

UNIVERSITAT POLITÈCNICA DE CATALUNYA

ESCOLA POLITÈCNICA SUPERIOR

D'ENGINYERIA DE VILANOVA I LA GELTRÚ



**SISTEMES OPERATIUS DISTRIBUÏTS I EN XARXA**

Groupy: a group membership service

Profesor/a: Jordi García

Estudiantes:

Fèlix Andrés Navarro  
Jordi Calatayud Álvarez

20/12/2024

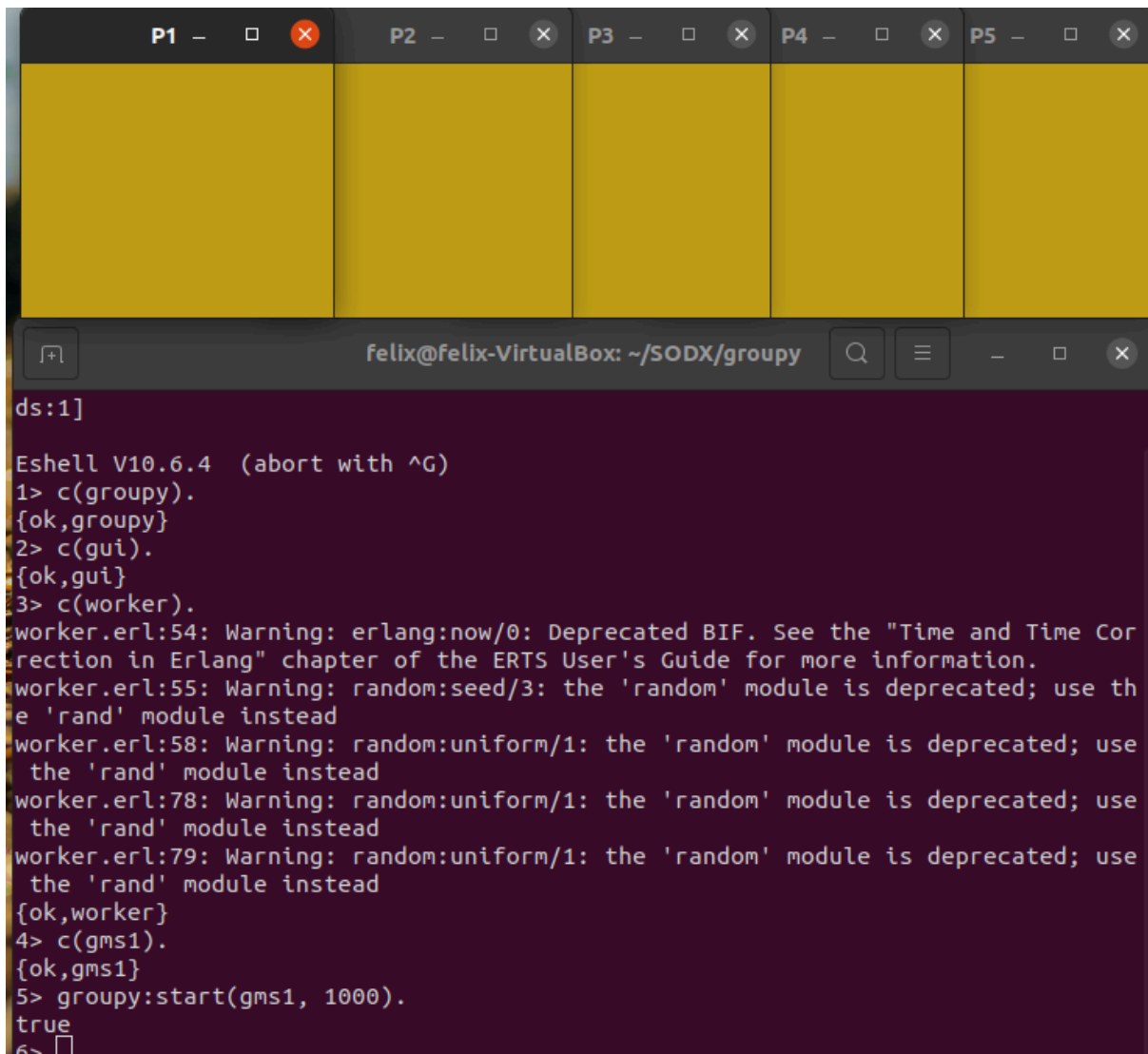
## GMS1

- 1) Do some experiments to see that you can create a group, add some peers and keep their state coordinated. You can use the following code to start and stop the whole system. Note that we are using the name of the module (i.e. gms1) as the parameter Module to the start procedure. All the workers but the first one need to know a member of the group in order to join. Sleep stands for up to how many milliseconds the workers should wait until the next message is sent.

```
Felix@Felix-VirtualBox:~/SODX/groupy$ erl
Erlang/OTP 22 [erts-10.6.4] [source] [64-bit] [smp:8:8] [ds:8:8:10] [async-threa
ds:1]

Eshell V10.6.4 (abort with ^G)
1> c(groupy).
{ok,groupy}
2> c(gui).
{ok,gui}
3> c(worker).
worker.erl:54: Warning: erlang:now/0: Deprecated BIF. See the "Time and Time Cor
rection in Erlang" chapter of the ERTS User's Guide for more information.
worker.erl:55: Warning: random:seed/3: the 'random' module is deprecated; use th
e 'rand' module instead
worker.erl:58: Warning: random:uniform/1: the 'random' module is deprecated; use
the 'rand' module instead
worker.erl:78: Warning: random:uniform/1: the 'random' module is deprecated; use
the 'rand' module instead
worker.erl:79: Warning: random:uniform/1: the 'random' module is deprecated; use
the 'rand' module instead
{ok,worker}
4> c(gms1).
{ok,gms1}
5> █
```

Aquí compilem tots els programes amb èxit.



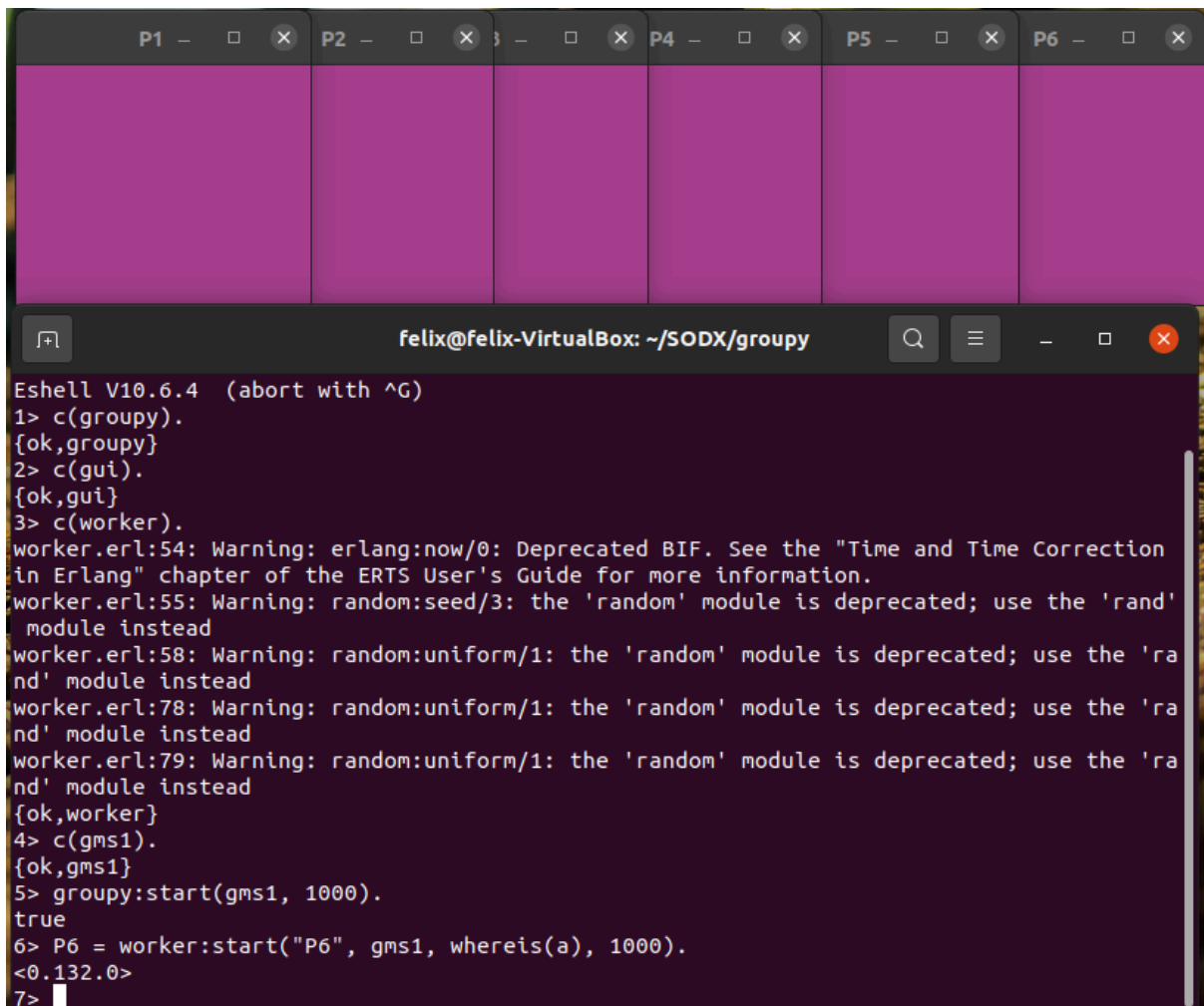
The screenshot shows a terminal window titled 'felix@felix-VirtualBox: ~/SODX/groupy'. The terminal output is as follows:

```
ds:1]
Eshell V10.6.4 (abort with ^G)
1> c(groupy).
{ok,groupy}
2> c(gui).
{ok,gui}
3> c(worker).
worker.erl:54: Warning: erlang:now/0: Deprecated BIF. See the "Time and Time Cor
rection in Erlang" chapter of the ERTS User's Guide for more information.
worker.erl:55: Warning: random:seed/3: the 'random' module is deprecated; use th
e 'rand' module instead
worker.erl:58: Warning: random:uniform/1: the 'random' module is deprecated; use
the 'rand' module instead
worker.erl:78: Warning: random:uniform/1: the 'random' module is deprecated; use
the 'rand' module instead
worker.erl:79: Warning: random:uniform/1: the 'random' module is deprecated; use
the 'rand' module instead
{ok,worker}
4> c(gms1).
{ok,gms1}
5> groupy:start(gms1, 1000).
true
6> 
```

Com podem veure aquí, al iniciar groupy amb gms1, s'obren 5 processos sincronitzats (canviant de color alhora).

```
6> P6 = worker:start("P6", gms1, whereis(a), 1000).
<0.132.0>
7> 
```

Fem la prova d'afegir un sisè procés, li diem que segueixi al procés líder (a), i li posem els mateixos milisegons (1000).



```
Eshell V10.6.4 (abort with ^G)
1> c(groupy).
{ok,groupy}
2> c(gui).
{ok,gui}
3> c(worker).
worker.erl:54: Warning: erlang:now/0: Deprecated BIF. See the "Time and Time Correction
in Erlang" chapter of the ERTS User's Guide for more information.
worker.erl:55: Warning: random:seed/3: the 'random' module is deprecated; use the 'rand'
module instead
worker.erl:58: Warning: random:uniform/1: the 'random' module is deprecated; use the 'ra
nd' module instead
worker.erl:78: Warning: random:uniform/1: the 'random' module is deprecated; use the 'ra
nd' module instead
worker.erl:79: Warning: random:uniform/1: the 'random' module is deprecated; use the 'ra
nd' module instead
{ok,worker}
4> c(gms1).
{ok,gms1}
5> groupy:start(gms1, 1000).
true
6> P6 = worker:start("P6", gms1, whereis(a), 1000).
<0.132.0>
7> 
```

Com podem veure, s'ha afegit el sisè procés sincronitzat amb el líder, i en conseqüència, amb els altres processos també.

- 2) Split the groupy module and make the needed adaptations to enable each worker to run in different machines. Remember how names registered in remote nodes are referred and how Erlang runtime should be started to run distributed programs.

Separarem el codi de groupy en 5 codis: groupy1, groupy2, groupy3, groupy4, groupy5. Cadascun per a un worker diferent. Com podem observar en els codis en cadascun hem separat la creació d'un worker de manera que els codis son una copia identica except per el nom.



```
P1 - □ × P2 - □ × P3 - □ × P4 - □ × P5 - □ ×
felix@felix-VirtualBox: ~/SODX/groupy
felix@felix-VirtualBox:~/SODX/groupy$ erl -name node1@192.168.1.221 -setcookie secret
Erlang/OTP 22 [erts-10.6.4] [source] [64-bit] [smp:8:8] [ds:8:8:10] [async-threads:1]

Eshell V10.6.4 (abort with ^G)
(node1@192.168.1.221)1> net_adm:ping('node2@192.168.1.153').
pong
(node1@192.168.1.221)2> groupymod:start(gms1, 1000).
true
```

Iniciem el node 1 a l'ordinador de torre, que serà el node 1, amb la seva respectiva IP, i iniciem el groupy modificat per a connexions entre diferents màquines. També es pot veure com he fet ping a l'altre màquina, i al respondre "pong" determinem que es poden comunicar.



```
felix@felix-VirtualBox: ~/SODX/groupy
felix@felix-VirtualBox:~/SODX/groupy$ erl -name node2@192.168.1.153 -setcookie s
ecret
Erlang/OTP 24 [erts-12.2.1] [source] [64-bit] [smp:1:1] [ds:1:1:10] [async-threa
ds:1] [jit]

Eshell V12.2.1 (abort with ^G)
(node2@192.168.1.153)1> groupymod:start_remote('node1@192.168.1.221', gms1, 1000
).
ok
(node2@192.168.1.153)2> □
```

RemoteWo... - □ ×



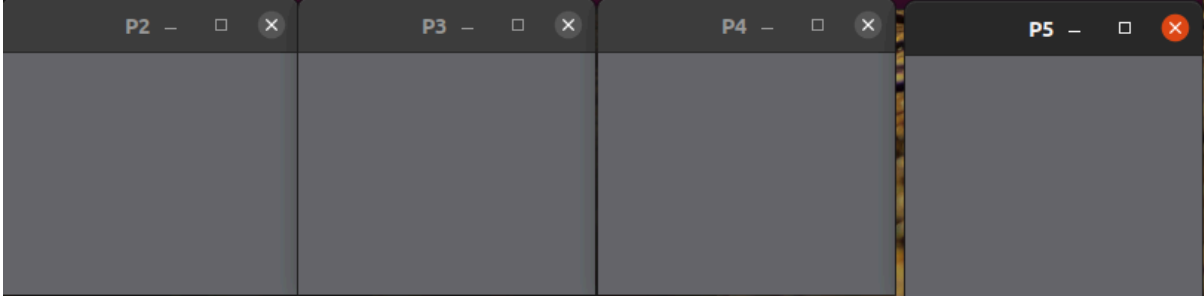
A l'altre màquina, iniciem un RemoteWorker que estarà sincronitzat al node1.

# GMS2

## Experiments.

Do some experiments to see if the peers can keep their state coordinated even if nodes crash.

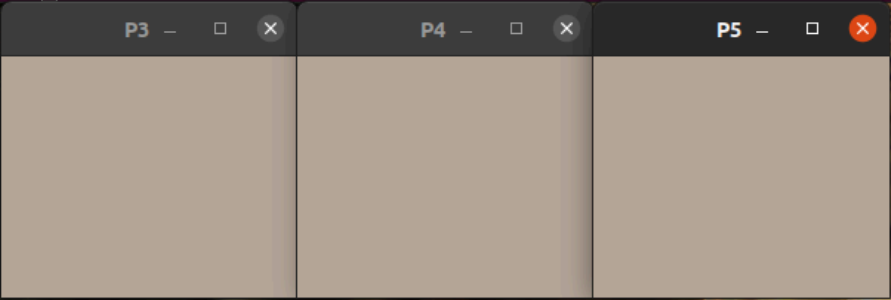
```
18> c(gms2).
{ok,gms2}
19> groupy:start(gms2, 1000).
true
20> whereis(a).
<0.200.0>
21> exit(whereis(a), kill).
true
slave P2: strange message {'DOWN',#Ref<0.3589070967.3533438980.234421>,process,<0.205.0>
,killed}
slave P3: strange message {'DOWN',#Ref<0.3589070967.3533438980.234425>,process,<0.205.0>
,killed}
slave P4: strange message {'DOWN',#Ref<0.3589070967.3533438980.234428>,process,<0.205.0>
,killed}
slave P5: strange message {'DOWN',#Ref<0.3589070967.3533438979.234445>,process,<0.205.0>
,killed}
Node P2 detected leader failure: <0.205.0>
Node P3 detected leader failure: <0.205.0>
Node P4 detected leader failure: <0.205.0>
Node P2 becoming leader
Node P3 elects new leader: <0.207.0>
Node P4 elects new leader: <0.207.0>
22> □
```



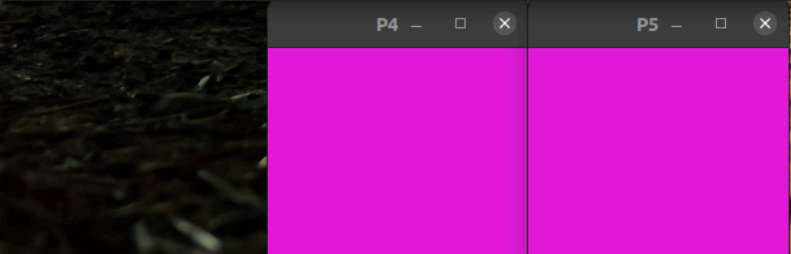
Com podem observar aquí, iniciem el gms2, i matem al líder. Es poden veure els missatges dels altres processos slaves detectant la fallada de líder, i proposant un de nou. Després d'escollir líder, els processos segueixen sincronitzats canviant de color alhora.

Aquí provem d'anar matant més processos, i com podem veure, tornen a escollir líder i segueixen sincronitzats.

```
felix@felix-VirtualBox: ~/SODX/groupy
,killed}
slave P4: strange message {'DOWN',#Ref<0.3589070967.3533438980.234428>,process,<0.205.0>
,killed}
slave P5: strange message {'DOWN',#Ref<0.3589070967.3533438979.234445>,process,<0.205.0>
,killed}
Node P2 detected leader failure: <0.205.0>
Node P3 detected leader failure: <0.205.0>
Node P4 detected leader failure: <0.205.0>
Node P2 becoming leader
Node P3 elects new leader: <0.207.0>
Node P4 elects new leader: <0.207.0>
22> whereis(a).
undefined
23> whereis(b).
<0.201.0>
24> exit(whereis(b), kill).
true
Node P4 detected leader failure: <0.207.0>
Node P3 detected leader failure: <0.207.0>
Node P5 detected leader failure: <0.207.0>
Node P4 elects new leader: <0.208.0>
Node P3 becoming leader
Node P5 elects new leader: <0.208.0>
25> 
```



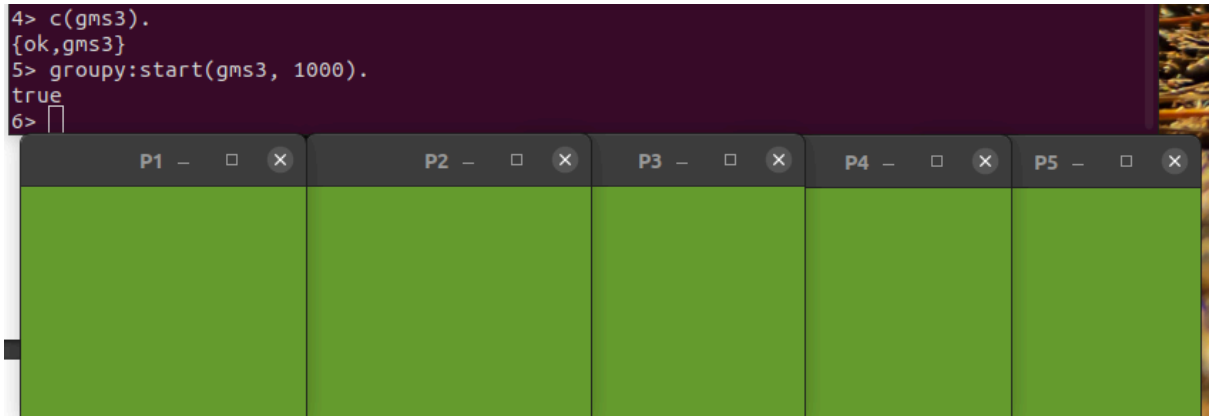
```
felix@felix-VirtualBox: ~/SODX/groupy
Node P2 becoming leader
Node P3 elects new leader: <0.207.0>
Node P4 elects new leader: <0.207.0>
22> whereis(a).
undefined
23> whereis(b).
<0.201.0>
24> exit(whereis(b), kill).
true
Node P4 detected leader failure: <0.207.0>
Node P3 detected leader failure: <0.207.0>
Node P5 detected leader failure: <0.207.0>
Node P4 elects new leader: <0.208.0>
Node P3 becoming leader
Node P5 elects new leader: <0.208.0>
25> whereis(c).
<0.202.0>
26> exit(whereis(c), kill).
true
Node P5 detected leader failure: <0.208.0>
Node P4 detected leader failure: <0.208.0>
Node P5 elects new leader: <0.209.0>
Node P4 becoming leader
27> 
```



# GMS3

Experiments.

- 1) Repeat the experiments to see if now the peers can keep their state coordinated even if nodes crash.



- 2) Try to keep a group rolling by adding more nodes as existing nodes die.

## What could possibly go wrong

- 1) ¿Cómo deberíamos cambiar la implementación para manejar los mensajes posiblemente perdidos?

Para manejar los mensajes perdidos, sería necesario implementar un mecanismo de retransmisión confiable. Esto incluiría que los nodos envíen confirmaciones (ACKs) al líder por cada mensaje recibido y que el líder mantenga un registro de mensajes enviados hasta que todos los nodos correctos confirmen su recepción. Además, los nodos podrían solicitar retransmisiones si detectan saltos en los números de secuencia, asegurando que los mensajes lleguen a todos los miembros del grupo.

- 2) ¿Cómo impactaría esto el rendimiento?

El rendimiento se vería afectado principalmente en términos de latencia y carga de red. Las confirmaciones y retransmisiones añadirían retrasos en la entrega de mensajes, además de aumentar la carga de trabajo del líder y el tráfico en la red. También crecería el consumo de memoria, ya que el líder necesitaría almacenar un historial de mensajes pendientes, lo que podría ser un problema en grupos grandes o con alta frecuencia de mensajes.

- 3) ¿Qué pasaría si sospechamos erróneamente que el líder ha fallado?



Si sospechamos incorrectamente que el líder ha fallado, el sistema podría entrar innecesariamente en un proceso de elección de un nuevo líder, interrumpiendo temporalmente el servicio. Esto podría llevar a duplicaciones o inconsistencias en los mensajes, especialmente si el líder original sigue activo. Además, estas elecciones innecesarias generarían sobrecarga en el sistema y reducirían su eficiencia, haciendo crucial implementar detectores de fallos más precisos para minimizar falsas alarmas.