Data Communication Networks, Lab #5

Instructor: Dr. MohammadReza Pakravan

TCP Congestion Control

1 Introduction

Objective of this homework is to create and analyse TCP connections and its congestion control mechanism using NS3. TCP provides a connection oriented, reliable, byte stream service. The term connection-oriented means the two applications using TCP must establish a TCP connection with each other before they can exchange data. It is a full duplex protocol, meaning that each TCP connection supports a pair of byte streams, one flowing in each direction. TCP includes a flow-control mechanism for each of these byte streams that allow the receiver to limit how much data the sender can transmit. TCP also implements a congestion-control mechanism.

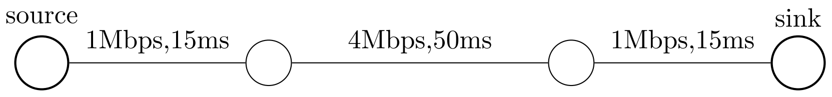
To create a reliable connection, a TCP agent expects to receive an Ack for each packet it sends to find out about the possible lost packets and retransmit them if necessary. Each transmitted packet contains a sequence number. Acknowledge packets contain the sequence number of the packets that the receiver expects. The sender uses these sequence numbers to find out which packets were received without any problem and which packets are lost in the channel. A TCP packet is considered to be lost if three repeated Acks for the same packet arrive at the source or a Time-Out occurs for the timer which was set at the time the packet was transmitted.

1. TCP Reno is a congestion control algorithm, which improves TCP Tahoe (the congestion control you have learned in class). Search the web to find about TCP Reno and explain what is the advantage of TCP Reno over TCP Tahoe. (you should at least describe the concept of Fast Recovery)

2 Simulation of TCP

2.1 Creating a single TCP connection

Create the following topology[1] in NS3 using the commands you've learned from previous labs. You may need to use classes such as *PointToPointHelper*, *NodeContainer*, *InternetStackHelper*, *Ipv4InterfaceContainer*, *Ipv4AddressHelper* and *NetDeviceContainer*. Set the base address to "*st.nu.1.0*" for gateways, "*st.nu.2.0*" for sources,"*st.nu.3.0*" for sinks and the netmask to "*255.255.255.0*", where *stnu* is the last four digits of your student number.



[1]See "tcp-bulk-send.cc" example for a simplified example.

Use the following line of code to set default congestion control mechanism as *TcpNewReno*:

```
Config::SetDefault ("ns3::TcpL4Protocol::SocketType", TypeIdValue (TcpNewReno
  ::GetTypeId ()));
```

You need to setup IP routing tables to get total IP-level connectivity. This is possible using the following line of code:

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

Now its time to determine the application with which the nodes will function. In previous exercises you have been familiar with several applications in the NS3 environment. In order to create a TCP connection between sources and sinks you may use *BulkSendHelper*

```
BulkSendHelper source ("ns3::TcpSocketFactory",InetSocketAddress (interface.
  GetAddress (i,addressIndex), port));
```

Where *i* is the interface number of an Ipv4 interface and *addressIndex* is the index of Ipv4InterfaceAddress. Then Use an *ApplicationContainer* to assign this application to the source node.

Create a *PacketSinkApplication* using *PacketSinkHelper* and install it on sink node with *Application-Container*.

You may run the simulation now. In order to analyse the results with wireshark, you should enable pcap tracing. Pcap traces in the NS3 system are of the form "<prefix>-<node id>-<device id>.pcap". You should also save the congestion window size in your simulation[2].

Note: Set duration of all simulations to 100s. In all parts of this lab, all TCP connections should end 3 seconds before the end of simulation.

1. Open the pcap file related to one of two gateway devices in wireshark. Explain the meaning of SYN packet at the beginning of the TCP connection.
2. Open *Conversations* window from *Statistics* menu of wireshark. Report the throughput of TCP connection.
3. Plot *Time Sequence* and *Throughput* graphs using *TCP Stream Graphs* sub-menu of *Statistics* menu. Explain the meaning of each plot and find the slow start phase on each graph.
4. Plot the congestion window size obtained from NS3 and compare it to the wireshark graphs.

2.2 Adding Error Model

In this part we want to add an error model to the middle link in the previous step. Error models are used to indicate that a packet should be considered to be errored, according to the underlying (possibly stochastic or empirical) error model. NS3 has various stochastic and deterministic error models. For stochastic error models we need to create random variables according to which the packets might be lost. A uniform random variable in [min, max] can be created using the following code:

```
Ptr<UniformRandomVariable> uv = CreateObject<UniformRandomVariable> ();
uv->SetAttribute ("Min", DoubleValue (min));
uv->SetAttribute ("Max", DoubleValue (max));
```

First error model is rate error model. Create a member of *RateErrorModel*. Create a uniform random variable (for this part you don't need to specify min and max since we need the default value which

[2]You may find how to save congestion window size in example "tcp-variants-comparison.cc"

is 0.0 and 1.0) and set random variable of *RateErrorModel* to this value. Set its unit to *RateErrorModel::ERROR_UNIT_PACKET* and set its error rate[3]. Then you need to add this error model to the attributes of point to point link.

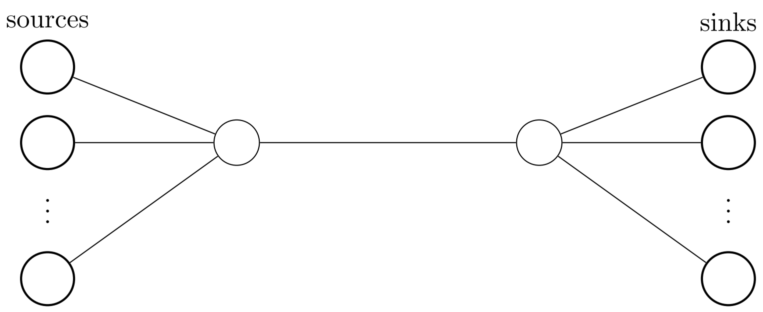
Second error model is burst error model. For every packet, the model generates a random number based on the decision variable, and compares it with the burst error rate to determine if a burst error event should occur. If a new error event occurs, the model will generate a new burst size to determine how many packets should be dropped in this particular burst error event in addition to the current packet.

Instead of the previous model, create a member of *BurstErrorModel*. Set random variable of *BurstErrorModel* similar to the previous model. Set burst rate to desired error rate divided by 2. Create another uniform random variable between 1 and 5. Assign this random variable to burst size. Add this error model to the attributes of point to point link.

1. Vary error rate between 0.0001 and 0.01. Plot throughput vs error rate and compare the results of both error models. 4 or 5 points are enough to obtain the plot.
2. Change error rate to 0.01 and plot *Time Sequence* and average *Throughput* graphs using wireshark and congestion window size obtained from NS3 for two error models and compare the results.
3. (Bonus) Digital computers can not generate true random numbers. Hence they use *pseudo-random number generators* (PRNG) to generate pseudo-random numbers required for simulation. Find out how NS3 generates pseudo-random numbers and explain how its PRNG works.

2.3 Multiple connections

Increase the number of flows to 8. Each flow should start at a time equal to one digit of your student number. You should create a TCP connection from source i to sink i starting at time equal to the ith digit of your student number in seconds. Similar to the first part, bandwidth of each link is 1Mbps (except the middle link). No error model is required for this part.



1. Open pcap file of one of gateways in wireshark. Go to conversations and report the throughput of different connections (please also add an screen-shot of this window to your report). Is the link shared between different connections in a fair way?
2. Open *I/O Graph* window from *Statistics* menu. Add a new graph and set its display filter to *ip.add==st.nu.x.y* where *st.nu.x.y* is the IP address of one of sinks or sources. Add another graph

[3]For example, in this model by setting ErrorRate to 0.1 and ErrorUnit to Packet, in the long run, around 10% of the packets will be lost.

and leave the display filter empty. Set Y axis of both graphs to *Bytes* and interval to 100ms. This graph shows the total throughput vs throughput of one stream.

3. Using the conversations window of wireshark calculate the link utilization of the middle link.
4. Instead of TCP connections, use UDP protocol for all flows. Calculate the link utilization of the middle link and report the throughput of different connections. Compare the obtained results for TCP and UDP.

---

3 What SHOULD I prepare?

You must upload a zip file ("*YourName_StudentNumber*.zip") for this assignment including the source code of all parts and a report in Portable Document Format (.pdf). The document should contain requested results, discussions, any necessary comments, plots and screen-shots of what you have done in each part.