ECE 6110 **2017 Fall Semester**

Project 1 – TCP Throughput Measurements

Assigned: September 6, 2017 Due: September 15, 2017

For this project, we will use ns-3 to measure TCP throughput through a given topology for a variety of TCP variants. You should complete the following steps and submit your results electronically on Tsquare as described below.

1. Create a new program in your ns-3 scratch folder named p1.cc for c++ or p1.py for python.

2. Create the network topology in ns-3 as described below:
   A) The topology will be made up of two PointToPointStar topologies connected together via their hubs.
      1. P2P Star topologies
         a) P2P star topology helpers can be found in src/point-to-point-layout/model/point-to-point-star.cc/.h. grep for the class name to find examples of their usage.
         b) The number of spokes on each star should be setup in your program to default to 8. However you must allow for the user to change this by passing –nSpokes=X where X is the number of spokes to create on each star.
         c) The data rate for the P2P star net devices should be set to **5Mbps**
         d) The channel delay for the P2P star channels should be set to **10ms**
      2. Hub connection
         a) Use a PointToPointHelper to create a link between the star's hubs
         b) The data rate should be set to **1Mbps**
         c) The channel delay should be set to **20ms**

3. The scenario we will be setting up is one where one of the stars will contain all of the data senders and one of the stars will contain all of the data receivers/sinks. Each sender will send data to the corresponding sink on the other star. Each node of the star will contain one source or sink depending on which star it is on.
   A) Data Senders and Receivers
      1. Use the PacketSinkHelper to install an instance of a packet sink on each of the spokes on the receiving star. Have all of these applications start at time 1.0 second.
      2. Install an instance of BulkSendApplication on each of the sending nodes. Set all of these applications to start at time 2.0 seconds and have them send data to the packet sink on the corresponding spoke on the receiving star.

4. The first line of code in your main function should be:
   SeedManager::SetSeed(1); //Or the corresponding python call

   This will ensure that all of our results are the same

5. Have your simulation run for 60 seconds

6. Once the simulation has completed, prior to calling Simulator::Destroy(), you'll need to collect data.
   A) Assuming your packet sink applications are stored in a variable called sinkApps you can use the following code to do this. NOTE: The format of the output (i.e. tabs/columns) must match! (totalRx should be an integer)

```
for(uint32_t i = 0; i < nSpokes; ++i) {

     Ptr<PacketSink> sink = DynamicCast<PacketSink>(sinkApps.Get(i));
     uint32_t bytesReceived = sink->GetTotalRx();
     totalRx += bytesReceived;
     std::cout << "Sink " << i << "\tTotalRx: " << bytesReceived * 1e-6 << "Mb";
     std::cout << "\tThroughput: " << (bytesReceived * 1e-6) / endTime << "Mbps" <<
            std::endl;
}

std::cout << std::endl;
std::cout << "Totals\tTotalRx: " << totalRx * 1e-6 << "Mb";
std::cout << "\tThroughput: " << (totalRx * 1e-6) / endTime << "Mbps" << std::endl;
```

Once you have your simulation working, you now need to examine the different TCP implementations. See Lines 274-329 of examples/tcp/tcp-variants-comparison.cc for an example of setting up each of the variants. Run your simulation script for each of the variants and place the output (produced from code above) into a .txt file. The number of spokes for these runs should be the default of 8. The files should be named like so:

TcpWestwoodPlus8.txt
TcpIllinois8.txt
etc.

Repeat this procedure with the number of spokes set to 1, 4, 16 and 32. In the end, for each variant, you should have an output produced from 1 spoke, 4 spokes, 8 spokes, 16 spokes and 32 spokes using the above naming convention. Create a graph using the format and program of your choice showing the total throughput for each of the variants for each spoke size. The graph should allow someone to determine the best variant for each spoke size. Save your graph as a jpeg named results.jpg.

Your submission should contain all of your output files, results.jpg and your .cc or .py file. Compress, using zip or tar.gz all of the files into a archive named p1.zip/.tar.gz and attach it as your submission.

If you get stuck, remember to look for examples, search online, and ask questions in class or via Piassa. Once you get the script working, you will need to perform a large number of runs so it is recommended that you start early.

Bonus: The assignment is worth 100 points. If you submit working scripts for c++ and python you will receive 25 bonus points (125/100).

Make sure you use the correct formatting for the output files and name them correctly.

PointToPointStarHelper documentation
https://www.nsnam.org/doxygen/classns3_1_1_point_to_point_star_helper.html

BulkSendApplication documentation

https://www.nsnam.org/doxygen/classns3_1_1_bulk_send_application.html

PacketSinkApplication documentation
https://www.nsnam.org/doxygen/group__packetsink.html