Data Networks
HW 3: MAC Sublayer Simulation

Dr. MohammadReza Pakravan

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.
Note: Mentioned point for each part/task is out of 100.

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. Note that we assume you installed NS3, NetAnim and Wireshark in "Mini HW".
First, lets see how we can compile and run a c++ program in NS3. 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("MAC_HW");
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;
    return 0;
}
```

Save the file with the name "Task_1_1.cc". Now, copy the file to the directory below:

```
<NS3_Directory>/ns3/ns-allinone-X.YZ/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>/ns3/ns-allinone-X.YZ/ns-X.YZ
```

and run this command:

```
./waf --run "scratch/your_code_name --a=10 --b=-4"
```

You should see a line in the output, showing the sum of a and b, which is 6.

Note that to build and run any c++ code in ns3, the basic command is `./waf --run "scratch/Filename"`.

In the previous example, the values of "a" and "b" are not specified in the code and are passed to the code via terminal. 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 is not recommended.

Task_1_1

You should provide a screenshot of the terminal output after running the code. Save the screenshot as "Task_1_1.jpg". Also copy the file "Task_1_1.cc". Finally put both files in your uploading files. (5 Points)

Network Topology

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

```
#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 shows how to create two 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.

```
    #include "ns3/point-to-point-module.h"
    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
    pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
    NetDeviceContainer devices;
    devices = pointToPoint.Install (nodes);
```

The code adds a link of *5Mbps* rate and *2ms* delay, between the two previous added nodes.

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.

```
#include "ns3/csma-module.h"
NodeContainer csmaNodes;
csmaNodes.Create (5);
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 them according to instructions for each section. The code, shows an example of building five nodes and assigning a shared link between them.

Network Layer Parameters

The main purpose of the Network 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 learn about this layer, later in 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. Set the base address to *"1a.bc.d.0"* and the netmask to *"255.255.255.0"*, where "a", "b, "c" and "d" are 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. You will learn about Application layer, later in the course. 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 to 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>/ns3/ns-allinone-X.YZ/ns-X.YZ/examples/tutorial/
```

It is highly recommended to see the aforesaid examples, to understand what to do and where to start.

Starting The Simulation

Till now, we gave you some information about building the environment, now you should just add the following 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. You can use Trace Metric software to analyze trace files.
- 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. You learned how to use NetAnim in "Mini HW". 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 on the internet. The output files of the simulation can be found inside the NS3 folder, the same place where the scratch folder is located.

Questions

802.11 Simulation

In this section, you should modify one of the example codes of NS3 which is "wifi-hidden-terminal.cc". You can find this code in the **examples/wireless/** directory. Create a topology as described in figure 1.
Set the default loss to be 120dB. Also, use IPv4 at the internet layer and assign an IP address of the same range to each node.

(a) Create two flows, $B_1 \longrightarrow A_1$ and $B_2 \longrightarrow A_2$. Measure the throughput once when CTS/RTS is enabled and again when it's not. What is the role of CTS/RTS in this example? Discuss the effect of CTS/RTS using the result. Does CTS/RTS improve throughput? Why? (15 Points)

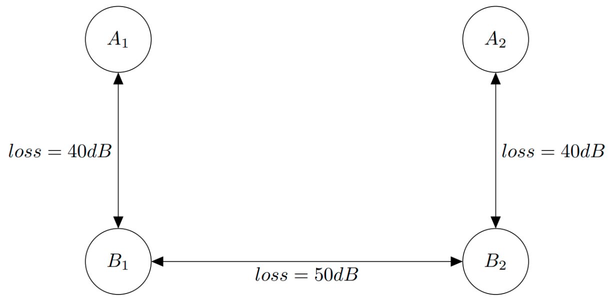

Figure 1: Wifi Network Topology

Task_2_1

Do NOT forget to put your c++ code in your uploading files. Save your code as "Wifi_Part_a.cc". You should provide a screenshot of the simulation result. To do so, take a screenshot of the terminal output after the simulation ends. Save the screenshot as "Wifi_Result.jpg" and put it in your uploading files.
Moreover using *AnimationInterface* in your code, you should provide an animation of the topology. Use the same positions for your nodes as figure 1. Save the animation as "Wifi_Anim.xml" and put it in your uploading files. Also provide couple of screenshots of the animation and save them as "Wifi_Anim_Screen_i.jpg".

(b) Enable pcap tracing and run your code. (5 Points)

Task_2_2

Save the .pcap files as "Wifi_PCAP_i.pcap" and put them in your uploading files.
Then open one of the .pcap files with Wireshark and provide a screenshot of the environment and save it as "Wifi_PCAP_Screen.jpg" and put it in your uploading files.
Also explain the role of each type of packet in your report.

(c) Reverse both flows of part "a" (change $B_i \longrightarrow A_i$ to $A_i \longrightarrow B_i$ for $i = 1, 2$). Repeat part "a" and observe the results. (10 Points)

Task_2_3

Do NOT forget to put your c++ code in your uploading files. Save your code as "Wifi_Part_c.cc". You should provide a screenshot of the simulation result. To do so, take a screenshot of the terminal output after the simulation ends. Save the screenshot as "Wifi_Result_Reverse.jpg" and put it in your uploading files.

Task_2_4

Observe the result and explain if there is any difference in the result after reversing the flow in your report. Explain how reversing the flow, can make a difference in the effect of CTS/RTS.

(d) Repeat part "a" but only reverse the left flow. So the flows is like $A_1 \longrightarrow B_1$ and $B_2 \longrightarrow A_2$. (5 Points)

Task_2_5

Do NOT forget to put your c++ code in your uploading files. Save your code as "Wifi_Part_d.cc". You should provide a screenshot of the simulation result. To do so, take a screenshot of the terminal output after the simulation ends. Save the screenshot as "Wifi_Result_Left_Reverse.jpg" and put it in your uploading files.

Task_2_6

Observe the result and explain if there is any difference in the result after reversing the left flow in your report. Discuss whether it is better to use CTS/RTS in this topology or not.

(e) Assume we have a complete graph among the nodes $B_1, B_2, ..., B_n$ and the loss power between any two of them is 40dB. Besides, for each i, we have a node A_i only connected to B_i with a 30dB link. This topology for $n = 3$ is shown in figure 2.

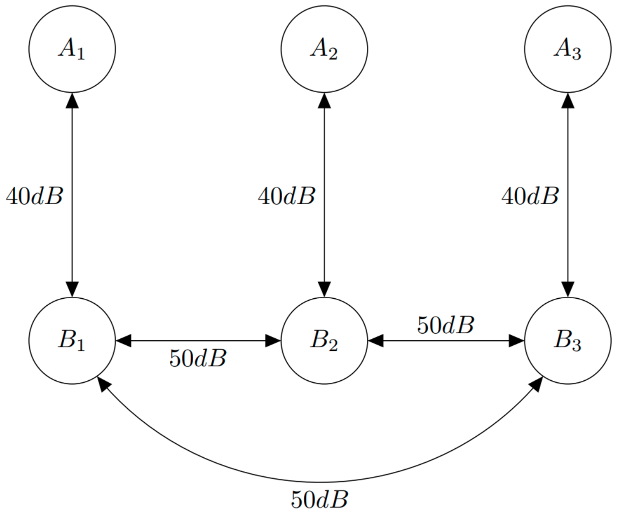

Figure 2: Wifi Network Topology for $n = 3$

Simulate this network for $n = 1, 2, ..., 15$.
Note that the code of this part might be tricky. As a hint do not forget to use loops to run the simulation for different values of "n". (20 Points)

Task_2_7

Do NOT forget to put your c++ code in your uploading files. Save your code as "Wifi_Part_e.cc". The simulation result is very long on the terminal. So you should provide **couple** of screenshots of the simulation result. To do so, take a screenshot of the terminal output after the simulation ends. Save the screenshots as "Wifi_Result_Full_i.jpg" and put them in your uploading files.

Task_2_8

Calculate the average throughput of the nodes for each value of "n" and then provide a plot of the average throughput versus "n". Save the plot as "Wifi_Plot.jpg" and put it in your uploading files.

Discuss the effect of ”n” on the throughput.

Simple Layer 2 System

In this exercise, we are going to get familiar with how to use bridges in NS-3 simulations. Assume that, as a network engineer you are requested to design a network between six nodes in a company. Each node in this network is a bridge. The topology of the network is as figure 3. In this figure, N_i is the i'th bridge and C_i is the i'th computer. The number on each link is the distance (cost) of the link. The IP address of each computer is also plotted. Note that "X" is the last two digits of your student number.
IMPORTANT NOTE: Look carefully at the name of the bridges. They are NOT in order.

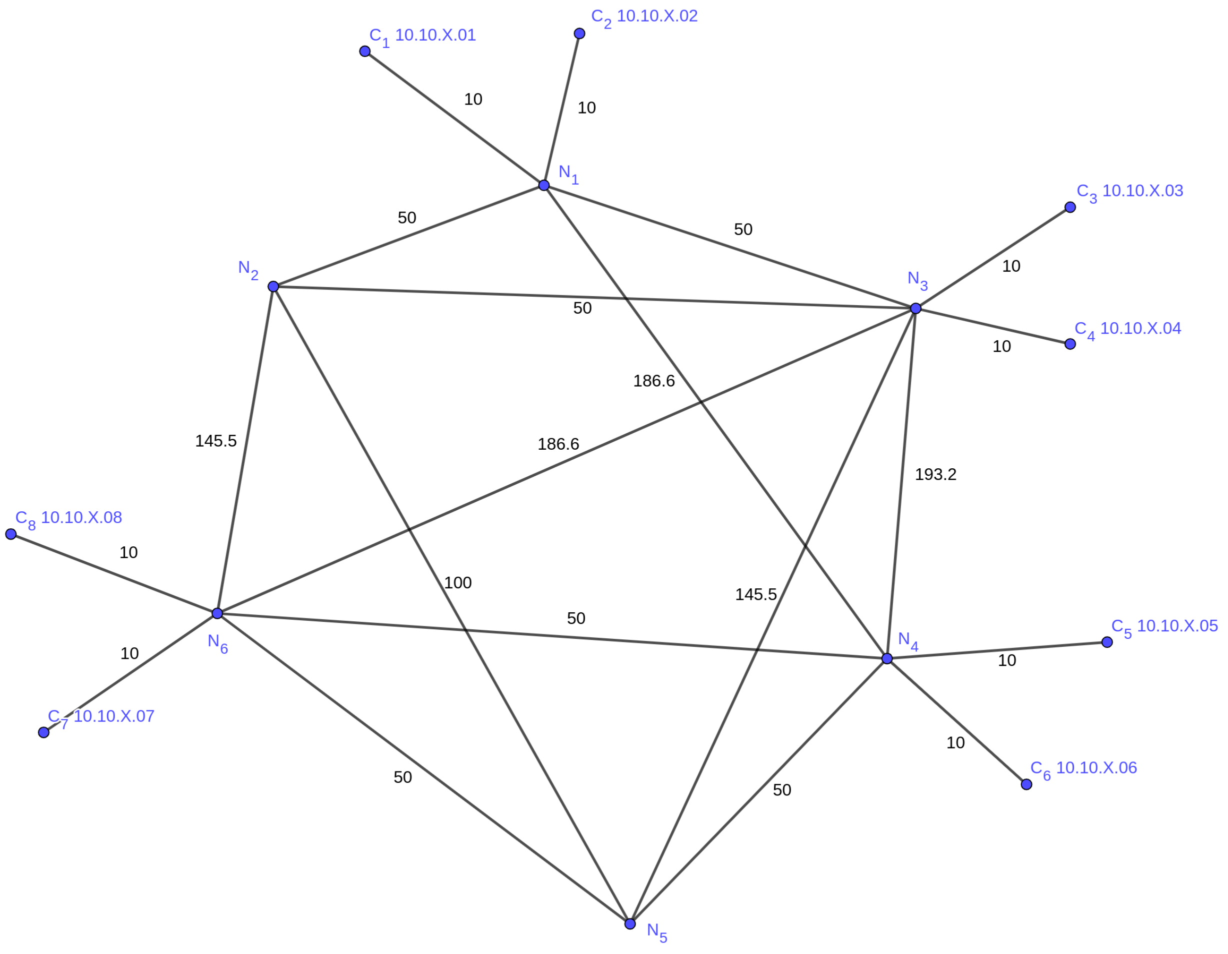

Figure 3: Topology of the System

(a) In the course you have learned that we must prevent existing of any loop in the design. Also in this part, we want to use the minimum cable in our design. These two constraints lead us to use the Minimum Spanning Tree (MST). Implement an algorithm to find the MST.
Hint: Find the MST at the beginning of your code and then define the links which will make the corresponding tree. Feel free to search across the web to find the code which calculates MST. It is acceptable if your code is obtained from web. BUT YOU SHOULD NOT GET THE CODE FROM YOUR FRIENDS.

Note: c++ and python are acceptable in this part. But, we recommend c++, which can be integrated into the next part and you will be awarded extra points if you do it. You can use any algorithm such as Borůvka's algorithm, Prim's algorithm, or Kruskal's algorithm and etc. (15 Points)

Task_3_1

Do NOT forget to put your c++ or python code in your uploading files. Save your code as "L2_Part_a.cc/.py".

(b) Simulate the result of part "a" with NS-3. Each connection uses the CSMA protocol for channel access at the link layer. The CSMA link bandwidth is 1 Mbps, and the one-way link delay is 1µs for each 50m in figure 3.
The application layer uses the OnOff application and generates packets with a 1000 Kbps data rate. Set the protocol to ns3::UdpSocketFactory. There are two UDP flows in this network simultaneously, as given below. Each flow's duration is 10s. (20 Points)
$C_1 \longrightarrow C_6$
$C_2 \longrightarrow C_7$

Task_3_2

Measure the throughput and delay of the UDP flow you have created.

(c) Change the packet generation rate to 10000 Kbps. (5 Points)

Task_3_3

Measure the throughput and delay of the UDP flow you have created.

(d) (Bonus) In this part assume that computers connected to nodes 3, 4, and 6 only communicate with computers connected to node 1. Also, the optimization problem in this part is minimizing the delay. Therefore we use the shortest path tree (SPT) with root node 1.
Implement an arbitrary SPT algorithm. (15 Points)

Task_3_4

Repeat parts b and c. Compare new results with previous ones.

What Should I Do?

You must do all the mentioned Tasks.
Also you should prepare a report and answer the questions. Report should include:

- Brief explanation of the API that you have used in your code.
- Comments on what you have changed in the example codes.
- Plots and justification of results.

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 Telegram comments platform.