Lab Report

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

Spring-2023

Lab No: 3

Name of Labwork: Simulation of a network where the limit and destination node are assigned in the ini file. Nodes will forward the message randomly to one of the output gates. The simulation will stop when the message reaches its destination or when a node's limit hits zero.

TEAM_DIVERGENT

Student's ID : C201249

Date of Performance : 24/07/2023
Date of Submission : 29/07/2023

Submitted To : Mr. AbdullahilKafi
Assistant Professor

Mark :

1. Introduction:

In this lab work, we designed and simulated a network topology using OMNeT++. The network consisted of six nodes interconnected through custom channels. Each node was represented by the node3_divergent module, programmed to forward messages randomly to other nodes until they reached a specified destination.

2. Description:

In this lab work, we constructed a divergent network topology consisting of six nodes using OMNeT++. The network was designed to simulate a message transmission scenario, where messages are generated at a source node and passed through multiple intermediate nodes before reaching a predefined destination node. The objective was to investigate the dynamics of message transmission and routing in this network, where messages are generated at a source node and forwarded through multiple intermediate nodes before reaching a destination node.

3. Module:

The **node3_divergent** module is the key component of the simulation, representing the behavior of individual nodes in the divergent network. It is implemented as a C++ class that inherits from the **cSimpleModule** class provided by OMNeT++. The main purpose of the **node3_divergent** module is to simulate the behavior of nodes in a divergent network topology.

4. NED file:

```
simple node3_divergent
{
    parameters:
        int limit = default(5);
        int destination = default(3);
        @display("i=block/routing");
    gates:
        inout gate[];  // declare two way connections
}
network lab3_divergent
{
    @display("bgi=background/green,t");
    types:
        channel Channel extends ned.DelayChannel
        {
            delay = 100ms;
        }
    submodules:
        divergent[6]: node3_divergent {
            @display("i=device/pc2");
        }
    connections:
        divergent[4].gate++ <--> Channel <--> divergent[5].gate++;
        divergent[3].gate++ <--> Channel <--> divergent[4].gate++;
        divergent[1].gate++ <--> Channel <--> divergent[4].gate++;
        divergent[1].gate++ <--> Channel <--> divergent[2].gate++;
        divergent[0].gate++ <--> Channel <--> divergent[1].gate++;
}
```

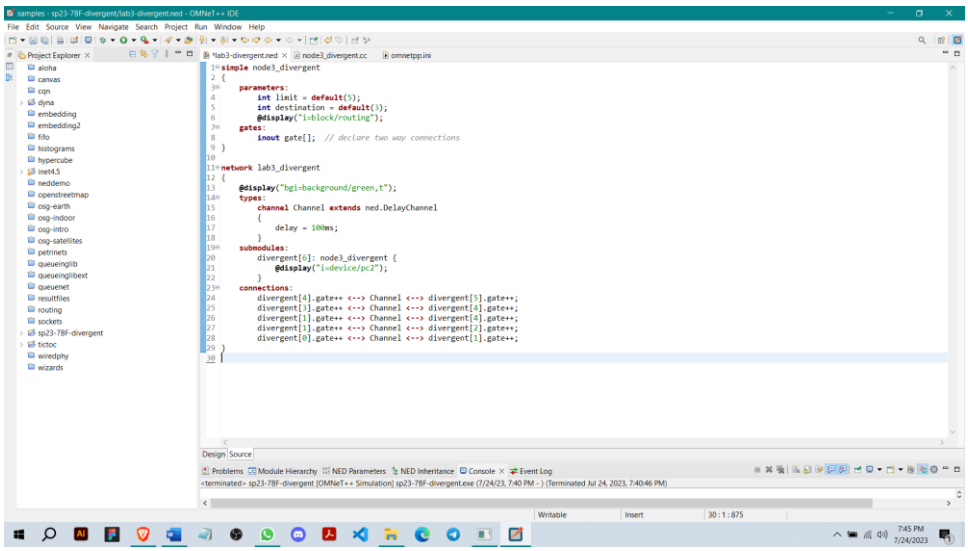

Fig 1: Snapshot of code in lab3-divergent.ned file

5. CC file:

```
#include <stdio.h>
#include <string.h>
#include <omnetpp.h>

using namespace omnetpp;

class node3_divergent: public cSimpleModule {
private:
   int counter;
   int dest;
protected:
   virtual void forwardMessage(cMessage *msg);
   virtual void initialize() override;
   virtual void handleMessage(cMessage *msg) override;
};
Define_Module(node3_divergent);

void node3_divergent::initialize() {
   counter = par("limit");
   dest = par("destination");

   if (getIndex() == 0) {
      // Boot the process scheduling the initial message as a self-message.
      char msgname[20];
      sprintf(msgname, "divergent-%d", getIndex());
      cMessage *msg = new cMessage(msgname);
      scheduleAt(0.0, msg);
   }
}
void node3_divergent::handleMessage(cMessage *msg) {
   counter--;
   if (getIndex() == dest) {
      // Message arrived.
```

```
        EV << "Message " << msg << " arrived.\n";
        delete msg;
    } else if (counter == 0) {
        EV << getName() << "'s counter reached zero, deleting message\n";
        delete msg;
    } else {
        forwardMessage(msg); // We need to forward the message.
    }
}
void node3_divergent::forwardMessage(cMessage *msg) {
    // In this example, we just pick a random gate to send it on.
    // We draw a random number between 0 and the size of gate `gate[]'.
    int n = gateSize("gate");
    int k = intuniform(0, n - 1);

    EV << "Forwarding message " << msg << " on gate[" << k << "]\n";
    // $o and $i suffix is used to identify the input/output part of a two way gate
    send(msg, "gate$o", k);
}
```

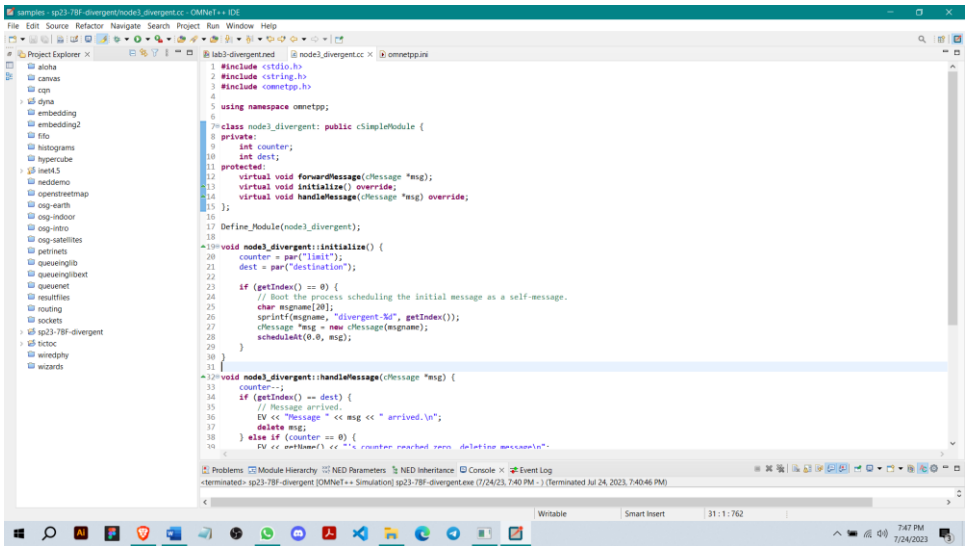

Fig 2: Snapshot of code in node2_divergent.cc file

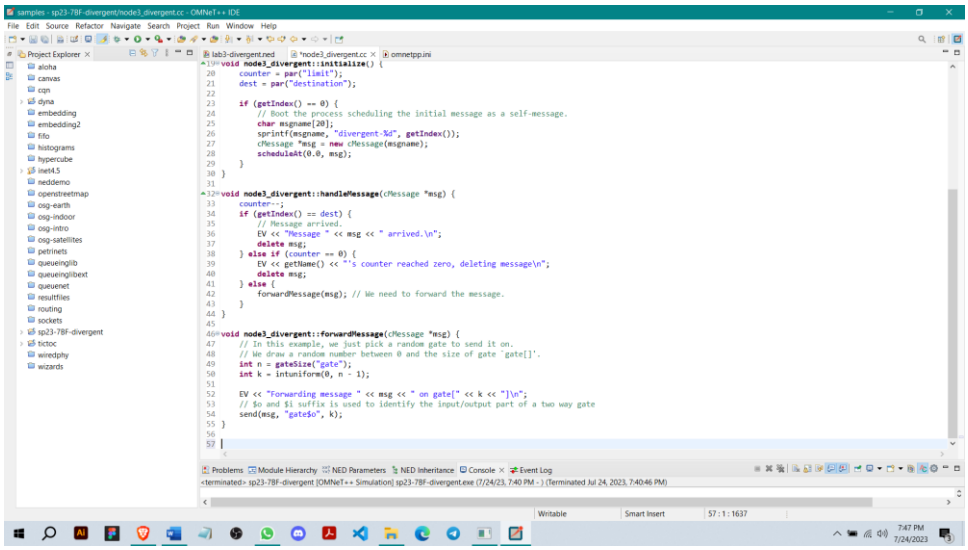

Fig 3: Snapshot of code in node2_divergent.cc file

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

[Config lab3_divergent_exp1]
network = lab3_divergent
record-eventlog = true
lab3_divergent.*.limit = 10
lab3_divergent.*.destination = 5

[Config lab3_divergent_exp2]
network = lab3_divergent
record-eventlog = true
lab3_divergent.*.limit = 4
lab3_divergent.*.destination = 5

[Config lab3_divergent_exp3]
network = lab3_divergent
record-eventlog = true
lab3_divergent.*.limit = 2
lab3_divergent.*.destination = 5

[Config lab3_divergent_exp4]
network = lab3_divergent
record-eventlog = true
lab3_divergent.*.limit = 4
lab3_divergent.*.destination = 0
```

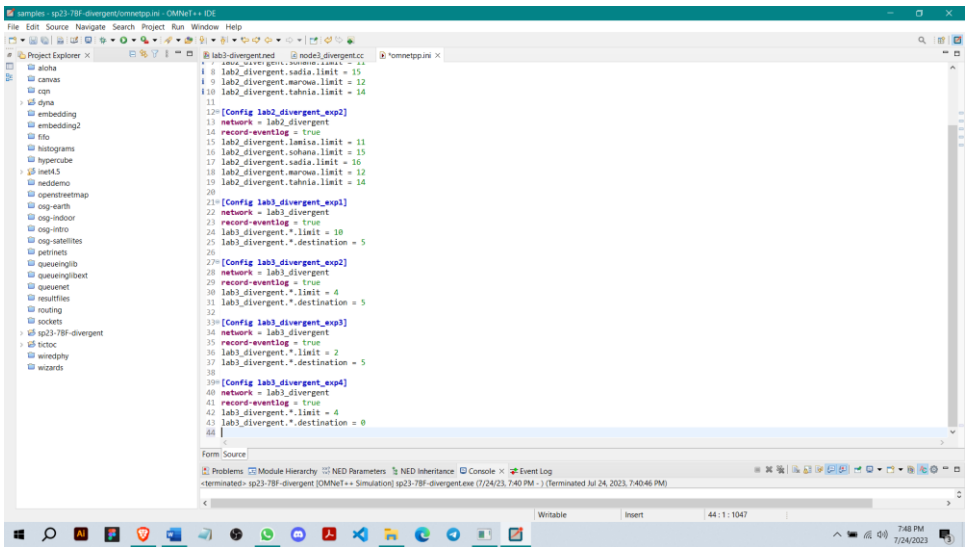

Fig 4: Snapshot of code in omnetpp.ini file

8. Build and Simulation:

By varying the forwarding limit (limit) and the destination index (dest), we can observe how these parameters affect the efficiency and reliability of message delivery in the network.

During the simulation, we varied the forwarding limits and destination nodes to observe their impact on network performance. We collected data on message delivery time, the number of hops taken by messages, and the percentage of successfully delivered messages.

Through the simulation, we can collect data on various metrics, such as message delivery time, the number of hops taken by messages, and the percentage of successfully delivered messages. This data can provide insights into the performance and behavior of the divergent network under different conditions.

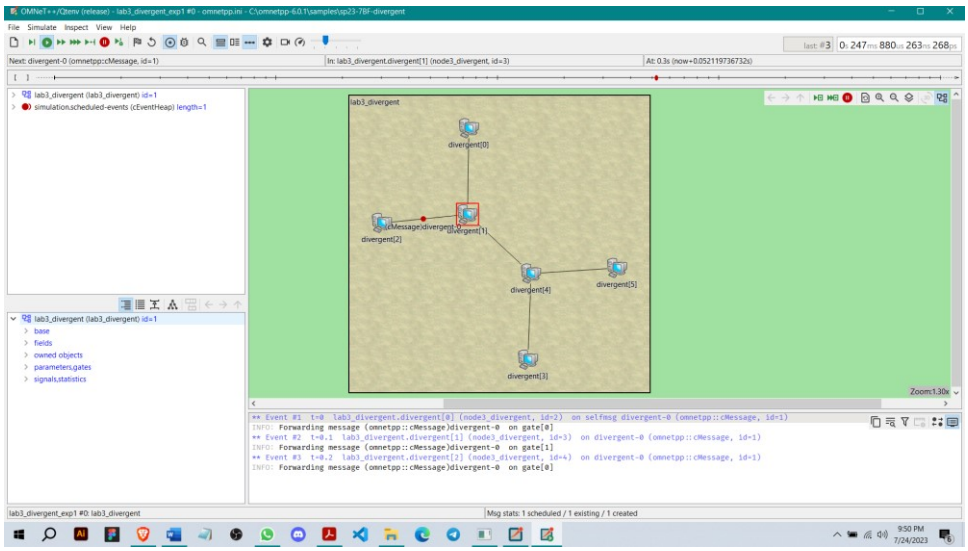

Fig 5: Building and Running simulation of config 1

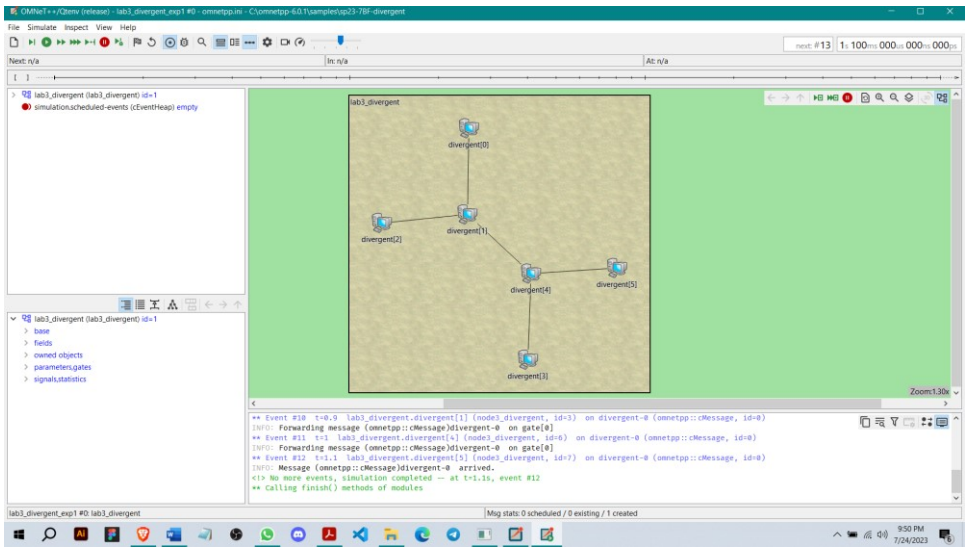

Fig 6: Snapshot of end of simulation for config 1

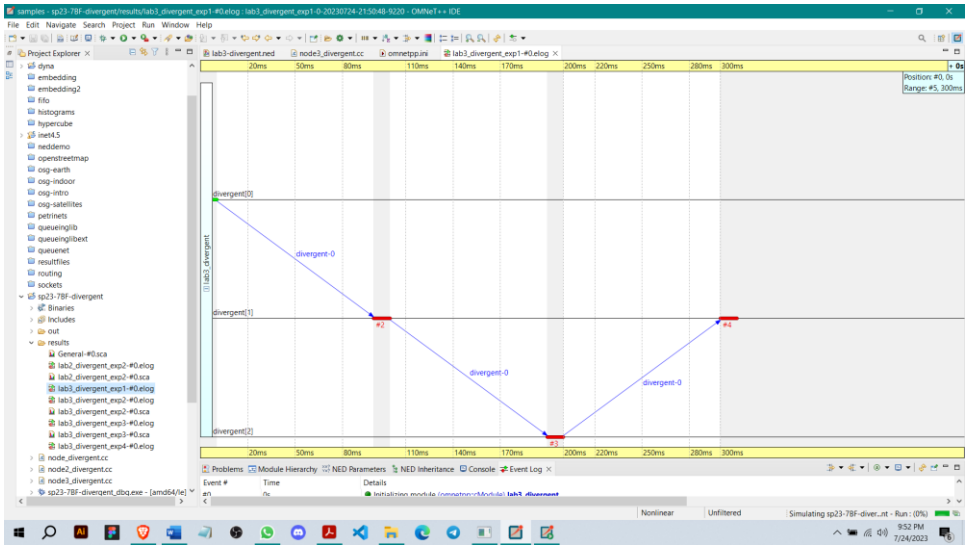

Fig 5: Snapshot of result (lab3_divergent_exp1-#0.elog)

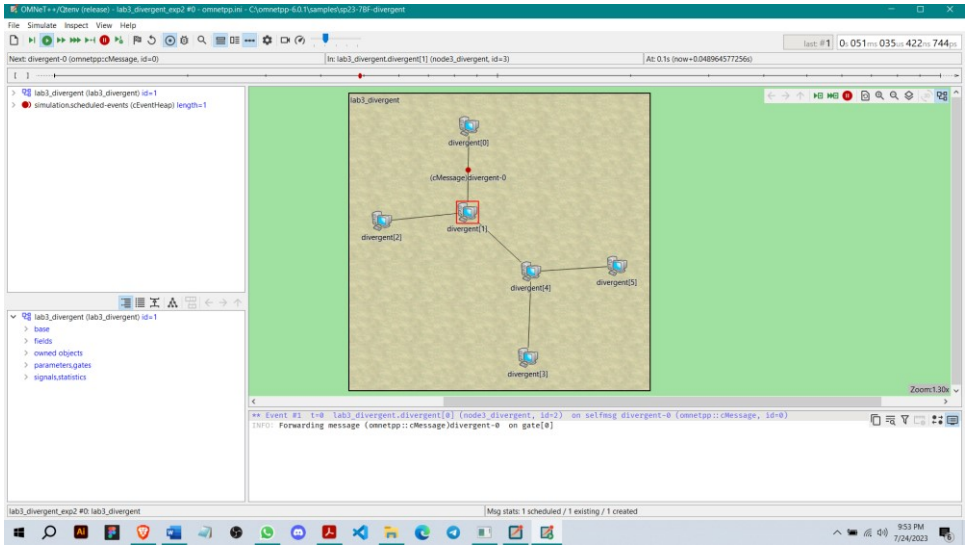

Fig 5: Building and Running config 2

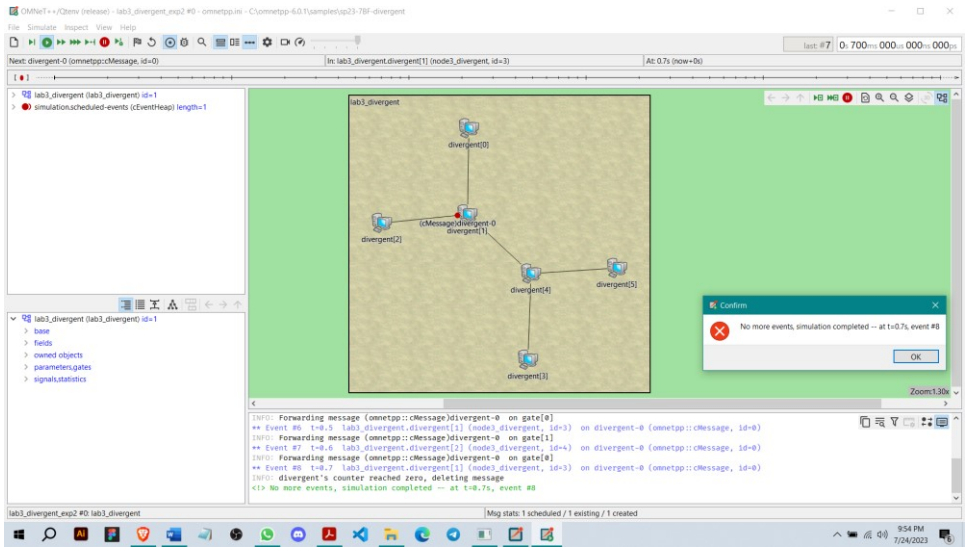

Fig 5: Snapshot of termination message for config 2

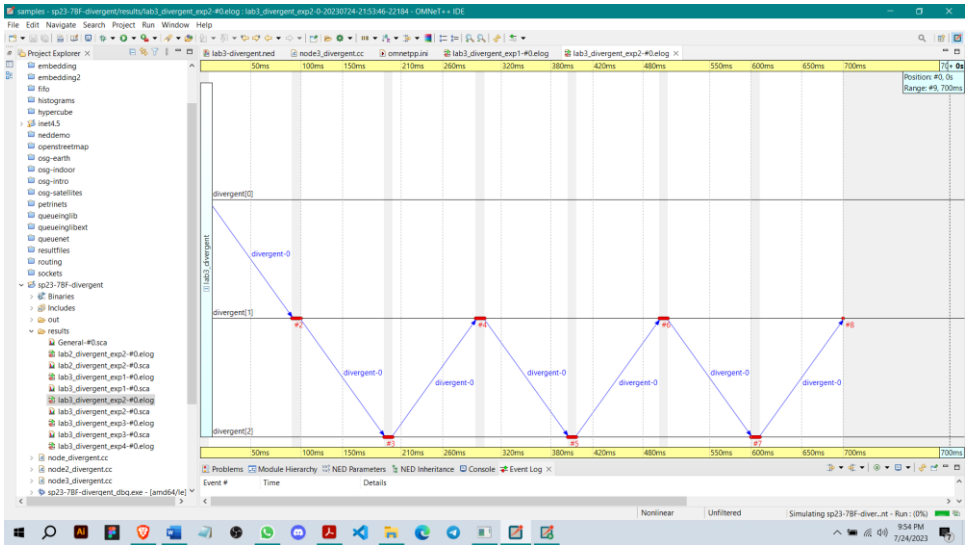

Fig 6: Snapshot of result (lab3_divergent_exp2-#0.elog)

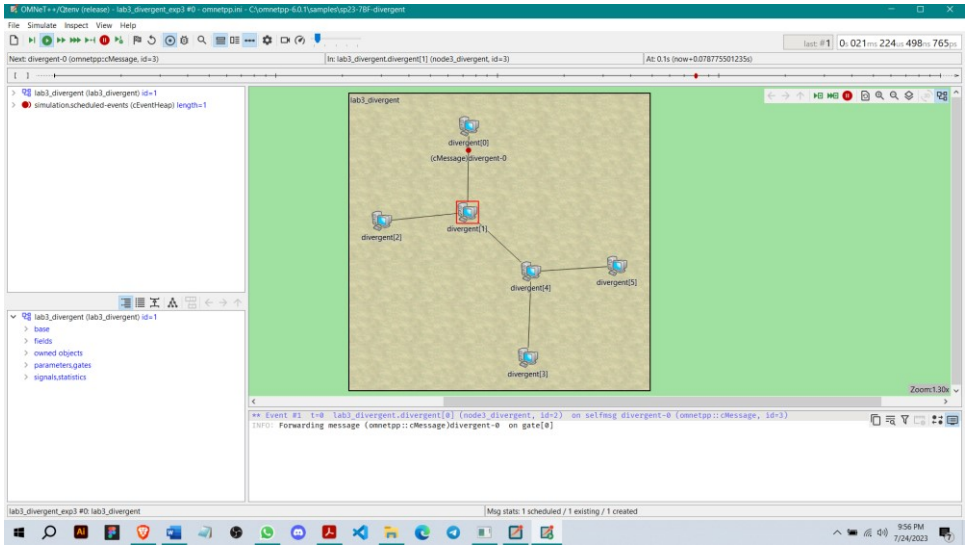

Fig 5: Building and Running config 3

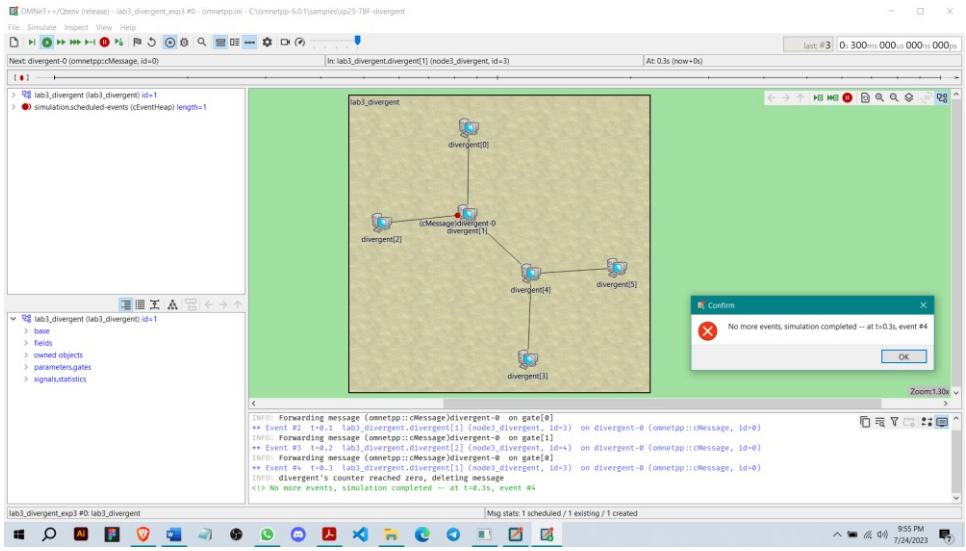

Fig 5: Snapshot of termination message for config 3

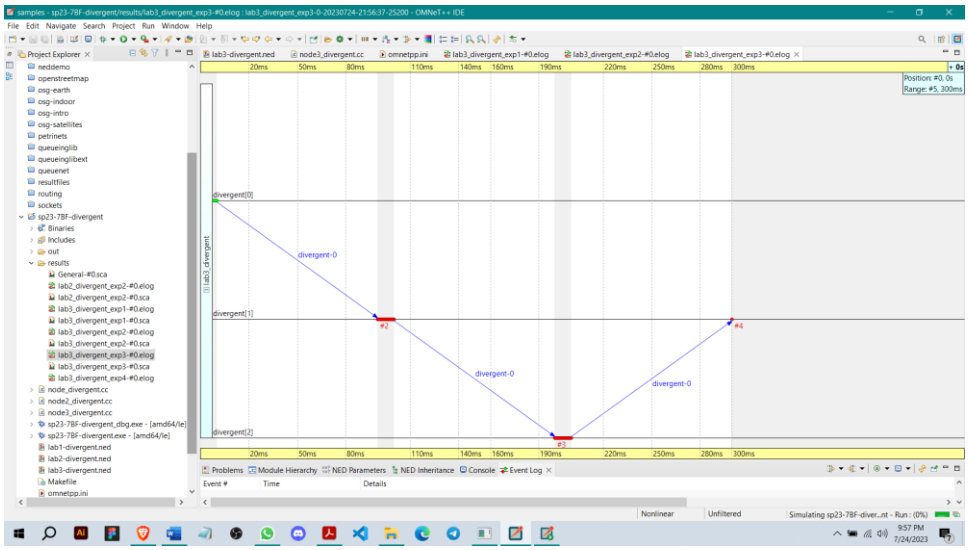

Fig 6: Snapshot of result (lab3_divergent_exp3-#0.elog)

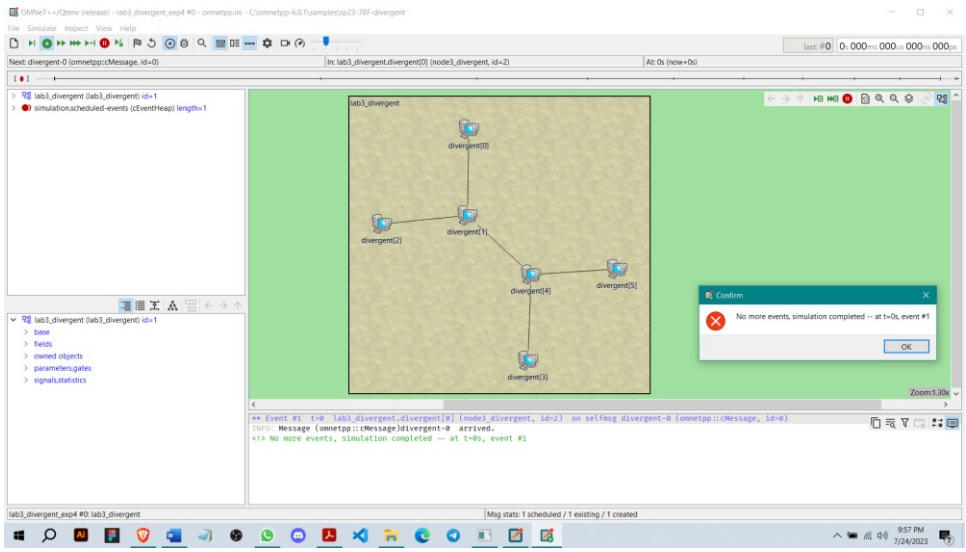

Fig 5: Building and Running and termination of config 4

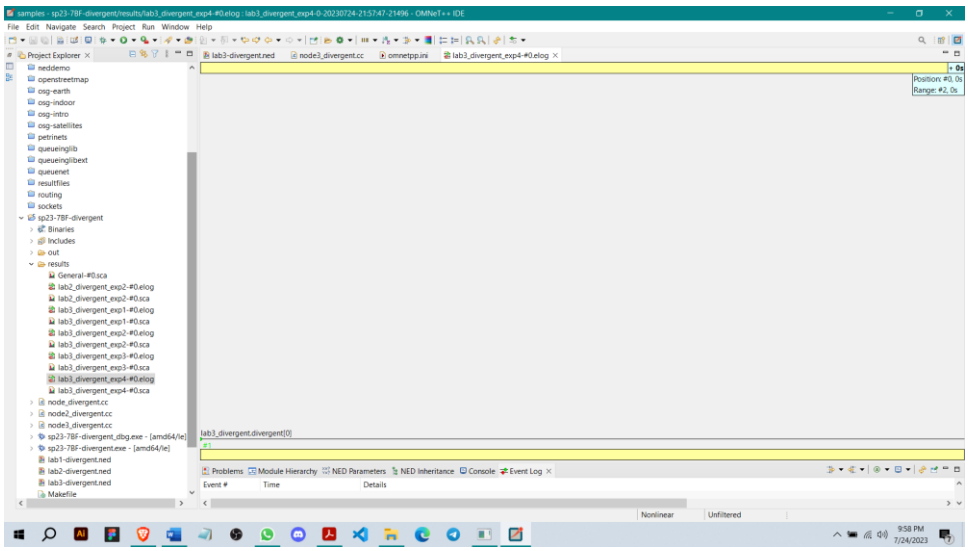

Fig 7: Snapshot of result (lab3_divergent_exp4-#0.elog)

9. Result Analysis:

The simulation results revealed interesting findings regarding message transmission in the divergent network topology. By varying the forwarding limit and destination node, we observed different patterns of message routing and delivery times. The network's efficiency was influenced by these parameters, as higher forwarding limits generally led to longer delivery times, while lower limits resulted in more dropped messages. The results and observations provide valuable insights for designing and optimizing divergent network topologies. Understanding how forwarding limits and destination nodes influence network performance enables us to make informed decisions when deploying and configuring communication networks.

6. Conclusion:

In conclusion, the implementation and simulation of the lab work provided a valuable learning experience in network simulation, demonstrating the principles of message routing in a **divergent** network using OMNeT++. The insights gained from the simulation can be applied to study and optimize communication networks, protocols, and algorithms.

Overall, the lab work deepened our understanding of network behavior and reinforced our skills in network modeling and analysis using OMNeT++.