Assignment

Problem 1

Following is one of the simple solutions for problem 1. Any solution that is reasonable was given marks upon evaluation, provided that submission included detailed description for the models, LTL property etc. The model should simulate the weather update controller as described in the assignment question, with at least 3 clients. Following will be considered for the given Promela model;

- State changes should be atomic to prevent implementation deadlocks
- Client connection protocol should be implemented as stated in the description
- Weather update protocol should be implemented as stated in the description

Question 1

Attached code ("weather-controller-model.pml") is a sample implementation of the model, this is not the only possible way. Every submission was checked thoroughly which had different ways of implementation. You would get marks as long as your model exhibits correct behavior which can be observed through execution.

Question 2

Key property for the above controller should be that manual update should be enabled eventually after its disabled, since the main objective is to propagate latest weather information via the communication manager. This can be written in LTL as:

```
#define p1 (allowedManualUpdate == 1)
ltl deadlock { [] (!p1 -> <> p1) }
```

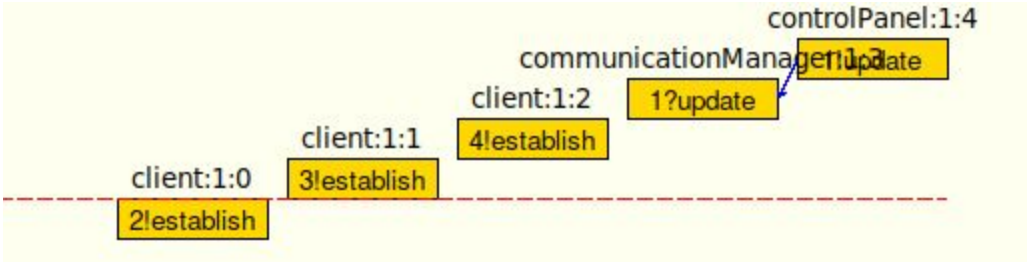
This can be formulated in multiple ways, marks were given for other equivalent LTL properties used, the only requirement is to define the property such that the continuous weather update is checked. Marks were given for properties with explanation for the variables/predicates in terms of the model.

Question 3

There is one possibility for a deadlock to occur as given below;

- An update request sent by the WCP when no client is connected
    - CM will go to “pre-updating” state and wait for clients to send a message for the update request, but since no client is connected it will remain in this state.
    - Clients will wait till the CM respond to their ‘establish’ request to initiate the connection, but since the CM is stuck so will the clients.
    - Panel will be blocked from manual updates, hence WCP is also stuck.

Using spin we can easily obtain this counter-example either using the LTL property defined above or using the safety property in iSpin.



Message sequence chart leading to deadlock

```
spin: trail ends after 24 steps
#processes: 5
 24:    proc  4 (controlPanel:1) weather-controller.pml:251 (state 2)
 24:    proc  3 (communicationManager:1) weather-controller.pml:149
(state 75)
 24:    proc  2 (client:1) weather-controller.pml:24 (state 3)
 24:    proc  1 (client:1) weather-controller.pml:24 (state 3)
 24:    proc  0 (client:1) weather-controller.pml:24 (state 3)
5 processes created
Exit-Status 0
```

Log messages at deadlock state

As shown above, if there is an update request from WCP when no clients are connected, CM will immediately change its status to ‘pre-updating’ and get stuck since no clients to send

messages. No client will be able to connect since CM is stuck, and WCP is disabled hence all processes gets stuck with no way to proceed.

From this property we can identify two other problems which are given below:

- During connection establishment
    - If a client fails to use new weather or get new weather, it will send a message fail to CM and CM would disconnect the client and go to ‘idle’ state.
    - Since the WCP is disabled, WCP is stuck and cannot proceed until CM enables manual update.
    - CM can only enable WCP when a client successfully establish the connection or when client responds ‘fail’ to the request ‘use new weather’.
    - Hence an undesired situation where weather updates cannot be done through the communication manager as needed.
- During connection establishment
    - A client can repeatedly try to make the connection but fails to get the new weather update
    - CP is blocked by CM until a connection is failed (only enabled if a client fails to use the weather update)

Question 4

There are multiple ways to fix the dead-lock without violating the given requirements. Any method that was fixing the dead-lock was given marks (even if it was violating the requirement). One way this can be rectified is by altering the protocol to proceed to ‘pre-update’ state only if there is any connected clients or go to ‘pre-update’ state, check if any client is connected.

Attached code (“fix-deadlock.pml”) is a sample fix of the model. You can verify the fix again using either the LTL property or the safety property in iSpin.

Problem 2

Following is one of the simple solutions for problem 2. Any solution that is reasonable was given marks upon evaluation, provided that submission included detailed description for the models, LTL property etc. The model should simulate the order management system as described in the assignment question, with at least 3 shuttles. Following will be considered for the given Promela model;

- State changes should be atomic to prevent implementation deadlocks
- Order management system should be implemented as stated in the description
- Railway management system should be implemented as stated in the description
- Shuttles should be implemented as stated in the description

Question 1

Attached code ("railway.pml") is a sample implementation of the model, this is not the only possible way. Every submission was checked thoroughly which had different ways of implementation. You would get marks as long as your model exhibits correct behavior which can be observed through execution.

Question 2

Two orders should be processed by the system which will eventually lead to all passengers transported to their destinations. The sequence of order announcement can be varied depending on the implementation, however all sequences should with-hold following:

- Both orders should be assigned to S_1
- S_1 cannot process both orders at the same time

There are 3 possible outcomes:

1.
    a. Order-1 is assigned to S_1
    b. S_1 loads passengers
    c. Reach to Station-3 and unloads passengers
    d. Order-2 is assigned to S_1
    e. Reach to Station-2 and load passenger
    f. Reach to Station-3 and unload passenger

2.
    a. Order-1 is assigned to S_1
    b. S_1 loads passengers
    c. Order-2 is assigned to S_1
    d. Reach to Station-3 and unloads passengers
    e. Reach to Station-2 and load passenger
    f. Reach to Station-3 and unload passenger
3.
    a. Order-1 is assigned to S_1
    b. Order-2 is assigned to S_1
    c. S_1 loads passengers
    d. Reach to Station-3 and unloads passengers
    e. Reach to Station-2 and load passenger
    f. Reach to Station-3 and unload passenger

Note: You can run the command “spin -B -T railway.pml” to obtain a valid sequence.