CS 549: Performance Analysis of Computer Networks
Lab Assignment 3

Assigned: April 20, 2024 *Due date*: April 30, 2024

In this assignment, you will learn how to create virtual network containers on a Linux OS. This is the same technology used by Docker. For creating the network, you will be using available mechanisms in a standalone Linux kernel, you should not need to download or install anything. You can create arbitrary mesh networks connecting the containers and can set link parameters including capacity, delay and packet loss. To simplify your work, write `shell/Perl/Python` scripts to setup the network namespaces, to run experiments. To monitor the traffic in the virtual network and analyse the data, you will need Wireshark. All the required steps (including the commands and examples) are described in the reference given below:

Ref: Anoushka Banerjee & Shaifu Gupta, CS549: Quick Guide to Virtual Networking, IIT Mandi, Apr 2020.
Available: `https://cloud.iitmandi.ac.in/f/2392f8b1f7824882a543/?dl=1`

Note that the exercises below are ordered such that each of them builds over the previous one. This is done to enable better understanding and ease of doing. You must first go over the reference linked above and then do the exercises in the sequence in which they appear. Use the reference as a guide to execute the exercises. All the required commands are available in the reference.

Your report must include code snippets (or scripts attached separately), observations presented in the form of figures, tables, graphs, charts (as necessary), and a discussion on the inferences you draw from the observations.

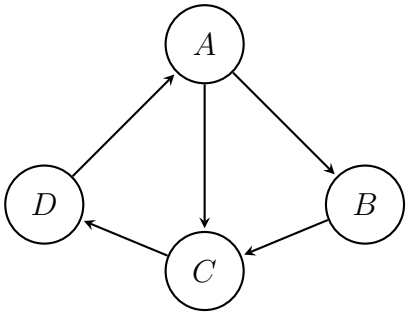
1. Create four network namespaces, say `NetNsA` ... `NetNsD`. In each of these, create one network interface. Experiment with the following:

   (a) Run ping between `NetNsA` and `NetNsB`. Observe the traffic using Wireshark.

   (b) Add a queue discipline with fixed loss of say 20%, and run `ping`. Observe the traffic using Wireshark.

   Report your observations.

2. In this exercise, you will experiment with link scheduling. Consider the toplogy shown below.



Here, each node represents the network namespace created in the above exercise and each edge represents the flow of ICMP packets (generated by `ping` command) between the respective network namespaces. Note that the edges are unidirectional.

Once the network namespaces are established and queue discplines are added, you would have created the required virtual network. You may then launch terminals parallelly and execute the `ping` command to enable flow of packets as per the edges shown in the graph. Under this setup, experiment with the following:

(a) Deactivate a particular edge by setting 100 % loss on the corresponding queue discipline, for a fixed duration of time. Revive the edge by removing the loss constraint on the queue discipline after the fixed duration. Observe the traffic using Wireshark.

(b) Suppose that the entire experiment is run for T time units. Divide T into N time slots. We will now use link scheduling to decide which edge(s) is to be activated in each time slot. Consider the following cases for link scheduling

   i. Probabilistic scheduling: Suppose that each edge is equally likely to be activated, activate exactly 1 edge in each time slot.
   ii. Deterministic scheduling: Divide the edges into maximally independent sets, and activate one set of edges in each time slot.
   iii. [Optional] Probabilistic scheduling of sets: Suppose that each of the above sets is equally likely to be activated, activate exactly 1 set of edges in each time slot.

   Write `Shell/Perl/Python` scripts to execute the two (or three) cases. Observe the traffic for each case using Wireshark. Compare the two (or three) cases in terms of throughput achieved.