Tey Shi Ying (1003829)

DSC PA2

Task 1 : Lamport's shared priority queue without Ricart and Agrawala's optimization

Do the following steps to run the file.

a. Change the package to main in task1.go
b. In cmd, go run task1.go

Task 2: Lamport's shared priority queue with Ricart and Agrawala's optimization

Do the following steps to run the file.

a. Change package to main2 in task1.go
b. Change the package to main in task2.go
c. In cmd, go run task2.go

Task 3: Centralized server protocol

Do the following steps to run the file.

a. Change package to main2 in task2.go
b. Change the package to main in task3.go
c. In cmd, go run task3.go

Experiments:

I run experiment for the number of nodes corresponding from each integer starting 1 and ending at 10 to get the time between first request and exit critical section vs number of nodes in the protocol. I store the values in a map[int]int where the key is the number of nodes and the value is the time elapsed.

2. Get time elapsed for lamport shared priority queue without Ricart- Agarwal

```
1619262866-9TIME ELAPSED: time elapsed for the experiment is : 118STORE_TIME_LIST: map[1:4 2:8 3:10 4:21 5:29 6:48 7:56 8:77 9:86 10:118]
```

Do the following steps to get the above output.

a. Change package to main2 in task3.go
b. Change the package to main in task1experiment.go
c. In cmd, go run task1experiment.go

3. Get time elapsed for lamport shared priority queue with Ricart- Agarwal

```
1619284481-5TIME ELAPSED: time elapsed for the experiment is : 42NO ELEMENT IN ARRAY, element: 1619284481-5, serverQueue:[]STORE_TIME_LIST: map[1:5 2:9 3:13 4:15 5:20 6:23 7:29 8:35 9:37 10:42]
```

Do the following steps to get the above output.

a. Change package to main2 in task1experiment.go
b. Change the package to main of task2experiment.go to main
c. In cmd, go run task2experiment.go

4. Get time elapsed for centralised server.

```
what is in serverQueue?[]TIME ELAPSED: time elapsed for the experiment is : 19STORE_TIME_LIST: map[1:2 2:4 3:6 4:8 5:10 6:12 7:14 8:16 9:17 10:19]
```

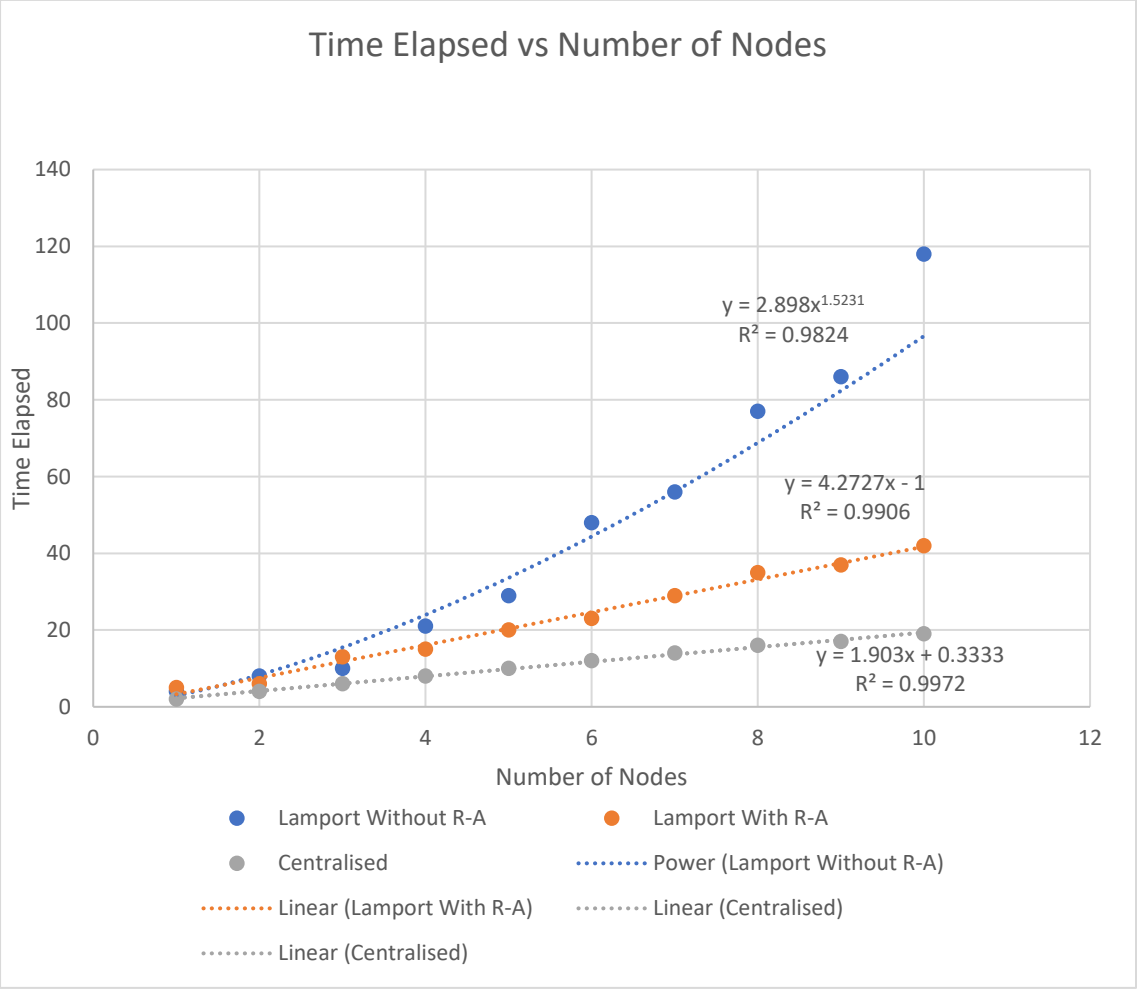
Do the following steps to get the above output.

a. Change package to main2 in task2experiment.go
b. Change the package to main of task3experiment.go to main
c. In cmd, go run task2experiment.go

Here is the table which shows the performance trend vs number of nodes.

	Lamport Without R-A	Lamport With R-A	Centralised
1	4	5	2
2	8	6	4
3	10	13	6
4	21	15	8
5	29	20	10
6	48	23	12
7	56	29	14
8	77	35	16
9	86	37	17
10	118	42	19

Here is the graph which shows the performance trend vs number of nodes.



We can see that in the original for lamport shared priority queue without Ricart-Agarwal, the relationship between the performance trend and number of nodes follows a power relationship. This means that it would be a hard to scale up the nodes as the time elapsed increases at an increasing rate as number of nodes in the protocol increase.

Meanwhile, the lamport shared priority queue with Ricart-Agarwal and centralised protocol has a linear relationship between the performance trend and number of nodes, which means there are lesser problems with scaling up.

As expected the most efficient protocol would be the centralised protocol with the best performance as its protocol has the least number of messages exchanged to acquire and release the lock.