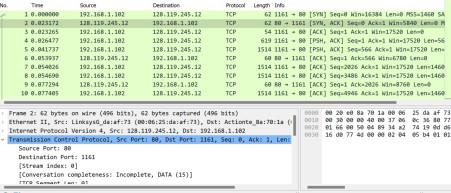
**Lab 9**

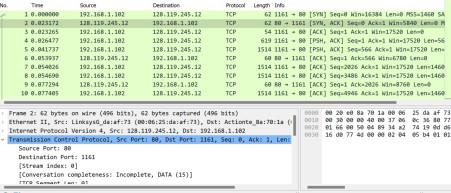
**LAB STATEMENT 1**

**Question NUMBER 1**

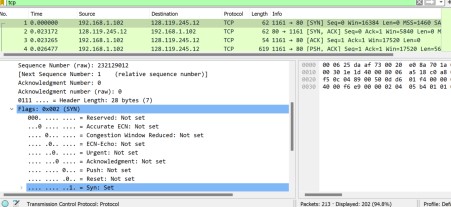
Tcp Source port=1161



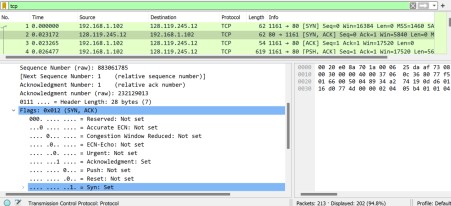
QUESTION 2



Question no 3



Question 4



**Question 5:** In packet 9, **Ack = 2026** and **Seq = 1**. Explain these values? Client acknowledges and shows that 2025 segments have been successfully received in 1st sequence of data transfer.

**Question 6:** In packet 16, **Ack = 7866** and **Seq = 1**. Explain these values? Client acknowledges and shows that up to 7865 segments have been successfully

received in sequence number 1 of data transfer

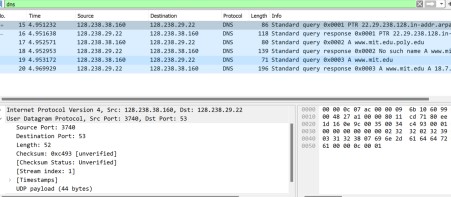
**Question 7:** Why Wireshark uses relative sequence and ack?

It uses relative sequence numbers for improvement of readability and usability. This makes numbers easier to read and compare for catching up errors and network anomalies

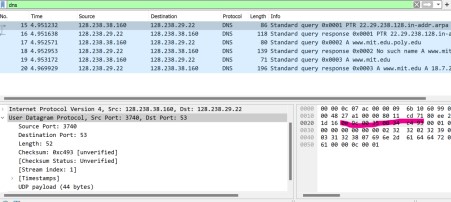
**LAB STATEMENT 2**

**Question 1**

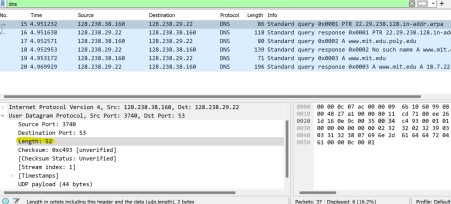
# **4, source, dest,length,checksum**



# Question 2



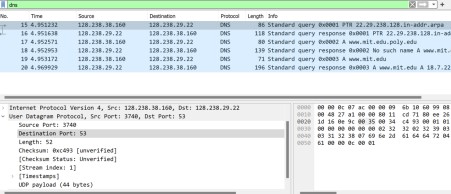
# Question no 3



**Length is 52 8 of the header and else 44 bytes are encapsulated data from network layer**

**Question no 4**

# 53



SERVER

#include <stdio.h>

#include <string.h>

#include <sys/socket.h> //socket

#include <arpa/inet.h> //inet\_addr #include <pthread.h>

int opentimes=0; void \*client\_handler(void \*arg) {

int client\_sock = \*((int\*)arg);

char buffer[1024];

int bytes\_received;

char checkbuffer[1024];

strcpy(checkbuffer, "DISCONNECT");

printf("OPEN times %d\n",opentimes);

opentimes++;

while (1) {

bytes\_received = recv(client\_sock, buffer, sizeof(buffer), 0);

if (strcmp(buffer,checkbuffer)==0) {

printf("Client disconnected\n");

close(client\_sock);

break;

}

buffer[bytes\_received] = '\0';

//strcat(buffer," :Server Response\0");

send(client\_sock, buffer, bytes\_received, 0);

}

close(client\_sock);

pthread\_exit(NULL);

}

int main(void)

{

int socket\_desc, client\_sock, client\_size;

struct sockaddr\_in server\_addr, client\_addr; //SERVER ADDR will have all the

server address

char server\_message[2000], client\_message[2000]; // Sending values from the

server and receive from the server we need this

pthread\_t threads[3];

//Cleaning the Buffers

memset(server\_message,'\0',sizeof(server\_message));

memset(client\_message,'\0',sizeof(client\_message)); // Set all bits of the padding

field//

//Creating Socket

socket\_desc = socket(AF\_INET, SOCK\_STREAM, 0);

if(socket\_desc < 0)

{

printf("Could Not Create Socket. Error!!!!!\n");

return -1;

}

printf("Socket Created\n");

//Binding IP and Port to socket

server\_addr.sin\_family = AF\_INET; /\* Address family = Internet \*/

server\_addr.sin\_port = htons(3000); // Set port number, using htons function to

use proper byte order \*/

server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1"); /\* Set IP address to localhost \*/

// BINDING FUNCTION

if(bind(socket\_desc, (struct sockaddr\*)&server\_addr, sizeof(server\_addr))<0)

{

printf("Bind Failed. Error!!!!!\n");

return -1;

}

printf("Bind Done\n");

//Put the socket into Listening State

int index\_thread=0;

if(listen(socket\_desc, 1) < 0)

{

printf("Listening Failed. Error!!!!!\n");

return -1;

}

printf("Listening for Incoming Connections.....\n");

//Accept the incoming Connections

client\_size = sizeof(client\_addr);

int index=0;

while(index<4){

client\_sock = accept(socket\_desc, (struct sockaddr\*)&client\_addr, &client\_size);

// heree particular client k liye new socket create kr rhaa ha

if (client\_sock < 0)

{

printf("Accept Failed. Error!!!!!!\n");

return -1;

}

printf("Client Connected with IP: %s and Port No: %i\n",

inet\_ntoa(client\_addr.sin\_addr), ntohs(client\_addr.sin\_port));

if (pthread\_create(&threads[index], NULL, client\_handler, &client\_sock) != 0) {

perror("Thread creation failed");

continue;

}

index++;

}

memset(server\_message,'\0',sizeof(server\_message));

memset(client\_message,'\0',sizeof(client\_message));

//Closing the Socket

close(client\_sock);

close(socket\_desc);

return 0;

## CLIENT

**#include <stdio.h>**

**#include <string.h>**

**#include <sys/socket.h> #include <arpa/inet.h> int main() {**

**int socket\_desc;**

**struct sockaddr\_in server\_addr;**

**char server\_message[2000], client\_message[2000]; memset(server\_message, '\0', sizeof(server\_message));**

**memset(client\_message, '\0', sizeof(client\_message));**

**socket\_desc = socket(AF\_INET, SOCK\_STREAM, 0); printf("Socket Created\n"); server\_addr.sin\_family = AF\_INET;**

**server\_addr.sin\_port = htons(3000);**

**server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");**

**if (connect(socket\_desc, (struct sockaddr \*)&server\_addr,**

**sizeof(server\_addr)) == -1)**

**{**

**printf("Connection Failed. Error!!!!!\n");**

**return -1;**

**}**

**printf("Connected\n");**

**while (1)**

**{**

**// Get input from the user**

**printf("Enter Message (or 'DISCONNECT' to exit): ");**

**fgets(client\_message, sizeof(client\_message), stdin);**

**// Send the message to the server**

**if (send(socket\_desc, client\_message, strlen(client\_message), 0) ==-1)**

**{**

**printf("Send Failed. Error!!!!\n");**

**return -1;**

**}**

**if (strcmp(client\_message, "DISCONNECT\n") == 0)**

**{**

**printf("Disconnecting from the server\n");**

**break;**

**}**

**// Receive the message back from the server**

**if (recv(socket\_desc, server\_message, sizeof(server\_message), 0) < 0)**

**{**

**printf("Receive Failed. Error!!!!!\n");**

**return -1;**

**}**

**printf("Server Response: %s\n", server\_message); memset(server\_message, '\0', sizeof(server\_message));**

**memset(client\_message, '\0', sizeof(client\_message));**

**}**

**// Closing the socket**

**close(socket\_desc);**

**return 0;**

**}**

