BN	Section	Student ID	اسم الطالب
21	**2**	**9220377**	**سهيله احمد محمد**
9	**2**	**9220291**	**رؤى عاطف محمد عبدالمجيد**

1 System Design

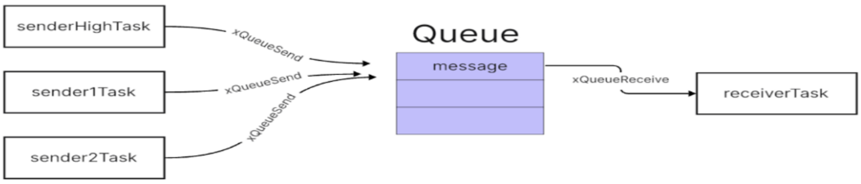


Figure 1 : Data structure design

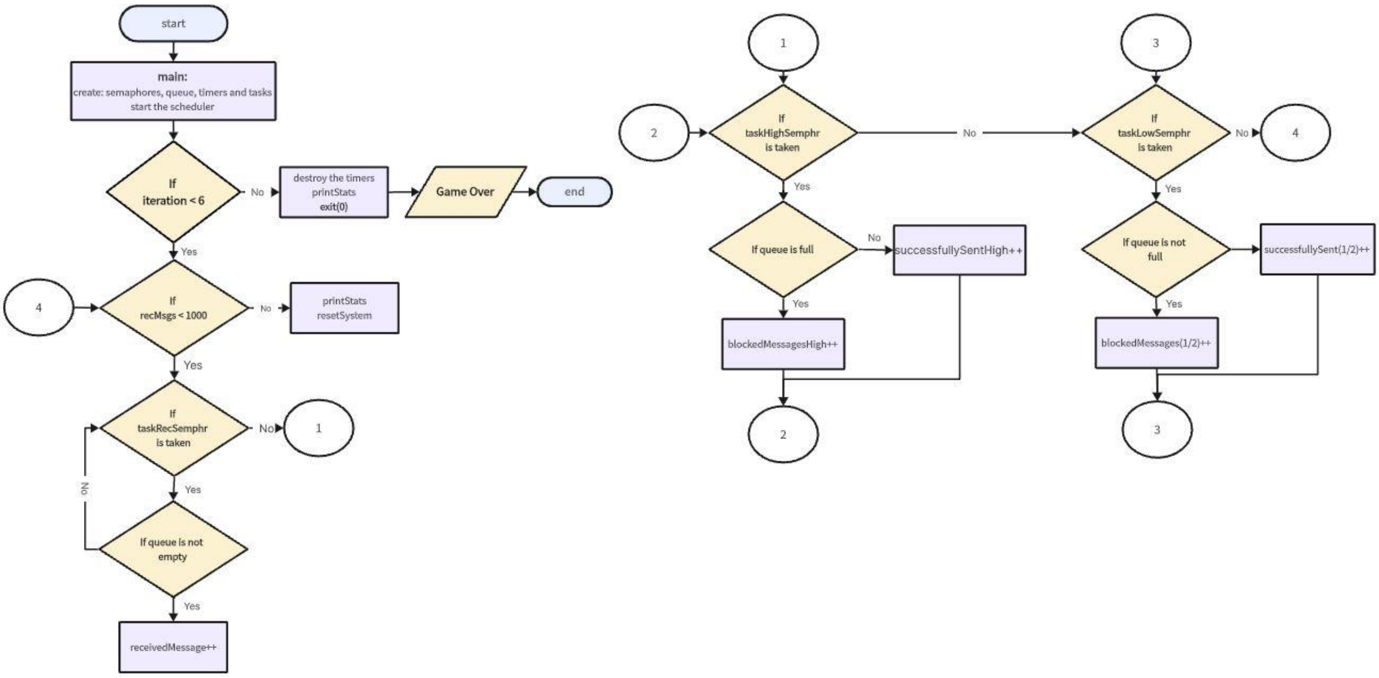


Figure 2: Message Sequence showing object interaction

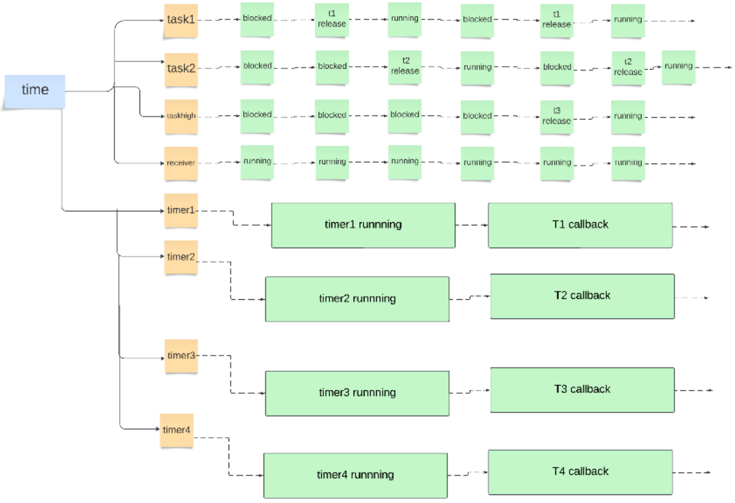


Figure 3: Message Sequence showing object interaction

2 Results and Discussion

Table 1:Queue size=3

iteration	Total sent	Total successfully sent	Total blocked	Sender1 sent	Sender1 blocked	Sender2 sent	Sender2 blocked	Sender High sent	Sender High blocked	Avg Tsender for all tasks	Avg Tsender1	Avg Tsender2	Avg Tsender High
1	3037	1002	2035	367	653	328	683	307	699	98	98	98	99
2	2148	1002	1146	333	382	358	365	311	399	139	139	138	140
3	1657	1002	655	341	209	326	225	335	221	181	181	181	179
4	1355	1002	353	341	112	331	121	330	120	221	220	221	222
5	1155	1002	153	330	54	346	40	326	59	260	260	259	260
6	1010	1001	9	335	2	336	2	330	5	301	301	300	303

Table 2:Queue size =10

iteration	Total sent	Total successfully sent	Total blocked	Sender1 sent	Sender1 blocked	Sender2 sent	Sender2 blocked	Sender High sent	Sender High blocked	Avg Tsender for all tasks	Avg Tsender1	Avg Tsender2	Avg TsenderHigh
1	3037	1009	2028	350	673	339	662	320	693	98	97	99	98
2	2147	1009	1138	341	382	327	382	341	374	139	138	141	139
3	1658	1009	649	336	219	330	220	343	210	181	180	181	181
4	1355	1009	346	338	110	344	112	327	124	221	223	219	221
5	1154	1009	145	329	51	334	51	346	43	260	262	260	257
6	1001	1001	0	334	0	331	0	336	0	302	301	304	300

Table 3 : plotting for queue =3 and queue =10

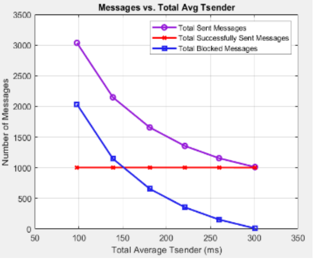
<table>
<tr>
<td>Queue = 3</td>
<td>


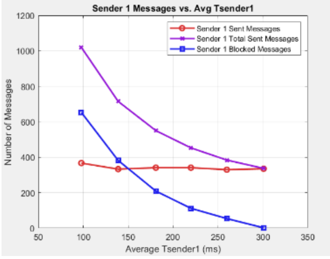
figure 4: total messages versus avg Tsender total
</td>
<td>


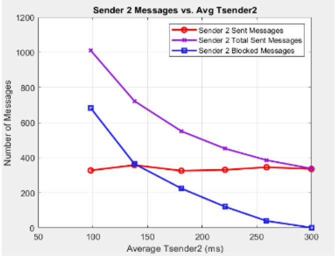
figure 5: sender1 total messages versus avg Tsender1
</td>
<td>


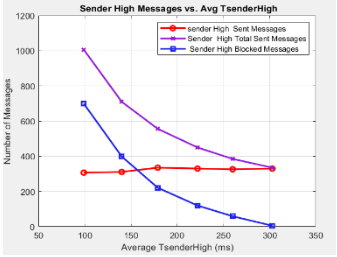
figure 6: sender2 total messages versus avg Tsender2
</td>
<td>


figure 7: sender high total messages versus avg Tsenderhigh
</td>
</tr>
</table>

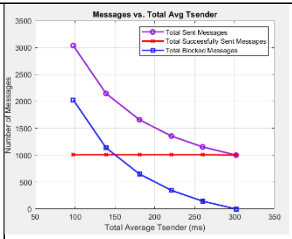
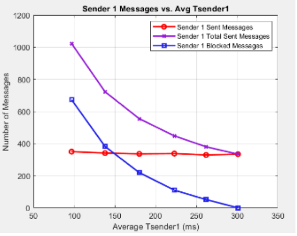
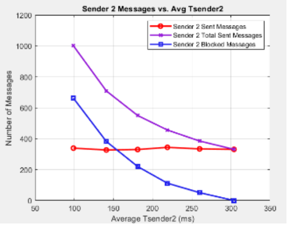
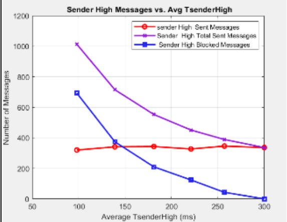
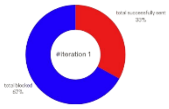
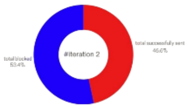
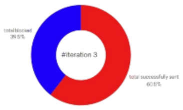
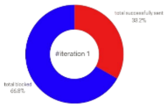
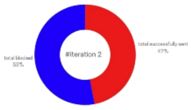
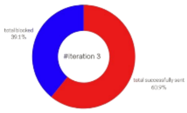
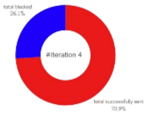
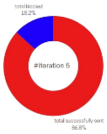
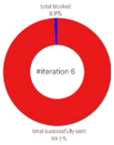
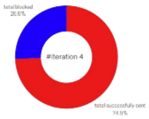
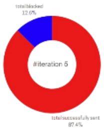
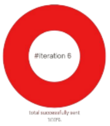
Queue = 10	 figure 8: total messages versus avg Tsender total	 figure 9: sender1 total messages versus avg Tsender1	 figure 10: sender2 total messages versus avg Tsender2	 figure 11: sender high total messages versus avg Tsender high

Table 4:Comparison between queue = 3 and queue = 10 in total messages:

Queue = 3			Queue = 10		
 	 	 	 	 	
 	 	 	 	 	

Comparison between queue = 3 and queue = 10 in total messages for sender1, sender2,sender high :

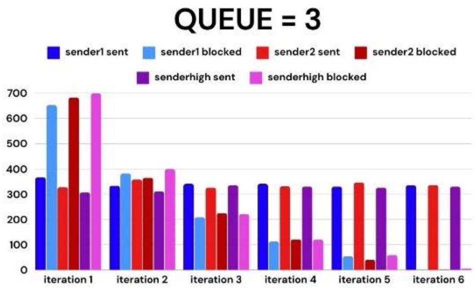


Figure 11:senders results in queue 3 in different iterations

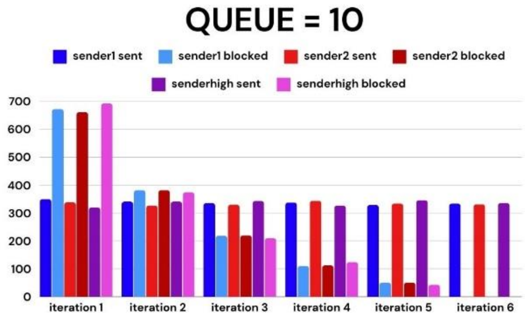


Figure 12: senders results in queue 10 on different iterations

3 Observations

1. Explain the gap between the number of sent and received messages in the running period.

Due to the randomness of Tsender: Tsender < Trec means faster send operation and Trec < Tsender means faster receive operation

2. What happens when queue size increases?

Sent messages increase and blocked messages decrease as there is more space in the queue.

4 Code snippets

```
int randomTime(int lower, int upper){

    return ((rand() % (upper - lower + 1)) +
lower);}
```

This function uses lower and upper limit of the current iteration to generate random Tsender periode

```
void resetSystem(){

    xQueueReset(Global_Queue_Handle);   //  Clear
queue

    if (iteration < 6)  // Configure upper and
lower limits

    {lowerLimit = arr1[iteration];

    upperLimit = arr2[iteration];  iteration++; }

    //you have reached the end of the arrays of
the bounds else

    {// Destroy all timers

        xTimerDelete(xTimer1, 0);

        xTimerDelete(xTimer2, 0);

        xTimerDelete(xTimer3, 0);

        xTimerDelete(xTimer4, 0);


printStats();printf("Gameover\n");exit(0);}}
```

When iterations reach 6, this function destroys the timers, prints the statistics, prints Game Over, and terminates the program

```
void receiver_task(void *p)//Receiver Task{

char str[20];

BaseType_t status;

while (1){

if(xSemaphoreTake(taskRecSemphr, portMAX_DELAY) ==
pdTRUE){

 status = xQueueReceive(Global_Queue_Handle, &str,
0);

if (status == pdPASS){receivedMessage++;}}}}
```

When the receiver task wakes up it takes the semaphore and tracks receive messages

```
int getAverageTsender(totalTsender,tsenderCount){

    if (tsenderCount == 0){ return 0; // Avoid
division by zero }

    return (totalTsender / tsenderCount);}
```

This function calculates total average Tsender and average Tsender for each task separately

```
void senderHigh_task(void *p) { char str[20];
TickType_t XYZ;

    BaseType_t status; while (1)

    {XYZ = xTaskGetTickCount();

if (xSemaphoreTake(taskHighSemphr, portMAX_DELAY)
== pdTRUE)

 {       sprintf(str,  "Time  is  %lu",  XYZ);
printf("%s\n", str);

         status                              =
xQueueSend(Global_Queue_Handle, &str, 0);

  if (status != pdPASS) {blockedMessagesHigh++; }
else

    {successfullySentHigh++;}}}}
```

We have 2 sender tasks of the same priority and one of a higher priority. When they wake up they take the semaphore and track the sent messages

```
static     void     timer1Callback(TimerHandle_t
xTimer)//sender

{   xSemaphoreGive(task1Semphr);

    Tsender = randomTime(lowerLimit, upperLimit);

    // Update the average total tracking variables
for Tsender

    totalTsender += Tsender;  tsenderCount++;

    //update average tracking variables for this
task

    totalTsender1 += Tsender;  tsenderCount1++;


if(!xTimerChangePeriod(xTimer,pdMS_TO_TICKS(Tsend
er),0))

    {puts("Failed to change the period\n");}}
```