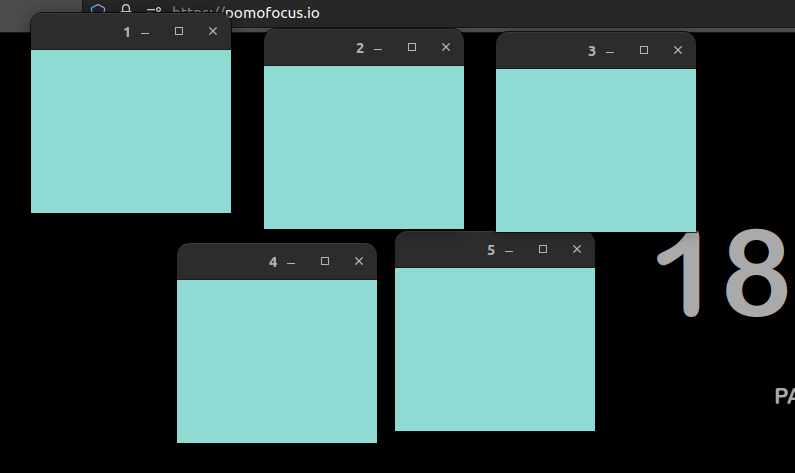
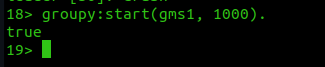
**Groupy: a group membership service**

**Pau Alcázar i Francesco Oncins**

**Experiments.** **i)** 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. **ii)** 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.

i) La resolució d’aquest primer experiment es troba en el document gms1.erl





Podem observar que els 5 workers estan sincronitzats ja que no hi ha hagut cap error.

ii) La resolució d’aquest segon experiment es troba en el document groupy\_nou.erl

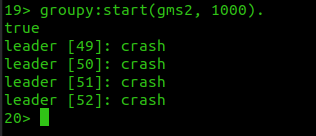
erl name <node\_name>@127.0.0.1 setcookie secret

groupy2:start(gms1, 1000, a, "1", 1, leader). -> per iniciar al lider

groupy2:start(gms1, 1000, <atom\_id>, <id>, <rand>, {a, 'leader@127.0.0.1'}). -> per iniciar a cada slave

La conclusió de l’experiment és la mateix que l’anterior, els workers estan sincronitzats.

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



Perquè els peers puguin mantenir el seu estat després que un o més nodes caiguin, hem creat el gms2.erl. La funció slave modificada al codi ara fa un seguiment del líder i reacciona als canvis o fracassos d’aquest. La nova funció election facilita la selecció d'un nou líder entre els esclaus existents si el líder actual falla.

Com s’ha comentat abans, el codi és correcte ja va canviant de lider quan és necessari.

**Experiments.** Repeat the experiments and see if you can have the state of the workers become out of synch. (gms2)

Hem afegit 2 noves funcions i canviat la inicialització del leader i slaves:  
bcast(Name, Msg, Nodes) ->

lists:foreach(fun(Node) ->

Node ! Msg,

crash(Name, Msg)

end,

Nodes).

crash(Name, Msg) ->

case rand:uniform(?arghh) of

?arghh ->

io:format("leader ~s CRASHED: msg ~w~n", [Name, Msg]),

exit(no\_luck);

\_ ->

ok

end.

init(Name, Master) ->

{A1,A2,A3} = now(),

random:seed(A1, A2, A3),

leader(Name, Master, []).

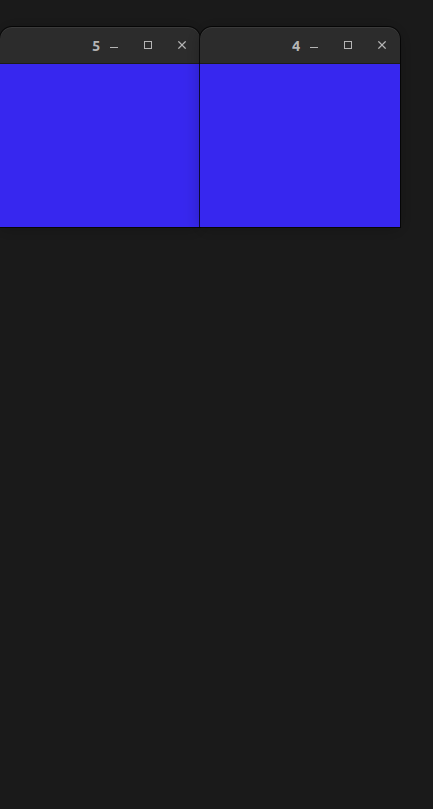
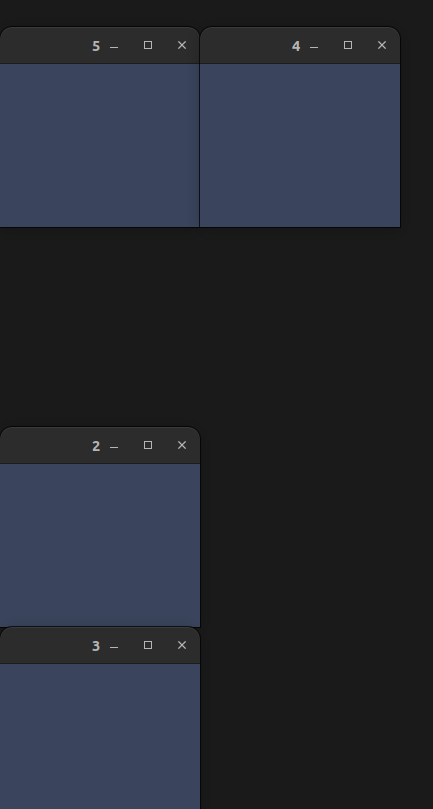
El mateix per la inicialització dels slaves

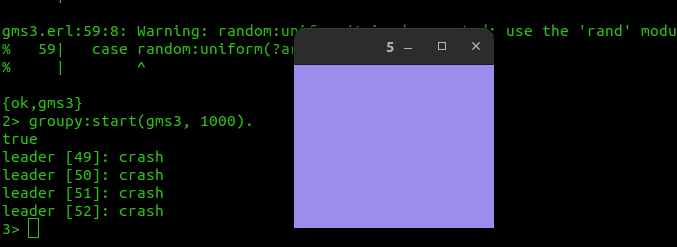
Si el lider falla al enviar els missatges de sincronització, aquesta falla.

**Open Questions.** Why is this happening?

Quan el bloqueig o l'aturada es produeix durant un multicast, alguns nodes poden obtenir el nou color mentre que la resta continuen amb l'anterior, i així es sincronitzen.

**Experiments.** **i)** Repeat the experiments to see if now the peers can keep their state coordinated even if nodes crash. **ii)** Try to keep a group rolling by adding more nodes as existing nodes die. (gms3)

i)



Sempre es manté sincronitzat, tot i que falli un procés es segueix mantenint sincronitzat.

ii) El codi que hi ha gms3, fa referència aquest 2on experiment.

**Open Questions.** **i)** How would we have to change the implementation to handle the possibly lost messages? **ii)** How would this impact performance? **iii)** What would happen if we wrongly suspect the leader to have crashed?

i) Si P1 es bloqueja mentre envia un multicast, els processos que van rebre el missatge abans que P1 es bloquegés haurien de tornar a transmetre aquest missatge als que no l'han pogut rebre a causa de l'error de P1 abans de canviar la vista.

ii) Implicaria que molts més missatges estarien errant per la xarxa, augmentant la latència i/o el rendiment, però tots els processos acabarien rebent tots els missatges.

iii) El líder es desincronitzaria respecte als altres, i els altres es sincronitzaran entre ells, ja que entre ells s'escolliria un nou líder.