FINAL PROJECT

Gal Koaz

01/06/2022

—

Communication and computing

—

CONTENTS

UDP Programming

1. Problem 1: Learn to use datagram sockets by example.
2. Problem 2: Create a gateway process that simulates datagram loss.

TCP programming

1. Part A: IP addresses, hostnames and HTTP
2. Part B: A simple web client

PROBLEM 1: Learn to use Datagram sockets

Learn to use Datagram sockets by Examples

After understanding and deepening the two files (send_udp.c, recv_udp.c), I did compile them and execute them both in my local host.
In order to run them I followed instructions written in the assignment first recv_udp and in another terminal send_udp.

Now, I went back to the code to comment for each line in the program the exact and brief explanation for it, We were then asked to write a function called printsin() according to the specifications below, uncomment the calls to printsin in recv_udp.c, and build the executable.

Documentation

Documentation from the terminal for the function “printsin” I wrote:
Terminal 1:

```
gal@gal-VirtualBox:/$ '/home/gal/CLionProjects/untitled/send2' 127.0.0.1
```

Terminal 2:

```
gal@gal-VirtualBox:~$ '/home/gal/CLionProjects/untitled/recv2'
RECV_UDP
Local socket is:: ip= 0.0.0.0, port= 13107
recv_udp:
Packet from:: ip= 127.0.0.1, port= 65233
Got data ::<22490>
```

PROBLEM 2: Create a gateway process that simulates datagram loss

Create a gateway process that simulates datagram loss

In this section, we were asked to write three programs whose main purpose is to create a gateway process that simulates data loss.

In the first part, I created a program named "source.c" which gets the hostname from the command-line, After I created a datagram socket to the host with the unique port "3333"
when the datagram socket carrying in its body integer number started from zero and increments each iteration of the infinite loop, then sleeps for one second.

In the second part, I created a program named “gateway.c” which takes the hostname from the command-line, and simulate the gateway packet loss in the way of every receive data we are using the unique port we are chosen before (In first part), Then we are following by the random number, If the value of the random variable is greater than 0.5, The data we received we are send it to the port “P+1” which means to port “3334”, otherwise the datagram is discarded and the process repeat, This gateway simulate an unreliable network that loses datagram with 50% probability.

In the third part, I created a program named “sink.c” which I created a socket to receive datagram from any host on port “3334”, Then all the data we receive I printed to the screen the information of where the datagram came from the IP address in dotted-decimal notation and what message it contains.

We compiled the three files: (“source.c”, “gateway.c”, “sink.c”),
Then I open three different terminals to run them separately.
First, I run the “sink.c” file to receive all the datagrams:

```
gal@gal-VirtualBox:~/CLionProjects/untitled$ '/home/gal/CLionProjects/untitled/sink'
```

Second, I run the “gateway.c” with the hostname “127.0.0.1”:

```
gal@gal-VirtualBox:~/CLionProjects/untitled$ '/home/gal/CLionProjects/untitled/gateway' 127.0.0.1
```

Third, I run the “source.c” with the hostname “127.0.0.1”:

```
gal@gal-VirtualBox:~/CLionProjects/untitled$ '/home/gal/CLionProjects/untitled/source' 127.0.0.1
```

After running these files, I can see in the terminals the simulation work as I expected to, the gateway sends the datagram to the sink if the random value is over 0.5 with the message information.

The Gateway:

```
Random variable ::0.818102 > 0.5
Got data ::<0>
ID process ::<21168>
Random variable ::0.531658 > 0.5
Got data ::<1>
ID process ::<21168>
Random variable ::0.688040 > 0.5
Got data ::<6>
ID process ::<21168>
Random variable ::0.658451 > 0.5
Got data ::<9>
ID process ::<21168>
```

In this picture I took a small sample, and this program ran to infinity. As you can see every Random variable over 0.5 the “Gateway” send the datagram with the same ID process and the same got data number to the “sink.c”, the Got data number increases every iteration of every datagram.

The Sink:

```
Packet from:: ip= 127.0.0.1, port= 62130
Got data ::<0>
ID process ::<21167>
Packet from:: ip= 127.0.0.1, port= 62130
Got data ::<1>
ID process ::<21167>
Packet from:: ip= 127.0.0.1, port= 62130
Got data ::<6>
ID process ::<21167>
Packet from:: ip= 127.0.0.1, port= 62130
Got data ::<9>
ID process ::<21167>
```

In this picture I took a small sample, and this program ran to infinity. As you can see the sink receive the datagrams from the gateway with the same Got data number and can see the whole message information the IP address and the port the packet gets from.

Our main goal has been achieved to create a gateway process that simulates datagram loss.

IP Addresses, Hostnames and HTTP

After understanding and deepening the two files (“net_server.c”, “net_client.c”), I did compile them and execute them both.
In one terminal window I run “net_server” and in another the “net_client”, there was some errors because we try to connect to the server with different port in the client, so we put the same port, and we still got error because the Ip-address is not defined yet.
Attempts connect to the server:

```
gal@gal-VirtualBox:/$ '/home/gal/CLionProjects/untitled2/cli'
Client is alive and establishing socket connection.
Error establishing communications: Network is unreachable
```

We were asked to type the command “nslookup” to find the Ip address of my computer, after finding we chose to work with the localhost Ip address and we change it in the client, we repeat the process of compiling and run them again and finally we got connection between the server and the client.
In the client we receive 10 sockets, and, in the server, we written 10 sockets and its look like that:

Server:

```
gal@gal-VirtualBox:~$ '/home/gal/CLionProjects/untitled2/ser'
Server has written 1 to socket.
Server has written 2 to socket.
Server has written 3 to socket.
Server has written 4 to socket.
Server has written 5 to socket.
Server has written 6 to socket.
Server has written 7 to socket.
Server has written 8 to socket.
Server has written 9 to socket.
Server has written 10 to socket.
```

Client:

```
gal@gal-VirtualBox:/$ '/home/gal/CLionProjects/untitled2/cli'
Client is alive and establishing socket connection.
Client has received 1 from socket.
Client has received 2 from socket.
Client has received 3 from socket.
Client has received 4 from socket.
Client has received 5 from socket.
Client has received 6 from socket.
Client has received 7 from socket.
Client has received 8 from socket.
Client has received 9 from socket.
Client has received 10 from socket.
Exiting now.
```

So far so good, now we were ask to change the “net_client.c” to accept any Ip address that’s make for us not to recompile every time we want to change the Ip just to make it generic, I have take a look in “nslookup.c” that give me inspired to do so after understand how it works and after writes the relevant comments in the code, I did rewrite “net_client.c” to accept any Ip address from the commend-line.

Terminal 1 Server:

```
gal@gal-VirtualBox:~$ '/home/gal/CLionProjects/untitled/server'
Server has written 1 to socket.
Server has written 2 to socket.
Server has written 3 to socket.
Server has written 4 to socket.
Server has written 5 to socket.
Server has written 6 to socket.
Server has written 7 to socket.
Server has written 8 to socket.
Server has written 9 to socket.
Server has written 10 to socket.
```

Terminal 2 Client:

```
gal@gal-VirtualBox:/$ '/home/gal/CLionProjects/untitled/client' 127.0.0.1
Client is alive and establishing socket connection.
Client has received 1 from socket.
Client has received 2 from socket.
Client has received 3 from socket.
Client has received 4 from socket.
Client has received 5 from socket.
Client has received 6 from socket.
Client has received 7 from socket.
Client has received 8 from socket.
Client has received 9 from socket.
Client has received 10 from socket.
The host address is 127.0.0.1
Exiting now.
```

As we can see the client accept any Ip address from the commend-line as we required to.

PART B: A Simple web client

A Simple web Client

First, I want to take a view and learn about “wget”, so I did try to fetch yahoo website with “wget” to show what’s happened,
After running the “wget” with URL of yahoo website I got this:

```
gal@gal-VirtualBox:~$ wget http://www.yahoo.com
URL transformed to HTTPS due to an HSTS policy
--2022-05-24 01:03:42--  https://www.yahoo.com/
Resolving www.yahoo.com (www.yahoo.com)... 87.248.100.215, 87.248.100.216, 2a00:
1288:110:c305::1:8001, ...
Connecting to www.yahoo.com (www.yahoo.com)|87.248.100.215|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: ‘index.html.2’

index.html.2            [  <=>               ] 714.38K  1.63MB/s    in 0.4s

2022-05-24 01:03:43 (1.63 MB/s) - ‘index.html.2’ saved [731523]
```

We got from “wget” the full source code of the index.html of yahoo website when we click the index.html its open in the browser which shows the open theme of yahoo.

In this Part the main target is to make a program that simple analog to “wget”, our program will take the URL and parsing to the forms: **<protocol>://<hostname>/<path>, <protocol>://<hostname>/<path>**

In My code I adjusted the code to fit both forms, so I take the full URL from the command-line argument send it to the parser, the parser organizes all the information we need such as port, hostname, path.

After I have completed all the instructions in the assignment for this section, I will review the program run for full URL of yahoo website.
We see the header is HTTP/1.0 301 Moved Permanently:

```
gal@gal-VirtualBox:/home$ '/home/gal/CLionProjects/untitled/web' http://www.yahoo.com
size of path 0
Web is alive and establishing socket connection.
TEST www.yahoo.com TEST
size of query: 45
GET / HTTP/1.0
HOST: www.yahoo.com


Write
HTTP/1.0 301 Moved Permanently
Date: Mon, 23 May 2022 22:17:10 GMT
Server: ATS
Cache-Control: no-store, no-cache
Content-Type: text/html
Content-Language: en
Connection: keep-alive
```

After that we asked to examine the result of the running program for the URL of yahoo with the path "/does-not-exist".
We see the same result the header is HTTP/1.0 301 Moved Permanently:

```
gal@gal-VirtualBox:/home$ '/home/gal/CLionProjects/untitled/web' http://www.yahoo.com/does
-not-exist
size of path 15
Web is alive and establishing socket connection.
TEST www.yahoo.com TEST
size of query: 46
GET /does-not-exist HTTP/1.0
HOST: www.yahoo.com


Write
HTTP/1.0 301 Moved Permanently
Date: Mon, 23 May 2022 22:19:51 GMT
Server: ATS
Cache-Control: no-store, no-cache
Content-Type: text/html
Content-Language: en
Connection: keep-alive
```

That's happen because yahoo with the path "/does-not-exist" **exists**.
So, I chose to examine the program on really website path really doesn't exist, I tried on the website "blankwebsite" with random path like "/dsadsa", then we can see the different:

```
gal@gal-VirtualBox:/home$ '/home/gal/CLionProjects/untitled/web' http://www.blankwebsite.c
om/dsadsa
size of path 7
Web is alive and establishing socket connection.
TEST www.blankwebsite.com TEST
size of query: 46
GET /dsadsa HTTP/1.0
HOST: www.blankwebsite.com


Write
HTTP/1.1 404 Not Found
Cache-Control: private
Content-Type: text/html
Server: Microsoft-IIS/8.5
Set-Cookie: ASPSESSIONIDAAQTDCBS=DNNHCIEBCMECLHDKLFBBGGCG; path=/
X-Frame-Options: SAMEORIGIN
Date: Mon, 23 May 2022 22:24:00 GMT
Connection: close
Content-Length: 20
```

Header HTTP/1.1 404 Not Found.

I'D TELL YOU A JOKE ABOUT UDP BUT YOU MIGHT NOT GET IT