Homework 3 : Loggy

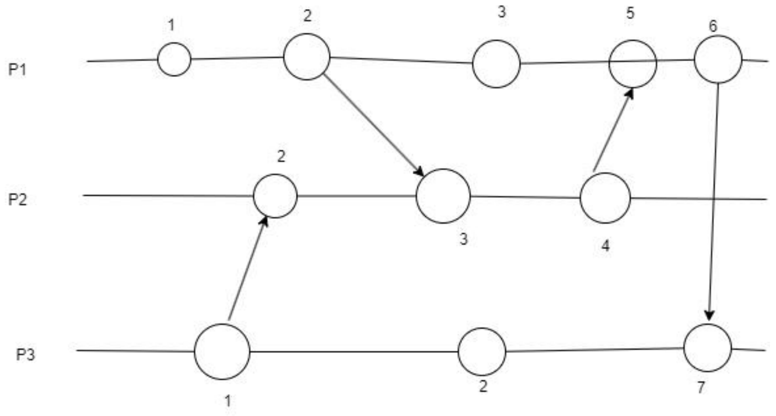
Reethika Ambatipudi,
September 28, 2022.

Introduction:

The motto of the assignment was to handle one of the major issues of the Distributed Systems which is lack of a global time.

Main Problems and Solutions:

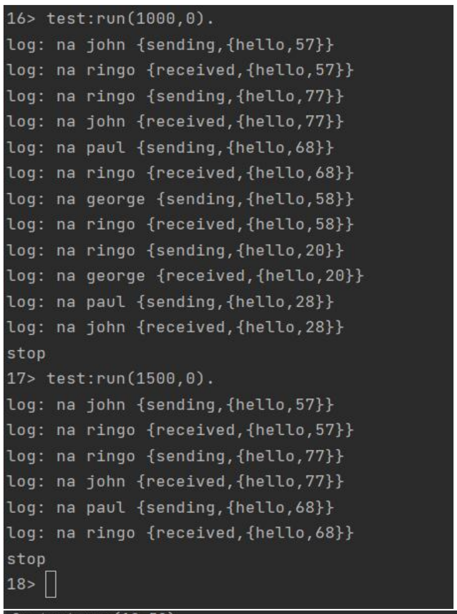
Understanding how a Lamport timestamp works was initial task. It was interesting. Initially counters of all the processes are set to 0. For every event in that particular process, the counter is incremented by 1. But the incrementation process slightly varies while receiving a message. The counter value of the sender at that particular event is sent in the message. It has to take the maximum value of the counter at it's process and the sender's process and then increment that.



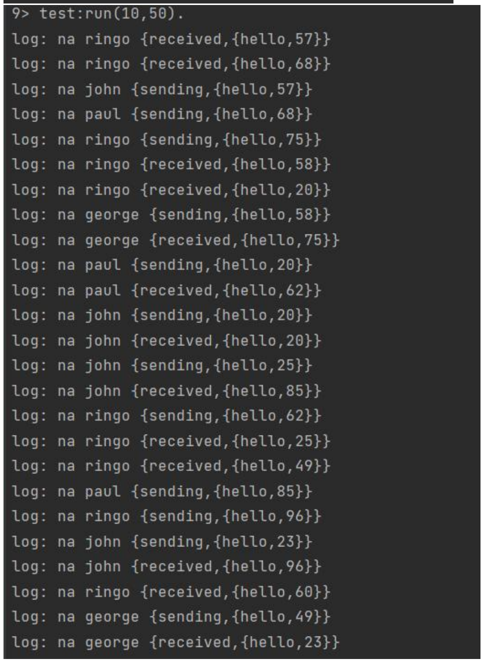
Lamport's timestamp process.

Evaluation:

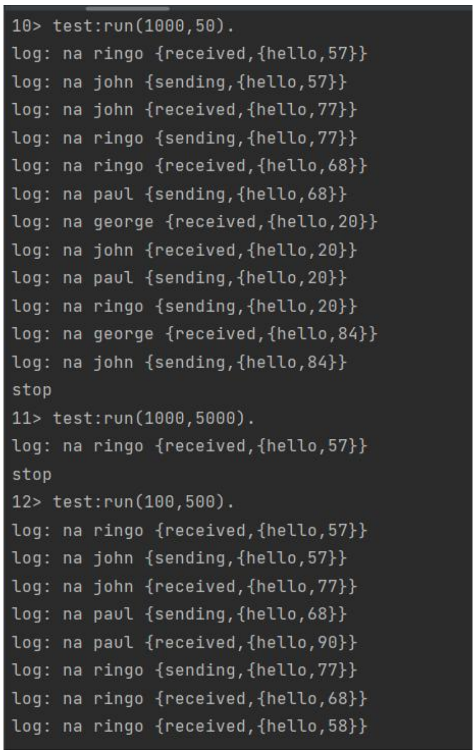
I tried to initiate 4 processes and find the order of the messages before even implementing the lamport time stamp. The causality was not maintained initially. Received messages were printed first and send messages later. Then with experimenting with the Sleep and Jitter values, I observed that when the jitter value was 0, causality was maintained and the send messages were printed before received messages.



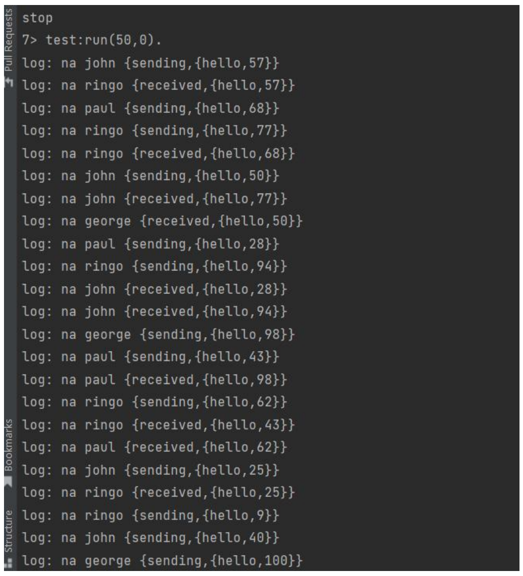
```
16> test:run(1000,0).
log: na john {sending,{hello,57}}
log: na ringo {received,{hello,57}}
log: na ringo {sending,{hello,77}}
log: na john {received,{hello,77}}
log: na paul {sending,{hello,68}}
log: na ringo {received,{hello,68}}
log: na george {sending,{hello,58}}
log: na ringo {received,{hello,58}}
log: na ringo {sending,{hello,20}}
log: na george {received,{hello,20}}
log: na paul {sending,{hello,28}}
log: na john {received,{hello,28}}
stop
17> test:run(1500,0).
log: na john {sending,{hello,57}}
log: na ringo {received,{hello,57}}
log: na ringo {sending,{hello,77}}
log: na john {received,{hello,77}}
log: na paul {sending,{hello,68}}
log: na ringo {received,{hello,68}}
stop
18> 
```



```
9> test:run(10,50).
log: na ringo {received,{hello,57}}
log: na ringo {received,{hello,68}}
log: na john {sending,{hello,57}}
log: na paul {sending,{hello,68}}
log: na ringo {sending,{hello,75}}
log: na ringo {received,{hello,58}}
log: na ringo {received,{hello,20}}
log: na george {sending,{hello,58}}
log: na george {received,{hello,75}}
log: na paul {sending,{hello,20}}
log: na paul {received,{hello,62}}
log: na john {sending,{hello,20}}
log: na john {received,{hello,20}}
log: na john {sending,{hello,25}}
log: na john {received,{hello,85}}
log: na ringo {sending,{hello,62}}
log: na ringo {received,{hello,25}}
log: na ringo {received,{hello,49}}
log: na paul {sending,{hello,85}}
log: na ringo {sending,{hello,96}}
log: na john {sending,{hello,23}}
log: na john {received,{hello,96}}
log: na ringo {received,{hello,60}}
log: na george {sending,{hello,49}}
log: na george {received,{hello,23}}
```

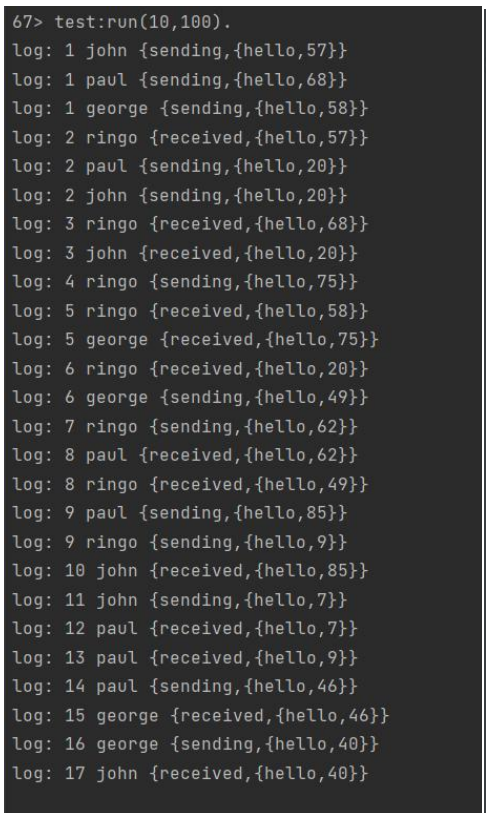


```
10> test:run(1000,50).
log: na ringo {received,{hello,57}}
log: na john {sending,{hello,57}}
log: na john {received,{hello,77}}
log: na ringo {sending,{hello,77}}
log: na ringo {received,{hello,68}}
log: na paul {sending,{hello,68}}
log: na george {received,{hello,20}}
log: na john {received,{hello,20}}
log: na paul {sending,{hello,20}}
log: na ringo {sending,{hello,20}}
log: na george {received,{hello,84}}
log: na john {sending,{hello,84}}
stop
11> test:run(1000,5000).
log: na ringo {received,{hello,57}}
stop
12> test:run(100,500).
log: na ringo {received,{hello,57}}
log: na john {sending,{hello,57}}
log: na john {received,{hello,77}}
log: na paul {sending,{hello,68}}
log: na paul {received,{hello,90}}
log: na ringo {sending,{hello,77}}
log: na ringo {received,{hello,68}}
log: na ringo {received,{hello,58}}
```

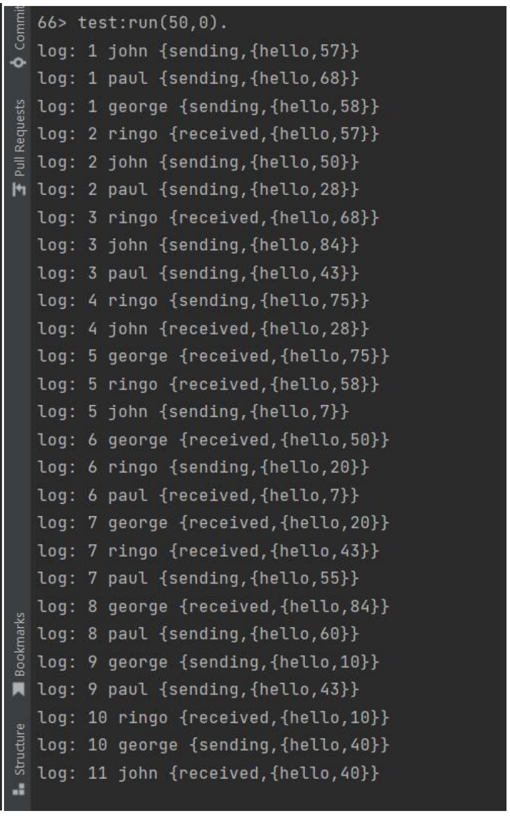


```
stop
7> test:run(50,0).
log: na john {sending,{hello,57}}
log: na ringo {received,{hello,57}}
log: na paul {sending,{hello,68}}
log: na ringo {sending,{hello,77}}
log: na ringo {received,{hello,68}}
log: na john {sending,{hello,50}}
log: na john {received,{hello,77}}
log: na george {received,{hello,50}}
log: na paul {sending,{hello,28}}
log: na ringo {sending,{hello,94}}
log: na john {received,{hello,28}}
log: na john {received,{hello,94}}
log: na george {sending,{hello,98}}
log: na paul {sending,{hello,43}}
log: na paul {received,{hello,98}}
log: na ringo {sending,{hello,62}}
log: na ringo {received,{hello,43}}
log: na paul {received,{hello,62}}
log: na john {sending,{hello,25}}
log: na ringo {received,{hello,25}}
log: na ringo {sending,{hello,9}}
log: na john {sending,{hello,40}}
log: na george {sending,{hello,100}}
```

After implementing the Lamport’s clock, the messages were in proper order and were also assigned the counters with proper order.Even here I have observed that Sleep value should be set larger than Jitter value to make sure it maintains the order of causality.



```
67> test:run(10,100).
log: 1 john {sending,{hello,57}}
log: 1 paul {sending,{hello,68}}
log: 1 george {sending,{hello,58}}
log: 2 ringo {received,{hello,57}}
log: 2 paul {sending,{hello,20}}
log: 2 john {sending,{hello,20}}
log: 3 ringo {received,{hello,68}}
log: 3 john {received,{hello,20}}
log: 4 ringo {sending,{hello,75}}
log: 5 ringo {received,{hello,58}}
log: 5 george {received,{hello,75}}
log: 6 ringo {received,{hello,20}}
log: 6 george {sending,{hello,49}}
log: 7 ringo {sending,{hello,62}}
log: 8 paul {received,{hello,62}}
log: 8 ringo {received,{hello,49}}
log: 9 paul {sending,{hello,85}}
log: 9 ringo {sending,{hello,9}}
log: 10 john {received,{hello,85}}
log: 11 john {sending,{hello,7}}
log: 12 paul {received,{hello,7}}
log: 13 paul {received,{hello,9}}
log: 14 paul {sending,{hello,46}}
log: 15 george {received,{hello,46}}
log: 16 george {sending,{hello,40}}
log: 17 john {received,{hello,40}}
```



```
66> test:run(50,0).
log: 1 john {sending,{hello,57}}
log: 1 paul {sending,{hello,68}}
log: 1 george {sending,{hello,58}}
log: 2 ringo {received,{hello,57}}
log: 2 john {sending,{hello,50}}
log: 2 paul {sending,{hello,28}}
log: 3 ringo {received,{hello,68}}
log: 3 john {sending,{hello,84}}
log: 3 paul {sending,{hello,43}}
log: 4 ringo {sending,{hello,75}}
log: 4 john {received,{hello,28}}
log: 5 george {received,{hello,75}}
log: 5 ringo {received,{hello,58}}
log: 5 john {sending,{hello,7}}
log: 6 george {received,{hello,50}}
log: 6 ringo {sending,{hello,20}}
log: 6 paul {received,{hello,7}}
log: 7 george {received,{hello,20}}
log: 7 ringo {received,{hello,43}}
log: 7 paul {sending,{hello,55}}
log: 8 george {received,{hello,84}}
log: 8 paul {sending,{hello,60}}
log: 9 george {sending,{hello,10}}
log: 9 paul {sending,{hello,43}}
log: 10 ringo {received,{hello,10}}
log: 10 george {sending,{hello,40}}
log: 11 john {received,{hello,40}}
```

Conclusion:

Lamport’s timestamp is useful but it has it’s own drawbacks. Concurrent processes and Causal processes are not really differentiated here. Introduction of Vector’s clock might be helpful.