به نام خدا

Sharif University of Technology

Department of Electrical Engineering

Data Network

Instructor: Dr. Pakravan

Data Link Layer

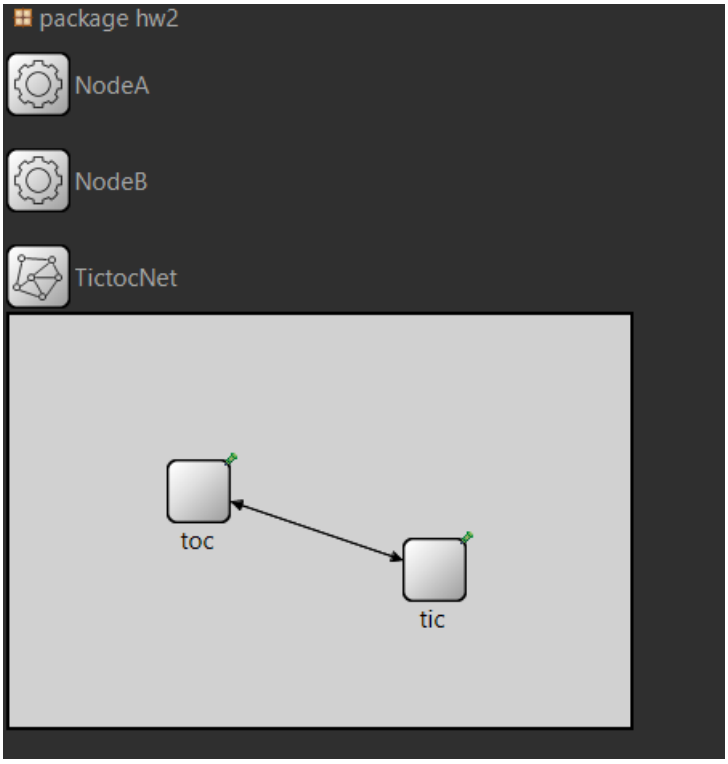
Mohammad Javad Amin

401211193

Fall 2023

سوال ۱

ابتدا ساختار شبکه که متشکل از دو نود است را تشکیل داده و ورودی و خروجی‌ها را با توجه به تاخیر مسیر متصل می‌کنیم. در این جا $t_{prop} = 2s$ فرض شده. با توجه به تفاوت عملکردی هر نود باید دو نوع نود را تعریف کرده NodeA and NodeB البته برای قسمت اول یک نوع هم کافی بوده ولی در ادامه نیاز به تعریف دو نوع است.




```
simple NodeA
{
    gates:
        input in;
        output out;
}

simple NodeB
{
    gates:
        input in;
        output out;
}


network TictocNet
{
    @display("bgb=387,257");
    submodules:
        tic: NodeA {
            @display("p=265,159");
        }
        toc: NodeB {
            @display("p=118,110");
        }
    connections:
        tic.out --> {  delay = 2000ms; } --> toc.in;
        tic.in <-- {  delay = 2000ms; } <-- toc.out;
}
```

حال با توجه به عملیاتی که نود Tic انجام می‌دهد باید کلاس آن را بنویسم. این نود، ارسال کنند بوده به روش stop and wait است. پیاده سازی به شکل زیر است.

```
void NodeA::initialize()
{
    // Initialize variables.
    timeout = 5.0;
    delay = 2.0;
    Tf = delay;
    timeoutEvent = new cMessage("timeout");
    sendEvent = new cMessage("Send");

    linkUtilizationVector.setName("LinkUtilizationVector");


    // Generate and send an initial message.
    scheduleAt(simTime() + delay, sendEvent);
}

void NodeA::handleMessage(cMessage *msg)
{
    if (msg == timeoutEvent)
    {
        // If we receive the timeout event, that means the packet hasn't
        // arrived in time and we have to re-send it.
        EV << "Timeout expired, resending the message \n";
        scheduleAt(simTime() + delay, sendEvent);
    }
    else if (msg == sendEvent)
    {
        // Ready to send another one.

        EV << "Sending a message.\n";
        cMessage *newMsg = new cMessage("Hello");
        send(newMsg, "out");
        scheduleAt(simTime() + timeout, timeoutEvent);
    }
    else
    {
        // Message arrived
        // Acknowledgement received
        delete msg;
        EV << "Timer cancelled.\n";
        cancelEvent(timeoutEvent);
        scheduleAt(simTime() + delay, sendEvent);

        TotalTime = simTime() - TotalTime;
        double utilization = (Tf / TotalTime) * 100;


        linkUtilizationVector.record(utilization);

        EV << "Link Utilization: " << utilization << "%" << endl;

        // Reset TotalTime
        TotalTime = simTime();
    }
}
```

ابتدا هر ارسال کانال باید به اندازه delay(t_f) صبر کند که بسته آماده انتقال شود. برای پیاده‌سازی این تاخیر از self_message استفاده می‌کنیم و یک Timer میگذاریم که بعد از t_f به نود بگوید بسته آماده ارسال است.

```
scheduleAt(simTime() + delay, sendEvent)
```

بعد از ارسال،زمان‌سنج t_{out} فعال می‌شود و اگر بسته ACK نیامد بسته مجدد ارسال می‌شود و اگر ACK آمد این زمان‌سنج متوقف شده و آماده ارسال بسته بعدی‌می‌شویم. این الگوریتم را نیز با استفاده از self_message پیاده‌سازی می‌کنیم. کل زمان انتقال برای هربسته را ذخیره کرده و با توجه به فرمول Utility آن بسته را حساب کرده و ذخیره می‌کنیم.

در نود TOC با احتمالی بسته ورودی را حذف می‌کنیم(گم می‌کنیم) و اگربسته را حذف نکردیم برای آن یک ACK فرستاده که طول فریم آن را ناچیز در نظر می‌گریم.

```
class NodeB : public cSimpleModule
{
  protected:
    virtual void handleMessage(cMessage *msg) override;
};

Define_Module(NodeB);

void NodeB::handleMessage(cMessage *msg)
{
    if (uniform(0, 1) > 0.8)
    {
        EV << "\"Losing\" message.\n";
        bubble("message lost");
        delete msg;
    }
    else
    {
        delete msg;
        cMessage *newMssg = new cMessage("ACK");
        EV << "Sending Acknowledgment.\n";
        send(newMssg, "out");
    }
}
```

برای چند سناریو مختلف نمودار Utility را به صورت لحظه‌ایی و میانگین رسم می‌کنیم.

باید t_{out} به صورت $t_{out} > 2t_{prop}$ تعین شود زیرا اگر این‌گونه طراجی نکنیم شبکه را به صورت ذاتی هر بسته را باید بیشتر از یک بار ارسال کنیم.

scenario A :

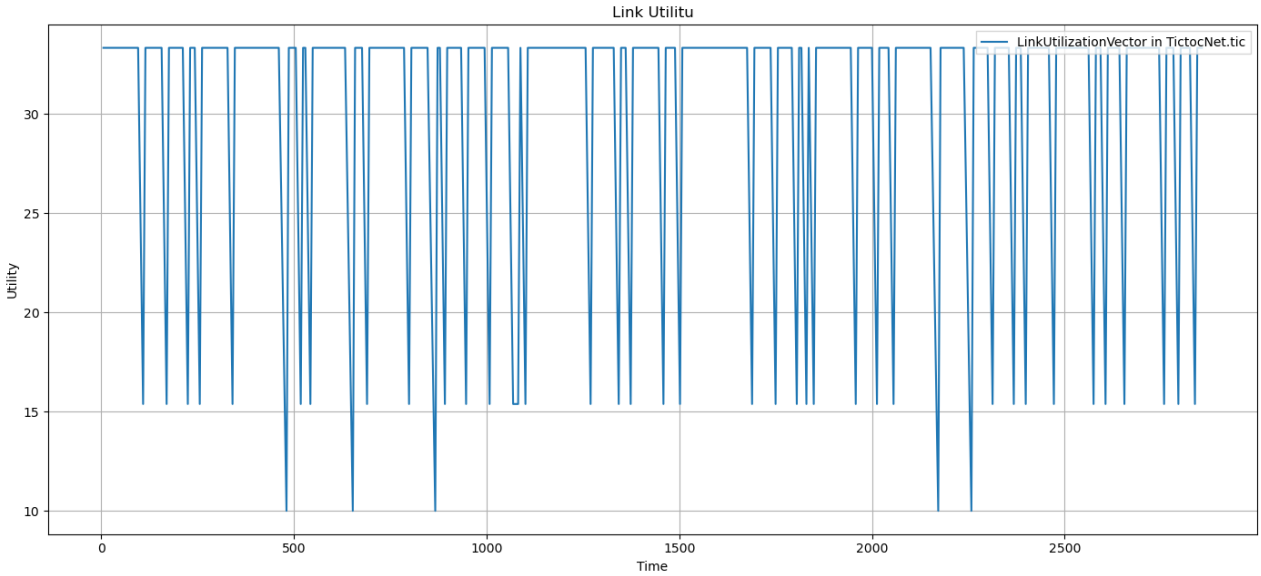
نمودار بهره کانال

$$P = 0.1\,, t_{prop} = 2s\,, t_f = 2s, t_{out} = 5s$$

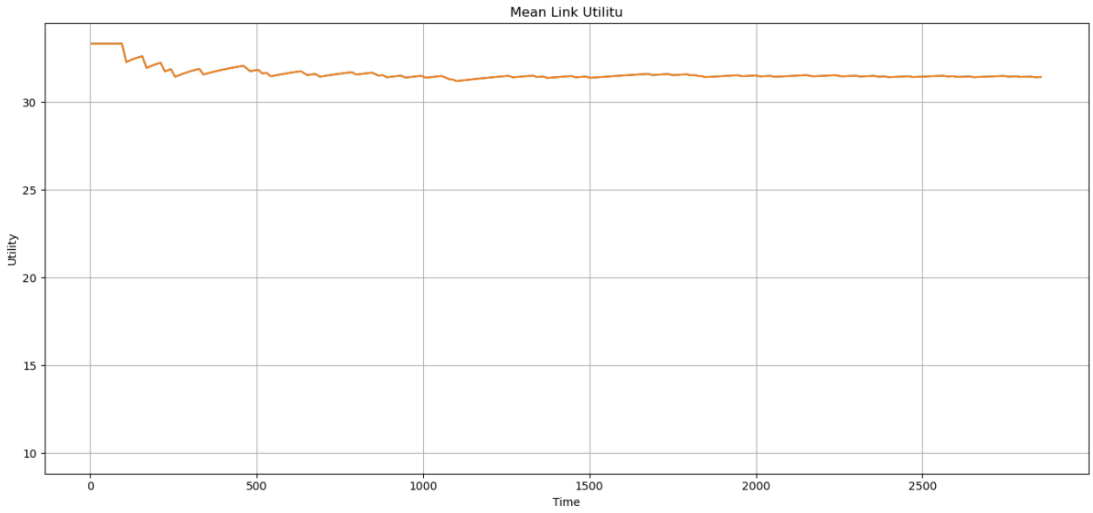
$$a = \frac{t_{prop}}{t_f}\,, \mathrm{U} = \frac{1-p}{1+2a}$$

$$U = 30\%$$

نمودار بهره کانال لحظه‌ایی



نمودار میانگین بهره کانال



Mean =31.4%

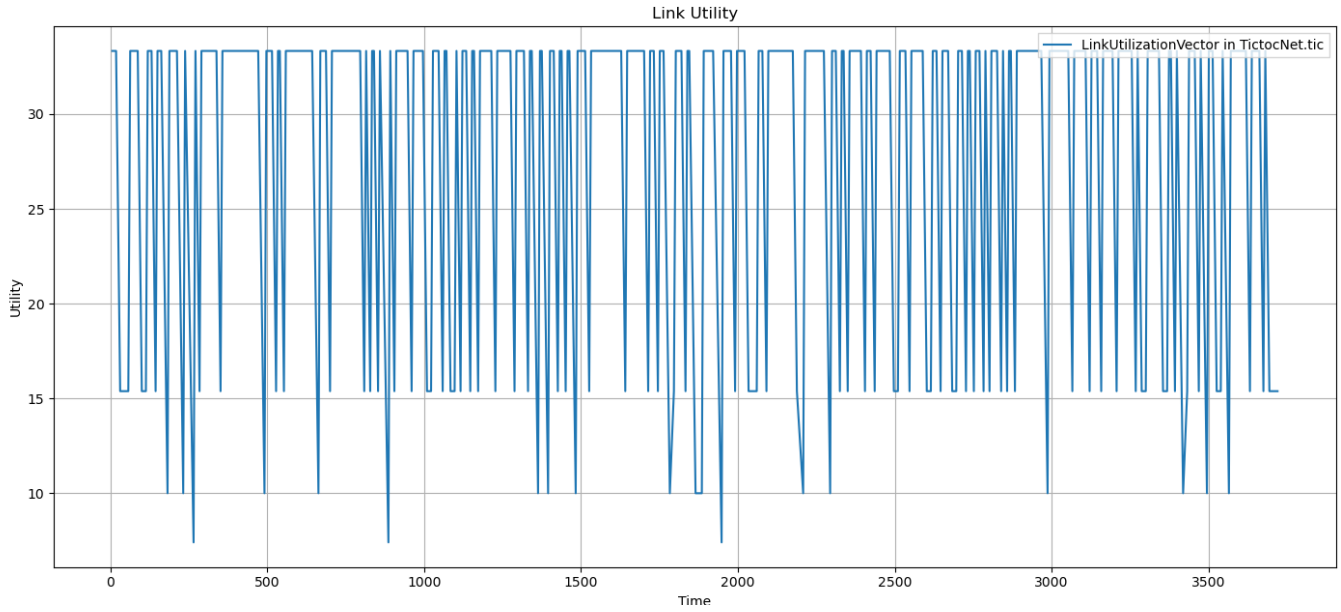
scenario B :

$$P = 0.2\ , t_{prop} = 2s\ , t_f = 2s, t_{out} = 5s$$

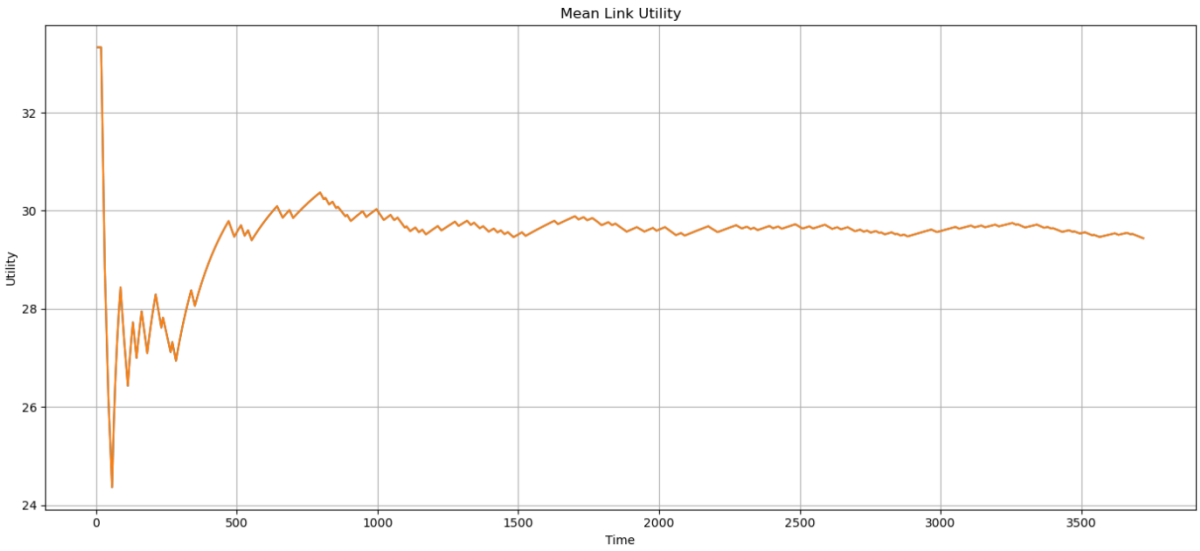
$$a = \frac{t_{prop}}{t_f}\ , \mathrm{U} = \frac{1-p}{1+2a}$$

$$U = 26.6\%$$

نمودار بهره کانال لحظه‌ایی



نمودار میانگین بهره کانال



Mean =29.4%

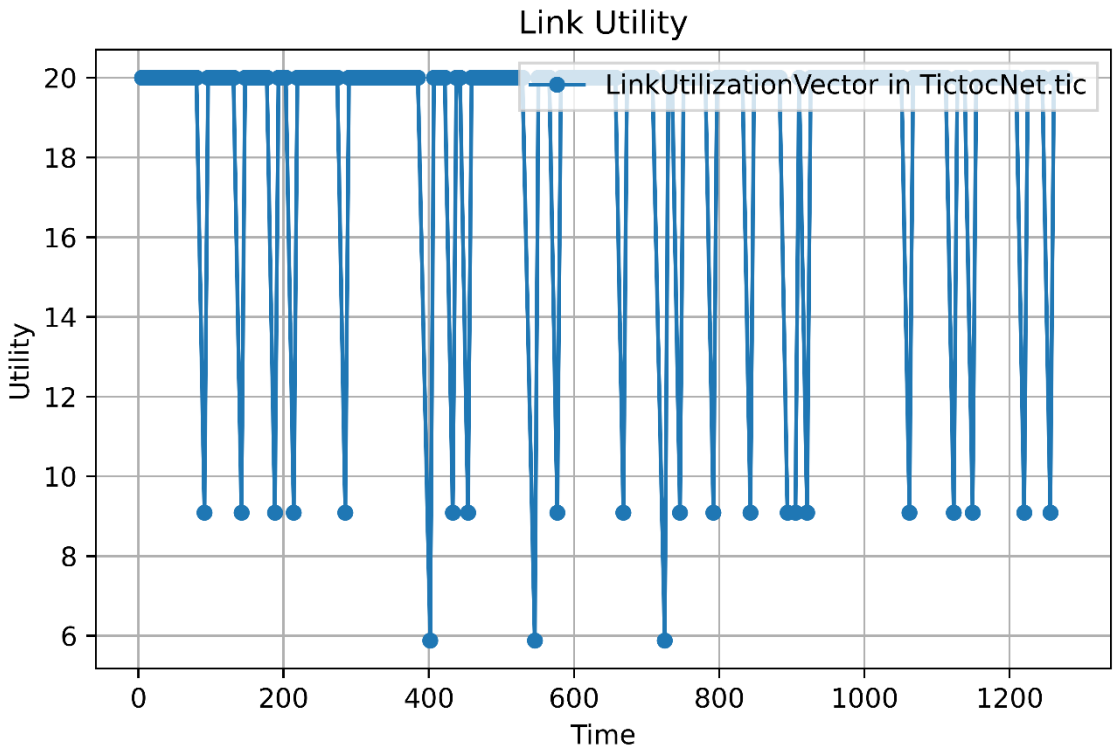
scenario C :

$$P = 0.1\,, t_{prop} = 2s\,, t_f = 1s, t_{out} = 5s$$

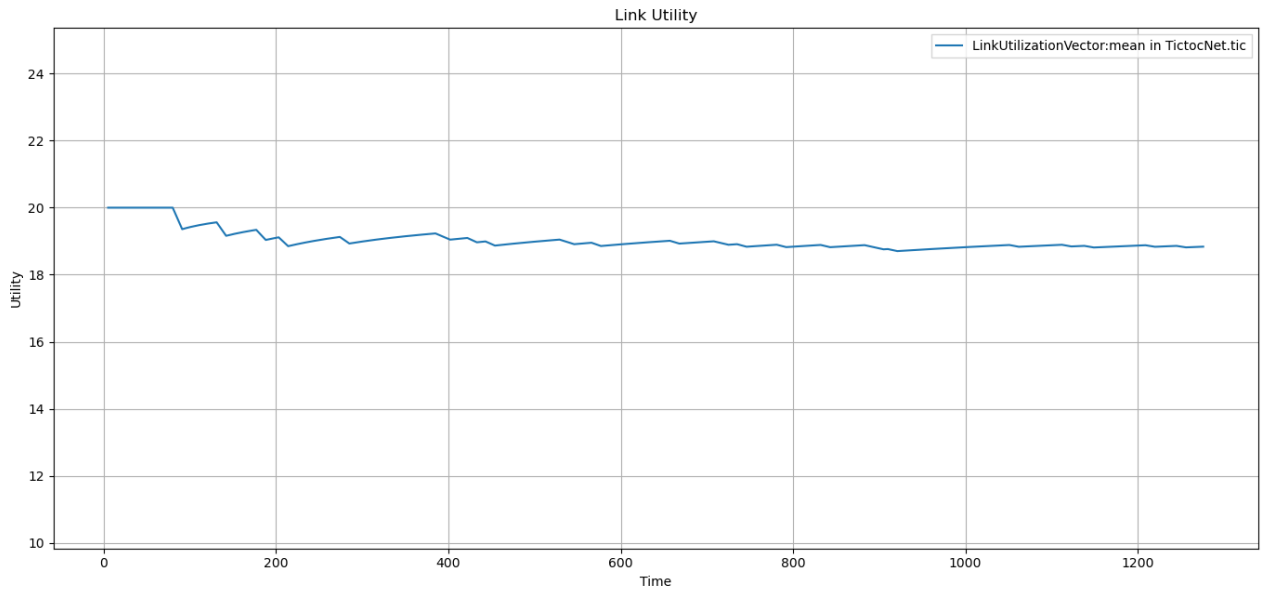
$$a = \frac{t_{prop}}{t_f}\,, \mathrm{U} = \frac{1-p}{1+2a}$$

$$U = 18\%$$

نمودار بهره کانال لحظه‌ایی

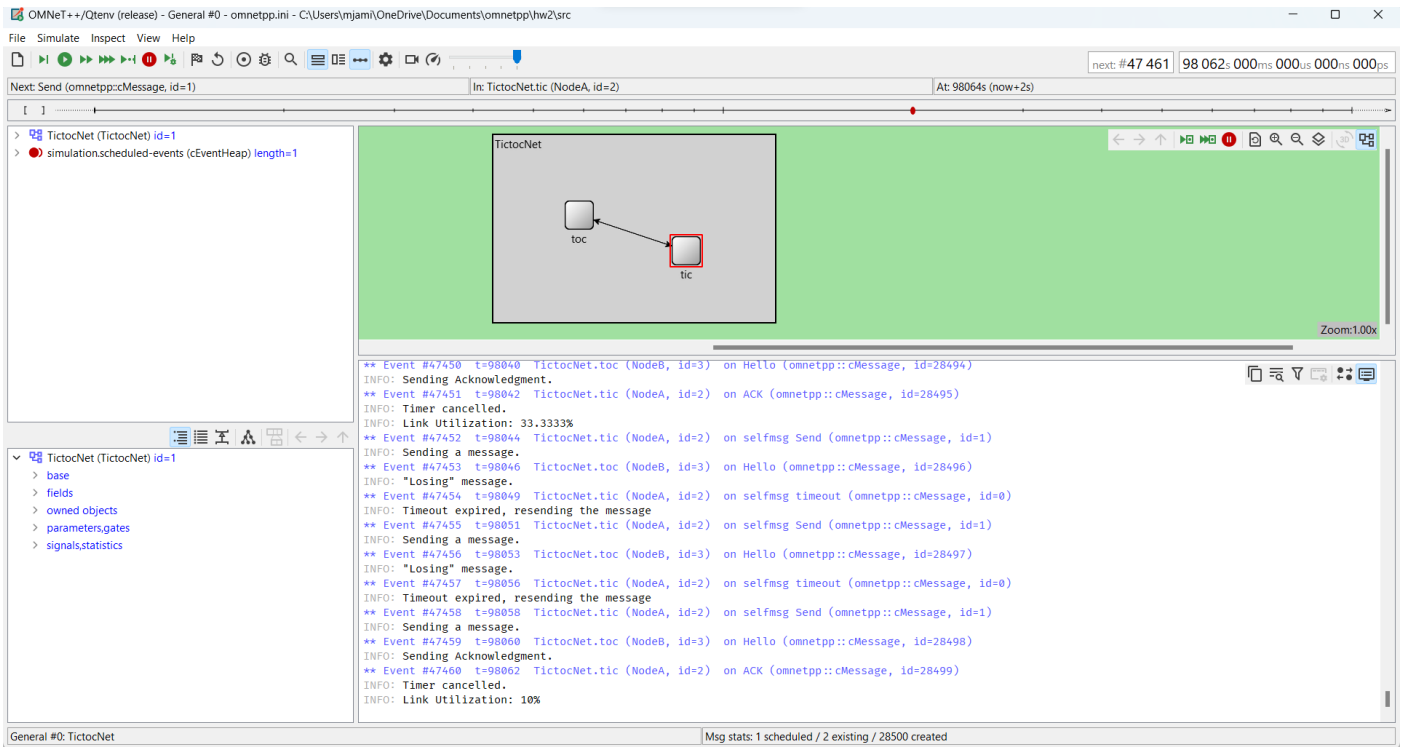


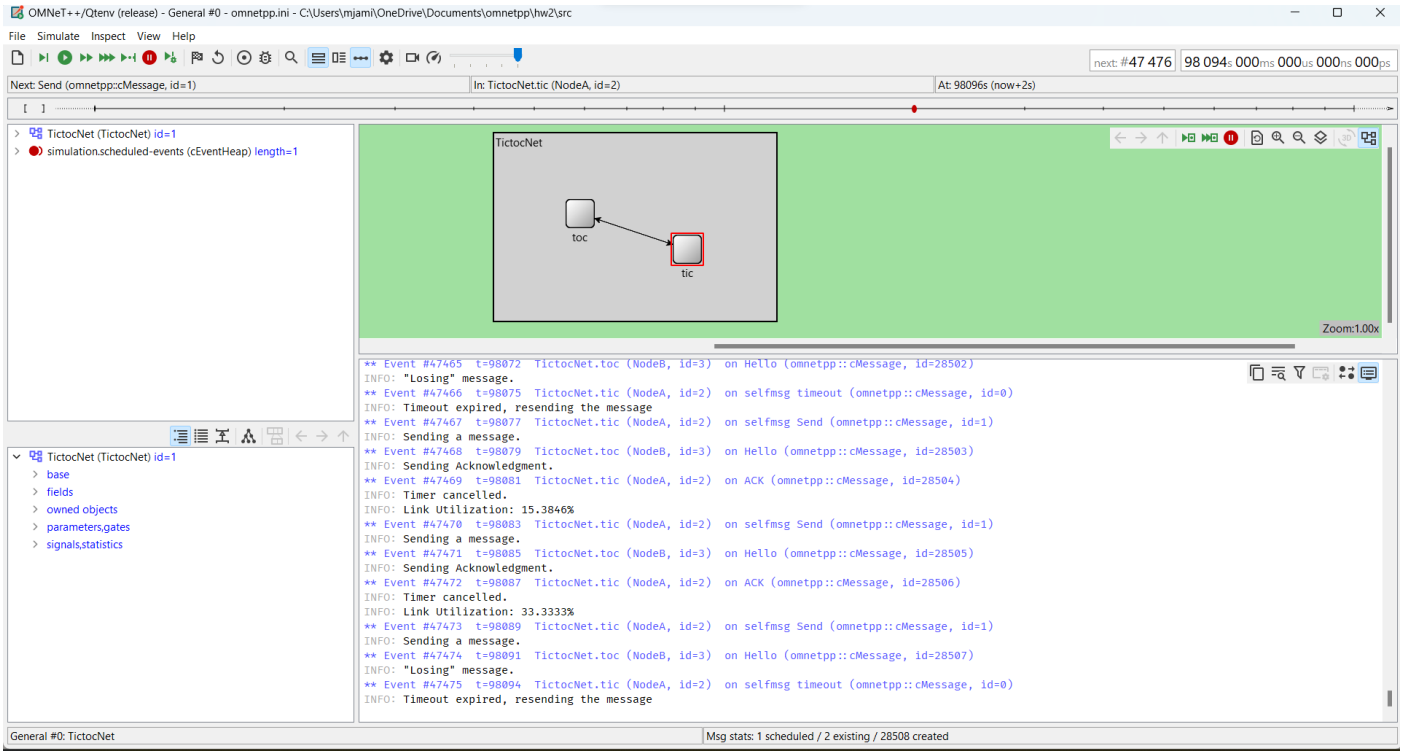
نمودار میانگین بهره کانال



Mean=18.8%

نحوه کار شبکه: (به صورت فیلم نیز موجود است)



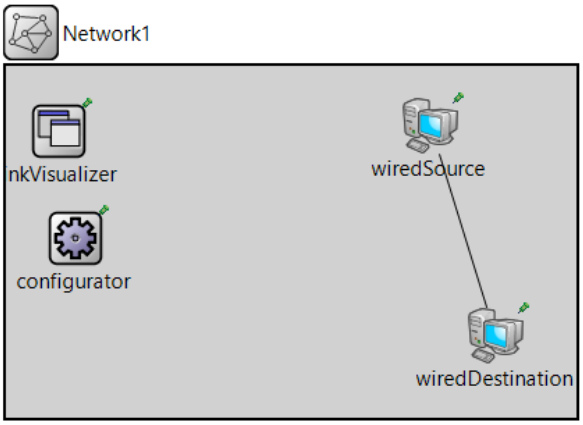


نتیجه: اگر شبکه را طبق پارامترها در هر سناریو اجرا کنیم در طولانی مدت و تابع میانگین بهره به مقدار تئوری نزدیک می‌شود.

سوال ۲

قسمت ۱

طبق خواسته سوال ساختار شبکه را رسم می‌کنیم.




```
import inet.networklayer.configurator.ipv4.Ipv4NetworkConfigurator;
import inet.node.ethernet.Eth100M;
import inet.node.inet.StandardHost;
import inet.visualizer.common.DataLinkVisualizer;
import ned.IdealChannel;


network Network1
{
    submodules:
        linkVisualizer: DataLinkVisualizer {
            parameters:
                @display("p=38,47");
        }
        configurator: Ipv4NetworkConfigurator {
            parameters:
                @display("p=50,124");
        }
        wiredSource: StandardHost {
            @display("p=305,43");
        }
        wiredDestination: StandardHost {
            @display("p=303,299");
        }
    connections:
        wiredSource.ethg++ <--> Eth100M <--> wiredDestination.ethg++;
}


@license(LGPL);
```

توجه: ابتدا از برای اتصال استفاده شد

```
wiredSource.ethg[0] <--> Eth100M <--> wiredDestination.ethg[0];
```

ولی پیام خطا زیر ظاهر شد

(inet::NodeBase)wiredSource: Gate index 0 out of range when accessing vector gate 'ethg$i[]' with size 0

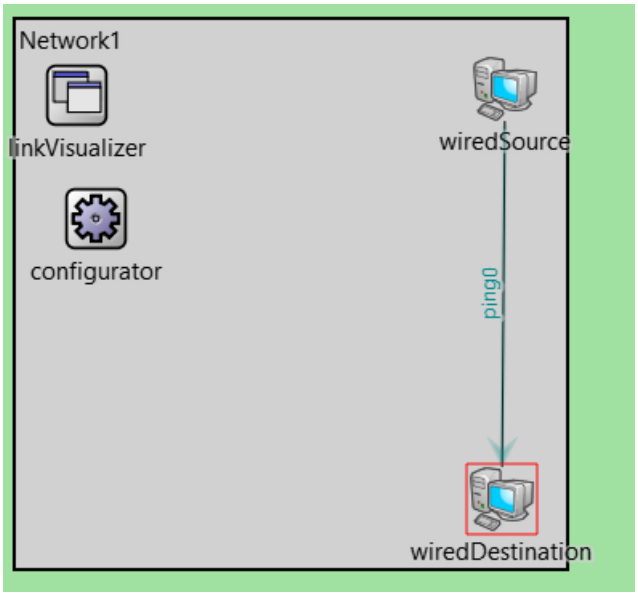
سپس بعد جست‌وجو در اینترنت به صورت زیر بازنویسی شد.

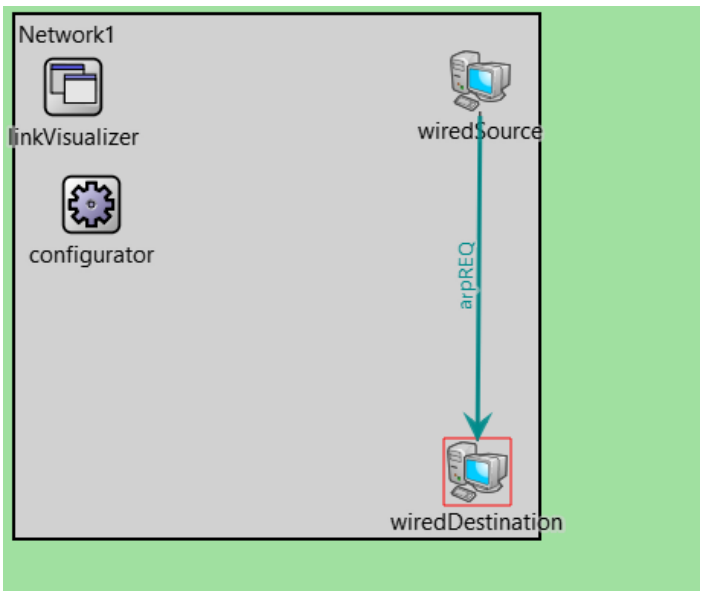
```
wiredSource.ethg++ <--> Eth100M <--> wiredDestination.ethg++;
```

برای فایل INI نیز تنظیمات زیر برای گرفتن پینگ از wiredSource به مقصد wiredDestination تنظیم می‌کنیم و همچنین برای مشاهده نمودن ترافیک و همچنین محو آن تنظیمات را انجام می‌دهیم.

```
[General]
network = Network1
*.wiredSource.numApps = 1
*.wiredSource.app[0].typename = "PingApp"
*.wiredSource.app[0].destAddr = "wiredDestination"
*.linkVisualizer.*.displayLinks = true
*.linkVisualizer.*.fadeOutTime = 1s
```

نحوه کارشبکه (به صورت فیلم هم موجود است).

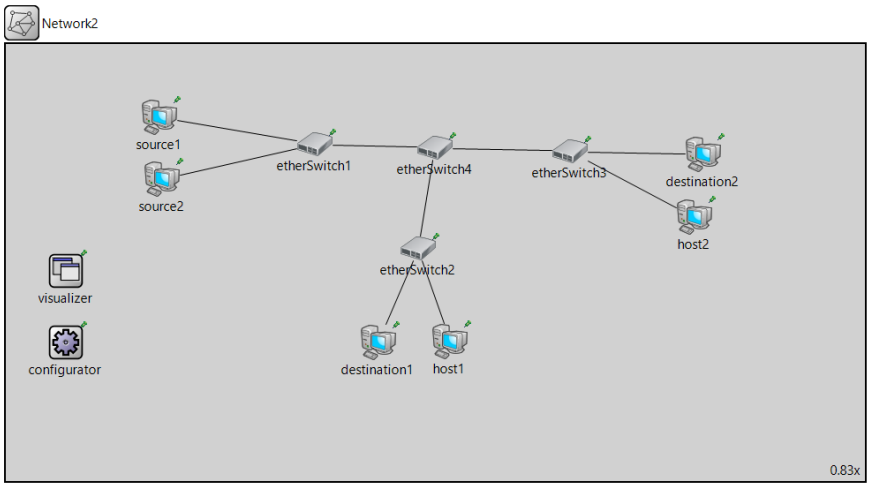




قسمت ۲

a)

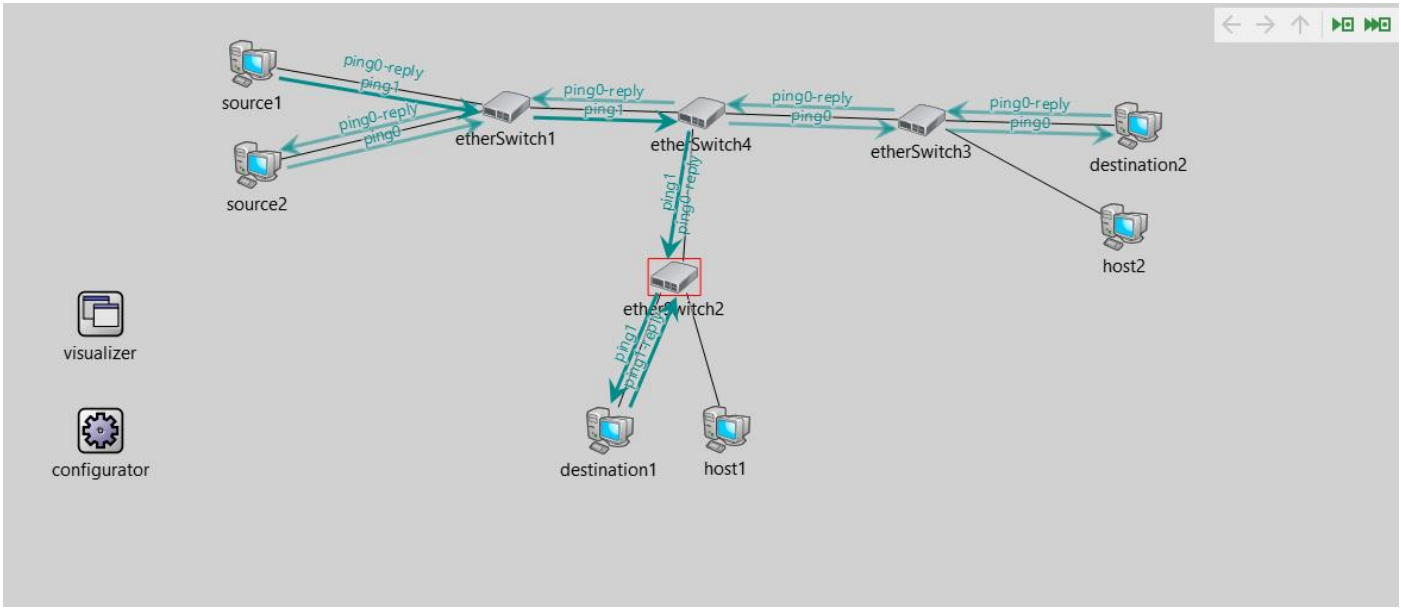
طبق شکل شبکه را رسم کرده و اتصالات را وصل می‌کنیم.



ابتدا Source1 باید Destination1 را پینگ کند. این فرایند را از زمان 0 شروع می‌کند و فاصله مانی هر پینگ را ۲ ثانیه قرارمی‌دهیم تا با پینگ Source2 به مقصد Destination2 که از زمان 1 شروع می‌شود و هر ۲ ثانیه تکرار می‌شود (این کار را برای نمایش بهتر انجام می‌دهیم.) تنظیمات INI را به صورت زیر انجام می‌دهیم.

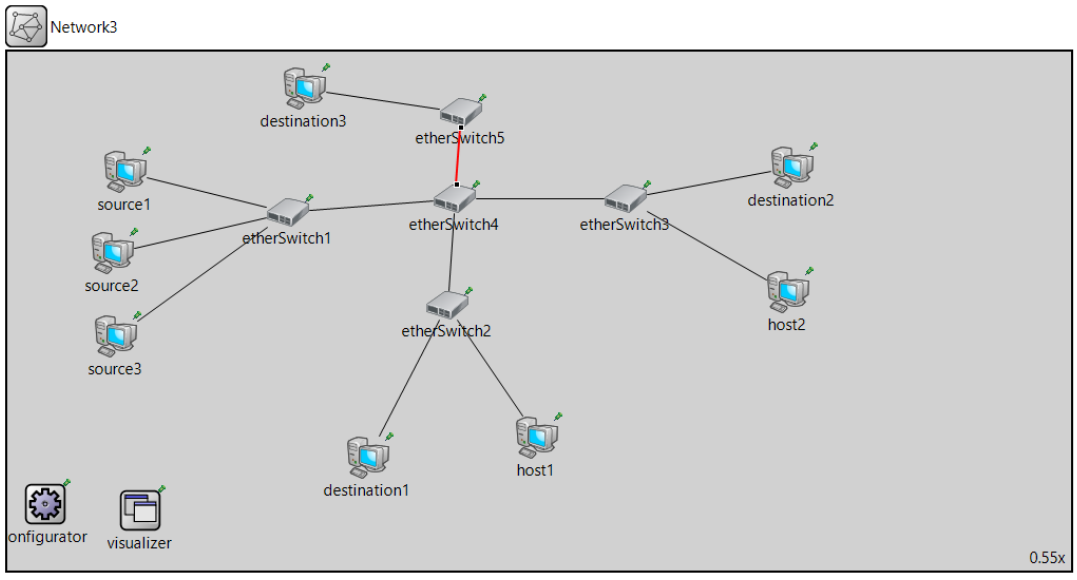
```
[General]
network = Network2

*.source*.numApps = 1
*.source*.app[0].typename = "PingApp"
*.source1.app[0].destAddr = "destination1"
*.source1.app[0].sendInterval = 2s
*.source1.app[0].startTime = 0.0s
*.source2.app[0].destAddr = "destination2"
*.source2.app[0].startTime = 1s
*.source2.app[0].sendInterval = 2s


*.visualizer.*.dataLinkVisualizer.displayLinks = true
*.visualizer.*.dataLinkVisualizer.fadeOutMode = "simulationTime"
*.visualizer.*.dataLinkVisualizer.fadeOutTime = 2s      # fade activity arrows
*.visualizer.*.dataLinkVisualizer.packetFilter = "ping*"
```

نحوه عمل کرد: (به صورت فیلم هم موجود است)



b)

طبق شکل شبکه را رسم کرده و اتصالات را وصل می‌کنیم.



ابتدا Source1 باید Destination1 را پینگ کند. این فرایند را از زمان 0 شروع می‌کند و فاصله مانی هر پینگ را ۳ ثانیه قرارمی‌دهیم و Source2 مقصد Destination2 را پینگ کرده که از زمان 1 شروع می‌شود و هر ۳ ثانیه تکرار می‌شود و Source3 مقصد Destination3 را پینگ کرده که از زمان 2 شروع می‌شود و هر ۳ ثانیه تکرار می‌شود. (این کار را برای نمایش بهتر انجام می‌دهیم) و همچنین میخواهیم ترافیک Source3 در مسیر به مقصدش مشخص باشد. تنظیمات INI را به صورت زیر انجام می‌دهیم.

```
[General]
network = Network3

*.source*.numApps = 1
*.source*.app[0].typename = "PingApp"

*.source1.app[0].destAddr = "destination1"
*.source1.app[0].sendInterval = 3s
*.source1.app[0].startTime = 0.0s

*.source2.app[0].destAddr = "destination2"
*.source2.app[0].startTime = 1s
*.source2.app[0].sendInterval = 3s

*.source3.app[0].destAddr = "destination3"
*.source3.app[0].startTime = 2s
*.source3.app[0].sendInterval = 3s


*.visualizer.*.dataLinkVisualizer.displayLinks = true
*.visualizer.*.dataLinkVisualizer.fadeOutMode = "simulationTime"
*.visualizer.*.dataLinkVisualizer.fadeOutTime = 3s  # fade activity arrows
*.visualizer.*.dataLinkVisualizer.packetFilter = "ping*"

*.visualizer.*.dataLinkVisualizer.nodeFilter = "source3 or etherSwitch{1,4,5} or destination3"
```

نحوه کار شبکه: (به صورت فیلم هم موجود است در فیلم توضیحاتی نیز ارائه شده)