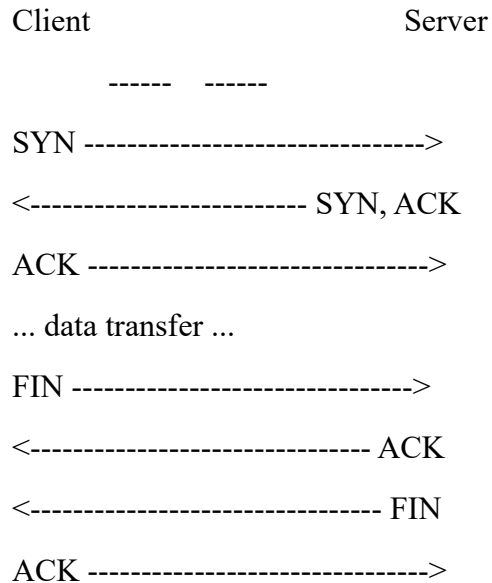


## Timing Diagram of Connection Management in TCP and UDP:

### 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.



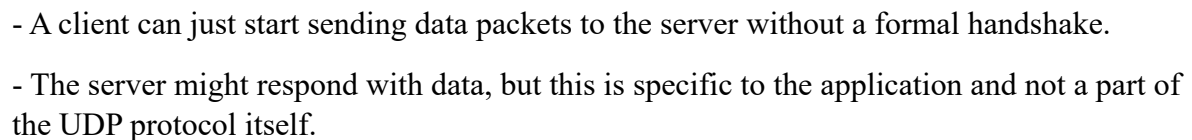
- SYN: The client sends a 'SYN' packet to the server to request a connection.
- SYN, ACK: The server responds with a 'SYN, ACK' packet to accept the connection.
- ACK: The client sends an 'ACK' packet to acknowledge the server's acceptance.

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'.

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

Client                      Server



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

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

window size based on perceived network conditions.

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

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

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

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

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<time.h>

//port value and the max size for buffer
#define PORT 9002 //the port users will be connecting to
#define BACKLOG 10 //how many pending connections queue will hold

int main(){
    //*****Time Setup*****
    time_t currentTime ;
    time(&currentTime);
    //*****

    int countClient = 0;

    int socket_descriptor = socket(AF_INET, SOCK_STREAM, 0);

    struct sockaddr_in serverAddress;
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_addr.s_addr=INADDR_ANY;
    serverAddress.sin_port=htons(PORT);

    //binding address
    bind(socket_descriptor, (struct sockaddr*)&serverAddress, sizeof(serverAddress));

    //listening to the queue
```

## TCPclient.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include "string.h"

#define PORT 9002 //the port users will be connecting to
#define MAXLINE 30 //for buffer size

int main(){

    int socket_descriptor = socket(AF_INET, SOCK_STREAM, 0);

    char serverResponse[MAXLINE];

    struct sockaddr_in serverAddress;

    serverAddress.sin_family = AF_INET;
    serverAddress.sin_addr.s_addr = INADDR_ANY;
    serverAddress.sin_port = htons(PORT);

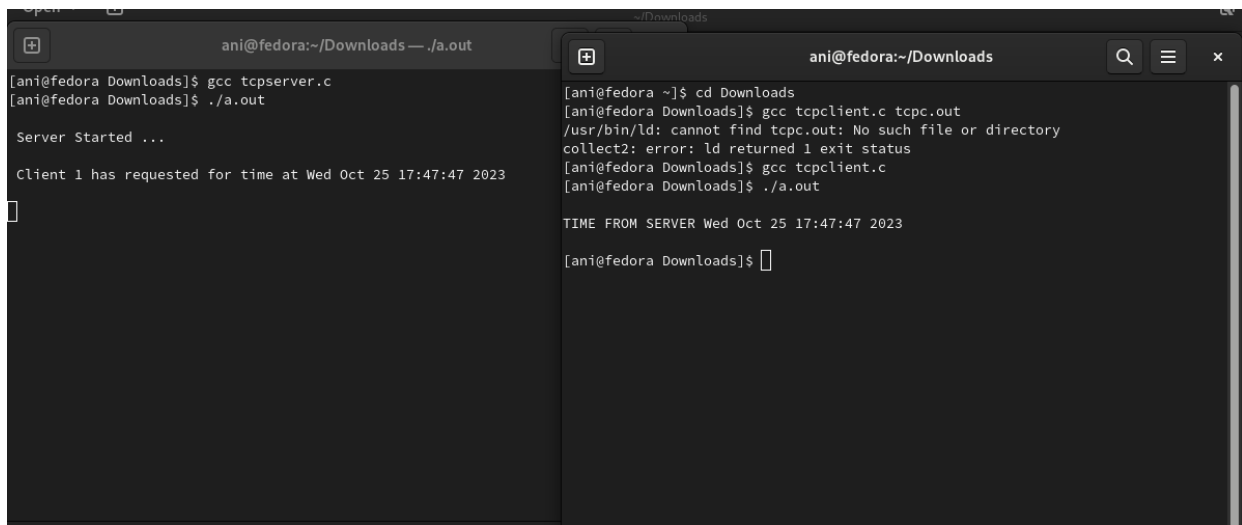
    connect(socket_descriptor, (struct sockaddr *)&serverAddress, sizeof(serverAddress));

    recv(socket_descriptor, serverResponse, MAXLINE-1, 0);

    printf("\nTIME FROM SERVER %s\n", serverResponse);

    return 0;
}
```

## Output:



The image shows two terminal windows side-by-side. The left window is titled 'ani@fedora:~/Downloads — ./a.out' and shows the compilation of 'tcpserver.c' and its execution. It outputs 'Server Started ...' and 'Client 1 has requested for time at Wed Oct 25 17:47:47 2023'. The right window is titled 'ani@fedora:~/Downloads' and shows the compilation of 'tcpclient.c' with an error, followed by its successful execution which outputs the time received from the server.

```
[ani@fedora Downloads]$ gcc tcpserver.c
[ani@fedora Downloads]$ ./a.out

Server Started ...

Client 1 has requested for time at Wed Oct 25 17:47:47 2023

[ani@fedora ~]$ cd Downloads
[ani@fedora Downloads]$ gcc tcpclient.c tcpclient.out
/usr/bin/ld: cannot find tcpclient.out: No such file or directory
collect2: error: ld returned 1 exit status
[ani@fedora Downloads]$ gcc tcpclient.c
[ani@fedora Downloads]$ ./a.out

TIME FROM SERVER Wed Oct 25 17:47:47 2023

[ani@fedora Downloads]$
```

## UDPClient.c

```
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <netdb.h>
#include <time.h>
#define SECURITYCODE "ABC123"
#define RCVBUFSIZE 120

void DieWithError(char *errorMessage);

void HandleUDPClient(int, char ip[], struct sockaddr_in, unsigned long

/** Structure for Documenting last served client */
struct{
    char ipAddress[16];
    time_t timeStamp;
} clientNode;

/** Structure for Linked List */
struct Node{
    char domainName[20];
    int count;
    char ipAddr[65];
    struct Node *next;
};

// Structure variable "l"
```

## UDPServer.c

```
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <stdbool.h>
#include <ctype.h>

#define RCVBUFSIZE 100

//***** FUNCTION PROTYPE *****//

void DieWithError(char *errorMessage);

char * toString(char str[], int num);

/* Function to validate the IP address entered by Client */
bool isValidIpAddress(char *ipAddress)
{
    struct sockaddr_in sa;
    int result = inet_pton(AF_INET, ipAddress, &(sa.sin_addr));
    return result != 0;
}

int main(int argc, char *argv[])
{
```

Output:

```
animeshraj@fedora:~  
[animeshraj@fedora ~]$ gcc dnsserver.c -o dns_server  
[animeshraj@fedora ~]$ gcc dnsclient.c -o dns_client
```

```
animeshraj@fedora:~ — ./dns_server  
[animeshraj@fedora ~]$ ./dns_server  
DNS Server started on port 9002  
Received query for domain: google.com  
Received query for domain: instagram.com
```

```
animeshraj@fedora:~  
[animeshraj@fedora ~]$ ./dns_client google.com  
IP address for google.com is 127.0.0.1  
[animeshraj@fedora ~]$ ./dns_client instagram.com  
IP address for instagram.com is 127.0.0.1
```