Multiplexing and Demultiplexing

Study-Ready Notes

Compiled by Andrew Photinakis

November 15, 2025

Contents

1 Introduction to Multiplexing and Demultiplexing

- Part of transport-layer services in the protocol stack
- Essential for managing multiple simultaneous network connections
- Works alongside other transport layer functions:
  - Connectionless transport: UDP
  - Principles of reliable data transfer
  - Connection-oriented transport: TCP
  - Principles of congestion control
  - TCP congestion control
  - Evolution of transport-layer functionality

[Summary: Multiplexing and demultiplexing are fundamental transport layer services that enable multiple applications to share network connections by properly directing data to the correct processes.]

2 Basic Concepts

2.1 Definitions

- **Multiplexing at sender**: Handling data from multiple sockets and adding transport header for later demultiplexing
- **Demultiplexing at receiver**: Using header information to deliver received segments to the correct socket

[Mnemonic: "Mix Up, Sort Out" - Multiplexing mixes data from multiple sources, demultiplexing sorts it out to the right destinations.]

3 How Demultiplexing Works

3.1 Key Components

- Host receives IP datagrams with:
  - Source IP address and destination IP address
  - Each datagram carries one transport-layer segment
  - Each segment has source and destination port numbers
- Host uses **IP addresses and port numbers** to direct segments to appropriate sockets

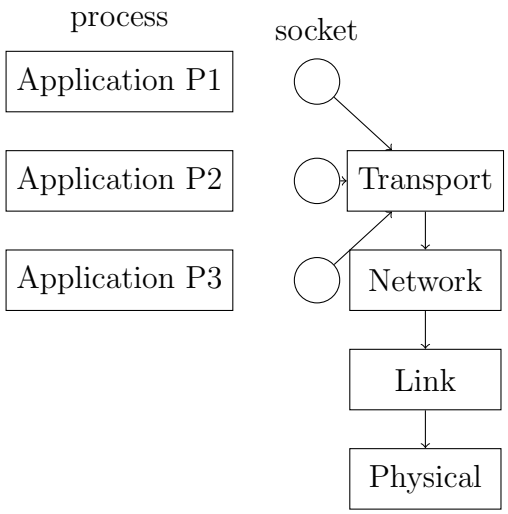

Figure 1: Multiplexing: Multiple application processes sending data through sockets to transport layer

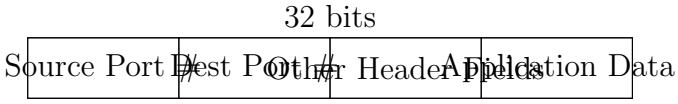

Figure 2: TCP/UDP segment format showing key fields for demultiplexing

3.2 Transport Segment Format

[Summary: Demultiplexing uses IP addresses and port numbers from segment headers to route incoming data to the correct application socket, ensuring each process receives its intended data.]

4 Connectionless Demultiplexing (UDP)

4.1 Socket Creation and Configuration

- When creating UDP socket, must specify *host-local* port number:

```
DatagramSocket mySocket1 = new DatagramSocket(12534);
```

- When creating datagram to send, must specify:
  - Destination IP address
  - Destination port number

4.2 Demultiplexing Process

- Receiving host checks destination port number in UDP segment

- Directs UDP segment to socket with that port number
- **Important**: IP/UDP datagrams with same destination port number but different source IP addresses and/or source port numbers are directed to the same socket

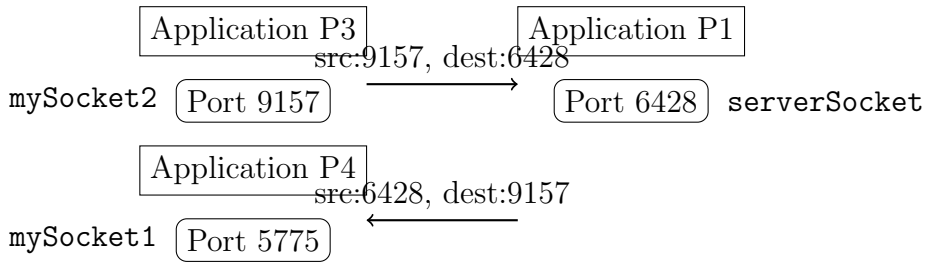

Figure 3: Connectionless demultiplexing example showing UDP socket communication

5 Connection-Oriented Demultiplexing (TCP)

5.1 TCP Socket Identification

- TCP socket identified by 4-tuple:
  - Source IP address
  - Source port number
  - Destination IP address
  - Destination port number
- Demultiplexing uses **all four values** to direct segment to appropriate socket

5.2 Server Socket Management

- Server may support many simultaneous TCP sockets
- Each socket identified by its own 4-tuple
- Each socket associated with a different connecting client

[Concept Map: Demultiplexing → UDP (destination port only) vs TCP (4-tuple: src/dest IP+port) → Enables multiple simultaneous connections → Essential for web servers handling multiple clients.]

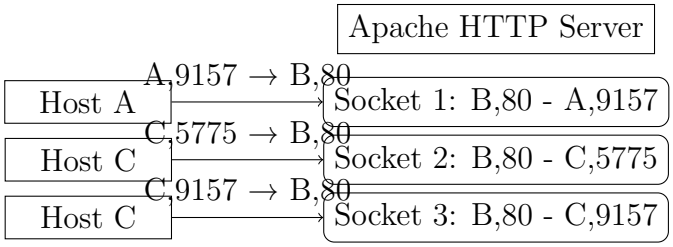

Figure 4: Connection-oriented demultiplexing: Three segments with same destination (B,80) but different sources are directed to different sockets

6 Real-World Analogies

6.1 Airport Security Checkpoint

- **Multiplexing**: Multiple passengers from different flights going through same security checkpoint
- **Demultiplexing**: Passengers directed to correct gates based on flight information
- TSA Pre vs Main Checkpoint: Different processing based on passenger type

6.2 Airline Boarding

- **Multiplexing**: All passengers for different flights in same terminal
- **Demultiplexing**: Passengers directed to specific gates (B14, 201E) based on destination
- Priority vs Economy: Different boarding groups within same flight

6.3 Highway System

- **Multiplexing**: Multiple routes converging into main highways
- **Demultiplexing**: Exits directing traffic to specific destinations (14th St Downtown, Broadway)

7 Key Differences: UDP vs TCP Demultiplexing

[Summary: UDP uses simple destination port-based demultiplexing where all traffic for a port goes to one socket, while TCP uses 4-tuple identification creating separate sockets for each connection, enabling multiple simultaneous conversations.]

Analogy	Multiplexing	Demultiplexing
Airport Security	All passengers through same checkpoint	Directed to correct gates based on flight
Highway System	Multiple routes merge into highway	Exits direct to specific streets
Postal System	All mail collected in same mailbox	Sorted by address for delivery

Table 1: Real-world analogies for multiplexing and demultiplexing

Connectionless (UDP)	Connection-Oriented (TCP)
Uses only destination port number	Uses 4-tuple: source/destination IP and port
Same socket receives all datagrams for that port	Separate socket for each connection
No connection establishment required	Connection setup before data transfer
Stateless demultiplexing	Stateful demultiplexing
Suitable for broadcast/multicast	Point-to-point communication
Example: DNS queries	**Example**: HTTP web traffic

Table 2: Comparison of UDP vs TCP demultiplexing approaches

8 Study Aids

8.1 Key Concepts to Master

- Understand the difference between multiplexing and demultiplexing
- Memorize the segment format and which fields are used for demultiplexing
- Know the exact 4-tuple used for TCP socket identification
- Be able to explain why TCP needs more complex demultiplexing than UDP
- Understand real-world analogies and how they relate to network concepts

[Mnemonic: "UDP: Simple Port, TCP: Four Report" - UDP uses simple port numbers, TCP requires four pieces of information (the 4-tuple).]

9 Summary

- Multiplexing and demultiplexing are based on segment/datagram header field values
- **UDP**: Demultiplexing using destination port number only

- **TCP**: Demultiplexing using 4-tuple (source and destination IP addresses and port numbers)
- Multiplexing/demultiplexing happen at all layers of the network stack
- These mechanisms enable multiple applications to share network resources efficiently