

**University Of Dhaka  
Department of Computer Science & Engineering**

CSE-2213: Data and Telecommunication Lab

2nd Year 2nd Semester

Session: 2022-2023

**Exp No. 1:** Implementation of Multiplexing and Demultiplexing Using Statistical TDM

**Submitted by:**

1. Farhan Bin Rabbani (Roll: FH-35)
2. Tamal Kanti Sarker (Roll: JN-39)

**Submitted to:**

1. Dr. Md. Mustafizur Rahman, Professor, Dept. of CSE, DU
2. Mr. Palash Roy, Lecturer, Dept. of CSE, DU

**Lab Report 2**

**Lab Report Name:**  Implementation of Multiplexing and Demultiplexing Using Statistical TDM

**Introduction:** Multiplexing is a technique that combines multiple signals into one signal over a shared medium. On the other hand, Demultiplexing is the reverse process of multiplexing, as it separates the combined signal and delivers it to the appropriate receiver or process based on identifiers like port numbers or channel IDs.

Multiplexing and Demultiplexing processes optimize bandwidth use by sharing a single physical channel among multiple logical streams. Multiple phone calls being transmitted over a single fiber optic cable, or multiple video/audio streams being combined over one internet connection, are some real-life examples of why Multiplexing and Demultiplexing are necessary. Different types of Multiplexing, as well as their corresponding Demultiplexing, are as follows,

**1. Time Division Multiplexing (TDM)** – Assigns fixed time slots to each signal. Each sender is allocated a fixed time slot in a round-robin fashion. If the sender has no data, the time slot may be wasted (in Synchronous TDM) or reassigned (in Statistical TDM).

**2. Frequency Division Multiplexing (FDM)** – Assigns different frequency bands. Different signals are transmitted over different frequency bands within the same channel. It is widely used in radio, television broadcasting, and cable TV.

**3. Statistical Multiplexing** – Dynamically allocates time slots based on demand.

**4. Wavelength Division Multiplexing (WDM)** – Used in fiber optics, with different wavelengths.

Time Division Multiplexing (TDM) has two subtypes:

**Synchronous TDM:** Each input channel is assigned a fixed time slot in a repeating sequence, regardless of whether it has data to send or not.

**Statistical TDM:** Time slots are dynamically allocated to input channels based on demand, meaning only channels with data are given slots, improving efficiency.

**Objectives:**

* To gather knowledge on the concept of Multiplexing (MUX) and Demultiplexing (DEMUX) in 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.

**Algorithms/Pseudocode:**

**Client-Side Algorithm for Multiplexing**

* We take the Server IP address, the number of input files (N) with their names, and the Time slot size (T) as inputs.
* We create an array of BufferedInputStream objects, a “done” array to track the number of read files, and a counter variable “activeFiles” to check the number of active files.
* We create a packet array of size N\*T to hold T bytes from each file and a Boolean flag “hasRealData” to check whether there is any actual data.
* For each stream i:

1. If the file is already done, we fill its slot in the packet with the filler byte (#).
2. If the file isn’t done, we read up to T bytes in the stream. If data is found, we store it in the corresponding section of the packet. If ‘EOF’ is detected, we pad the rest of the bytes with ‘#’ and mark done[i] = true.
3. If a file is finished, we decrement “activeFiles”.

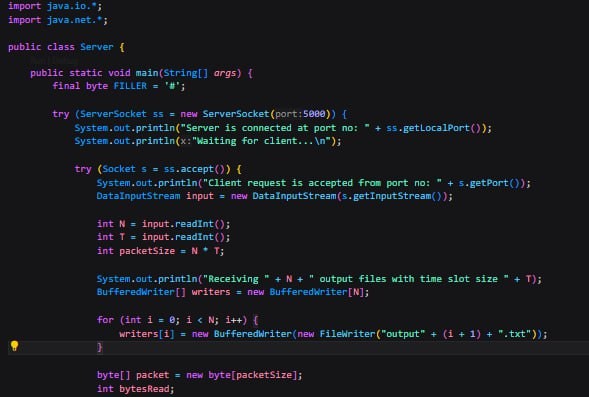
* We stop when all files have reached EOF.

**Server-Side Algorithm for Demultiplexing**

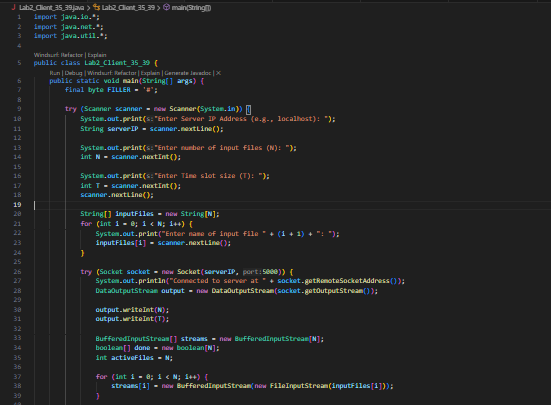
* After receiving the number of files (N) and the Time slot (T) from the client, we create N output files.
* We loop through the sent packet.

1. We extract T bytes from the packet for each time slot.
2. We write only the non-filler bytes to the corresponding output files.

**Implementation:**

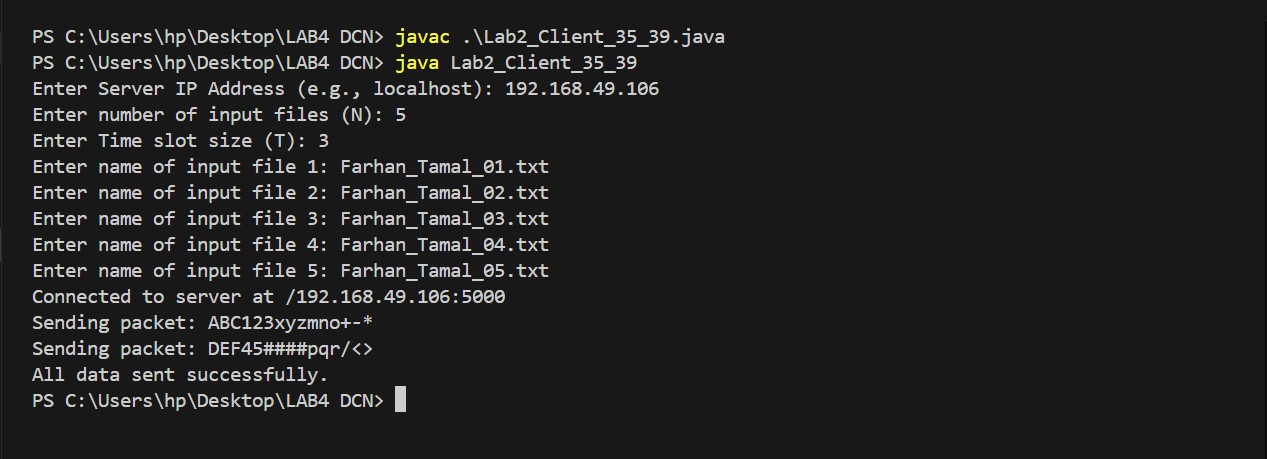
****

**Fig:** Server-Side

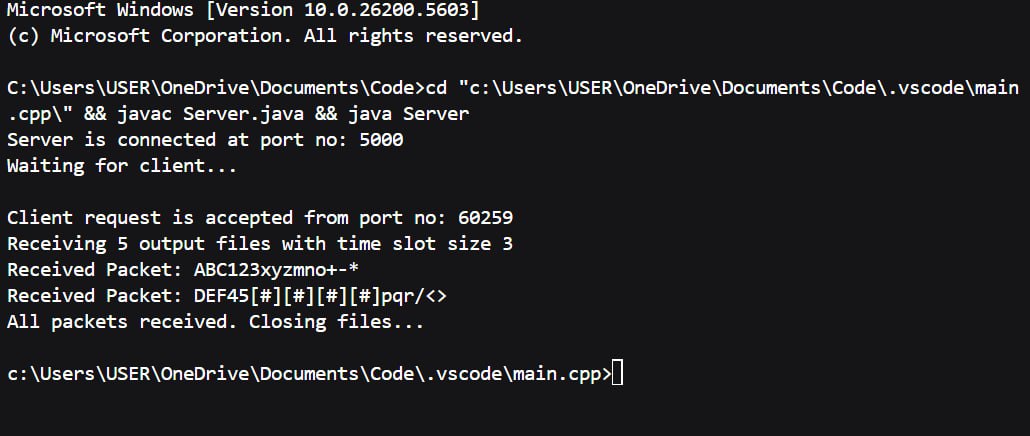
****

**Fig:** Client-Side

**Result Analysis:**

****

**Fig:** Client-Side



**Fig:** Server-Side

The client reads a fixed chunk of bytes (T) from every ‘N’ input file and pads ‘#’ if a file reaches EOF. The constructed packet is sequentially sent to the server.

The server demultiplexes the received packets into individual output files, ignoring the ‘#’s. This reconstruction gives us the original files.

**Discussion:**

**Synchronous TDM**

* Fixed time slots for each input channel, regardless of data availability
* Simple – fixed schedule, no need to identify data source
* Easy – fixed time slots make synchronization straightforward
* Fixed-size frames sent in round-robin order
* Best for uniform or real-time traffic (e.g., audio, video)

**Statistical TDM**

* Time slots are dynamically assigned only to channels with data
* More complex – requires addressing the info to identify the sender
* Harder – dynamic slots require more synchronization logic
* Variable-size frames with headers identifying the source
* Best for bursts or irregular traffic (e.g., computer data, file transfers)

**Implementation of Sync TDM**

* **Sender:** Always sends data in fixed order. If a channel has no data, filler data (#) is padded.
* **Receiver:** Reads the incoming stream and splits it into fixed slots assigned to known channels. Filler data is ignored.

**Implementation of Stat TDM**

* **Sender:** Before each transmission, the presence of data is checked. Only active channel data is sent along with the source ID.
* **Receiver:** This reads the label to identify the corresponding channel and routes the data to the appropriate file.

**Learning and Difficulties:**

* Packet construction logic must be checked with caution, otherwise the wrong sequence of data stream may be sent.
* EOF must be handled with care, otherwise padding (#) may be done incorrectly, and data may be skipped.
* Difficulty in keeping track of which part of the packet belongs to which channel must be considered; otherwise, data may be incorrectly received on the receiver’s end.

**Conclusion:**

We were successful in the implementation of Multiplexing and Demultiplexing using Statistical TDM, as well as learning about their necessity and applications in datacom.