



# UNIVERSITY OF DHAKA

Department of Computer Science and Engineering

CSE-3111 : Computer Networking Lab

Lab Report 5: **Implementation of flow control and reliable data transfer through management of timeout, fast retransmit, cumulative acknowledgment, loss of data and acknowledgment packets**

**Submitted By:**

Joty Saha (Roll-51)  
Ahona Rahman (Roll-59)

**Submitted To:**

Dr. Md. Abdur Razzaque  
Md. Mahmudur Rahman  
Md. Ashraful Islam  
Md. Fahim Arefin

**Submitted On:** February 14, 2024

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Objectives . . . . .	2
<b>2</b>	<b>Theory</b>	<b>3</b>
2.1	Flow Control . . . . .	3
2.2	Reliable Data Transfer . . . . .	5
2.3	Handling Loss of Data and Acknowledgment Packets . . . . .	5
<b>3</b>	<b>Methodology</b>	<b>6</b>
3.1	Task 1: Implement TCP Flow Control . . . . .	6
3.2	Task 2: Implement Reliable Data Transfer . . . . .	6
3.3	Task 3: Analyze Results . . . . .	7
<b>4</b>	<b>Experimental result</b>	<b>8</b>
4.1	Task 1- Implement TCP Flow Control . . . . .	8
4.1.1	After Running Server Code: . . . . .	8
4.1.2	After Running Client Code: . . . . .	8
4.1.3	Server Code After Running Client: . . . . .	9
4.2	Task 2- Implement reliable data transfer . . . . .	10
4.2.1	After Running Server Code: . . . . .	10
4.2.2	After Running Client Code: . . . . .	10
4.2.3	Packet Loss and Re-transmit in Client Code: . . . . .	11
4.2.4	Server Code After Running Client: . . . . .	12
<b>5</b>	<b>Experience</b>	<b>13</b>

# 1 Introduction

The Transmission Control Protocol (TCP) stands as a cornerstone in modern networking, providing reliable and ordered delivery of data between applications over IP networks. TCP's ability to control the flow of data between a sender and a receiver, coupled with its mechanisms for ensuring reliable data transfer and cumulative acknowledgment, is vital for the robust operation of networked systems. By understanding TCP's flow control and reliable data transfer mechanisms, network engineers can optimize performance, enhance reliability, and mitigate potential issues in data transmission.

## 1.1 Objectives

The objectives of this topic are centered around gaining a comprehensive understanding of how TCP manages data flow and ensures reliable data transfer. Specifically, the objectives include:

1. **Gathering Knowledge about TCP Flow Control:** Explore how TCP regulates the transmission of data between communicating hosts, preventing overwhelming receivers and optimizing network utilization. Understand the sliding window flow control mechanism employed by TCP to manage data flow effectively.
2. **Learning about Reliable Data Transfer in TCP:** Investigate the strategies and mechanisms TCP employs to achieve reliable data transfer, ensuring that data is delivered accurately and in the correct order despite potential network disruptions or packet loss. Delve into TCP's acknowledgment and retransmission mechanisms to understand how it guarantees data integrity.
3. **Understanding Cumulative Acknowledgment:** Examine how TCP utilizes cumulative acknowledgment to streamline the acknowledgment process, reducing overhead and enhancing efficiency. Learn how cumulative acknowledgment enables TCP to acknowledge multiple packets with a single acknowledgment, simplifying the acknowledgment process without compromising reliability.

By addressing these objectives, this topic aims to provide a solid foundation in TCP's flow control and reliable data transfer mechanisms. Through theoretical exploration and practical examples, participants will gain insights into the inner workings of TCP and its pivotal role in ensuring seamless communication across diverse network environments.

## 2 Theory

TCP (Transmission Control Protocol) is a foundational protocol in computer networking that ensures reliable and ordered delivery of data between applications over IP networks. It achieves this reliability through a suite of mechanisms designed to handle various aspects of data transmission, including flow control and reliable data transfer.

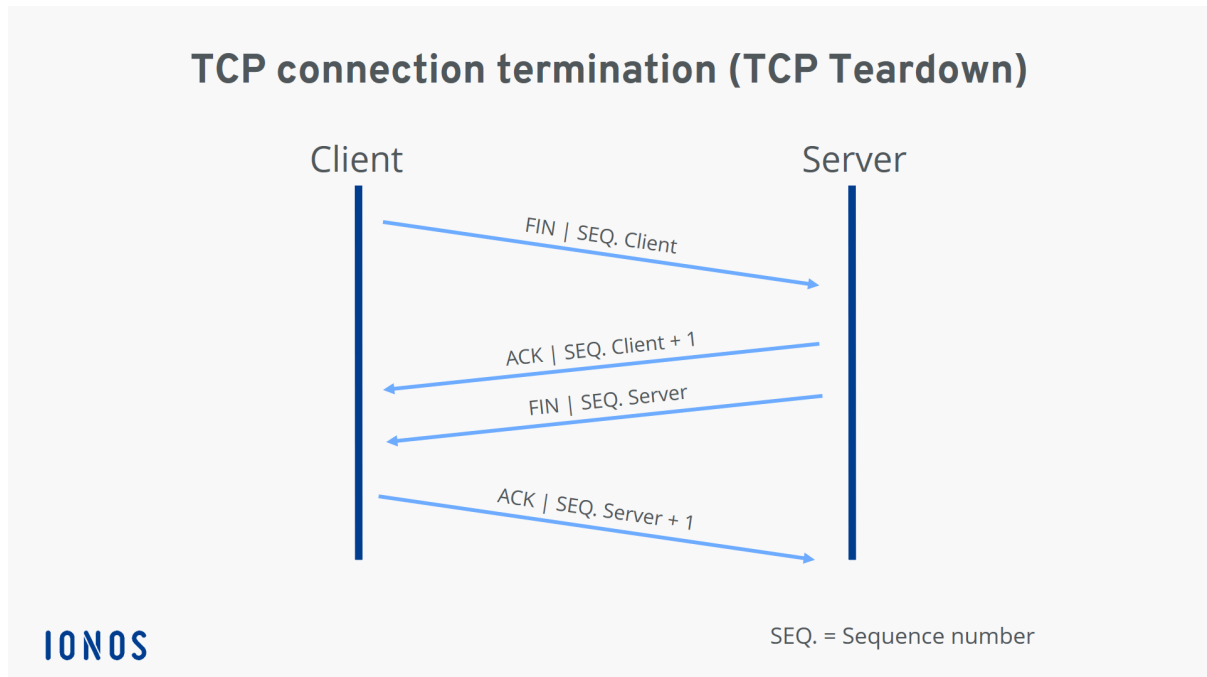


Figure 1: TCP

Let's explore each of these mechanisms:

### 2.1 Flow Control

TCP Flow Control is a protocol designed to manage the data flow between the user and the server. It ensures that there is a specific bandwidth for sending and receiving data so the data can be processed without facing any major issues. In order to achieve this, the TCP protocol uses a mechanism called the sliding window protocol.

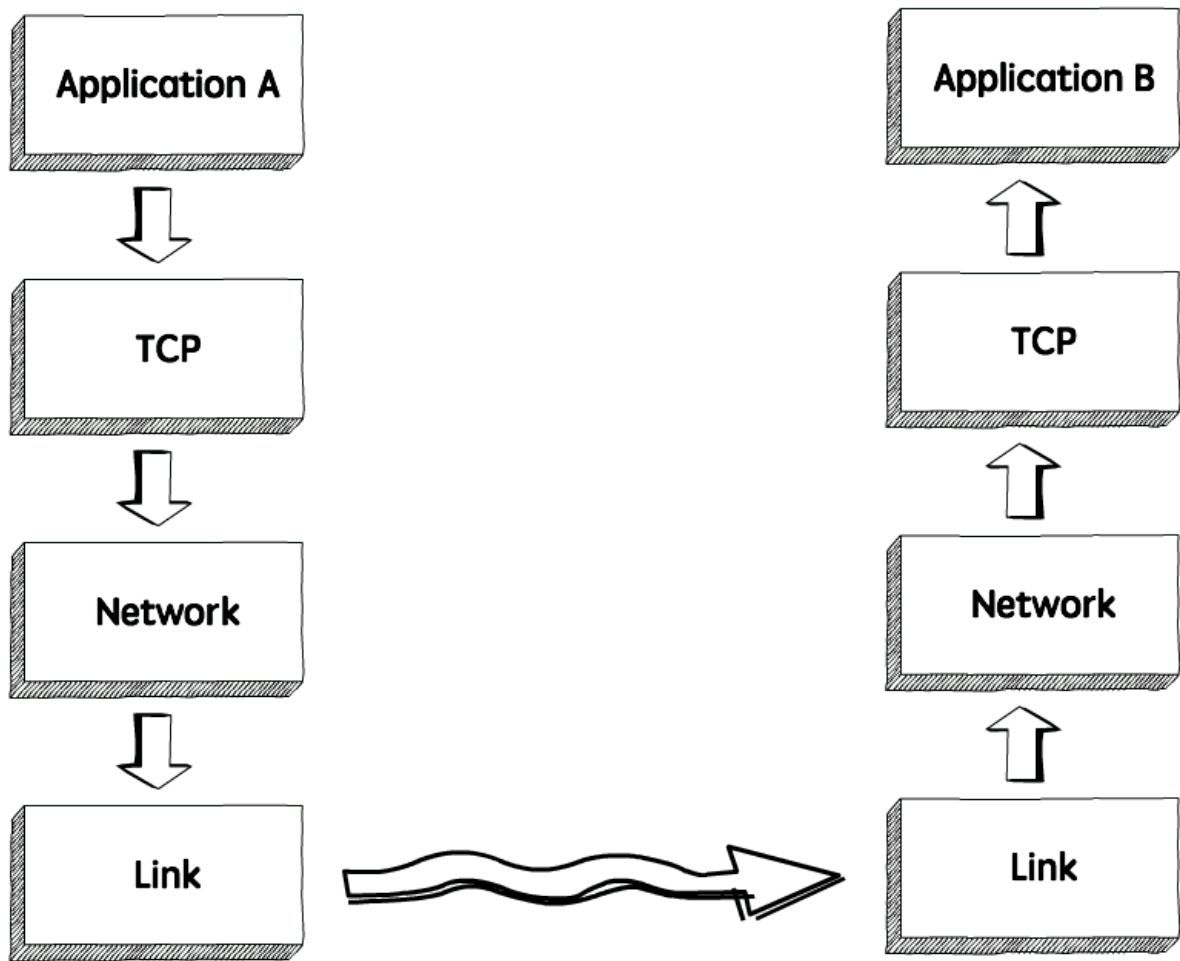


Figure 2: Flow Control

TCP manages the flow of data between communicating hosts using a sliding window flow control mechanism:

- **Sliding Window Mechanism:** The sender maintains a sliding window that represents the range of sequence numbers it is willing to send. The receiver advertises its available buffer space through the TCP window size field, indicating how much data it can receive without overflowing its buffer.
- **Dynamic Adjustment:** The sender adjusts the size of the window dynamically based on acknowledgments received from the receiver and network conditions. This helps prevent overwhelming the receiver with too much data and optimizes network utilization.

## 2.2 Reliable Data Transfer

TCP ensures reliable data transfer by employing several mechanisms:

- **Timeout Management:** TCP uses timeout values to detect lost packets. When a sender transmits a packet, it starts a timer. If an acknowledgment for that packet is not received before the timer expires, the sender assumes the packet is lost and retransmits it.
- **Fast Retransmit:** In addition to timeouts, TCP employs fast retransmit to quickly recover from packet loss. If the sender receives duplicate acknowledgments for the same data segment, it assumes that the next packet was lost and retransmits it without waiting for the timeout to expire.
- **Cumulative Acknowledgment:** TCP uses cumulative acknowledgment to acknowledge receipt of multiple contiguous data segments with a single acknowledgment. This reduces acknowledgment overhead and enhances efficiency, especially in high-latency networks.
- **Selective Retransmission:** TCP can selectively retransmit only those packets that are lost or not received correctly, rather than retransmitting the entire window of data. This minimizes unnecessary retransmissions and conserves network resources.

## 2.3 Handling Loss of Data and Acknowledgment Packets

TCP implements mechanisms to handle loss of data and acknowledgment packets:

- **Retransmission:** When a packet is lost or not acknowledged within the timeout period, TCP retransmits the packet to ensure delivery.
- **Duplicate Detection:** TCP receivers detect duplicate packets and discard them to prevent duplicate processing and ensure data integrity.
- **Negative Acknowledgment (NACK):** Some TCP variants support NACKs, which allow the receiver to explicitly indicate missing or out-of-order packets to the sender, facilitating faster recovery from packet loss.

By implementing these mechanisms, TCP ensures reliable and efficient data transfer across diverse network conditions, making it a cornerstone protocol in modern networking.

## 3 Methodology

### 3.1 Task 1: Implement TCP Flow Control

- **Configure Server for TCP Flow Control:**
  - Set the receive window size on the server to determine how much data it is willing to accept before sending an acknowledgment.
  - Adjust the receive window size based on the available buffer space and network conditions.
- **Configure Clients for Cumulative Acknowledgment:**
  - Modify the clients to use cumulative acknowledgment, where the receiver sends an acknowledgment for the highest sequence number it has received in order.
  - Ensure that the clients assume all packets up to that sequence number have been received.
- **Testing TCP Flow Control:**
  - Send data packets from the clients to the server and monitor the flow control mechanisms.
  - Measure the throughput, latency, and packet loss rate under varying data transfer conditions.

### 3.2 Task 2: Implement Reliable Data Transfer

- **Start Timer and Calculate SampleRTT:**
  - Start a timer after sending each data packet and record the time it takes to receive an acknowledgment (SampleRTT).
  - Calculate the SampleRTT values for each packet transmitted.
- **Configure Clients for EWMA Equation:**
  - Implement the EWMA (Exponential Weighted Moving Average) equation on the clients to estimate the round-trip time (RTT).
  - Calculate the retransmission timeout (RTO) value using the EWMA equation to adjust for variations in network conditions.
- **Implement Cumulative ACK:**
  - Ensure that the clients implement cumulative acknowledgment to acknowledge receipt of multiple contiguous data segments with a single acknowledgment.
- **Implement Fast Retransmit Algorithm:**
  - Implement the fast retransmit algorithm on the clients to detect packet loss based on receiving three duplicate acknowledgments for the same packet.
  - Retransmit the missing packet without waiting for the timeout to expire.
- **Testing Reliable Data Transfer:**
  - Send data packets from the clients to the server and monitor the reliable data transfer mechanisms.
  - Measure the reliability, accuracy, and efficiency of data transfer under varying network conditions.

### 3.3 Task 3: Analyze Results

- **Analyze Captured Network Traffic:**

- Capture and analyze network traffic using tools such as Wireshark to evaluate the performance of TCP flow control and reliability control algorithms.
- Examine packet traces to identify any anomalies, such as packet loss, retransmissions, or delays.

- **Compare Results Under Different Network Conditions:**

- Compare the performance metrics (e.g., throughput, latency) of TCP flow control and reliability control algorithms under different network conditions (e.g., bandwidth, latency, congestion).
- Analyze how variations in network conditions impact the effectiveness of the implemented algorithms and identify areas for improvement.

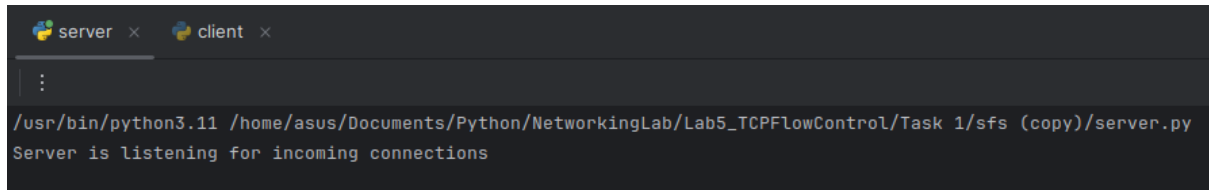
By following this methodology for each task, we can gain insights into the performance of TCP flow control and reliability control mechanisms and their impact on network communication.



## 4 Experimental result

### 4.1 Task 1- Implement TCP Flow Control

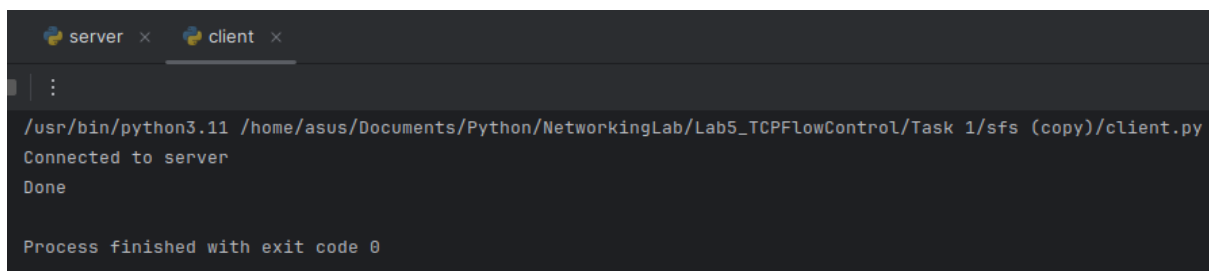
#### 4.1.1 After Running Server Code:

A terminal window with two tabs: 'server' and 'client'. The 'server' tab is active. The terminal shows the command `/usr/bin/python3.11 /home/asus/Documents/Python/NetworkingLab/Lab5_TCPFlowControl/Task 1/sfs (copy)/server.py` and the output `Server is listening for incoming connections`.

```
server x client x
:
/usr/bin/python3.11 /home/asus/Documents/Python/NetworkingLab/Lab5_TCPFlowControl/Task 1/sfs (copy)/server.py
Server is listening for incoming connections
```

Figure 3: Content of Server

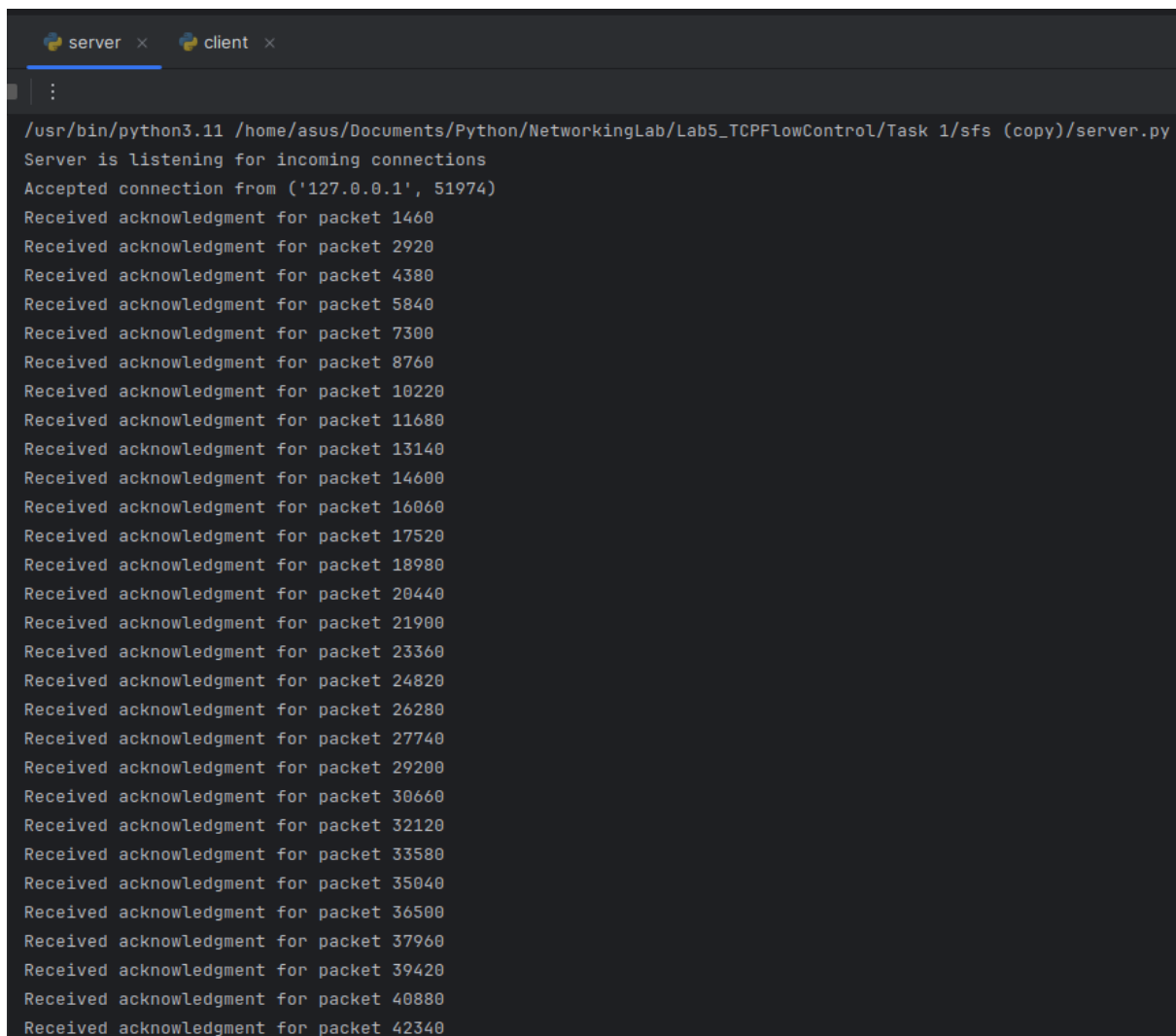
#### 4.1.2 After Running Client Code:

A terminal window with two tabs: 'server' and 'client'. The 'client' tab is active. The terminal shows the command `/usr/bin/python3.11 /home/asus/Documents/Python/NetworkingLab/Lab5_TCPFlowControl/Task 1/sfs (copy)/client.py` and the output `Connected to server`, `Done`, and `Process finished with exit code 0`.

```
server x client x
:
/usr/bin/python3.11 /home/asus/Documents/Python/NetworkingLab/Lab5_TCPFlowControl/Task 1/sfs (copy)/client.py
Connected to server
Done
Process finished with exit code 0
```

Figure 4: Server send msg to Client

#### 4.1.3 Server Code After Running Client:

A screenshot of a terminal window with two tabs: 'server' and 'client'. The 'server' tab is active. The terminal shows the execution of a Python script. The output indicates the server is listening, accepted a connection from '127.0.0.1' on port 51974, and then received a series of acknowledgments for packets ranging from 1460 to 42340 in increments of 240.

```
/usr/bin/python3.11 /home/asus/Documents/Python/NetworkingLab/Lab5_TCPFlowControl/Task 1/sfs (copy)/server.py
Server is listening for incoming connections
Accepted connection from ('127.0.0.1', 51974)
Received acknowledgment for packet 1460
Received acknowledgment for packet 2920
Received acknowledgment for packet 4380
Received acknowledgment for packet 5840
Received acknowledgment for packet 7300
Received acknowledgment for packet 8760
Received acknowledgment for packet 10220
Received acknowledgment for packet 11680
Received acknowledgment for packet 13140
Received acknowledgment for packet 14600
Received acknowledgment for packet 16060
Received acknowledgment for packet 17520
Received acknowledgment for packet 18980
Received acknowledgment for packet 20440
Received acknowledgment for packet 21900
Received acknowledgment for packet 23360
Received acknowledgment for packet 24820
Received acknowledgment for packet 26280
Received acknowledgment for packet 27740
Received acknowledgment for packet 29200
Received acknowledgment for packet 30660
Received acknowledgment for packet 32120
Received acknowledgment for packet 33580
Received acknowledgment for packet 35040
Received acknowledgment for packet 36500
Received acknowledgment for packet 37960
Received acknowledgment for packet 39420
Received acknowledgment for packet 40880
Received acknowledgment for packet 42340
```

Figure 5: Content of Server

## 4.2 Task 2- Implement reliable data transfer

Start a timer after sending data packets and calculate SampleRTT values for each of the packets.

### 4.2.1 After Running Server Code:

```
server x client x
|
:
/usr/bin/python3.11 /home/asus/Documents/Python/NetworkingLab/Lab5_TCPFlowControl/Task 2/server.py
Server is listening for incoming connections
```

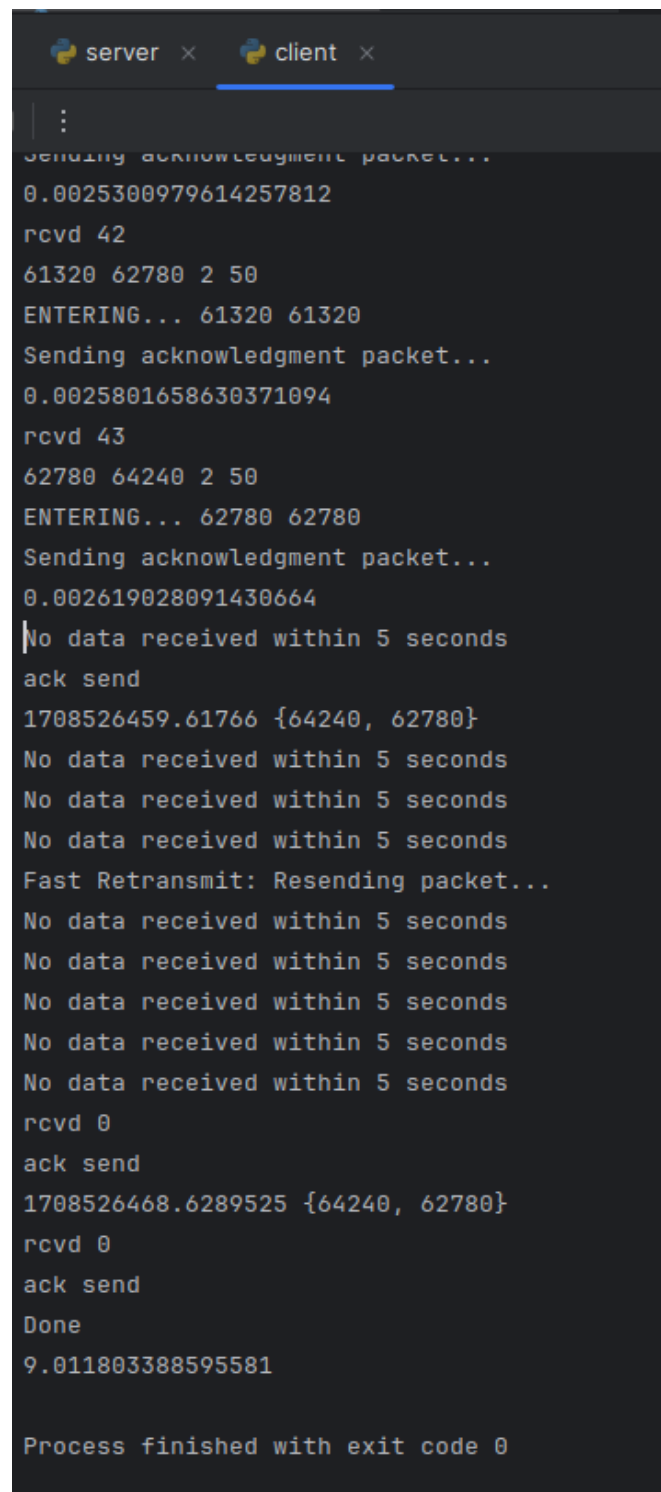
Figure 6: Content of Server

### 4.2.2 After Running Client Code:

```
server x client x
:
/usr/bin/python3.11 /home/asus/Documents/Python/NetworkingLab/Lab5_TCPFlowControl/Task 2/client.py
Connected to server
rcvd 0
0 1460 5 50
ENTERING... 0 0
Sending acknowledgment packet...
0.00035452842712402344
rcvd 1
1460 2920 5 50
ENTERING... 1460 1460
Sending acknowledgment packet...
0.00043201446533203125
rcvd 2
2920 4380 5 50
ENTERING... 2920 2920
Sending acknowledgment packet...
0.0004875659942626953
rcvd 3
```

Figure 7: Server send msg to Client

### 4.2.3 Packet Loss and Re-transmit in Client Code:

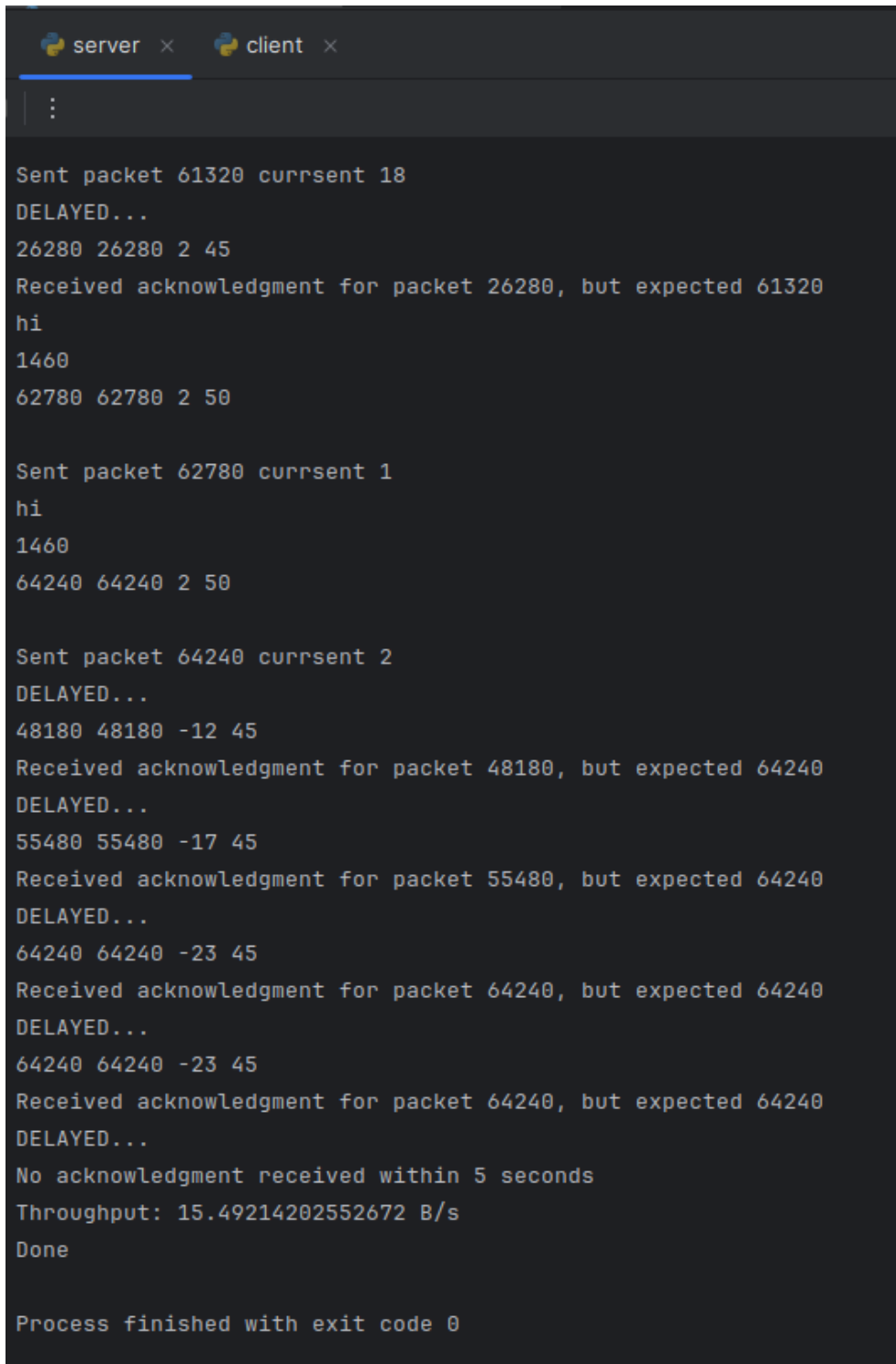


```
server x client x
:
Sending acknowledgment packet...
0.0025300979614257812
rcvd 42
61320 62780 2 50
ENTERING... 61320 61320
Sending acknowledgment packet...
0.0025801658630371094
rcvd 43
62780 64240 2 50
ENTERING... 62780 62780
Sending acknowledgment packet...
0.002619028091430664
No data received within 5 seconds
ack send
1708526459.61766 {64240, 62780}
No data received within 5 seconds
No data received within 5 seconds
No data received within 5 seconds
Fast Retransmit: Resending packet...
No data received within 5 seconds
No data received within 5 seconds
No data received within 5 seconds
No data received within 5 seconds
No data received within 5 seconds
rcvd 0
ack send
1708526468.6289525 {64240, 62780}
rcvd 0
ack send
Done
9.011803388595581

Process finished with exit code 0
```

Figure 8: Packet Loss and Re-transmit in Client

#### 4.2.4 Server Code After Running Client:

A screenshot of a terminal window with two tabs: 'server' and 'client'. The 'server' tab is active. The terminal displays a series of log messages from a server program. It shows the sending of packets (61320, 62780, 64240) and the receipt of acknowledgments. There are several 'DELAYED...' messages and some acknowledgment messages that indicate the received packet number is different from the expected one (e.g., 'Received acknowledgment for packet 26280, but expected 61320'). The log ends with a throughput calculation and a 'Done' message.

```
server x client x
:
Sent packet 61320 currsent 18
DELAYED...
26280 26280 2 45
Received acknowledgment for packet 26280, but expected 61320
hi
1460
62780 62780 2 50

Sent packet 62780 currsent 1
hi
1460
64240 64240 2 50

Sent packet 64240 currsent 2
DELAYED...
48180 48180 -12 45
Received acknowledgment for packet 48180, but expected 64240
DELAYED...
55480 55480 -17 45
Received acknowledgment for packet 55480, but expected 64240
DELAYED...
64240 64240 -23 45
Received acknowledgment for packet 64240, but expected 64240
DELAYED...
64240 64240 -23 45
Received acknowledgment for packet 64240, but expected 64240
DELAYED...
No acknowledgment received within 5 seconds
Throughput: 15.49214202552672 B/s
Done

Process finished with exit code 0
```

Figure 9: Content of Server

## 5 Experience

1. **Experiment Setup:** Configured TCP/IP network environment with multiple virtual machines emulating client-server communication.
2. **Record Addition:** Implemented timeout management, fast retransmit, cumulative acknowledgment, and selective retransmission algorithms within the TCP protocol stack.
3. **Server Initialization:** Initiated TCP server instances to handle incoming connections and manage data transmission.
4. **Verification:** Validated server operation and data transfer reliability through extensive testing using custom-built client applications.
5. **Communication Format:** Defined a structured message exchange format for TCP server-client interaction, including packet structure and data encoding schemes.
6. **Resolution Processes:** Implemented and evaluated iterative and recursive resolution mechanisms within the TCP protocol stack to optimize data transfer efficiency and reliability.
7. **Challenges Faced:** Addressed challenges related to fine-tuning timeout values, detecting and recovering from packet loss, and optimizing flow control parameters for varying network conditions.
8. **Key Learnings:** Deepened understanding of TCP fundamentals, including flow control mechanisms, reliable data transfer techniques, and error recovery strategies.
9. **Practical Skills:** Acquired hands-on experience in managing complex networked systems, troubleshooting communication issues, and optimizing performance in TCP/IP environments. Developed proficiency in distributed system management and TCP protocol stack configuration.

## References

- [1] <https://www.javatpoint.com/flow-control-vs-congestion-control>
- [2] <https://www.geeksforgeeks.org/difference-between-flow-control-and-congestion-control/>
- [3] <https://www.geeksforgeeks.org/principle-of-reliable-data-transfer-protocol/>
- [4] [https://youtu.be/GFD\\_0-SeCxs?si=SK\\_WbSTgoXv5f0zR](https://youtu.be/GFD_0-SeCxs?si=SK_WbSTgoXv5f0zR)
- [5] <https://youtu.be/fHUeG6VF1SY?si=QrMgjD1UJnEpFNLc>