# 1- System Design

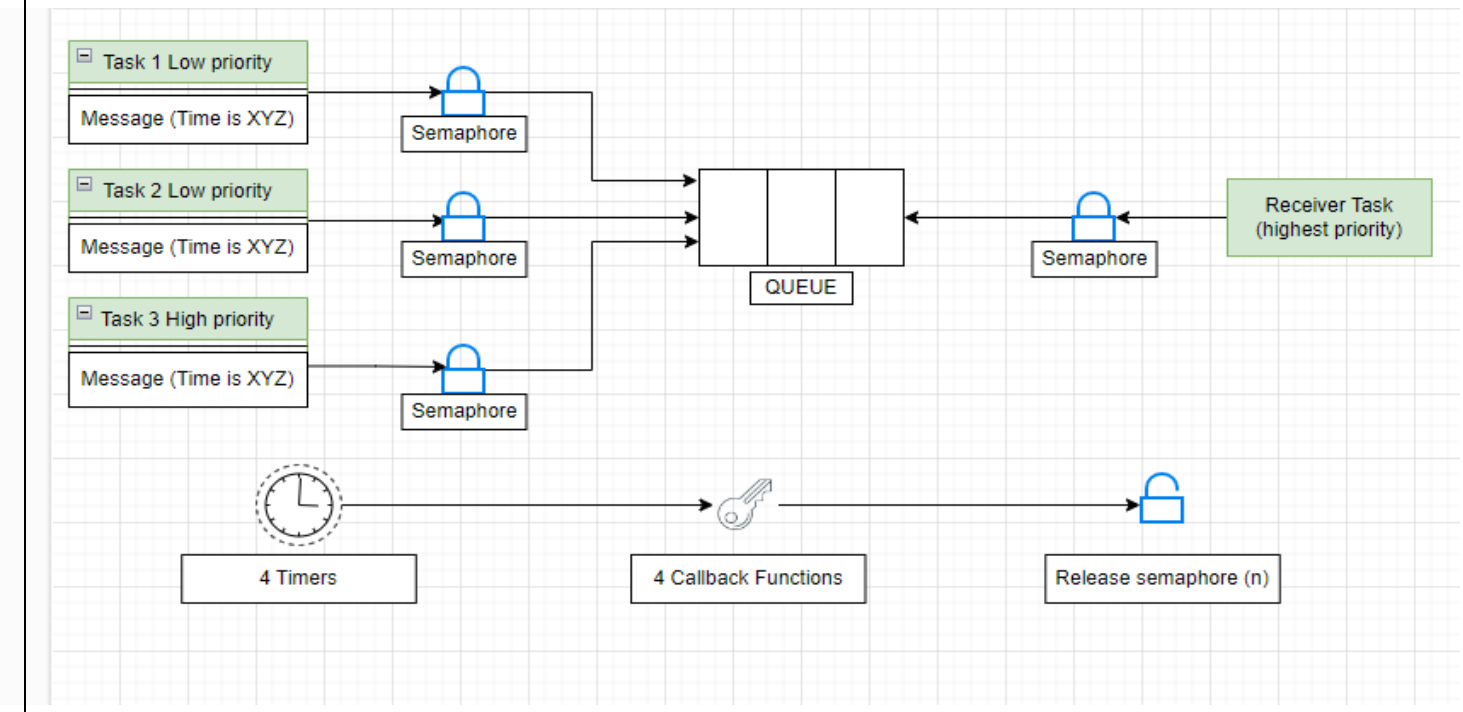



Figure 1: Simple diagram of how the program works

## 1.2 Main structures

- **3 sender tasks and one receiver task**

Two sender tasks with equal priority one, one sender task with priority two, and a receiver task with the highest priority equal to three. These tasks communicate with each other through a queue, as shown in Figures 2 and 3

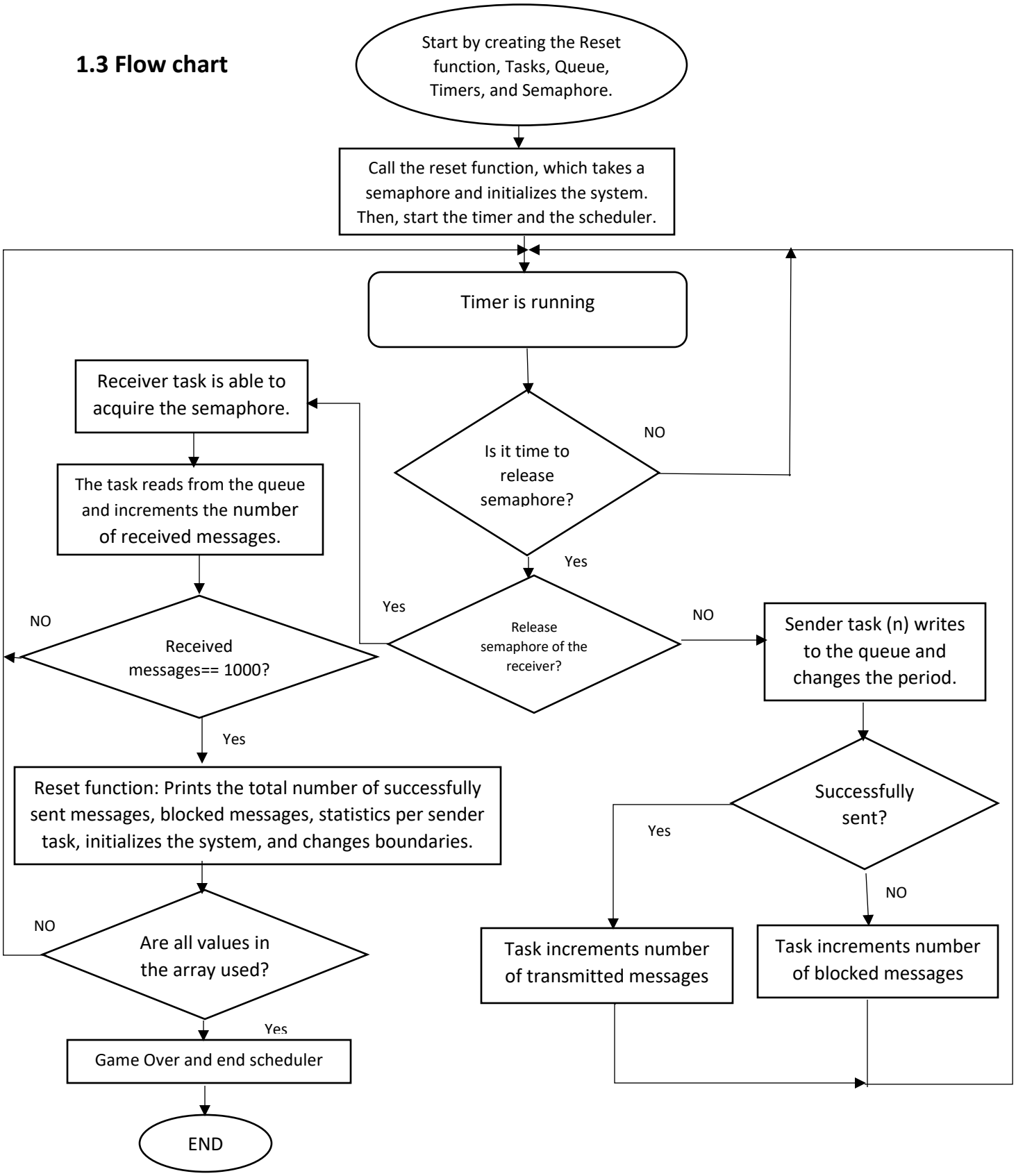
- **Queue**

Queue of specified length which provides communication between tasks.

- **Timer**

Four timers, one for each task which release the **semaphore** and call the **reset function** at the end of the program as shown in Figure(4).

## 1.3 Flow chart



## 1.4 Quick overview of the code

### Reset function

```
void reset()
{
	if(state==0)
	{   state++;
	     (System initialization);
	    xQueueReset(MyQueue);
	    xSemaphoreTake(semaphore1, 0);xSemaphoreTake(semaphore2, 0);
	    xSemaphoreTake(semaphore3, 0);xSemaphoreTake(semaphore4, 0);
	    return;
	}
	if(RecivedCounter==1000&&state>0&&state<7)
		{
		 state++; (System initialization);
		}
	if(state==7)
	{
		printf("%s\n","Game Over"); vTaskEndScheduler();
	}
}
```

This code performs three main functions: Firstly, upon program execution, it initializes the system. Secondly, when the received message counter reaches 1000, it not only reinitializes the system but also prints task statistics, adjusts the period boundary, and finally, it stops the timer and tasks concludes all iterations.

### Callback functions

```
static void SCallback1( TimerHandle_t xTimer )
{
	xSemaphoreGive(semaphore1);
}

static void SCallback2( TimerHandle_t xTimer )
{
	xSemaphoreGive(semaphore2);
}

static void SCallback3( TimerHandle_t xTimer )
{
	xSemaphoreGive(semaphore3);
}

static void RCallback( TimerHandle_t xTimer )
{
	xSemaphoreGive(semaphore4);
	reset();
}
```

This code demonstrates the process of utilizing callback functions to release semaphores and call the reset function.

## 2- Results and Discussion

We have noticed that the gap between the number of sent and received messages arises due to the fast pace at which the 3 senders operate. The receiver, however, can only read one message at a time from the queue, even if multiple messages are present. Consequently, a substantial number of messages become blocked if the queue has a small size or many messages will be sent and will be ignored if the queue has a very large size.

We have observed that the size of the queue has a direct impact on the number of successful sent and blocked messages. Increasing the queue size results in a corresponding decrease of blocked messages as illustrated in Figure (8), which represents the results obtained using a queue size of 10. However, we have also found that the **period of the sender** tasks has a **greater influence** on the sent messages. Notably, when the period of sender tasks is increased, there is a decrease in blocked messages, as it allows the receiver sufficient time to read the queue, and there is a slight discrepancy between the number of received messages and the total number of successfully sent messages. This occurs because the receiver only reads just 1000 messages, and does not continue reading despite the presence of additional messages in the queue.

As illustrated in Figures (4), (5), and (6), it is evident that an increase in the period of sender tasks leads to a decrease in the number of blocked messages.

## 3- Tables and Figures

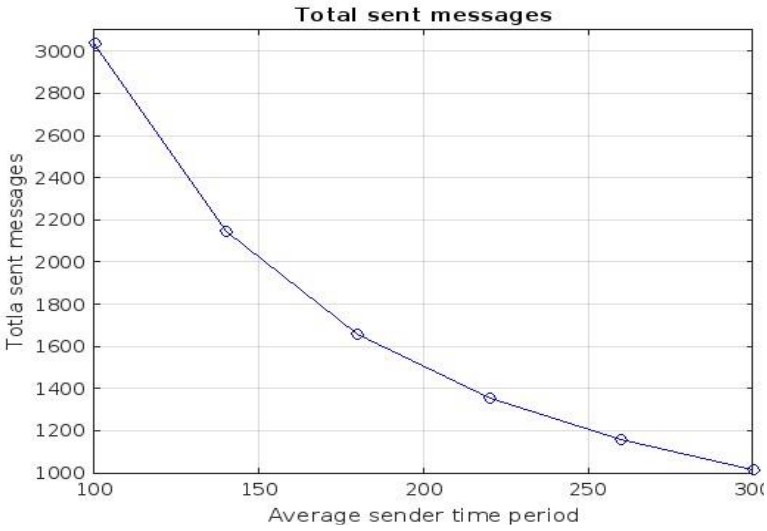



Figure 2: Total sent messages (size = 3)

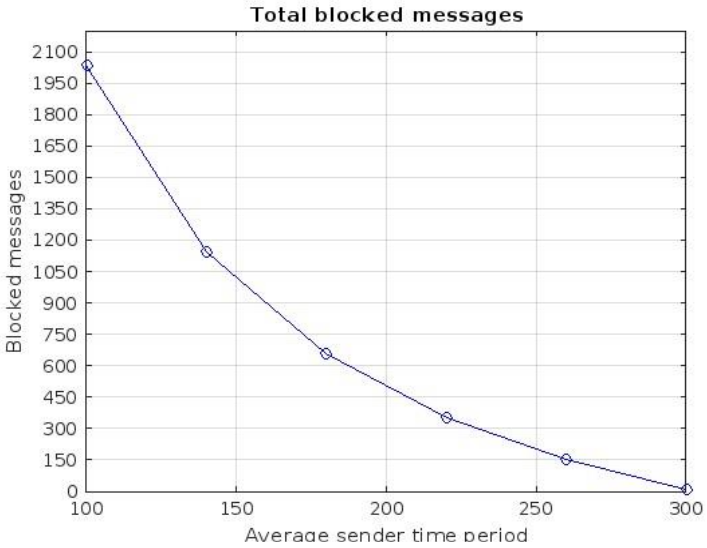



Figure 3: Total blocked messages (size = 3)

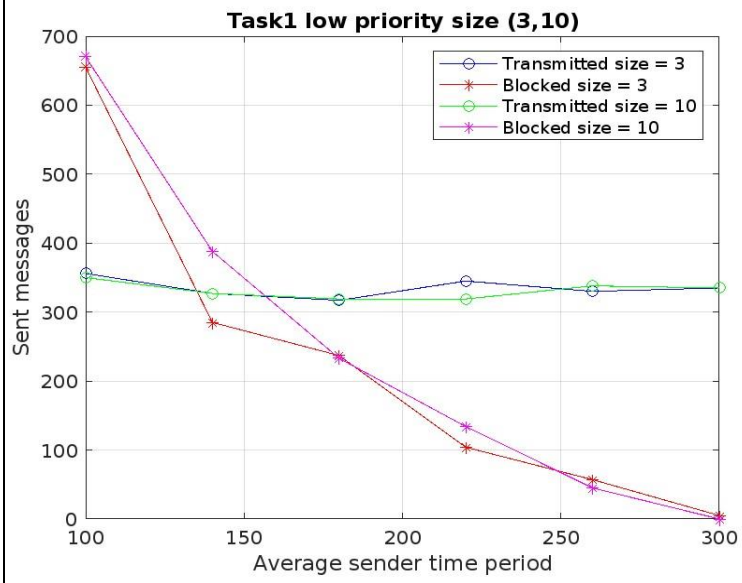



Figure 4: Task 1 low priority (size 3&10)

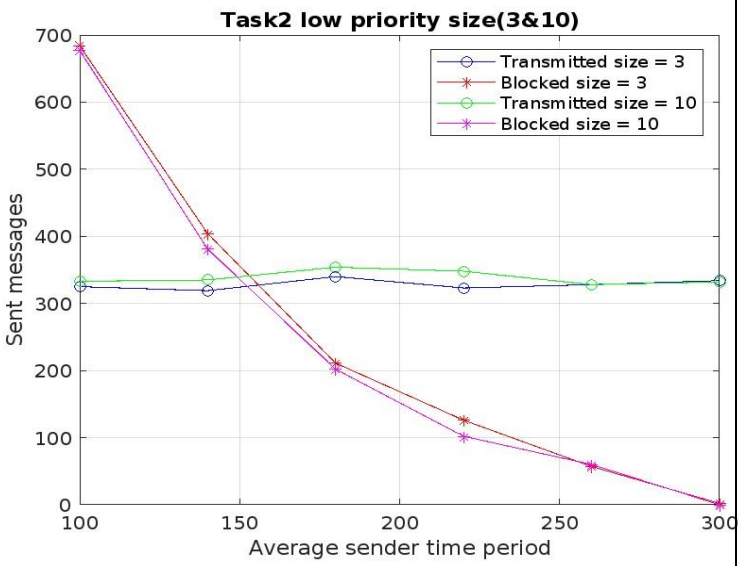



Figure 5: Task 2 low priority (size 3&10)

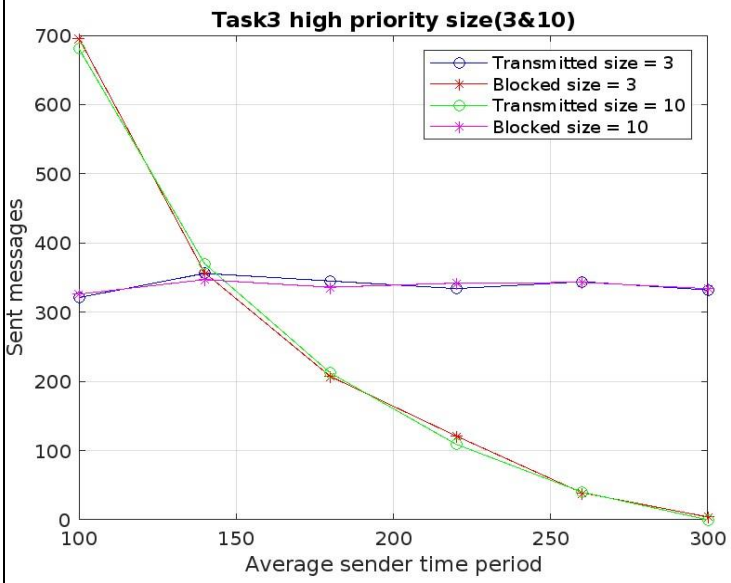



Figure 6: Task 3 high priority (size 3&10)

Figures 4, 5, and 6 illustrate the distinctions between successful and blocked messages across various boundaries with queue sizes of 3 and 10.

```
The total number of successfully sent messages is : 1001
The total number of blocked  messages is : 9
The total number of transmitted message of task 1 is : 333
The total number of blocked message of task 1 is : 4
The total number of transmitted message of task 2 is : 335
The total number of blocked message of task 2 is : 4
The total number of transmitted message of task 3 is : 333
The total number of blocked message of task 3 is : 1
The total number of receivers is : 1000
The Lower bound is   : 200
The upper bound is  : 400
The average sender time period of task 1 is : 301
The average sender time period of task 2 is : 300
The average sender time period of task 3 is : 304
Game Over
```

Figure 7: Example results for a queue of size = 3

```
The total number of successfully sent messages is : 1001
The total number of blocked  messages is : 0
The total number of transmitted message of task 1 is : 335
The total number of blocked message of task 1 is : 0
The total number of transmitted message of task 2 is : 332
The total number of blocked message of task 2 is : 0
The total number of transmitted message of task 3 is : 334
The total number of blocked message of task 3 is : 0
The total number of receivers is : 1000
The Lower bound is   : 200
The Upper bound is   : 400
The average sender timer period of task 1 is : 301
The average sender timer period of task 2 is : 303
The average sender timer period of task 3 is : 302
Game Over
```

Figure 8: Example results for a queue of size = 10

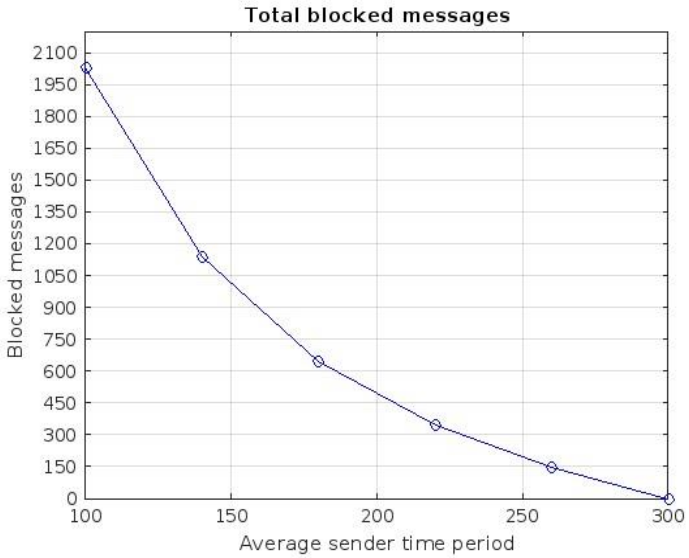



Figure 9: Total blocked messages (size = 10)

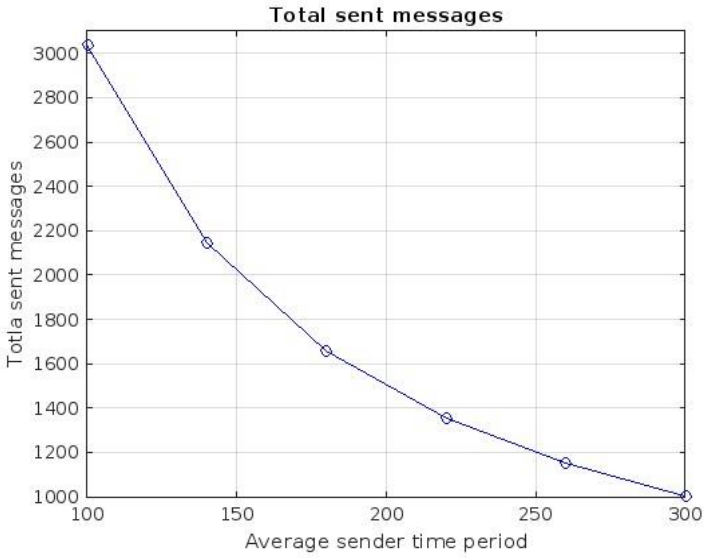



Figure 10: Total sent messages (size = 10)

Figure 7 and 8 demonstrate that the number of blocked messages in a queue of size 10 is smaller than that in a queue of size 3. Furthermore, increasing the period of the sender task results in a decrease in the number of blocked messages, as depicted in figure 9 and 10.