Project II (Task I): Transmission Control Protocol (TCP)

Assigned: April 3, 2019
Task I due: April 13, 2019 (30% of the grade)

We have covered reliable data transfer and the different aspects of TCP in class. Your task in this project is to implement TCP from the ground up. We are going to structure this projects with two different task mainly: reliable data transfer, and congestion control.

IMPORTANT: You should use the Mininet VM that was provided. Mininet **??** is a wonderful tool that you can use to emulate different network setups and topologies. You can easily create networks of several FTP clients all contacting the same FTP server. You can emulate different link speeds, end-to-end delays, and even packet losses. The beauty of Mininet is that you can script the entire test-case using a python script and run it, generate the results and then evaluate what happened.

1 Task I: Simplified TCP sender/receiver (Reliable data transfer)

Your first task is to implement a "Reliable Data Transfer" protocol, following the description of section 3.5.4 (page 268) from the textbook. The idea here is to build a simplified TCP sender and receiver that is capable of handing packet losses.

You will need to implement the following functionalities:

- sending packets to the network based on a fixed sending window size (e.g. WND of 10 packets)
- sending acknowledgments back from the receiver and handling what to do when receiving ACKs at the sender
- a Timeout mechanism to deal with packet loss

1.1 Starter-code

We have provided you with a simple starter-code that consists of the following:

- rdt receiver: this holds the implementation of a simple reliable data transfer protocol (rdt) receiver.
- rdt sender: this holds the implementation of a simple reliable data transfer protocol (rdt) sender.

The simple rdt protocol is implemented on top of the UDP transport protocol. In addition to the rdt protocol, we have also provided you with a Mininet script and a bash script of a simple network topology with two hosts connected through a network cloud. The bash script enables you to define a number of parameters for a specific run such as: the targeted runtime, the emulated network type, the packet loss percentage, etc. We have also added the necessary plotting scripts to see the receiver throughput and the transport protocol sending window size over time.

1.2 Handin

You need to submit your final code through your Github repository. You can setup a private repository on Github and share it with:

local AD Github: jc2977

When you commit your repository, make sure that you name it project2-name1-name2, for example "project2-Jay-Nabil". Make sure that you do regular commits with meaningful comments so that we can track your progress.

Your repository should contain the following files:

1. Makefile: make sure all the variables and paths are set correctly such that your program compiles in the handin directory. The Makefile should build executable named rdt_sender and rdt_receiver.

2. All of your source code: (files ending with .c, .h, etc. only, no .o files and no executables)

3. readme.txt: file containing a brief description of your design of your TCP simple sender and receiver.

4. run.sh and run.py: the bash script(s) and the Mininet script for your test scenarios.

Groups formation and git repository setup
This project is a group project and you must find exactly one partner to work with. Once you have settled on a partner, you need to inform us by setting a git repository and sharing it with me. In you repository name you will have the names of the partners such as "Project2-Jay-Nabil".

Final Submission
The final submission should include a working reliable data transfer protocol.

Late submissions
Late submissions will be handled according to the policy given in the course syllabus.

References

[1] Mininet, An Instant Virtual Network on your Laptop (or other PC), `http://mininet.org/`