Lab Report

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

Spring-2023

Lab No: 5

Name of Labwork: Creating own message type that has fields source, destination and hop count. Show in a table the hope counts for various set of source and destination. Find the average hop count.

TEAM_DIVERGENT

Student's ID : C201249

Date of Performance : 31/07/2023
Date of Submission : 13/08/2023

Submitted To : Mr. AbdullahilKafi
Assistant Professor

Mark :

1. Introduction:

In this lab work, we utilized OMNeT++, a powerful network simulation framework, to study and analyze the performance of a divergent communication network scenario. The network consists of six nodes connected through delay channels, representing a typical divergent topology. Each node can send and receive data packets using a custom message format named frameDivergent. The objective of this lab is to investigate the behavior of the network under various conditions and evaluate its performance in terms of packet delivery and hop count.

2. Description:

The lab work involves simulating a divergent communication network using OMNeT++. The network comprises six nodes, labeled as node5_divergent, each equipped with communication interfaces. The nodes are interconnected using delay channels (Channel), which introduce a 100ms delay in packet transmission. The communication among nodes in the network is represented using the custom message format frameDivergent. The "node5_divergent" module is equipped with communication interfaces (gates) through which it can send and receive messages. It employs a simple routing algorithm to forward messages to other nodes based on random gate selection.

3. Module:

The **node5_divergent** module is the key component of the simulation, representing the behavior of individual nodes in the divergent network. This module represents the nodes within the network. Each "node5_divergent" module is responsible for generating, forwarding, and receiving data packets (messages) in the network. It implements the custom message format frameDivergent to carry information such as the source, destination, and hop count of the messages

4. NED file:

```
simple node5_divergent {
   parameters:
         int source = default(0);
         int destination = default(5);
   gates:
      inout gate[];
}
network lab5_divergent
{
   @display("bgi=background/terrain");
   types:
      channel Channel extends ned.DelayChannel
      {
         delay = 100ms;
      }
   submodules:
      divergent[6]: node5_divergent {
         @display("i=old/comp");
      }
   connections:
      divergent[0].gate++ <--> Channel <--> divergent[1].gate++;
      divergent[1].gate++ <--> Channel <--> divergent[2].gate++;
      divergent[1].gate++ <--> Channel <--> divergent[4].gate++;
      divergent[3].gate++ <--> Channel <--> divergent[4].gate++;
      divergent[4].gate++ <--> Channel <--> divergent[5].gate++;
```

}

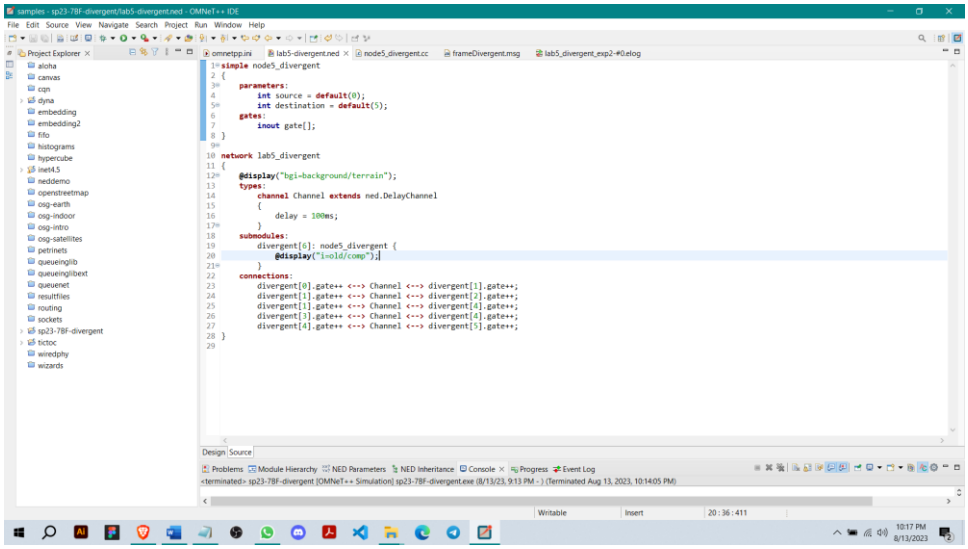

Fig 1: Snapshot of code in lab5-divergent.ned file

5. CC file:

```
#include <stdio.h>
#include <string.h>
#include <omnetpp.h>
using namespace omnetpp;
#include "frameDivergent_m.h"
class node5_divergent : public cSimpleModule
{
  protected:
    virtual frameDivergent *generateMessage();
    virtual void forwardMessage(frameDivergent *msg);
    virtual void initialize() override;
    virtual void handleMessage(cMessage *msg) override;
};

Define_Module(node5_divergent);
void node5_divergent::initialize()
{
    // Module 0 sends the first message
    if (getIndex() == 0) {
        frameDivergent *msg = generateMessage();
        scheduleAt(0.0, msg);
    }
}
void node5_divergent::handleMessage(cMessage *msg)
{
    frameDivergent *ttmsg = check_and_cast<frameDivergent *>(msg);

    if (ttmsg->getDestination() == getIndex()) {
        EV << "Message " << ttmsg << " arrived after " << ttmsg->getHopCount() << "
hops.\n";
        bubble("ARRIVED, starting new one!");
        delete ttmsg;
```

```
        EV << "Generating another message: ";
        frameDivergent *newmsg = generateMessage();
        EV << newmsg << endl;
        forwardMessage(newmsg);
    }
    else {
        forwardMessage(ttmsg);
    }
}
frameDivergent *node5_divergent::generateMessage()
{
    // Produce source and destination addresses.
    int src = getIndex();  // our module index
    int n = getVectorSize();  // module vector size
    int dest = intuniform(0, n-2);
    if (dest >= src)
        dest++;

    char msgname[20];
    sprintf(msgname, "divergent-%d-to-%d", src, dest);
    // Create message object and set source and destination field.
    frameDivergent *msg = new frameDivergent(msgname);
    msg->setSource(src);
    msg->setDestination(dest);
    return msg;
}
void node5_divergent::forwardMessage(frameDivergent *msg)
{
    // Increment hop count.
    msg->setHopCount(msg->getHopCount()+1);
    // Same routing as before: random gate.
    int n = gateSize("gate");
    int k = intuniform(0, n-1);
    EV << "Forwarding message " << msg << " on gate[" << k << "]\n";
    send(msg, "gate$o", k);
}
```

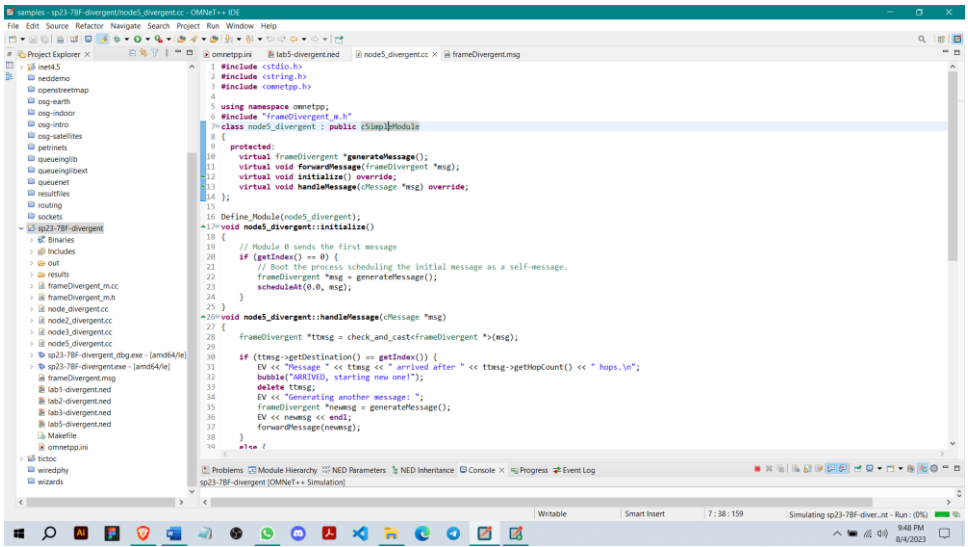

Fig 2: Snapshot of code in node2_divergent.cc file

7. MSG File:

```
message frameDivergent
{
    int source;
    int destination;
    int hopCount = 0;
}
```

8. INI File: ***(only for LAB5)***

```
[General]

[Config lab5_divergent_exp1]
network = lab5_divergent
record-eventlog = true
lab5_divergent.divergent[*].source = 1
lab5_divergent.divergent[*].destination = 5

[Config lab5_divergent_exp2]
network = lab5_divergent
record-eventlog = true
lab5_divergent.divergent[*].source = 5
lab5_divergent.divergent[*].destination = 0
```

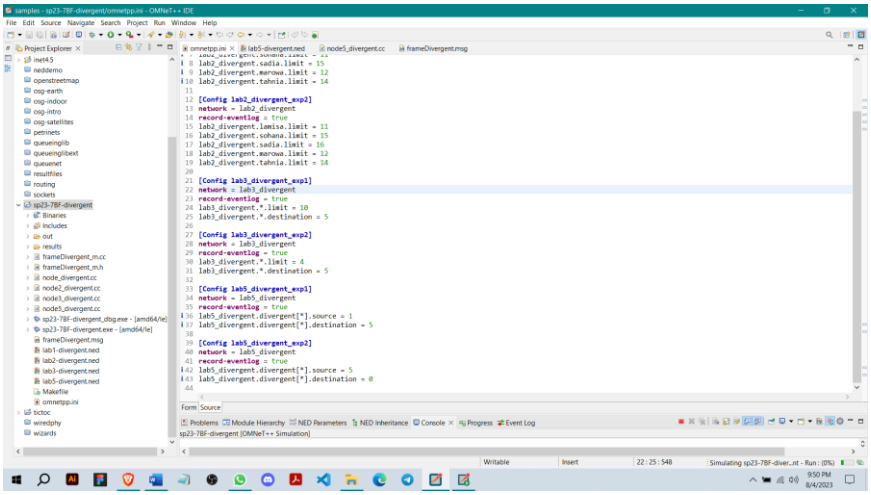

Fig 3: Snapshot of code in omnetpp.ini file

9. Build and Simulation:

During the simulation, we collect data on packet delivery and hop count for each message transmission in the divergent network. The simulation runs for a defined period, and multiple messages are generated and forwarded throughout the network.

The key aspects we investigate during the simulation include:
Packet Delivery: We analyze the percentage of successfully delivered messages in the network to assess its reliability under various conditions.
Hop Count: We observe the average hop count for messages transmitted through the network. A lower hop count indicates efficient routing, minimizing delays and resource consumption.

Message Routing: We examine the routing behavior of the network by tracing the paths taken by individual messages from the source to the destination.

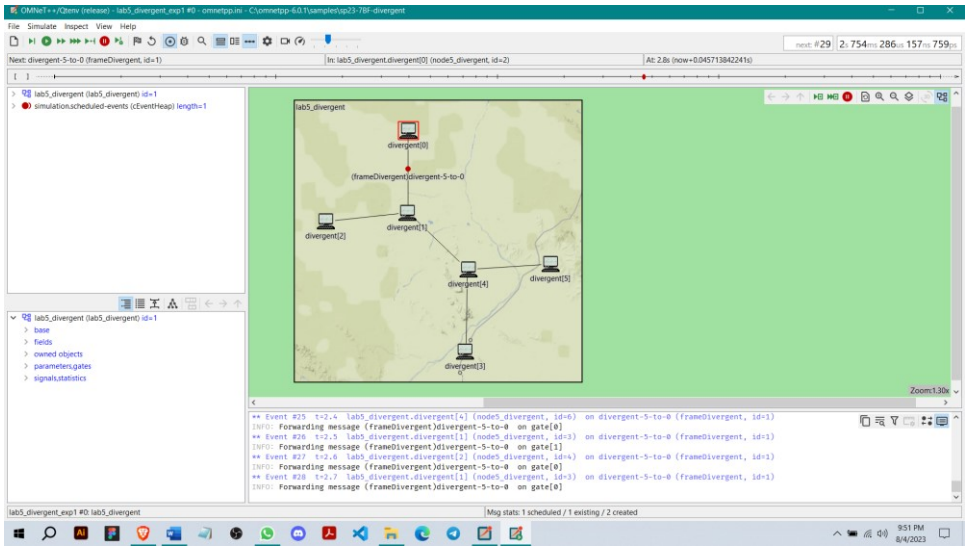

Fig 4: Building and Running simulation of lab5_divergent_exp1

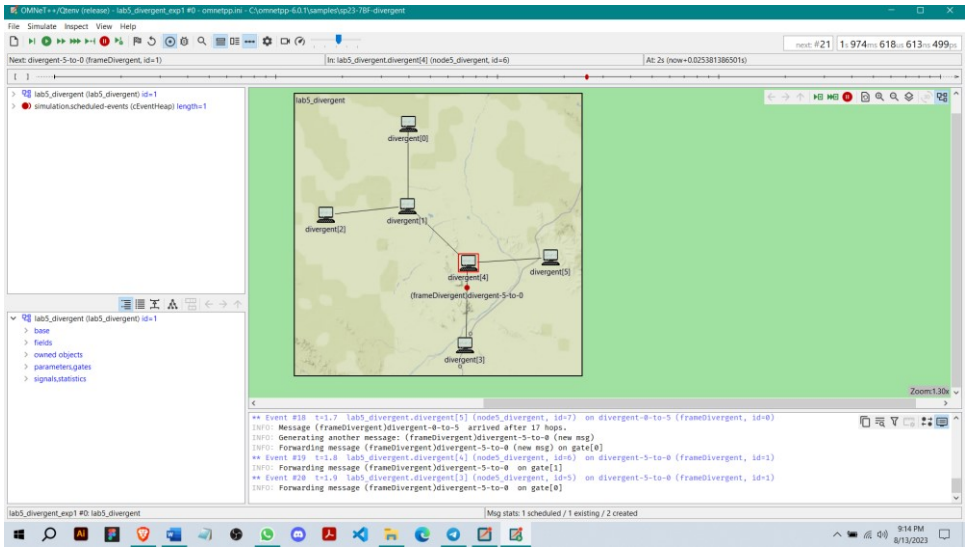

Fig 5: Snapshot of **ARRIVED** message for lab5_divergent_exp1

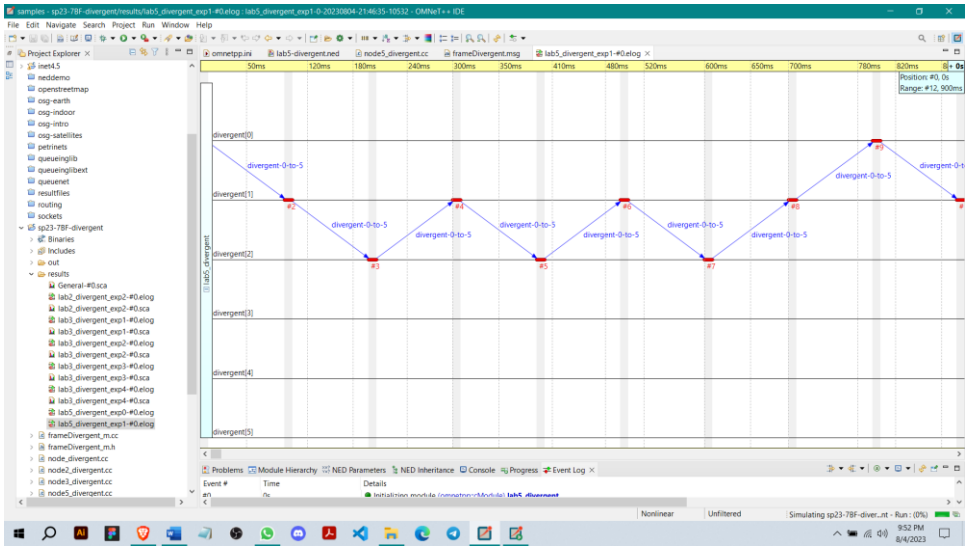

Fig 6: Snapshot of result (lab5_divergent_exp1-#0.elog)

10. Result Analysis:

The simulation starts with node 0 sending the first message to another random node in the network. When a node receives a message, it checks if it is the intended destination. If so, the node increments the hop count, logs the arrival of the message, and generates a new message to forward to another random node in the network. If the message is not destined for the current node, it forwards the message to another random node in its vicinity.

The experiment calculated the quantity of time and the maximum number of events required to send the message and complete all network cycles. Here, the message was sent from "divergent[0]" and then "hopped" between each node. The total amount of time it took for the message to arrive to "divergent[5]" was also computed, and the same procedure was used to determine the hop count of other nodes as well.

Here is a table for several hop count from different source and destination:

Source	Destination	Number of Hops
Divergent[0]	Divergent[5]	**17**
Divergent[5]	Divergent[0]	**11**
Divergent[0]	Divergent[3]	**45**
Divergent[3]	Divergent[5]	**2**
Divergent[5]	Divergent[4]	**1**
Divergent[4]	Divergent[5]	**29**
Divergent[5]	Divergent[0]	**31**
Divergent[0]	Divergent[1]	**1**
Divergent[1]	Divergent[4]	**1**
Divergent[4]	Divergent[2]	**4**

The average hop count for this simulation is = (17+11+45+2+1+29+31+1+1+4) / 10
= 142 / 10
= 14.2
≈ **14**

Therefore, **Average Hop Count** for the three above experiments is **14**.

11. Conclusion:

In conclusion, this lab work in OMNeT++ deepens our understanding of divergent communication networks and demonstrates the utility of simulation tools for network analysis and optimization. The knowledge gained from this study is valuable for designing and optimizing real-world communication networks, ensuring efficient data delivery and reliable performance.