Timing Diagram of Connection Management in TCP and UDP:

1. TCP:

TCP uses a three-way handshake to establish a connection and a four-way handshake to terminate

it. Here's a basic timing diagram for connection establishment and termination in TCP.

Client Server

------ ------

SYN -------------------------------->

<-------------------------- SYN, ACK

ACK -------------------------------->

... data transfer ...

FIN -------------------------------->

<-------------------------------- ACK

<--------------------------------- FIN

ACK -------------------------------->

- SYN: The client sends a `SYN` packet to the server to request a connection.

- SYN, ACK: The server responds with a `SYN, ACK` packet to accept the connection.

- ACK: The client sends an `ACK` packet to acknowledge the server's acceptance.

To terminate the connection:

- FIN: One side (let's say the client) sends a `FIN` packet indicating it's finished sending data.

- ACK: The other side (server) sends an `ACK` to acknowledge the `FIN`.

- FIN: The server then sends its own `FIN` packet.

- ACK: The client sends an `ACK` to acknowledge the server's `FIN`.

2. UDP:

UDP is a connectionless protocol, so there's no formal connection setup or teardown process. The

timing diagram for UDP is simpler:

Client Server

------ ------

Data ---------------------------->

<-------------- Data (optional)

- A client can just start sending data packets to the server without a formal handshake.

- The server might respond with data, but this is specific to the application and not a part of the UDP protocol itself.

Congestion in TCP:

Yes, TCP can experience network congestion, which can lead to packet losses and reduced

throughput.

Congestion Handled in TCP:

1. Congestion Control Algorithms: TCP uses various algorithms like Slow Start, Congestion Avoidance,

and Fast Retransmit to detect and handle congestion. The basic idea is to adjust the congestion

window size based on perceived network conditions.

2. Slow Start: Initially, TCP starts with a small congestion window and doubles its size every round trip

time (RTT) until it detects packet loss or reaches a threshold.

3. Congestion Avoidance (Additive Increase): TCP enters congestion avoidance mode once the

threshold is reached. In this mode, the congestion window is increased linearly (by one Maximum

Segment Size (MSS) every RTT) until packet loss is detected.

4. Fast Retransmit: If TCP detects packet loss (often through three duplicate ACKs), it will retransmit

the missing segment without waiting for a timer.

5. Fast Recovery: After fast retransmitting, instead of collapsing the congestion window size to the beginning, TCP reduces it to half (multiplicative decrease) and then resumes congestion avoidance.

6. Retransmission Timeouts: If an acknowledgment isn't received within an expected timeframe, TCP

assumes the packet was lost and retransmits it, reducing the congestion window.

7. Explicit Congestion Notification (ECN): Allows routers to notify endpoints about network

congestion proactively so they can reduce their transmission rate before packet losses occur.

Through these mechanisms, TCP can adapt its data transfer rate based on the current network

conditions, ensuring efficient and fair use of the network.

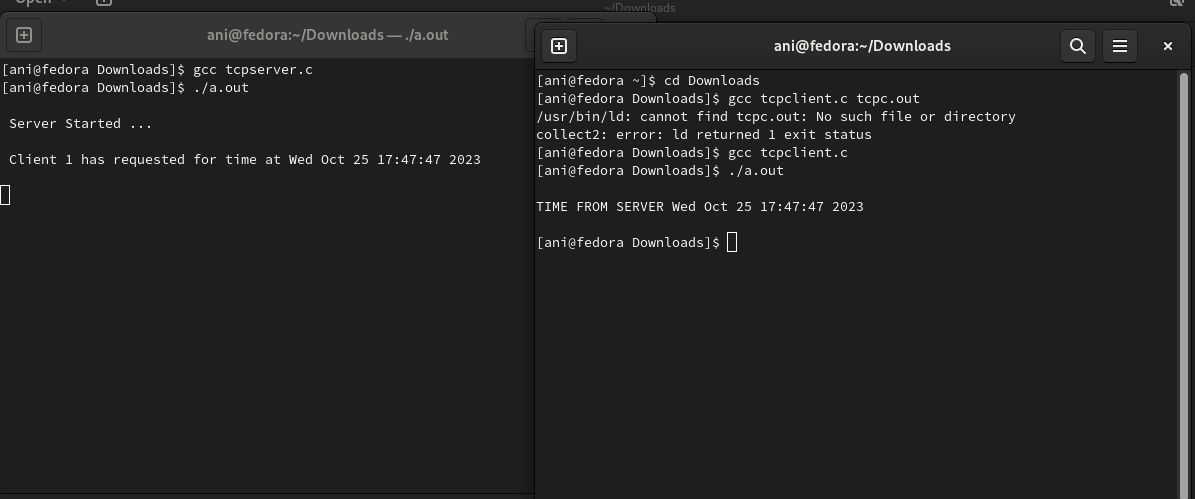
TCP server.c



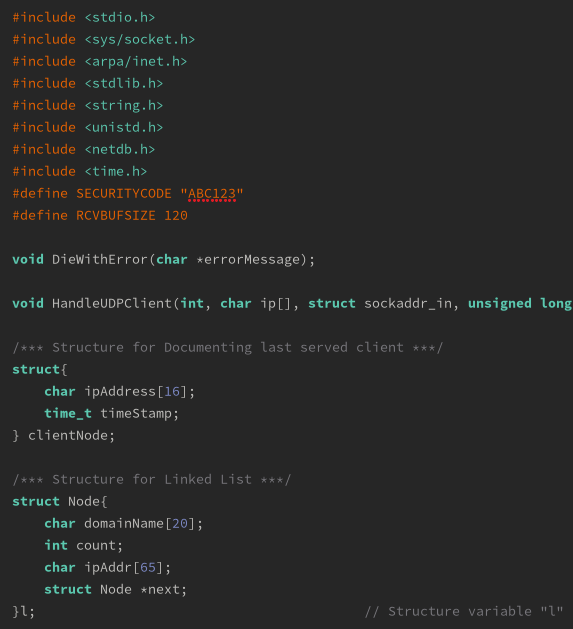
TCPclient.c



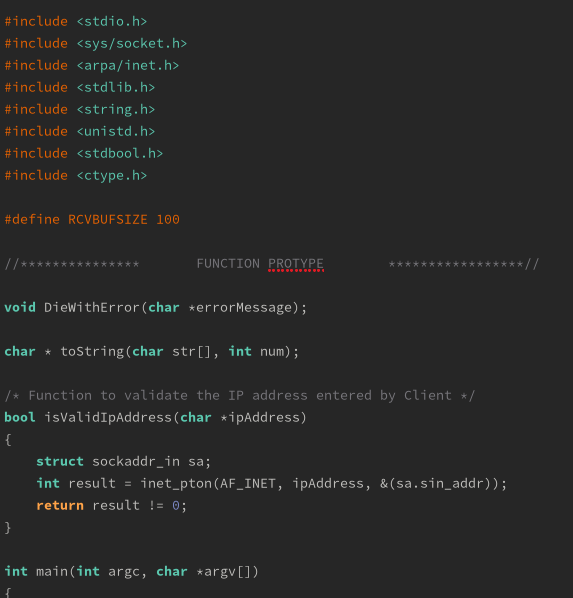
Output:



UDPClient.c



UDPServer.c



Output:

