Lab Report

Course Title: Computer Networks Laboratory
Course Code: CSE-3634

Spring-2023

Lab No: 2

Name of Labwork: Implement a network where nodes have different limit to forward the message. When limit reaches 0 the node will not forward the message and the simulation will stop.

TEAM_DIVERGENT

Student's ID : C201249
Date of Performance : 17/07/2023
Date of Submission : 23/07/2023

Submitted To : Mr. AbdullahilKafi
Assistant Professor

Mark :

1. Introduction:

The purpose of this lab experiment was to enhance the created simple module network of 5 nodes named (Lamisa, Sohana, Sadia, Marowa, Tahnia) by adding functions and changing icons and background color in OMNeT++.

2. Description:

Here we make the model look a bit prettier in the GUI by adding icons and color to the background. Then we record log by using **"record-eventlog"** in ini file. Then we add state variable like counter which will count the number of times the message will be sent and we limit it using **"limit"** function and we take the counter value from ini file using **{par("limit")}** function.

3. Module:

In this lab experiment simple module and compound module are used. In order to describe the operation of a simple module, one or more virtual member functions must be declared in a C++ class which has to be subclassed from **cSimpleModule**. By means of the **Define_Module()** macro, the class must be registered with OMNeT++. A compound module groups other modules into a larger unit. A compound module may have gates and parameters like a simple module, but no active behavior is associated with it. When there is a temptation to add code to a compound module, then encapsulate the code into a simple module, and add it as a submodule.

4. NED file:

```
simple node2_divergent
{
   parameters:
      int limit = default(5);
   gates:
      input in;
      output out;
}

network lab2_divergent
{
   @display("bgb=591.336,507.74402,#FFBDFF");
   submodules:
      lamisa: node2_divergent {
         @display("p=312.696,106.296005;i=abstract/person");
      }
      sadia: node2_divergent {
         @display("p=474.72,397.32;i=device/wifilaptop");
      }
      sohana: node2_divergent {
         @display("p=473.68802,234.264;i=device/pc4");
      }
      marowa: node2_divergent {
         @display("p=113.520004,397.32;i=device/pc2");
      }
      tahnia: node2_divergent {
```

```
        @display("p=113.520004,106.296005;i=device/pocketpc");
    }
  connections:
    lamisa.out --> {  delay = 100ms; } --> sohana.in;
    sadia.in <-- {  delay = 100ms; } <-- sohana.out;
    sadia.out --> {  delay = 100ms; } --> marowa.in;
    marowa.out --> {  delay = 100ms; } --> tahnia.in;
    lamisa.in <-- {  delay = 100ms; } <-- tahnia.out;
}
```

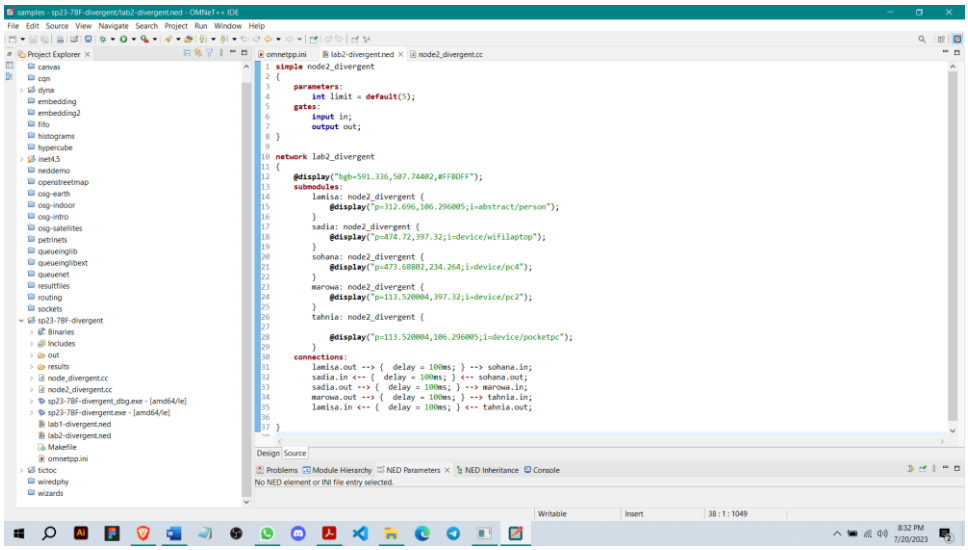

Fig 1: Snapshot of code in lab2-divergent.ned file

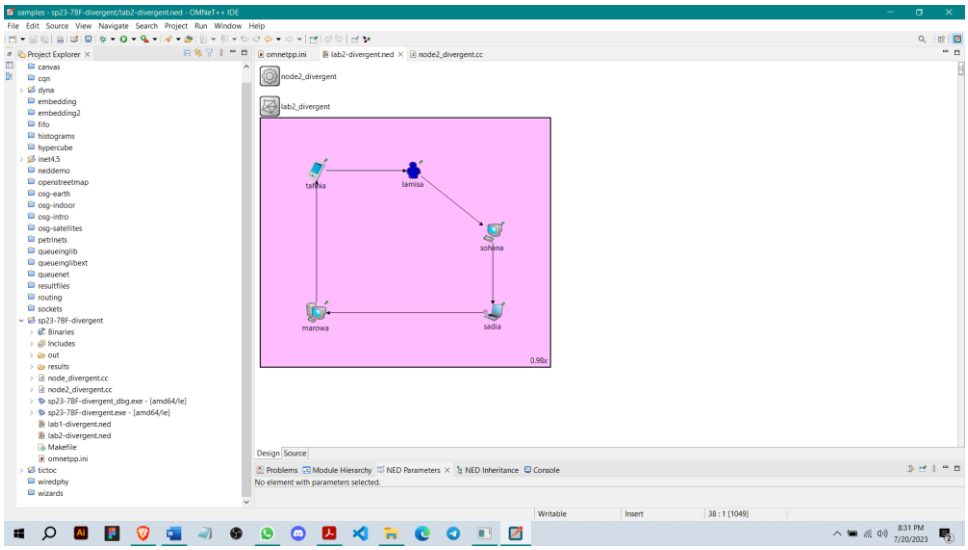

Fig 2: Snapshot of Design in lab2-divergent.ned file

5. CC file:

```
#include <string.h>
#include <omnetpp.h>
using namespace omnetpp;

class  node2_divergent : public cSimpleModule
{
private:
```

```
    int counter;
    protected:
    virtual void initialize() override;
    virtual void handleMessage(cMessage *msg) override;
};

// The module class needs to be registered with OMNeT++
Define_Module( node2_divergent);
void  node2_divergent::initialize()
{
    counter = par("limit");
    WATCH(counter);

    if (strcmp("lamisa", getName()) == 0) {
        cMessage *msg = new cMessage("lab2 Work Done");
        send(msg, "out");
    }
}

void  node2_divergent::handleMessage(cMessage *msg)

{
    counter--;
    if (counter == 0) {
            // If counter is zero, delete message. If you run the model, you'll
            // find that the simulation will stop at this point with the message
            // "no more events".
            EV << getName() << "'s counter reached zero, deleting message\n";
            delete msg;
        }
        else {
            EV << getName() << "'s counter is " << counter << ", sending back message\n";
            send(msg, "out");
        }
}
```

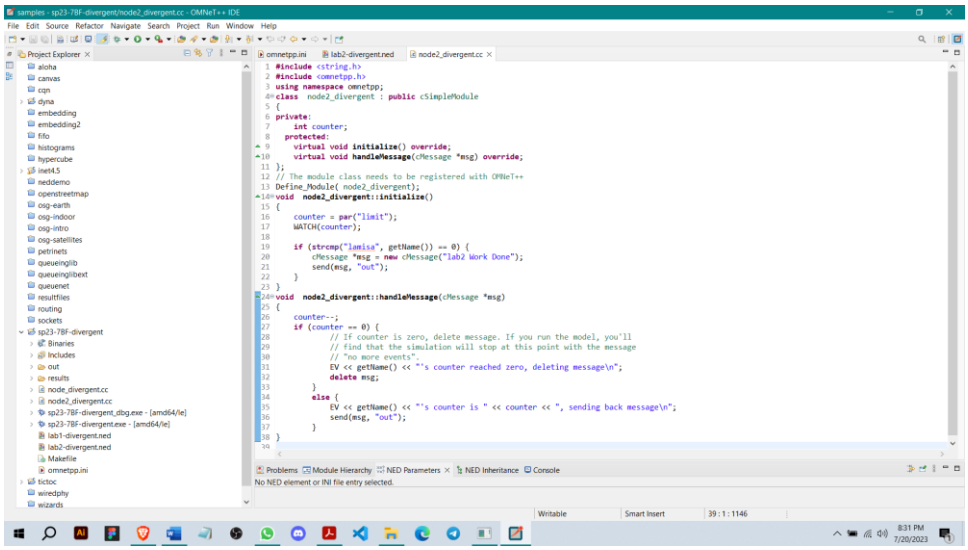

Fig 3: Snapshot of code in node2_divergent.cc file

7. INI File:

```
[General]

[Config lab1_divergent_exp1]

network = lab1_divergent
record-eventlog = true
lab2_divergent.lamisa.limit = 10
lab2_divergent.sohana.limit = 11
lab2_divergent.sadia.limit = 15
lab2_divergent.marowa.limit = 12
lab2_divergent.tahnia.limit = 14

[Config lab2_divergent_exp2]

network = lab2_divergent
record-eventlog = true
lab2_divergent.lamisa.limit = 11
lab2_divergent.sohana.limit = 15
lab2_divergent.sadia.limit = 16
lab2_divergent.marowa.limit = 12
lab2_divergent.tahnia.limit = 14
```

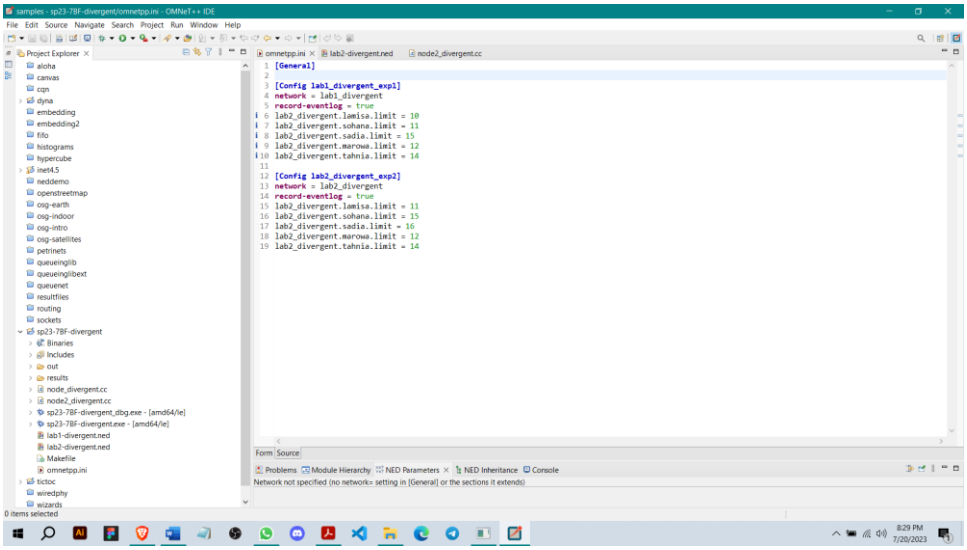

Fig 4: Snapshot of code in omnetpp.ini file

8. Build and Simulation:

Fig 5 demonstrates the scenario of building the project and running the simulation. Again, in Fig 6 the termination of the simulation is illustrated. Also, there is a deleted message shown.

After running the simulation result files will be created automatically in the project folder. Fig 7 demonstrates the result file lab2_divergent_exp2-#0.elog .

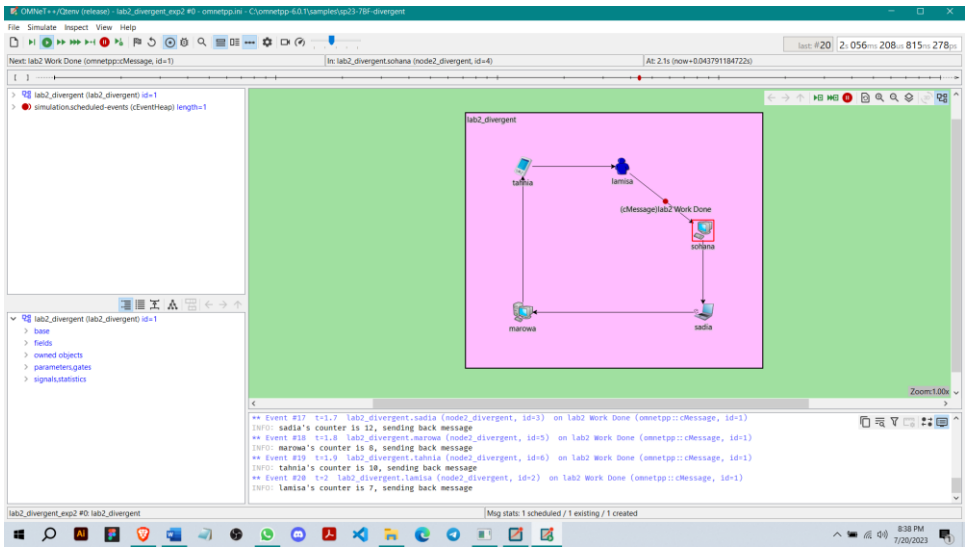

Fig 5: Building and Running

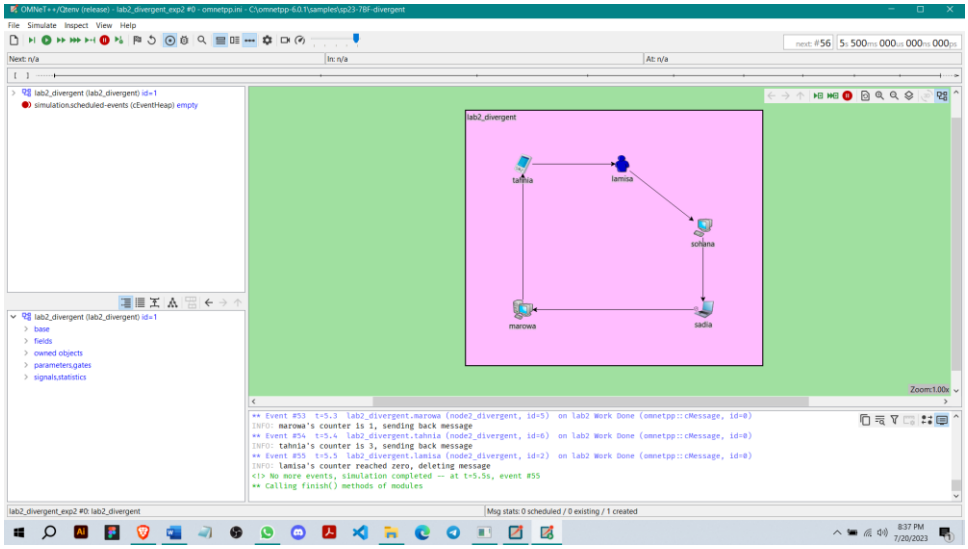

Fig 6: Terminate and deleting message

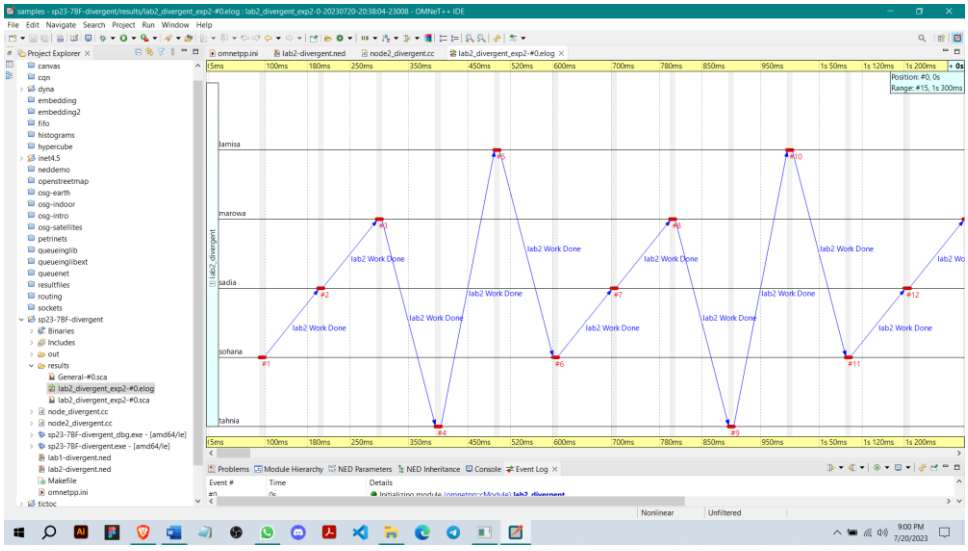

Fig 7: Snapshot of result (lab2_divergent_exp2-#0.elog)

9. Result Analysis:

The simulation of the module network successfully demonstrated the message circulation loop among the five nodes (Lamisa, Sohana, Sadia, Marowa, Tahnia) till the counter becomes zero and when it will become zero the message will be deleted. When limit reaches 0 the node will not forward the message and the simulation will stop. EV<< to directs the event info and limit value. Also record-eventlog = true in the ini file shows the sequence diagram.

6. Conclusion:

In conclusion, the simple network with 5 modules will show message in enhanced GUI form as we design and it will terminate when the counter value reaches 0 resulting deleting the message which is transmitted in circular loop throughout the network.