CS3205: Introduction to Computer Networks

Assignment 3, Submission Deadline: 5/05/24, 23:59PM

Full Marks: 100 **Instructor: Ayon Chakraborty**

Activity 1: Emulate controlled network environments with Linux `netem`.

Through this activity you will learn how to setup a controlled network experiment with various network performance metrics as the key tuning knobs for such control. Go through this `manpage`, https://man7.org/linux/man-pages/man8/tc-netem.8.html in particular, the usage of delay, loss and rate parameters. You can also experiment with other parameters like packet duplication, reordering etc. Look at the `manpage` examples very carefully and run them on your system. For our experiments, we need the following controls.

- Control the delay per packet with some optional jitter as well as the ability to draw the latency values from a distribution, e.g., Normal or Uniform.
- Control the packet loss rate, typically stated as a percentage.
- Control the maximum bandwidth (rate limit) – *ofcourse the emulated maximum bandwidth must be lesser than the maximum bandwidth available in the actual link.*

Tinker with a set of values of bandwidth (B), packet loss rate (L) and delay (D) for the two applications that you have created in the previous assignment – (i) Webpage loading and (ii) Audio streaming. *Note that the `netem` can run either on the server or the client, the config must be done accordingly (traffic affected uplink or downlink)*

a. If we imagine B, L and D as three axes in the 3D space – for what zones in the 3D space do you see a noticeable change in your audio streaming experience. What is it mostly dependent on: B, L or D?

b. Which parameter severely affects the webpage load time? For instance, can we define a hyperplane that separates out a "good experience zone" from a "bad experience zone"? Remember to use the incognito browsing mode and close the window after each load.

(*questions "a" and "b" are **optional** and you may skip them. Only make sure you are able to configure the emulated network which can be verified partially using a ping or an internet download.*)

Task 2. Implementation and Performance Evaluation of Stop-and-Wait, Go-Back-N and Selective-Repeat Protocols

In this assignment your task is to have a working implementation and performance evaluation of the three reliability protocols – Stop-and-Wait (SW), Go-Back-N (GBN) and Selective-Repeat (SR) built on top of an unreliable UDP transport. The implementation can be Python based, but if you are willing to have a C++ based

implementation, you are most welcome. [*It is better to write your own code for such exercises, rather than debugging readymade public code. Before trying out ChatGPT etc, go through publicly available implementations, https://github.com/haseeb-saeed/go-back-N, https://github.com/chetanborse007/SelectiveRepeat etc*].

The underlying task that we consider in this assignment is to transfer a file (loco.jpg) from system A to system B via UDP with additional reliability protocols (SW, GBN, SR) built on top of it.

System A. In the UDP server code, the following should be considered.

- Create a packet generator class that reads the entire file and generates packets of size K Kilobytes each, the filename and value of K being passed as an argument.
- The sender code should be able to choose one of the algorithms: SW, GBN and SR which is passed as an argument along with the window size and retransmission timeout (RTO). In case of SW, the window size can be ignored.
- The main program should invoke the above classes/functions along with the initial parameters – filename to read, packet size, window size, retransmission time out value.

System B. In the UDP client, calculate the total time elapsed between the time of the first request being sent to the time when the download is completed. Additionally, configure **`netem`** to limit the maximum (downlink) bandwidth and configure a few nominal values of latency and packet loss rate.

Present an analysis on the download time as a function of bandwidth, latency and packet loss. Setup the following network experiment.

Fix bandwidth to be 100KBps and packet size of 25KBs, the window size to be a value between 5 and 10 and the RTO to be 200ms.
Vary packet loss percentages as 0.1, 0.5, 1, 1.5, 2 and 5 (e.g., `tc qdisc change dev eth0 root netem loss 0.1%`) – total 6 values
For each packet loss rate value, vary the latency value as a normally distributed random variable with means 50ms, 100ms, 150ms, 200ms, 250ms and 500ms (e.g., `tc qdisc change dev eth0 root netem delay` *mean*`ms` *dev*`ms distribution normal`) and deviation fixed to 10ms – total 6 values.
(feel free to choose any other combinations of values that you find interesting to demonstrate the tradeoff)

Save the download times as a 2D matrix (6x6) where each cell can be identified with a certain <**loss, latency**> tuple. You should save the matrices for each protocol: SW, GBN and SR. *It is better and easier if you can automate this process (say using a bash or Python script) of changing the* `netem` *parameters and rerun the UDP client.*

Plot three heatmaps, where the download time corresponds to the heat value. Use a common colourbar range for all the three heatmaps for ease of visual comparison. Write a few general insights that you are able to draw from the data. Note that insights do not correspond to qualitative description of quantitative data, rather

some high-level conclusions that you can draw. Include a one-page PDF file for the same.

Submission Instructions

Put all your source code, data (the three matrices – SW, GBN, SR), the PDF file along with a README file in a directory names <roll_1>_<roll_2>, and compress it in the tar.gz format.

Failure in naming the files as instructed will draw additional penalty.

Late Submission Policy

The deadline is at 23:59pm on 5th May 2024.

Late submission by each additional hour will attract a penalty of 1%. For example, if you delay the submission by one complete day, you will face a penalty of 24%.