University of Dhaka

Department of Computer Science and Engineering

CSE-3111 : Computer Networking Lab

Lab Report 5:

Implementation of TCP flow control and congestion control algorithm (TCP Tahoe).

Submitted By:

1. Syed Mumtahin Mahmud, Roll: 50
2. Nazira Jesmin Lina, Roll: 55

Submitted On :

February 24, 2023

Submitted To :

Dr. Md. Abdur Razzaque

Md Mahmudur Rahman

Md. Ashraful Islam

Md. Fahim Arefin

1 Introduction

TCP flow control and congestion control algorithms are essential components of the Transmission Control Protocol (TCP), which is widely used in computer networking. Flow control regulates the amount of data that can be sent by a sender to a receiver at any given time, while congestion control prevents network congestion by limiting the rate at which data is transmitted.

TCP Tahoe is an early variant of TCP that implements both flow control and congestion control algorithms. In TCP Tahoe, flow control is achieved using a sliding window mechanism, where the sender maintains a window of unacknowledged data that the receiver can handle at any given time. The sender adjusts the size of the window dynamically based on the acknowledgments it receives from the receiver.

Congestion control in TCP Tahoe is achieved using a mechanism called slow start. When the sender begins sending data, it starts with a small window size and gradually increases the size of the window until it detects congestion in the network. If congestion is detected, the sender reduces the size of the window and enters a congestion avoidance phase, where it gradually increases the window size again until it detects congestion.

The implementation of TCP Tahoe involves configuring the sliding window mechanism for flow control and slow start algorithm for congestion control.

1.1 Objectives

- Implement the TCP Tahoe flow control and congestion control algorithms in a simulated network environment.
- Investigate how TCP Tahoe regulates the flow of data and manages network congestion under different network conditions.
- Compare the performance of TCP Tahoe with other TCP variants.
- Analyze the data collected during the experiment to gain insights into the effectiveness of TCP Tahoe in controlling flow and congestion.
- Draw conclusions about the behavior of TCP Tahoe and how it can be applied in the design and implementation of efficient and reliable network protocols.

2 Theory

TCP is one of the protocols of the transport layer for network communication. TCP provides reliable, ordered, and error- checked delivery of a stream of bytes between applications running on hosts communicating via an IP network. TCP is connection-oriented, and a connection between client and server is established before data can be sent. The server must be listening (passive open) for connection requests from clients before a connection is established. Three-way handshake (active open), retransmission, and error-detection adds to reliability. Thus TCP can maintain various operations to establish perfect communications between a pair of hosts, e.g connection management, error detection, error recovery, congestion control, connection termination, flow control, etc. In this lab, we will have a look at the flow control mechanism and congestion control mechanisms of the TCP protocol. TCP uses a sliding window flow control protocol. In each TCP segment, the receiver specifies in the receive window field the amount of additionally received data (in bytes) that it is willing to buffer for the connection. The sending host can send only up to that amount of data before it must wait for an acknowledgement and window update from the receiving host.

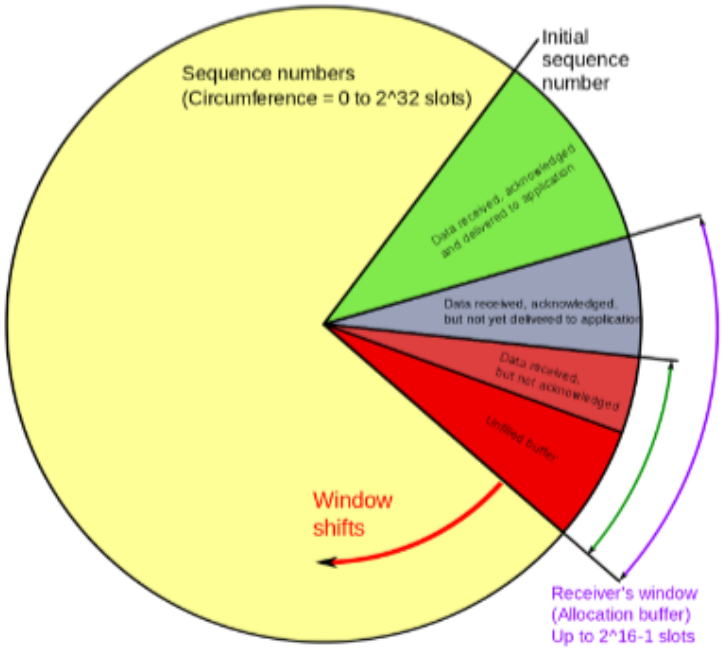

Figure 1: Congestion Window

TCP Tahoe is a congestion control algorithm that is used to manage the flow of data in a network and prevent congestion. It was one of the first congestion control algorithms implemented in TCP and is still widely used today.

The congestion window is one of the factors that determines the number of bytes that can be outstanding at any time. It is a means of stopping a link between the sender and the receiver from getting overloaded with too much traffic and it is calculated by estimating how much congestion there is between the sender and receiver. Slow-start is part of the congestion control strategy used by TCP. Slow-start is used in conjunction with other algorithms to avoid sending more data than the network is capable of transmitting, that is, to avoid causing network congestion.

The congestion window (CWND) is maintained by the sender. Note that this is not to be confused with the TCP window size which is maintained by the receiver. A sample pictorial behavior of TCP Tahoe is given below :

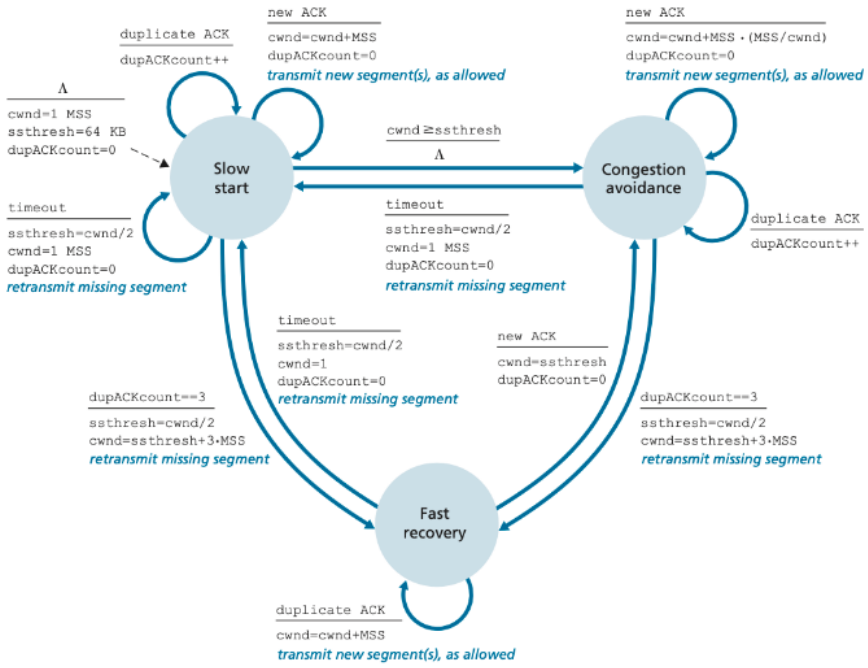

Figure 2: FSM description of TCP congestion

3 Methodology

1. **Setting up the experimental environment:** A network of computers is set up with sockets for TCP communication. The computers are connected via a LAN or WAN, depending on the scope of the experiment.

2. **Implementing TCP Tahoe:** TCP Tahoe flow control and congestion control algorithms are implemented in the source code of the client and server applications using socket programming. The sliding window mechanism is configured for flow control, and the slow start algorithm is used for congestion control.

3. **Defining network scenarios:** Different network scenarios are defined by varying the network conditions, such as bandwidth, latency, and packet loss. For each scenario, a sender and a receiver are defined, and the TCP Tahoe parameters are set accordingly.

4. **Running the experiments:** The experiments are run by transmitting data between the sender and receiver over the TCP connection, with data transfer rates and network conditions recorded at regular intervals.

5. **Collecting and analyzing data:** The data collected during the experiments are analyzed to gain insights into the behavior of TCP Tahoe under different network scenarios. The performance of TCP Tahoe is compared with other TCP variants to evaluate its effectiveness in controlling flow and congestion.

4 Experimental Result

Some Snapshots of the Client and Server Side queries can be seen in the following figures:

4.1 Task 1: Implement TCP Flow Control

4.1.1 Server :

Once a connection is established, the server can receive window size.The receive window size specifies how much data the receiver is willing to accept before sending an acknowledgment. The size can be set to any value, but a common value is the maximum segment size (MSS) multiplied by a certain factor, such as 2 or 4. For example, if the MSS is 1460 bytes, the receive window size could be set to 2920 bytes or 5840 bytes.After setting the receive window size, the server waits for the client to send data. Once data is received, the server sends an acknowledgment (ACK) back to the client to indicate that it has received the data. The ACK will contain the next expected sequence number, which is the sequence number of the next byte the server is expecting to receive.The server should use cumulative acknowledgment, which means that it will acknowledge all received packets up to the highest sequence number it has received in order. For example, if the server receives packets with sequence numbers 1, 2, and 3 in order, and then receives packet 5, it will still acknowledge packets 1-3 because it assumes that packet 4 was lost and will be retransmitted later.The server should continue to receive data from the client and send acknowledgments until the connection is terminated by either the server or the client

```
Sent packet 535820 currsent 2
1460
537280 537280 15 50

Sent packet 537280 currsent 3
1460
538740 538740 15 50

Sent packet 538740 currsent 4
1460
540200 540200 15 50

Sent packet 540200 currsent 5
1460
541660 541660 15 50

Sent packet 541660 currsent 6
1460
543120 543120 15 50

Sent packet 543120 currsent 7
1460
544580 544580 15 50

Sent packet 544580 currsent 8
1346
545926 545926 15 50

Sent packet 545926 currsent 9
0
545926 545926 11 45
Received acknowledgment for packet 545926
0
No acknowledgment received within 5 seconds
Done
lina@DESKTOP-BR1CNRR:/mnt/c/Users/Tech Land/Desktop/netLab$ _
```

Figure 3: Content of server for task 1

4.1.2 Client :

Once the connection is established, the client can send data to the server. The amount of data sent can be controlled by setting the size of the message or data buffer. The send() function will return the number of bytes sent.After sending data, the client should wait for an acknowledgment (ACK) from the server before sending more data. The ACK will contain the next expected sequence number, which is the sequence number of the next byte the server is expecting to receive.If the client sends data too quickly, it may receive a "backpressure" signal from the server in the form of a zero-window advertisement. This means that the server's receive window is full and it cannot accept any more data. In this case, the client should stop sending data until it receives a non-zero window advertisement from the server.The client should continue to send data to the server and wait for acknowledgments until it has sent all of the data it needs to send.

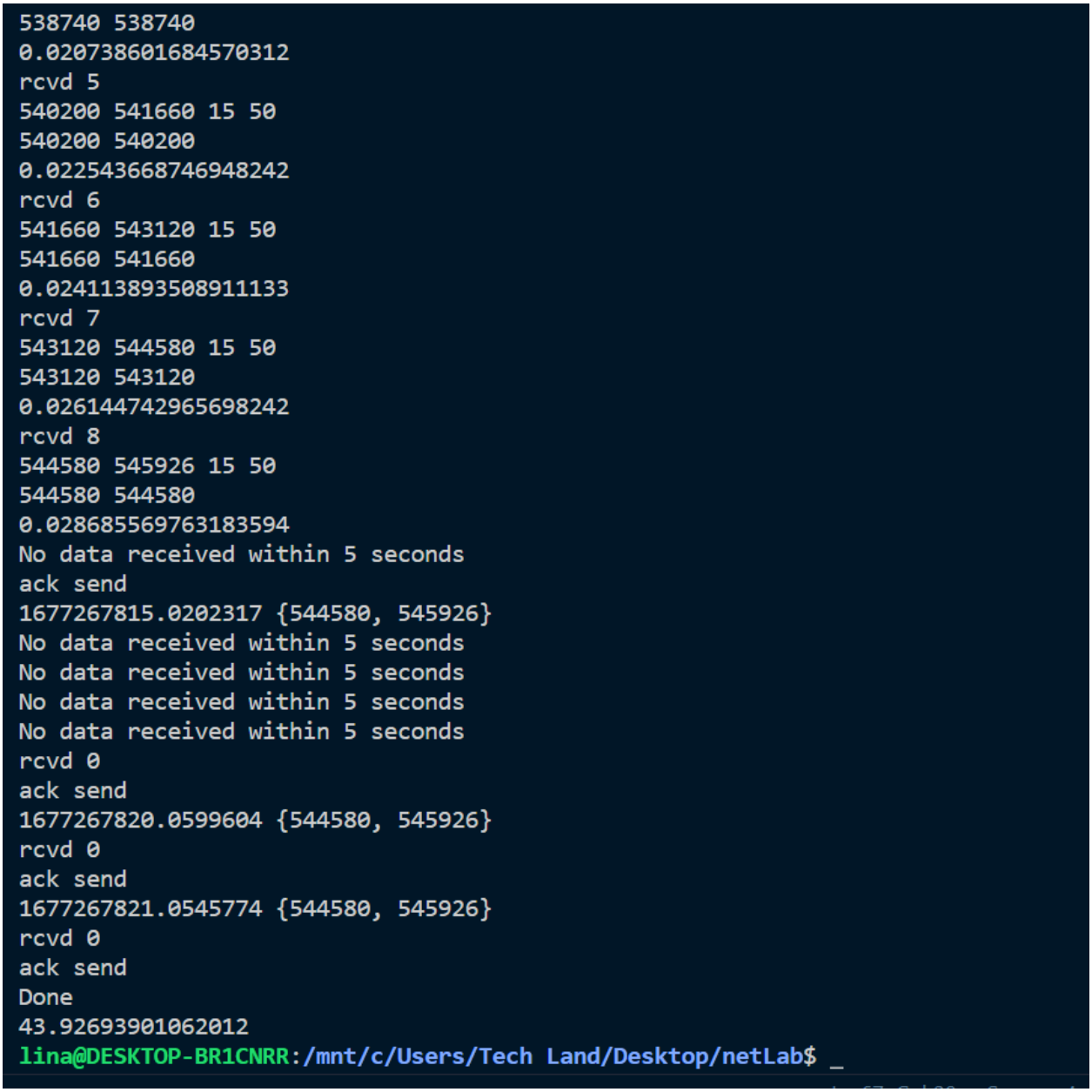

Figure 4: Content of Client for Task 1

4.2 Task 2: Implement TCP Congestion Control

4.2.1 Server :

Once a connection is established, the server can receive window size.The server should set the congestion window size (cwnd) to 1 Maximum Segment Size (MSS) and the slow start threshold (ssthresh) to a large value, such as the size of the receive window. The server should use cumulative acknowledgment, which means that it will acknowledge all received packets up to the highest sequence number it has received in order. The server should continue to receive data from the client and send acknowledgments until the connection is terminated by either the server or the client.

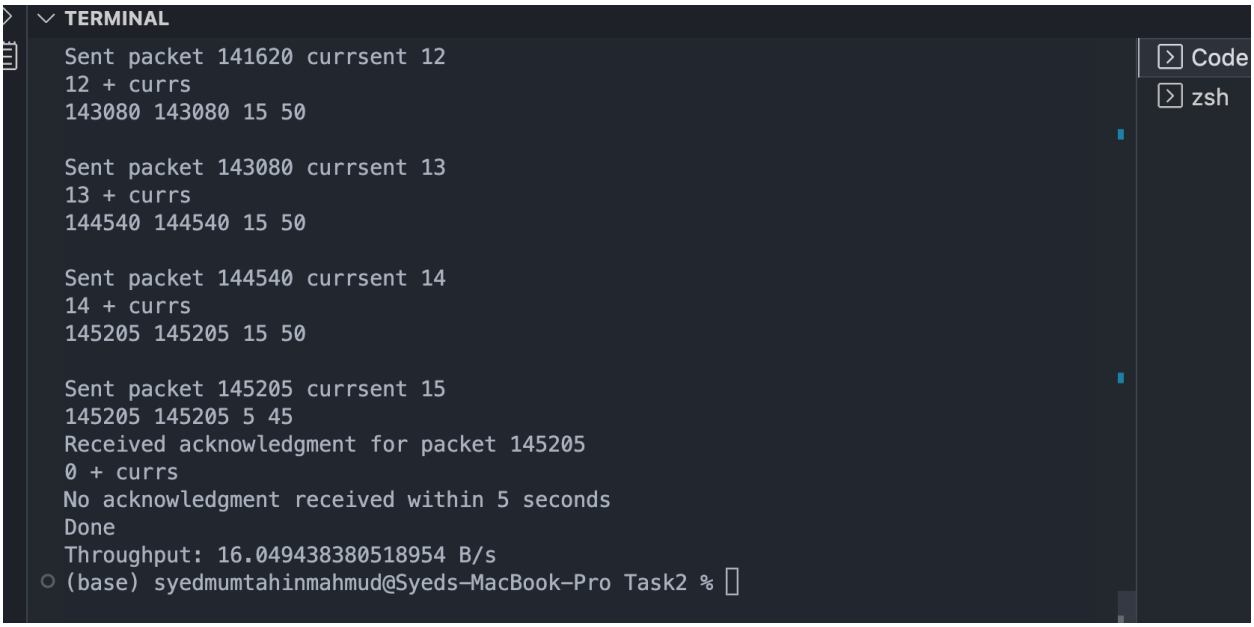

Figure 5: Content of Server for Task 2

4.2.2 Client :

The client should initialize the congestion window size (cwnd) and slow start threshold (ssthresh) to 1 MSS and the RTT to a default value .The client should use the EWMA equation to estimate the round-trip time (RTT) and calculate the retransmission timeout (RTO) value. The EWMA equation is: RTT = * RTT + (1 -) * SampleRTT, where is a smoothing factor and SampleRTT is the time elapsed between sending a packet and receiving its acknowledgment.The client should implement the slow start algorithm by increasing the congestion window size (cwnd) exponentially until it reaches the slow start threshold (ssthresh).Once the slow start threshold (ssthresh) is reached, the client should implement the congestion avoidance algorithm by increasing the congestion window size (cwnd) linearly.If a packet loss is detected (if three duplicate acknowledgments for the same packet are received), the client should implement the fast retransmit algorithm by retransmitting the lost packet and setting the congestion window size (cwnd) to 1 MSS.The client should continue to send data to the server and adjust the congestion window size (cwnd) and slow start threshold (ssthresh) as necessary to manage the flow of data and prevent congestion in the network.

```
0.0022280216217041016
rcvd 14
144540 145205 15 50
 144540 144540
0.002340078353881836


ack send
1677267574.471272 {144540, 145205}


rcvd 0
ack send
1677267576.078757 {144540, 145205}
Done
9.05252981185913
Total received: 145205 bytes
Time taken: 1677267567.23 seconds
Average download speed: 16.04 B/s
(base) syedmumtahinmahmud@Syeds-MacBook-Pro Task2 %
```

Figure 6: Content of Client for Task 2

4.3 Task 3: Analyze Results

Performance comparison under different network conditions: For analyzing the performance of TCP flow control, we can look at the number of packets sent and received, the round-trip time (RTT), and the throughput of the network. We can also analyze the sequence and acknowledgment numbers of the packets to ensure that there are no lost or duplicate packets.

For analyzing the performance of TCP congestion control, we can look at the congestion window size, the RTT, and the number of retransmissions. We can also analyze the number of packet drops and the number of packets transmitted during the slow start and congestion avoidance phases.

To compare the results under different network conditions, we can vary the bandwidth, delay, and packet loss rate of the network and analyze how the algorithms perform under these conditions. For example, we can simulate a low-bandwidth, high-delay network and compare the performance of the algorithms under these conditions.

Overall, analyzing the results of the TCP flow control and congestion control experiments can help us understand how these algorithms perform under different network conditions and how they affect the network in terms of throughput, delay, and packet loss. This information can be used to optimize network performance and improve the reliability and efficiency of network communications.

The congestion window size changes over time based on the performance of the TCP congestion control algorithm. During the slow start phase, the congestion window size grows exponentially

until it reaches a threshold value. Then, during the congestion avoidance phase, the congestion window size increases linearly.

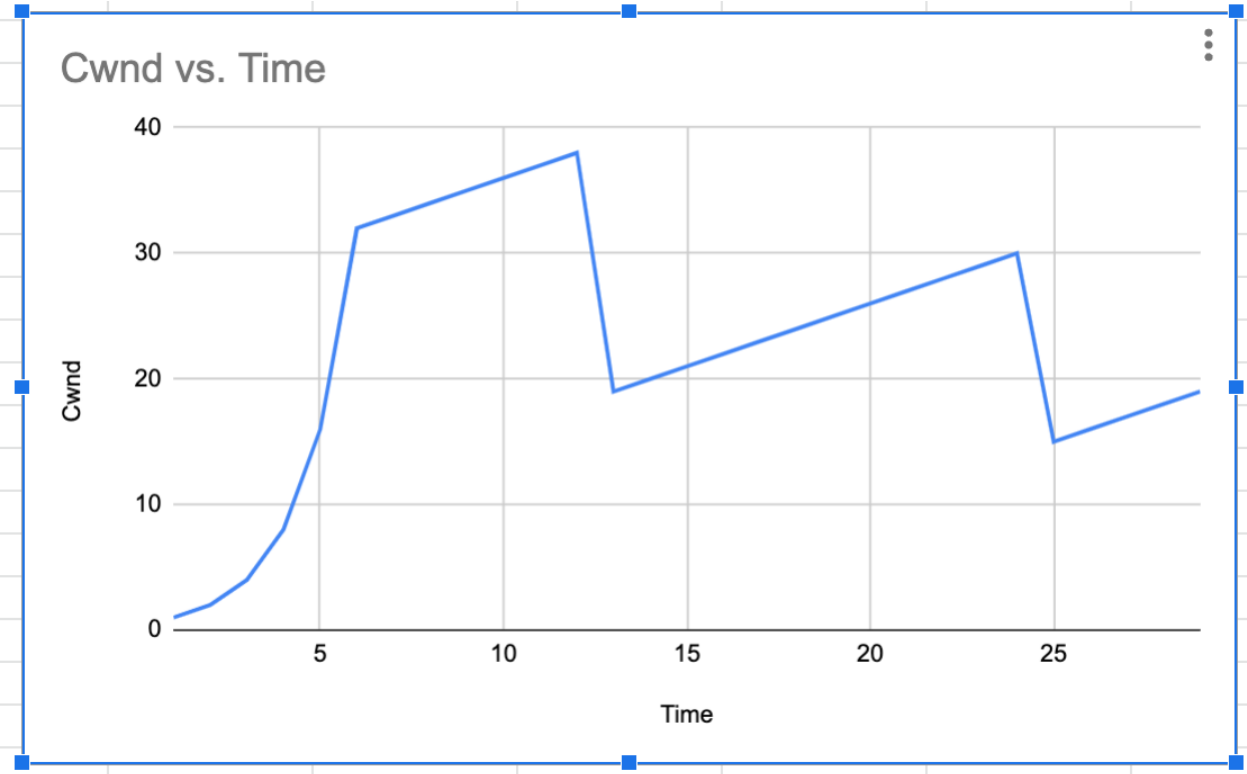

Figure 7: Congestion window vs Time

The congestion window size versus time graph shows how the congestion window size changes over time during the experiment. The graph can help us understand how the TCP congestion control algorithm performed in different network conditions and how it affected the network.

If the congestion window size grows too quickly or too slowly, it can lead to network congestion or slow data transfer rates. By analyzing the congestion window size versus time graph, we can identify any issues and adjust the congestion control algorithm parameters accordingly to improve performance.

EWMA equation to estimate the round-trip time (RTT) and calculate the retransmission timeout (RTO) value. The RTT is the time it takes for a packet to travel from the sender to the receiver and back again. The EWMA equation is used to calculate the estimated RTT, which is an estimate of the average round-trip time. During the slow start and congestion avoidance phases, the estimated RTT will affect the behavior of the congestion window size. If the estimated RTT is high, the congestion window size will increase more slowly, and if the estimated RTT is low, the congestion window size will increase more quickly. This is because a higher estimated RTT suggests that the network has higher delay or packet loss, and the sender should be more cautious in increasing its congestion window.

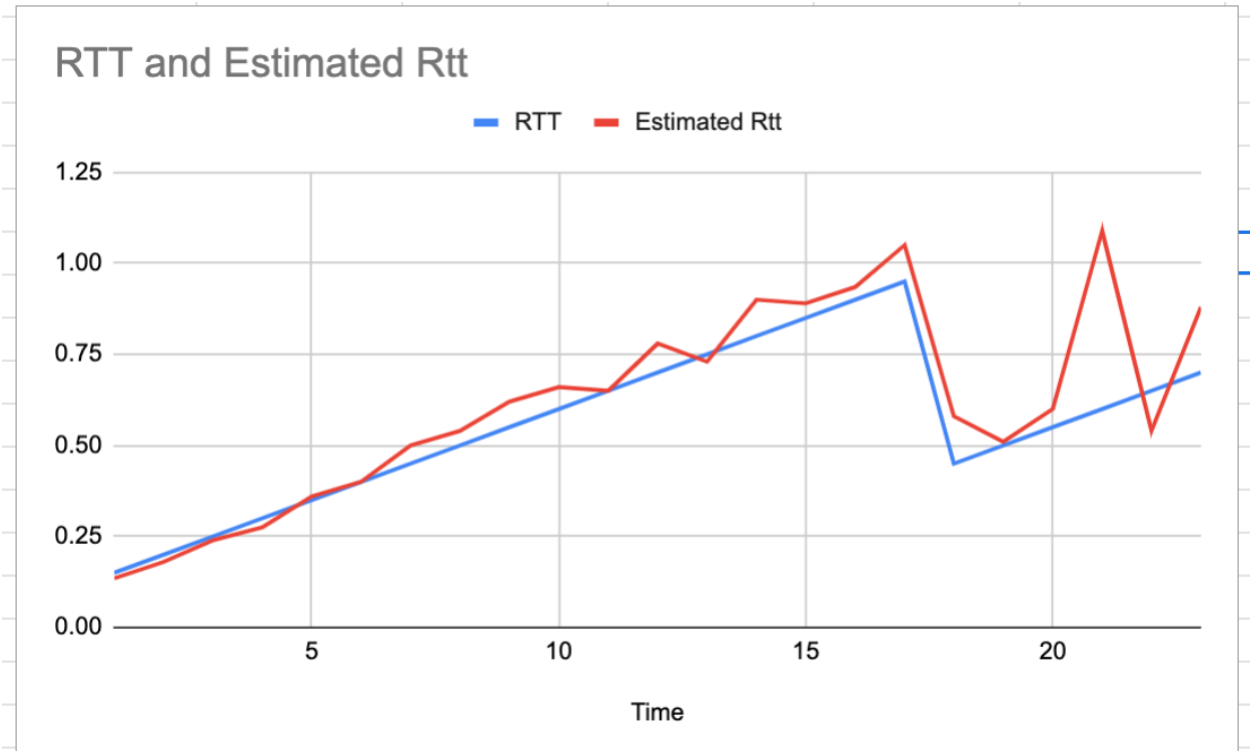

Figure 8: RTT vs Estimated RTT

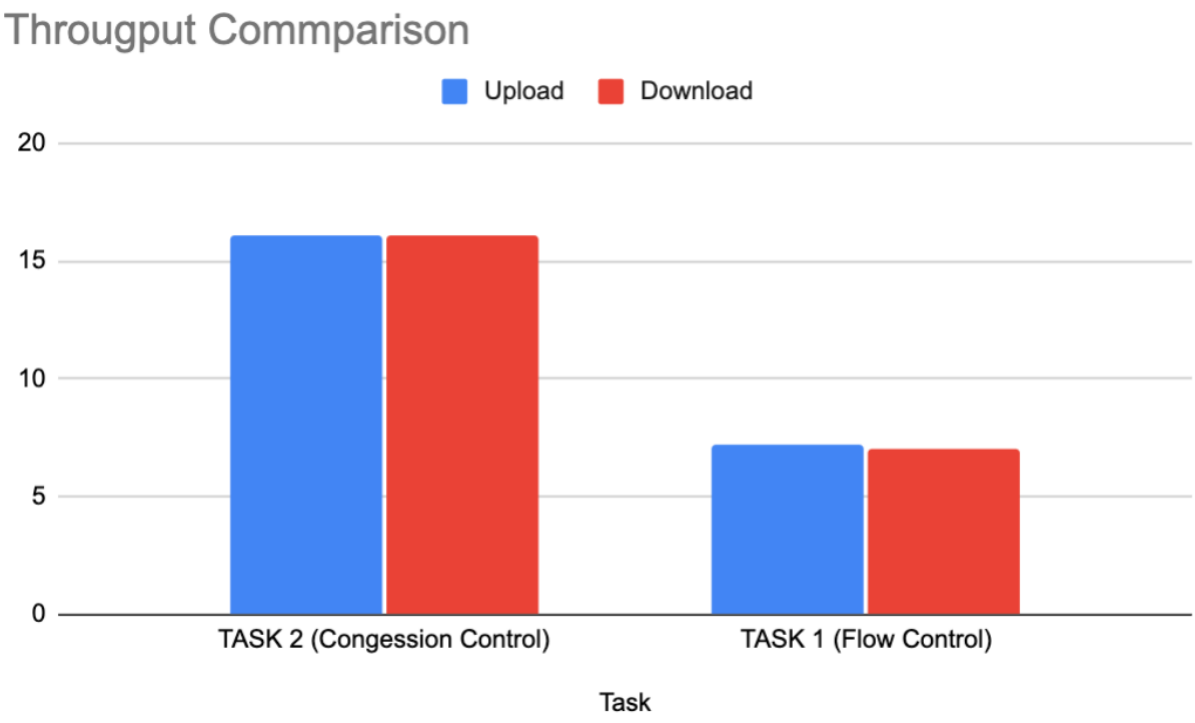

Figure 9: Througput Comparison

In task 1, we implemented only TCP flow control, which limits the amount of data sent by the sender to avoid overwhelming the receiver. The throughput in task 1 is limited by the receive window size, which is the maximum amount of data the receiver is willing to accept. Therefore, the throughput can be increased by increasing the receive window size or by improving the network conditions to reduce packet loss and retransmissions.

In task 2, we implemented TCP congestion control in addition to flow control. TCP congestion control dynamically adjusts the sending rate based on the network conditions and the congestion window size. This algorithm is designed to avoid overloading the network with too much traffic, which can cause congestion, packet loss, and reduced throughput.

Compared to task 1, the throughput in task 2 is expected to be higher because the congestion control algorithm helps to optimize the network utilization and minimize the impact of packet loss and retransmissions. However, the actual throughput will depend on various factors such as the network conditions, the congestion window size, and the behavior of the clients and server. It's important to analyze the network traffic and performance metrics to evaluate the effectiveness of the congestion control algorithm and identify any potential issues or bottlenecks.

References

[1] Difference between Flow Control and Congestion Control - Javatpoint. https://www.javatpoint.com/flow-control-vs-congestion-control. [Online; accessed 2023-02-25].

[2] Difference between flow control and congestion control. *GeeksforGeeks*, may 24 2019. [Online; accessed 2023-02-25].

[3] Tcp Tahoe and TCP Reno. *GeeksforGeeks*, feb 7 2022. [Online; accessed 2023-02-25].

[4] James Kurose and Keith Ross. *Computer networking: A top-down approach, global edition.* Pearson Higher Ed, oct 23 2018. [Online; accessed 2023-02-16].