CES-27 Distributed Processing
1th Activity
Prof Hirata and Prof Juliana
Carlos Matheus Barros da Silva August 2019

Task 1

It was implemented a Lamport's *Scalar Logic Clock* simulation. The implementation was made in Go and it can be seen on the Repository[1].

Suggested Test Case

As It was suggested, it was conducted a test with 3 terminal windows, on each one was opened one task of the program as shown in the Code 1.

The test was made according to the model represented on Figure 1. The results can be seen on the terminal windows shown from Figure 2 to Figure 4.

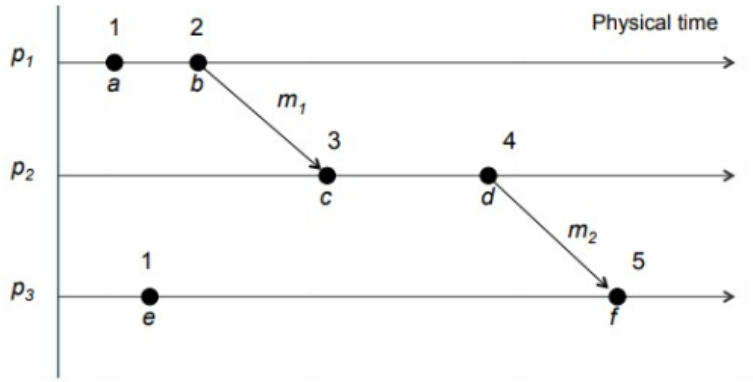

Figure 1: Model representing the execution of Task 1 example test case.

```
Terminal 1: go run Process.go 1 :10004 :10003 :10002
Terminal 2: go run Process.go 2 :10004 :10003 :10002
Terminal 3: go run Process.go 3 :10004 :10003 :10002
```

Code 1: Code that was run on each of the 3 terminal window on the execution of Task 1 example test case.

As expected, the logical clock on each process was updated to the bigger one, if the incomming message logical clock time was greater than the actual time on that process, and then it was also increased by one. This logic was applied and because of it the simulation on the terminals matched the model represented on Figure 1.

Built Test Case

It was built a test case with 4 terminal windows, on each one was opened one task of the program as shown in the Code 2.

The test was made according to the model represented on Figure 1. The results can be seen on the terminal windows shown from Figure 6 to Figure 9.

[1] https://github.com/CarlosMatheus/CES-27/tree/master/lab01

Figure 2: Window 1 after execution of Task 1 example test case.

Figure 3: Window 2 after execution of Task 1 example test case.

```
Terminal 1: go run Process.go 1 :10004 :10003 :10002 :10001
Terminal 2: go run Process.go 2 :10004 :10003 :10002 :10001
Terminal 3: go run Process.go 3 :10004 :10003 :10002 :10001
Terminal 4: go run Process.go 4 :10004 :10003 :10002 :10001
```

Code 2: Code that was run on each of the 4 terminal window on the execution of Task 1 built test case.

As expected, the logical clock on each process was updated to the bigger one, if the incomming message logical clock time was greater than the actual time on that process, and then it was also increased by one. This logic was applied and because of it the simulation on the terminals matched the model represented on Figure 1.

Figure 4: Window 3 after execution of Task 1 example test case.

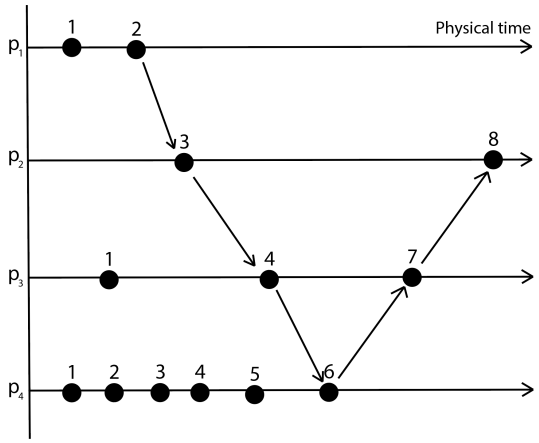

Figure 5: Model representing the execution of Task 1 built test case.

Task 2

Suggested Test Case

As It was suggested, it was conducted a test with 3 terminal windows, on each one was opened one task of the program as shown in the Code 3.

The test was made according to the model represented on Figure 10. The results can be seen on the terminal windows shown from Figure 11 to Figure 13.

```
Terminal 1: go run Process.go 1 :10004 :10003 :10002
Terminal 2: go run Process.go 2 :10004 :10003 :10002
Terminal 3: go run Process.go 3 :10004 :10003 :10002
```

Code 3: Code that was run on each of the 3 terminal window on the execution of Task 2 example test case.

As expected, the logical clock on each process was updated to the bigger one, if the incomming message logical clock time was greater than the actual time on that process, and then

Figure 6: Window 1 after execution of Task 1 built test case.

Figure 7: Window 2 after execution of Task 1 built test case.

it was also increased by one. This logic was applied and because of it the simulation on the terminals matched the model represented on Figure 10.

Figure 8: Window 3 after execution of Task 1 built test case.

Figure 9: Window 4 after execution of Task 1 built test case.

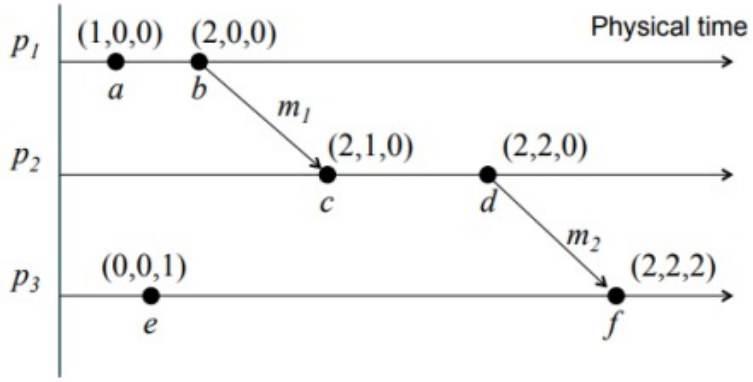

Figure 10: Model representing the execution of Task 1 example test case.

Figure 11: Window 1 after execution of Task 2 example test case.

Figure 12: Window 2 after execution of Task 2 example test case.

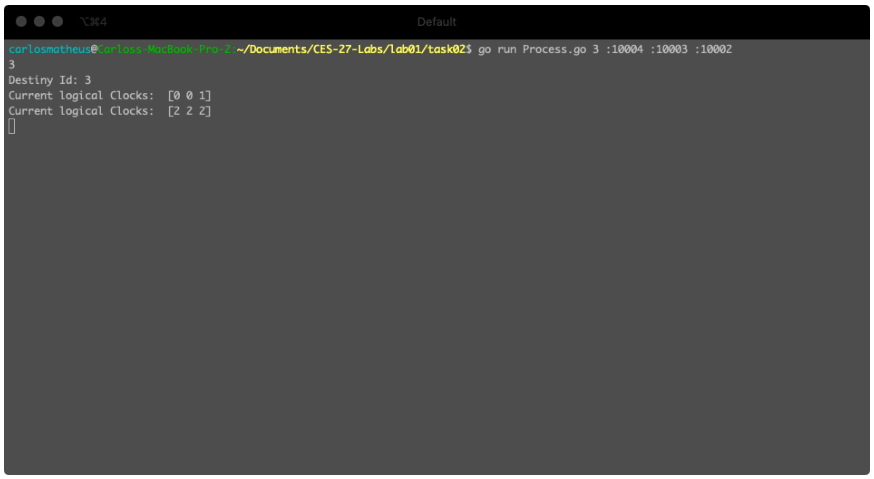

Figure 13: Window 3 after execution of Task 2 example test case.