ECE 6110 Lab Assignment 1: Topologies

Before starting the assignment, you should download the latest version of ns-3 (at the time of this lab creation, ns-3.36.1) from https://www.nsnam.org/ and go through the ns-3 tutorial, Chapters 4-7. Chapter 4 takes you step-by-step through the process of downloading and installing ns-3. Chapters 5-7 provide the concepts you will need to complete this assignment.

Note for Windows users: Windows users should install VirtualBox and Linux prior to downloading ns-3.

Notes on Random Numbers in ns-3: Random number generation is very important in simulations and, in particular, the ability to exactly reproduce the same "random" scenario multiple times is often required. This is handled automatically by ns-3 but requires the programmer to use the built-in ns-3 random number generation methods. Information about using ns-3 random number generation can be found in the Doxygen documentation on the ns-3 Web site under Modules->Core->Random Variables. Please use these built-in ns-3 methods instead of other random number generators such as rand() from the C standard library. Note that there is no need to set a seed for the ns-3 random number generator.

Turn-in Instructions: Create a separate folder for each part of the assignment inside a main folder named "Lab1_<your_last_name>_<your_first_name>". Then create a single tarball of the main folder and all of its subfolders and submit it in Canvas.

Important programming instructions: All code must be hand typed in your editor of choice. No auto-formatting is allowed to change the text that you type in manually. Also, make sure to use the exact program name and command-line parameter names specified in each part. You will get points deducted if you do not follow these instructions even if your code is functionally correct!!

Alternate application for extra credit: This lab has been designed to closely follow the ns-3 tutorial, which uses the UdpEchoServer and UdpEchoClient applications to send packets and get replies back. This can also be done using ping, which uses the ICMP protocol that is part of the IP protocol stack. For extra credit, implement the entire lab using ping instead of UdpEchoServer and UdpEchoClient. For help using ping in ns-3, see V4Ping in the ns-3 Doxygen documentation and look at the example program in $NS_DIR/src/csma/examples/csma-ping.cc, where $NS_DIR is the directory in which you installed ns-3.

Part 1: Working with Point-to-Point Links

In this part, you will modify the example (first.cc) described in Chapter 5 of the tutorial. You will set up multiple client nodes, each one with a point-to-point link to the server. The number of clients and the number of packets sent by each client should be settable through command-line parameters up to a maximum of 5 clients and 5 packets per client. In the absence of a command-line argument, your program should assign a default value of 1 to those parameters. Name your program "lab1-part1.cc" and name the command line parameters "nClients" and "nPackets".

An example of the network topology with 3 clients is shown below, where n0 is the server and n1, n2, and n3 are the clients.

```
//
//       10.1.2.0          10.1.1.0
// n2 -------------- n0 -------------- n1
//    point-to-point     point-to-point
//                    |
//                  p |
//                  2 | 10.1.3.0
//                  p |
//                    |
//
//                   n3
```

Important notes:

1. The clients' requests should all be sent to the same IP address, which following the set-up in first.cc is 10.1.1.2. Since the clients after n1 are on different subnets, this will require the use of a concept introduced later in the ns-3 tutorial (in the second.cc example) to make sure that the requests from all clients reach the server.
2. How to set up and use command-line parameters are also introduced in the second.cc example.
3. Set the start times for the UdpEchoClient apps on the clients to be different random times between 2 and 7 seconds. Set the stop time for the UdpEchoClient apps, the UdpEchoServer app, and the simulation to 20 seconds.
4. Use the SetAttribute() method to change the port number for the UdpEchoServer app to 15.
5. All other parameters should have the same values as in first.cc, i.e. the data rate and delay of each point-to-point link, and the interval and packet size for each UdpEchoClient.

Turn-in instructions: In your Part 1 subfolder, include the source code of your program, a screen shot showing the command line and the output when there are 5 clients and 4 packets per client, and a text file containing the output for the same case.

Part 2: Working with an Ethernet Network

In this part, you will modify the example (second.cc) described in Section 7.1 of the tutorial. You will add a second point-to-point link from the last node on the CSMA network to another node, which will contain the UdpEchoServer. The number of packets sent by the UdpEchoClient should be settable through a command-line parameter up to a maximum of 20 packets. In the absence of a command-line argument, your program should assign a default value of 1 to the number of packets sent. Set the parameters of the second point-to-point link (delay and data rate) to be the same as the first point-to-point link. Name your program "lab1-part2.cc" and name the command-line parameters "nCsma" and "nPackets".

An example of the network topology with 4 CSMA nodes (3 "extra" nodes) is shown below, where n0 is the client and n5 is the server.

```
//
//       10.1.1.0                          10.1.3.0
// n0 -------------- n1   n2   n3   n4 -------------- n5
//    point-to-point  |    |    |    | point-to-point
//                    ================
//                      LAN 10.1.2.0
```

Important note: Make sure to extend the stop times of the client and server to be long enough for the maximum simulation run of 20 packets.

Modify the original second.cc to send up to 20 packets in a similar manner. Now, for a run of 10 packets and 5 CSMA nodes (4 "extra" nodes), plot the end-to-end delay of a request-response pair vs. the packet number for your modified topology and the original topology (10 data points for each topology). Answer the following questions related to the delays. You will need to do some analysis of the PCAP files to answer these questions. Collect the same PCAP files as in the original second.cc program and any additional ones you need to help you answer these questions:

1. If your program is running correctly, you should notice that the end-to-end delay for the first request-response is significantly longer than the subsequent ones. For this question, focus on the delays after the first one. Since the point-to-point link delays are 2 ms and the CSMA delays are in microseconds, you might expect the end-to-end request-response delay to be around 4 ms for

the original topology and around 8 ms for the modified topology. Is this what you see? If not, explain the discrepancy focusing solely on the characteristics of the point-to-point links and ignoring other delays. Provide a calculation that gives a better rough estimate of end-to-end delays for the two topologies.

2. Now, consider the delay of the first request-response sequence. Looking carefully through the relevant PCAP files, state where this additional delay occurs and explain why it happens .

Turn-in instructions: In your Part 2 subfolder, include the source code of your program with the modified topology and the following for your 10-packet runs:

- a screen shot showing the command line and the output for your modified topology,
- a text file containing the output for the same case,
- all PCAP files that you captured for the same case (at a minimum, the 3 PCAPs captured by the original second.cc program),
- the end-to-end delay vs. packet number plots for the original topology and your modified topology, and
- the answers to the two questions above.

Part 3: Working with WiFi Networks

In this part, you will modify the example (third.cc) described in Section 7.3 of the tutorial. You will replace the Ethernet LAN in the example with a second WiFi network having the same number of nodes as the first WiFi network. You should create a second physical channel for the second WiFi network. The STA nodes for the second network should move within the same region as the nodes in the first network. (This models a scenario where there are 2 APs in an area operating on different non-overlapping wireless channels, nodes in the area can associate to either of the APs, and communications on the two networks can coexist without interference because they operate on different frequency bands.) As in Part 2, modify your program to have the number of packets as a command-line parameter up to a maximum of 20 packets and make sure all end times in your simulation are long enough to accommodate that many packets. An example of the modified topology with 3 WiFi STA nodes per network is shown below, where n7 is the client and n4 is the server. Name your program "lab1-part3.cc" and name the command-line parameters "nWifi" and "nPackets". (Note that nWifi should be no greater than 9 here, instead of 18 as it was in third.cc, since the total number of WiFi nodes is 2*nWifi.)

```
//
//   Wifi 10.1.3.0
//                 AP
//  *    *    *    *
//  |    |    |    |    10.1.1.0
// n7   n6   n5   n0 -------------- n1   n2   n3   n4
//                     point-to-point  |    |    |    |
//                                     *    *    *    *
//                                     AP
//                                        Wifi 10.1.2.0
```

As in Part 2, plot the end-to-end request-response delay vs. packet number for your modified topology for a run with 10 packets and 4 WiFI STA nodes per network. What is different about the end-to-end delays for packets 2 through 10 as compared to the results from Part 2? What do you think explains this difference? (*Hint:* Why do you think there is no parameter to set the data rate on the WiFi channels the way there was for both point-to-point and CSMA channels?)

Turn-in instructions: In your Part 3 subfolder, include the source code of your program with the modified topology, a screen shot showing the command line and the output for the 10 packet and 4 STA nodes per network case, a text file containing the output for the same case, the end-to-end delay vs. packet number plot, and the answers to the two questions above.