University of Dhaka

Department of Computer Science and Engineering

CSE-3111 : Computer Networking Lab

Lab Report 6 : Implementation of TCP Reno and New Reno congestion control algorithms and their performance analysis.

Submitted By:

Name: Md Shamsur Rahman Sami

Roll No : 57

Name: Md Rakib Hossain

Roll No : 55

Submitted On :

March 16, 2024

Submitted To :

Dr. Md. Abdur Razzaque

Contents

1 Introduction

1.1 Objective

The primary objective of this experiment is to implement and evaluate the performance of TCP Reno and New Reno congestion control algorithms in a simulated network environment. Specifically, the experiment aims to:

1. Implement TCP Reno and New Reno algorithms using suitable programming languages or network simulation tools.
2. Analyze the behavior of TCP Reno and New Reno under varying network conditions, including different levels of bandwidth, latency, and packet loss.
3. Compare the performance of TCP Reno and New Reno in terms of key metrics such as throughput, packet loss rate, and round-trip time (RTT).
4. Evaluate the advantages and limitations of TCP Reno and New Reno in different network scenarios.
5. Investigate the impact of network parameters on the performance of TCP Reno and New Reno algorithms.

By achieving these objectives, this experiment seeks to provide insights into the behavior and effectiveness of TCP Reno and New Reno congestion control algorithms, contributing to a better understanding of their practical implications in real-world network environments.

2 Theory

2.1 TCP Congestion Control

Transmission Control Protocol (TCP) is a widely used transport layer protocol in computer networks. One of the key functions of TCP is congestion control, which aims to prevent network congestion by regulating the rate at which data is sent over the network.

TCP congestion control mechanisms operate based on feedback received from the network. When a sender transmits data, it expects to receive acknowledgments (ACKs) from the receiver indicating successful receipt of the data. If the sender receives ACKs promptly, it assumes that the network is

operating normally and gradually increases the transmission rate. However, if the sender detects packet loss or other signs of congestion, it reduces the transmission rate to alleviate congestion and prevent further packet loss.

2.2 TCP Reno

TCP Reno is a classic congestion control algorithm widely implemented in TCP implementations. It operates based on a combination of slow start and congestion avoidance mechanisms.

2.2.1 Slow Start

In the slow start phase, the sender starts with a conservative transmission rate and gradually increases it as acknowledgments are received. Initially, the sender starts by sending one segment of data. Upon receiving an acknowledgment for this segment, it doubles the transmission rate and sends two segments. This process continues until a congestion event occurs.

2.2.2 Congestion Avoidance

Once the sender's congestion window (cwnd) reaches a certain threshold, it switches to the congestion avoidance phase. In this phase, the sender increases the transmission rate more gradually, adding one segment to the congestion window for each round-trip time (RTT). This prevents rapid increases in transmission rate that could lead to congestion.

2.3 TCP New Reno

TCP New Reno is an enhancement of TCP Reno that improves its performance in scenarios involving fast retransmit and fast recovery.

2.3.1 Fast Retransmit

In TCP New Reno, if the sender receives multiple duplicate ACKs for the same data segment, it assumes that the segment has been lost and performs a fast retransmit. This means that the sender retransmits the missing segment without waiting for a timeout.

2.3.2 Fast Recovery

After performing a fast retransmit, TCP New Reno enters the fast recovery phase. During fast recovery, the sender reduces its congestion window to

half of its current value and retransmits the missing segment. It then enters a congestion avoidance phase similar to TCP Reno.

These mechanisms allow TCP New Reno to recover from packet loss more efficiently than TCP Reno, leading to improved performance in networks with high packet loss rates.

3 Methodology

3.1 Experimental Setup

The experiments were conducted using a network simulation environment implemented in Python. The simulation environment consisted of a sender node and a receiver node connected by a simulated network link. The sender node implemented the TCP Reno and TCP New Reno congestion control algorithms, while the receiver node provided feedback to the sender regarding packet loss and acknowledgments.

3.2 Experimental Procedure

The experiments were designed to evaluate the performance of TCP Reno and TCP New Reno under varying network conditions. The following steps were followed:

1. **Initialization:** The sender node and receiver node were initialized, and the network link parameters were configured, including bandwidth, delay, and packet loss rate.

2. **Data Transfer:** A file of predefined size was transferred from the sender to the receiver using TCP Reno and TCP New Reno congestion control algorithms.

3. **Measurement:** Various performance metrics were measured during the data transfer, including throughput, packet loss rate, round-trip time, and congestion window size.

4. **Analysis:** The collected data was analyzed to compare the performance of TCP Reno and TCP New Reno under different network conditions. Statistical analysis was performed to identify significant differences between the two algorithms.

3.3 Experimental Variables

Several variables were considered during the experiments to assess their impact on the performance of TCP Reno and TCP New Reno. These variables included:

- Bandwidth: The available bandwidth of the network link.
- Delay: The propagation delay between the sender and receiver nodes.
- Packet Loss Rate: The rate at which packets were lost in the network.
- File Size: The size of the file transferred during the experiments.

3.4 Data Collection

During each experiment, data was collected on various performance metrics, including:

- Throughput: The rate at which data was successfully transferred over the network link.
- Packet Loss Rate: The percentage of packets lost during transmission.
- Round-Trip Time (RTT): The time taken for a packet to travel from the sender to the receiver and back.
- Congestion Window Size: The size of the congestion window maintained by the sender node.

4 Experimental Result

4.1 Task 1

Implement TCP Congestion Control

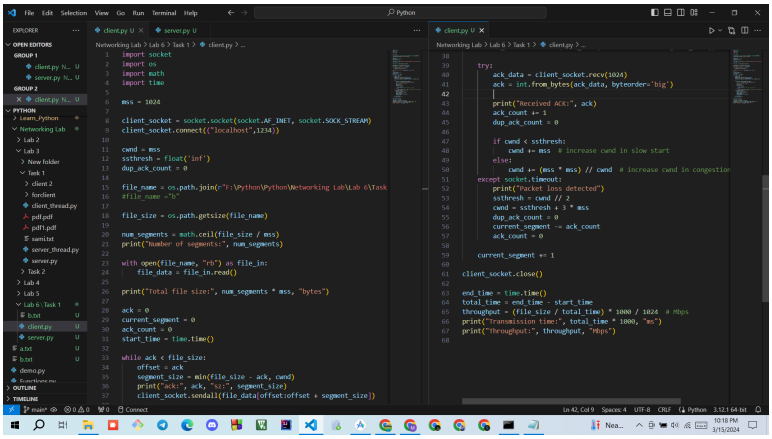

Figure 1: Client side code

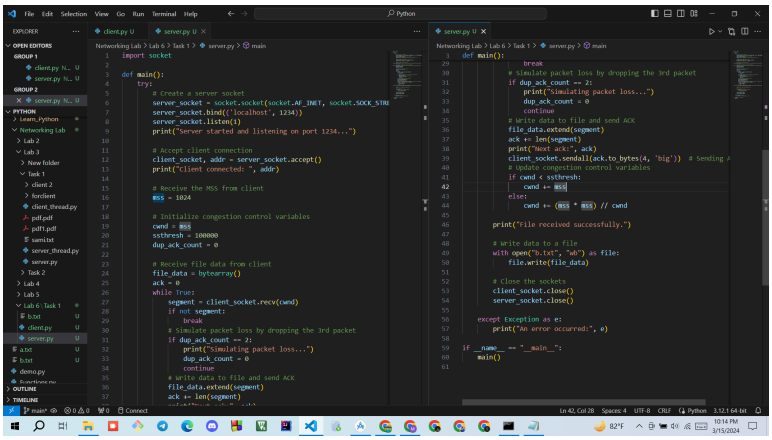

Figure 2: Server Side Code

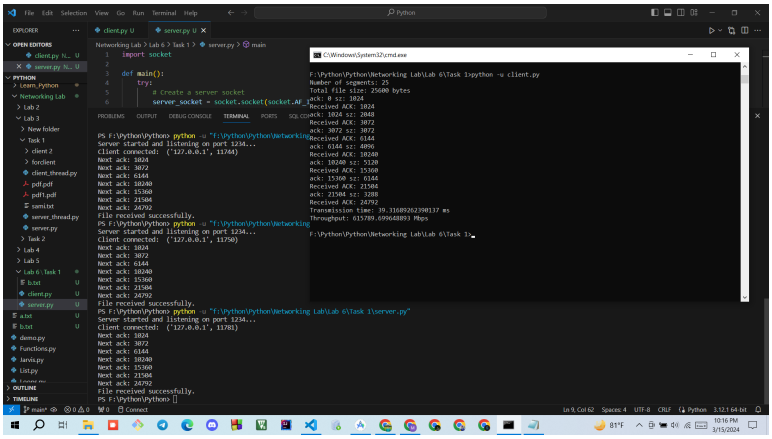

Figure 3: Result

5 Experience

1. Set up TCP communication between clients and a server.
2. Implemented TCP congestion control algorithms.
3. Faced challenges such as packet loss and network optimization.
4. Troubleshot synchronization issues between components.
5. Gained insights into TCP congestion control and its impact on network performance.

References

[1] Flow Control and Congestion Control :`https://www.geeksforgeeks.org/difference-between-flow-control-and-congestion-control/`

[2] TCP Tahoe and TCP Reno :`https://www.geeksforgeeks.org/tcp-tahoe-and-tcp-reno/`

[3] Youtube: `https://youtube.com/watch?v=M6pG9sEVTkg`