CSE 2213 – Data and Telecommunication Lab Report



Lab Report Name:

Implementation of Multiplexing and Demultiplexing Using Statistical TDM

*Group: B(Even)*

Submitted By

**Md. Mahmudur Rahman Moin (AE-30)**

**Md. Sadman Shihab (SH-36)**

29th Batch

Department of Computer Science & Engineering

University of Dhaka

Submitted On

28 May 2025

Table of Contents

[1. Introduction 3](#_heading=h.ustrejc3oc3l)

[1.1 M](#_heading=h.n1g92nnn65yo)ultiplexing and Demultiplexing [3](#_heading=h.n1g92nnn65yo)

[1.2](#_heading=h.ilwc6vj8zhyi) Types of Multiplexing and Demultiplexing3

[1.3](#_heading=h.90jk9nu6uwgv) Synchronous TDM and Statistical TDM4

[2. O](#_heading=h.3v8croa465r)bjectives5

[3. A](#_heading=h.uev69kqee9ix)lgorithms5

3[.1](#_heading=h.n1g92nnn65yo) Client Side for Multiplexing5

3[.2](#_heading=h.ilwc6vj8zhyi) Server Side for Demultiplexing6

4. Implementation 7

4[.1 C](#_heading=h.n1g92nnn65yo)lient Side - Multiplexing7

4[.2](#_heading=h.ilwc6vj8zhyi) Server Side - Demultiplexing8

5. Result Analysis 9

5[.1](#_heading=h.n1g92nnn65yo) Client Side - Multiplexing9

5[.2](#_heading=h.ilwc6vj8zhyi) Server Side - Demultiplexing 11

6. Discussion14

7. Learning and Difficulties 16

7[.1](#_heading=h.n1g92nnn65yo) Learning16

7[.2](#_heading=h.ilwc6vj8zhyi) Difficulties 16

8. Conclusion 16

# 

# Introduction

In modern data communication systems, efficient use of transmission resources is essential. Multiplexing (MUX) and Demultiplexing (DEMUX) are critical techniques for achieving this. Multiplexing enables multiple data streams to share a single communication channel, while demultiplexing separates the combined stream back into individual data flows at the receiving end.

## 1.1 Multiplexing and Demultiplexing

In data communication, **multiplexing (MUX)** refers to the technique of combining multiple signals or data streams into a single transmission channel. At the receiving end, **demultiplexing (DEMUX)** is used to separate the combined signal back into the original individual data streams. This process is essential for efficiently utilizing bandwidth and reducing the cost and complexity of network infrastructure.

The necessity of MUX and DEMUX arises from the increasing demand for simultaneous data transmission between multiple devices over a limited number of communication lines. Without multiplexing, each device would require a dedicated link, resulting in excessive wiring, higher costs, and inefficient resource utilization. MUX and DEMUX enable **shared use of transmission media**, allowing for scalable and cost-effective communication systems.

## 1.2 Types of Multiplexing and Demultiplexing

1. **Frequency Division Multiplexing (FDM):**

* Multiple signals are transmitted simultaneously over different frequency bands within the same channel.
* Common in radio and television broadcasting.

2. **Time Division Multiplexing (TDM):**

* Signals share the same channel but are transmitted in different time slots.
* Suitable for digital signals.
* Two types: *Synchronous TDM* and *Statistical TDM* (explained below).

3. **Wavelength Division Multiplexing (WDM):**

* A type of FDM used in optical fiber networks, where each signal is assigned a unique wavelength (color) of light.
* Allows for high data transfer rates over fiber optic cables.

4. **Code Division Multiplexing (CDM):**

* Multiple signals are transmitted simultaneously over the same frequency band using unique codes.
* Widely used in mobile communication systems like CDMA.

## 1.3 Synchronous TDM and Statistical TDM

1. **Synchronous Time Division Multiplexing (Sync TDM):**

* In Sync TDM, each device is assigned a fixed time slot in a repeating cycle, regardless of whether the device has data to send.
* While simple to implement, it may lead to inefficient bandwidth utilization if some time slots remain unused.

2**. Statistical Time Division Multiplexing (Stat TDM):**

* In Stat TDM, time slots are dynamically allocated based on demand. Only active data sources are assigned slots.
* This improves channel efficiency and better utilizes bandwidth, especially when data traffic is bursty or irregular.

# 2. Objectives

The main objectives of performing this lab are:

* To gather knowledge on the concept of Multiplexing (MUX) and Demultiplexing (DEMUX) data communication.
* To simulate how multiple data streams can be sent over a shared communication medium.
* To demonstrate how demultiplexing helps identify and deliver data to the correct application.
* To implement Statistical Time Division Multiplexing and Demultiplexing
* To reconstruct individual streams accurately at the receiver end.

# 3. Algorithms

## 3.1 Client Side for Multiplexing

1. Open ***N*** input files.

2. Initialize an empty frame.

3. Loop until all files reach EOF:

* For each file:
  + If data is available:
    - Read 1 byte.
    - Create a data unit: [Stream ID, Data]
    - Append to the current frame.
* Send the frame over the socket to the server.

## 3.2 Server-side for Demultiplexing

1. Create ***N*** output files.

2. Accept connection from client.

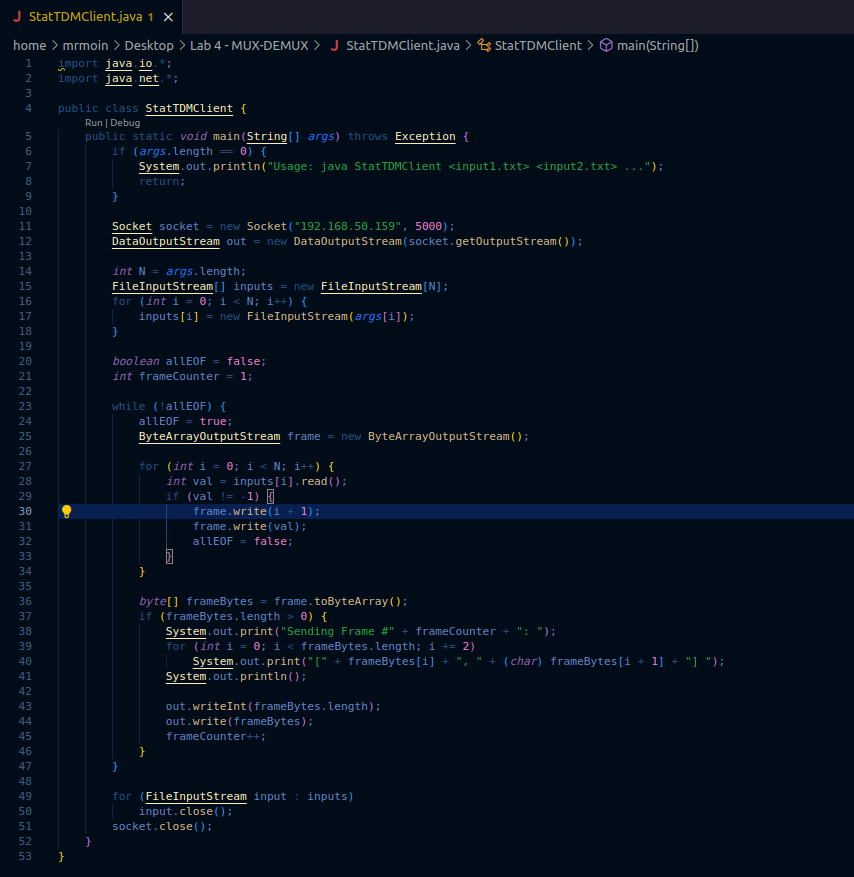
3. While data is received:

* Extract ***N*** bytes
* For each data unit in the frame:
  + Parse Stream ID and data pairs.
  + Write valid data to the corresponding output file using stream IDs.

4. Close all files and socket.

# 4. Implementation

## 4.1 Client Side – Multiplexing

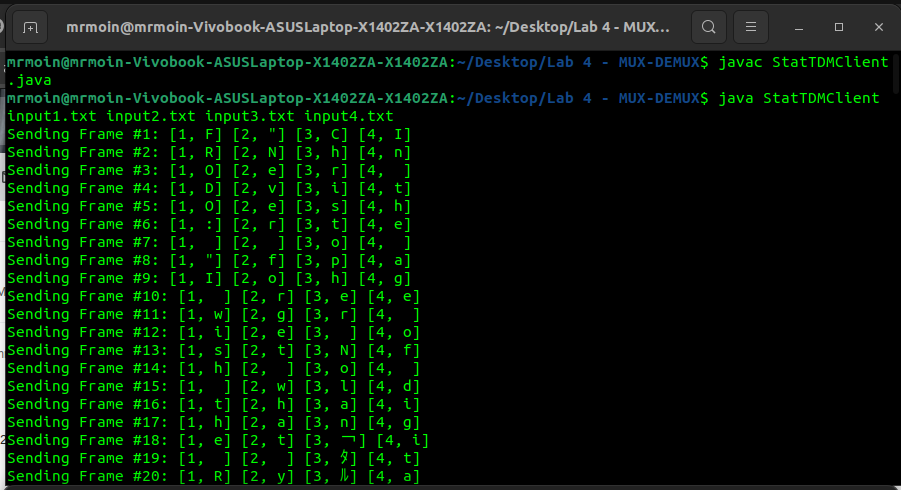


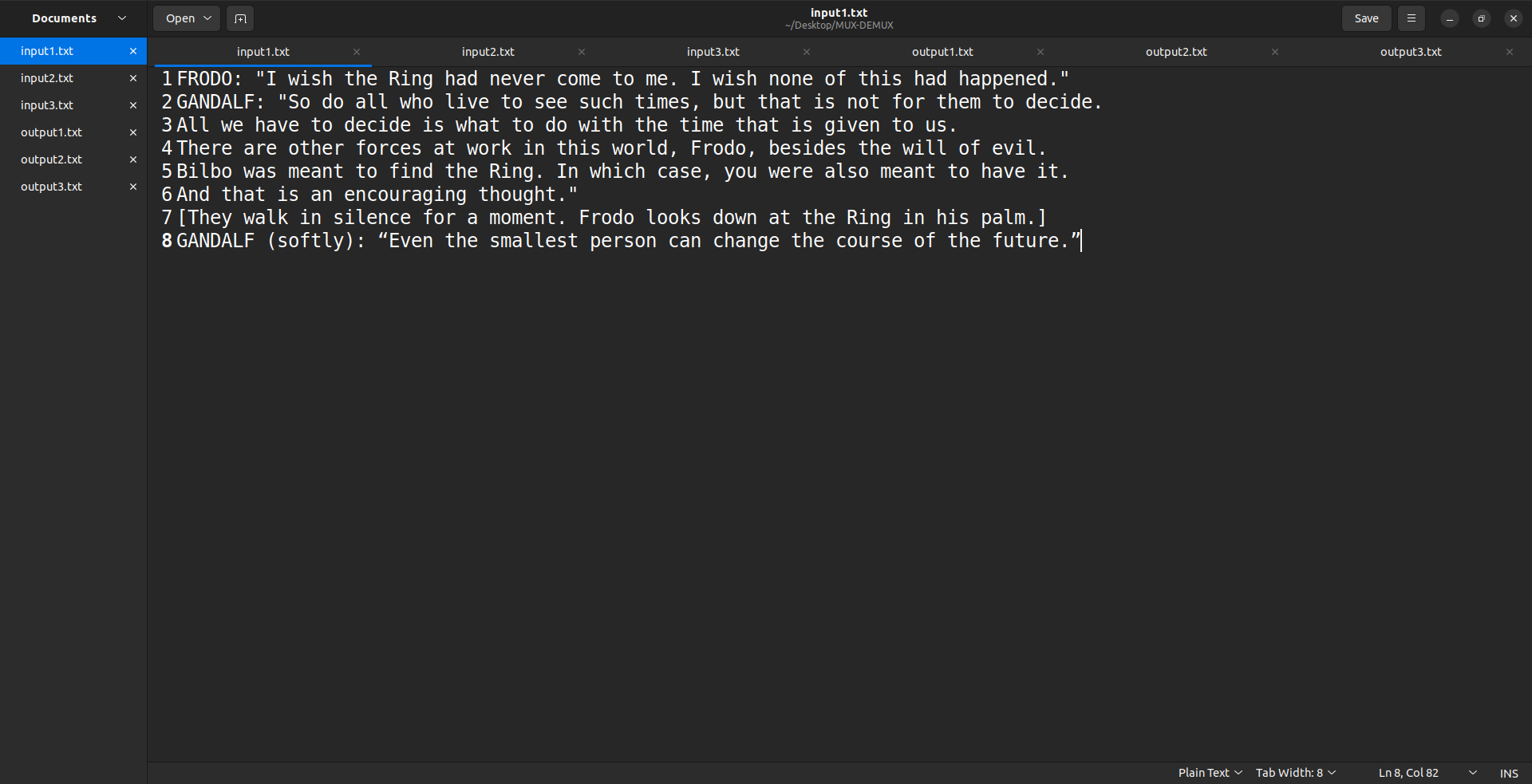
## 4.2 Server side - Demultiplexing

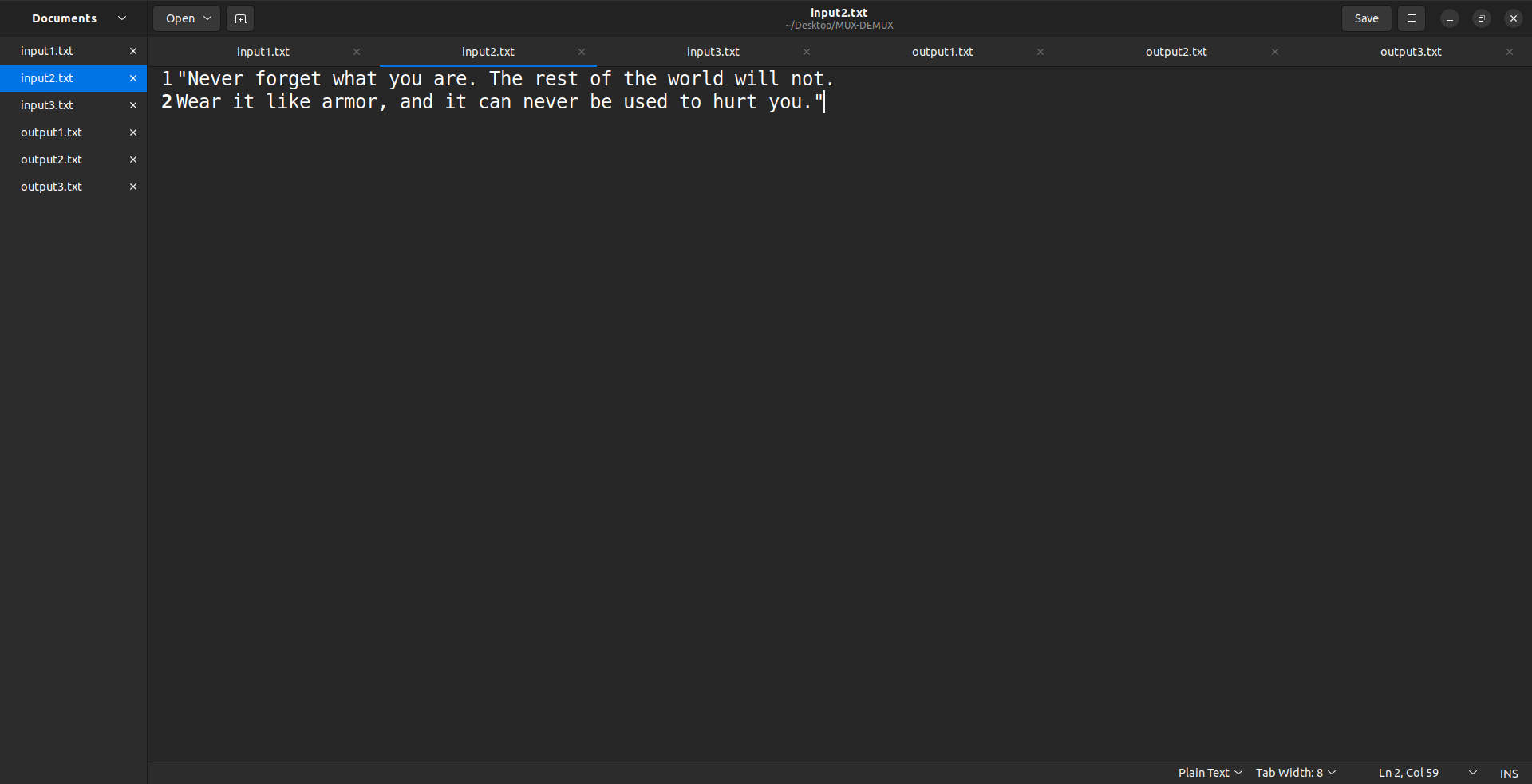
### 

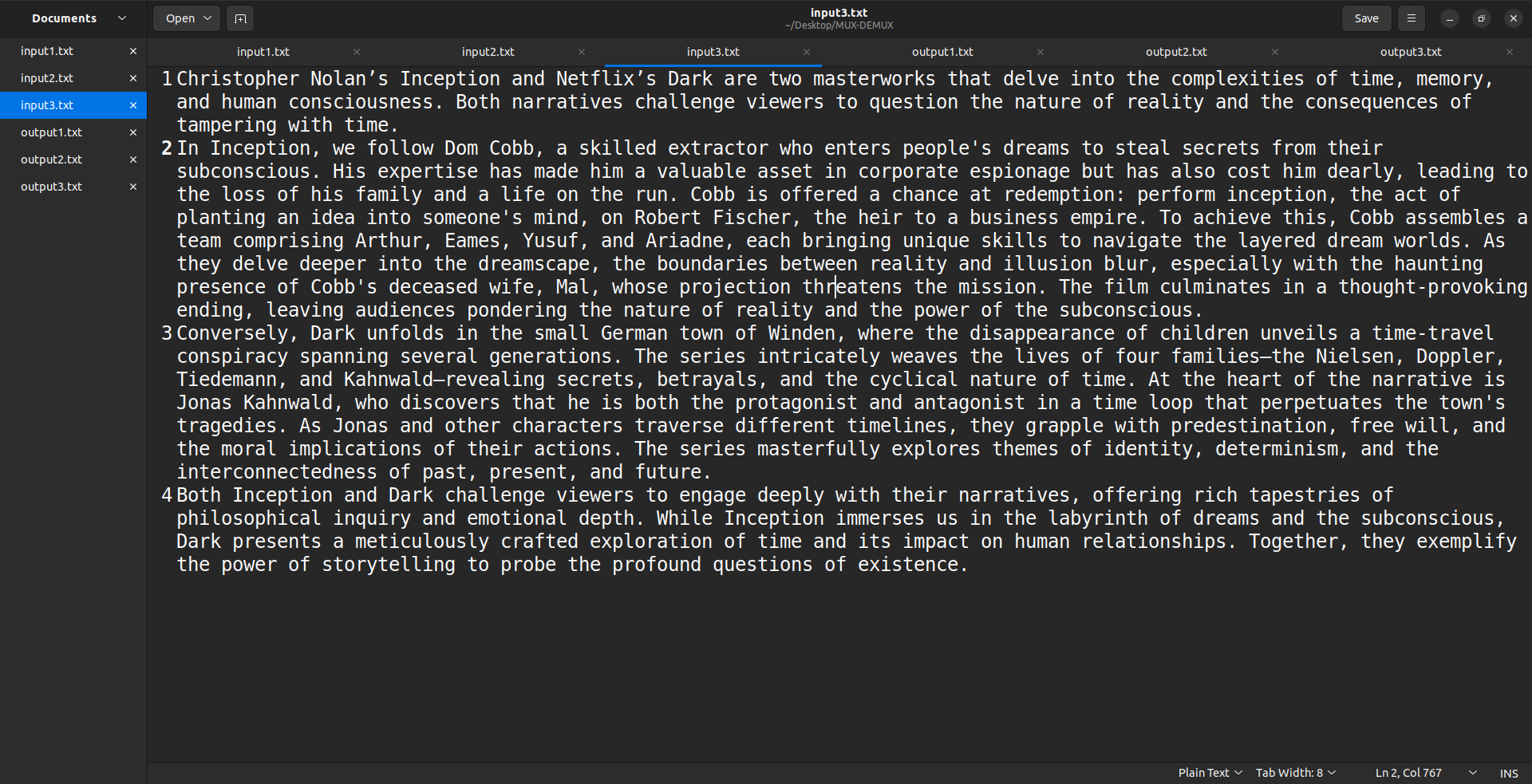
# 5. Result Analysis

## 5.1 Client side - Multiplexing



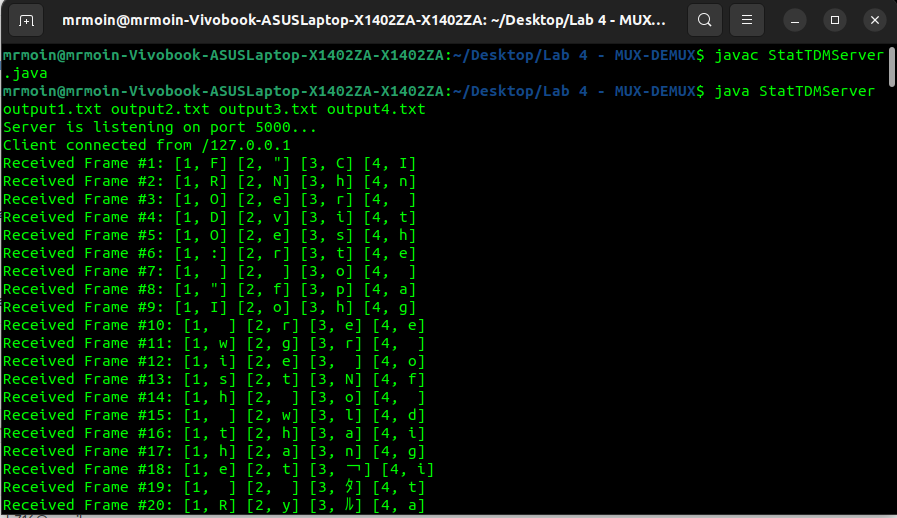


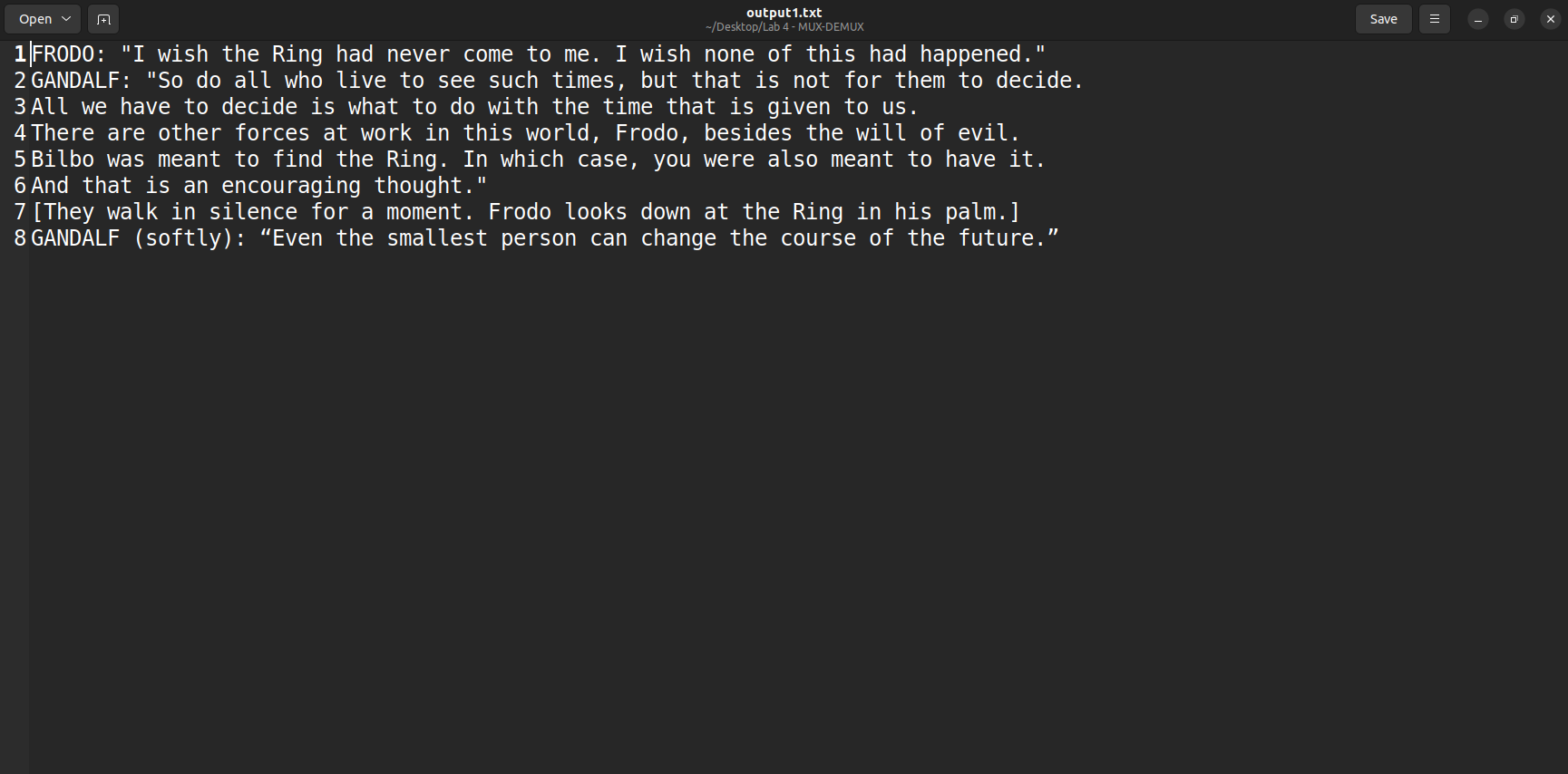


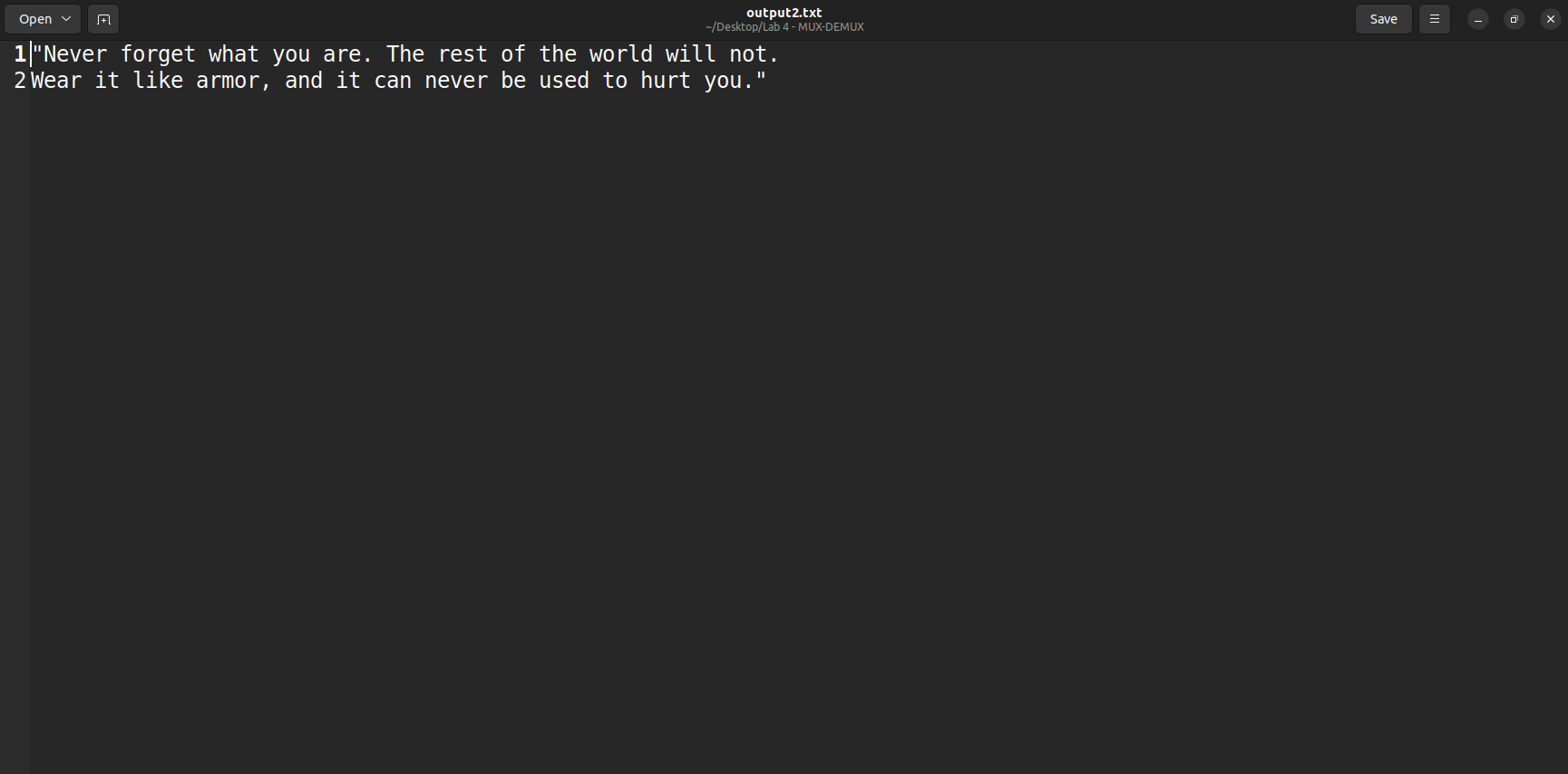


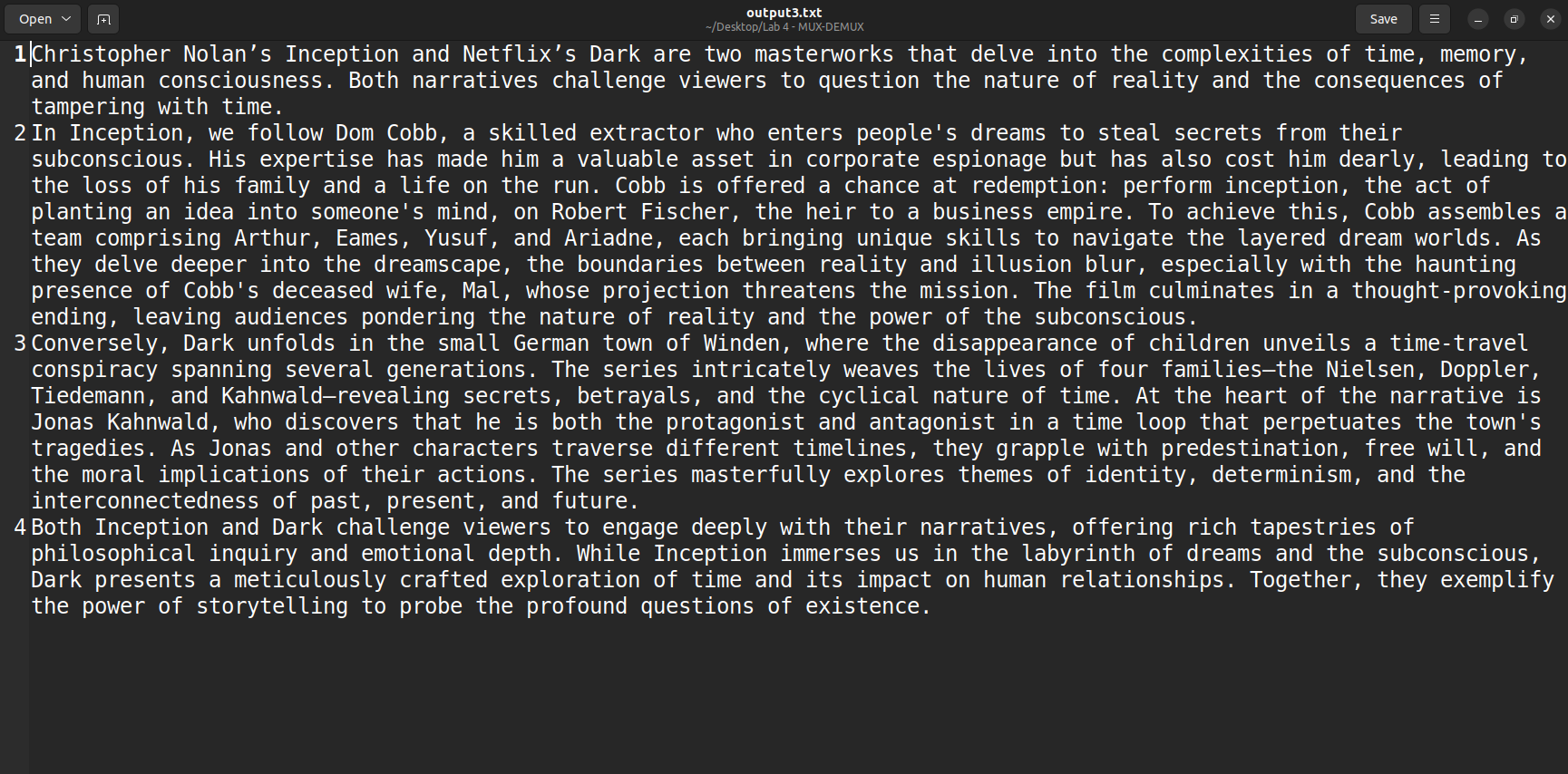


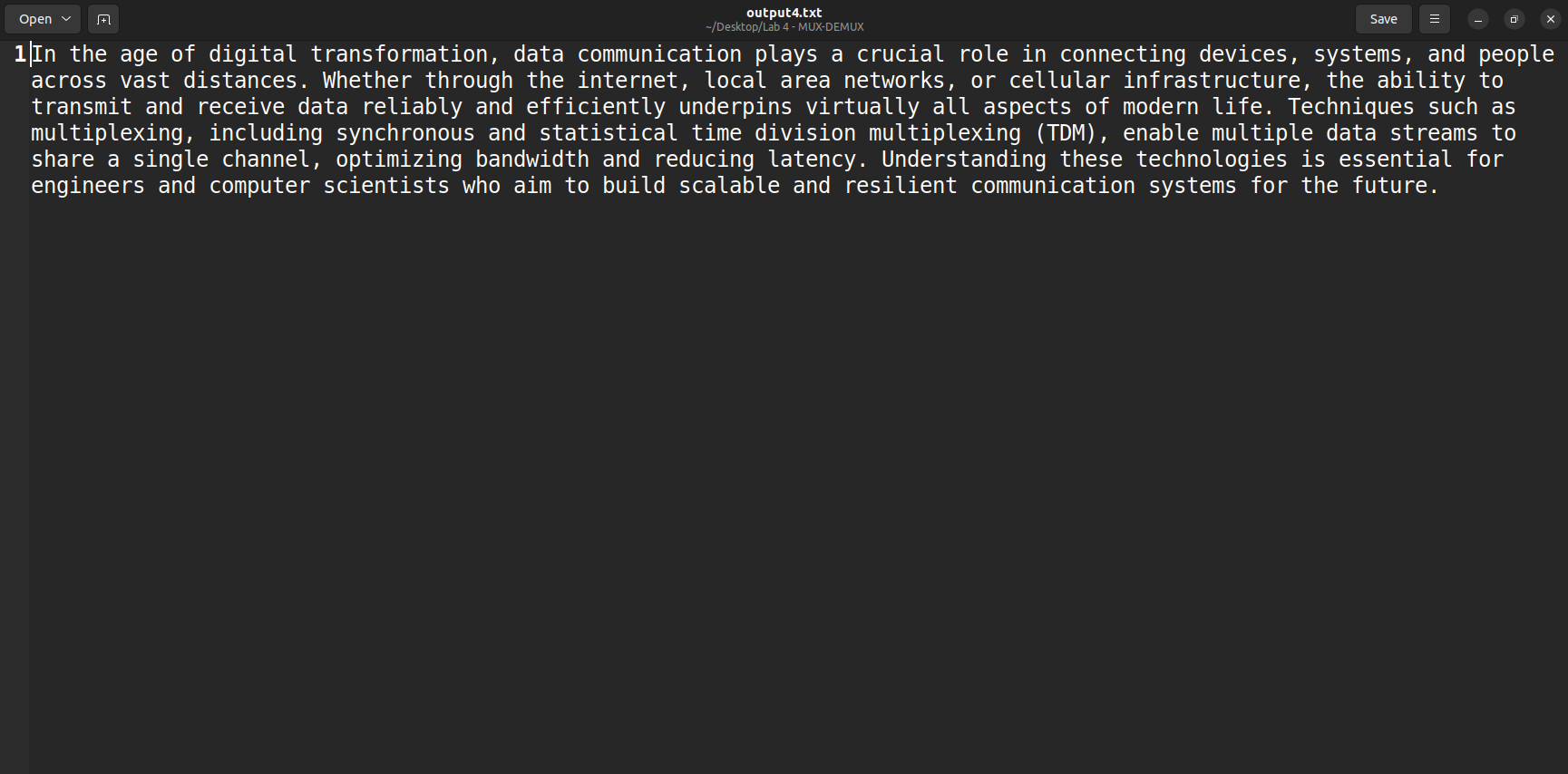
## 5.2 Server side - Demultiplexing











6. Discussion

**Comparative Analysis: Synchronous TDM vs Statistical TDM**

|  |  |  |  |
| --- | --- | --- | --- |
| | Aspect | Synchronous TDM | Statistical TDM | | --- | --- | --- | |
| |  |  |  | | --- | --- | --- | | Slot Allocation | Fixed time slots for each channel, even if no data is present | Dynamic time slots are assigned only to active channels | |
| |  |  |  | | --- | --- | --- | | Efficiency | Less efficient—bandwidth is wasted when sources have no data | More efficient—bandwidth is utilized only by active sources | |
| |  |  |  | | --- | --- | --- | | Complexity | Simpler to design and implement | More complex due to the need for dynamic slot management | |
| |  |  |  | | --- | --- | --- | | Overhead | Minimal control information required (frame synchronization only) | Higher overhead due to need for addressing or headers in each data packet | |
| |  |  |  | | --- | --- | --- | | Latency | Predictable and constant latency | Variable latency depending on traffic load and scheduling | |
| |  |  |  | | --- | --- | --- | | Bandwidth Utilization | Poor under low or bursty traffic conditions | High under variable or bursty traffic conditions | |
| |  |  |  | | --- | --- | --- | | Suitability | Suitable for systems with constant, predictable traffic | Suitable for systems with bursty or unpredictable traffic patterns | |

Analysis in Terms of Implementation:

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Synchronous TDM Implementation** | **Statistical TDM Implementation** |
| Framing | Simple cyclic frame structure; each sender's slot appears in a fixed order | Variable-length frames or packets with address headers required |
| Multiplexer Logic | Cyclic counter and fixed scheduling logic | Requires buffer management, dynamic scheduling, and addressing |
| Demultiplexer Logic | Can directly split data by a fixed position in the frame | Must parse the header to identify the destination, then route accordingly |
| Error Handling | Easier to detect synchronization loss | Must include mechanisms to handle dropped or misrouted packets |
| Scalability | Scales poorly with many idle channels | Scales better, as bandwidth is shared more flexibly |
| Development Complexity | Easier to code and test; predictable behavior | Requires additional logic for dynamic allocation and parsing |

# 7. Learning and Difficulties

## 7.1 Learning

* Practical implementation of multiplexing and demultiplexing.
* Socket communication using Java.
* Sending data through Statistical TDM
* How statistical multiplexing minimizes idle transmission.

## 7.2 Difficulties

* Managing EOF
* Designing dynamic frame creation and parsing logic.
* Debugging socket-based communication errors.

8. Conclusion

The implementation of Statistical Time Division Multiplexing (Stat-TDM) in this experiment provided valuable insights into how data from multiple sources can be efficiently transmitted over a shared communication channel. Unlike Synchronous TDM, where fixed time slots are assigned to each source regardless of data availability, Stat-TDM dynamically allocates slots only to active sources. This approach minimizes bandwidth waste and increases overall transmission efficiency, especially when dealing with input sources that have uneven or intermittent data.

Through this lab, we gained hands-on experience in managing multiple data streams, constructing and parsing multiplexed frames, and handling client-server communication using Java sockets. We also learned how to include identifying headers (stream IDs) with each data unit to allow accurate demultiplexing at the receiving end. This experiment closely mirrors how real-world systems optimize network bandwidth, such as in packet-switched networks and modern telecommunications.

The lab also highlighted the additional complexity involved in Stat-TDM, including handling variable-length frames, managing file EOF conditions, and dynamically building transmission units. Despite these challenges, the final implementation successfully reconstructed the original data streams with accuracy.

In summary, this lab not only reinforced theoretical knowledge of multiplexing and demultiplexing but also strengthened practical programming skills in socket communication, data serialization, and system-level I/O management. Such foundational understanding is essential for designing scalable and efficient data communication systems in the real world.