

Seminar Report: Muty

March 20, 2019

Carles Tornel
Jesus Alfaro
Ricard Abril

1 Introduction

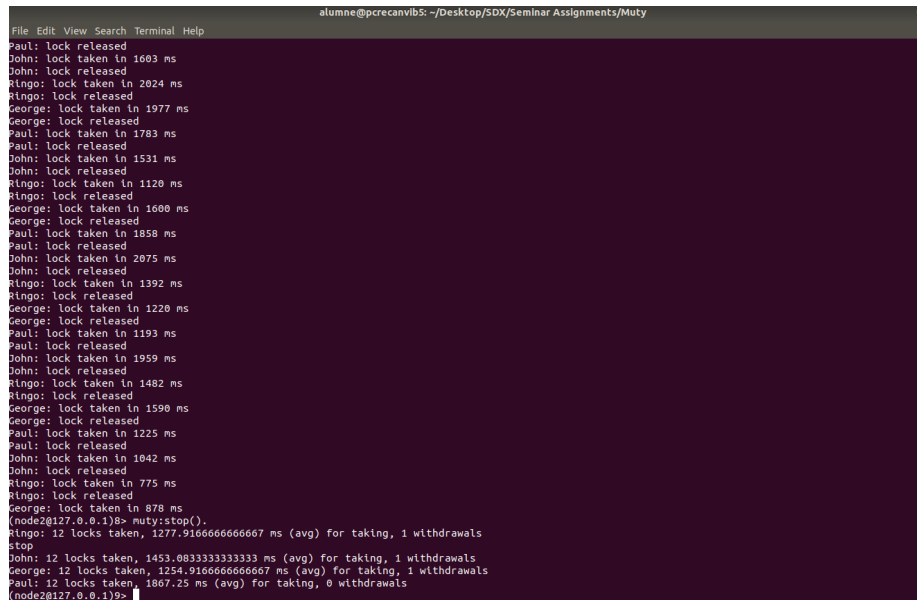
2 Experiments

Some Testing

1. Make tests with different Sleep and Work parameters to analyze how this lock implementation responds to different contention degrees.

Test 1: Sleep time inferior a work time

Si el temps de sleep és considerablement més petit que el temps de work, la probabilitat de que diferents workers demanin accés a la regió crítica serà més alta, quan això passa, aquets workers esperaran un missatge d' OK de la resta, però aquets també es estaran en la mateixa situació, de forma que ens trobarem davant d'un deadlock. Encara que al estar durant 8 segons en aquesta situació, el programa allibera els lock, per tal de poder seguir executant-se. Tal i com podem veure en la següent imatge:



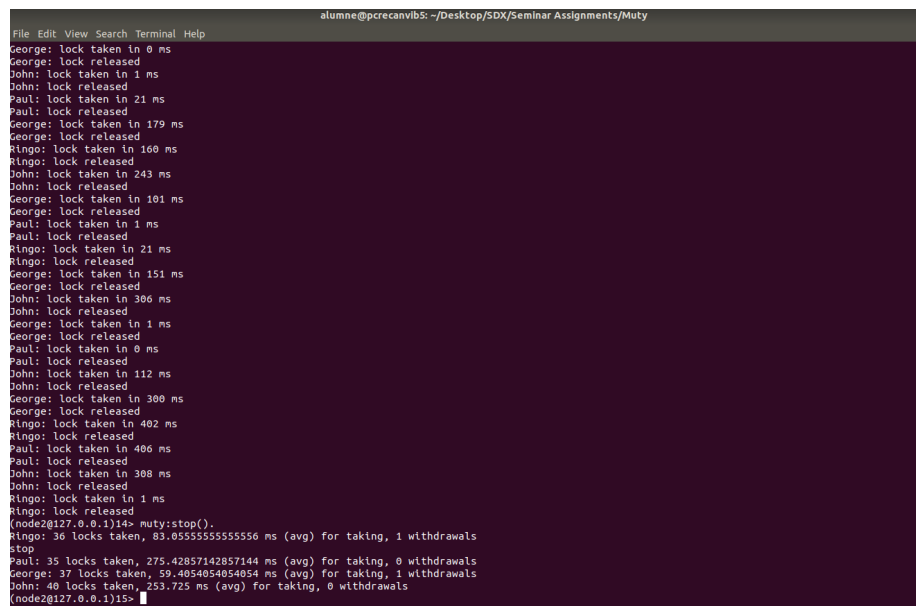
```
alumn@pcrecanvib5: ~/Desktop/SDX/Seminar Assignments/Muty
Paul: lock released
John: lock taken in 1603 ms
John: lock released
Ringo: lock taken in 2624 ms
Ringo: lock released
George: lock taken in 1977 ms
George: lock released
Paul: lock taken in 1783 ms
Paul: lock released
John: lock taken in 1531 ms
John: lock released
Ringo: lock taken in 1120 ms
Ringo: lock released
George: lock taken in 1600 ms
George: lock released
Paul: lock taken in 1858 ms
Paul: lock released
John: lock taken in 2075 ms
John: lock released
Ringo: lock taken in 1392 ms
Ringo: lock released
George: lock taken in 1220 ms
George: lock released
Paul: lock taken in 1193 ms
Paul: lock released
John: lock taken in 1959 ms
John: lock released
Ringo: lock taken in 1482 ms
Ringo: lock released
George: lock taken in 1590 ms
George: lock released
Paul: lock taken in 1225 ms
Paul: lock released
John: lock taken in 1042 ms
John: lock released
Ringo: lock taken in 775 ms
Ringo: lock released
George: lock taken in 878 ms
(node2@127.0.0.1)> muty:stop().
Ringo: 12 locks taken, 1277.916666666667 ms (avg) for taking, 1 withdrawals
stop
John: 12 locks taken, 1453.083333333333 ms (avg) for taking, 1 withdrawals
George: 12 locks taken, 1254.916666666667 ms (avg) for taking, 1 withdrawals
Paul: 12 locks taken, 1867.25 ms (avg) for taking, 0 withdrawals
(node2@127.0.0.1)>
```

També ens podem trobar amb la situació de que el temps de work, sigui més gran que el withdrawal time, per tant els workers a la espera superarien aquet temps. Tal i com podem observar:

```
alunne@pcrecanvib: ~/Desktop/SDX/Seminar Assignments/Muty
File Edit View Search Terminal Help
(node2@127.0.0.1)9>
(node2@127.0.0.1)9>
(node2@127.0.0.1)9>
(node2@127.0.0.1)9>
(node2@127.0.0.1)9> muty:start(lock
lock1 lock2 lock3
(node2@127.0.0.1)9> muty:start(lock1:
module_info/0 module_info/1 start/1
(node2@127.0.0.1)9> muty:start(lock1:
module_info/0 module_info/1 start/1
(node2@127.0.0.1)9> muty:start(lock1,6000,8500).
ok
John: lock taken in 0 ms
John: lock released
Ringo: lock taken in 3427 ms
Ringo: lock released
George: lock taken in 4790 ms
Paul: giving up
John: giving up
George: lock released
Ringo: lock taken in 6499 ms
Ringo: lock released
Paul: lock taken in 1395 ms
Paul: lock released
John: lock taken in 7447 ms
John: lock released
George: lock taken in 7864 ms
Ringo: giving up
George: lock released
Paul: lock taken in 3718 ms
Paul: lock released
Ringo: lock taken in 6161 ms
Ringo: lock released
John: lock taken in 7936 ms
George: giving up
John: lock released
Paul: lock taken in 7960 ms
George: giving up
(node2@127.0.0.1)10> muty:stop().
Ringo: 3 locks taken, 5362.333333333333 ms (avg) for taking, 1 withdrawals
stop
Paul: 2 locks taken, 2556.5 ms (avg) for taking, 1 withdrawals
John: 3 locks taken, 5127.666666666667 ms (avg) for taking, 1 withdrawals
George: 2 locks taken, 6327.0 ms (avg) for taking, 2 withdrawals
(node2@127.0.0.1)11>
```

Test 2: Sleep time superior a work time

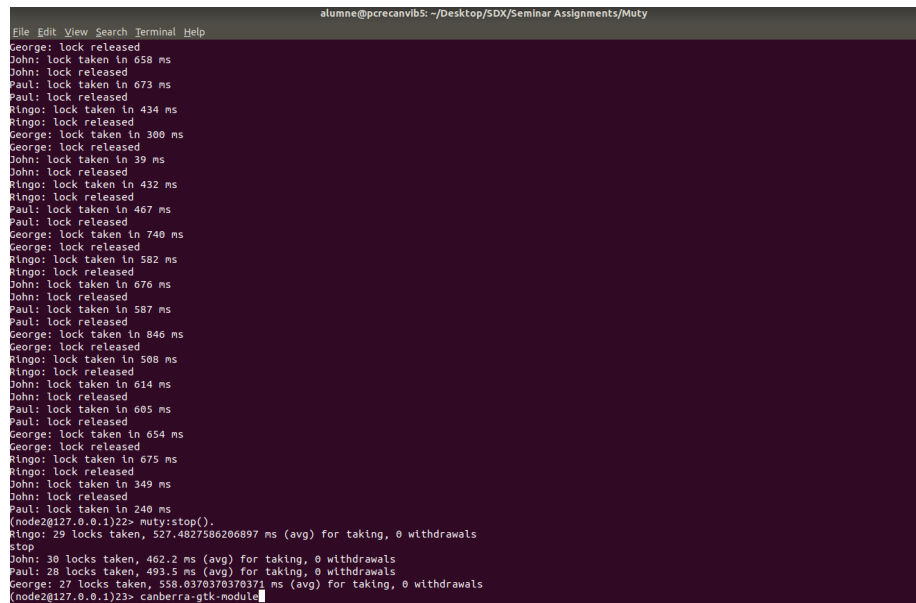
Si el temps de sleep, és més gran que el temps de work, estarem provocant, que les peticions d'accés a la regió crítica dels diferents workers, estigui més espatllada, i per tant la possibilitat de trobar-nos en una situació de deadlock serà més baixa. Encara que no es una solució definitiva, ja que encara que essent menys probable també ens podem trobar amb una situació de deadlock, tal i com podem observar en la següent imatge:



```
alumnne@pcrecanvibS: ~/Desktop/SDX/Seminar Assignments/Muty
File Edit View Search Terminal Help
George: lock taken in 0 ms
George: lock released
John: lock taken in 1 ms
John: lock released
Paul: lock taken in 21 ms
Paul: lock released
George: lock taken in 179 ms
George: lock released
Ringo: lock taken in 160 ms
Ringo: lock released
John: lock taken in 243 ms
John: lock released
George: lock taken in 101 ms
George: lock released
Paul: lock taken in 1 ms
Paul: lock released
Ringo: lock taken in 21 ms
Ringo: lock released
George: lock taken in 151 ms
George: lock released
John: lock taken in 300 ms
John: lock released
George: lock taken in 1 ms
George: lock released
Paul: lock taken in 0 ms
Paul: lock released
John: lock taken in 112 ms
John: lock released
George: lock taken in 300 ms
George: lock released
Ringo: lock taken in 402 ms
Ringo: lock released
Paul: lock taken in 406 ms
Paul: lock released
John: lock taken in 308 ms
John: lock released
Ringo: lock taken in 1 ms
Ringo: lock released
(node2@127.0.0.1)14> muty:stop().
Ringo: 36 locks taken, 83.05555555555556 ms (avg) for taking, 1 withdrawals
stop
Paul: 35 locks taken, 275.42857142857144 ms (avg) for taking, 0 withdrawals
George: 37 locks taken, 59.4054054054054 ms (avg) for taking, 1 withdrawals
John: 40 locks taken, 253.725 ms (avg) for taking, 0 withdrawals
(node2@127.0.0.1)15>
```

Test 3: Sleep time i work time iguals

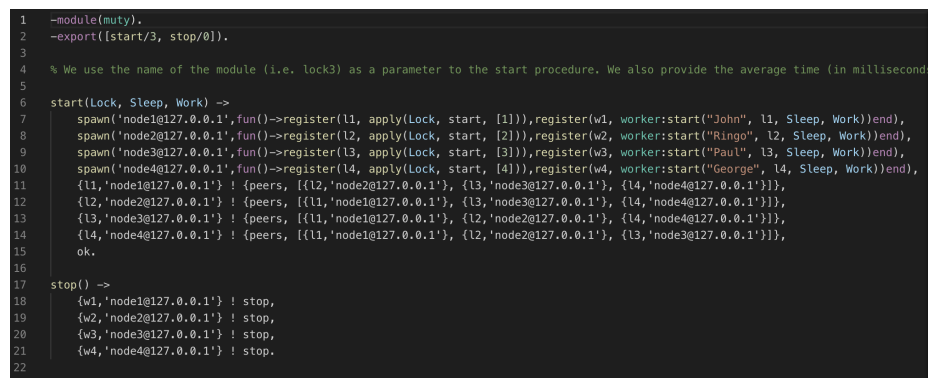
En aquesta situació, a la llarga també ens podriem trobar amb dead-locks, encara que seran improbables com quan el sleep time es menor al work time. Ho podem observar en la següent imatge:



```
alunne@pcrecanvib5: ~/Desktop/SDX/Seminar Assignments/Muty
File Edit View Search Terminal Help
George: lock released
John: lock taken in 658 ms
John: lock released
Paul: lock taken in 673 ms
Paul: lock released
Ringo: lock taken in 434 ms
Ringo: lock released
George: lock taken in 300 ms
George: lock released
John: lock taken in 39 ms
John: lock released
Ringo: lock taken in 432 ms
Ringo: lock released
Paul: lock taken in 467 ms
Paul: lock released
George: lock taken in 740 ms
George: lock released
Ringo: lock taken in 582 ms
Ringo: lock released
John: lock taken in 676 ms
John: lock released
Paul: lock taken in 587 ms
Paul: lock released
George: lock taken in 846 ms
George: lock released
Ringo: lock taken in 588 ms
Ringo: lock released
John: lock taken in 614 ms
John: lock released
Paul: lock taken in 605 ms
Paul: lock released
George: lock taken in 654 ms
George: lock released
Ringo: lock taken in 675 ms
Ringo: lock released
John: lock taken in 349 ms
John: lock released
Paul: lock taken in 240 ms
(node2@127.0.0.1)22> muty:stop().
Ringo: 29 locks taken, 527.4827586206897 ms (avg) for taking, 0 withdrawals
stop
John: 30 locks taken, 462.2 ms (avg) for taking, 0 withdrawals
Paul: 28 locks taken, 493.5 ms (avg) for taking, 0 withdrawals
George: 27 locks taken, 558.0370370370371 ms (avg) for taking, 0 withdrawals
(node2@127.0.0.1)23> canberra-gtk-module
```

2. Adapt the muty module to create each worker-lock pair in a different Erlang instance (that is, john and l1 should run in a node, ringo and l2 in another, and so on). Remember how processes are created remotely, how names registered in remote nodes are referred, and how Erlang runtime should be started to run distributed programs.

En la imatge següent podem veure les modificacions en el fitxer muty, per a que inici els workers en diferents instàncies de erlang



```
1 -module(muty).
2 -export([start/3, stop/0]).
3
4 % We use the name of the module (i.e. lock3) as a parameter to the start procedure. We also provide the average time (in millisecond
5
6 start(Lock, Sleep, Work) ->
7   spawn('node1@127.0.0.1', fun() -> register(l1, apply(Lock, start, [1])), register(w1, worker:start("John", l1, Sleep, Work))end),
8   spawn('node2@127.0.0.1', fun() -> register(l2, apply(Lock, start, [2])), register(w2, worker:start("Ringo", l2, Sleep, Work))end),
9   spawn('node3@127.0.0.1', fun() -> register(l3, apply(Lock, start, [3])), register(w3, worker:start("Paul", l3, Sleep, Work))end),
10  spawn('node4@127.0.0.1', fun() -> register(l4, apply(Lock, start, [4])), register(w4, worker:start("George", l4, Sleep, Work))end),
11  {l1, 'node1@127.0.0.1'} ! {peers, [{l2, 'node2@127.0.0.1'}, {l3, 'node3@127.0.0.1'}, {l4, 'node4@127.0.0.1'}]},
12  {l2, 'node2@127.0.0.1'} ! {peers, [{l1, 'node1@127.0.0.1'}, {l3, 'node3@127.0.0.1'}, {l4, 'node4@127.0.0.1'}]},
13  {l3, 'node3@127.0.0.1'} ! {peers, [{l1, 'node1@127.0.0.1'}, {l2, 'node2@127.0.0.1'}, {l4, 'node4@127.0.0.1'}]},
14  {l4, 'node4@127.0.0.1'} ! {peers, [{l1, 'node1@127.0.0.1'}, {l2, 'node2@127.0.0.1'}, {l3, 'node3@127.0.0.1'}]},
15  ok.
16
17 stop() ->
18  {w1, 'node1@127.0.0.1'} ! stop,
19  {w2, 'node2@127.0.0.1'} ! stop,
20  {w3, 'node3@127.0.0.1'} ! stop,
21  {w4, 'node4@127.0.0.1'} ! stop.
22
```

Tal i com podem veure en la imatge, els diferents workers s'executen en instàncies de erlang diferents.

The screenshot displays three terminal windows on a Kali Linux desktop. The leftmost window shows a list of lock times for various users, including Ringo, John, Paul, and George. The middle window shows a netcat listener on 0.0.0.0:4444, which has received a connection from 10.10.10.10. The rightmost window shows a netcat client connecting to 10.10.10.10:4444 and sending a 'c' command.

```

tor@tor-virtual-machine: ~/Escritorio/SDX/Seminar Assignments/Muty
Archivo Editor Ver Buscar Terminal Ayuda
Ringo: lock taken in 187 ms
Ringo: lock released
John: lock taken in 300 ms
John: lock released
Paul: lock taken in 343 ms
Paul: lock released
George: lock taken in 340 ms
George: lock released
Ringo: lock taken in 369 ms
Ringo: lock released
John: lock taken in 381 ms
John: lock released
Paul: lock taken in 296 ms
Paul: lock released
George: lock taken in 250 ms
George: lock released
Ringo: lock taken in 82 ms
Ringo: lock released
John: lock taken in 124 ms
John: lock released
Paul: lock taken in 232 ms
Paul: lock released
George: lock taken in 418 ms
(node@127.0.0.1)>

tor@tor-virtual-machine: ~/Escritorio/SDX/Seminar Assignm...
Archivo Editor Ver Buscar Terminal Ayuda
tor@tor-virtual-machine: ~/Escritorio/SDX/Seminar Assignm...
./mutys erl -name node@127.0.0.1 -setcookie secret
Erlang OTP 20 [erts-9.3.3] [source] [64-bit] [smp:4]
i4:10] [async-threads:10] [kernel-poll:false]

Eshell V9.3.3.3 (abort with ^G)
(node@127.0.0.1)> c(gul).
(ok,gul)
(node@127.0.0.1)> c(lock1).
(ok,lock1)
(node@127.0.0.1)> c(nuty).
(ok,nuty)
(node@127.0.0.1)> c(worker).
(ok,worker)
(node@127.0.0.1)> c(nuty).
(ok,nuty)
(node@127.0.0.1)>

tor@tor-virtual-machine: ~/Escritorio/SDX/Seminar Assignm...
Archivo Editor Ver Buscar Terminal Ayuda
tor@tor-virtual-machine: ~/Escritorio/SDX/Seminar Assignm...
./mutys erl -name node@127.0.0.1 -setcookie secret
Erlang OTP 20 [erts-9.3.3] [source] [64-bit] [smp:4]
i4:10] [async-threads:10] [kernel-poll:false]

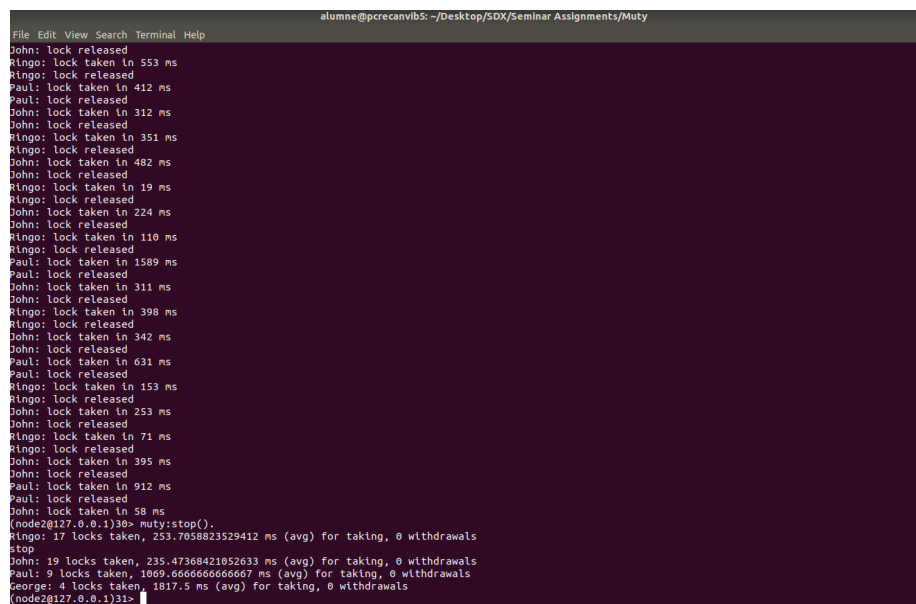
Eshell V9.3.3.3 (abort with ^G)
(node@127.0.0.1)> c(gul).
(ok,gul)
(node@127.0.0.1)> c(worker).
(ok,worker)
(node@127.0.0.1)> c(lock).
(error,non_existing)
(node@127.0.0.1)> c(lock1).
(ok,lock1)
(node@127.0.0.1)> c(nuty).
(ok,nuty)
(node@127.0.0.1)> c(nuty).
(ok,nuty)
(node@127.0.0.1)>

```

Resolving deadlock

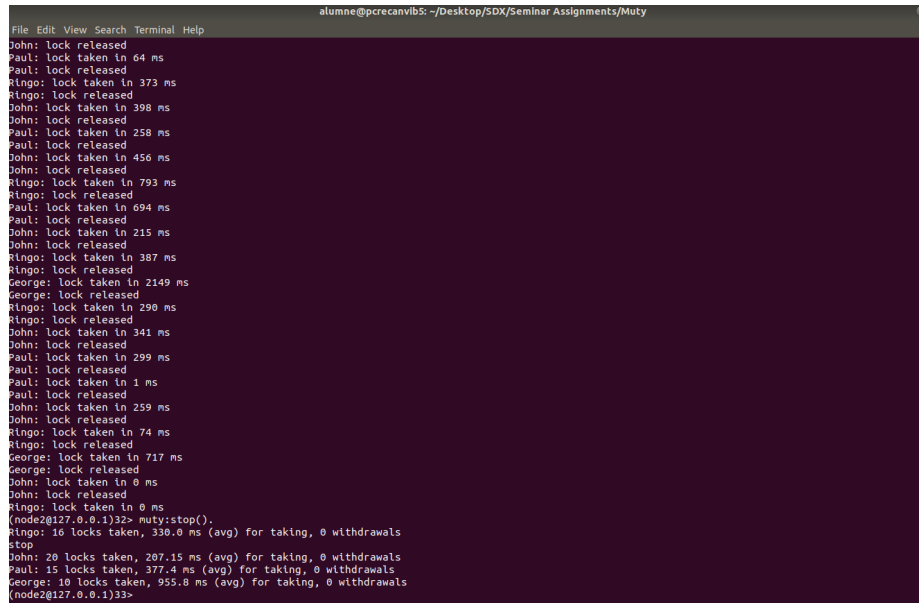
1. Repeat the previous tests to compare the behavior of this lock with respect to the previous one.

Test 1: Sleep time inferior a work time Per evitar els deadlocks, ara en cas de que diversos workers vulguin accedir a la regió crítica del codi al mateix temps, el worker amb el ID més petit serà el que rebrà el missatge OK i podrà accedir. Amb aquest sistema evitarem els deadlocks, encara que el repartiment de feina, no serà massa equitatiu, ja que els workers amb ID més petit sempre tindran més prioritat, i per tant en cas de "empat", sempre acabaran fent la feina. Això ho podem observar en la següent imatge:



```
File Edit View Search Terminal Help
alumni@pcrecanvib5: ~/Desktop/SDX/Seminar Assignments/Muty
John: lock released
Ringo: lock taken in 553 ms
Ringo: lock released
Paul: lock taken in 412 ms
Paul: lock released
John: lock taken in 312 ms
John: lock released
Ringo: lock taken in 351 ms
Ringo: lock released
John: lock taken in 402 ms
John: lock released
Ringo: lock taken in 19 ms
Ringo: lock released
John: lock taken in 224 ms
John: lock released
Ringo: lock taken in 110 ms
Ringo: lock released
Paul: lock taken in 1589 ms
Paul: lock released
John: lock taken in 311 ms
John: lock released
Ringo: lock taken in 398 ms
Ringo: lock released
John: lock taken in 342 ms
John: lock released
Paul: lock taken in 631 ms
Paul: lock released
Ringo: lock taken in 153 ms
Ringo: lock released
John: lock taken in 253 ms
John: lock released
Ringo: lock taken in 71 ms
Ringo: lock released
John: lock taken in 395 ms
John: lock released
Paul: lock taken in 912 ms
Paul: lock released
John: lock taken in 58 ms
(node2@127.0.0.1)30> muty:stop().
Ringo: 17 locks taken, 253.7058823529412 ms (avg) for taking, 0 withdrawals
stop
John: 19 locks taken, 235.47368421052633 ms (avg) for taking, 0 withdrawals
Paul: 9 locks taken, 1069.6666666666667 ms (avg) for taking, 0 withdrawals
George: 4 locks taken, 1817.5 ms (avg) for taking, 0 withdrawals
(node2@127.0.0.1)31>
```

Test 2: Sleep time superior a work time En aquesta situació, al tenir un sleep time major al work time, la probabilitat de que dos workers intentin accedir a la regió crítica al mateix temps es redueix, per tant el repartiment de feina serà més equitatiu, encara que dista de ser perfecte. Podem observar-ho a la següent imatge:



```
alunne@pcrecanvib5: ~/Desktop/SDX/Seminar Assignments/Muty
File Edit View Search Terminal Help
John: lock released
Paul: lock taken in 64 ms
Paul: lock released
Ringo: lock taken in 373 ms
Ringo: lock released
John: lock taken in 398 ms
John: lock released
Paul: lock taken in 258 ms
Paul: lock released
John: lock taken in 456 ms
John: lock released
Ringo: lock taken in 793 ms
Ringo: lock released
Paul: lock taken in 694 ms
Paul: lock released
John: lock taken in 215 ms
John: lock released
Ringo: lock taken in 387 ms
Ringo: lock released
George: lock taken in 2149 ms
George: lock released
Ringo: lock taken in 290 ms
Ringo: lock released
John: lock taken in 341 ms
John: lock released
Paul: lock taken in 299 ms
Paul: lock released
Paul: lock taken in 1 ms
Paul: lock released
John: lock taken in 259 ms
John: lock released
Ringo: lock taken in 74 ms
Ringo: lock released
George: lock taken in 717 ms
George: lock released
John: lock taken in 0 ms
John: lock released
Ringo: lock taken in 0 ms
(node2@127.0.0.1)32> muty:stop().
stop
Ringo: 16 locks taken, 330.0 ms (avg) for taking, 0 withdrawals
John: 20 locks taken, 207.15 ms (avg) for taking, 0 withdrawals
Paul: 15 locks taken, 377.4 ms (avg) for taking, 0 withdrawals
George: 10 locks taken, 955.8 ms (avg) for taking, 0 withdrawals
(node2@127.0.0.1)33>
```

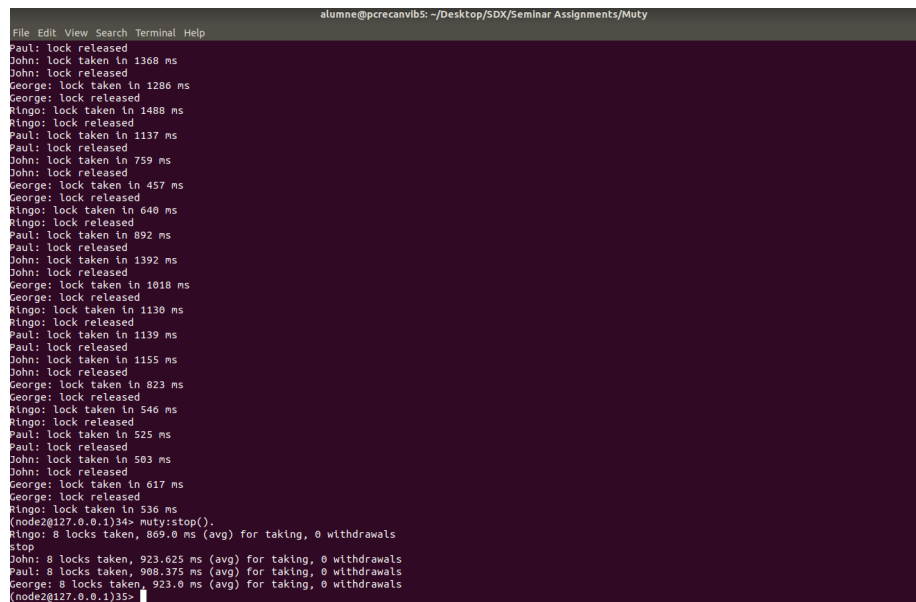
En aquest cas, ja no cal fer l'experiment per el cas de que el work time sigui igual al sleep time, ja que el resultat serà el mateix.

Lamport's time

1. Repeat the previous tests to compare this version with the former ones.

Test 1: Sleep time inferior a work time

En aquest cas, per millorar la implementació anterior, hem aplicat el algorisme de Lamport, que ens permetrà un millor repartiment de la feina. Això es degut a que en el moment que dos workers demanin accés a la regió crítica, el que tingui el clock més petit, serà el que accedirà, no el que tingui el ID més petit. Per altra banda, en cas de que el temps de work sigui molt més gran que el temps de sleep, els workers poden enviar missatges de release. Tal i com podem observar en la següent imatge:



```
File Edit View Search Terminal Help
alume@pcrecanvib5: ~/Desktop/SDX/Seminar Assignments/Muty
Paul: lock released
John: lock taken in 1368 ms
John: lock released
George: lock taken in 1286 ms
George: lock released
Ringo: lock taken in 1488 ms
Ringo: lock released
Paul: lock taken in 1137 ms
Paul: lock released
John: lock taken in 759 ms
John: lock released
George: lock taken in 457 ms
George: lock released
Ringo: lock taken in 640 ms
Ringo: lock released
Paul: lock taken in 892 ms
Paul: lock released
John: lock taken in 1392 ms
John: lock released
George: lock taken in 1018 ms
George: lock released
Ringo: lock taken in 1130 ms
Ringo: lock released
Paul: lock taken in 1139 ms
Paul: lock released
John: lock taken in 1155 ms
John: lock released
George: lock taken in 823 ms
George: lock released
Ringo: lock taken in 546 ms
Ringo: lock released
Paul: lock taken in 525 ms
Paul: lock released
John: lock taken in 503 ms
John: lock released
George: lock taken in 617 ms
George: lock released
Ringo: lock taken in 536 ms
(node2@127.0.0.1)34> muty:stop().
Ringo: 8 locks taken, 869.0 ms (avg) for taking, 0 withdrawals
stop
John: 8 locks taken, 923.625 ms (avg) for taking, 0 withdrawals
Paul: 8 locks taken, 988.375 ms (avg) for taking, 0 withdrawals
George: 8 locks taken, 923.0 ms (avg) for taking, 0 withdrawals
(node2@127.0.0.1)35>
```

Per altra part, tal i com passa en la secció anterior, si el temps de work és més gran que el temps de withdrawal, hi haurà workers, que no seran capaços de accedir a la regió crítica.

Test 2: Sleep time superior a work time

En el cas de que el sleep time sigui més gran que el work time, els workers, no tindran problema per accedir a la regió crítica, ja que com podem observar en la següent imatge el temps de lock és pràcticament 0.

```
alunne@pcrecanvib5: ~/Desktop/SDX/Seminar Assignments/Mutx
File Edit View Search Terminal Help
Ringo: lock taken in 0 ms
Ringo: lock released
George: lock taken in 0 ms
George: lock released
John: lock taken in 1 ms
John: lock released
Ringo: lock taken in 1 ms
Ringo: lock released
Paul: lock taken in 0 ms
Paul: lock released
John: lock taken in 1 ms
John: lock released
Paul: lock taken in 1 ms
Paul: lock released
Paul: lock taken in 0 ms
Paul: lock released
Paul: lock taken in 1 ms
Paul: lock released
Ringo: lock taken in 0 ms
Ringo: lock released
George: lock taken in 7 ms
George: lock released
George: lock taken in 0 ms
George: lock released
John: lock taken in 0 ms
John: lock released
Ringo: lock taken in 2 ms
Ringo: lock released
Paul: lock taken in 5 ms
Paul: lock released
George: lock taken in 1 ms
George: lock released
Paul: lock taken in 0 ms
Paul: lock released
John: lock taken in 0 ms
John: lock released
Ringo: lock taken in 1 ms
Ringo: lock released
(node2@127.0.0.1)42> mutx:stop().
Ringo: 19 locks taken, 0.7368421852631579 ms (avg) for taking, 0 withdrawals
stop
John: 20 locks taken, 0.65 ms (avg) for taking, 0 withdrawals
Paul: 20 locks taken, 1.05 ms (avg) for taking, 0 withdrawals
George: 24 locks taken, 0.9166666666666666 ms (avg) for taking, 0 withdrawals
(node2@127.0.0.1)43>
```

3 Open questions

4 Personal opinion