University Practical EXAM – Computer Networks

Name – Shubham Pareek

RA1911003010946

Semester/Sec – 5th  / F2

AIM - **To establish connection from a client to server located in an unknown location using wireless device.**

Procedure

Server location is unknown and client wants to connect using wireless device so we can use Remote command using UDP as the relevant protocol communication.

Server

* Include the necessary header files and create socket function with AF\_INET.
* Initialize server address to 0 using the bzero function.
* Assign the sin\_family to AF\_INET, sin\_addr to INADDR\_ANY, sin\_port to dynamically assigned port number.
* Bind the local host and with in loop receive the command to be executed from the client.
* Append text and execute the command using system call.
* Send the result of execution to the Client using a file buffer.

Client

* Include the necessary header files and create socket function with AF\_INET.
* Initialize server address to 0 using the bzero function.
* Assign the sin\_family to AF\_INET and get the server IP address and port number from console.
* Using gethostbyname() assign it to a hostent structure and assign it to sin\_adde of server.
* Obtain the command to be executed in the server and send command to server.
* Received the output from the server and print it on the console.

Program

UDPserver.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define PORT 8080

#define MAXLINE 1024

// Driver code

int main() {

int sockfd;

char buffer[MAXLINE];

char \*hello = "Hello from server";

struct sockaddr\_in servaddr, cliaddr;

// Creating socket file descriptor

if ( (sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0 ) {

perror("socket creation failed");

exit(EXIT\_FAILURE);

}

memset(&servaddr, 0, sizeof(servaddr));

memset(&cliaddr, 0, sizeof(cliaddr));

// Filling server information

servaddr.sin\_family = AF\_INET; // IPv4

servaddr.sin\_addr.s\_addr = INADDR\_ANY;

servaddr.sin\_port = htons(PORT);

// Bind the socket with the server address

if ( bind(sockfd, (const struct sockaddr \*)&servaddr,

sizeof(servaddr)) < 0 )

{

perror("bind failed");

exit(EXIT\_FAILURE);

}

int len, n;

len = sizeof(cliaddr); //len is value/resuslt

n = recvfrom(sockfd, (char \*)buffer, MAXLINE,

MSG\_WAITALL, ( struct sockaddr \*) &cliaddr,

&len);

buffer[n] = '\0';

printf("Client : %s\n", buffer);

sendto(sockfd, (const char \*)hello, strlen(hello),

MSG\_CONFIRM, (const struct sockaddr \*) &cliaddr,

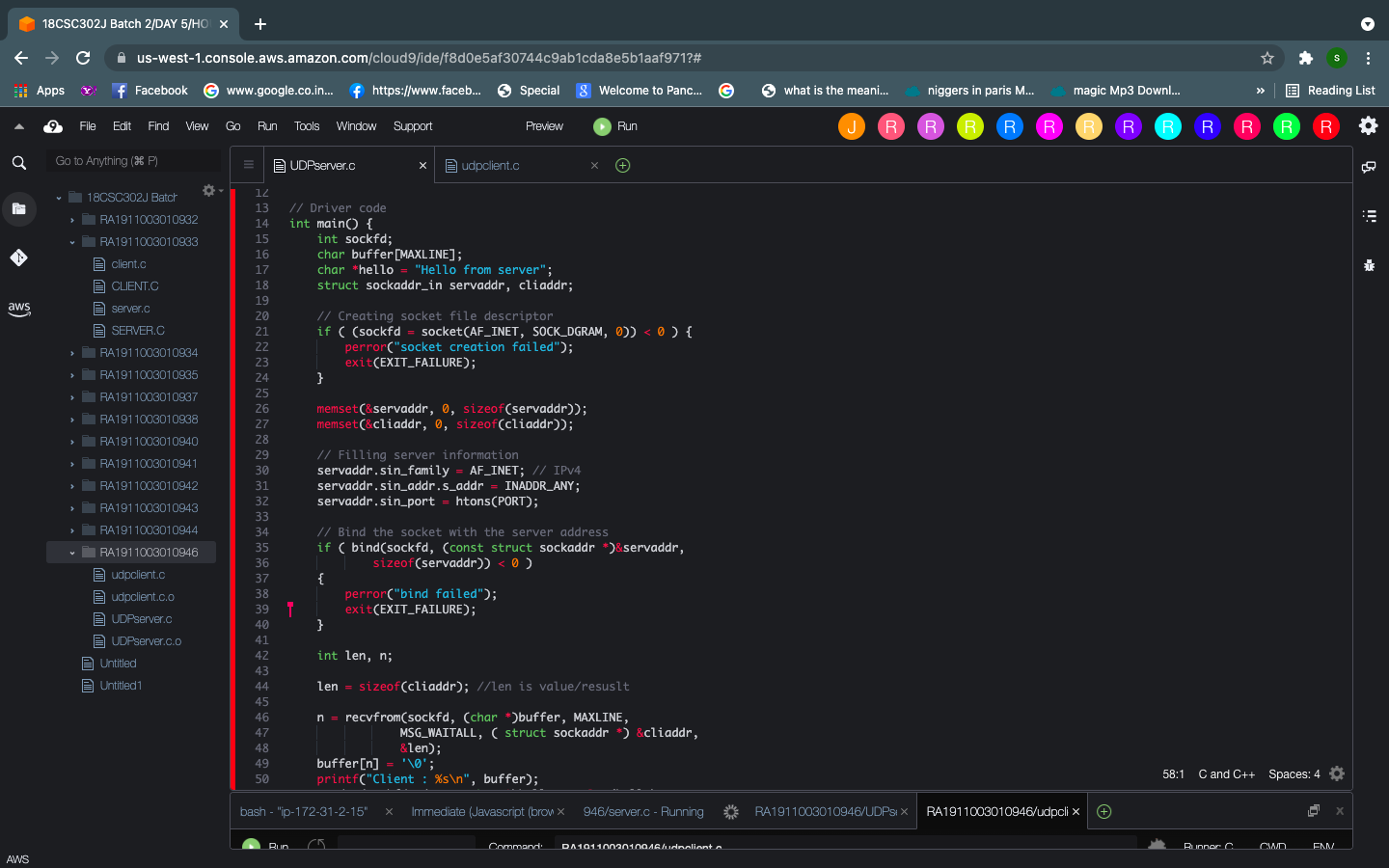
len);

printf("Hello message sent.\n");

return 0;

}

Screenshot



UDPclient.c

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <netinet/in.h>

#define PORT 8080

#define MAXLINE 1024

// Driver code

int main() {

int sockfd;

char buffer[MAXLINE];

char \*hello = "Hello from client";

struct sockaddr\_in servaddr;

// Creating socket file descriptor

if ( (sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0 ) {

perror("socket creation failed");

exit(EXIT\_FAILURE);

}

memset(&servaddr, 0, sizeof(servaddr));

// Filling server information

servaddr.sin\_family = AF\_INET;

servaddr.sin\_port = htons(PORT);

servaddr.sin\_addr.s\_addr = INADDR\_ANY;

int n, len;

sendto(sockfd, (const char \*)hello, strlen(hello),

MSG\_CONFIRM, (const struct sockaddr \*) &servaddr,

sizeof(servaddr));

printf("Hello message sent.\n");

n = recvfrom(sockfd, (char \*)buffer, MAXLINE,

MSG\_WAITALL, (struct sockaddr \*) &servaddr,

&len);

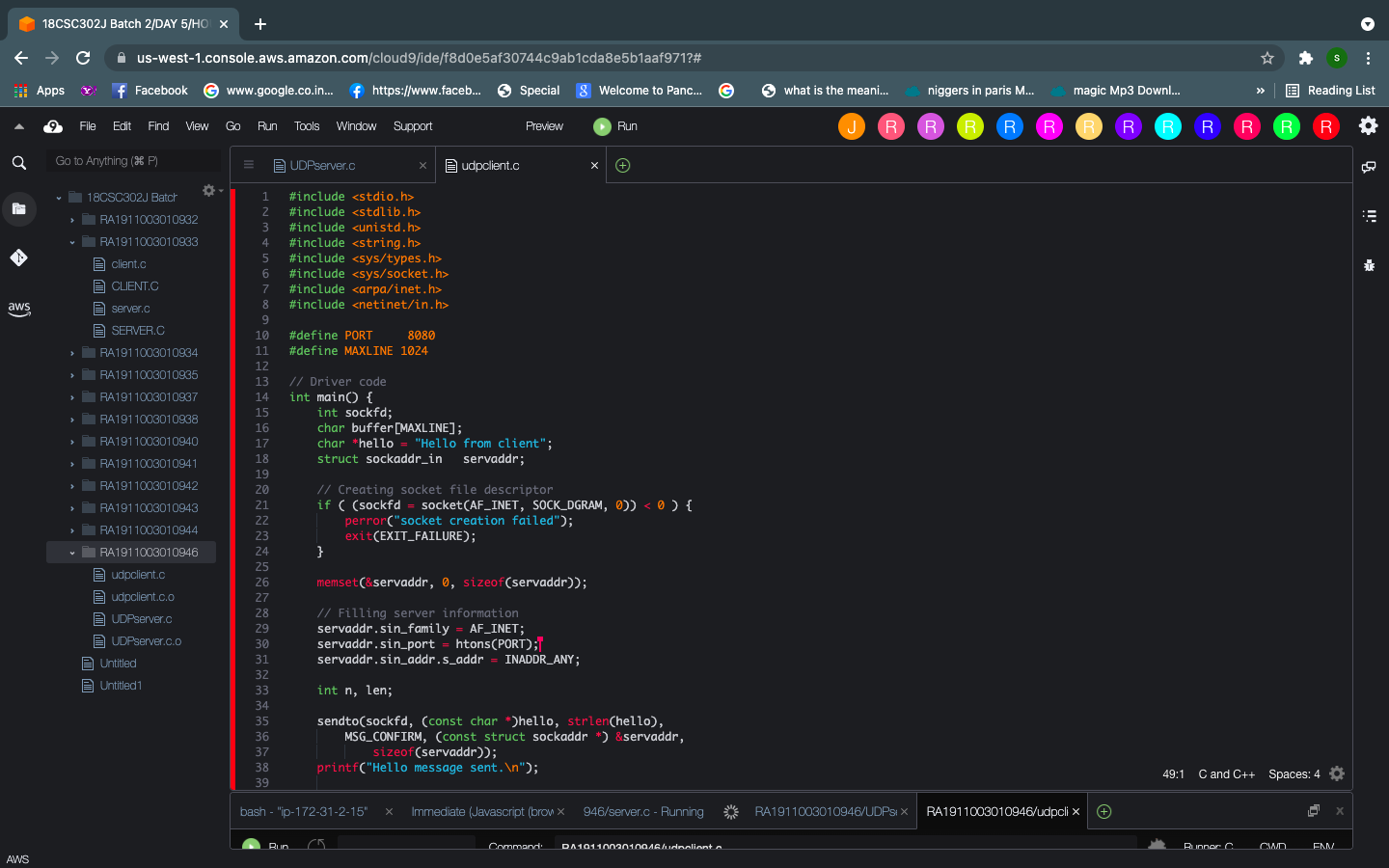
buffer[n] = '\0';

printf("Server : %s\n", buffer);

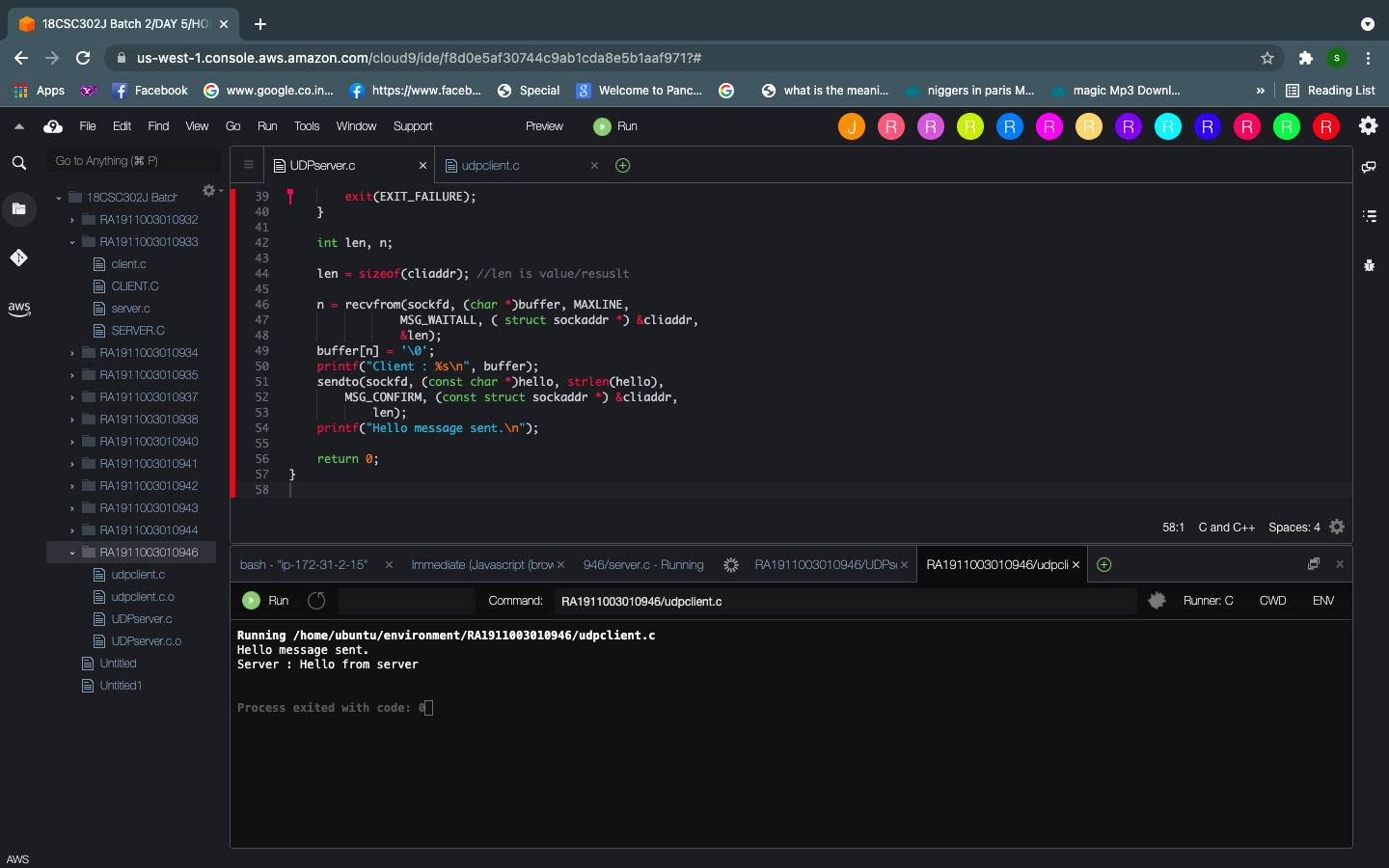
close(sockfd);

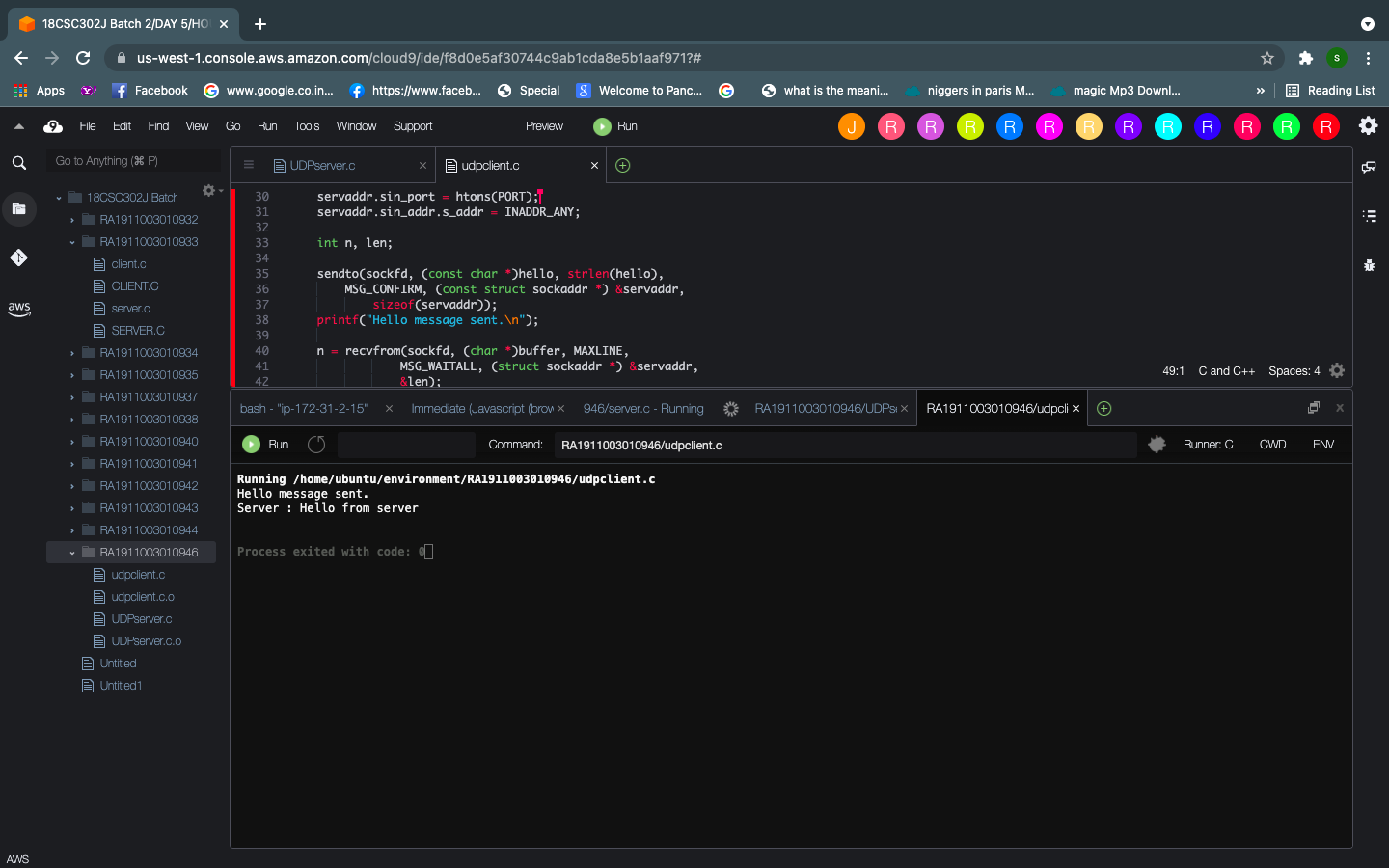
return 0;

}



Output –





Result – Desired output was presented in AWS terminal