Data Networks
HW3: MAC Layer

Dr. Mohammad Reza Pakravan

Introduction

You should have installed NS3 and been familiar with compiling codes using this simulator. In this Homework, we intend to simulate two naive environments to examine what we have grasped in the MAC sublayer.

Disciplined Coding Style

A typical NS3 code contains creating a chain of sublayers. Here we demonstrate the classic code sequence for creating 802.11 and 802.3 environments. You are encouraged to use this manner to prevent further errors in your scripts but feel free to change and add anything that suits your style.

- The first step of every network simulation is adding several 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 shows how to create two nodes. The second simulation step is defining network topology and adding links between nodes. If you have t nodes and want to connect these two 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 two nodes created before with a *5Mbps* rate and *2ms* delay.

- To create a bus, you can use the line below:

```
#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 the specification of *100Mbps* rate and a *6560ns* delay. These two parameters are the main attributes that you should set according to your scenario. You are also encouraged to check this for more details.

- To create a wireless physical layer, you can use *YansWifiPhyHelper* and *YansWifiChannelHelper*.

  Use WifiMacHelper to create MAC layers for WiFi. Set MAC type of stations and access point to *ns3::StaWifiMac* and *ns3::ApWifiMac* respectively.

  Create *NetDeviceContainer* for access points and stations by installing physical layer and MAC layer on the nodes using *WifiHelper*. Set WiFi standard to *WIFI_STANDARD_80211g* and WiFi remote station manager to *ns3::IdealWifiManager* before installation.

  Create a mobility model using MobilityHelper and use ns3::GridPositionAllocator to set the position of the stations. Set mobility model to *ns3::ConstantPositionMobilityModel*

- We are aware that the Network layer is not covered in the course yet, but we recapitulate the code you need to place this layer to cut a long story short. 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. The code below shows the process of defining IP addresses and setting up 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");
  ```

  The base consists 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 searching. Set the netmask to *"255.255.255.0"*.

  For creating a CSMA interface, the codes below will help you:

  ```
  Ipv4InterfaceContainer csmaInterfaces;
  csmaInterfaces = address.Assign (csmaDevices);
  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
  ```

- After doing the preceding parts, 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 that 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 *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 a UDP echo client another one a UDP echo server. You can set the *MaxPackets*, *Interval*, and *PacketSize* attribute 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*.

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

- NS3 has two types of text outputs: trace files and Pcap files.
  - Trace files contain helpful information about the events that happen. You can parse them yourself (for example, in MATLAB). 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 helpful output type of NS3 is the XML output file, which includes 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, open a terminal and cd to the NetAnim directory located in the NS3 directory and type *./NetAnim*. Then, open the *.xml* file in NetAnim

    Moreover, to monitor the statistics of data flows, you can use *FlowMonitor*. Measuring delay or bandwidth is much easier than parsing ASCII trace files if you want to measure delay or bandwidth. You can read more about *FlowMonitor* in the docs. Simulation output files are inside the NS3 folder, the same address as the scratch folder.

*** You would find several helpful tutorial codes in:

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

Questions

Analyzing CTS/RTS (20 Points)

In this part, we use *wifi-hidden-terminal.cc*, which is available in the *<NS3-Directory>/examples/wireless* directory.

- (10 points) Run this code and explain what it does. Justify its results.
- (10 points) Can you give an example in which CTS/RTS adversely affects the throughput, i.e., you can achieve higher throughput when CTS/RTS is disabled? You don't need to program it with NS-3. Just theoretically explain why it works.

Simulating a Wireless Network (40 Points)

The purpose of this exercise is to simulate a wireless network with determined conditions and analyze the throughput of the network by changing other parameters.

- Change the default value of minimum and maximum contention window size using the following command:(Set minCw to 31 and maxCw to 1023)

```
Config::SetDefault ("ns3::Txop::MinCw", UintegerValue (minCw));
Config::SetDefault ("ns3::Txop::MaxCw", UintegerValue (maxCw));
```

- Use NodeContainer to keep track of a set of node pointers. Create a NodeContainer for the access point and another NodeContainer for n stations.
- Use YansWifiPhyHelper and YansWifiChannelHelper to create WiFi physical layer and then create the physical channel by installing these helpers properly.
- Use WifiMacHelper to create MAC layers for WiFi. Set MAC type of stations and access point to ns3::StaWifiMac and ns3::ApWifiMac respectively.
- Create NetDeviceContainer for access point and stations by installing physical layer and MAC layer on the nodes usingWifiHelper. Set WiFi standard to WIFI_PHY_STANDARD_80211g and WiFi remote station manager to ns3::IdealWifiManager before installation.
- Create mobility model using MobilityHelper and use ns3::GridPositionAllocator to set position of the stations. Set mobility model to ns3::ConstantPositionMobilityModel.
- Use InternetStackHelper to aggregate UDP functionality to existing Nodes. Install the InternetStackHelper on all nodes.
- Create Ipv4InterfaceContainer by assigning IP address to nodes using Ipv4AddressHelper. Set the base to "192.168.0yz.0" where "yz" is the last two digits of your student number.
- Create an ApplicationContainer and use OnOffHelper to add constant bit rate applications to the nodes. Set the protocol to ns3::UdpSocketFactory. Set the address parameter of OnOffHelper, which is the address of the remote node to send traffic to, to the address of access point.
- In order to find the throughput of the network in different scenarios, you may enable flow monitor on all nodes using FlowMonitorHelper. For a working example, see *wifi-hidden-terminal.cc*.

Create a wireless network with 8 stations and one access point. Create UDP traffic from only one of these stations to the access point. Set UDP packet size to 15000 bytes. Plot the network throughput versus the data rate of UDP flow.

Hint1: For OnOffHelper, you can use the following code:

```
ApplicationContainer cbrApps;
uint16_t cbrPort = 12345;
OnOffHelper onOffHelper ("ns3::UdpSocketFactory", Address (InetSocketAddress (AP_ADDR, cbrPort)));
onOffHelper.SetAttribute ("PacketSize", UintegerValue (packetSize));
onOffHelper.SetAttribute ("OnTime",  StringValue ("ns3::ConstantRandomVariable[Constant=1]"));
onOffHelper.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Constant=0]"));

// Setting the Attributes of the Flow
onOffHelper.SetAttribute ("DataRate", StringValue ("DATA_RATE_bps"));
onOffHelper.SetAttribute ("StartTime", TimeValue (Seconds (1.000000)));
cbrApps.Add (onOffHelper.Install (STATION_NODE);
```

Hint2: For plotting, you can use this link or use any other methods you prefer, e.g., Excel. It's entirely up to you; just make sure to describe it in your report.

Data Link Layer Switching (60 Points)

*** Take a look at the files *<NS3-directory>/src/bridge/examples* to learn how to use bridges in NS-3.

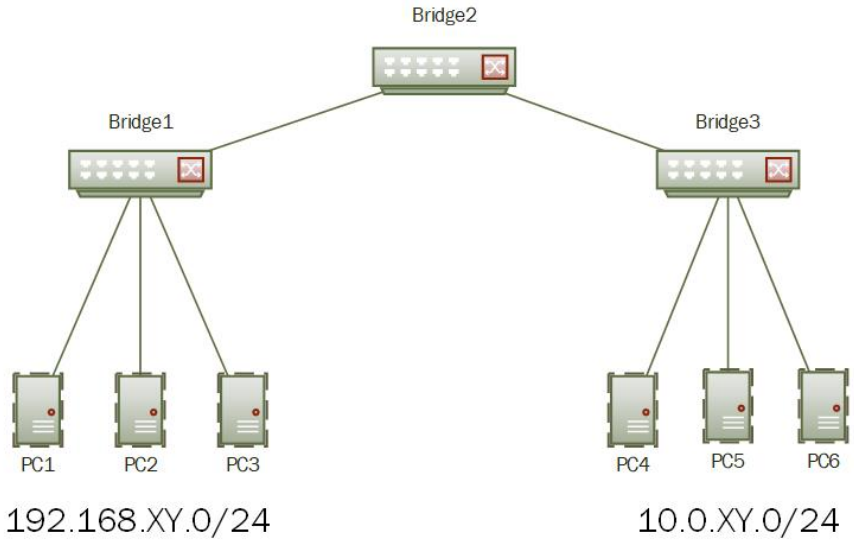

Figure 1: Network topology of servers and bridges

For this part, you'll have to create a topology of nodes and bridges as in figure 1. These nodes use the CSMA protocol for channel access at the link layer. The CSMA link bandwidth is 1 Mbps, and the oneway link delay is 1μs. Every node uses IPv4 at the Internet layer and owns an IP address as in figure 1. (XY is the last two digits of your student number) The application layer uses the OnOff application and generates

packets with a 1000 Kbps data rate. Set the protocol to ns3::UdpSocketFactory. There is one UDP flow in this network, as given below.

- PC1 → PC6

Now answer the following question:

(a) (10 Points) What are the attributes of *BridgeNetDevice* that you can change? Discuss *ExpirationTime* specifically and explain what problems might happen if it is set to an extremely short or very long time.

(b) (20 Points) Measure the throughput of the UDP flow you created. Then add another CSMA link from *bridge1* to *bridge3* and measure the throughput again. Compare the results.

(c) (10 Points) What will happen in the last part if we replace all existing links with 100Mbps and 1μs new links?

(d) (20 **Bonus** points) Show all pairs of <MAC Address, Output Port> that are saved in *bridge1* on every 500ms interval. For this part you should have two flows: PC1 → PC6 & PC4 → PC3. The former must start from time 0.0 seconds and the latter from 1.0 seconds. Also, the simulation must run for 5 seconds.

Note: Don't forget to enable pcap tracing and upload the generated pcap files for this part. However, you don't need to upload them if they are large (explain in your final report).

What Should I Do?

You must upload (.cc/.tr/.pcap/.xml) files about each part separated in specific folders. You should also answer the questions in your report. The report must include:

- a brief explanation of the API that you have used in your code
- plots and justification of results

Aggregate all files and make sure to name the compressed file to DN_MAC_#Student-ID.zip. If you have any questions regarding the problem statement or understanding the concept, feel free to ask in Quera.