Lab Report

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

Spring-2023

Lab No: 6

Name of Labwork: Statistics collection and plotting the random routing network consisting of 10 nodes.

TEAM_DIVERGENT

Student's ID	:	C201249
Date of Performance	:	07/08/2023
Date of Submission	:	06/08/2023

Submitted To	:	Mr. AbdullahilKafi Assistant Professor
Mark	:	

1. Introduction:

The experiment is conducted using OMNeT++, a versatile discrete event simulation framework widely employed for modeling and analyzing various network scenarios. The provided codes consist of NED (Network Description) and C++ files, which together define the simulation setup, network topology, node behavior, and communication protocols. This lab aims to demonstrate the impact of divergent routing on hop counts in a simulated network environment using the OMNeT++ simulation framework.

2. Description:

In this lab work, we will present the setup of the divergent routing simulation, the analysis of hop counts, and the interpretation of the obtained results. By evaluating the impact of the divergent routing strategy on hop counts, we aim to gain insights into the trade-offs between randomness and efficiency in packet delivery within a network. The lab focuses on a network employing random routing with **10 nodes**, allowing to explore the impact of random forwarding on packet hop counts and gain insights into statistical analysis and visualization techniques. By simulating a network with 10 nodes employing random routing, students gain hands-on experience in collecting various types of data and generating informative charts. The focus of the lab is to analyze the mean hop count, utilizing scalar, histogram, and vector data, and interpreting the insights drawn from the analysis.

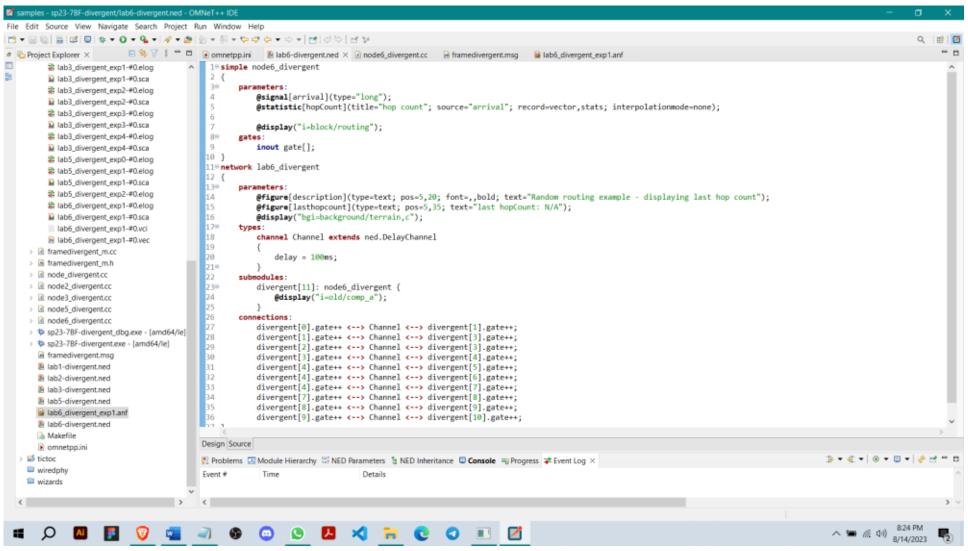
3. Module:

The **node6_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 "node6_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 node6_divergent
{
   parameters:
      @signal[arrival](type="long");
      @statistic[hopCount](title="hop count"; source="arrival"; record=vector,stats;
interpolationmode=none);

      @display("i=block/routing");
   gates:
      inout gate[];
}

network lab6_divergent
{
   parameters:
      @figure[description](type=text; pos=5,20; font=,,bold; text="Random routing example -
displaying last hop count");
      @figure[lasthopcount](type=text; pos=5,35; text="last hopCount: N/A");
      @display("bgi=background/terrain,c");
   types:
      channel Channel extends ned.DelayChannel
      {
         delay = 100ms;
```

```
        }
    submodules:
        divergent[11]: node6_divergent {
            @display("i=old/comp_a");
        }
    connections:
        divergent[0].gate++ <--> Channel <--> divergent[1].gate++;
        divergent[1].gate++ <--> Channel <--> divergent[3].gate++;
        divergent[2].gate++ <--> Channel <--> divergent[3].gate++;
        divergent[3].gate++ <--> Channel <--> divergent[4].gate++;
        divergent[4].gate++ <--> Channel <--> divergent[5].gate++;
        divergent[4].gate++ <--> Channel <--> divergent[6].gate++;
        divergent[4].gate++ <--> Channel <--> divergent[7].gate++;
        divergent[7].gate++ <--> Channel <--> divergent[8].gate++;
        divergent[8].gate++ <--> Channel <--> divergent[9].gate++;
        divergent[9].gate++ <--> Channel <--> divergent[10].gate++;
}
```



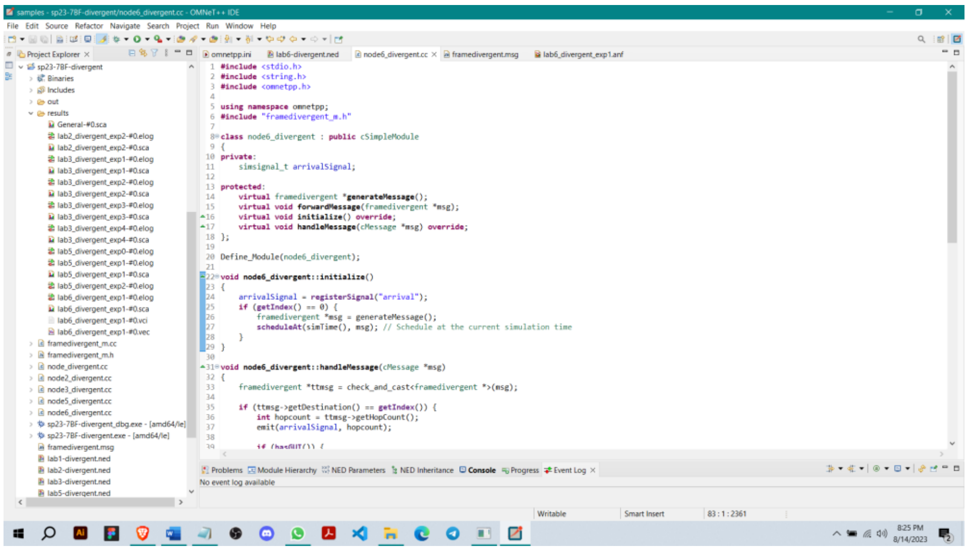
```
simple node6_divergent
{
    parameters:
        @signal[arrival](type="long");
        @statistic[hopCount](title="hop count"; source="arrival"; record=vector,stats; interpolationmode=none);

        @display("i=block/routing");
    gates:
        inout gate[];
}
network lab6_divergent
{
    parameters:
        @figure[description](type=text; pos=5,20; font=,,bold; text="Random routing example - displaying last hop count");
        @figure[lasthopcount](type=text; pos=5,35; text="last hopCount: N/A");
        @display("bgi=background/terrain,c");
    types:
        channel Channel extends ned.DelayChannel
        {
            delay = 100ms;
        }
    submodules:
        divergent[11]: node6_divergent {
            @display("i=old/comp_a");
        }
    connections:
        divergent[0].gate++ <--> Channel <--> divergent[1].gate++;
        divergent[1].gate++ <--> Channel <--> divergent[3].gate++;
        divergent[2].gate++ <--> Channel <--> divergent[3].gate++;
        divergent[3].gate++ <--> Channel <--> divergent[4].gate++;
        divergent[4].gate++ <--> Channel <--> divergent[5].gate++;
        divergent[4].gate++ <--> Channel <--> divergent[6].gate++;
        divergent[4].gate++ <--> Channel <--> divergent[7].gate++;
        divergent[7].gate++ <--> Channel <--> divergent[8].gate++;
        divergent[8].gate++ <--> Channel <--> divergent[9].gate++;
        divergent[9].gate++ <--> Channel <--> divergent[10].gate++;
```

Fig 1: Snapshot of code in lab6-divergent.ned file

5. CC file:

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

using namespace omnetpp;
#include "framedivergent_m.h"

class node6_divergent : public cSimpleModule
{
private:
    simsignal_t arrivalSignal;

protected:
    virtual framedivergent *generateMessage();
    virtual void forwardMessage(framedivergent *msg);
    virtual void initialize() override;
    virtual void handleMessage(cMessage *msg) override;
};

Define_Module(node6_divergent);

void node6_divergent::initialize()
{
    arrivalSignal = registerSignal("arrival");
    if (getIndex() == 0) {
        framedivergent *msg = generateMessage();
        scheduleAt(simTime(), msg); // Schedule at the current simulation time
    }
}

void node6_divergent::handleMessage(cMessage *msg)
{
    framedivergent *ttmsg = check_and_cast<framedivergent *>(msg);

    if (ttmsg->getDestination() == getIndex()) {
        int hopcount = ttmsg->getHopCount();
        emit(arrivalSignal, hopcount);

        if (hasGUI()) {
            char label[50];
            sprintf(label, "last hopCount = %d", hopcount);
            cCanvas *canvas = getParentModule()->getCanvas();
            cTextFigure *textFigure = check_and_cast<cTextFigure *>(canvas-
>getFigure("lasthopcount"));
            textFigure->setText(label);
        }

        EV << "Message " << ttmsg << " arrived after " << hopcount << " hops.\n";
        bubble("ARRIVED, starting new one!");
        delete ttmsg;
        framedivergent *newmsg = generateMessage();
        forwardMessage(newmsg);
    } else {
        forwardMessage(ttmsg);
    }
}

framedivergent *node6_divergent::generateMessage()
{
    int src = getIndex();
    int n = getVectorSize();
    int dest = intuniform(0, n - 2);
    if (dest >= src)
        dest++;

    char msgname[20];
    sprintf(msgname, "tic-%d-to-%d", src, dest);
```

```
        framedivergent *msg = new framedivergent(msgname);
        msg->setSource(src);
        msg->setDestination(dest);
        msg->setHopCount(0); // Initialize hop count to 0
        return msg;
    }

    void node6_divergent::forwardMessage(framedivergent *msg)
    {
        msg->setHopCount(msg->getHopCount() + 1);
        int n = gateSize("gate$o");
        int k = intuniform(0, n - 1);

        EV << "Forwarding message " << msg << " on gate[" << k << "]\n";
        send(msg, "gate$o", k);
    }
```


samples - sp23-7BF-divergent/node6_divergent.cc - OMNeT++ IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
sp23-7BF-divergent
Binaries
Includes
out
results
General-#0.sca
lab2_divergent_exp2-#0.elog
lab2_divergent_exp2-#0.sca
lab3_divergent_exp1-#0.elog
lab3_divergent_exp1-#0.sca
lab3_divergent_exp2-#0.elog
lab3_divergent_exp2-#0.sca
lab3_divergent_exp3-#0.elog
lab3_divergent_exp3-#0.sca
lab3_divergent_exp4-#0.elog
lab3_divergent_exp4-#0.sca
lab5_divergent_exp0-#0.elog
lab5_divergent_exp1-#0.elog
lab5_divergent_exp1-#0.sca
lab5_divergent_exp2-#0.elog
lab6_divergent_exp1-#0.elog
lab6_divergent_exp1-#0.sca
lab6_divergent_exp1-#0.vci
lab6_divergent_exp1-#0.vec
framedivergent_m.cc
framedivergent_m.h
node_divergent.cc
node2_divergent.cc
node3_divergent.cc
node5_divergent.cc
node6_divergent.cc
sp23-7BF-divergent_dbg.exe - [amd64/le]
sp23-7BF-divergent.exe - [amd64/le]
framedivergent.msg
lab1-divergent.ned
lab2-divergent.ned
lab3-divergent.ned
lab5-divergent.ned
omnetpp.ini
lab6-divergent.ned
node6_divergent.cc
framedivergent.msg
lab6_divergent_exp1.anf
Problems
Module Hierarchy
NED Parameters
NED Inheritance
Console
Progress
Event Log
No event log available
Writable
Smart Insert
83 : 1 : 2361
8:25 PM
8/14/2023

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

using namespace omnetpp;
#include "framedivergent_m.h"

class node6_divergent : public cSimpleModule
{
private:
    simsignal_t arrivalSignal;

protected:
    virtual framedivergent *generateMessage();
    virtual void forwardMessage(framedivergent *msg);
    virtual void initialize() override;
    virtual void handleMessage(cMessage *msg) override;
};

Define_Module(node6_divergent);

void node6_divergent::initialize()
{
    arrivalSignal = registerSignal("arrival");
    if (getIndex() == 0) {
        framedivergent *msg = generateMessage();
        scheduleAt(simTime(), msg); // Schedule at the current simulation time
    }
}

void node6_divergent::handleMessage(cMessage *msg)
{
    framedivergent *ttmsg = check_and_cast<framedivergent *>(msg);

    if (ttmsg->getDestination() == getIndex()) {
        int hopcount = ttmsg->getHopCount();
        emit(arrivalSignal, hopcount);

        if (hasGUI()) {
```

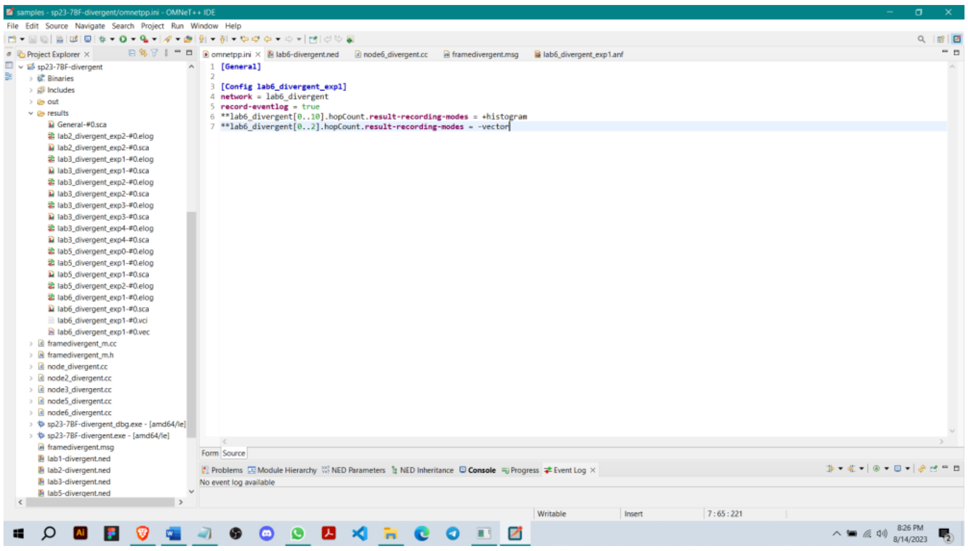
Fig 2: Snapshot of code in node6_divergent.cc file

7. MSG File:

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

8. INI File: *(only for LAB6)*

```
[General]

[Config lab6_divergent_exp1]

network = lab6_divergent
record-eventlog = true

**lab6_divergent[0..10].hopCount.result-recording-modes = +histogram
**lab6_divergent[0..2].hopCount.result-recording-modes = -vector
```



```
[General]

[Config lab6_divergent_expl]
network = lab6_divergent
record-eventlog = true
**lab6_divergent[0..10].hopCount.result-recording-modes = +histogram
**lab6_divergent[0..2].hopCount.result-recording-modes = -vector
```

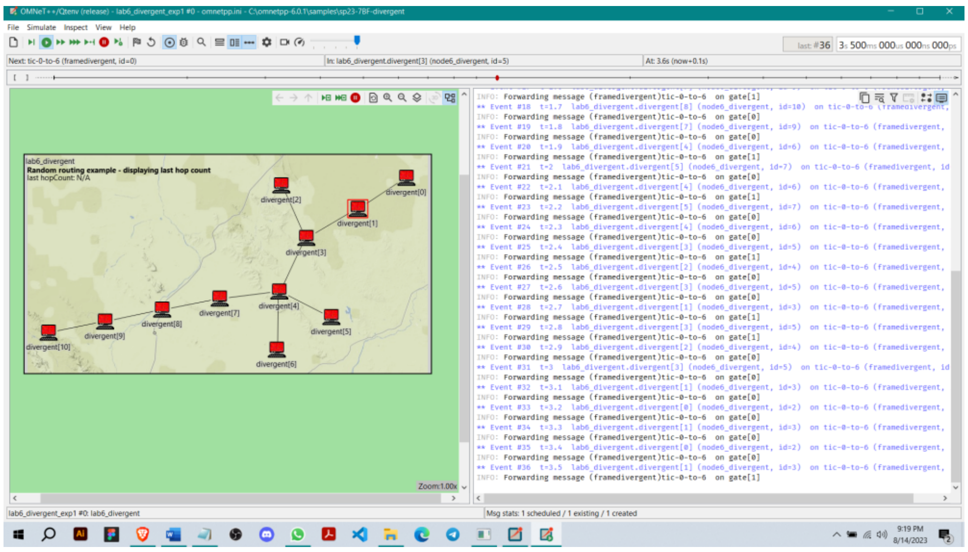
Fig 3: Snapshot of code in omnetpp.ini file

9. Build and Simulation:

The simulation's focal point is a network of 10 interconnected nodes, each employing the random routing algorithm to determine packet paths. The simulation is designed to investigate the behavior of a computer network consisting of 10 nodes using the random routing algorithm.

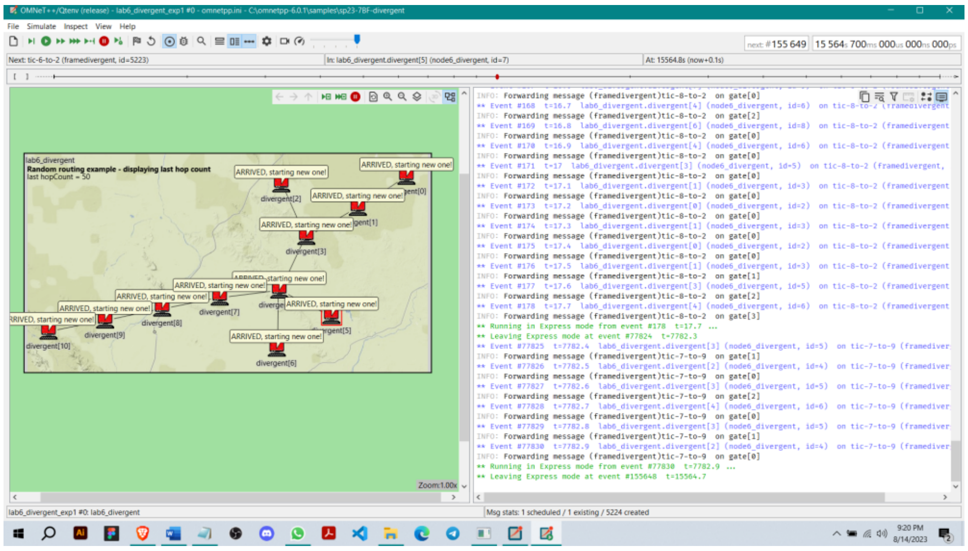
The simulation aims to analyze the impact of random routing on packet forwarding, particularly focusing on hop counts, and to provide insights into the variability and efficiency of this routing strategy. The network comprises 10 nodes interconnected by communication channels. Each node implements the random routing algorithm, where packets are forwarded randomly among available paths. At the start of the simulation, a set of messages is generated with source-destination pairs representing communication between nodes.

The simulation tracks the hop count of each packet, recording the number of intermediate nodes traversed.



```
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[1]
** Event #18  t=1.7  lab6_divergent.divergent[8] (node6_divergent, id=10)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[0]
** Event #19  t=1.8  lab6_divergent.divergent[7] (node6_divergent, id=9)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[0]
** Event #20  t=1.9  lab6_divergent.divergent[4] (node6_divergent, id=6)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[1]
** Event #21  t=2  lab6_divergent.divergent[5] (node6_divergent, id=7)  on tic-0-to-6 (framedivergent, id
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[0]
** Event #22  t=2.1  lab6_divergent.divergent[4] (node6_divergent, id=6)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[1]
** Event #23  t=2.2  lab6_divergent.divergent[5] (node6_divergent, id=7)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[0]
** Event #24  t=2.3  lab6_divergent.divergent[4] (node6_divergent, id=6)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[0]
** Event #25  t=2.4  lab6_divergent.divergent[3] (node6_divergent, id=5)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[1]
** Event #26  t=2.5  lab6_divergent.divergent[2] (node6_divergent, id=4)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[0]
** Event #27  t=2.6  lab6_divergent.divergent[3] (node6_divergent, id=5)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[0]
** Event #28  t=2.7  lab6_divergent.divergent[1] (node6_divergent, id=3)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[1]
** Event #29  t=2.8  lab6_divergent.divergent[3] (node6_divergent, id=5)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[1]
** Event #30  t=2.9  lab6_divergent.divergent[2] (node6_divergent, id=4)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[0]
** Event #31  t=3  lab6_divergent.divergent[3] (node6_divergent, id=5)  on tic-0-to-6 (framedivergent, id
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[0]
** Event #32  t=3.1  lab6_divergent.divergent[1] (node6_divergent, id=3)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[0]
** Event #33  t=3.2  lab6_divergent.divergent[0] (node6_divergent, id=2)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[0]
** Event #34  t=3.3  lab6_divergent.divergent[1] (node6_divergent, id=3)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[0]
** Event #35  t=3.4  lab6_divergent.divergent[0] (node6_divergent, id=2)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[0]
** Event #36  t=3.5  lab6_divergent.divergent[1] (node6_divergent, id=3)  on tic-0-to-6 (framedivergent,
INFO: Forwarding message (framedivergent)tic-0-to-6  on gate[1]
```

Fig 4: Simulation of Lab6



next: #155 649   15 564s 700ms 000us 000ns 000ps
At: 15564.8s (now+0.1s)

```
INFO: Forwarding message (framedivergent)tic-8-to-2  on gate[0]
** Event #168  t=16.7  lab6_divergent.divergent[4] (node6_divergent, id=6)  on tic-8-to-2 (framedivergent
INFO: Forwarding message (framedivergent)tic-8-to-2  on gate[2]
** Event #169  t=16.8  lab6_divergent.divergent[6] (node6_divergent, id=8)  on tic-8-to-2 (framedivergent
INFO: Forwarding message (framedivergent)tic-8-to-2  on gate[0]
** Event #170  t=16.9  lab6_divergent.divergent[4] (node6_divergent, id=6)  on tic-8-to-2 (framedivergent
INFO: Forwarding message (framedivergent)tic-8-to-2  on gate[0]
** Event #171  t=17  lab6_divergent.divergent[3] (node6_divergent, id=5)  on tic-8-to-2 (framedivergent,
INFO: Forwarding message (framedivergent)tic-8-to-2  on gate[0]
** Event #172  t=17.1  lab6_divergent.divergent[1] (node6_divergent, id=3)  on tic-8-to-2 (framedivergent
INFO: Forwarding message (framedivergent)tic-8-to-2  on gate[0]
** Event #173  t=17.2  lab6_divergent.divergent[0] (node6_divergent, id=2)  on tic-8-to-2 (framedivergent
INFO: Forwarding message (framedivergent)tic-8-to-2  on gate[0]
** Event #174  t=17.3  lab6_divergent.divergent[1] (node6_divergent, id=3)  on tic-8-to-2 (framedivergent
INFO: Forwarding message (framedivergent)tic-8-to-2  on gate[0]
** Event #175  t=17.4  lab6_divergent.divergent[0] (node6_divergent, id=2)  on tic-8-to-2 (framedivergent
INFO: Forwarding message (framedivergent)tic-8-to-2  on gate[0]
** Event #176  t=17.5  lab6_divergent.divergent[1] (node6_divergent, id=3)  on tic-8-to-2 (framedivergent
INFO: Forwarding message (framedivergent)tic-8-to-2  on gate[1]
** Event #177  t=17.6  lab6_divergent.divergent[3] (node6_divergent, id=5)  on tic-8-to-2 (framedivergent
INFO: Forwarding message (framedivergent)tic-8-to-2  on gate[2]
** Event #178  t=17.7  lab6_divergent.divergent[4] (node6_divergent, id=6)  on tic-8-to-2 (framedivergent
INFO: Forwarding message (framedivergent)tic-8-to-2  on gate[3]
** Running in Express mode from event #178  t=17.7 ...
** Leaving Express mode at event #77824  t=7782.3
** Event #77825  t=7782.4  lab6_divergent.divergent[3] (node6_divergent, id=5)  on tic-7-to-9 (framediver
INFO: Forwarding message (framedivergent)tic-7-to-9  on gate[1]
** Event #77826  t=7782.5  lab6_divergent.divergent[2] (node6_divergent, id=4)  on tic-7-to-9 (framediver
INFO: Forwarding message (framedivergent)tic-7-to-9  on gate[0]
** Event #77827  t=7782.6  lab6_divergent.divergent[3] (node6_divergent, id=5)  on tic-7-to-9 (framediver
INFO: Forwarding message (framedivergent)tic-7-to-9  on gate[2]
** Event #77828  t=7782.7  lab6_divergent.divergent[4] (node6_divergent, id=6)  on tic-7-to-9 (framediver
INFO: Forwarding message (framedivergent)tic-7-to-9  on gate[0]
** Event #77829  t=7782.8  lab6_divergent.divergent[3] (node6_divergent, id=5)  on tic-7-to-9 (framediver
INFO: Forwarding message (framedivergent)tic-7-to-9  on gate[1]
** Event #77830  t=7782.9  lab6_divergent.divergent[2] (node6_divergent, id=4)  on tic-7-to-9 (framediver
INFO: Forwarding message (framedivergent)tic-7-to-9  on gate[0]
** Running in Express mode from event #77830  t=7782.9 ...
** Leaving Express mode at event #155648  t=15564.7
```

Msg stats: 1 scheduled / 1 existing / 5224 created

Fig 5: End of Simulation of Lab6

10. Result Analysis:

After conducting simulations of the 10-node network employing random routing and collecting various data types, it's time to delve into the analysis of the results. The data collected provides insights into the behavior of the network and the impact of random routing on packet forwarding. Here, we focus on the analysis of statistical metrics, including the mean hop count, distribution of hop counts, and their implications.

Scalar, histogram, and vector data are collected during the simulation. Scalar data includes metrics like simulation time. Histogram data captures hop count distribution. Vector data stores individual hop counts for each packet.

Here are the **Data** browsed after simulation—

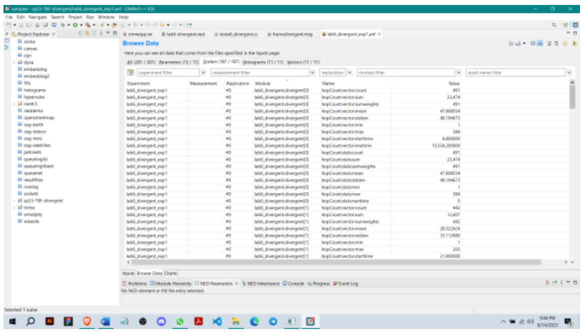


Fig 6: Scalar Data

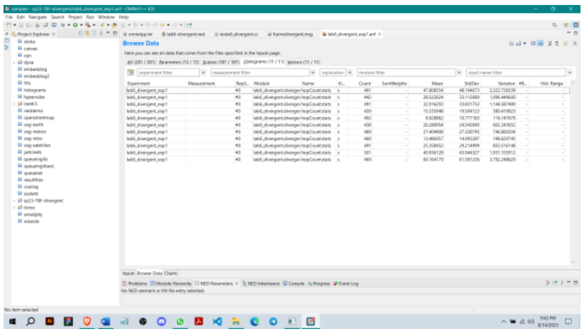


Fig 7: Histogram Data

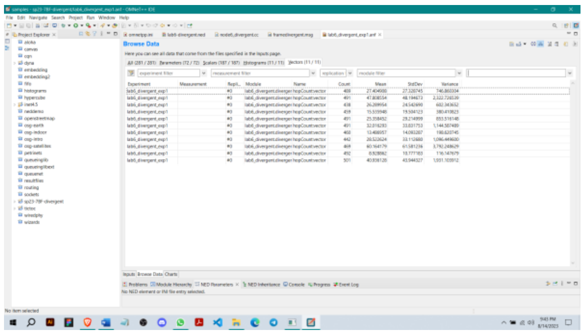


Fig 8: Vector Data

The **mean hop count** is a critical performance metric that reflects the average number of intermediate nodes a packet traverses from source to destination. By calculating the mean hop count from the collected vector data, we gain an understanding of how random routing affects the overall path length.

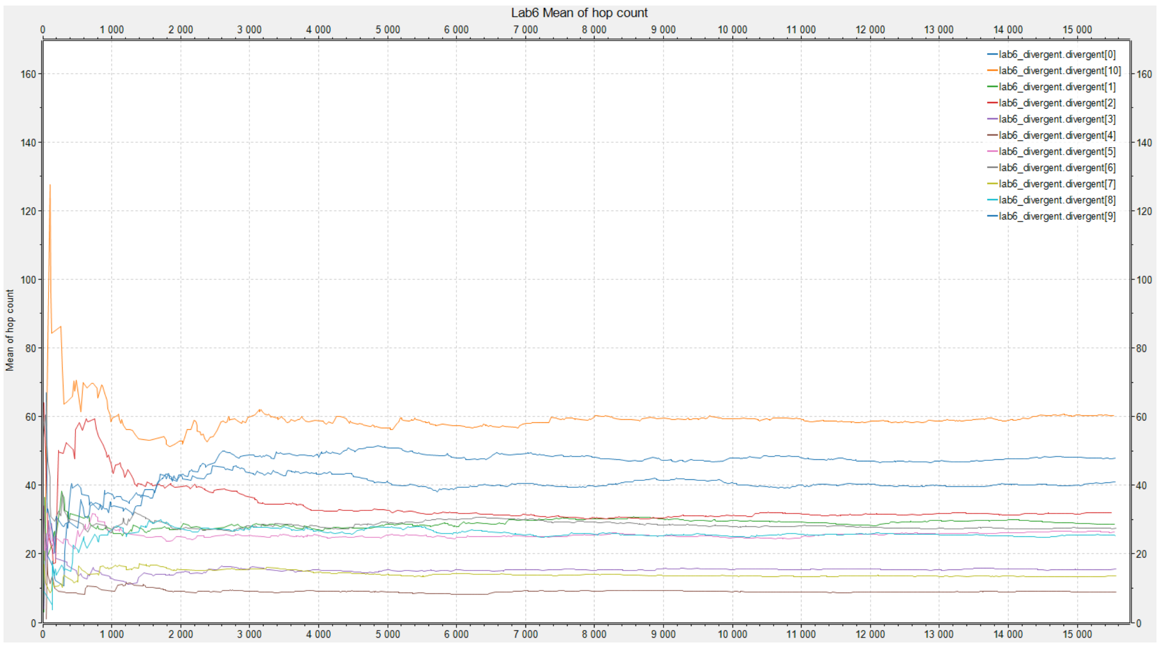


Fig 9: Mean of Hop Count

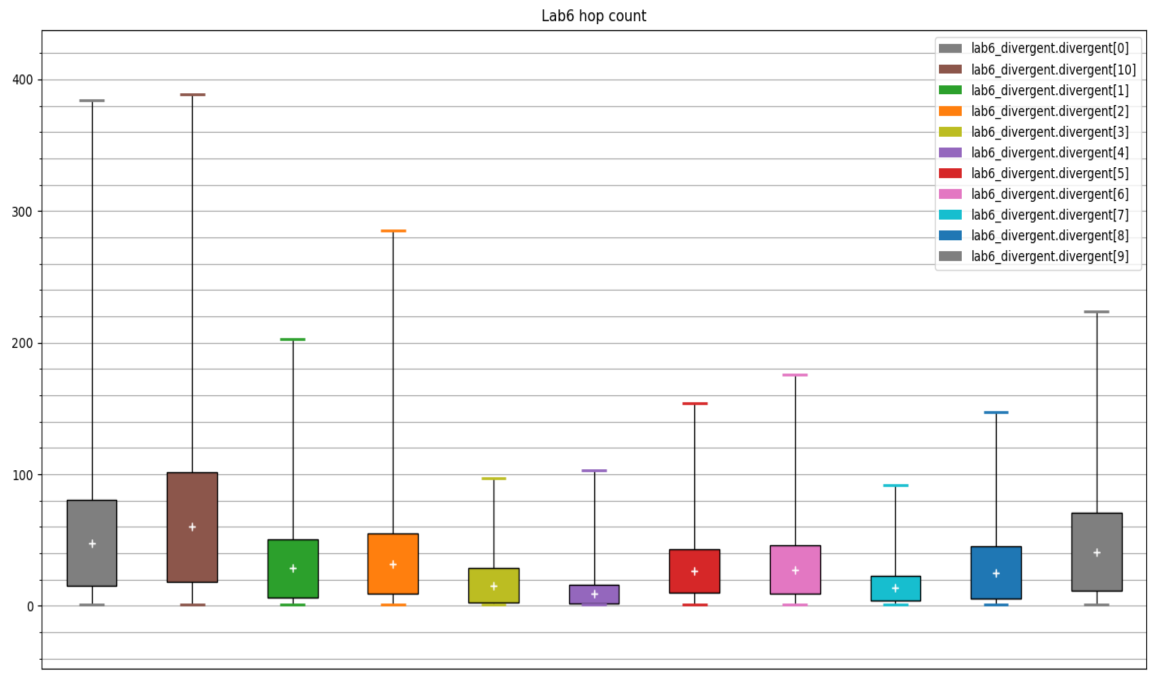


Fig 10: Histogram of Hop Count

By examining the histogram data, we can identify whether packets typically take shorter, direct paths or traverse more intermediate nodes. A narrower distribution implies consistent routing, while a broader distribution suggests randomness in forwarding decisions. To effectively convey the results, data visualization tools such as Matplotlib can be utilized to create graphs and charts. Visual representations of mean hop counts and hop count distributions provide a clearer understanding of trends and variations, making it easier to communicate findings to stakeholders and peers. A higher mean hop count might signify more exploration of available paths.

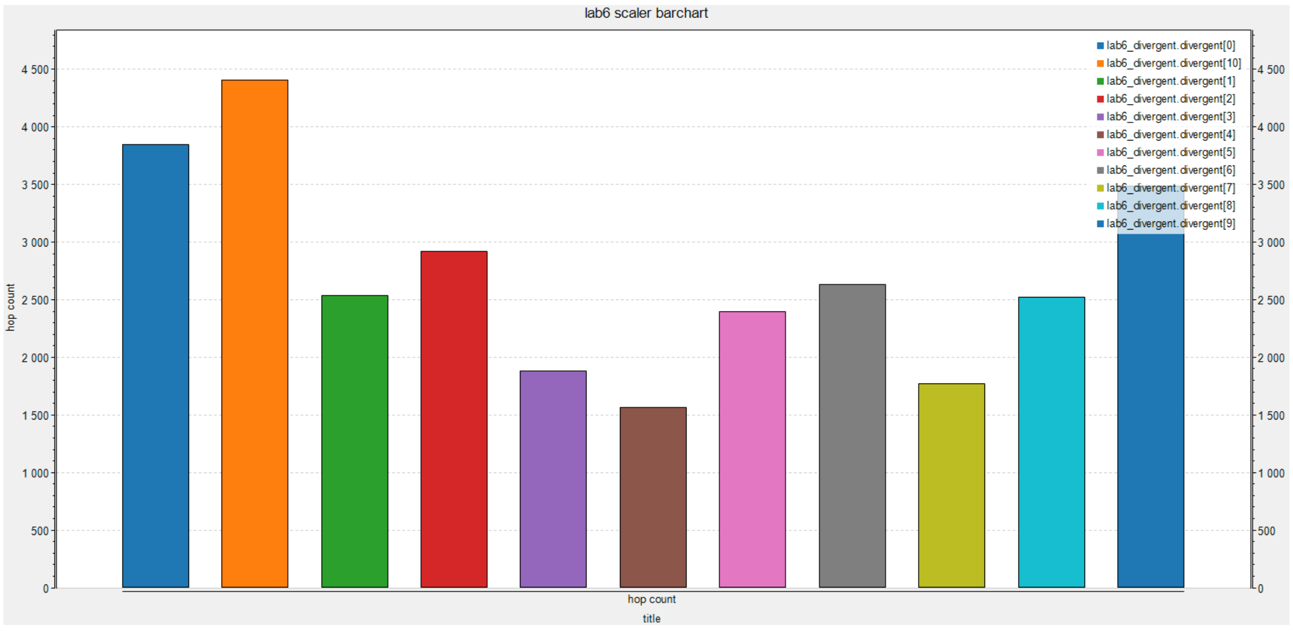


Fig 11: Scalar Bar chart

11. Conclusion:

In conclusion, this lab work in OMNeT++ the simulation journey has unveiled the dynamics of random routing's impact on network behavior. Through meticulous data collection, statistical analysis, and visualization, we've witnessed the significance of mean hop counts and hop count distributions. This exploration prompts discussions on the balance between randomness and efficiency in network strategies. The simulation's legacy encourages further investigation, fostering a deeper understanding of routing algorithms and network performance.