Data Networks
HW 3: MAC Sublayer Simulation With NS3

Dr. Mohammad reza Pakravan

Due on April 24, 2020

Introduction

This homework is about the MAC sublayer and its simulation using the NS3 software. NS3 is a powerful simulation tool used for testing and analyzing various network protocols and scenarios. It is based on the C++ language and provides the user with a predefined API to design a hypothetical network and modify the important parameters. Moreover it provides a python interface via bindings to its C++ api for those familiar with python. Therefor, a background of C or Python programming is needed for an efficient use of the software.

Getting Familiar With NS3

Command line functions

In this section we will get familiar with the basic functions that are needed in almost every simulation and also the routine procedures that have to be done constantly. It is assumed that you have installed NS, NetAnim and Wireshark. To install NS3 and NetAnim you can read N3 docs. The installation process can be tricky, so if you haven't installed it yet, do it ASAP to have more time for the exercises.
First, lets see how we can compile and run a program. Open a text editor and copy the code below, which is a simple program to add two values and print the output:

```
#include "ns3/core-module.h"
using namespace ns3;
NS_LOG_COMPONENT_DEFINE("Lab#2");
int main(int argc, char * argv[]) {
    int a;
    int b;
    CommandLine myCMD;
    myCMD.AddValue("a", "This is a", a);
    myCMD.AddValue("b", "This is b", b);
    myCMD.Parse(argc, argv);
    std::cout << "a+b=" << a + b;
    return0;
}
```

Save the file with the .cc extension. Now, copy the file to the directory below:

```
<NS3_Directory>/ns-X.YZ/scratch
```

Where NS3 Directory is the place where you have installed NS3 and ns-X.YZ is a folder inside the NS3 folder, with X, Y and Z specifying the version you are using. Scratch is the folder where you should put your codes to be able to compile and run them.
Now open a terminal and go to the directory below:

```
<NS3_Directory>/ns-X.YZ/scratch
```

and run this command:

```
./waf --run "scratch/file_name_without_.cc --a=2 --b=5"
```

You should see a line in the output, showing the sum of a and b, which is 7.
Two important notes:

- The basic command for building and running is `./waf --run "scratch/Filename"`. File name should not have the .cc extension.
- In this example, the values of a and b are not specified in the code and are passed to the code via the terminal. This is enabled using the command line functions. Search the web to get more familiar with these functions. Notice also the way the command mentioned above changes to pass the values of a and b. You will need this functionality while testing a similar scenario with different values for a specified set of parameters. Changing the values inside the code can be tiring in the long run.

Network Topology

The first step of every network simulation is adding a number of nodes. We use *NodeContainer* class to add nodes to the scenario.

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
using namespace ns3;
int main (int argc, char *argv[])
{
    NodeContainer nodes;
    nodes.Create (2);
```

The above code show hot to create 2 nodes. The second step of simulation is to define network topology and add links between nodes. If you have t nodes and want to connect these 2 nodes you can use PointToPointHelper.

```
PointToPointHelper pointToPoint;
pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
```

The code adds a link between the 2 nodes created before with *5Mbps* rate and *2ms* delay.
In this exercise we define shared topology to simulate different MAC protocols. For this purpose we use *CsmaHelper* class to create a bus topology between nodes. (Check this).

```
#include "ns3/csma-module.h"
CsmaHelper csma;
csma.SetChannelAttribute ("DataRate", StringValue ("100Mbps"));
csma.SetChannelAttribute ("Delay", TimeValue (NanoSeconds (6560)));
NetDeviceContainer csmaDevices;
csmaDevices = csma.Install (csmaNodes);
```

The code above creates a bus with *100Mbps* rate and *6560ns* delay. These two parameters are important attributes which you should set for each scenario; You must set those according to instructions for each section.

Network Layer Parameters

The network layer is the layer before Data Link layer. The main purpose of this layer is to route the packets from the destination to the source. One of the most popular network layer protocols is the IP. You will gain sufficient knowledge about this layer during the course. In this example, the nodes need IP addresses to communicate with each other. First of all, check the *InternetStackHelper* class and create an instance of this class. Install this instance on the nodes that you've created in the last part. Then, look for the *ipv4AddressHelperclass*, instantiate one and learn how to set a base for the IP addresses. The base is consisted of a base address and a netmask. These are IP address parameters, so if you don't know what they mean, don't worry, although I suggest a little bit of searching. Set the base address to *"1s.tn.u.0"* and the netmask to *"255.255.255.0"*, where stnu is the last four digits of your student number. You must also create an *Ipv4InterfaceContainer* to hold the IP interfaces that you have assigned to the nodes. The code below shows the process to define ip addresses and to setup network layer routing.

```
#include "ns3/internet-module.h"
#include "ns3/ipv4-global-routing-helper.h"

InternetStackHelper stack;
stack.Install(csmaNodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign (csmaDevices);
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

Building Flows

After doing the preceding part, now it is time to determine the application with which the nodes will function. There exist several applications in the NS3 environment. We will work with two of them, The *OnOffApplication* and the *UdpEcho*.

- In the *OnOffApplication*, a node which works with the application sends a Constant Bit Rate (CBR) stream to a destination at regular times, which is called the *OnTime*. At other times, the node sleeps until it reaches another *OnTime*. The interval in which the node sleeps is called the *OffTime*. The node alternates back and forth between these two states. In this part, you should check the documentation and learn how to use the *OnOffHelper* class to make a node have the functionality mentioned above.

- In the *UdpEcho*, you should use the *UdpEchoServerHelper* and *UdpEchoClientHelper* classes to make one of your nodes be an UDP echo client and the other one an UDP echo server. You can set the *MaxPackets*, *Interval* and *PacketSize* attributes to any value that you want. You should use two *ApplicationContainers* for your nodes: one for the client and one for the server. You should also determine a start time and a stop time for both of the *ApplicationContainers*.

For sample code refer to tutorial codes in:

```
NS3 Directory/ns-X.YZ/examples/tutorial/
```

Starting The Simulation

You should just add the following two lines to your code before running it.

```
Simulator::Stop(Seconds(Time));
Simulator::Run();
```

NetAnim, Wireshark and other analysis

NS3 has two types of text outputs: trace files and Pcap files.

- Trace files contain useful information about the events that happen. You can parse them yourself (in MATLAB for example). In order to enable trace files as an output, you can use *AsciiTraceHelper* class.
- Pcap files contain similar information and you can view them using Wireshark or tcpdump. Another useful output type of NS3 is the XML output file, which contains the schematic of the network and the nodes. To enable this kind of output, you need the *AnimationInterface* class. After enabling XML as an output for the simulation, You can view the XML files using NetAnim. To do this simply open a terminal and cd to the NetAnim directory located in the NS3 directory and type *./NetAnim*. Then, Just open *.xml* file in NetAnim. Moreover, to monitor statistics of data flows you can use FlowMonitor. If you want to measure delay, or bandwidth it is much easier than parsing ascii trace files. You can read more about FlowMonitor in the docs. The output files of the simulation can be found inside the NS3 folder, the same place where the scratch folder is located.

Questions

The purpose of this exercise is to compare different MAC protocols obtain the following diagram for Non-persistent CSMA, p-persistent CSMA and ALOHA.

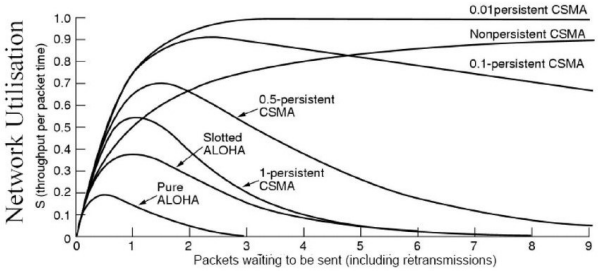

Figure 1: MAC protocols performance comparison

It is assumed that you have learned the NS3 basics which mentioned above. In this part, you'll build a network topology and study performance parameters over different experiments. First, you should build a network topology of 8 csma nodes as described by following figure. These nodes use CSMA protocol for

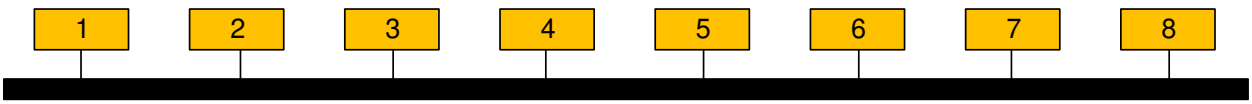

Figure 2: Network Topology

channel access at the link layer. The CSMA link bandwidth is *1024 Kbps* and the oneway link delay is 2

ms. Every node uses IPv4 at the Internet layer. The application layer uses UDP echo application where the echo messages are generated at different data generation rates. There are four different UDP flows in this network as given below.

- Flow 1: Node 1 $\rightarrow$ Node 5
- Flow 2: Node 2 $\rightarrow$ Node 6
- Flow 3: Node 7 $\rightarrow$ Node 3
- Flow 4: Node 8 $\rightarrow$ Node 4

In the first section you need to measure the performance of the CSMA network with respect to following performance metrics:

- Throughput: Average amount of data bits successfully transmitted per unit time.
- Forwarding Delay: Average end to end delay (including the queuing delay and the transmission delay) experienced by the CSMA frames.

Measure the performance of the CSMA protocols in terms of the above metrics and plot a graph (for every metrics, there should be one graph) with respect to $[16, 32, 64, 128, 256, 512, 1024]$ Kbps application layer traffic generation rate.
The implemented CSMA protocol in NS3 is Non-persistent CSMA. In the next section you should implement p-persistent CSMA. The implementation of *CsmaNetDevice* is in file:

```
NS3 Directory/ns-X.YZ/src/csma/model/csma-net-device.cc
```

modify CsmaNetDevice::TransmitStart function to act as p-persistent CSMA protocol, then repeat the previous scenario for following scenarios and collect the results.(Your StdID = abcdefgh)

- $p = min(f, g, h)/10$
- $p = max(f, g, h)/10$

Finally, modify the "TransmitStart" function to act as ALOHA protocol and repeat the analysis. Compare the protocols in terms of "throughput" and "forwarding delay".

Notes

- To implement p-persistent CSMA and ALOHA look at their flowcharts in the course slide carefully. Missing small details lead to simulation failure.
- You should compute the Throughput and Forwarding delay.
- You need to measure the link layer performance or the network performance, not the per node performance. Therefore you should consider all the CSMA frames from all the communication pairs while calculating the performance metrics.
- You need to change the packet size and the time interval between consecutive UDP echo packets to find out the data generation rate. For instance, if packet size is 16 Kb, and time interval is 0.25 sec, then data generation rate is 64Kbps.

What Should I Do?

You must upload (.cc/.tr/.pcap/.xml) files about each part separated in specific folders. Don't forget about edited versions of csma-net-device.cc. You should also answer the questions in your report. Report should include

- brief explanation of the API that you have used in your code
- comments on what you have changed in csma-net-device.cc
- plots and justification of results
- screen shot of network animator

Compress all files and rename the compressed file to STUDENT_ID_HW3.zip.
If you have any questions regarding the problem statement or understanding the concept, feel free to ask in cw forum.