Assignment Introduction

CS4211 - Formal Methods for Software Engineering

Prepared by: Zhiyu Fan

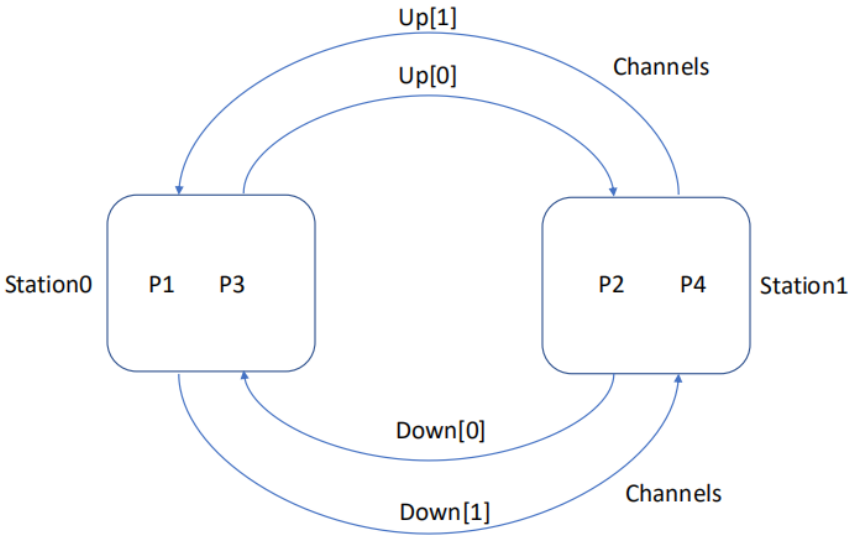
Notes

- This assignment is due before **23 October 2020, 9:59 PM**. No late submissions!
- Tool to be used: SPIN model checker http://spinroot.com/spin/whatispin.html
- This is an individual assignment. **Acts of plagiarism are subjected to disciplinary action by the university**

Notes

- Submission Instructions:
  - Create a folder named your matriculation number YourMatricNumber, e.g. U123456M.
  - Create the following files in this folder (name these files exactly as instructed.):
    - Assignment 1: Create three folders inside it
      - Question 1: The modified spin source and LTL property check files
      - Question 2: Your spin source implementation and LTL property check files
      - Question 3: Your spin source implementation and LTL property check files
    - Report.pdf
    - Readme.txt: all information required to reproduce the verification of your code
    - ListofFiles.pdf: Explaining all the files, if there are more files.
  - Zip the entire YourMatricNumber folder (including the folder itself and all files in it) into a file YourMatricNumber.zip.
  - Submit YourMatricNumber.zip to the Luminus Workbin Folder **LabSubmission**.

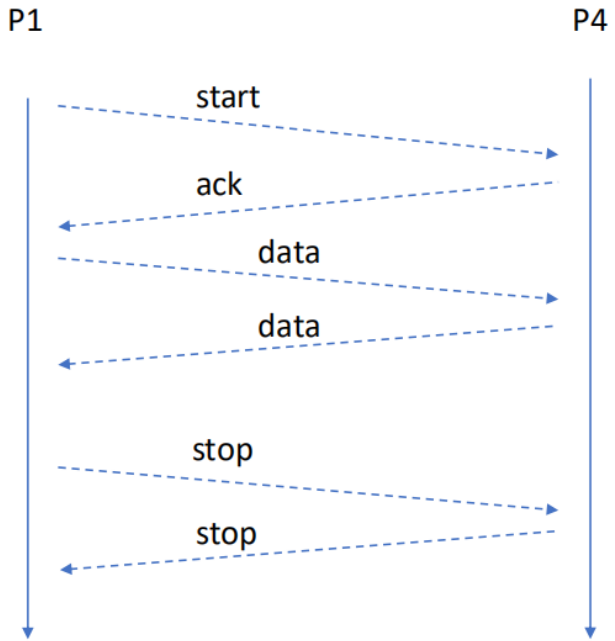
Question 1: Communication between two stations



- Two stations
- Each station has two operators (processes)
- Data communication via two Channels
  - One for sending actual data
  - One for sending ack signal

Question 1: Communication between two stations

Communication Demonstration



Questions [5 marks]

- Try to formalize the property that no communication between stations is aborted in the above protocol, i.e. any communication that is started (with a start signal) is finished (with a stop signal). Use Linear time temporal logic (LTL) to express your property description. **(1 mark)**

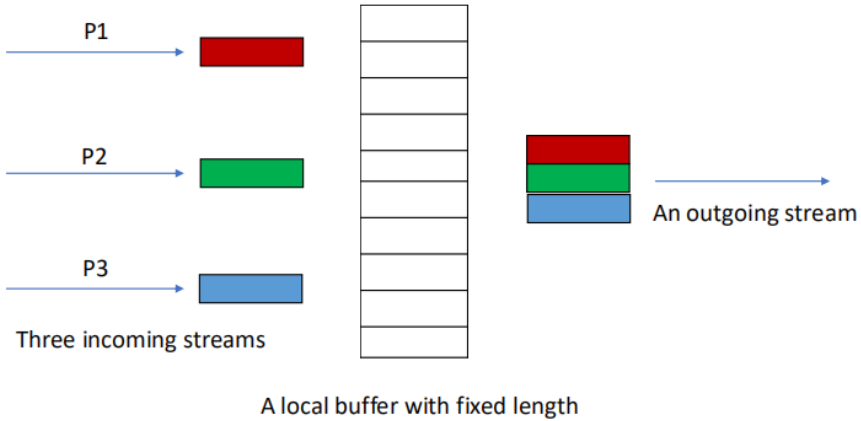
- The above property is not true for our protocol. Use the SPIN toolkit to find out why it is not true. Augment the provided model to make the above property true. **(4 marks)**

Question 2: Deadlock Verification

We have a process P which reassembles data packets it received to messages with **fixed buffer space**.

There could have **more than two** types of data packets sent to P

The messages reassembled by P must includes **all** types of data packets



NUS National University of Singapore

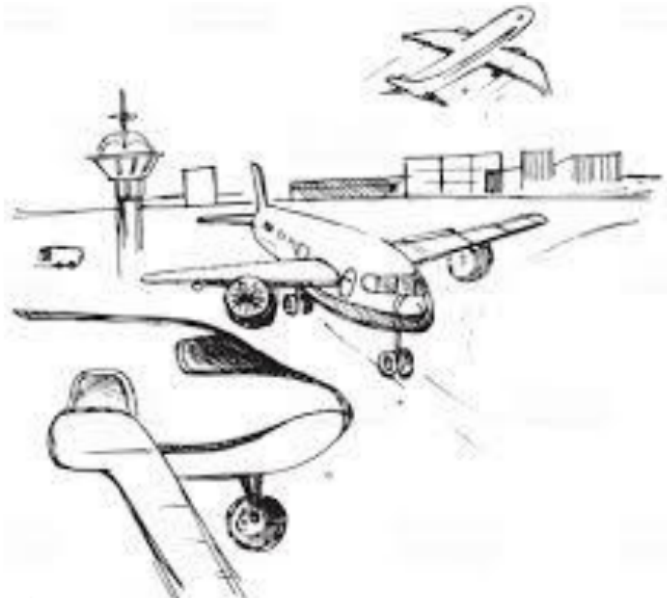
Questions [9 marks]

- Write a Promela model of a network node with three incoming streams of messages, and one outgoing stream. **(5 marks)**
  - Require that each message to be sent on the outgoing stream contains a structure of 3 fields, one fieldof type red, one of type green and of type blue (i.e., you need one of each type of incoming message to assemble one outgoing message.)
  - Each of the incoming data streams produces only messages of one specific type (say red, green, and blue for the three streams)
  - Use handshake for the incoming messages and store them inside the node in a local buff.

- Explain how the reassembly deadlock can occur for the given model, provide a detailed description (textual explanation) for your answer. **(2 marks)**

- Define the key property in Linear-time temporal logic (LTL), use SPIN to prove that it can be violated, and show the counter-example. **(2 marks)**

Question 3: Weather update system [11 marks]



Models:

Control weather updating to all weather-aware clients

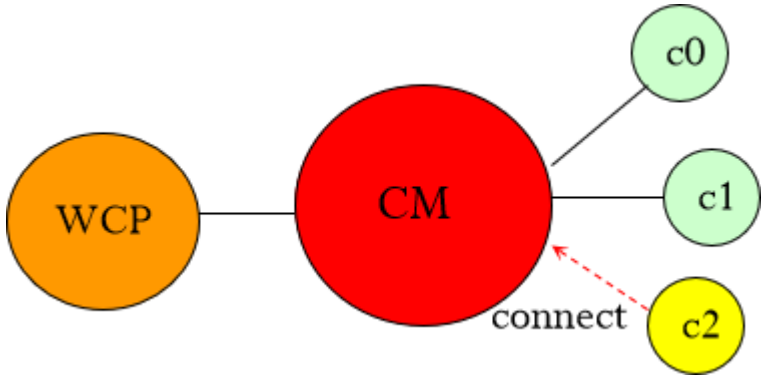
- A weather control panel (WCP)
- Many weather-aware clients
- A communication manager (CM)

Behaviors:

- Client Initialization
- Weather Update

Behavior 1: Client Initialization

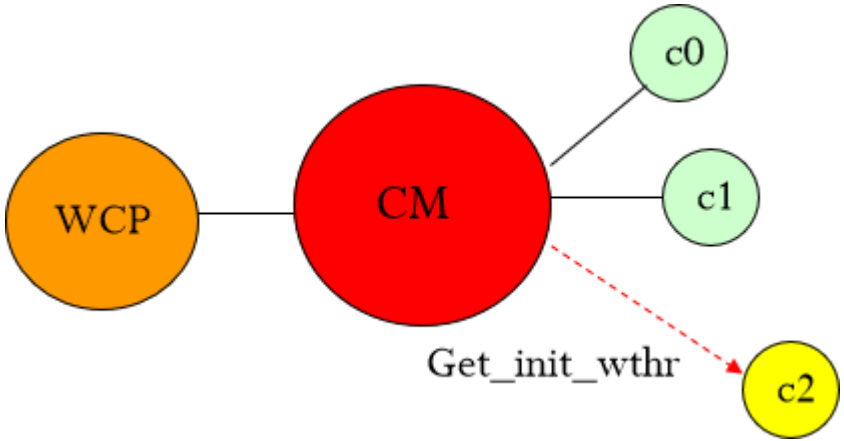
- A disconnected weather-aware client can establish a connection by sending a connecting request to the CM.



Behavior 1: Client Initialization

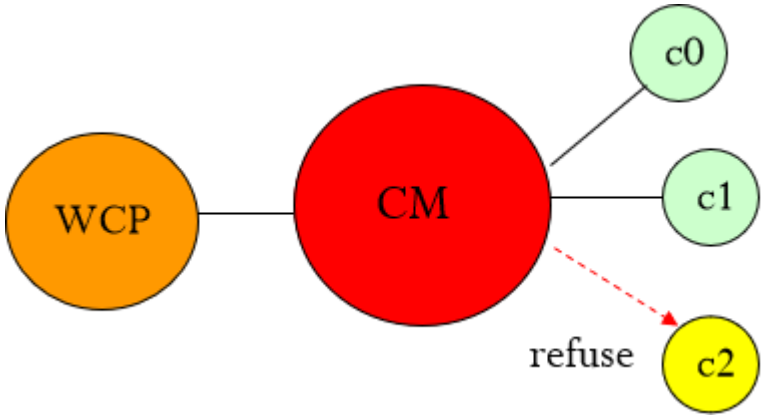
- If the CMs status is idle when the connecting request is received, set both its own status and the connecting clients status to **pre-initializing**, and **disable** the weather control panel.

- Send a message to instruct the newly connected client to **get the new weather information**, and then set both its own status and the clients status to **initializing**.



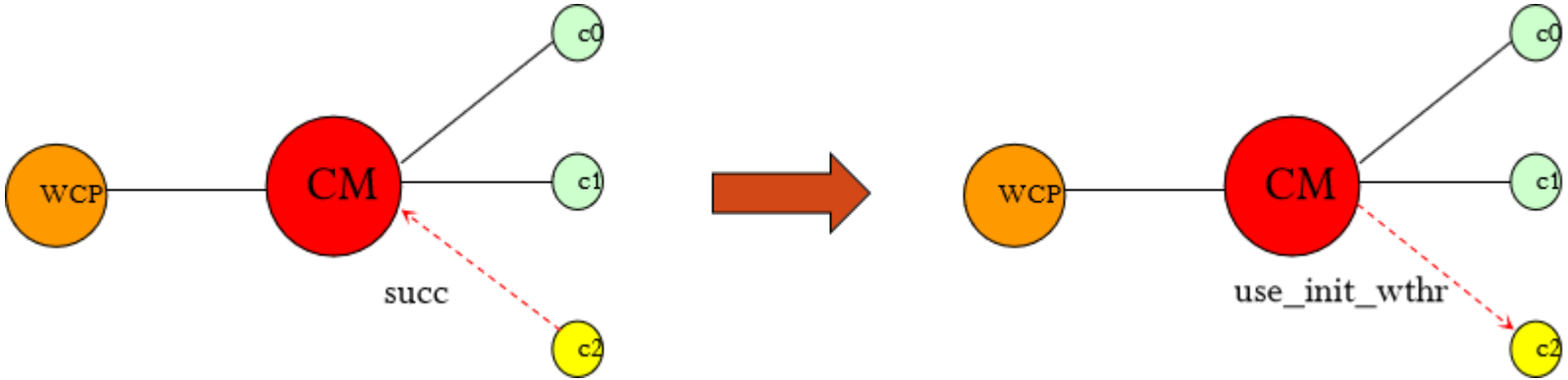
Behavior 1: Client Initialization

- If the CMs status is **not** idle.
- The CM will send a message to the client to **refuse** the connection, and the client remains **disconnected**.



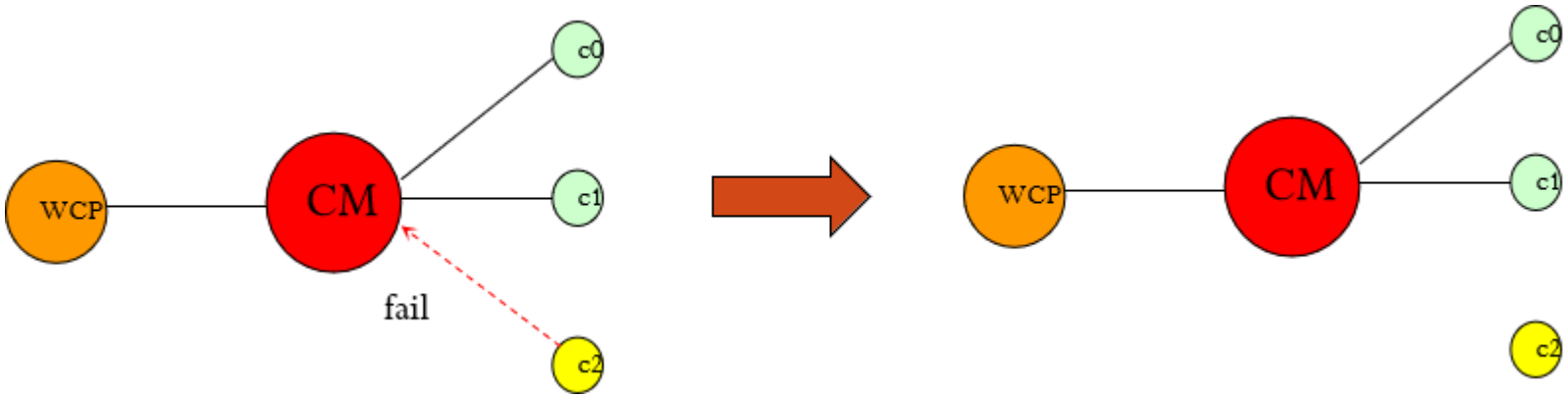
Behavior 1: Client Initialization

- If the client reports **success** for getting the new weather.
- The CM will send another message to inform the client to **use the weather information**, and then set both its own status and the clients status to **post-initializing**.



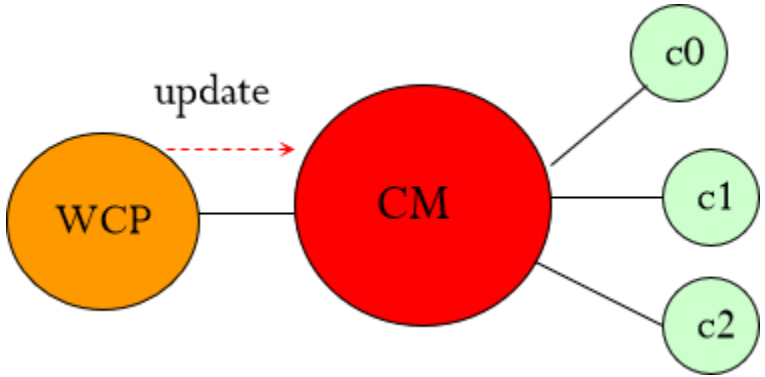
Behavior 1: Client Initialization

- If the client reports **fail** for getting the new weather.
- The CM will **disconnect** the client and set its own status back to **idle**.



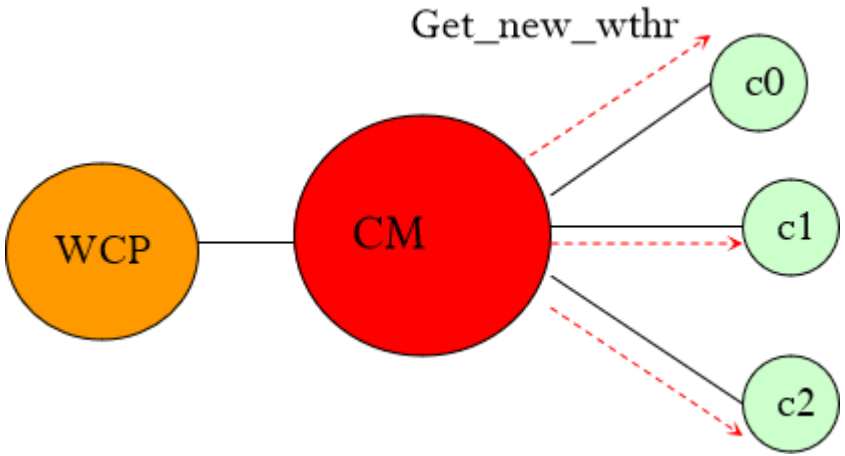
Behavior 2: Weather Update

- User can manually update new weather information only when the WCP is **enabled**. By clicking the update button on the WCP, a update message is sent to the CM.



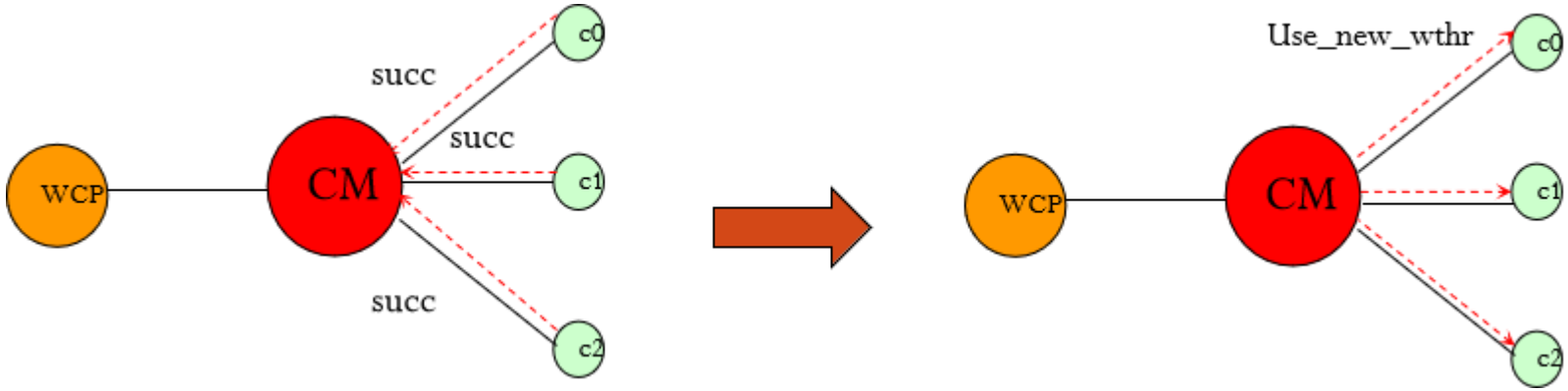
Behavior 2: Weather Update

- When the CM is **idle** and receives update request from the WCP, it will set its own status and all the connected weather-aware clients status to **pre-updating**.
- **Disable** the WCP from any further updating requests before the completion of current update.
- When CMs status is **pre-updating**, it will send messages to instruct all connected clients to **get the new weather** information, and then set its own status and the clients status to **updating**.



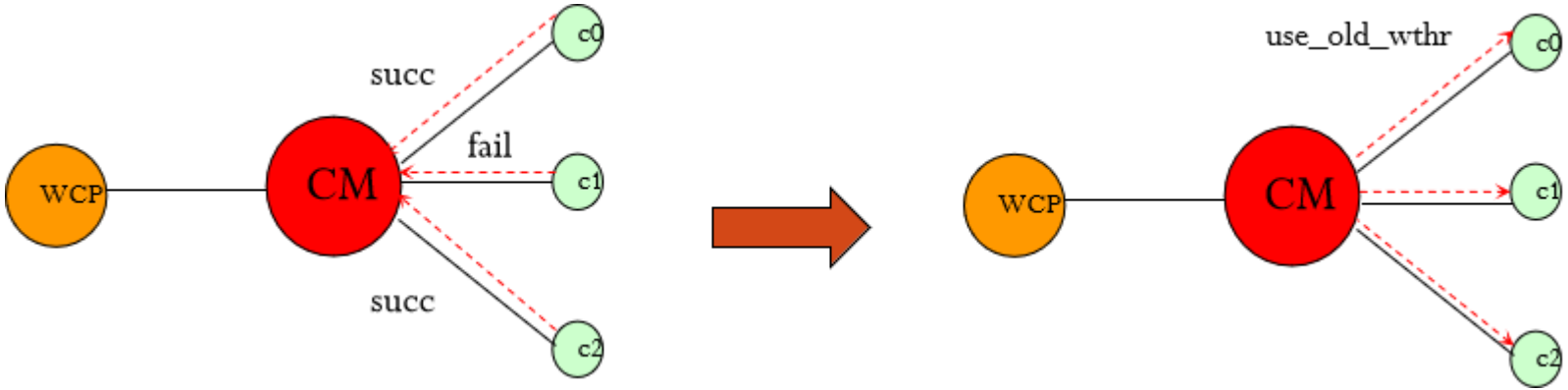
Behavior 2: Weather Update

- If **all** the clients report **success** for **getting the new weather**, the CM will send messages to inform the clients to **use the new weather information**, and then set its own status and the clients status to **post-updating**.



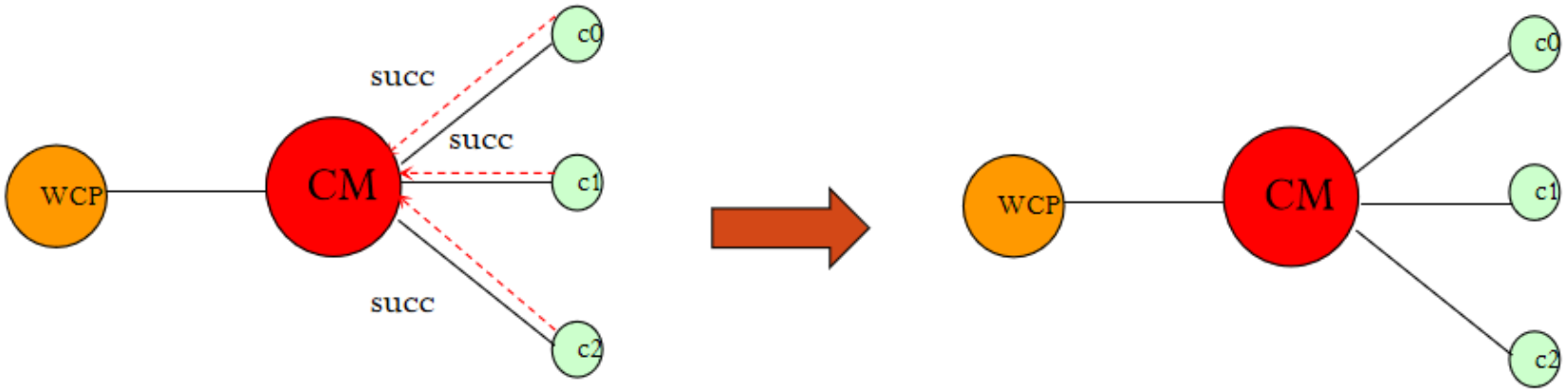
Behavior 2: Weather Update

- Otherwise, if **any** of the connected clients reports **failure for getting the new weather**, the CM will send messages to **all** clients to **use their old weather information**, and then set its own status and the clients status to **post-reverting**.



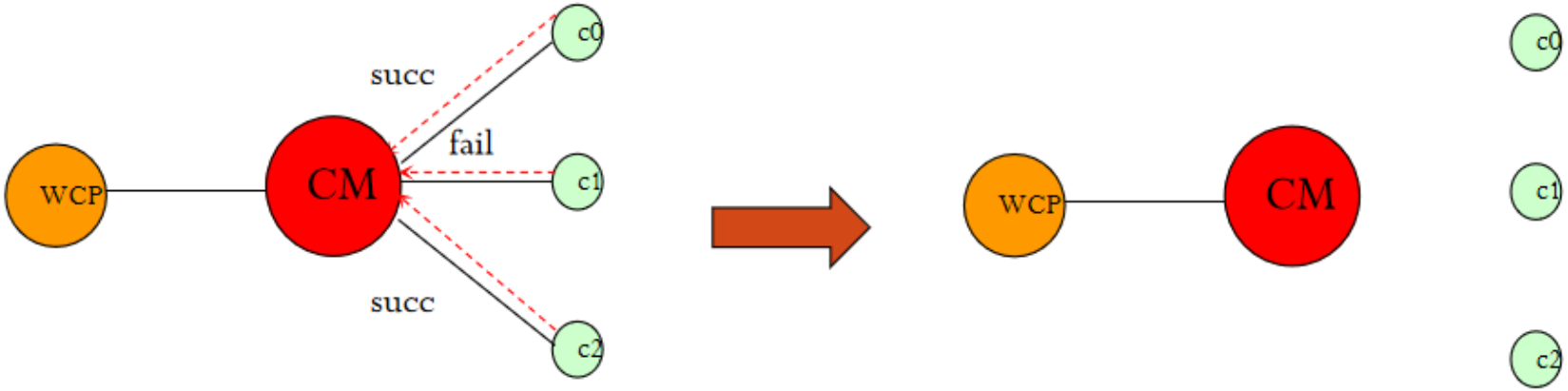
Behavior 2: Weather Update

- When CMs status is **post-updating**, if all the clients report **success for using the new weather**, the updating is **completed**. The CM will set its own status and the clients status to **idle**, and **re-enable** the WCP.



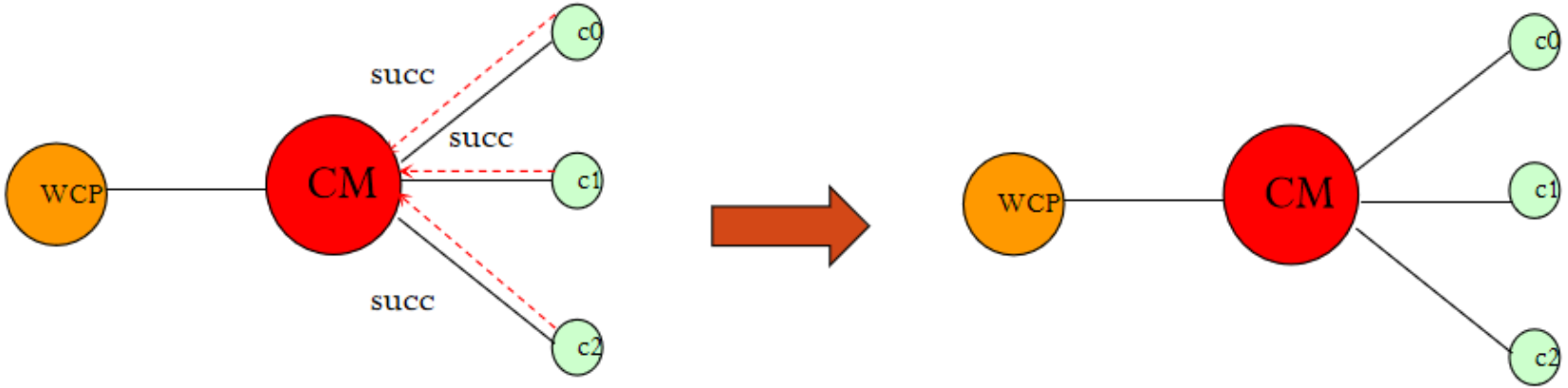
Behavior 2: Weather Update

- Otherwise, if any of the connected clients reports failure for using the new weather, the CM will disconnect all connected clients, re-enable the WCP, and set its own status back to idle.



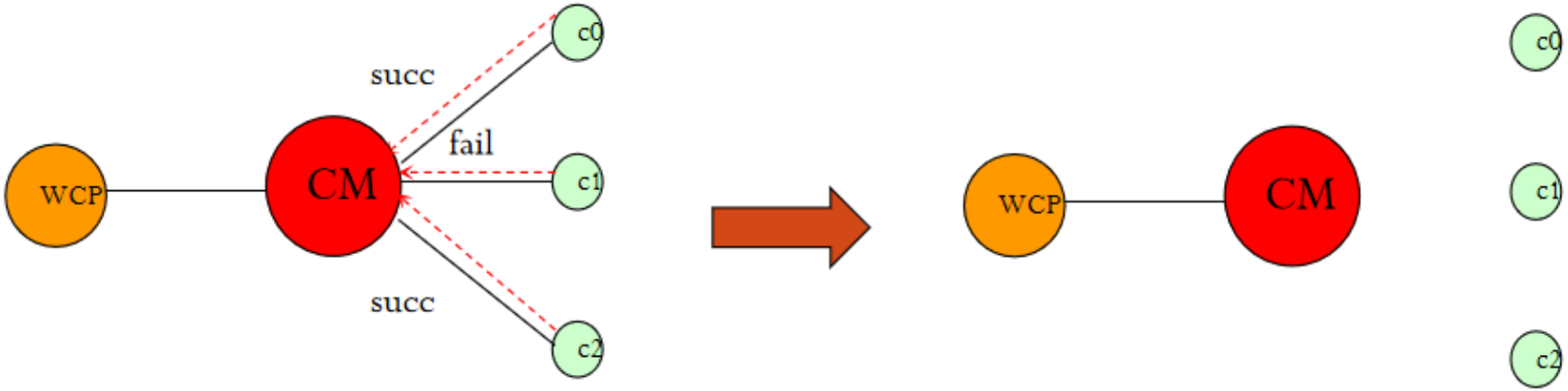
Behavior 2: Weather Update

- When CMs status is **post-reverting**, if all the clients report **success** for using the old weather, the reverting is **completed**. The CM will set its own status and the clients status to **idle**, and **re-enable** the WCP.



Behavior 2: Weather Update

- Otherwise, if **any** of the connected clients reports **failure for using the old weather**, the CM will disconnect all connected clients, **re-enable** the WCP, and set its own status back to **idle**.



Questions [11 marks]

- Assumption:
    - You can assume any fixed number of clients.
    - Initially the WCP is **enabled** for manual weather updating.
    - The CM is at **idle** state and all the clients are **disconnected.**
- Write a Promela model for the above controller.
    - Correct implementation of client initialization (**3 marks**)
    - Correct implementation of weather update (**3 marks**)

- The key property of this system is to be able to propagate the latest weather update to all connected clients via the communication manager. Define the key property in Linear-time temporal logic (LTL) (**1 mark**)

- Show that there exists a deadlock and provide a clear interpretation of the counter-example obtained from SPIN. Any additional problems you find in the protocol will of course distinguish your answer and earn more credit (**2 marks**)

- Implement a solution which makes the model deadlock free and verify the deadlock free property. (**2 marks**