# Question 1

## Part 2

Assumption: *“propagate the latest weather update”* means that all connected clients will not only be notified of the latest weather but will also receive the latest weather.

#define isLatestSent (cm\_chan ? <USER\_UPDATED\_WEATHER, DUMMY\_VAL, DUMMY\_VAL>)

Let us define

* isLatestSent – whether the WCP has sent the update message to the CM.
* are\_all\_new\_successful- whether all the connected clients reported success for using the new weather.

Then, the LTL property used is:

[] (isLatestSent -> <> are\_all\_new\_successful )

It means:

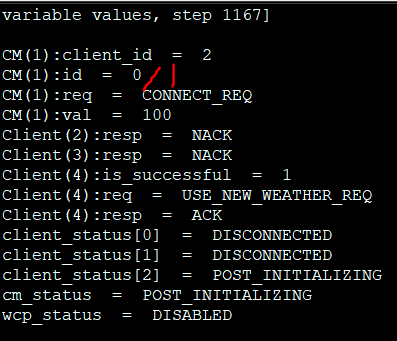
Globally, if the Communication Manager is sent a USER\_UPDATED\_WEATHER message, then eventually all connected clients will send a USE\_NEW\_WEATHER\_RESP response with is\_successful = 1.

## Part 3

describe the counter example you get from SPIN and explain how it can violate the property. Include the screenshot of the counter example (e.g. message sequence chart) in the report.

No message is sent to client in some cases? Client blocks on receive

Property can be “every receive should eventually get a response from the corresponding send”?

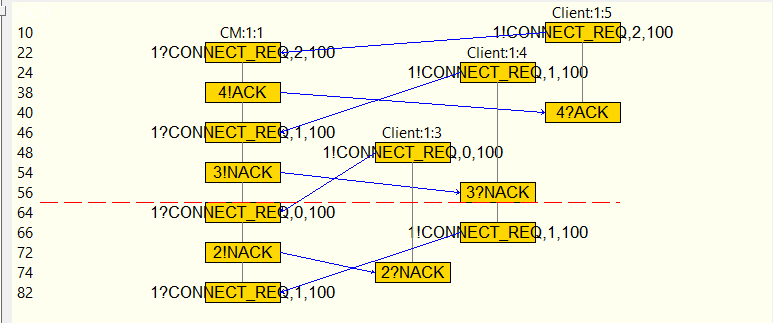


Without atomic for PRE\_INIT and INIT in CM

## Part 4

explain the deadlock free implementation with proof.

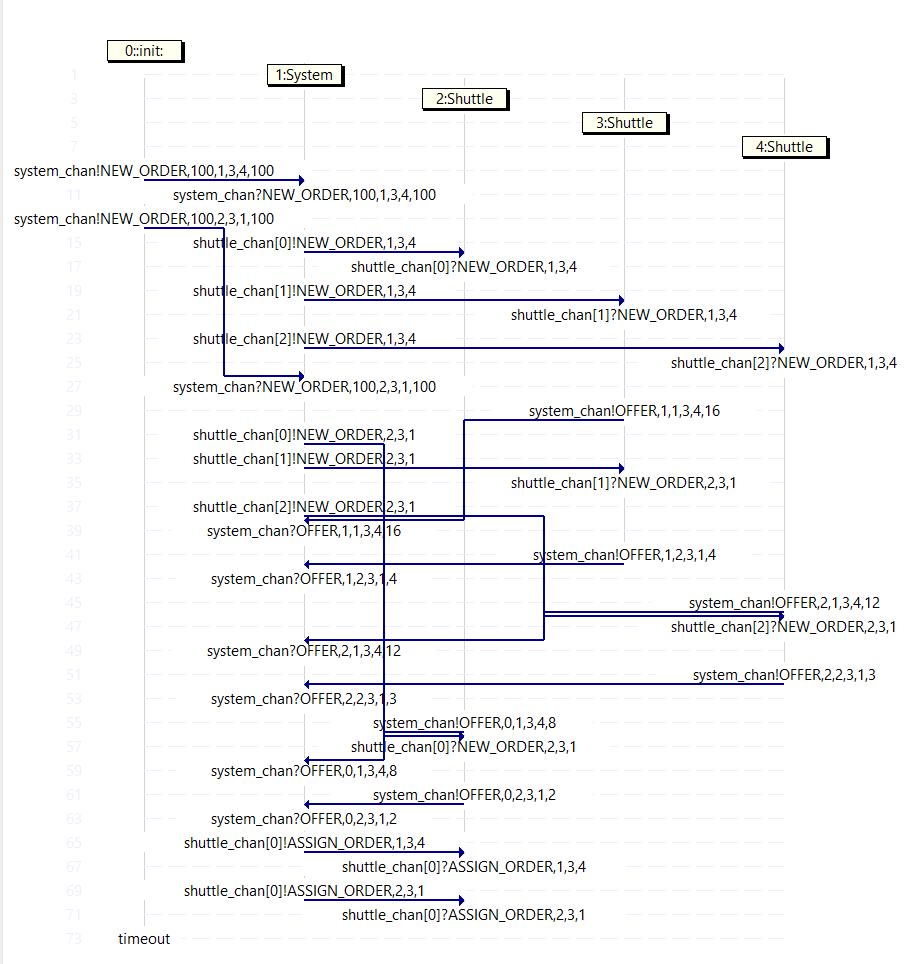
Send NACK or add timeout and disconnect?



# Question 2

## Part 2

Using SPIN and running spin -M railway.pml gives this Message Sequence Chart that shows one sequence of events that leads to the desired results.



### Sequence of events

#### Assignment of orders:

* Management System receives a NEW\_ORDER with start=1, dest=3, size=4 (order 1).
* Management System sends order 1 to the 3 shuttles using the shuttle\_chan[idx] buffer.
* Management System receives a NEW\_ORDER with start=2, dest=3, size=1 (order 2).
* Shuttle 2 sends an offer for order 1 to Management System with payment=16.
* Management System sends order 2 to the 3 shuttles using the shuttle\_chan[idx] buffer.
* Shuttle 2 sends an offer for order 2 to Management System with payment=4.
* Shuttle 3 sends an offer for order 1 to Management System with payment=12.
* Shuttle 3 sends an offer for order 2 to Management System with payment=3.
* Shuttle 1 sends an offer for order 1 to Management System with payment=8.
* Shuttle 1 sends an offer for order 2 to Management System with payment=2.
* Management System sends an ASSIGN\_ORDER message to Shuttle 1 for order 1.
* Management System sends an ASSIGN\_ORDER message to Shuttle 1 for order 2.

#### Processing of orders by Shuttle 1:

* Shuttle 1 processes the orders one at a time, starting from the earliest one.
* It chooses the initial direction to travel in based on the distance of the route to the start station.
* It then travels in the chosen direction, while checking that the track is used by only one shuttle.
* Once the start station (Station 1 and then Station 2) is reached, it loads the passengers.  
  It chooses the direction to travel in based on the distance of the route to the destination station.
* It then travels in the chosen direction, while checking that the track is used by only one shuttle.
* Once the destination station (Station 3 and then Station 3) is reached, it unloads the passengers.

At the end of the execution, the shuttles are:

* Shuttle 1: Stationary at Station 3 with no load
* Shuttle 2: Stationary at Station 1 with no load
* Shuttle 3: Stationary at Station 2 with no load

This can be seen from the Data Window in iSpin after the run terminates. For example, the screenshot below shows the values of the variables in the process corresponding to Shuttle 1 (note the values of curr\_station , load and status ) after running a simulation with seed = 123 on iSpin. Similar results are observed for Shuttle 2 and Shuttle 3.

