CENG 519 - Network Security

Term Project Phase 1 Report

Yusuf Şahin - 2380889

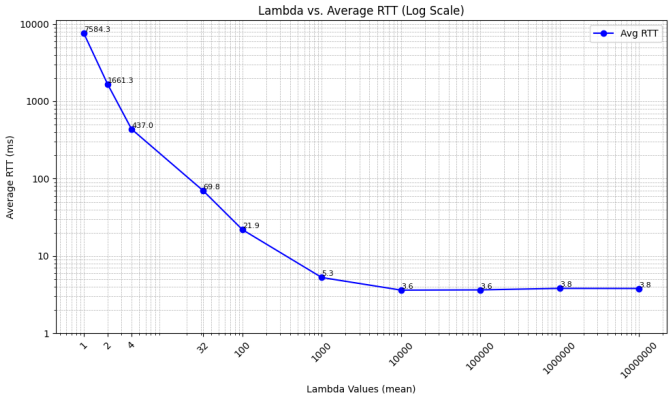

Figure 1. Plot for average RTT for ping packets

1. Report

The graph shows how the average RTT changes as you add exponentially distributed random delays to packets. At first, as the delay increases, the RTT drops significantly, but after a certain point, it levels out and doesn't change much when the lambda gets higher as the expected delay reduces.

I tested the latency by pinging a secure network alongside an insecure one using the command:

```
        ping insec -30 -i 0.1
```

This command sends 30 ping packets with a 0.1-second interval between each, aiming to measure the average rtt between networks.

2. All ping results

The excerpt below presents the output from the ping results, including the case with no added delay, labeled as *no_delay*.

```
lambda  = 1
rtt min/avg/max/mdev = 339.776/7584.346/9815.589/2369.67
    ↪ 1 ms, pipe 10
lambda  = 2
rtt min/avg/max/mdev = 212.761/1661.333/4236.702/1039.26
    ↪ 0 ms, pipe 5
lambda = 4
rtt min/avg/max/mdev = 34.666/437.041/1051.609/286.345
    ↪ ms, pipe 2
lambda = 32
rtt min/avg/max/mdev = 19.693/69.774/129.713/29.152 ms
lambda = 100
rtt min/avg/max/mdev = 6.598/21.868/60.776/12.507 ms
lambda = 1000
rtt min/avg/max/mdev = 3.541/5.263/9.711/1.543 ms
lambda = 10000
rtt min/avg/max/mdev = 3.499/3.615/3.974/0.112 ms
lambda = 100000
rtt min/avg/max/mdev = 3.510/3.640/3.946/0.123 ms
lambda = 1000000
rtt min/avg/max/mdev = 3.702/3.809/3.955/0.060 ms
lambda = 10000000
rtt min/avg/max/mdev = 3.384/3.798/3.962/0.112 ms
lambda = no_delay
rtt min/avg/max/mdev = 3.517/3.794/4.014/0.112 ms
```

Code 1. ping results

3. Random Delay calculation function

This code generates an exponentially distributed random delay using the inverse transform sampling method. The function get_expo_random(double lambda) generates a random value using the exponential distribution with rate parameter lambda. Before sending a packet, the program sleeps for the generated delay (converted to microseconds using usleep), introducing randomness in packet transmission timing.

```
double get_expo_random(double lambda) {
    double u;
    u = rand() / (RAND_MAX + 1.0);
    return -log(1 - u) / lambda;
}
.
.
.
// before sending the packets sleep for random amount of
    ↪ time
double lambda = 10000000;
double sleep_time = get_expo_random(lambda);
usleep((double)(sleep_time * 1e6));
.
.
.
```

Code 2. sample generator from exponential distribution