Report

Computer Networks -1

Assignment-1

Akhil George Thomas -- CS17BTECH11047

Q1

The following two features were added to the programs:

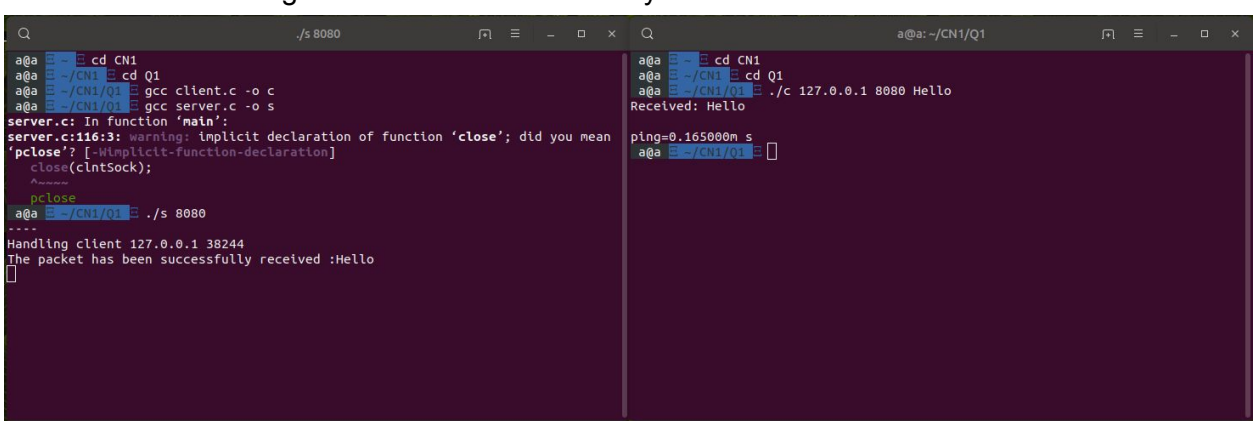
1. Ping

Calculates the ping (the delay in milli sec) in the communication. This is an important metric to test the reachability of a node in a network.

2. Parity

Implemented a parity character, like a parity bit, similar to a checksum available in portocols like HTTPv4.

It checks if the message were transmitted correctly.



Q2

Easy

The socket is kept alive till the termination of either the client or the server.

Client:

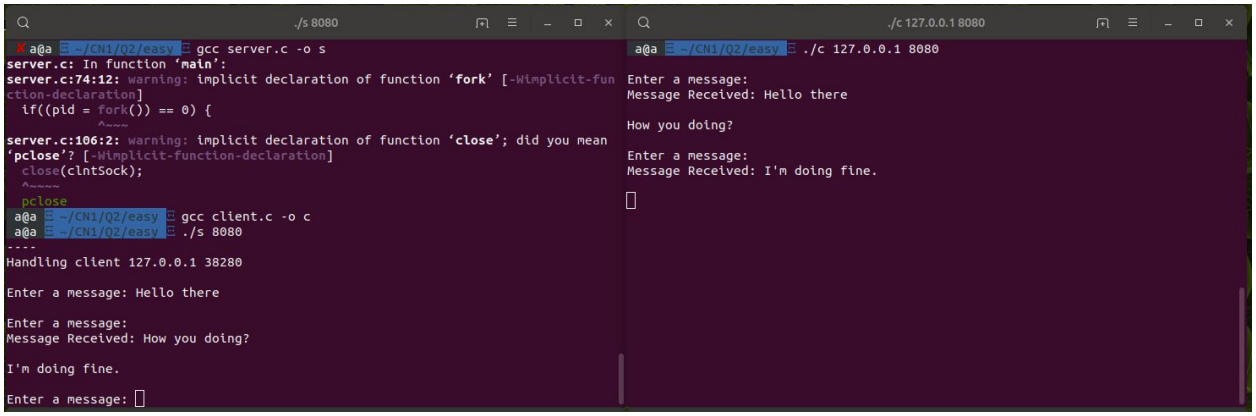
2 threads are created: one to send messages to the server,while the other is used to receive any messages from the server.

The threads were created using simple fork().

Server:

2 threads are created: one to send messages to the client,while the other is used to receive any messages from the client.

The threads were created using simple fork().



Normal

A message from one client is forwarded to the other client.
Only two clients are allowed to connect to the server.

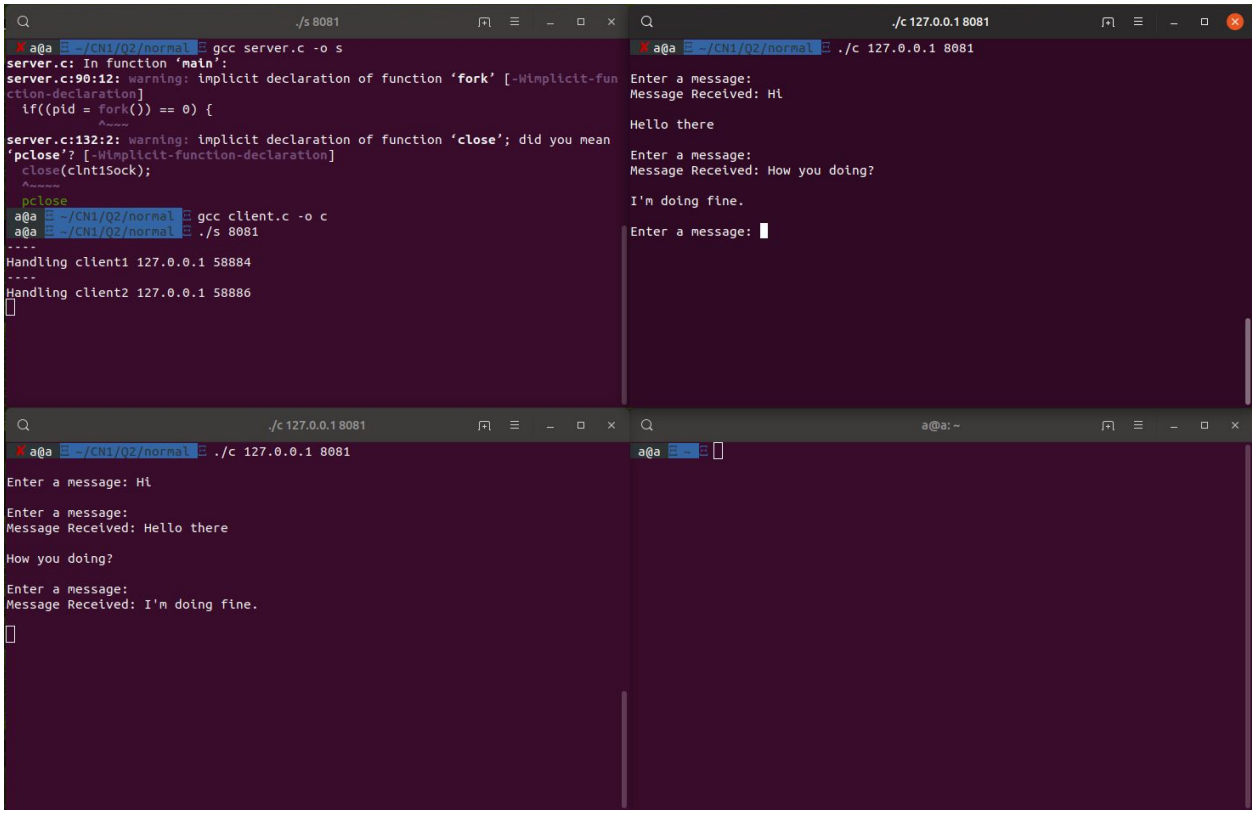
Client:

2 threads are created: one to send messages to the server,while the other is used to receive any messages from the server.
The threads were created using simple fork().

Server:

2 threads are created: one to handle messages from client1,while the other is used with client2.
The threads were created using simple fork().

The message is forwarded using send(). This is triggered when a message is received.



Hard

A clientNum is used to identify each client uniquely, an unsigned integer from 0 to N.

Client:

2 threads are created: one to send messages to the server,while the other is used to receive any messages from the server.

The threads were created using simple fork().

The destination's clientNum is encoded into the last character of the message to the server.

Server:

A maximum of N+1 threads run at a time. N threads handle recv() from each client while one thread handles any new client requests.

A new POSIX thread is created when a new client request is made.

The last character of the received message are truncated, and the clientNum is found.

```
a@a: ~/CN1
a@a ~/CN1 gcc server.c -o s -lpthread
server.c: In function 'main':
server.c:127:3: warning: implicit declaration of function 'close'; did you mean
'pclose'? [-Wimplicit-function-declaration]
   close(clntSock[i]);
   ^~~~~
   pclose
a@a ~/CN1 gcc client.c -o c -lpthread
a@a ~/CN1
```

```
./s 9063
a@a ~/CN1 ./s 9063
----
Handling client0 127.0.0.1 40628
0
----
Handling client1 127.0.0.1 40630
1

to server: Hello?!

to server: HI
```

```
./c 127.0.0.1 9063
a@a ~/CN1 ./c 127.0.0.1 9063

Enter client Num: 1

Enter a message: Hello?!

Enter client Num:
Message Received: HI
```

```
./c 127.0.0.1 9063
a@a ~/CN1 ./c 127.0.0.1 9063

Enter client Num:
Message Received: Hello?!
0

Enter a message: HI

Enter client Num:
```

```
./s 8080
^C
a@a ~/CN1/Q2/hard gcc client.c -o c
a@a ~/CN1/Q2/hard gcc server.c -o s -lpthread
server.c: In function 'main':
server.c:127:3: warning: implicit declaration of function 'close'; did you mean
'pclose'? [-Wimplicit-function-declaration]
   close(clntSock[i]);
   ^~~~~
   pclose
a@a ~/CN1/Q2/hard ./s 8082
bind() failed: Address already in use
a@a ~/CN1/Q2/hard ./s 8080
----
Handling client0 127.0.0.1 38388
0
----
Handling client1 127.0.0.1 38390
1


Hi.
```

```
./c 127.0.0.1 8080
a@a ~/CN1/Q2/hard ./c 127.0.0.1 8080

Enter client Num: 1

Enter a message:
Enter client Num: Hi.

Enter a message:
Enter client Num:
Message Received:


Message Received: When will client2 join?

1

Enter a message:
Enter client Num: Look client 2 just joined

Enter a message:
Enter client Num: 2

Enter a message:
Enter client Num: Hello
```

```
./c 127.0.0.1 8080
a@a ~/CN1/Q2/hard ./c 127.0.0.1 8080

Enter client Num:
Message Received:


Message Received: Hi.

0

Enter a message:
Enter client Num: When will client2 join?

Enter a message:
Enter client Num:
Message Received:

.


Message Received: Look client 2 just joined
```

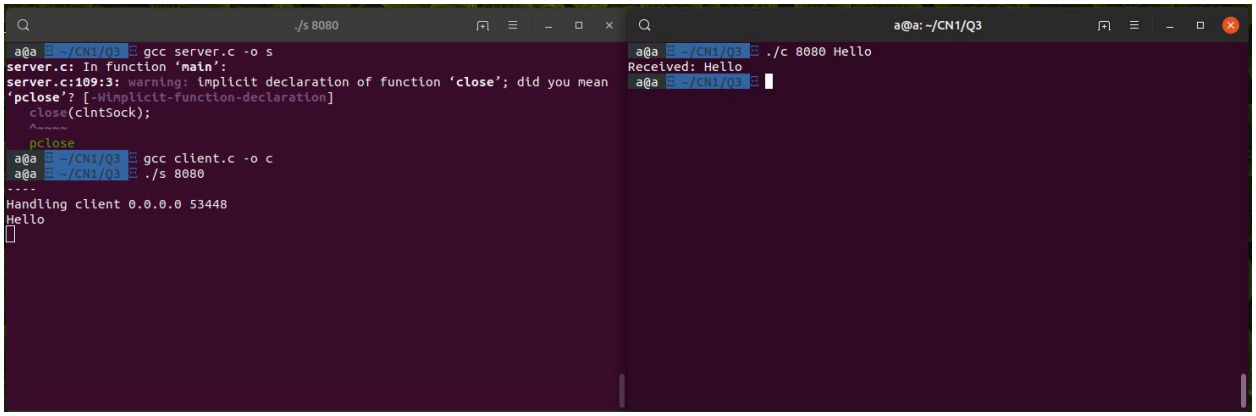
```
./c 127.0.0.1 8080
a@a ~/CN1/Q2/hard ./c 127.0.0.1 8080

Enter client Num:
Message Received:


Message Received: Hello
```

Q3

The socket is opened using IPv6 TCP protocol. AF_INET6 is used as argument while creating the socket. The data structure used to store the address is sockaddr_in6 instead of sockaddr_in. This supports additional data members like sin6_flowinfo and sin6_scope_id.

Client:
All the rules to start a IPv6 connection socket are similar to that of a IPv4 socket.

Server:
All the rules to start a IPv6 connection socket are similar to that of a IPv4 socket.



References:

http://man7.org/linux/man-pages/man7/ip.7.html
http://man7.org/linux/man-pages/man7/ipv6.7.html
https://www.cs.cmu.edu/afs/cs/academic/class/15213-f16/www/lectures/21-netprog1.pptx
https://www.cs.cmu.edu/~dga/15-441/S08/lectures/03-socket.ppt
https://stackoverflow.com/questions/5956516/getaddrinfo-and-ipv6
https://gist.github.com/inaz2/0e77c276a834ad8e3131