**Week 5: Iterative Socket Programming in UDP and TCP**

Komal Mathur, CSE B2, 220905546

**My notes**

1. bind – to name socket
2. error handling very important in these types of codes
3. ip config - to get ip address of host
4. int newsockid = accept(sockid, &clientAddr, &addrLen); --these are output arguments, not input
5. To get ip address of computer: $ip address
6. TCP – ephimeral port number needs to be changed for each run, as long as you dont stop the server you dont have to change the port number (only use **ctrl+C** to terminate a program otherwise port number gets blocked)
7. UDP – ephimeral may be same between runs

**Sample TCP Program:** Write an iterative TCP client server program where client sends a message to server and server echoes back the message to client. Client should display the original message and echoed message. Note: As socket is also a file descriptor, we can use read and write system calls to receive and send data.

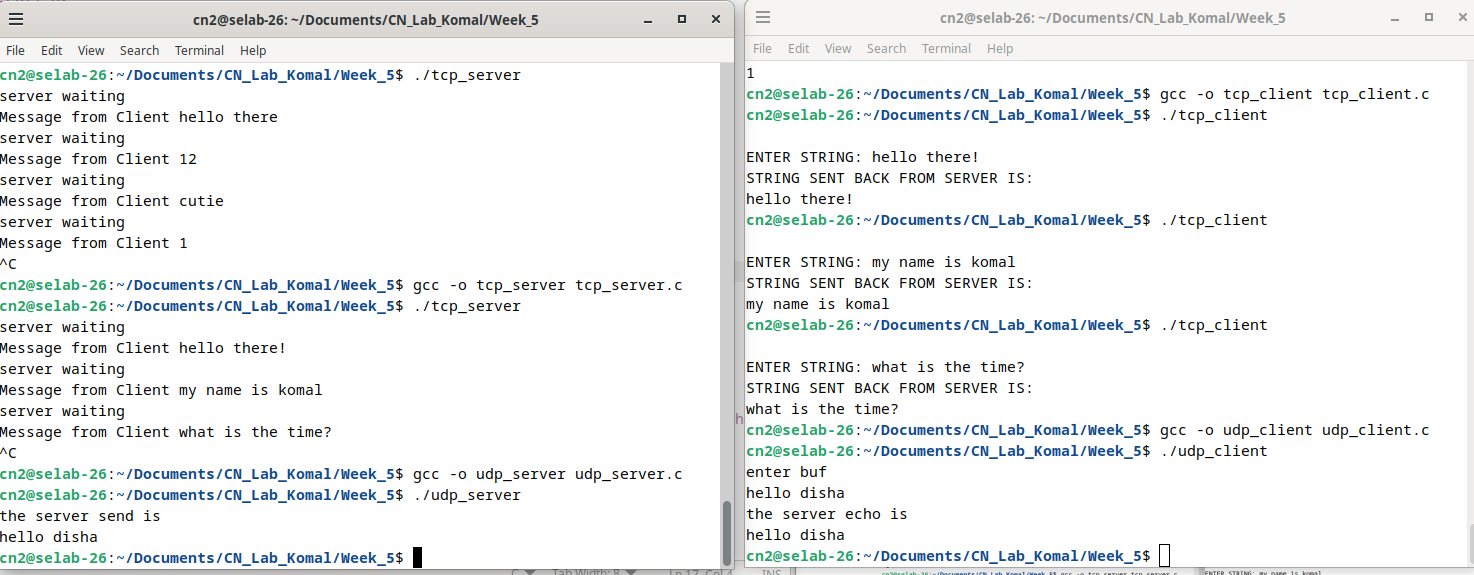
A screenshot of a computer screen

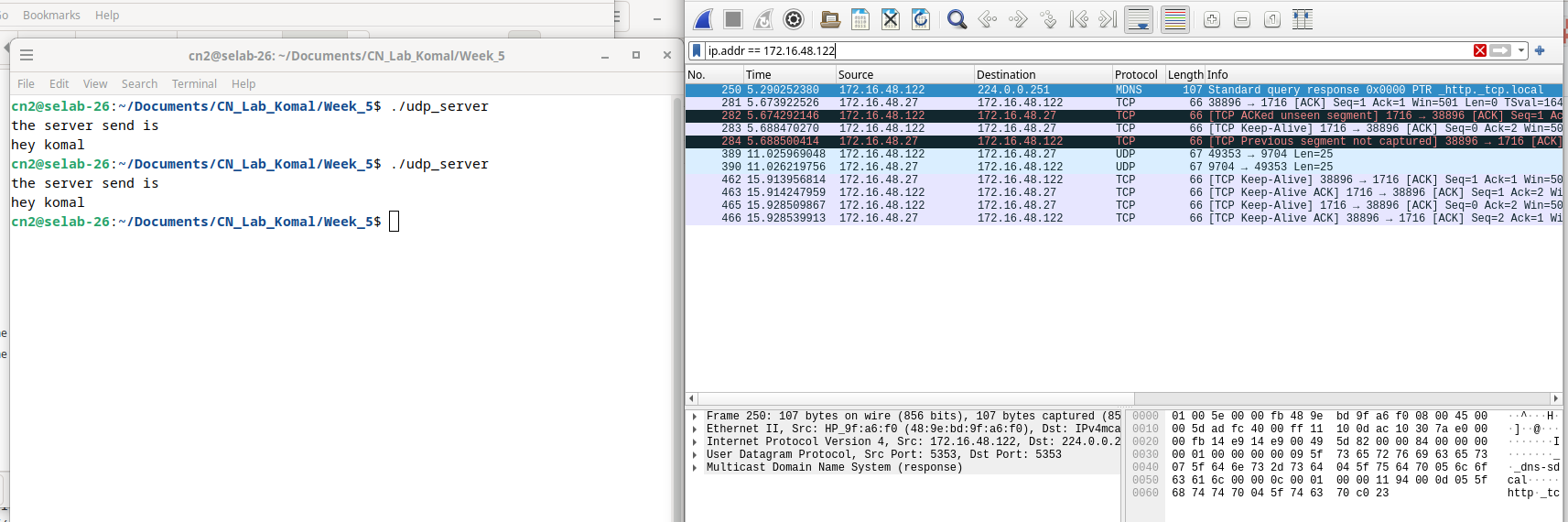
Description automatically generated

A screenshot of a computer

Description automatically generated

**Sample UDP Code**





**Lab Exercises:**

1. **Write an iterative TCP client-server program where the client accepts a sentence from the user and sends it to the server. The server will check for duplicate words in the string. Server will find number of occurrences of duplicate words present and remove the duplicate words by retaining single occurrence of the word and send the resultant sentence to the client. The client displays the received data on the client screen. The process repeats until the user enter the string “Stop”. Then both the processes terminate.**

**// TCP Server code:**

#include<stdio.h>

#include<string.h>

#include<unistd.h>

#include<fcntl.h>

#include<sys/stat.h>

#include <arpa/inet.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#define PORTNO 10208

#define BUFFER\_SIZE 1024

int word\_exists(char words[][BUFFER\_SIZE], int count, const char \*word) {

for (int i = 0; i < count; i++) {

if (strcmp(words[i], word) == 0) {

return 1;

}

}

return 0;

}

// Function to remove duplicates and form the result

void remove\_duplicates(const char \*input, char \*output) {

char words[100][BUFFER\_SIZE];

int word\_count = 0;

char \*token = strtok((char \*)input, " ");

// Extract words from the input sentence

while (token != NULL) {

if (!word\_exists(words, word\_count, token)) {

strcpy(words[word\_count++], token);

}

token = strtok(NULL, " ");

}

// Form the result without duplicates

output[0] = '\0';

for (int i = 0; i < word\_count; i++) {

strcat(output, words[i]);

strcat(output, " ");

}

// Remove trailing space

if (strlen(output) > 0) {

output[strlen(output) - 1] = '\0';

}

}

int main()

{

int sockfd,newsockfd,portno,clilen,n=1;

struct sockaddr\_in seraddr,cliaddr;

int i,value;

char buf[256], result[256];

// create an unnamed socket for the server

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

//Name the socket

seraddr.sin\_family = AF\_INET;

seraddr.sin\_addr.s\_addr = inet\_addr("172.16.48.27");// \*\*

seraddr.sin\_port = htons(PORTNO);

bind(sockfd,(struct sockaddr \*)&seraddr,sizeof(seraddr));

//Create a connection queue and wait for clients

listen(sockfd,5);

while (1) {

memset(buf, 0, sizeof(buf));

printf("server waiting");

//Accept a connection

clilen = sizeof(clilen);

newsockfd=accept(sockfd,(struct sockaddr \*)&cliaddr,&clilen);

//Read and write to client on client\_sockfd (Logic for problem mentioned here)

n = read(newsockfd,buf,sizeof(buf));

buf[n] = '\0';

if (strcmp(buf, "stop") == 0) break;

printf(" \nMessage from Client %s \n",buf);

remove\_duplicates(buf, result);

n = write(newsockfd,result,sizeof(buf));

}

}

**// TCP Client Code:**

#include <sys/types.h>

#include <sys/socket.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/stat.h>

#include <arpa/inet.h>

#include <stdio.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <string.h>

int main() {

int len, result, sockfd, n = 1;

struct sockaddr\_in address;

char ch[256], buf[256];

// Create a socket for the client

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

if (sockfd < 0) {

perror("Socket creation failed");

exit(1);

}

// Name the socket as agreed with the server

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = inet\_addr("172.16.48.27");

address.sin\_port = htons(10208);

len = sizeof(address);

// Connect your socket to the server’s socket

result = connect(sockfd, (struct sockaddr \*)&address, len);

if (result == -1) {

perror("CLIENT ERROR");

close(sockfd);

exit(1);

}

// You can now read and write via sockfd

while(1){

printf("\nENTER STRING: ");

scanf(" %[^\n]s", ch);

if (strcmp(ch, "stop") == 0) break;

write(sockfd, ch, strlen(ch));

printf("STRING SENT BACK FROM SERVER IS:\n");

n = read(sockfd, buf, sizeof(buf));

printf("%s\n", buf);

}

close(sockfd);

return 0;

}

1. **Write an iterative UDP client-server program where the client sends rows of a matrix, and the server combines them together as a matrix.**