Middle East Technical University

Department of Computer Engineering

CENG 435

Data Communications and Networking
Fall 2023–2024
Programming Assignment

Due date: 2024-01-05 23:59

1 Introduction

In this assignment, you are going to implement a file transfer program using TCP and UDP Layer 4 protocols talking across an unreliable channel. The assignment will cover *socket programming*. You will use `python` to implement it, with groups of 2. In this programming assignment, you are going to transfer 10 *large* objects and 10 *small* objects. The large objects will be at least 1000 times larger than the small objects.

1. Develop a socket application using TCP that transfers objects. A "single persistent" TCP connection will be employed. Note that you will transfer 1 large object and 1 small object consecutively in the TCP implementation.

2. Develop another socket application using UDP. Implement a reliable pipelined data transport protocol that overcomes the head-of-line blocking together with the application logic. Divide large objects into small pieces, tag those pieces and convey tagged pieces in an interleaved fashion employing a mixture of pieces from large and small objects. Note that, you have to collate pieces back into the right order intact. Your objective is to perform better than TCP.

For this assignment, you will run experiments that repeat at least 30 times. Plot comparison figures and show the impact of these two approaches on the time to download the 10 large and 10 small objects. Show means and 95% confidence intervals in your figures. Use TC (topology control) `netem` feature (https://man7.org/linux/man-pages/man8/tc.8.html) on Linux to irregularities to the link between the server and the client.

Before starting the implementation, please read Section 3.4 – *Principles of Reliable Data Transfer* from *Kurose & Ross* to cover the theory. For the practice, you can read Beej's Guide to Network Programming (Using Internet Sockets) alongside socket - Low-level networking interface for Python.

You can discuss the homework and ask your questions on the discussion forum thread on our ODTUClass page. I am also available at yigit@ceng.metu.edu.tr, for questions that are not suited to the public forum.

2 Setup

Follow the linked GitHub repository to set up your *test* environment using Docker containers. Make sure to note down the IP addresses of the *server* and the *client* using `ip addr`. Then, ensure that the connection between the containers work using `ping`. Now, you can run `ip neigh` on each container to see which interface is used to access the other container. You can use `tc` rules on those interfaces.

2.1 Manipulating Traffic

We will write `netem` (Network Emulator) rules using `tc` to make the connection between the server and the client *a lot worse*. You can refer to the lab manual from Jed Crandall's course for more information regarding `tc/netem`.

You should run the rules twice inside each container, one for each interface.

```
# on the server, for the server's interface
tc qdisc add dev eth0 root netem delay 100ms 50ms
# on the client, for the client's interface
tc qdisc add dev eth0 root netem delay 100ms 50ms
```

While developing, you might want to start with a good channel between the client and the server, and run the experiments after you can handle the default link between the containers.

The following are useful as well;

```
# see the current rules in effect
tc qdisc show
# deleting a rule using change is cumbersome
# delete alltogether and start from scratch
# mind the interface
tc qdisc del dev eth0 root
```

3 Implementation

You are implementing an object transfer application in `python`. For both the TCP and the UDP implementations, you will use client-server architecture. The object will transfer from server to client.

3.1 Experiments

After completing your TCP and your reliable pipelined data transport protocol on UDP implementations, run the following experiments in isolation. Change only one parameter of the link per experiment and set the other parameters to default no-error values. Make sure that you discard the tc/netem rules after each experiment to reset the rules (`tc qdisc del dev eth0 root`).

- Benchmark, no tc/netem rules applied
- Packet loss: 0%, 5%, 10%, 15%
- Packet duplication: 0%, 5%, 10%
- Packet corruption: 0%, 5%, 10%
- Packet delay: 100ms - uniform distribution, 100ms - normal distribution

3.2 Notes

Please comment your code thoroughly, on *what* your implementation does rather than *how*. Leaving comments to explain your thought process and choices will help me as well as you when you return to work on your homework the following day. I will read your code in its entirety alongside your report.

Your report should include plots and figures that show the impact of file transfer with TCP and UDP, over at least 30 runs per experiment. It should also include 95% confidence interval and mean. You have to write a report that analyzes the figures and draws conclusions.

You cannot run Docker nor tc/netem on ineks. So, while you can develop on any machine you want, you have to test on your personal machines that can run the given Docker containers. Please bear this requirement in mind while you are forming your groups.

Implement this assignment using `python`, using the version that is defined on the given Docker container. Regarding socket programming, you are only allowed to use python socket interface and no other libraries. You are allowed to use libraries unrelated to socket programming, within reason. Employ common sense as needed.

You can check the reliability of your implementation using the MD5 checksums.

4 Submission

This is a group assignment, groups will consist of 2 people. Regarding people outside your group, discussing abstract level ideas regarding your implementation is encouraged. You can build on top of the code examples given in the Linux `man` pages and external resources such as "Beej's Guide" as long as you indicate their origin through code comments. However, using implementation specific code that is not your own is strictly forbidden and constitutes as cheating. This includes but not limited to friends, previous homework, CENG homework repositories on GitHub, large language models or the Internet in general. The violators will get no grade from this assignment and will be punished according to the department regulations.

Your submissions will be tested on the same containers given here.

Use the "Programming Assignment - Report Submission" module on our ODTUClass page to submit your report. Archive everything related to your implementation as a `.tar.gz` file and submit it to the "Programming Assignment - Code". Only one member should submit, but write group members' names clearly on your report and drop an `authors.txt` file into your `tar.gz` archives.

5 Grading

Your submissions will be graded primarily on your report. Naturally, your implementation of the protocol will constitute a portion of this assignment. A lower portion of the grade is reserved for the code comments.