MIIDDLE EAST TECHNICAL UNIVERSITY

ELECTRICAL-ELECTRONICS ENGINEERING

EE444 Introduction to Computer Networks

HW3 - OMNeT++

Mehmet ATAŞ 2304020

06.07.2022

HW3 - OMNeT++

Introduction

The homework consists of two parts. In the first part, I simulate an Ethernet network with a shared medium. In the second part, I simulate TCP traffic. I create multiple projects for each question. To be able to use INET, I reference the INET project in project properties.

Installation & Preparation

I download OMNeT++ 6.0. To complete the installation, I follow the instructions given in the "doc/InstallGuide". I create new projects for the solutions to the homework. An OMNeT++ project consists of a .ned file that defines the network that will be simulated and a .ini file that defines simulation parameters.

Data-Link Layer – Ethernet

1. For this question, I create an Ethernet network that uses a common bus. (try in the zip file)

(a) I create a .ned file for my network. I define a parameter for node count and arrays for EthernetHost and WireJunction. WireJunction is used for creating a bus. I connect WireJunctions serially using an Eth10M link. Then connect each EthernetHost to a WireJunction using an Eth10M link.

package.ned × omnetpp.ini General.anf

```
package try.simulations;
import inet.node.ethernet.Eth10M;
import inet.node.ethernet.EthernetHost;
import inet.node.inet.StandardHost;
import inet.physicallayer.wired.common.WireJunction;
import inet.tutorials.protocol.Node;
@license(LGPL);
//
// TODO documentation
//
network try
{
    parameters:
        int numberOfNodes;
        @display("bgb=684,456");
    submodules:
        ethernetHost[numberOfNodes]: EthernetHost {
            @display("p=453,323;b=79,104");
        }
        wireJunction[numberOfNodes]: WireJunction {
            @display("p=149,307;b=80,111");
        }
    connections:
        for i=0..(numberOfNodes)-2 {
            wireJunction[i].port++ <--> Eth10M <--> wireJunction[i+1].port++;
            wireJunction[i].port++ <--> Eth10M <--> ethernetHost[i].ethg;
        }
        wireJunction[numberOfNodes-1].port++ <--> Eth10M <--> ethernetHost[numberOfNodes-1].ethg;
}
```

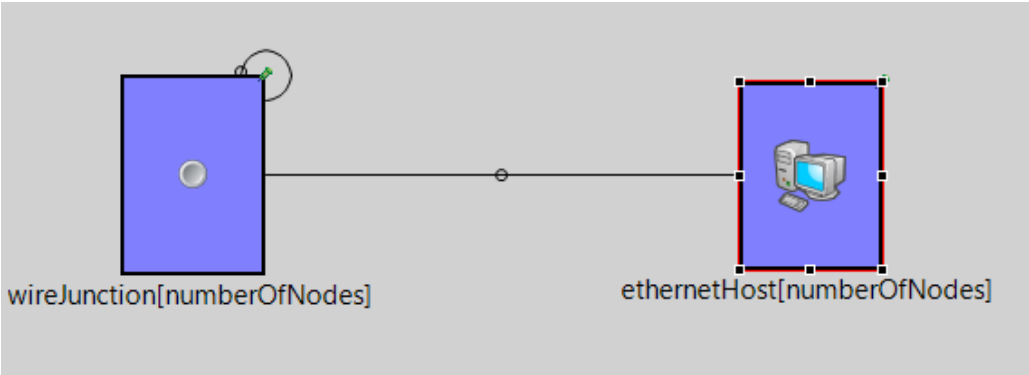

Figure 1. Ned file for question 1 part 1 (source and design)

(b) I create a .ini file to configure my simulation. I designate destination addresses, request length and response length. I set my send interval as exponential.

```
[General]
network = try
sim-time-limit = 1s
simtime-resolution = ps

#WireJunction
#EthernetHost
**.numberOfNodes = 16
**.cli.reqLength = 1024B
**.cli.respLength = 1024B
**.ethernetHost[0].cli.destAddress = ""
**.cli.destAddress = "ethernetHost[0]"
**.cli.sendInterval = exponential(1s)
```

Figure 2. ini file for question 1 part 1

(c) I simulate my network for different node counts of 2, 4, 8, 16, 32, 64 and 128. I plot the node count vs channel efficiency graph. Channel efficiency is defined as how much payload is sent through the channel per second per physical rate (10Mbps for our case. Required parameters for my calculations are generated automatically in the Results folder.

try.ethernetHost[0].srv

Node count		
2	packetReceived:sum(packetBytes) (scalar)	1,024
4	packetReceived:sum(packetBytes) (scalar)	1,024
8	packetReceived:sum(packetBytes) (scalar)	3,072
16	packetReceived:sum(packetBytes) (scalar)	11,264
32	packetReceived:sum(packetBytes) (scalar)	23,552
64	packetReceived:sum(packetBytes) (scalar)	50,176
128	packetReceived:sum(packetBytes) (scalar)	105,472

While calculating, I used the number of received packets on Host0.

As expected, we can increase the channel efficiency by increasing node count.

$$\textbf{Channel Efficiency} = \frac{packetReceived * 8bit/Bytes}{10Mbps * SimTime}$$

SimTime = 1s

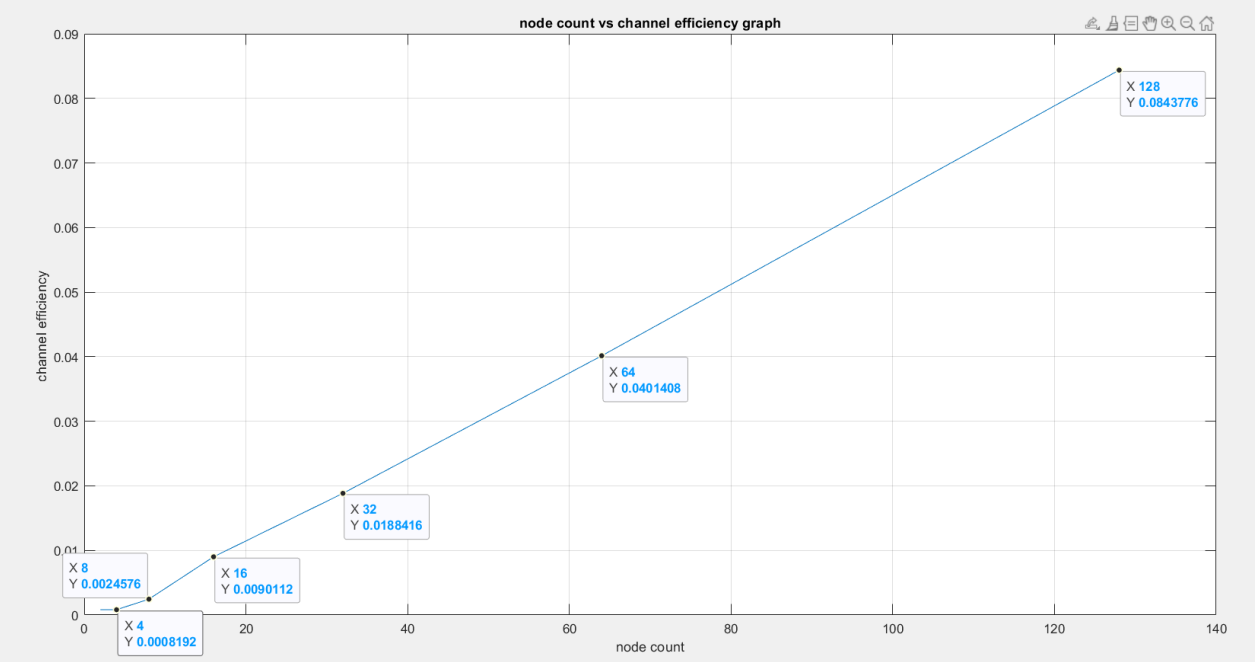

Figure 3. Plot of the node count vs channel efficiency graph

(d) Using 16 nodes, I simulate for different frame lengths of 64, 128, 256, 512 and 1024. I plot the channel efficiency vs frame length graph.

Frame lengths

64 packetReceived:sum(packetBytes) (scalar) 704

128 packetReceived:sum(packetBytes) (scalar) 1,408

256 packetReceived:sum(packetBytes) (scalar) 2,816

512 packetReceived:sum(packetBytes) (scalar) 5,632

1024 packetSent:sum(packetBytes) (scalar) 11,264

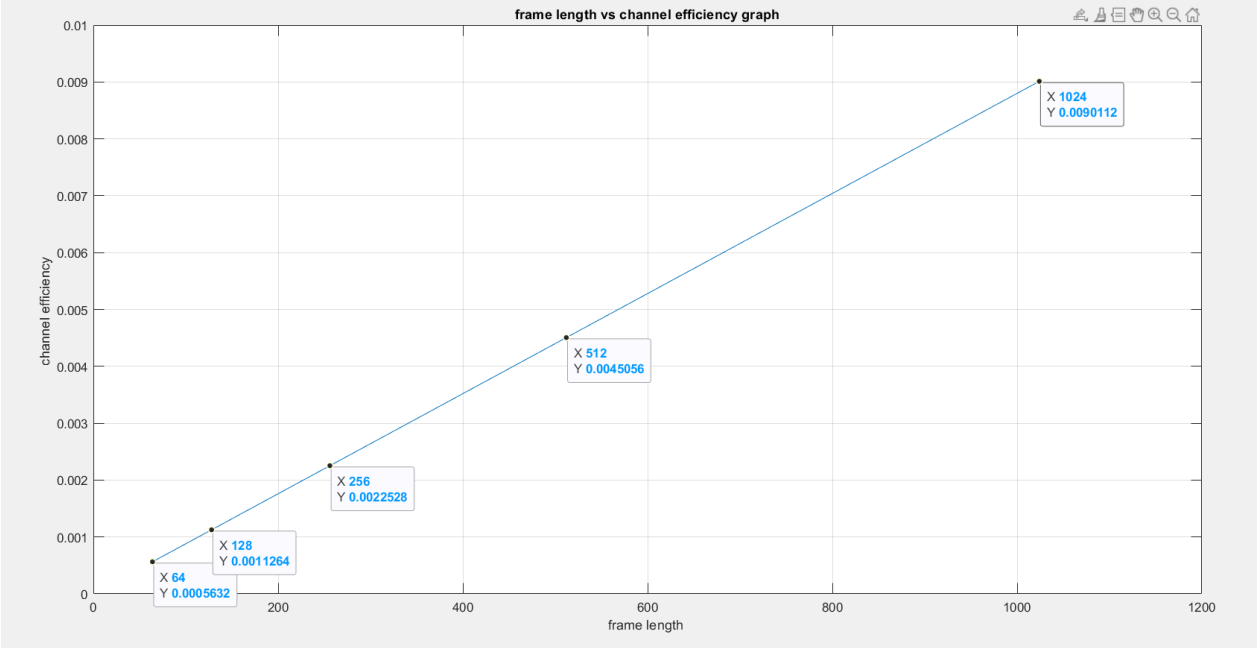

Figure 4. Plot of the frame length vs channel efficiency graph

As expected, we can increase the channel efficiency by increasing frame length.

packetReceived:count is the same for all frame length because sendInterval is kept constant.

packetReceived:count = $\frac{\text{packetReceived:sum}}{frameLength}$

However, since our frame length changes, our channel efficiency alters.

(e) Using 16 nodes, I divide the exponential parameter for your send interval by 2 and simulate. If we compare the efficiency with my previous simulation with 32 nodes.

Before, 32 nodes - 1024 frame length - exp(1s)

packetReceived:sum(packetBytes) (scalar)	23,552

Now, 16 nodes - 1024 frame length - exp(0.5s)

packetReceived:sum(packetBytes) (scalar)	27,648

When we decrease node count and send interval to half of their values, we see that packetReceived:sum doesn't change significantly. Because, decreasing node count decreases the channel efficiency and decreasing send interval increases the channel efficiency. Thus, they compensate each other.

2. Add a switch to divide the bus into two equal-size collision domains and repeat the same simulations in Question 1. (trysecond in the zip file)

When I add a switch, I split the collision domain into two. When a host on the left will send a packet to another host on the left, it will not go to the right. This also applies to the other side. However, we should consider that when the right side sends to the left, there are equal numbers of nodes on the right and left, and the probability of sending packets is equal. Normal collisions will decrease if we put the switch. If the destination was chosen randomly, both sides should be equal.

```
    submodules:
        ethernetHost[numberOfNodes]: EthernetHost {
            @display("p=338,87;b=79,105");
        }
        wireJunction[int((numberOfNodes)/2)]: WireJunction {
            @display("p=149,307;b=80,111");
        }
        wireJunctionnn[int((numberOfNodes)/2)]: WireJunction {
            @display("p=549,307;b=80,111");
        }
        ethernetSwitch: EthernetSwitch {
            @display("p=316,308;b=60,62");
        }
    connections:
        for i=0..(int(((numberOfNodes)/2)-2)) {
            wireJunction[i].port++ <--> Eth10M <--> wireJunction[i+1].port++;
            wireJunction[i].port++ <--> Eth10M <--> ethernetHost[i].ethg;
        }
        wireJunction[int(((numberOfNodes)/2)-1)].port++ <--> Eth10M <--> ethernetHost[int(((numberOfNodes)/2)-1)].ethg;
        ethernetSwitch.ethg++ <--> Eth10M <--> wireJunction[int(((numberOfNodes)/2)-1)].port++;
        ethernetSwitch.ethg++ <--> Eth10M <--> wireJunctionnn[0].port++;
        for i=0..(int(((numberOfNodes)/2)-2)) {
            wireJunctionnn[i].port++ <--> Eth10M <--> wireJunctionnn[i+1].port++;
            wireJunctionnn[i].port++ <--> Eth10M <--> ethernetHost[int(i+((numberOfNodes)/2))].ethg;
        }
        wireJunctionnn[int (((numberOfNodes/2))-1)].port++ <--> Eth10M <--> ethernetHost[int ((numberOfNodes)-1)].ethg;
}
```

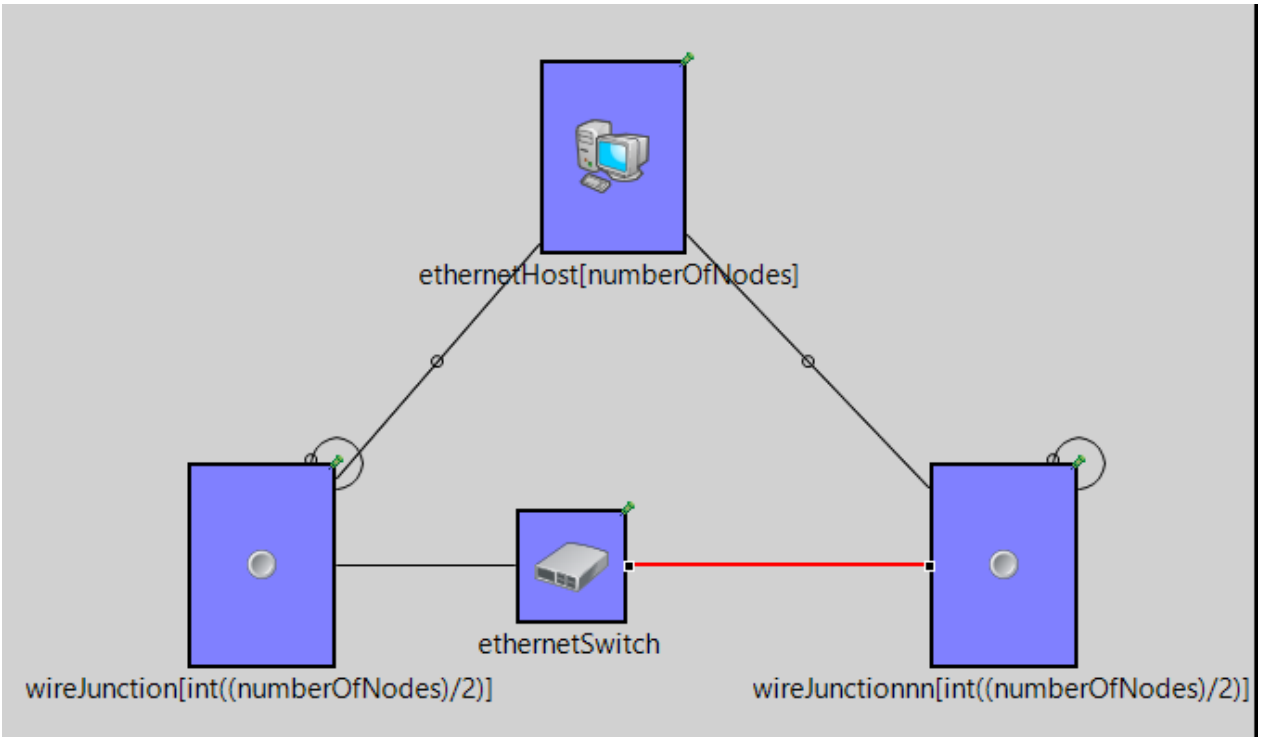

Figure 5. Ned file for question 1 part 2 (source and design)

```
[General]
network = trysecond
sim-time-limit = 1s
simtime-resolution = ps

#WireJunction
#EthernetHost
**.numberOfNodes = 32
**.cli.reqLength = 1024B
**.cli.respLength = 1024B
**.ethernetHost[0].cli.destAddress = ""
**.cli.destAddress = "ethernetHost[0]"
**.cli.sendInterval = exponential(1s)
```

Figure 6. ini file for question 1 part 2

Node count		
2	packetReceived:sum(packetBytes) (scalar)	0
4	packetReceived:sum(packetBytes) (scalar)	1,024
8	packetReceived:sum(packetBytes) (scalar)	3,072
16	packetReceived:sum(packetBytes) (scalar)	10,240
32	packetReceived:sum(packetBytes) (scalar)	21,504
64	packetReceived:sum(packetBytes) (scalar)	54,272
128	packetReceived:sum(packetBytes) (scalar)	102,400

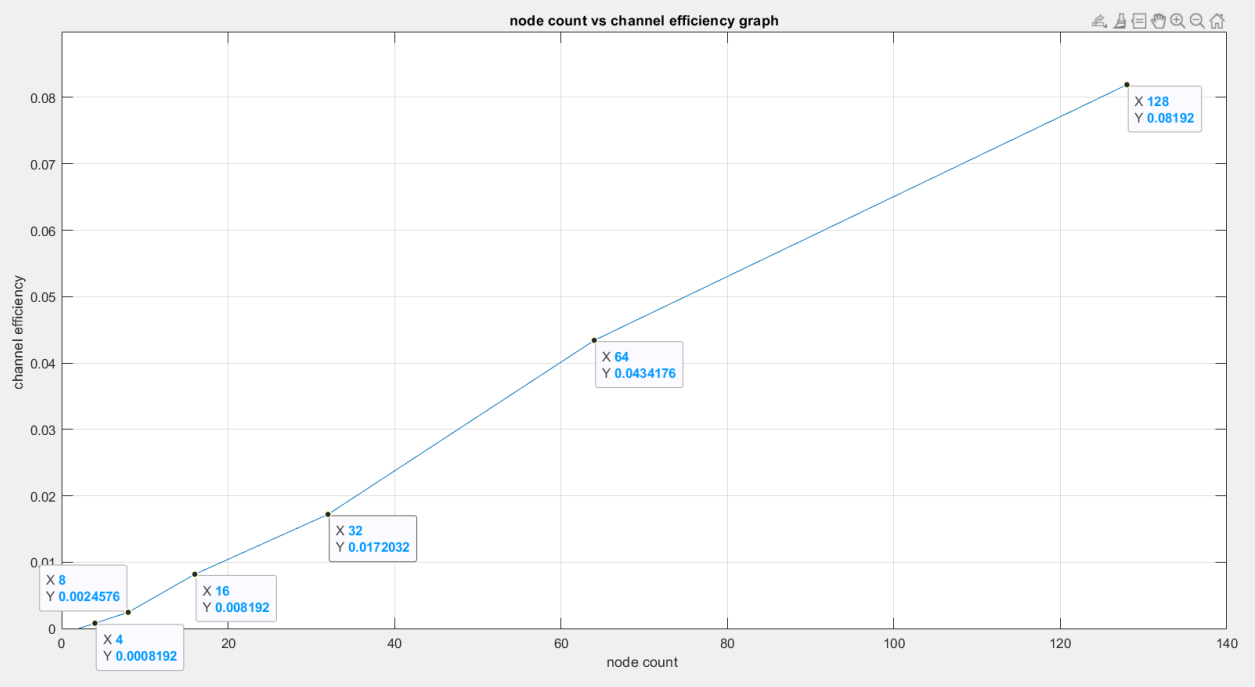

Figure 7. Plot of the node count vs channel efficiency graph

Frame lengths

64 packetReceived:sum(packetBytes) (scalar) 640

128 packetReceived:sum(packetBytes) (scalar) 1,280

256 packetReceived:sum(packetBytes) (scalar) 2,560

512 packetReceived:sum(packetBytes) (scalar) 5,120

1024 packetReceived:sum(packetBytes) (scalar) 10,240

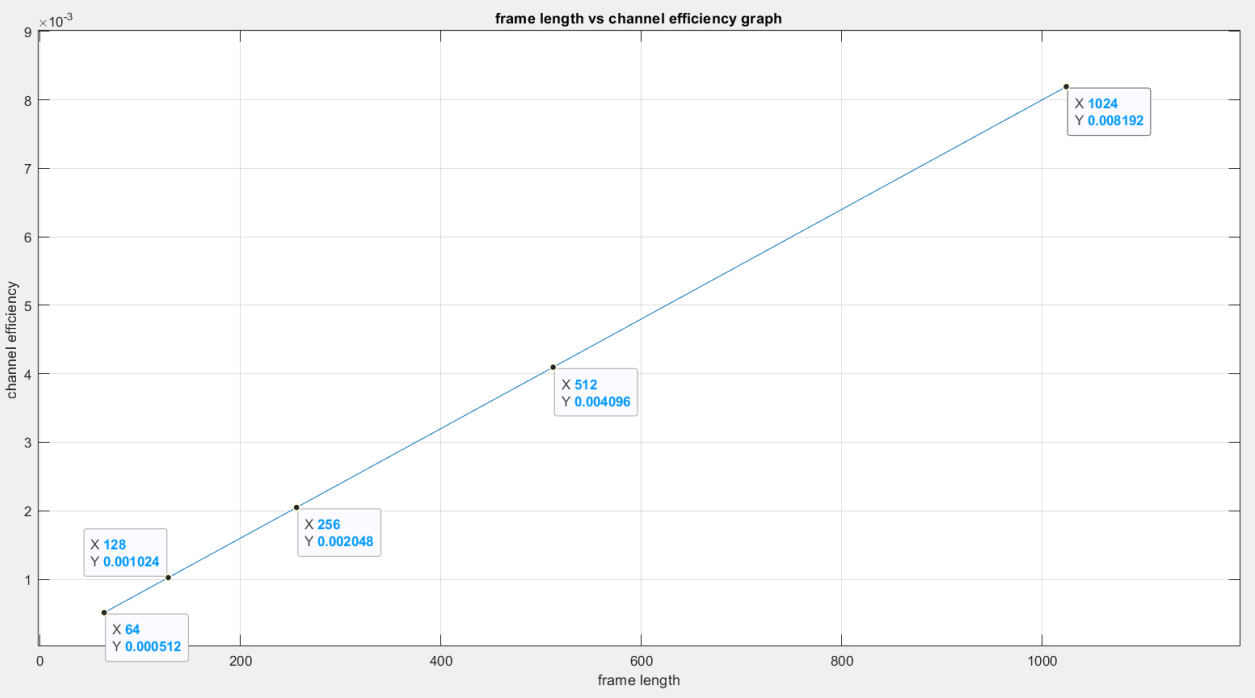

Figure 8. Plot of the frame length vs channel efficiency graph

As expected, we can increase the channel efficiency by increasing frame length.

packetReceived:count is the same for all frame length because sendInterval is kept constant.

packetReceived:count = $\frac{\text{packetReceived:sum}}{frameLength}$

However, since our frame length changes, our channel efficiency alters.

Before, 32 nodes - 1024 frame length - exp(1s)

packetReceived:sum(packetBytes) (scalar) 21,504

Now, 16 nodes - 1024 frame length - exp(0.5s)

packetReceived:sum(packetBytes) (scalar) 26,624

When we decrease node count and send interval to half of their values, we see that packetReceived:sum doesn't change significantly. Because, decreasing node count decreases the channel efficiency and decreasing send interval increases the channel efficiency. Thus, they compensate each other.

Appendix

Total time spent on report writing: 8 hours

References

[1] https://docs.omnetpp.org/tutorials/tictoc/

[2] OMNeT++ Simulation Manual Version 6.x