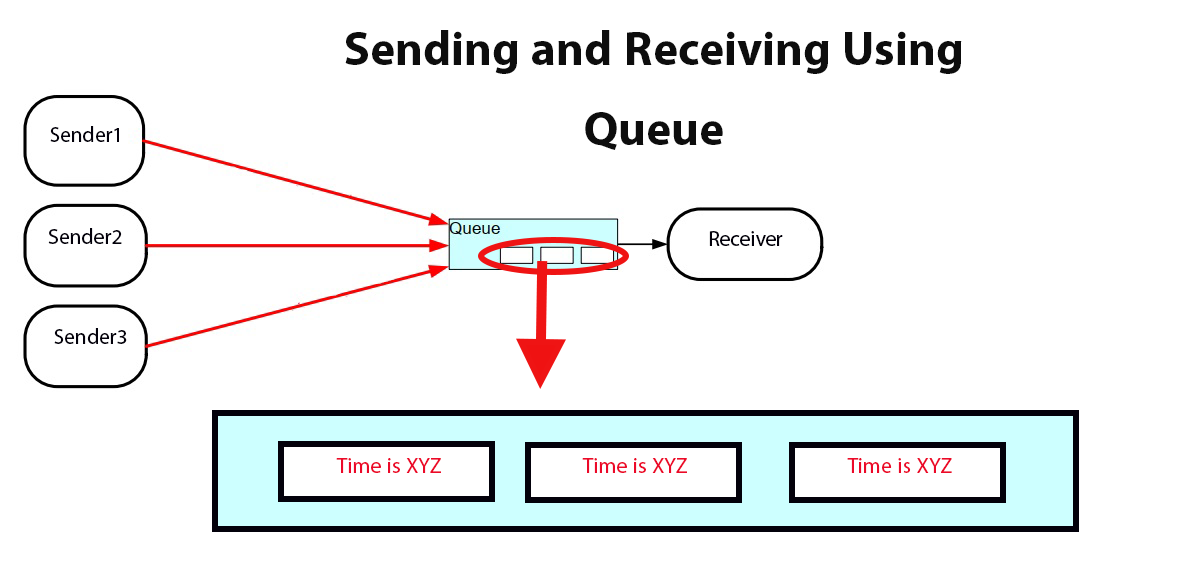


Figure 2: Communication through Queue

## Timeline:

As shown in figure [3], The recevier task has the highest pioroty so it pre-empts the tasks.[2]

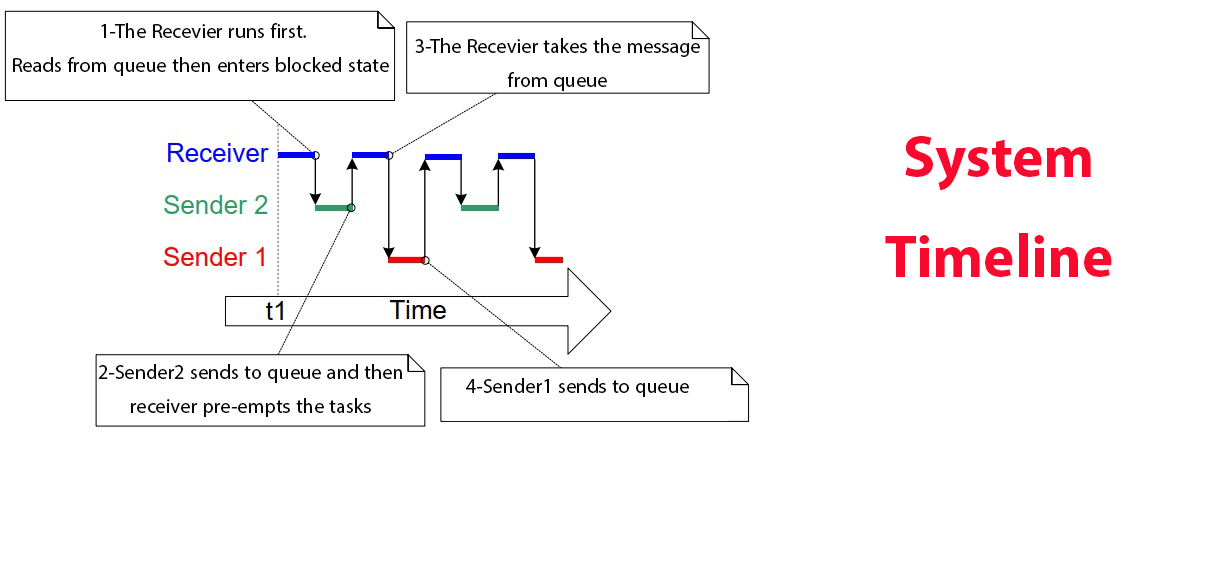


Figure 3: System Timeline

## Code Snippets:

A white background with black text

Description automatically generatedThis is the function that gives random uniform distribution value based on given boundaries.[1]

## Pseudo Code:

### The Sender:

## The Sender waits for the semaphore and then send message if queue’s not full.

SenderTask(){

while(1){

if (semaphore is available){

if(queu is full){

blockedcounter++;

}

else{

message="Time is XYZ"

QueueSend(message);

sendercounter++;

changePeriod(SenderTimer,random value);

}

}

}

### The Receiver:

## The Receiver waits for the semaphore and then receive message if queue’s not empty.

ReceiverTask(){

while(1){

if (semaphore is available){

QueueReceive(message);

if(message recived){

receivedcounter++;

if(receivedcounter>=max\_message){

printstatistics();

resetcounters();

iteration++;

if (last boundry is used)

endProgram();

}

}

}

}

}

### Main Function:

The main creates everything needed for the system and start the timers and scheduler.

main()

{

create semaphores,timers,queue;

create sendersTask,receiverTask;

settimer(sendersTimer,random value);

settimer(receiverTimer, constant value);

if(something went wrong in creation)

return;

startTimers;

StartScheduler();

return 0;

}

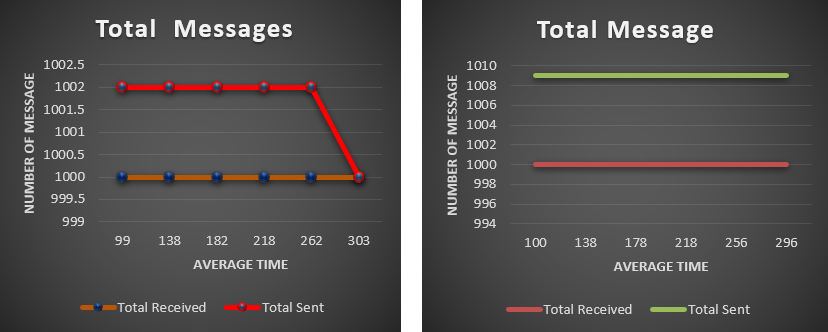
# Results and Discussion

The figures on the right are when queue size=3, left is queue size=10.

A graph of time and send

Description automatically generatedIn figure [4], We can see that the Blocked number decays as the range of boundary increases.

Figure 4: number of blocked messages vs Avg Tsender

In figure [5], There’s a gap between total received and sent as the queue is filled by the senders and the receiver takes the last message.

Gap = queue size-1

Gap = queue size-1

Figure 5: Total Received and Sent vs Avg Tsender

The figure on the top are when queue size=3, bottom is queue size=10.

In figure [6], We can see that even if there’s higher priority task the results are random.



Figure 6: Senders Tasks’ Counters in each Iteration

# References

1. "Generating a uniform distribution of INTEGERS in C," Stack Overflow, <https://stackoverflow.com/questions/3063168/generating-a-uniform-distribution-of-integers-in-c>. Accessed: Jun. 18, 2024.
2. "Mastering the FreeRTOS Real Time Kernel - A Hands-On Tutorial Guide," Real Time Engineers Ltd., 2016. Chapter 4