# CENG519 - TPPhase1 Report

Denizcan Yılmaz
2444172

March 2025

## 1 Implementation

For this project, I used the Go programming language to implement the required functionality. I started by duplicating the existing ‘go-processor‘ directory into a new folder named ‘go-processor-tpphase1‘. This allowed me to work independently on the TPPhase1 modifications without affecting the base project.

I then configured the necessary files for Docker-based execution. Specifically, I updated the ‘Dockerfile‘ and ‘docker-compose.yml‘ files to reflect the new directory structure and ensure that the system could build and run the containerized version of ‘go-processor-tpphase1‘.

The primary addition in this phase was a new function named ‘addRandomDelay‘, implemented in the ‘main.go‘ file. This function introduces a randomized delay to simulate network latency using an exponential distribution. The implementation is as follows:

```
func addRandomDelay() {
    // setting the mean delay from lambda
    lambda := 1 / (0.05)
    delaySec := rand.ExpFloat64() / lambda

    delayDuration := time.Duration(delaySec * float64(time.Second))
    time.Sleep(delayDuration)
    fmt.Printf("Random delay: %v\n", delayDuration)
}
```

In this function, a lambda value is defined as the reciprocal of the mean delay (in seconds). Using Go’s ‘math/rand‘ package, an exponentially distributed random delay is generated. The delay is then applied using ‘time.Sleep()‘, effectively pausing execution for a randomized period. This simulates varying network conditions that might be encountered in real-world environments.

## 2 Experiment and Data Collection

After successfully building the Docker image and launching the containers using ‘docker-compose up‘, I initiated a series of pings from the ‘insec‘ container to the ‘sec‘ container using the command:

```
ping sec -c 50 > ping_{average_delay}ms.txt
```

This command sends 50 ping requests and logs the output into a text file. I repeated this process for different lambda values by modifying the ‘addRandomDelay‘ function accordingly. For each lambda setting, I collected the corresponding output file and stored it under the directory ‘tpphase1_report/ping_data‘ for further analysis.

# 3 Analysis and Plotting

To analyze the results, I developed a Python script that parses the collected ping output files. The script extracts relevant data such as round-trip times (RTT) and computes statistical metrics like the mean RTT. It then visualizes the results by plotting the average RTT against the mean random delay.

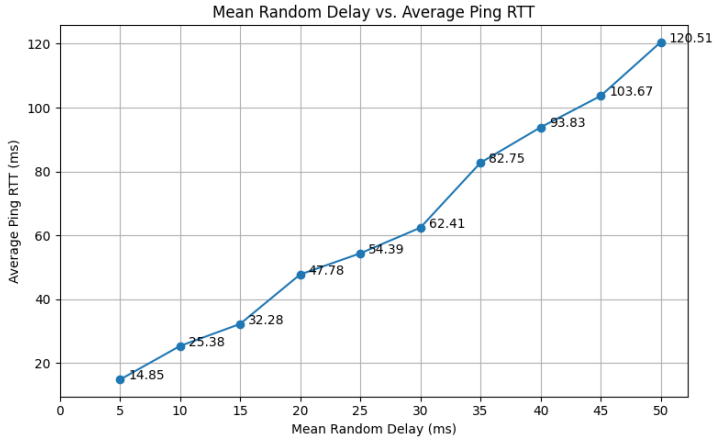


Figure 1: Mean ping RTT vs. random delay.

As shown in Figure 1, the average RTT increases as the mean random delay increases. This outcome confirms the expected relationship between artificially introduced delays and overall network latency.

# 4 Project Repository

The source code and resources related to this project are publicly available on GitHub:

`https://github.com/denizcan-yilmaz/middlebox`

The repository contains the modified Go processor implementation, Docker configuration files, experimental data under the `tpphase1_report_ping_data` directory, and the report and the Python analysis script used for generating plots and evaluating results under `tpphase1_report` directory.