**Assignment 1 - UDP-x-Time-Function**

**Computer Network**

**Program: MScIT Sem-2**

**Group ID : 28**

**Student Name Student ID**

Dev Adnani 202212012

Saif Saiyed 202212083

**202212012**

3.3 -2 UDP

Server.c

// Server side implementation of UDP client-server model

#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/result

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,

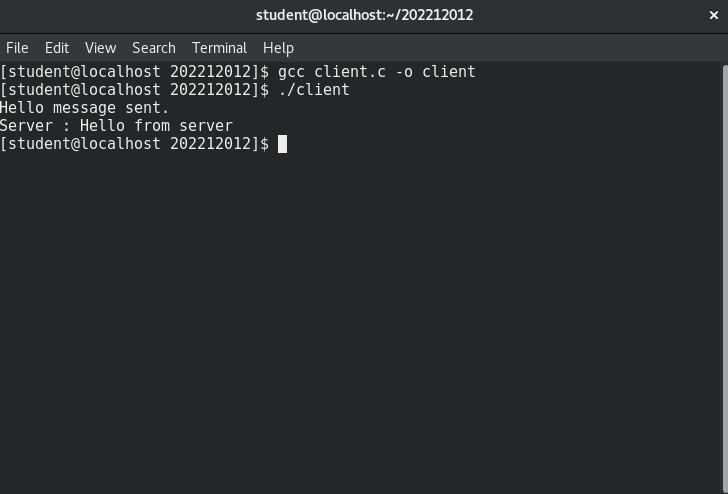
len);

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

return 0;

}

Screenshots :



Client.c

// Client side implementation of UDP client-server model

#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';

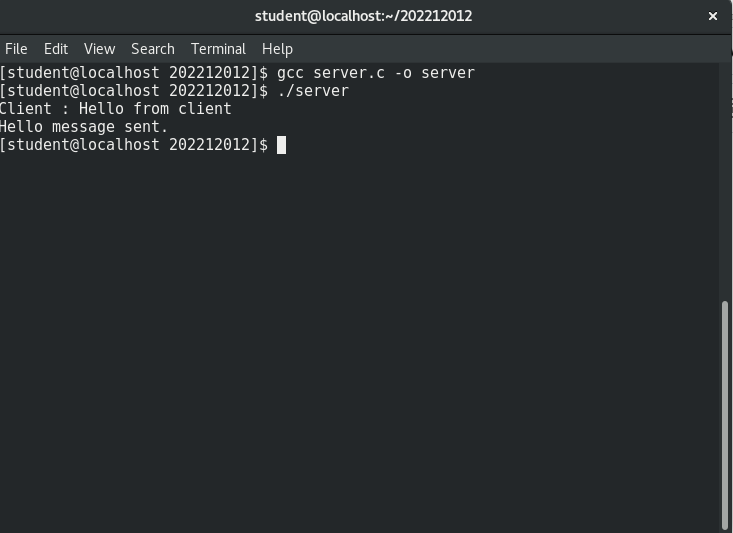
printf("Server : %s\n", buffer);

close(sockfd);

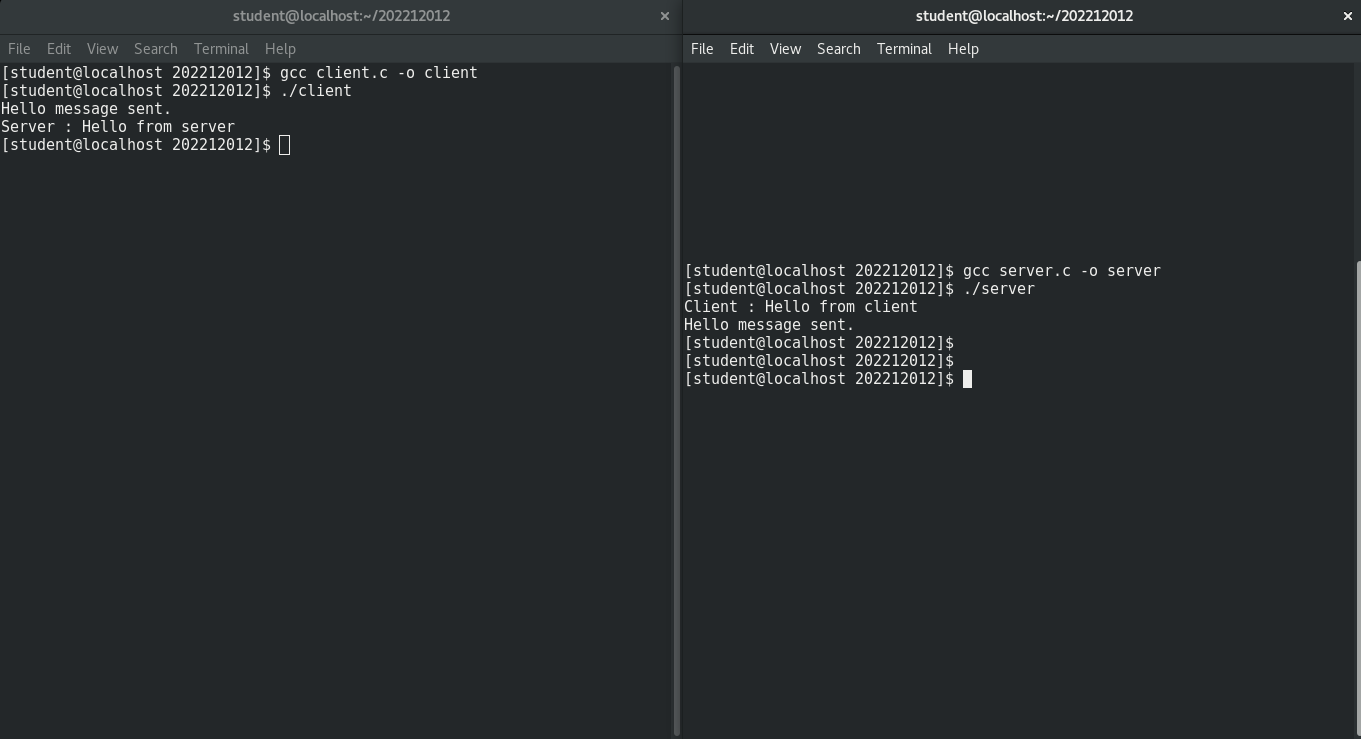
return 0;

}

Screenshots :



Mix-Screenshots :

****

Time Function

Client.c

#include <arpa/inet.h>

#include <stdio.h>

#include <string.h>

#include <sys/socket.h>

#include <unistd.h>

#define PORT 80012

int main(int argc, char const\* argv[])

{

int sock = 0, valread, client\_fd;

struct sockaddr\_in serv\_addr;

char\* hello = "Hello from client";

char buffer[1024] = { 0 };

if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

printf("\n Socket creation error \n");

return -1;

}

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

// Convert IPv4 and IPv6 addresses from text to binary

// form

if (inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr)

<= 0) {

printf(

"\nInvalid address/ Address not supported \n");

return -1;

}

if ((client\_fd

= connect(sock, (struct sockaddr\*)&serv\_addr,

sizeof(serv\_addr)))

< 0) {

printf("\nConnection Failed \n");

return -1;

}

send(sock, hello, strlen(hello), 0);

printf("Hello message sent\n");

valread = read(sock, buffer, 1024);

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

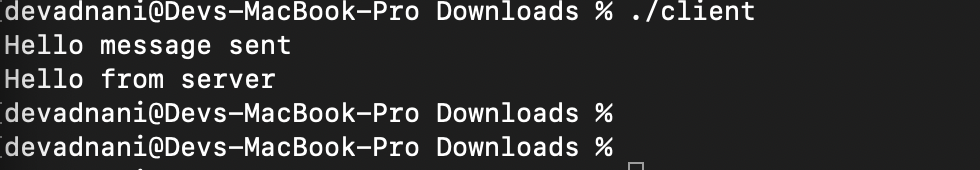
// closing the connected socket

close(client\_fd);

return 0;

}

Client Screenshot



Server.c

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <unistd.h>

#include <time.h>

#define PORT 80012

int main(int argc, char const \*argv[])

{

double sum = 0;

double add = 1;

int server\_fd, new\_socket, valread;

struct sockaddr\_in address;

int opt = 1;

int addrlen = sizeof(address);

char buffer[1024] = {0};

char \*hello = "Hello from server";

time\_t begin, end;

time(&begin);

int iterations = 1000 \* 1000 \* 1000;

for (int i = 0; i < iterations; i++)

{

sum += add;

add /= 2.0;

}

// Creating socket file descriptor

if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

{

perror("socket failed");

exit(EXIT\_FAILURE);

}

// Forcefully attaching socket to the port 8080

if (setsockopt(server\_fd, SOL\_SOCKET, SO\_REUSEADDR, &opt, sizeof(opt)))

{

perror("setsockopt");

exit(EXIT\_FAILURE);

}

address.sin\_family = AF\_INET;

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

address.sin\_port = htons(PORT);

// Forcefully attaching socket to the port 8080

if (bind(server\_fd, (struct sockaddr \*)&address,

sizeof(address)) < 0)

{

perror("bind failed");

exit(EXIT\_FAILURE);

}

if (listen(server\_fd, 3) < 0)

{

perror("listen");

exit(EXIT\_FAILURE);

}

if ((new\_socket = accept(server\_fd, (struct sockaddr \*)&address,

(socklen\_t \*)&addrlen)) < 0)

{

perror("accept");

exit(EXIT\_FAILURE);

}

valread = read(new\_socket, buffer, 1024);

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

send(new\_socket, hello, strlen(hello), 0);

printf("Hello message sent\n");

// closing the connected socket

close(new\_socket);

time(&end);

time\_t elapsed = end - begin;

printf("Time measured: %ld seconds. \n", elapsed);

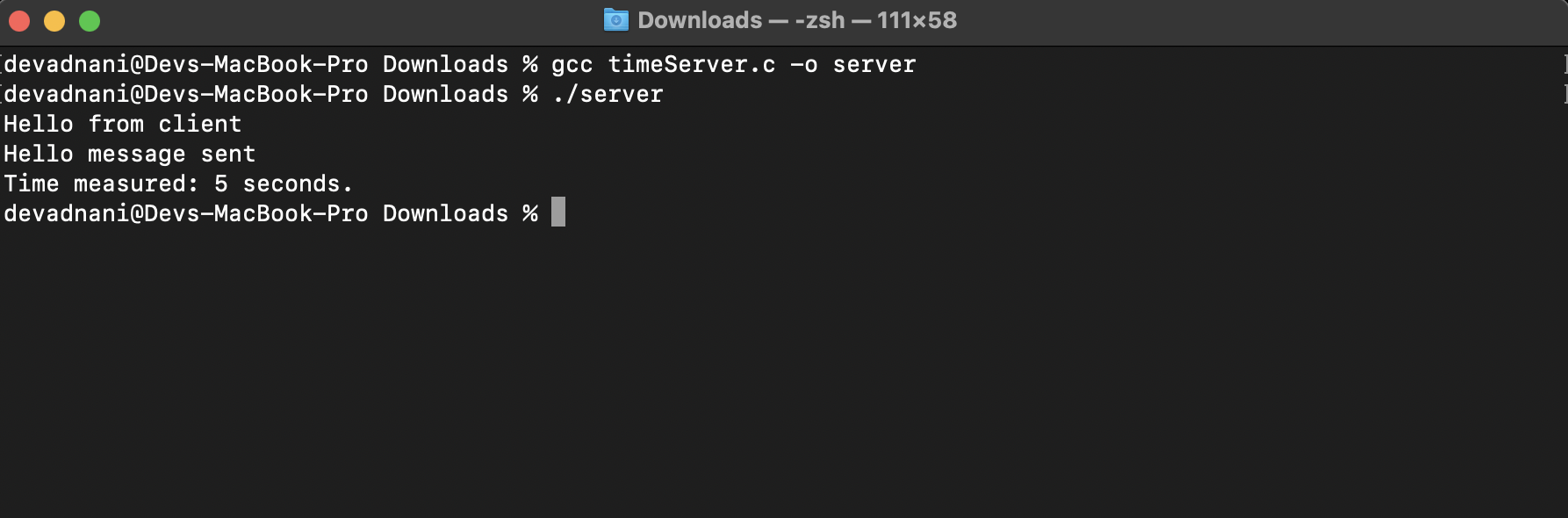
// closing the listening socket

shutdown(server\_fd, SHUT\_RDWR);

return 0;

}

Screenshot :



**202212083**

3.3 Exercise : 2 UDP

Create UDP server and client using socket programming. Make them communicate with each other by sending packets between them.

Server Code (server.c)

// Server side implementation of UDP client-server model

#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/result

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,

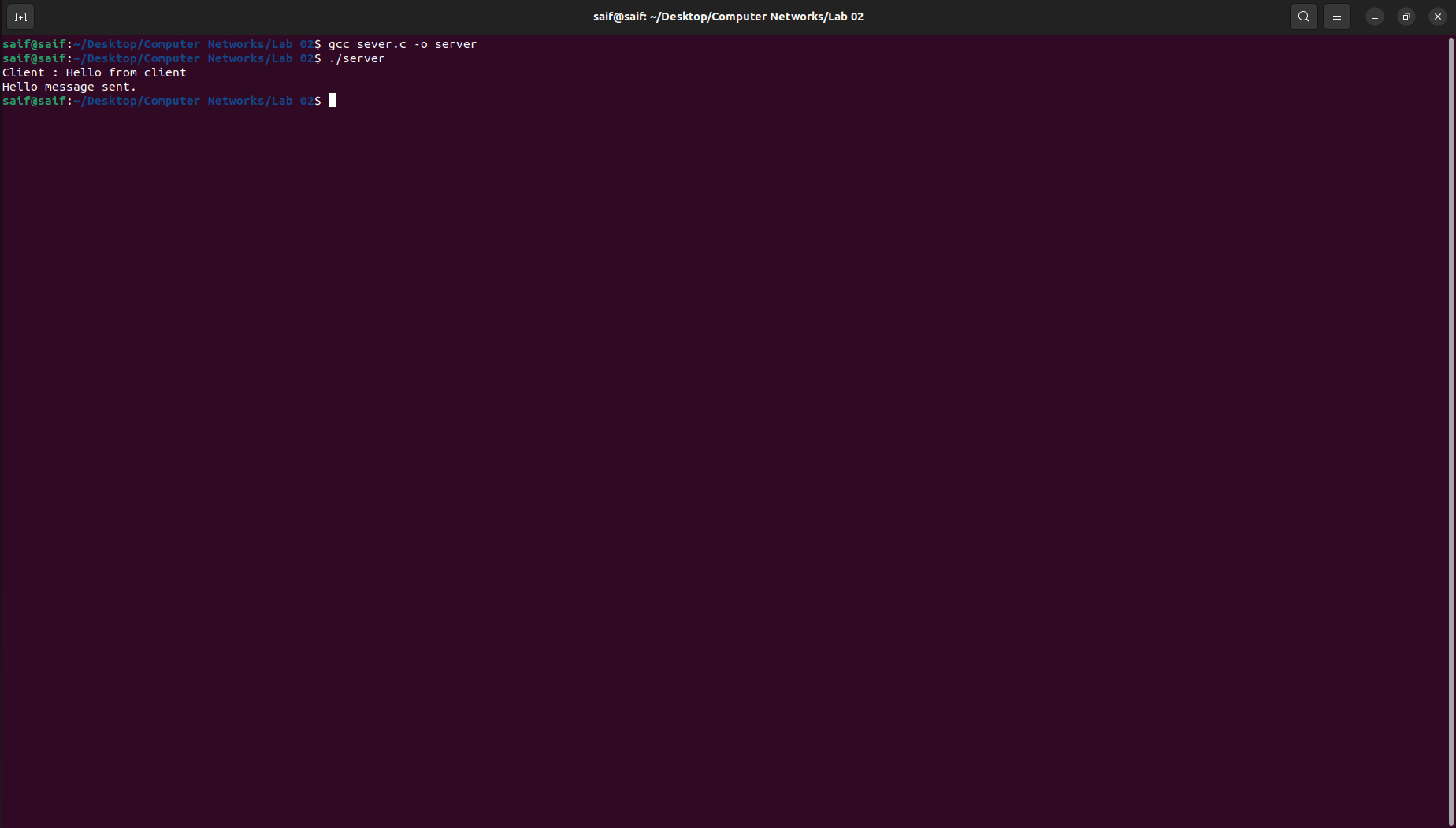
len);

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

return 0;

}

Output:



Client Code (client.c)

// Client side implementation of UDP client-server model

#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';

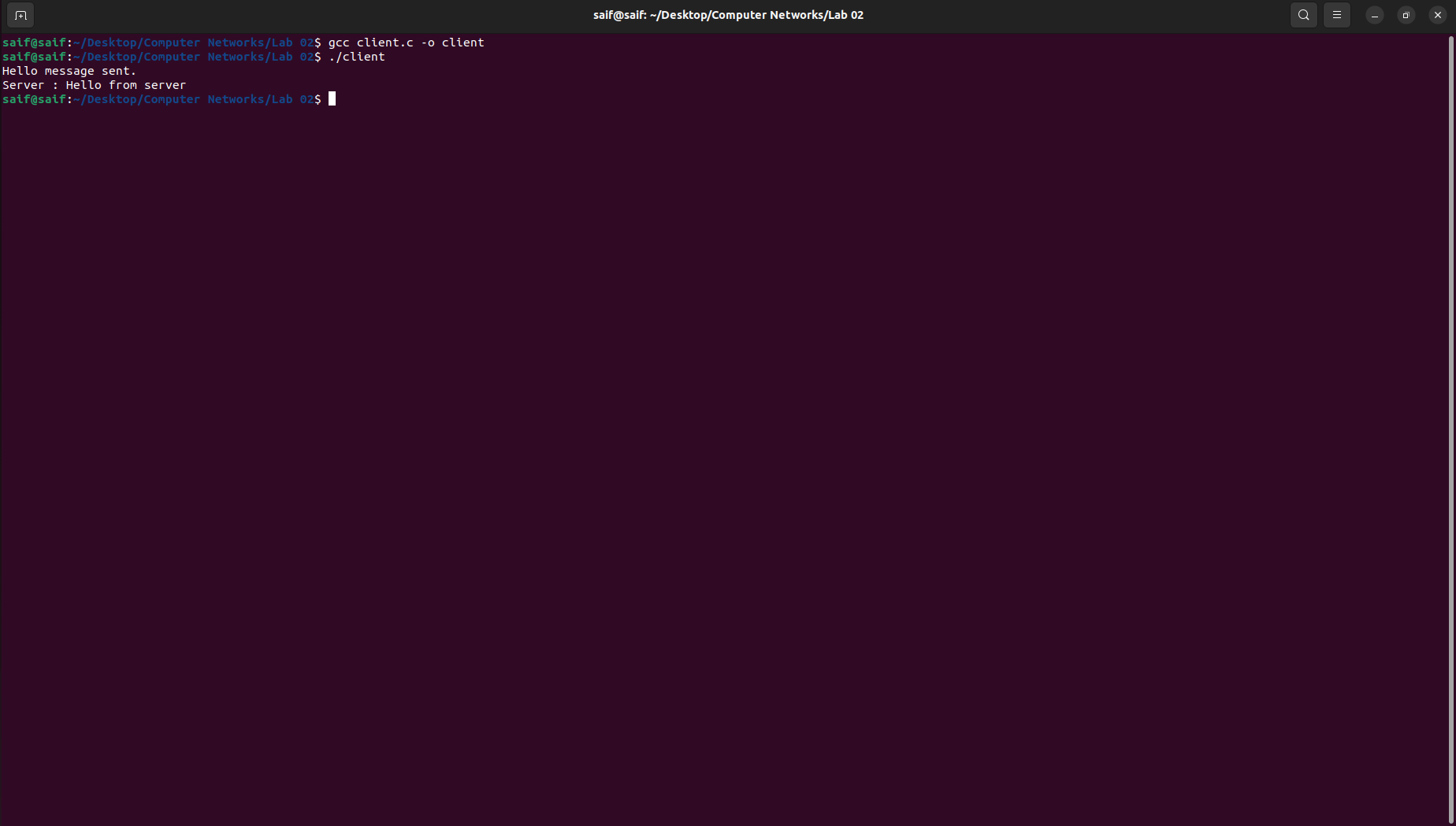
printf("Server : %s\n", buffer);

close(sockfd);

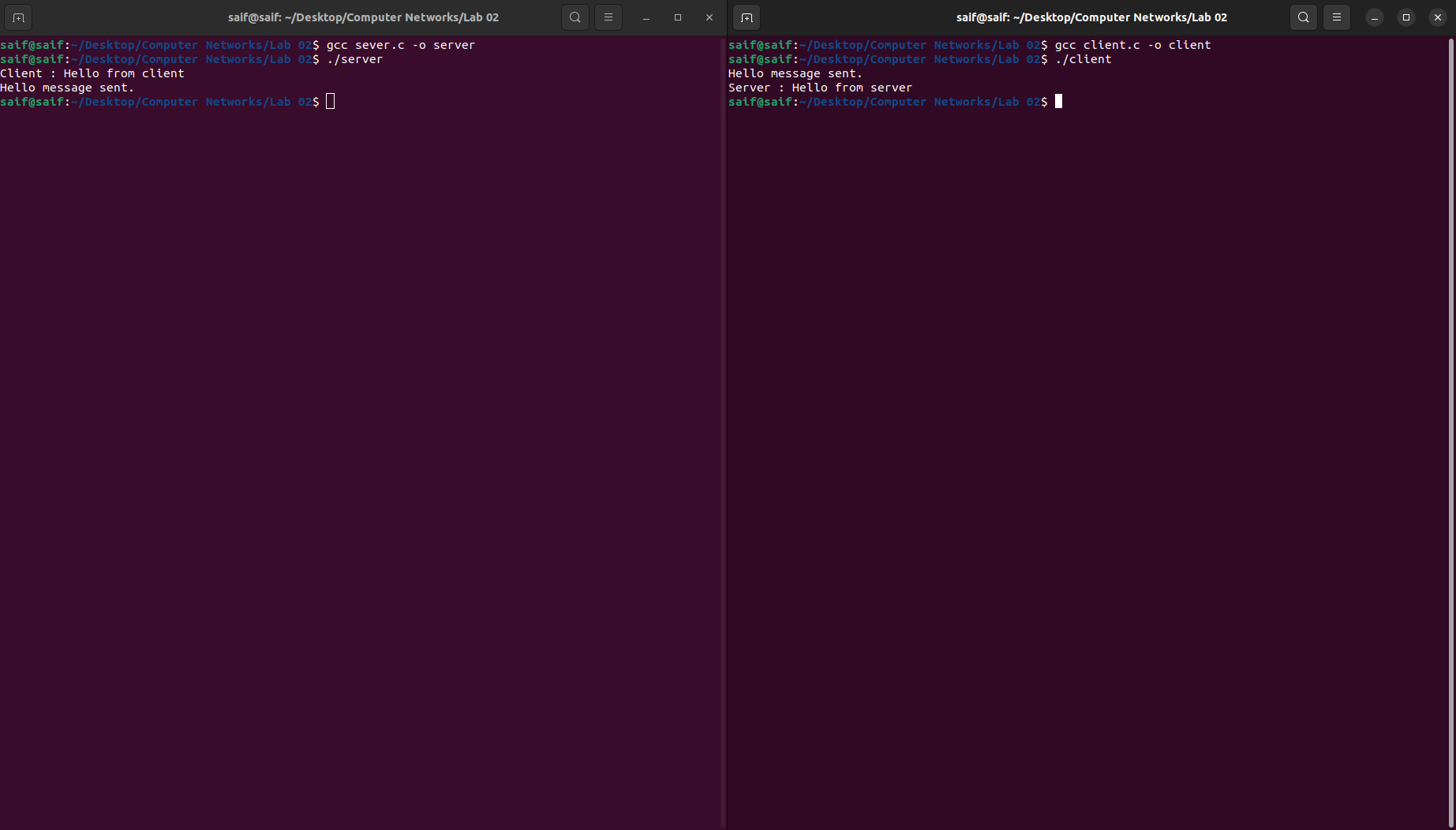
return 0;

}

Output:



Client & Server:



Measure Execution Time - Time Function:

Server Code (TimeServer.c)

#include <netinet/in.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <unistd.h>

#include <time.h>

#define PORT 8163

int main(int argc, char const\* argv[])

{

double sum=0;

double add=1;

int server\_fd, new\_socket, valread;

struct sockaddr\_in address;

int opt = 1;

int addrlen = sizeof(address);

char buffer[1024] = { 0 };

char\* hello = "Hello from server";

time\_t begin ,end;

time(&begin);

int iterations=1000\*1000\*1000;

for(int i=0;i<iterations;i++)

{

sum+=add;

add/=2.0;

}

// Creating socket file descriptor

if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

perror("socket failed");

exit(EXIT\_FAILURE);

}

// Forcefully attaching socket to the port 8080

if (setsockopt(server\_fd, SOL\_SOCKET, SO\_REUSEADDR | SO\_REUSEPORT, &opt, sizeof(opt))) {

perror("setsockopt");

exit(EXIT\_FAILURE);

}

address.sin\_family = AF\_INET;

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

address.sin\_port = htons(PORT);

// Forcefully attaching socket to the port 8080

if (bind(server\_fd, (struct sockaddr\*)&address, sizeof(address))

< 0) {

perror("bind failed");

exit(EXIT\_FAILURE);

}

if (listen(server\_fd, 3) < 0) {

perror("listen");

exit(EXIT\_FAILURE);

}

if ((new\_socket = accept(server\_fd, (struct sockaddr\*)&address, (socklen\_t\*)&addrlen))

< 0) {

perror("accept");

exit(EXIT\_FAILURE);

}

valread = read(new\_socket, buffer, 1024);

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

send(new\_socket, hello, strlen(hello), 0);

printf("Hello message sent\n");

// closing the connected socket

close(new\_socket);

time(&end);

time\_t elapsed=end-begin;

printf("Time measured: %ld seconds. \n",elapsed);

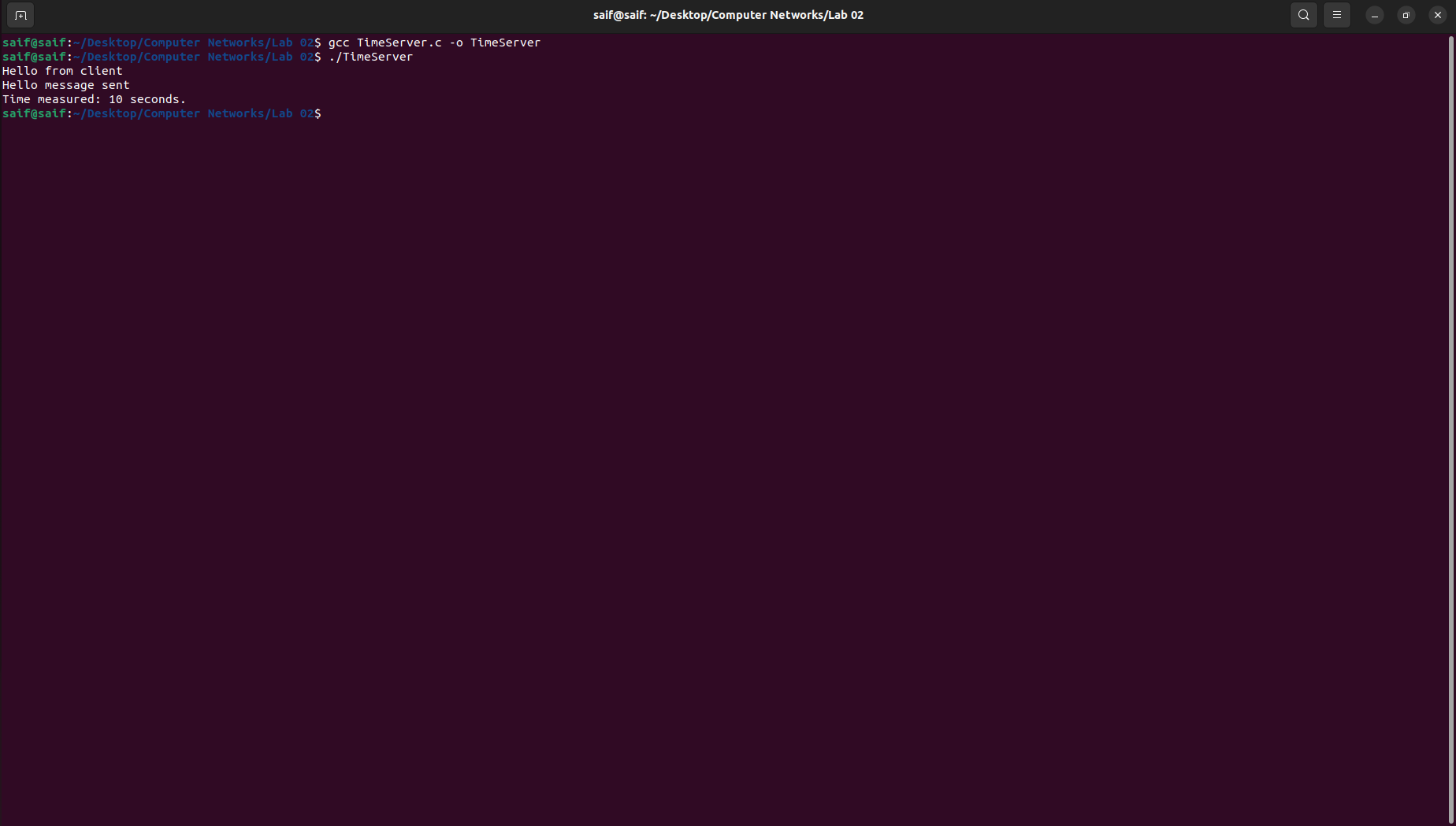
// closing the listening socket

shutdown(server\_fd, SHUT\_RDWR);

return 0;

}

Output:



Client Code (TimeClient.c)

#include <arpa/inet.h>

#include <stdio.h>

#include <string.h>

#include <sys/socket.h>

#include <unistd.h>

#define PORT 8163

int main(int argc, char const\* argv[])

{

int sock = 0, valread, client\_fd;

struct sockaddr\_in serv\_addr;

char\* hello = "Hello from client";

char buffer[1024] = { 0 };

if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0) {

printf("\n Socket creation error \n");

return -1;

}

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

// Convert IPv4 and IPv6 addresses from text to binary

// form

if (inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr)

<= 0) {

printf(

"\nInvalid address/ Address not supported \n");

return -1;

}

if ((client\_fd

= connect(sock, (struct sockaddr\*)&serv\_addr,

sizeof(serv\_addr)))

< 0) {

printf("\nConnection Failed \n");

return -1;

}

send(sock, hello, strlen(hello), 0);

printf("Hello message sent\n");

valread = read(sock, buffer, 1024);

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

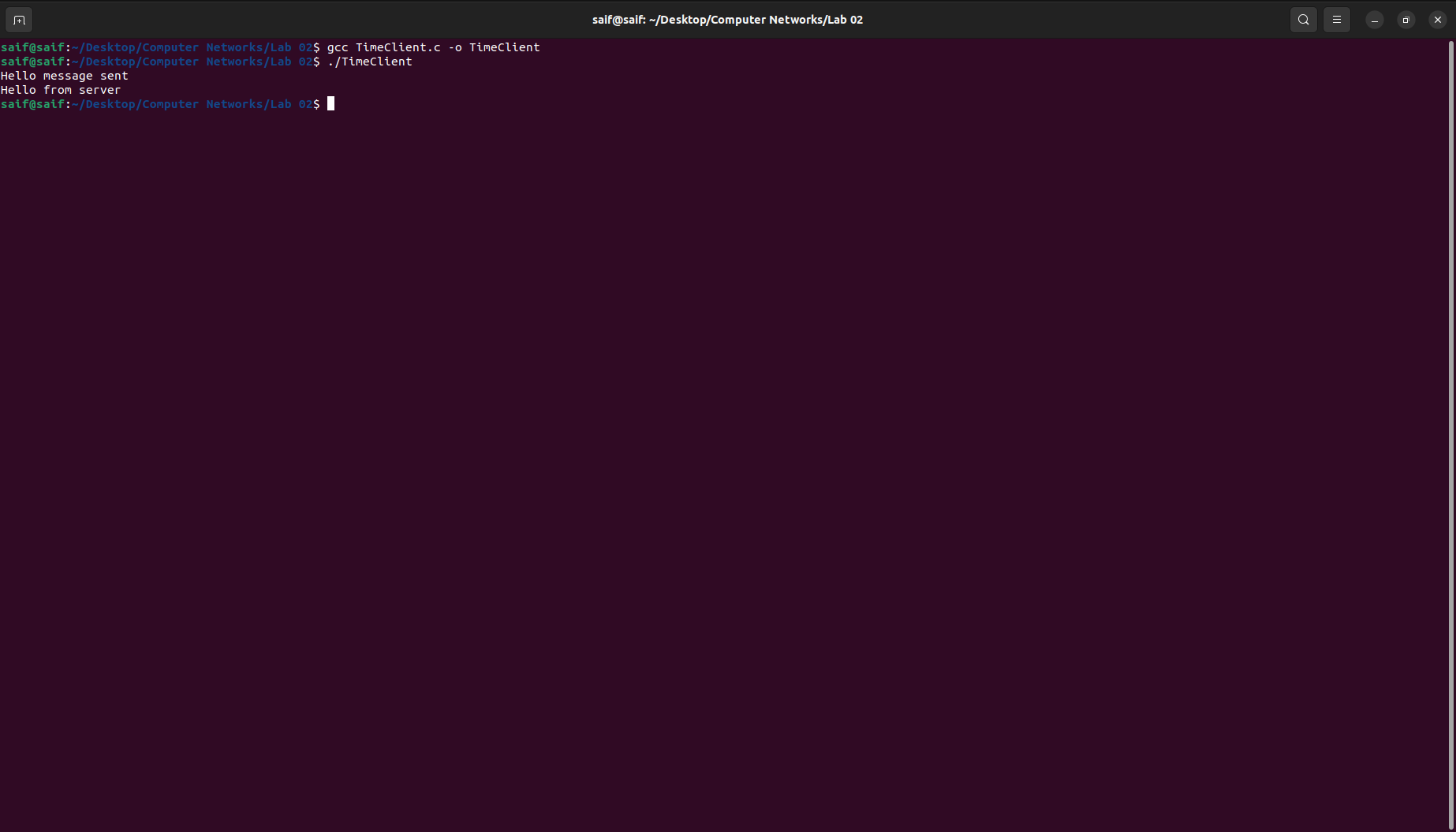
// closing the connected socket

close(client\_fd);

return 0;

}

Output:



Time Function - Client & Server:

