Report – MRS Assessed Exercise 2465714a

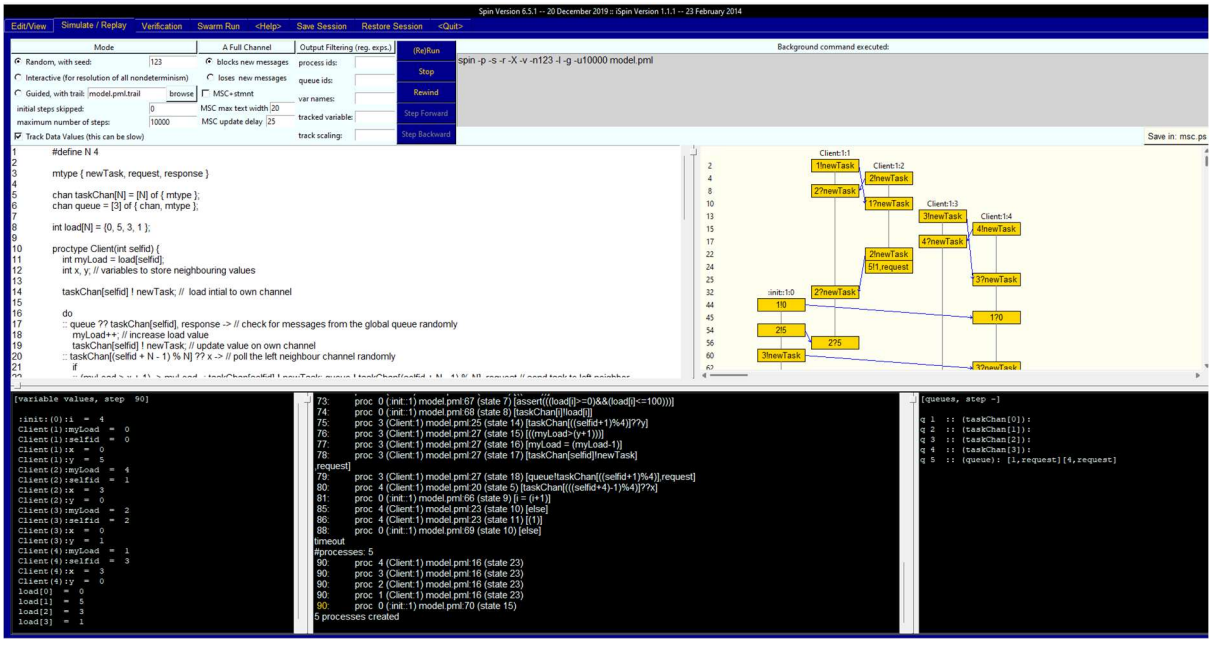
Problem:

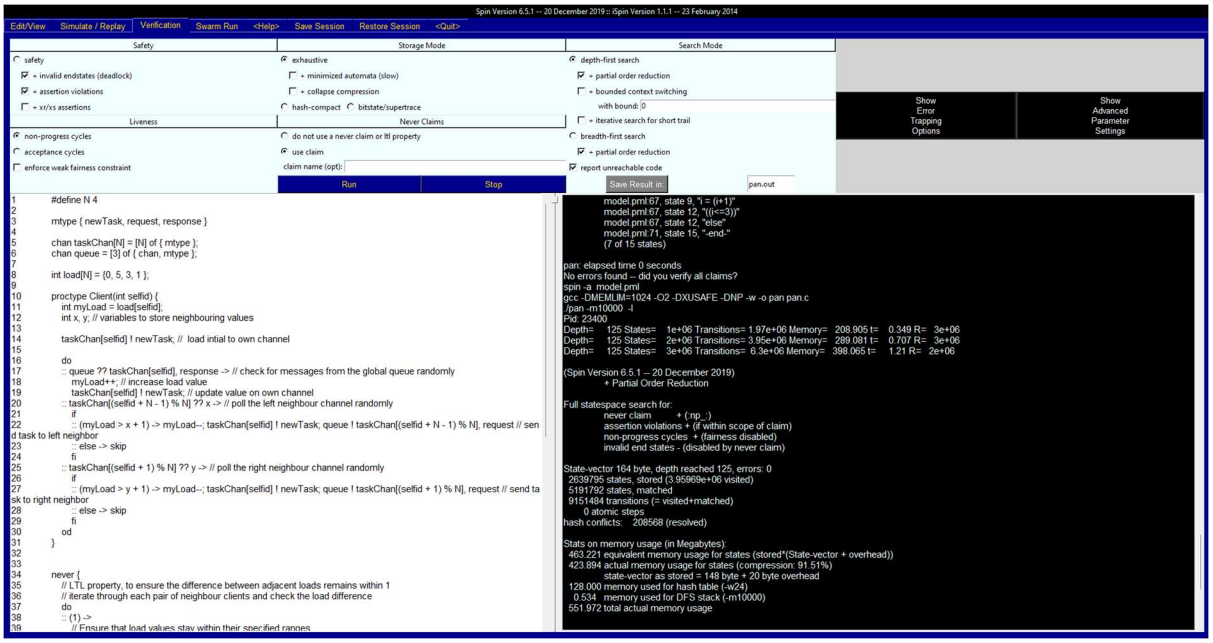
The task in hand was to create a promela model which simulates a load balancer protocol which can deal with four client processes with their corresponding states. The balance is maintained by ensuring each client keeps a balanced load with their neighbours. It does this by comparing current load states then redistributing appropriately if there are differences greater than one. Each client process interacts with their adjacent clients or neighbours via channels, receiving load data.

Model:

When an imbalance is detected, a task is sent to the neighbour with the lower current load, in the comparison which detected the imbalance. To ensure balance the model must ensure that the difference between neighbour's loads remains within 1 and the global queue handles messages correctly. The model assumes messages between the client processes and the global queue are deterministic and are always received and sent. It is also assumed that load adjustments are made instantly and that loads values are integers within a certain range which can be adjusted.

Here are screenshots of the model being tests and verified. (Screenshots will also be attached if it is difficult to view here.)



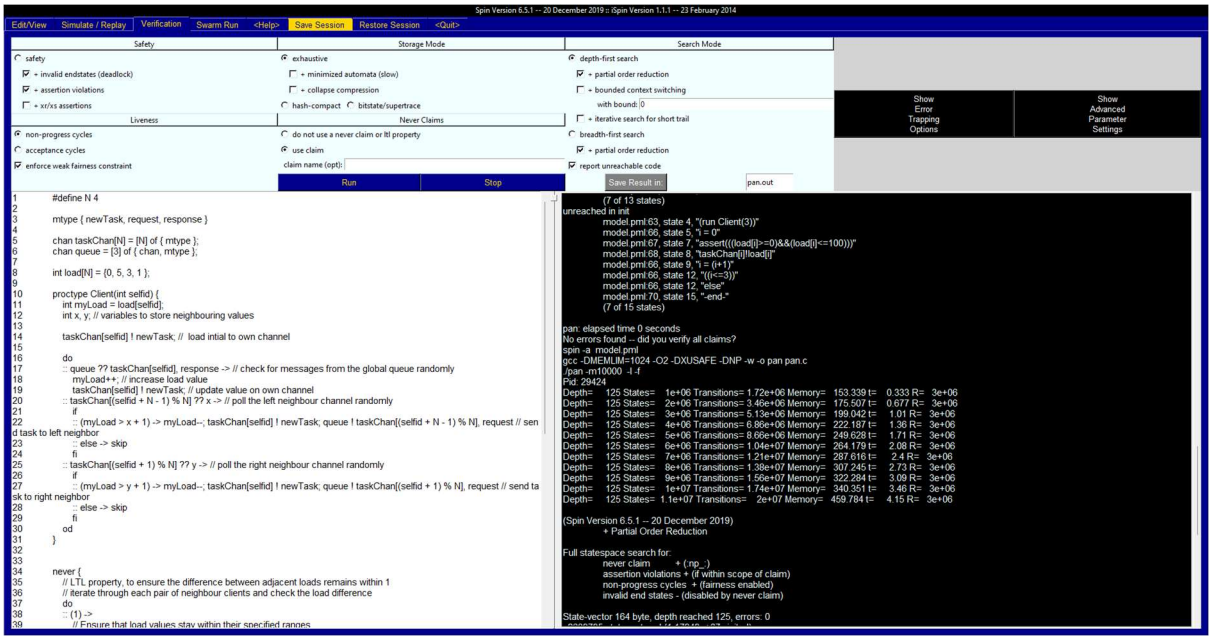



```
C:\Workspace\Promela\AE>spin -a model.pml

C:\Workspace\Promela\AE>spin model.pml
warning: never claim not used in random simulation
      timeout
#processes: 5
                queue 1 (taskChan[0]):
                queue 2 (taskChan[1]):
                queue 3 (taskChan[2]):
                queue 4 (taskChan[3]):
                queue 5 (queue): [1,request]
                load[0] = 0
                load[1] = 5
                load[2] = 3
                load[3] = 1
 83:     proc  4 (Client:1) model.pml:16 (state 23)
 83:     proc  3 (Client:1) model.pml:16 (state 23)
 83:     proc  2 (Client:1) model.pml:16 (state 23)
 83:     proc  1 (Client:1) model.pml:16 (state 23)
 83:     proc  0 (:init::1) model.pml:71 (state 15) <valid end state>
5 processes created

C:\Workspace\Promela\AE>
```

Here you can see that my specification will satisfy the property ϕ when Weak Fairness is applied. (Screenshots will also be attached if it is difficult to view here.)



We will consider the follow scenario/execution path. In order to show that ϕ will not hold when weak fairness is not applied.

Load[0] = 9, Load[0] = 5, Load[0] = 2, Load[0] = 1

Step 1: Client[0] is selected for checking and decreases its load by 1 as there is an imbalance

Step 2: As there is no weak fairness, Client[0] is selected again and its load decreases once more

Step 3: Client[0] will continue to be chosen for execution. This results in the neighbours not being given the chance to balance their load.

This leads to a persistent imbalance, and the load of Client 0 will become much lower than its neighbours as they increase, this violates the property ϕ (LTL) property. Thus, the system fails to achieve load balance as other clients remain at higher values. This example shows that without weak fairness a single high load client can dominate the execution of the protocol leading to a violation of ϕ, weak fairness ensures that each client will have a chance to be checked and balance maintained.