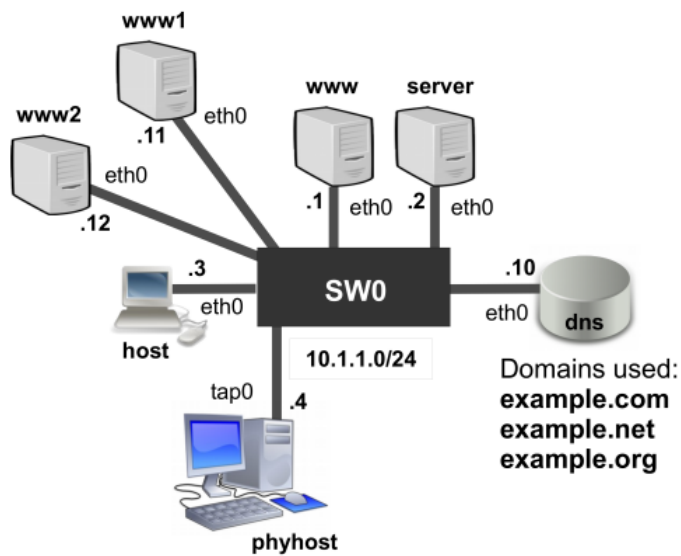


P6: WWW



Exercise 1.1– In this exercise, we are going to practice with the WEB service. To do so, start the scenario **www** shown in Figure 1.8 on your physical host (**phyhost**) by typing the following command:

```
//from terminal
simctl www-new sh
start
get www
```

```
//each virtual machines
ls /etc/nginx
```

1. Capture the traffic on **tap0** and use a **lynx** browser in the host virtual machine to connect to **www.example.com** on port **8080**. Which is the IP address associated to **www.example.com**? Why the browser is not able to establish a TCP connection with the server? Describe the DNS and TCP traffic captured.

```
//from host
lynx www.example.com:8080
```

```
Alert!: Unable to connect to remote host.

Looking up 8080 first
Looking up 8080
Making HTTP connection to 8080
Alert!: Unable to connect to remote host.

lynx: Can't access startfile http://8080/
```

Capturing from SimNet0						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe:fd:00:00:04:00	Broadcast	ARP	42	Who has 10.1.1.1? Tell 10.1.1.3
2	0.000141268	fe:fd:00:00:03:00	fe:fd:00:00:04:00	ARP	42	10.1.1.10 is at fe:fd:00:00:03:00
3	0.000218747	10.1.1.3	10.1.1.10	DNS	75	Standard query 0xbcc1 AAAA www.example.com
4	0.001233050	10.1.1.10	10.1.1.3	DNS	119	Standard query response 0xbcc1 AAAA www.example.com SOA dns.example.com
5	0.001692313	10.1.1.3	10.1.1.10	DNS	87	Standard query 0x43cd AAAA www.example.com
6	0.001827784	10.1.1.10	10.1.1.3	DNS	131	Standard query response 0x43cd No such name AAAA www.example.com SOA dns.example.com
7	0.002141223	10.1.1.3	10.1.1.10	DNS	75	Standard query 0xeade2 A www.example.com
8	0.002459081	10.1.1.10	10.1.1.3	DNS	125	Standard query response 0xeade2 A www.example.com A 10.1.1.1 NS dns.example.com A 10.1.1.10
9	0.025557790	10.1.1.3	10.1.1.10	DNS	75	Standard query 0xd4bb AAAA www.example.com
10	0.025995528	10.1.1.10	10.1.1.3	DNS	119	Standard query response 0xd4bb AAAA www.example.com SOA dns.example.com
11	0.026376705	10.1.1.3	10.1.1.10	DNS	87	Standard query 0x07a0 AAAA www.example.com
12	0.026590187	10.1.1.10	10.1.1.3	DNS	131	Standard query response 0x07a0 No such name AAAA www.example.com SOA dns.example.com
13	0.026924738	10.1.1.3	10.1.1.10	DNS	75	Standard query 0x7f86 A www.example.com
14	0.027126598	10.1.1.10	10.1.1.3	DNS	125	Standard query response 0x7f86 A www.example.com A 10.1.1.1 NS dns.example.com A 10.1.1.10
15	0.067863032	fe:fd:00:00:04:00	Broadcast	ARP	42	Who has 10.1.1.1? Tell 10.1.1.3
16	0.067980394	fe:fd:00:00:01:00	fe:fd:00:00:04:00	ARP	42	10.1.1.1 is at fe:fd:00:00:01:00
17	0.068068988	10.1.1.3	10.1.1.1	TCP	74	2924 → 8080 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=4294961810 TSecr=0 WS=2
18	0.068122430	10.1.1.1	10.1.1.3	TCP	54	8080 → 2924 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
19	0.072431642	fe:fd:00:00:01:00	fe:fd:00:00:04:00	ARP	42	Who has 10.1.1.3? Tell 10.1.1.1
20	0.072543409	fe:fd:00:00:04:00	fe:fd:00:00:01:00	ARP	42	10.1.1.3 is at fe:fd:00:00:04:00

We can observe here, that the DNS server informs the web ip is 10.1.1.1. But we cannot establish the connection because there is not a record of example.com in SOA of the dns server. And, www send rst, ack, its cause the port used by DNS is 80 and we specified port 8080.

The DNS resolves for ipv4 but not for ipv6. The ip of the machine is 10.1.1.1 and we can obtain its MAC with ARP, but we can't establish a TCP connection. That means that there is no daemon listening to that port.

2. Capture the traffic on tap0 and repeat the previous experiment, but this time execute a netcat in the www machine listening on port 8080. Which version of HTTP is using the browser? Is the connection closed? Describe the DNS and HTTP traffic and kill the netcat to finish

```
//from www
```

```
nc -l -p 8080
```

```
www:~# nc -l -p 8080
GET / HTTP/1.0
Host: www.example.com:8080
Accept: text/html, text/plain, text/css, text/sgml, */*;q=0.01
Accept-Encoding: gzip, compress, bzip2
Accept-Language: en
User-Agent: Lynx/2.8.7dev.9 libwww-FM/2.14 SSL-MM/1.4.1

Hey hola guapa
```

```
//from host
```

```
lynx www.example.com:8080
```

15	0.082498676	10.1.1.3	10.1.1.1	TCP	74	4300 → 8080 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=15954856 TSecr=0 WS=2
16	0.082632279	10.1.1.1	10.1.1.3	TCP	74	8080 → 4300 [SYN, ACK] Seq=9 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=15959327 TSecr=15954
17	0.083167761	10.1.1.3	10.1.1.1	TCP	66	4300 → 8080 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSval=15954856 TSecr=15959327
18	0.105455511	10.1.1.3	10.1.1.1	HTTP	294	GET / HTTP/1.0
19	0.105923973	10.1.1.1	10.1.1.3	TCP	66	8080 → 4300 [ACK] Seq=1 Ack=229 Win=6864 Len=0 TSval=15959330 TSecr=15954858
20	5.035970206	fe:fd:00:00:06:00	fe:fd:00:00:05:00	ARP	42	Who has 10.1.1.10? Tell 10.1.1.3
21	5.036237512	fe:fd:00:00:05:00	fe:fd:00:00:06:00	ARP	42	10.1.1.10 is at fe:fd:00:00:05:00
22	137.051432041	10.1.1.1	10.1.1.3	TCP	66	8080 → 4300 [FIN, ACK] Seq=1 Ack=229 Win=6864 Len=0 TSval=15973026 TSecr=15954858
23	137.074160784	10.1.1.3	10.1.1.1	TCP	66	4300 → 8080 [ACK] Seq=229 Ack=2 Win=5840 Len=0 TSval=15968554 TSecr=15973026
24	140.077168755	10.1.1.3	10.1.1.1	TCP	66	4300 → 8080 [FIN, ACK] Seq=229 Ack=2 Win=5840 Len=0 TSval=15968555 TSecr=15973026
25	140.077688158	10.1.1.1	10.1.1.3	TCP	66	8080 → 4300 [ACK] Seq=2 Ack=230 Win=6864 Len=0 TSval=15973327 TSecr=15968555

Http version is HTTP/1.0.

Now we have a daemon listening to the port 8080. The connection is open because there is no http response.

A TCP connection is opened between port 4300 from the host(10.1.1.1) and the port 8080 from the server(10.1.1.3).

3. Let's take a look to the nginx web server in the www machine. How many websites are available? Is there any website enabled?

//from www

ls /etc/nginx/sites-available/

ls /etc/nginx/sites-enabled/

```
www:~# ls /etc/nginx/sites-available/
balancer  default  non-existing-sites
www:~# ls /etc/nginx/sites-enabled/
default
```

There are 3 websites available, but only default is enabled.

4. Open the default configuration. What is the virtual host name(s) for this site? Where is the website content placed?

//from www

cat /etc/nginx/sites-enabled/default

```
www:~# cat /etc/nginx/sites-enabled/default
server {
    listen      80;
    # .example.com could be used for both example.com and *.example.com
    # "" is used to attend requests with no "host" header
    server_name www.example.com example.com "";

    location / {
        root    /var/www;
        index   index.html index.htm;
    }

    location ~ ^/cgi/ {
        # Disable gzip (it makes scripts feel slower since they have to comple$
        # before getting gzipped)
        gzip off;

        # Required to forbid default content browsing
        autoindex off;

        # Default document root
        root /var/www/cgi-bin;

        # Changing the url according to what fastcgi expects
        rewrite ^/cgi/(.*) /$1 break;

        # Fastcgi parameters, include the standard ones
        include /etc/nginx/fastcgi_params;

        # Fastcgi socket for library communication
        fastcgi_pass unix:/tmp/cgi.sock;

        # Setting the script filename
        fastcgi_param SCRIPT_FILENAME /var/www/cgi-bin$fastcgi_script_name;
    }
}
www:~#
```

The virtual host names are www.example.com and the website content is placed on /var/www;

5. Open the non-existing-sites configuration. What do you think the purpose of this site configuration is?

//from www

cat /etc/nginx/sites-available/non-existing-sites

```
www:~# cat /etc/nginx/sites-available/non-existing-sites
server {
    listen      80 default_server;
    server_name _;
    return 503  "No server is currently configured for the requested host." ;
}

www:~#
```

The purpose of this configuration is to inform about the errors. Any request to a server which is not correspondent, will arrive in this configuration and will show the message.

6. Enable the non-existing-sites by typing, from the sites-enabled folder, the following command: ln -s ../sites-available/non-existing-sites .

//from www

cd /etc/nginx/sites-enabled

ln -s /etc/nginx/sites-available/non-existing-sites

ls -la

```
www:/etc/nginx/sites-enabled# ls -la
total 8
drwxr-xr-x 2 root root 4096 Oct 28 19:15 .
drwxr-xr-x 4 root root 4096 Feb 10 2020 ..
lrwxrwxrwx 1 root root   26 Feb 10 2020 default -> ../sites-available/default
lrwxrwxrwx 1 root root   45 Oct 28 19:15 non-existing-sites -> /etc/nginx/sites-available/non-existing-sites
```

We added a soft link for non-existing-sites

7. Capture the traffic on tap0 and start the nginx Web server in the www machine. www# /etc/init.d/nginx start On the host machine, execute a netcat to connect to the nginx server that you have just started. Over the connection established with netcat and using HTTP 1.0, send an HTTP GET request for the resource "/". Which response do you obtain? Describe the HTTP traffic captured for the GET request.

//from www

/etc/init.d/nginx start

//from host

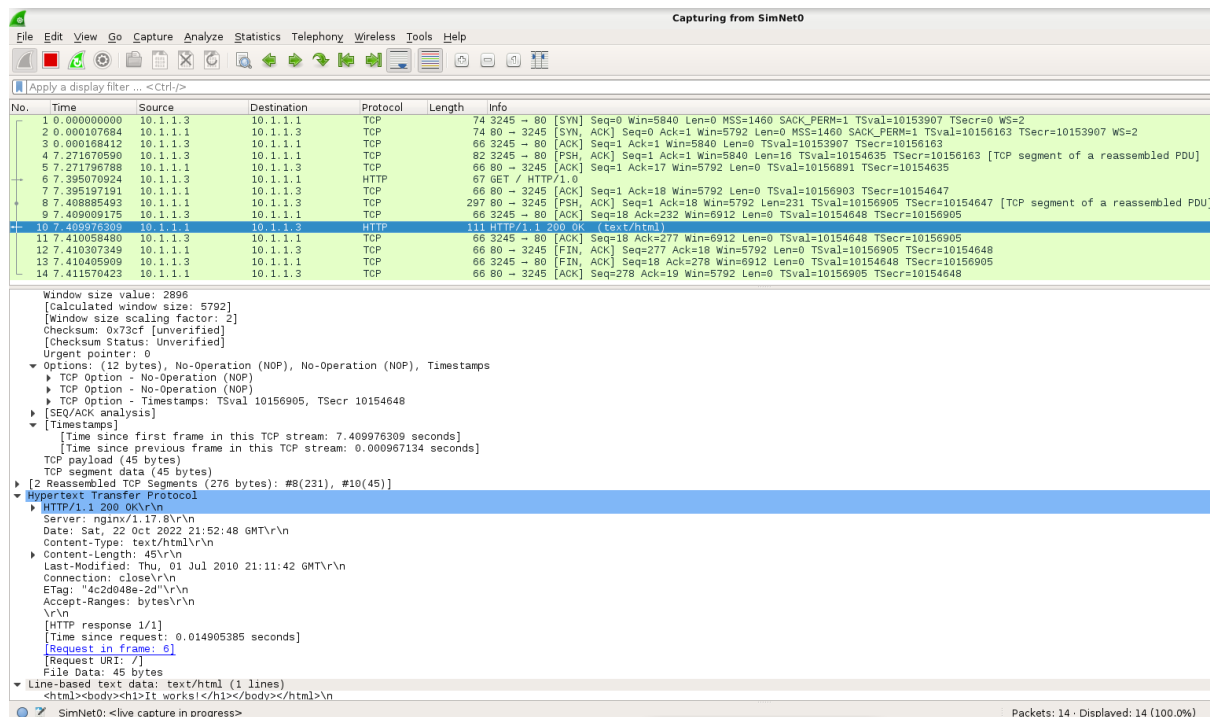
nc 10.1.1.1 80

GET / HTTP/1.0

```
host:~# nc 10.1.1.1 80
GET / HTTP/1.0

HTTP/1.1 200 OK
Server: nginx/1.17.8
Date: Sat, 22 Oct 2022 21:52:48 GMT
Content-Type: text/html
Content-Length: 45
Last-Modified: Thu, 01 Jul 2010 21:11:42 GMT
Connection: close
ETag: "4c2d048e-2d"
Accept-Ranges: bytes

<html><body><h1>It works!</h1></body></html>
```

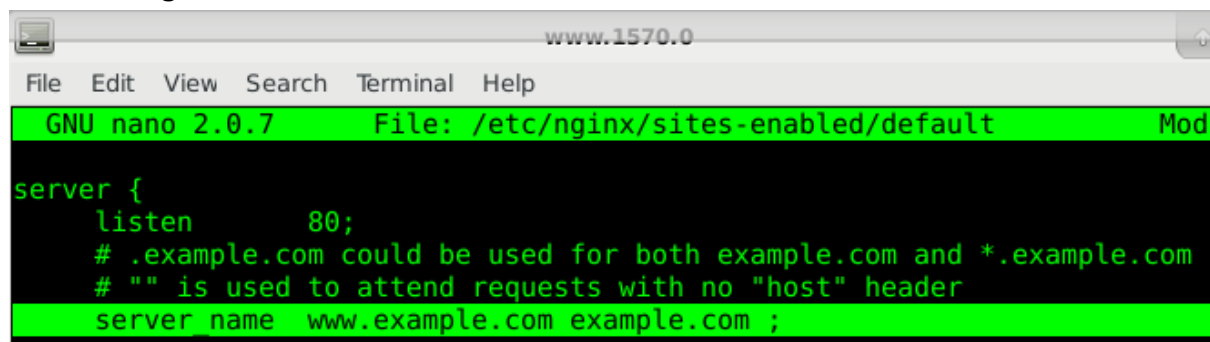


The response is ok, 200 ok.

8. Now edit the default configuration and remove the "" from the server_name section. Send again the HTTP GET request for the resource "/". Which response do you obtain now? Why? Hint: How many sites are enabled in the nginx server? What is the purpose of each of them?

//from www

nano /etc/nginx/sites-enabled/default



(we do this to attend requests without the host header)

/etc/init.d/nginx restart

//from host

nc 10.1.1.1 80

GET / HTTP/1.0

```

host:~# nc 10.1.1.1 80
GET / HTTP/1.0

HTTP/1.1 503 Service Temporarily Unavailable
Server: nginx/1.17.8
Date: Fri, 28 Oct 2022 17:44:18 GMT
Content-Type: application/octet-stream
Content-Length: 57
Connection: close

No server is currently configured for the requested host.

```

The response obtained comes from the file `/etc/nginx/sites-enabled/non-existing-sites`. There are two sites enabled. The `non-existing-sites` is for attending petitions with FQDN that don't match with our domain. Useful in case the DNS fails.

In other words, in this case as we don't specify no host headers and we drop "", then the requests are attended by non-existing sites.

9. Again, over the connection established with netcat and using HTTP 1.1, send an HTTP GET request for the resource "/" targeting the host `www.home.com` (or any other hostname you want) in `www`. Which response do you obtain now? Why? Describe the HTTP traffic captured for the GET request.

//from host

nc 10.1.1.1 80

GET www.home.com/ HTTP/1.1

#GET / HTTP/1.1

```

host:~# nc 10.1.1.1 80
GET www.home.com/ HTTP/1.1
HTTP/1.1 400 Bad Request
Server: nginx/1.17.8
Date: Sat, 05 Nov 2022 19:48:43 GMT
Content-Type: text/html
Content-Length: 157
Connection: close

<html>
<head><title>400 Bad Request</title></head>
<body>
<center><h1>400 Bad Request</h1></center>
<hr><center>nginx/1.17.8</center>
</body>
</html>
host:~#
host:~#

```

We obtain 400 Bad Request because is a bad request. As we see before, default only attend request of www.example.com and `example.com` (in 1.1 we have to specify the host header), but we put www.home.com for that reason we got an error message.

10. Send a GET request for the resource “/doc.html”. Which response do you obtain for each request? Is there a resource called doc.html in the www server? Describe the HTTP traffic captured for the GET request.

//from host

nc 10.1.1.1 80

GET /doc.html HTTP/1.0

```
host:~# nc 10.1.1.1 80
GET /doc.html HTTP/1.0

HTTP/1.1 503 Service Temporarily Unavailable
Server: nginx/1.17.8
Date: Sat, 05 Nov 2022 22:26:45 GMT
Content-Type: text/html
Content-Length: 57
Connection: close

No server is currently configured for the requested host.host:~#
```

```
www:/var/www# ls
cgi-bin  index.html
```

As we saw before in the root file /var/www we don't have any doc.html, for that reason we received an error message, because the server www couldn't find the file doc.html. Another interesting fact is now, in HTTP 1.1 the connection is closed after the request.

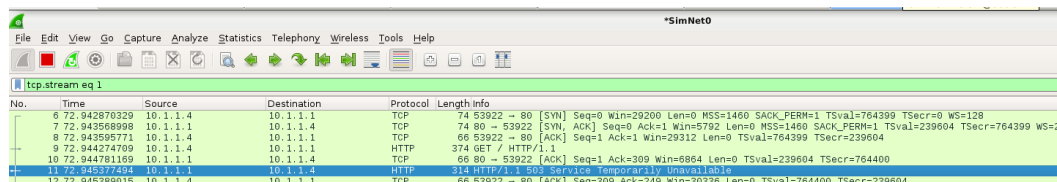
11. Configure the tap0 interface of the physical host (phyhost) with the IP address 10.1.1.4/24. After that, ask for “/” and “/doc.html” from the phyhost using a firefox browser and the IP address 10.1.1.1. Describe the HTTP traffic captured. Can you use the name www.example.com from the phyhost? why? Propose a way to reach the www machine when typing www.example.com.

//from terminal

sudo ip a add 10.1.1.4/24 dev SimNet0

//firefox

<http://10.1.1.1>



The image shows a Wireshark packet capture window titled "SimNet0". The packet list pane shows several TCP and HTTP packets. The selected packet is an HTTP GET request to / HTTP/1.1, which resulted in a 503 Service Temporarily Unavailable response. The packet details pane shows the response status and headers.

No.	Time	Source	Destination	Protocol	Length	Info
6	72.942870329	10.1.1.4	10.1.1.1	TCP	74	53922 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=764399 TSecr=0 WS=128
7	72.942868998	10.1.1.1	10.1.1.4	TCP	74	80 → 53922 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=239604 TSecr=764399 WS=2
8	72.942859771	10.1.1.4	10.1.1.1	TCP	66	53922 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=764399 TSecr=239604
9	72.944274709	10.1.1.4	10.1.1.1	HTTP	374	GET / HTTP/1.1
10	72.944781169	10.1.1.1	10.1.1.4	TCP	66	80 → 53922 [ACK] Seq=1 Ack=309 Win=6854 Len=0 TSval=239604 TSecr=764400
11	72.945377494	10.1.1.1	10.1.1.4	HTTP	314	HTTP/1.1 503 Service Temporarily Unavailable
12	72.945389015	10.1.1.4	10.1.1.1	TCP	66	53922 → 80 [ACK] Seq=309 Ack=249 Win=30336 Len=0 TSval=764400 TSecr=239604

we got an error. 503: Service Temporarily Unavailable

//from terminal

cat /etc/hosts

```
telem@debian:~$ cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 debian

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

10.1.1.1 www.example.com
```

//from terminal

firefox www.example.com



It works!

//from terminal

firefox www.example.com/doc.html



Can you use the name www.example.com from the phyhost? why? Propose a way to reach the www machine when typing www.example.com.

We really can use www.example.com because in the file `/etc/hosts` the name is active with the ip `10.1.1.1`.

In case of other cases, you may have the last line commented, then you should uncomment this line.

A screenshot of a terminal window showing the `/etc/hosts` file being edited with `GNU nano 2.7.4`. The file content is as follows:

```
127.0.0.1    localhost
127.0.1.1    debian

# The following lines are desirable for IPv6 capable hosts
::1         localhost ip6-localhost ip6-loopback
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters

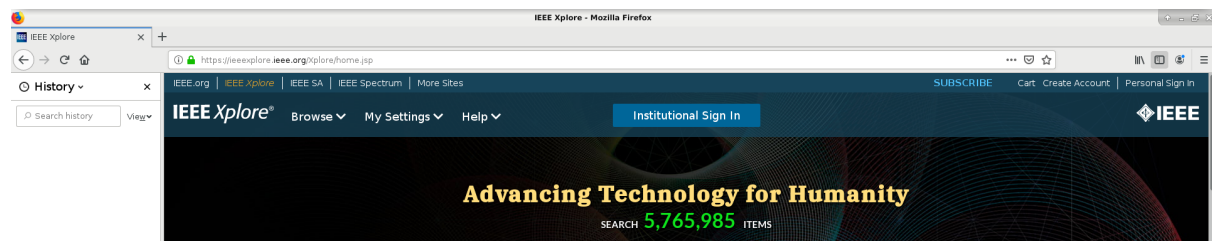
#10.1.1.1    www.example.com
```

Exercise 1.2– In this exercise we are going to practice with basic HTML content and hyperlinks.

1. Start a capture on the physical NIC, i.e. `enp8s0`. In the phyhost open a firefox browser and request for the index of a complex webpage, such as ieeexplore.ieee.org.

//from terminal

firefox



Describe the HTTP traffic captured. In particular, discuss the GET requests that you observe and the number of connections. To do this analysis easier, you can take one or combine both of the following approaches:

*enp0s3						
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help						
http						
No.	Time	Source	Destination	Protocol	Length	Info
8	0.087645040	10.0.2.15	34.107.221.82	HTTP	342	GET /success.txt HTTP/1.1
10	0.100530143	34.107.221.82	10.0.2.15	HTTP	270	HTTP/1.1 200 OK (text/plain)
76	23.163374707	10.0.2.15	184.25.36.60	HTTP	373	GET / HTTP/1.1
78	23.495250647	184.25.36.60	10.0.2.15	HTTP	223	HTTP/1.1 301 Moved Permanently
15...	26.005610739	10.0.2.15	142.250.185.3	OCSP	442	Request
15...	26.013546395	10.0.2.15	142.250.185.3	OCSP	428	Request
15...	26.085034827	142.250.185.3	10.0.2.15	OCSP	768	Response
15...	26.132165203	142.250.185.3	10.0.2.15	OCSP	756	Response
18...	26.483639187	10.0.2.15	142.250.185.3	OCSP	428	Request
18...	26.602035457	142.250.185.3	10.0.2.15	OCSP	756	Response
19...	26.957216652	10.0.2.15	18.161.108.38	OCSP	434	Request
19...	27.072587187	18.161.108.38	10.0.2.15	OCSP	1060	Response
22...	27.419768214	10.0.2.15	142.250.185.3	OCSP	428	Request
22...	27.465050891	10.0.2.15	142.250.185.3	OCSP	428	Request
22...	27.536374441	142.250.185.3	10.0.2.15	OCSP	756	Response
22...	27.583971465	142.250.185.3	10.0.2.15	OCSP	756	Response
24...	28.171555442	10.0.2.15	142.250.185.3	OCSP	427	Request
24...	28.208349236	10.0.2.15	142.250.185.3	OCSP	427	Request
24...	28.287240679	142.250.185.3	10.0.2.15	OCSP	755	Response
24...	28.329569668	142.250.185.3	10.0.2.15	OCSP	755	Response
26...	28.795408299	10.0.2.15	93.184.220.29	OCSP	425	Request
26...	28.842218259	93.184.220.29	10.0.2.15	OCSP	853	Response
27...	29.332956337	10.0.2.15	93.184.220.29	OCSP	425	Request
28...	29.373938399	93.184.220.29	10.0.2.15	OCSP	853	Response
28...	29.405744720	10.0.2.15	18.161.108.38	OCSP	434	Request
28...	29.529177877	18.161.108.38	10.0.2.15	OCSP	1060	Response

- In Wireshark, you can use the option statistics . conversations, go to the label for TCP and then, use the option follow stream for analyzing the data transmitted through each TCP connection.

Wireshark - Conversations - enp0s3												
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B
10.0.2.15	51160	104.17.24.14	443	27	13 k	13	1,855	14	11 k	23.978341	0.1317	112 k
10.0.2.15	51162	104.17.24.14	443	22	5,506	11	1,406	11	4,100	23.978692	0.0595	189 k
10.0.2.15	52770	184.25.36.60	443	96	181 k	48	9,762	48	171 k	24.048197	4.8826	17 k
10.0.2.15	47826	23.39.97.55	443	19	7,295	10	1,489	9	5,806	24.048946	10.1811	1,170
10.0.2.15	47828	23.39.97.55	443	19	7,517	10	1,485	9	6,032	24.048982	10.1811	1,166
10.0.2.15	47830	23.39.97.55	443	29	20 k	15	2,458	14	18 k	24.049004	10.1811	1,929
10.0.2.15	47832	23.39.97.55	443	23	10 k	12	1,952	11	8,051	24.049113	10.1809	1,533
10.0.2.15	47834	23.39.97.55	443	26	11 k	13	2,334	13	8,861	24.049219	10.1809	1,834
10.0.2.15	52782	184.25.36.60	443	141	305 k	69	9,481	72	296 k	24.050324	0.9317	81 k
10.0.2.15	52784	184.25.36.60	443	170	393 k	83	18 k	87	374 k	24.050761	4.5383	33 k
10.0.2.15	52786	184.25.36.60	443	410	673 k	195	23 k	215	649 k	24.051125	4.5872	41 k
10.0.2.15	52788	184.25.36.60	443	284	718 k	137	29 k	147	689 k	24.051485	7.24546	32 k
10.0.2.15	40180	54.192.111.38	443	37	52 k	17	1,934	20	50 k	24.132767	0.1621	95 k
10.0.2.15	47846	23.39.97.55	443	17	2,239	9	1,228	8	1,011	24.179347	5.8649	1,675
10.0.2.15	56770	92.124.92.215	443	62	74 k	30	3,068	32	71 k	24.706698	3.9458	145 k
10.0.2.15	43976	199.232.82.217	443	37	24 k	18	2,299	19	22 k	25.877549	0.6137	29 k
10.0.2.15	42400	216.58.215.130	443	135	395 k	60	6,350	75	189 k	25.927247	3.2289	468 k
10.0.2.15	39816	34.111.234.236	443	36	21 k	17	2,257	19	18 k	25.927775	0.7746	23 k
10.0.2.15	38724	142.250.185.3	80	15	4,106	8	1,587	7	2,519	25.992593	2.2947	5,532
10.0.2.15	38726	142.250.185.3	80	15	4,080	8	1,573	7	2,507	25.994395	2.3382	5,388
10.0.2.15	52806	184.25.36.60	443	98	201 k	48	8,866	50	192 k	26.243176	2.2772	31 k
10.0.2.15	57692	142.250.200.198	443	27	8,757	13	1,960	14	6,797	26.412513	0.2255	69 k
10.0.2.15	38732	142.250.185.3	80	7	1,486	4	610	3	876	26.467218	0.1348	36 k
10.0.2.15	45784	46.51.206.5	443	27	20 k	14	1,853	13	18 k	26.799563	0.4136	35 k
10.0.2.15	27330	18.161.108.38	80	11	3,404	6	1,104	5	2,300	26.931744	2.5075	7,083
10.0.2.15	46364	142.250.200.66	443	28	8,791	13	1,775	15	7,016	27.368646	0.2093	67 k
10.0.2.15	56834	143.250.138.162	443	36	11 k	14	1,820	15	9,733	27.3719345	0.7065	66 k

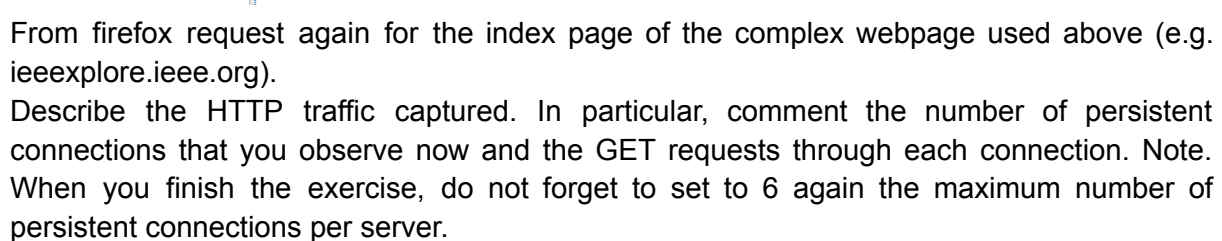
- Open the developer tools (e.g. push control+shift+i). A new section will show up in the bottom of the browser. There, one can analyze any matter related with the browser performance. Select the tab named Network, which is responsible for displaying the traffic being exchanged (e.g. when HTTP requests are committed).

Inspector Console Debugger Style Editor Performance Memory Network Storage Accessibility									
Filter URLs									
Status	Method	Domain	File	Cause	Type	Transferred	Size	0 ms	2.56 s
301	GET	ieeexplore.ieee.org	/	document	html	9.85 KB	29.98 ...	295 ms	2.56 s
301	GET	ieeexplore.ieee.org	/	document	html	10.56 KB	29.98 ...	288 ms	5.12 s
200	GET	ieeexplore.ieee.org	home.jsp	document	html	11.03 KB	29.98 ...	374 ms	7.68 s
200	GET	www.ieee.org	settings.js	script	js	900 B	448 B	361 ms	10.24 s
200	GET	cdnjs.cloudflare.com	cookieconsent.min.css	stylesheet	css	3.85 KB	1.61 KB	2592 ms	
200	GET	cdnjs.cloudflare.com	cookieconsent.min.js	script	js	6.24 KB	19.34 ...	1624 ms	
200	GET	ieeexplore.ieee.org	simplePassMeter.min.css?cv=20221004_000001	stylesheet	css	1.05 KB	1.19 KB	41 ms	
200	GET	ieeexplore.ieee.org	styles.css?cv=20221004_000001	stylesheet	css	3.81 KB	13.04 ...	359 ms	
200	GET	ieeexplore.ieee.org	customStyles.css?cv=20221004_000001	stylesheet	css	137.24 KB	810.37 ...	387 ms	
200	GET	ieeexplore.ieee.org	idangerous.swiper.css?cv=20221004_000001	stylesheet	css	1.52 KB	2.80 KB	268 ms	
200	GET	ieeexplore.ieee.org	jquery-ui-1.8.19.custom.css?cv=20221004_000001	stylesheet	css	6.70 KB	32.63 ...	133 ms	
200	GET	ieeexplore.ieee.org	jquery-3.5.1.min.js?cv=20221004_000001	script	js	30.93 KB	87.38 ...	374 ms	
200	GET	ieeexplore.ieee.org	ckeditor.js?cv=20221004_000001	script	js	183.56 KB	613.98 ...	212 ms	
200	GET	ieeexplore.ieee.org	tealiumTagsData.js?cv=20221004_000001	script	js	1.57 KB	5.08 KB	69 ms	
200	GET	ieeexplore.ieee.org	tealiumAnalytics.js?cv=20221004_000001	script	js	4.75 KB	17.72 ...	380 ms	
200	GET	ieeexplore.ieee.org	Mathjax.js?config=default	script	js	19.37 KB	61.86 ...	318 ms	
200	GET	ieeexplore.ieee.org	ie7Sniffer.css?cv=20221004_000001	stylesheet	css	1.06 KB	695 B	138 ms	
200	GET	ieeexplore.ieee.org	js.cookie.js?cv=20221004_000001	script	js	2.14 KB	3.78 KB	223 ms	
200	GET	ieeexplore.ieee.org	fingerPrint2.js?cv=20221004_000001	script	js	15.48 KB	58.73 ...	145 ms	
200	GET	ieeexplore.ieee.org	fingerPrint.js?cv=20221004_000001	script	js	1.08 KB	665 B	288 ms	
200	GET	ieeexplore.ieee.org	runtime.js?cv=20221004_000001	script	js	1.36 KB	1.16 KB	207 ms	
200	GET	ieeexplore.ieee.org	polyfills.js?cv=20221004_000001	script	js	34.88 KB	101.97 ...	206 ms	

121 requests 12.04 MB / 4.42 MB transferred Finish: 9.49 s DOMContentLoaded: 2.39 s load: 7.36 s

Status	Method	Domain	File	Cause	Type	Transferred	Size	Time	Time	Time	Time
200	GET	connect.facebook.net	1591804690854330?v=2.9.8&r=stable	script	js	8.61 KB	25.90 ...	6 ms			
200	GET	www.google.com	ga-audiences?t=sr&ai=16_v=4&sf_rd=16_v=16_v=j8&tid=UA-930232...	img	gif	698 B	42 B	54 ms			
200	GET	www.google.es	ga-audiences?t=sr&ai=16_v=4&sf_rd=16_v=16_v=j8&tid=UA-930232...	img	gif	698 B	42 B	57 ms			
200	GET	www.google.com	ga-audiences?t=sr&ai=16_v=4&sf_rd=16_v=16_v=j8&tid=UA-2313717...	img	gif	698 B	42 B	53 ms			
200	GET	www.google.es	ga-audiences?t=sr&ai=16_v=4&sf_rd=16_v=16_v=j8&tid=UA-2313717...	img	gif	698 B	42 B	56 ms			
200	GET	connect.facebook.net	1093886321010798?v=2.9.8&r=stable	script	js	85.68 KB	293.59 ...	88 ms			
200	GET	www.facebook.com	/tr?id=1591804690854330&ev=PageView&dl=https://www.iece.org/&rl=...	plain	353 B	0 B	227 ms				
200	GET	www.facebook.com	/tr?id=1093886321010798&ev=PageView&dl=https://www.iece.org/&rl=...	plain	353 B	0 B	6 ms				
200	GET	s3.amazonaws.com	index.html	subdocument	html	2.18 KB	1.80 KB	562 ms			
200	GET	insight.adsrvr.org	up?adv=gmfw&ref=https://www.iece.org/&id=lp&wud&supv=1.1.0	subdocument	html	266 B	0 B	336 ms			
200	GET	www.facebook.com	/tr?id=1093886321010798&ev=Microdata&dl=https://www.iece.org/&rl=...	plain	353 B	0 B	13 ms				
200	GET	munchkin.marketo.net	munchkin.js	script	js	1.21 KB	0 B	437 ms			
200	GET	munchkin.js	munchkin.js	script	js	5.06 KB	10.68 ...	16 ms			
200	POST	756-gph-899.mktosp.com	visitWebPage?_mchNc=16669717032216_mchCn=6_mchId=756-GPH-899...	beacon	plain	307 B	2 B	933 ms			
200	GET	app-ab24.marketo.com	getForm?munchkinId=756-GPH-899&form=1023&url=https://www.iece.org...	script	js	1.31 KB	2.62 KB	225 ms			
200	GET	munchkin.marketo.net	munchkin.js	script	js	cached	1.22 KB				
200	POST	region1.google-analytics.com	collect?v=2&tid=G-W779Y0QXP&utm=20ea06_p=1820842518&cid=575...	beacon	plain	504 B	0 B	39 ms			
200	GET	app-ab24.marketo.com	forms2.css	stylesheet	css	3.02 KB	13.05 ...	95 ms			
200	GET	app-ab24.marketo.com	forms2-theme-simple.css	stylesheet	css	752 B	826 B	44 ms			
200	GET	app-ab24.marketo.com	XDFrame	subdocument	html	1.16 KB	1.76 KB	149 ms			
200	GET	app-ab24.marketo.com	forms2.min.js	script	js	cached	207.58 ...				
200	GET	munchkin.marketo.net	munchkin.js	script	js	cached	1.22 KB				

```
//from firefox
about:config
//in search
network.http.max-persistent-connections-per-server
```



Status	Method	Domain	File	Cause	Type	Transferred	Size	Time	Size	Time	Size	Time
200	GET	leeseoplane.lee.org	604918b-0318-4527-4288-38994+3ab6_highlight.jpg	img	jpeg	18.50 KB	17.89				676 ms	
200	GET	adservice.google	integrator.js?domain=leeseoplane.lee.org	script	js	1.08 KB	107 B				648 ms	
200	GET	dpm.demdex.net	r6d_vissd_ver=3.0.0-6_fieldGroup=MC6_d_rtd=jsan6_d_ver=26_yerf-...	json	1.10 KB	369 B					64 ms	
200	GET	leeseoplane.demdex.net	destd5.html?id=0	subdocument	html	3.37 KB	6.82 KB				553 ms	
200	GET	smetrics-leeseoplane.lee.org	ie6d_vissd_ver=3.0.0-6_fieldGroup=AdmcorpId=BE92CC25A1F2B0A495...	shr	js	722 B	48 B				456 ms	
200	GET	adservice.google	integrator.js?domain=leeseoplane.lee.org	script	js	783 B	107 B				538 ms	
302	GET	cm.everesttech.net	d6d_vissd=83222646705702609379821895272826024	img	gif	536 B	42 B				838 ms	
200	GET	smetrics-leeseoplane.lee.org	s9139a260313317a0d=1n6r=1a6f=1a6d=269/2022.19.15.5-12064d...	img	gif	768 B	43 B				45 ms	
200	GET	securepubads.g.doubleclick.net	4497376127651295321053080=correlator=26283614977266=vid=447...	plain	1.43 KB	714 B					18 ms	
200	GET	00362ba89550ac6dc1799f	container.html	subdocument	html	3.86 KB	5.95 KB				363 ms	
200	GET	leeseoplane.lee.org	MathMenu.js?v=2.7.4	script	js	12.15 KB	37.35				12 ms	
200	GET	dpm.demdex.net	lrs_dpid=116&dpsid=Yiw0WAAAFuGhtn	img	gif	929 B	42 B				73 ms	
200	GET	dst0tufpuf.cloudfront.net	ieee-button=03862ab8e4e6ad785629a658f0033.png	img	png	2.52 KB	1.96 KB				388 ms	
200	GET	pagead2.googlesyndication.com	sodar?sv=2006261d6gts6v=20221025016tsewme	shr	json	11.55 KB	14.30				233 ms	
200	GET	tpc.googlesyndication.com	sodar2.js	script	js	7.02 KB	16.91				628 ms	
200	GET	leeseoplane.lee.org	MathZoom.js?v=2.7.4	script	js	3.74 KB	8.46 KB				12 ms	
200	GET	tpc.googlesyndication.com	runner.html	subdocument	html	5.74 KB	12.52				124 ms	
200	GET	www.google.com	afama	subdocument	html	1.40 KB	783 B				592 ms	
200	GET	pagead2.googlesyndication.com	y9b0z6=0e4d0f9b7V3UC+amOWL_RU205aPwjs	script	js	16.44 KB	16.47				16 ms	
200	GET	pagead2.googlesyndication.com	sodar?id=sodar36v=256&id=get_20211025016h=1790130875532056v=...	img	html	568 B	0				63 ms	
200	GET	tpc.googlesyndication.com	generate_2647v6dA	plain	320 B	0					12 ms	
200	GET	pagead2.googlesyndication.com	sodar?id=sodar36v=256&id=261m=get_202105016h=179013087553210...	img	html	568 B	0				63 ms	

2n request

Status	Method	Domain	File	Cause	Type	Transferred	Other	Size	Time	Speed	Cache	Throttling
200	GET	standards.ieee.org	ieee_apple_touch_icon.png	img	webp	2.46 KB		1.91 KB	65 ms			
200	GET	standards.ieee.org	ieee_favicon.ico	img	x-icon	1.97 KB		1.97 KB	38 ms			
200	GET	snaps.lcln.ch	insight.min.js	script	js	780 B (traced)		597 B	16 ms			
200	GET	googleads.g.doubleclick.net	/pageadviewthroughconversion/976807109/random=16669773517356v...	script	js	2.17 KB		2.44 KB	164 ms			
200	POST	region1.google-analytics.com	collect?v=26tid=G-43K51H1X0F>m=Zoea0q6_p=761328864&cid=5354...	beacon	plain	504 B	0 B		230 ms			
200	GET	app=28.marketo.com	forms2.css	stylesheet	css	3.01 KB		13.05 ...	78 ms			
200	GET	app=28.marketo.com	forms2-theme-simple.css	stylesheet	css	740 B		826 B	72 ms			
200	POST	region1.google-analytics.com	collect?v=26tid=G-PRKVE68LM6>m=Zoea0q6_p=761328864&cid=5354...	beacon	plain	504 B	0 B		144 ms			
200	POST	region1.google-analytics.com	collect?v=26tid=G-43K03VE2B>m=Zoea0q6_p=761328864&cid=5354...	beacon	plain	504 B	0 B		39 ms			
200	POST	region1.google-analytics.com	collect?v=26tid=G-HDL2M65576>m=Zoea0q6_p=761328864&cid=5354...	beacon	plain	504 B	0 B		38 ms			
200	GET	www.google-analytics.com	/pagead/ips-user/976807109/random=16669773517356v&cid=166...	img	gif	761 B		42 B	43 ms			
200	GET	www.google-analytics.com	/pagead/ips-user/976807109/random=16669773517356v&cid=166...	img	gif	761 B		42 B	163 ms			
200	POST	region1.google-analytics.com	collect?v=26tid=G-DM725P60E>m=Zoea0q6_p=761328864&cid=5354...	beacon	plain	504 B			46 ms			
200	GET	vars.hotjar.com	box-c1417f7848595d0dcac01c8f9546dbb.html	subdocument	html	1.69 KB		2.91 KB	264 ms			
200	GET	snaps.lcln.ch	insight.beta.min.js	script	js	4.84 KB		10.78 ...	24 ms			
200	GET	munchkin.marketo.net	munchkin.js	script	js	5.06 KB		12.60 ...	33 ms			
200	GET	www.google-analytics.com	analytics.js	script	js	20.20 KB		49.05 ...	152 ms			
200	GET	script.hotjar.com	msubids.5a17f10a21a6f98b41.js	script	js	65.25 KB		253.41 KB	247 ms			
200	GET	app=28.marketo.com	scriptname	subdocument	html	1.16 KB		1.76 KB	426 ms			
200	GET	munchkin.marketo.net	munchkin.js	script	js	cached		1.23 KB				
200	GET	snaps.lcln.ch	insight.beta.min.js	script	js	4.84 KB (traced)		12.78 ...	35 ms			
200	OPTIONS	lcln.ch	insight.beta.min.js	token	xhr	plain	437 B	0 B	433 ms			

Comparing the captures, we observe that with 6 as a max number of persistent connections there is more spontaneous request than with the 2 max number of persistent connections. It's logic, because with 6 you can do more request than with 2.

3. Under the directory /tmp of the virtual machine www you will find files with images. In www, copy these images to a directory called “images” relative to the DocumentRoot of the NGINX’s default site (named “default”). Note. You must create the “images” directory.

```
//from www
```

```
cd /tmp/
```

```
www:/tmp# ls
cgi.sock  upc1.gif  upc2.gif  upc3.png
www:/tmp#
```

```
cd /var/www
```

```
mkdir images
```

```
cp /tmp/upc1.gif /tmp/upc2.gif /tmp/upc3.png /var/www/images/
```

cd images

Is

```
www:/etc/nginx/sites-enabled# ls /tmp/
cgi.sock  upcl.gif  upc2.gif  upc3.png
www:/etc/nginx/sites-enabled# cd /var/www/
www:/var/www# ls
cgi-bin  index.html
www:/var/www# mkdir images
www:/var/www# cp /tmp/upcl.gif /tmp/upc2.gif /tmp/upc3.png /var/www/images/
www:/var/www# ls
cgi-bin  images  index.html
www:/var/www# cd images/
www:/var/www/images# ls
upcl.gif  upc2.gif  upc3.png
www:/var/www/images#
```

Modify the HTML index of the server and create local hyperlinks to these images.

Describe how you do it.

```
//from www
```

nano index.html

<html>

```

<head>
<title> Hello There</title>
</head>
<body>
<p> YEAAHH </p>
<p>You can visit the UPC home page at <a href='http://www.upc.edu'>UPC HOME </a>
</p>
<a href=/images/upc1.gif> UPC_1 </a>
<a href=/images/upc2.gif> UPC_2 </a>
<a href=/images/upc3.png> UPC_3 </a>
</body>
</html>

```

```

GNU nano 2.0.7 File: index.html

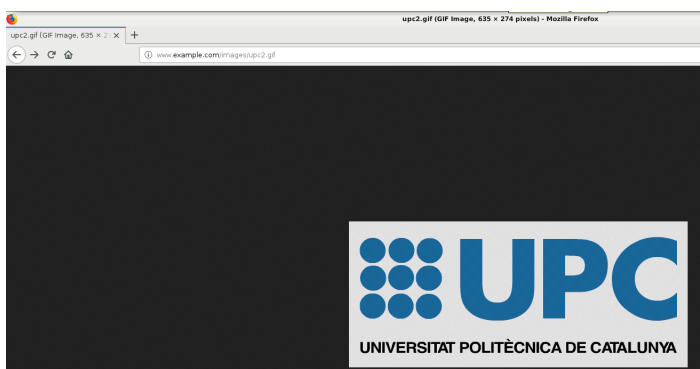
<html>
<head>
<title> Hello World</title>
</head>
<body>
<p>Hello <b>World</b>!!!!!!</p>
<p>You can visit the UPC home page at <a href="http://www.upc.edu">UPC home</a>$
<a href=/images/upc1.gif> UPC_1 </a>
<a href=/images/upc2.gif> UPC_1 </a>
<a href=/images/upc3.png> UPC_1 </a>
</body>
</html>

```

we edit the index file with nano and put the images with hyperlink.

//from terminal

firefox www.example.com



The hyperlink works, but its has to be connected via www.example.com and not via <http://10.1.1.1> because in the pyhost, in file /etc/host the example.com is active.

4. Start a capture on the tap0 interface. Execute a netcat in the phyhost to connect to 10.1.1.1 port 80 redirecting the output of this command to a file called response.http. Through the established connection type an HTTP request for the resource upc1.gif using HTTP 1.0.

```
telem@debian:~$ nc 10.1.1.1 80 >response.http
GET /images/upc1.gif HTTP/1.0
invalid port >response.http
WTF??
```

Explain how you do it.

Edit appropriately the file response.http with mousepad or vi to obtain the original image upc1.gif. Save the image with the name image.gif and test that it is correct using the command display: phyhost# display image.gif You should be able to see an UPC logo. After that, repeat the process using HTTP 1.1.

5. Repeat the process to obtain upc1.gif but this time use the command wget. Explain how you do it (consult the manual page of wget if necessary). Which version of HTTP is used by wget?

//from terminal

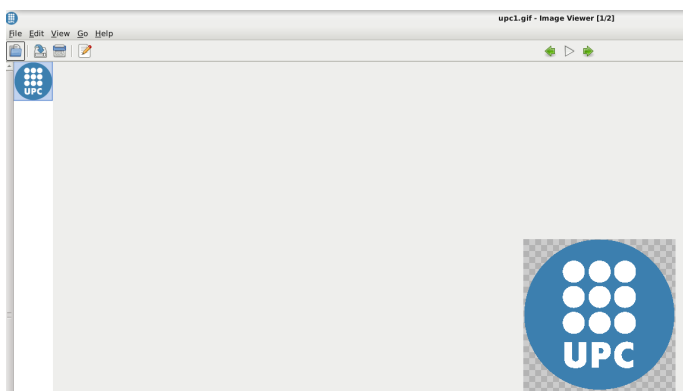
wget http://www.example.com/images/upc1.gif

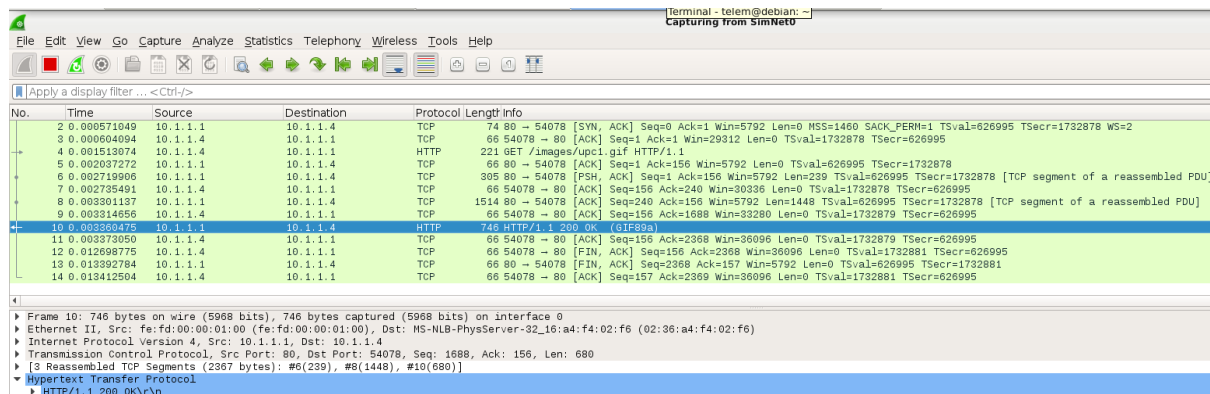
```
telem@debian:~$ wget http://www.example.com/images/upc1.gif
--2022-11-09 22:21:25-- http://www.example.com/images/upc1.gif
Resolving www.example.com (www.example.com)... 10.1.1.1
Connecting to www.example.com (www.example.com)|10.1.1.1|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2128 (2.1K) [image/gif]
Saving to: 'upc1.gif'

upc1.gif          100%[=====] 2.08K  --.-KB/s   in 0s

2022-11-09 22:21:25 (120 MB/s) - 'upc1.gif' saved [2128/2128]

telem@debian:~$
```





In Wireshark we can see the HTTP version is HTTP 1.1.

6. Get a console on the the server and edit the Start nginx on the server virtual machine. On this machine, modify its HTML index to include HTTP hyperlinks to the images upc1.gif and upc2.gif that are in www. Use domain names (not IP addresses) to create these hyperlinks. Explain how you do it.

//from server

/etc/init.d/nginx start

nano /var/www/index.html

```
server:~# cat /var/www/index.html
<html>
<head>
<title> server</title>
</head>
<body>
<p>Hello <b>World</b>!!!!!!</p>
<p>You can visit the UPC home page at <a href="http://www.upc.edu">UPC home</a>.
</p>



</body>
</html>
server:~#
```

Instead of hyperlinking we just specified the complete path of the images.

7. Stop the nginx server in www. Start a capture on tap0. From phyhost, use a firefox browser to request for the index page of server. Now, from host (virtual machine), use a lynx browser to request for the index page of server using the short name (server) and the fully qualified name (server.example.com).

Describe how you do it and explain the DNS, TCP and HTTP traffic captured.

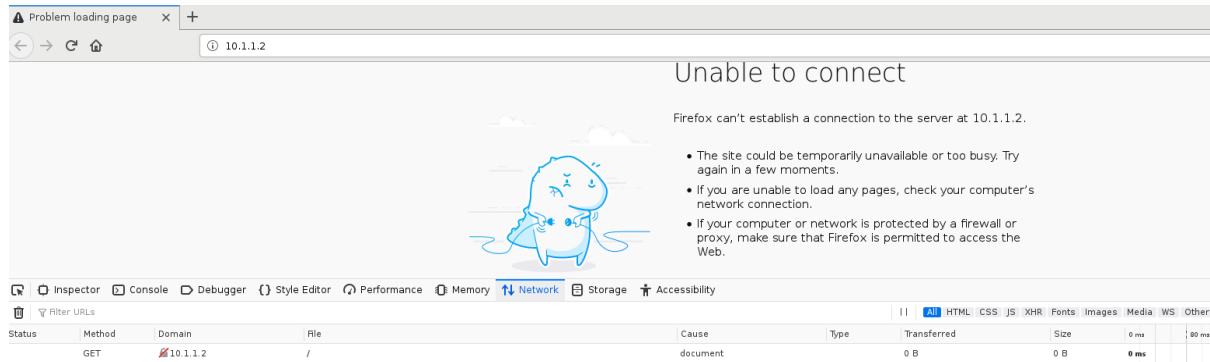
//from server

/etc/init.d/nginx stop

```
server:~# /etc/init.d/nginx stop
Stopping nginx: nginx.
```

//from terminal

firefox http://10.1.1.2



At first, the connection wasn't possible because the daemon was stopped. But after restarting the server, we can obtain the images.

DNS, FRAME???

8. Start nginx in the www machine. Start a capture on tap0. From phyhost, use a firefox browser to request for the index page of the server machine. Describe how you do it and explain the HTTP traffic captured.

//from www

/etc/init.d/nginx start



FALTA posar "DETALLS". como son tcp en cada gets, etc, etc, etc

Exercise 1.3– In this exercise we are going to practice with CGIs and HTML forms using the GET method.

1. Enable the CGI written in Bash according to the code Code 1.6, so that it can be run through the nginx server. Name the script datecgi.sh. Be careful with the written code if you cut paste the content from the PDF, since that operation often includes unwanted extra characters. Describe the steps of your configuration.

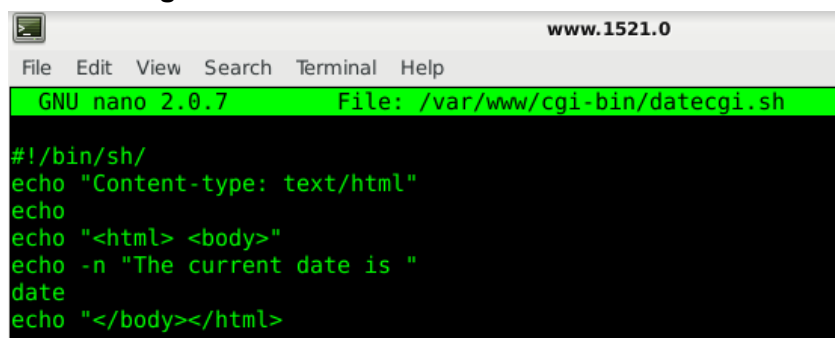
//from www

cat /etc/nginx/sites-enabled/default

```
# Default document root
root /var/www/cgi-bin;
```

cd /var/www/cgi-bin/

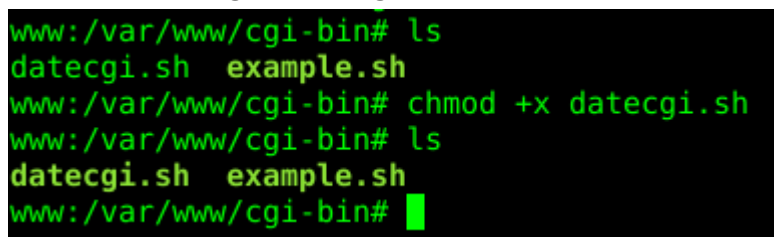
nano datecgi.sh

A screenshot of a terminal window titled 'www.1521.0'. The terminal shows the nano 2.0.7 text editor editing the file /var/www/cgi-bin/datecgi.sh. The content of the file is:#!/bin/sh/echo "Content-type: text/html"echoecho "<html> <body>"echo -n "The current date is "dateecho "</body></html>"

```
#!/bin/sh/
echo "Content-type: text/html"
echo
echo "<html> <body>"
echo -n "The current date is "
date
echo "</body></html>"
```

```
#!/bin/sh
echo "Content-type: text/html"
echo
echo "<html> <body>"
echo -n "The current date is "
date
echo "</body> </html>"
```

chmod +x datecgi.sh #to give permission

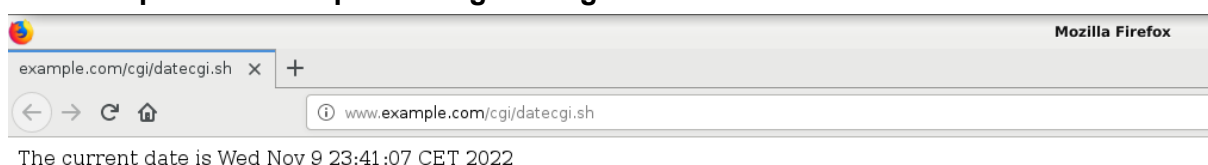
A screenshot of a terminal window showing the execution of the chmod command. The prompt is www:/var/www/cgi-bin#. The user enters 'ls', then 'chmod +x datecgi.sh', then 'ls' again, and finally a blank line.

```
www:/var/www/cgi-bin# ls
datecgi.sh  example.sh
www:/var/www/cgi-bin# chmod +x datecgi.sh
www:/var/www/cgi-bin# ls
datecgi.sh  example.sh
www:/var/www/cgi-bin#
```

2. Using the browser at the phyhost, open the URL <http://www.example.com/cgi/datecgi.sh> and verify that the CGI works as expected.

//from terminal

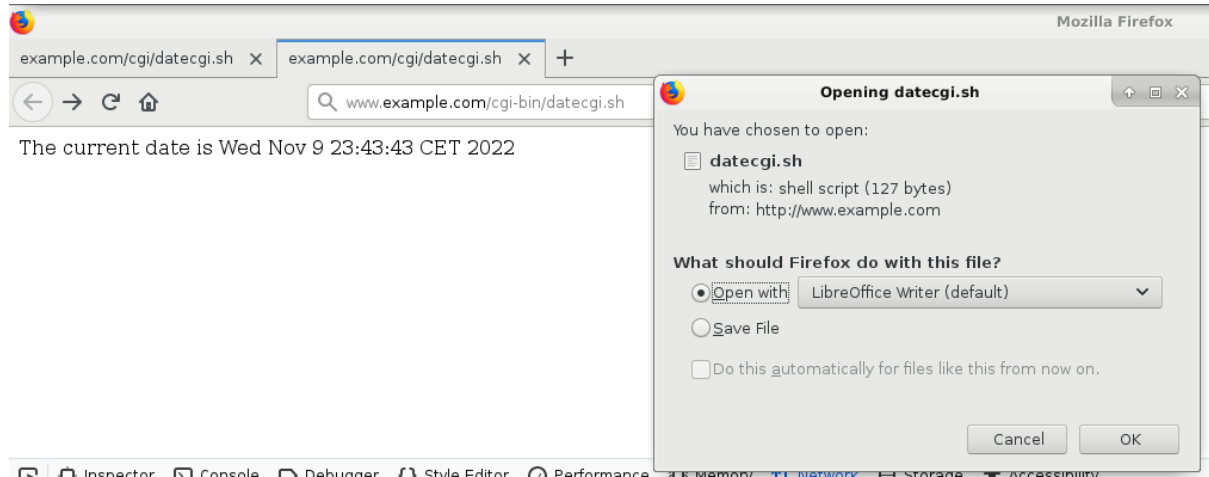
firefox <http://www.example.com/cgi/datecgi.sh>



3. Open the URL `http://www.example.com/cgi-bin/datecgi.sh`. Why does the browser try to download the file `datecgi.sh` instead of running the CGI? Hint: check the location sections of the default configuration.

//from firefox

`http://www.example.com/cgi-bin/datecgi.sh`



//from www

`cat /etc/nginx/sites-enabled/default`

```
www.14946.0
File Edit View Search Terminal Help
# .example.com could be used for both example.com and *.example.com
# "" is used to attend requests with no "host" header
server_name www.example.com example.com "";

location / {
    root /var/www;
    index index.html index.htm;
}

location ~ ^/cgi/ {
    # Disable gzip (it makes scripts feel slower since they have to comple$
    # before getting gzipped)
    gzip off;

    # Required to forbid default content browsing
    autoindex off;

    # Default document root
    root /var/www/cgi-bin;

    # Changing the url according to what fastcgi expects
    rewrite ^/cgi/(.*) /$1 break;

    # Fastcgi parameters, include the standard ones
    include /etc/nginx/fastcgi_params;

    # Fastcgi socket for library communication
    fastcgi_pass unix:/tmp/cgi.sock;

    # Setting the script filename
    fastcgi_param SCRIPT_FILENAME /var/www/cgi-bin$fastcgi_script_name;
}

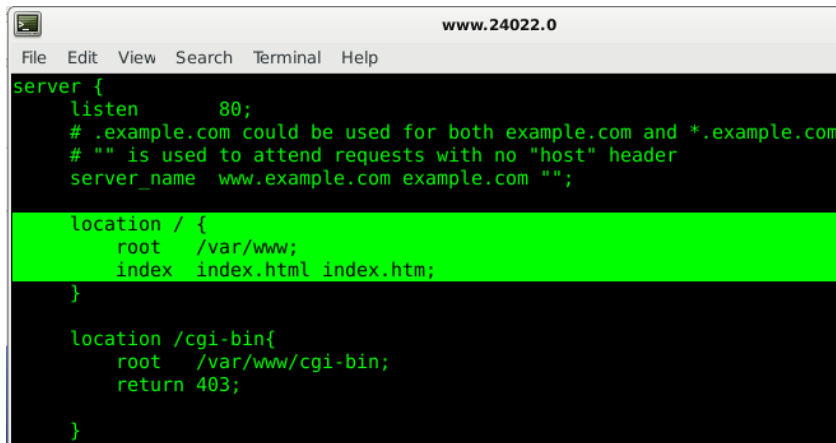
www:/var/www/cgi-bin# Cannot get script name, are DOCUMENT_ROOT and SCRIPT_NAME (or SCRIPT_FILENAME)
set and is the script executable?
cat /etc/nginx/sites-enabled/default
```

As we know, CGI are an executable program, they are used to generate resource when its requested. So, `cgi-bin` is an executable program and the `cgi` without `bin` is a resource.

4. Update the default configuration so that any access to the content stored in the /cgi-bin directory returns a 403 error message. Clean the browser cache before testing the solution. Hint: Look at the non-existing-sites configuration.

//from www

nano /etc/nginx/sites-available/default



```
server {
    listen      80;
    # .example.com could be used for both example.com and *.example.com
    # "" is used to attend requests with no "host" header
    server_name www.example.com example.com "";

    location / {
        root    /var/www;
        index   index.html index.htm;
    }

    location /cgi-bin{
        root    /var/www/cgi-bin;
        return 403;
    }
}
```

/etc/init.d/nginx restart

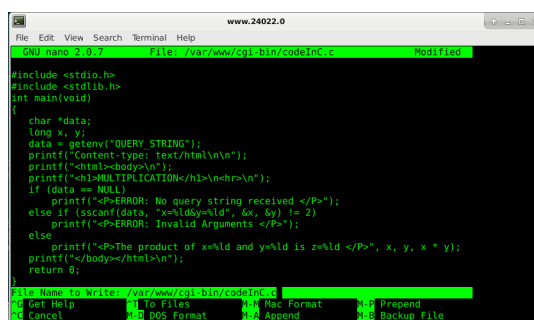
5. Enable now the CGI written in C of Code 1.7, which builds an HTML form on demand. Describe the steps of your configuration.

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char *data;
    long x, y;
    data = getenv("QUERY_STRING");
    printf("Content-type: text/html\n\n");
    printf("<html><body>\n");
    printf("<h1>MULTIPLICATION</h1>\n<hr>\n");
    if (data == NULL)
        printf("<P>ERROR: No query string received </P>");
    else if (sscanf(data, "x=%ld&y=%ld", &x, &y) != 2)
        printf("<P>ERROR: Invalid Arguments </P>");
    else
        printf("<P>The product of x=%ld and y=%ld is z=%ld </P>", x, y, x * y);
    printf("</body></html>\n");
    return 0;
}
```

//from www

cd /var/www/cgi-bin/

nano codeInC.c

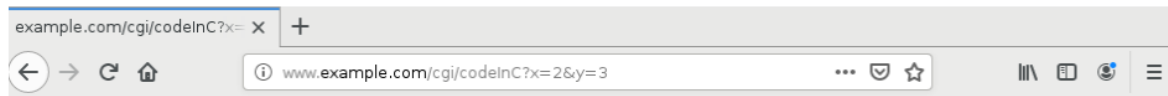


```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char *data;
    long x, y;
    data = getenv("QUERY_STRING");
    printf("Content-type: text/html\n\n");
    printf("<html><body>\n");
    printf("<h1>MULTIPLICATION</h1>\n<hr>\n");
    if (data == NULL)
        printf("<P>ERROR: No query string received </P>");
    else if (sscanf(data, "x=%ld&y=%ld", &x, &y) != 2)
        printf("<P>ERROR: Invalid Arguments </P>");
    else
        printf("<P>The product of x=%ld and y=%ld is z=%ld </P>", x, y, x * y);
    printf("</body></html>\n");
    return 0;
}
```

gcc codeInC.c -o codeInC #as c file, need to compile, (memories of FO :())

chmod +x codeInC
/etc/init.d/nginx restart

<http://www.example.com/cgi/codeInC?x=2&y=3>



MULTIPLICATION

The product of x=2 and y=3 is z=6

Exercise 1.4– In this exercise we are going to practice with the Web service using multiple domains and multiple IP addresses.

1. First, be sure that nginx is running properly, by checking which software is actually managing the port 80. Now we are going to add the name www.example.net in the dns server to be translated to the IP address 10.1.1.1. To do so, add an A register in the file /etc/bind/db.example.net of the dns machine and restart bind:

Notice that after this configuration, www.example.com and www.example.net both are translated to the same IP address (10.1.1.1). Check that the configuration is correct with pings from host.

//from www

/etc/init.d/nginx start

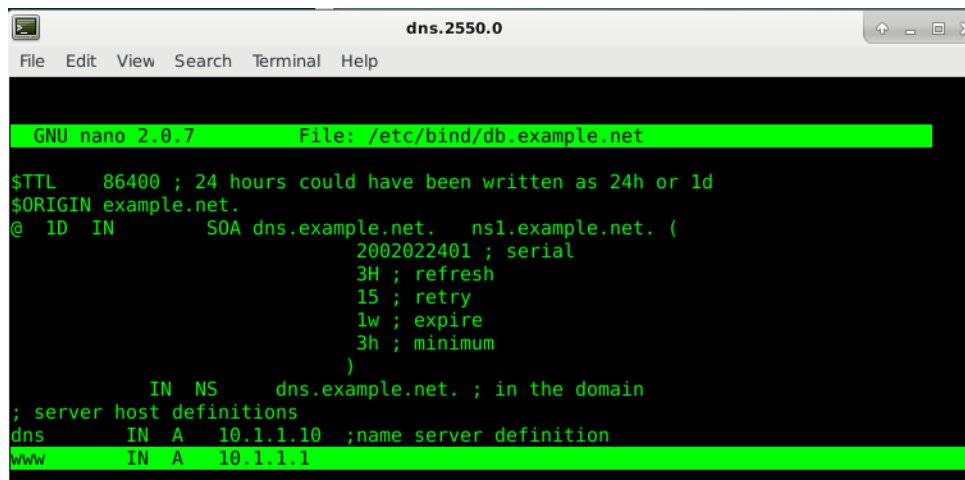
netstat -tnlp | grep nginx

```
www:~# netstat -tnlp | grep nginx
tcp        0      0 0.0.0.0:80                0.0.0.0:*               LISTEN
1178/nginx
```

//from dns

nano /etc/bind/db.example.net

www IN A 10.1.1.1



/etc/init.d/bind9 restart

#as we know of previous labs, after editing .conf files we need to restart

```
//from host
ping -f www.example.net
ping -f www.example.com
```

```
host:~# ping -f www.example.com
PING www.example.com (10.1.1.1) 56(84) bytes of data.

[1]+  Stopped                  ping -f www.example.com
host:~# ping -f www.example.net
PING www.example.net (10.1.1.1) 56(84) bytes of data.

[2]+  Stopped                  ping -f www.example.net
```

As we can see the webs have ip 10.1.1.1

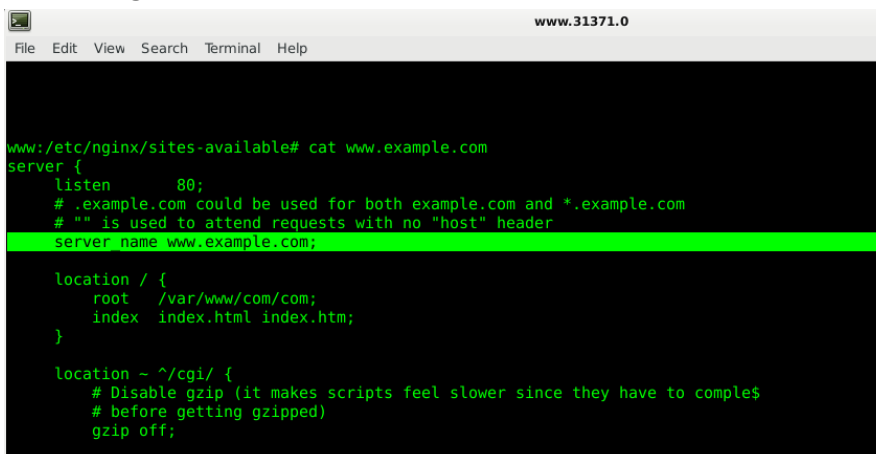
2. For the previous domain names, we are going to create and activate two sites (or "virtual hosts") in nginx. To do so, type the following:

```
//form www
cd /etc/nginx/sites-available/
cp default www.example.com
cp default www.example.net
sed -i "s/\var/www/\var/www/com/g" www.example.com
sed -i "s/\var/www/\var/www/net/g" www.example.net
sed -i "s/server_name.*/server_name www.example.com;/" www.example.com
sed -i "s/server_name.*/server_name www.example.net;/" www.example.net
```

```
www:/etc/nginx/sites-available# cd /etc/nginx/sites-available/
www:/etc/nginx/sites-available# cp default www.example.com
cp: overwrite `www.example.com'?
www:/etc/nginx/sites-available# sed -i "s/\var/www/\var/www/com/g" www.example.com
www:/etc/nginx/sites-available# sed -i "s/\var/www/\var/www/net/g" www.example.net
www:/etc/nginx/sites-available# sed -i "s/server_name.*/server_name www.example.com;/" www.example.com
www:/etc/nginx/sites-available# sed -i "s/server_name.*/server_name www.example.net;/" www.example.net
www:/etc/nginx/sites-available#
```

Open the text files and verify that both, document root and server name have been properly changed. Now, we need to enable these sites.

```
//from www
cat /etc/nginx/sites-available/www.example.com
cat /etc/nginx/sites-available/www.example.net
```



```
www:/etc/nginx/sites-available# cat www.example.com
server {
    listen      80;
    # .example.com could be used for both example.com and *.example.com
    # "" is used to attend requests with no "host" header
    server_name www.example.com;

    location / {
        root    /var/www/com;
        index   index.html index.htm;
    }

    location ~ ^/cgi/ {
        # Disable gzip (it makes scripts feel slower since they have to comple$
        # before getting gzipped)
        gzip off;
    }
}
```

First we need to disable the default website, as it attended requests to any address in the domains example.com and example.net. To do so, type:

//from www (hacer uno por uno, no todo seguido)

cd /etc/nginx/sites-enabled/

rm default

ln -s ../sites-available/www.example.com #s= set a softlink

ln -s ../sites-available/www.example.net

/etc/init.d/nginx reload

```
www:/etc/nginx/sites-enabled# ls
default
www:/etc/nginx/sites-enabled# sudo rm -r default
www:/etc/nginx/sites-enabled# ls
www:/etc/nginx/sites-enabled# ln -s ../sites-available/www.example.com #s= softlink
www:/etc/nginx/sites-enabled# ln -s ../sites-available/www.example.net
www:/etc/nginx/sites-enabled# ls
www.example.com  www.example.net
www:/etc/nginx/etc/nginx/sites-enabled# sed -i "s/\var/www/\var/www/comwww:/etc/nginx/sites-enabled#
www:/etc/init.d/nginx reload
www:/etc/nginx/sites-enabled# /etc/init.d/nginx reload
Reloading nginx configuration: nginx.
```

Generate two different index.html files for each domain and put them on the right place.

Remember, contents of www.example.com and www.example.net must be placed in the folders /var/www/com

and /var/www/net respectively. These folders need to be created in advance by typing:

//from www

mkdir /var/www/com

mkdir /var/www/net

nano com/index.html

(crear fichero html)

nano net/index.html

(crear fichero html)

```
www:/var/www# ls
cgi-bin  com  index.html  net
www:/var/www# cat com/index.html
<html><body><h1>Hello des de www.example.com!</h1></body></html>
www:/var/www# cat net/index.html
<html><body><h1>Hello des de www.example.net!</h1></body></html>
www:/var/www#
```

/etc/init.d/nginx reload

Capture on tap0 and describe how you test this configuration. Try also to connect directly with the IP address 10.1.1.1. Discuss the results.

//from terminal

nc 10.1.1.1 80

GET / HTTP/1.1

host: www.example.com:80

per comprar q et surt el html del net, host: www.example.net:80

```

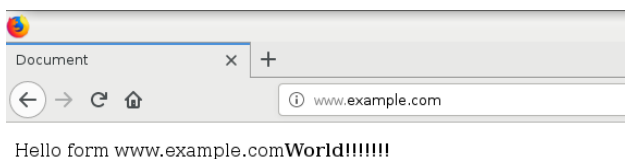
telem@debian:~$ nc 10.1.1.1 80
GET / HTTP/1.1
host: www.example.com:80

HTTP/1.1 200 OK
Server: nginx/1.17.8
Date: Thu, 10 Nov 2022 15:56:39 GMT
Content-Type: text/html
Content-Length: 321
Last-Modified: Thu, 10 Nov 2022 15:50:13 GMT
Connection: keep-alive
ETag: "636d1db5-141"
Accept-Ranges: bytes

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <p>Hello form www.example.com<b>World</b>!!!!!!</p>
</head>
<body>
</body>
</html>

```

//from firefox



if we put another host, or we didn't put the host, we will receive 400 error.

```

telem@debian:~$ nc 10.1.1.1 80
GET / HTTP/1.1

HTTP/1.1 400 Bad Request
Server: nginx/1.17.8
Date: Thu, 10 Nov 2022 16:06:11 GMT
Content-Type: text/html
Content-Length: 157
Connection: close

<html>
<head><title>400 Bad Request</title></head>
<body>
  <center><h1>400 Bad Request</h1></center>
  <hr><center>nginx/1.17.8</center>
</body>
</html>
telem@debian:~$

```

3. Now we are going to test a configuration where the domain `www.example.org` is translated by the dns server to two different IP addresses: `10.1.1.1` and `10.1.1.2`. The bind9 server uses a round robin strategy for translating names with multiple IP addresses (a different IP address for each query in a cyclic way). So, add the A registers that you consider necessary in the file `/etc/bind/db.example.org` in dns and reload the server.

Then, activate this domain in www and server with the following commands:

//from dns

nano /etc/bind/db.example.org

`www IN A 10.1.1.1`

`www IN A 10.1.1.2`

```
dns:~# cat /etc/bind/db.example.org
$TTL      86400 ; 24 hours could have been written as 24h or 1d
$ORIGIN   example.org.
@ 1D IN    SOA dns.example.org.  ns1.example.org. (
                                2002022401 ; serial
                                3H ; refresh
                                15 ; retry
                                1w ; expire
                                3h ; minimum
                                )
                IN NS      dns.example.org. ; in the domain
; server host definitions
dns      IN  A   10.1.1.10 ;name server definition
www      IN  A   10.1.1.1
www      IN  A   10.1.1.2
```

Create the same `index.html` file in both web servers (remember to place it properly), test the configuration and discuss the results. For what do you think that this configuration is useful?

//from www

cd /etc/nginx/sites-available/

cp default www.example.org

sed -i "s/\/var\/www\/\/var\/www\/org/g" www.example.org

sed -i "s/server_name.*/server_name www.example.org;/" www.example.org

cd /etc/nginx/sites-enabled/

rm default 2> /dev/null

ln -s ../sites-available/www.example.org

/etc/init.d/nginx reload

mkdir /var/www/org

nano index.html

```
www:/var/www/org# cat index.html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <p>Hello form www.example.com<b>AVER SI FUNCIONA COÑ00</b>!!!!!!</p>
</head>
<body>

</body>
</html>
www:/var/www/org# /etc/init.d/nginx reload
Reloading nginx configuration: nginx.
```

//from terminal

nc 10.1.1.1 80

GET / HTTP/1.1

host: www.example.org:80

```
telem@debian:~$ nc 10.1.1.1 80
GET / HTTP/1.1
host: www.example.org:80

HTTP/1.1 200 OK
Server: nginx/1.17.8
Date: Thu, 10 Nov 2022 16:26:49 GMT
Content-Type: text/html
Content-Length: 339
Last-Modified: Thu, 10 Nov 2022 16:25:54 GMT
Connection: keep-alive
ETag: "636d2612-153"
Accept-Ranges: bytes

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <p>Hello form www.example.com<b>AVER SI FUNCIONA COÑ00</b>!!!!!!</p>
</head>
<body>

</body>
</html>
```

This configuration is very useful when there a huge traffic, so with this configuration the traffic is distributed better.

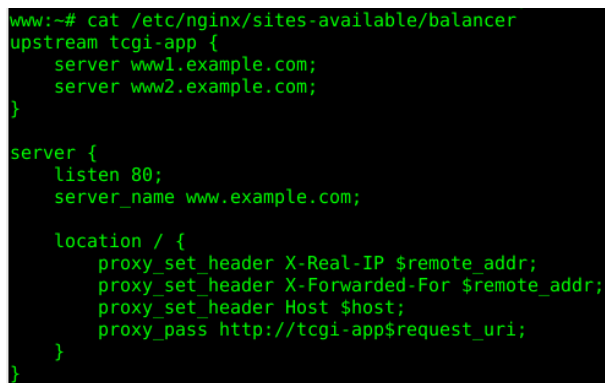
Exercise 1.5— Let's do something similar to what we've done in the last exercise, but taking a different approach.

Instead of modifying the DNS server, which is often a time-consuming and error-prone task, we are going to use a reverse proxy (i.e. this is what the nginx server actually is) to distribute the traffic to several web servers. In other words, we are going to build a load balancer.

1. In the machine www, go to the sites-available folder and open the balancer configuration file. Check it out and figure out what each code section does.

//from www

cat /etc/nginx/sites-available/balancer



```
www:~# cat /etc/nginx/sites-available/balancer
upstream tcgi-app {
    server www1.example.com;
    server www2.example.com;
}

server {
    listen 80;
    server_name www.example.com;

    location / {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header Host $host;
        proxy_pass http://tcgi-app$request_uri;
    }
}
```

We can observe in the balancer configuration, in the server section, balancer is attending requests of www.example.com from port 80. and pass those requests to [http://tcgi-app\\$...\(url request\)](http://tcgi-app$...(url request)).

And in the first line we can see the link is loaded in 2 servers. And by default, nginx balances the traffic using a round-robin approach, i.e., iteratively moves from the first server to the second and then to the third and so on, until all the servers are visited.

2. Still in the machine www, setup the nginx configuration so that only the balancer configuration is run. Reload the nginx server so that all the changes are applied. From a browser (either in the phyhost or in the host), try to access the url www.example.com. Is it working?

//from www

cd /etc/nginx/sites-enabled

rm www.example.org www.example.net www.example.com

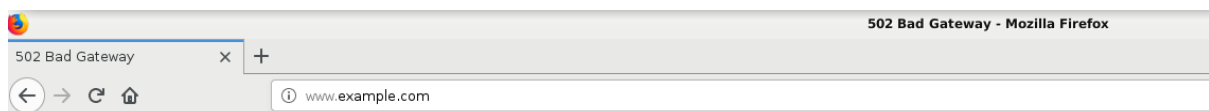
ln -s ../sites-available/balancer

/etc/init.d/nginx reload

```

www:/etc/nginx/sites-enabled# ls
www.example.com www.example.net www.example.org
www:/etc/nginx/sites-enabled#
www:/etc/nginx/sites-enabled# www:~# cat /etc/nginx/sites-available/balancer
www:/etc/nginx/sites-enabled# rm www.example.org www.example.net www.example.com
rm: remove symbolic link `www.example.org'? yorg www.example.net www.example.com
rm: remove symbolic link `www.example.net'? y
rm: remove symbolic link `www.example.com'? y
www:/etc/nginx/sites-enabled# ls
www:/etc/nginx/sites-enabled# ln -s ../sites-available/balancer
www:/etc/nginx/sites-enabled# ls
balancer
www:/etc/nginx/sites-enabled# /etc/init.d/nginx reload
Reloading nginx configuration: nginx.
www:/etc/nginx/sites-enabled#

```



502 Bad Gateway

nginx/1.17.8

It doesn't work, because the server of balancer ww1 and ww2 aren't active.

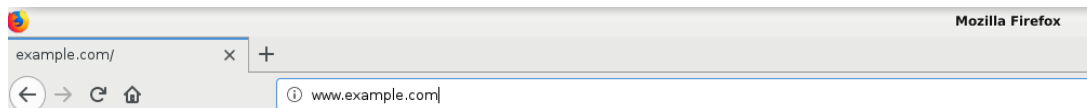
3. Start the nginx servers at the ww1 and ww2 machines. Try again to access the url www.example.com. Check the HTTP response received by the client. According to that response, who attended the request? Who actually attended the request? Analyze the data flow and discuss the results.

//from ww1

/etc/init.d/nginx start

//from ww2

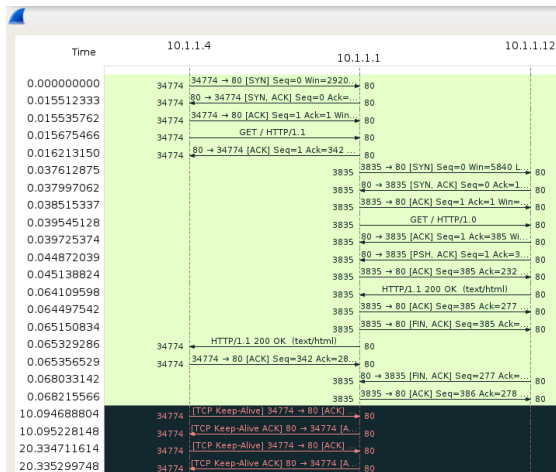
/etc/init.d/nginx start



It works!

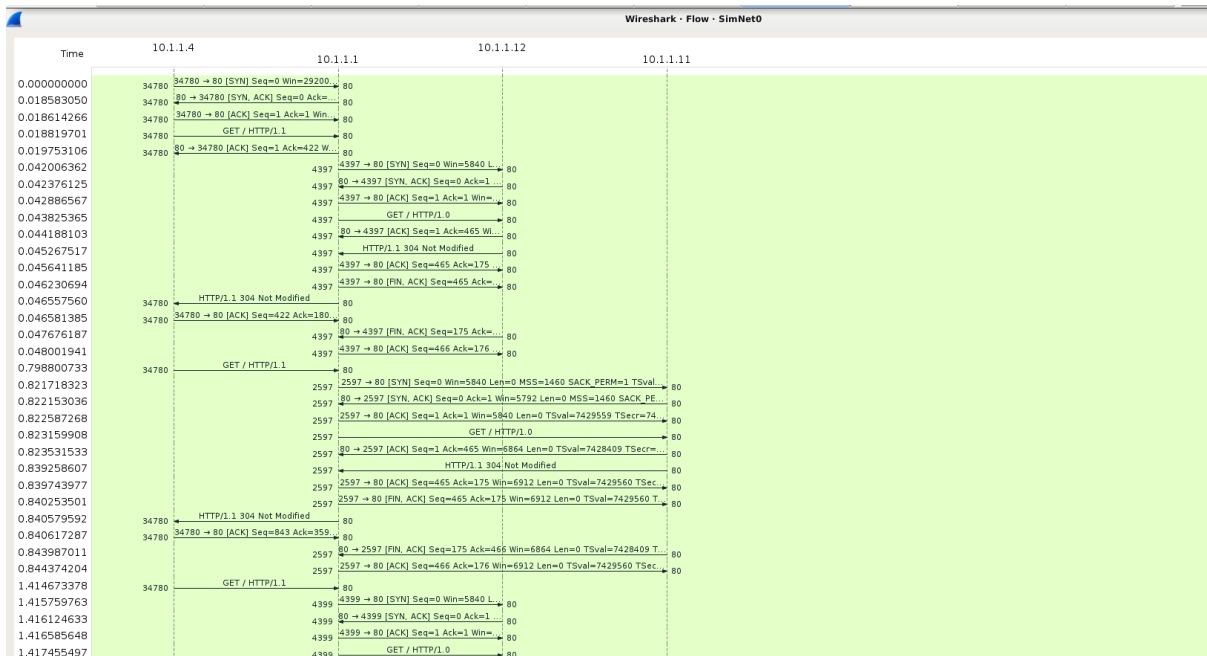
Capturing from SimNet0

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	10.1.1.4	10.1.1.1	TCP	74	34774 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=17343780 TSecr=0 WS=128
2	0.014937912	fe:fd:00:00:01:00	Broadcast	ARP	42	Who has 10.1.1.4? Tell 10.1.1.1
3	0.014957492	ba:5f:1f:c2:3d:31	fe:fd:00:00:01:00	ARP	42	10.1.1.4 is at ba:5f:1f:c2:3d:31
4	0.015512333	10.1.1.1	10.1.1.4	TCP	74	80 → 34774 [SYN, ACK] Seq=0 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=7387108 TSecr=17343780 WS=2
5	0.015535752	10.1.1.4	10.1.1.1	TCP	66	34774 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=17343784 TSecr=7387108
6	0.015675466	10.1.1.4	10.1.1.1	HTTP	407	GET / HTTP/1.1
7	0.016213150	10.1.1.1	10.1.1.4	TCP	66	80 → 34774 [ACK] Seq=1 Ack=342 Win=6864 Len=0 TSval=7387110 TSecr=17343784
8	0.037192295	fe:fd:00:00:01:00	Broadcast	ARP	42	Who has 10.1.1.12? Tell 10.1.1.1
9	0.037371213	fe:fd:00:00:03:00	fe:fd:00:00:01:00	ARP	42	10.1.1.12 is at fe:fd:00:00:03:00
10	0.037612875	10.1.1.1	10.1.1.12	TCP	74	3835 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=7387110 TSecr=0 WS=2
11	0.037997062	10.1.1.12	10.1.1.1	TCP	74	80 → 3835 [SYN, ACK] Seq=0 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=7387110 WS=2
12	0.038515337	10.1.1.1	10.1.1.12	TCP	66	3835 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSval=7387113 TSecr=7385034
13	0.039451128	10.1.1.1	10.1.1.12	HTTP	450	GET / HTTP/1.0
14	0.039725374	10.1.1.1	10.1.1.1	TCP	66	80 → 3835 [ACK] Seq=1 Ack=385 Win=6864 Len=0 TSval=7385034 TSecr=7387113
15	0.044872039	10.1.1.12	10.1.1.1	TCP	297	80 → 3835 [PSH, ACK] Seq=1 Ack=385 Win=6864 Len=231 TSval=7385034 TSecr=7387113 [TCP segment of a reassembled PDU]
16	0.045138824	10.1.1.1	10.1.1.12	TCP	66	3835 → 80 [ACK] Seq=385 Ack=232 Win=6912 Len=0 TSval=7387113 TSecr=7385034
17	0.064109598	10.1.1.12	10.1.1.1	HTTP	111	HTTP/1.1 200 OK (text/html)
18	0.064497542	10.1.1.1	10.1.1.12	TCP	66	3835 → 80 [ACK] Seq=385 Ack=277 Win=6912 Len=0 TSval=7387114 TSecr=7385036
19	0.065150834	10.1.1.1	10.1.1.12	TCP	66	3835 → 80 [FIN, ACK] Seq=385 Ack=277 Win=6912 Len=0 TSval=7387114 TSecr=7385036
20	0.065329286	10.1.1.1	10.1.1.4	HTTP	347	HTTP/1.1 200 OK (text/html)
21	0.065356529	10.1.1.1	10.1.1.1	TCP	66	34774 → 80 [ACK] Seq=342 Ack=282 Win=30336 Len=0 TSval=17343796 TSecr=7387114
22	0.068033142	10.1.1.12	10.1.1.1	TCP	66	80 → 3835 [FIN, ACK] Seq=277 Ack=386 Win=6864 Len=0 TSval=7385036 TSecr=7387114
23	0.068215566	10.1.1.1	10.1.1.12	TCP	66	3835 → 80 [ACK] Seq=386 Ack=278 Win=6912 Len=0 TSval=7387114 TSecr=7385036
24	5.230761598	ba:5f:1f:c2:3d:31	fe:fd:00:00:01:00	ARP	42	Who has 10.1.1.1? Tell 10.1.1.4
25	5.231369397	fe:fd:00:00:01:00	ba:5f:1f:c2:3d:31	ARP	42	10.1.1.1 is at fe:fd:00:00:01:00
26	10.094688804	10.1.1.4	10.1.1.1	TCP	66	[TCP keep-alive] 34774 → 80 [ACK] Seq=341 Ack=282 Win=30336 Len=0 TSval=17346304 TSecr=7387114
27	10.095228148	10.1.1.1	10.1.1.4	TCP	66	[TCP keep-alive ACK] 80 → 34774 [ACK] Seq=282 Ack=342 Win=6864 Len=0 TSval=7388118 TSecr=17343796



The request is attending by www, but www ask to www2 to response the request from the hosts. (10.1.1.1= www, 10.1.1.12=ww2).

4. Access again the url www.example.com. Do it several times. Discuss the results.



Now the request is forwarded to ww1 also. ww2 and ww1 attend requests alternating.

5. Let's simulate a service failure by stopping the nginx server at the www2 machine.

Try to access the url www.example.com from a browser (either in the phyhost or in the host). Do it several times. Check the nginx logs and discuss the results. Restart the nginx server at the www2 machine. Send a few requests and discuss the results.

//from ww2

/etc/init.d/nginx stop

Capturing from SimNet0					
No.	Time	Source	Destination	Protocol	Length/Info
22	10.315715832	10.1.1.4	10.1.1.1	TCP	66 34806 → 80 [ACK] Seq=316 Ack=282 Win=30336 Len=0 TSval=17644242 TSecr=7507293
23	10.402067541	10.1.1.4	10.1.1.1	HTTP	313 GET /favicon.ico HTTP/1.1
24	10.402744294	10.1.1.1	10.1.1.11	TCP	74 3844 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=7507302 TSecr=0 WS=2
25	10.402937307	10.1.1.11	10.1.1.1	TCP	74 80 → 3844 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=7506152 TSecr=7507302 WS=2
26	10.403128720	10.1.1.1	10.1.1.11	TCP	66 3844 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSval=7507302 TSecr=7506152
27	10.403579754	10.1.1.1	10.1.1.11	HTTP	356 GET /favicon.ico HTTP/1.0
28	10.40367242	10.1.1.11	10.1.1.1	TCP	66 80 → 3844 [ACK] Seq=1 Ack=291 Win=6864 Len=0 TSval=7506152 TSecr=7507302
29	10.404365227	10.1.1.11	10.1.1.1	HTTP	369 HTTP/1.1 404 Not Found (text/html)
30	10.404598424	10.1.1.1	10.1.1.11	TCP	66 3844 → 80 [ACK] Seq=291 Ack=304 Win=6912 Len=0 TSval=7507302 TSecr=7506152
31	10.404915634	10.1.1.1	10.1.1.11	TCP	66 3844 → 80 [FIN, ACK] Seq=291 Ack=304 Win=6912 Len=0 TSval=7507302 TSecr=7506152
32	10.405085596	10.1.1.1	10.1.1.4	HTTP	374 HTTP/1.1 404 Not Found (text/html)
33	10.405102944	10.1.1.4	10.1.1.1	TCP	66 34806 → 80 [ACK] Seq=563 Ack=590 Win=31360 Len=0 TSval=17644265 TSecr=7507302
34	10.405428844	10.1.1.11	10.1.1.1	TCP	66 80 → 3844 [FIN, ACK] Seq=304 Ack=292 Win=6864 Len=0 TSval=7506152 TSecr=7507302
35	10.405626580	10.1.1.1	10.1.1.11	TCP	66 3844 → 80 [ACK] Seq=292 Ack=305 Win=6912 Len=0 TSval=7507302 TSecr=7506152
36	11.690692877	10.1.1.4	10.1.1.1	HTTP	487 GET / HTTP/1.1
37	11.692375741	10.1.1.1	10.1.1.12	TCP	74 2288 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=7507431 TSecr=0 WS=2
38	11.692720296	10.1.1.12	10.1.1.1	TCP	54 80 → 2288 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
39	11.693584045	10.1.1.1	10.1.1.11	TCP	74 3846 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=7507431 TSecr=0 WS=2
40	11.693896882	10.1.1.11	10.1.1.1	TCP	74 80 → 3846 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=7506281 TSecr=7507431 WS=2
41	11.694274330	10.1.1.1	10.1.1.11	TCP	66 3846 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSval=7507431 TSecr=7506281
42	11.695187301	10.1.1.1	10.1.1.11	HTTP	530 GET / HTTP/1.0
43	11.695536999	10.1.1.11	10.1.1.1	TCP	66 80 → 3846 [ACK] Seq=1 Ack=465 Win=6864 Len=0 TSval=7506281 TSecr=7507431
44	11.696271499	10.1.1.11	10.1.1.1	HTTP	240 HTTP/1.1 304 Not Modified
45	11.696702800	10.1.1.1	10.1.1.1	TCP	66 80 → 3846 [FIN, ACK] Seq=175 Ack=465 Win=6864 Len=0 TSval=7506281 TSecr=7507431
46	11.697306103	10.1.1.1	10.1.1.11	TCP	66 3846 → 80 [ACK] Seq=465 Ack=175 Win=6912 Len=0 TSval=7507431 TSecr=7506281
47	11.697858206	10.1.1.1	10.1.1.11	TCP	66 3846 → 80 [FIN, ACK] Seq=465 Ack=176 Win=6912 Len=0 TSval=7507431 TSecr=7506281
48	11.698175247	10.1.1.11	10.1.1.1	TCP	66 80 → 3846 [ACK] Seq=176 Ack=466 Win=6864 Len=0 TSval=7506281 TSecr=7507431
49	11.698539315	10.1.1.1	10.1.1.4	HTTP	245 HTTP/1.1 304 Not Modified
50	11.698571956	10.1.1.4	10.1.1.1	TCP	66 34806 → 80 [ACK] Seq=984 Ack=769 Win=32512 Len=0 TSval=17644588 TSecr=7507431
51	12.796682191	10.1.1.4	10.1.1.1	HTTP	487 GET / HTTP/1.1
52	13.363346743	10.1.1.1	10.1.1.11	TCP	74 3847 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=7507431 TSecr=0 WS=2

At the beginning, the connection is the same as previous, host->www, www->ww1/2. But now, as the www2 is stop, we obtain RTS(disconnected), after that, the requests are attended by only www1.

6. The nginx reverse proxy can take several approaches on load balancing. One of them is to weight the balance, so that not all the servers receive the same amount of requests. This is achieved by typing weight=value next to a server forwarding indication, such as which indicates that www1 will receive twice the requests recieved by www2. Modify the balancer configuration in the www machine to make www1 machine attend the 75% of the web traffic addressed to the www machine. Check the results by requesting the index web page several times. Discuss the results.

//from www2

/etc/init.d/nginx start

//from www

nano /etc/nginx/sites-available/balancer

```

www.0
File Edit View Search Terminal Help

[ Wrote 17 lines ]

www:/etc/nginx/sites-enabled# cat /etc/nginx/sites-available/balancer
upstream tcgi-app {
    server www1.example.com weight=4;
    server www2.example.com weight=1;
}

server {
    listen 80;
    server_name www.example.com;

    location / {
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header Host $host;
        proxy_pass http://tcgi-app$request_uri;
    }
}

www:/etc/nginx/sites-enabled#

```

/etc/init.d/nginx reload

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.1.1.1	4778	10.1.1.11	80	12	1,442	6	762	6	680	0.004082	0.0053	1,141 k	1,018 k
10.1.1.1	4779	10.1.1.11	80	10	1,269	5	628	5	641	0.113986	0.0023	—	—
10.1.1.1	4780	10.1.1.11	80	10	1,314	5	802	5	512	1.136253	0.0047	—	—
10.1.1.1	4781	10.1.1.11	80	10	1,314	5	802	5	512	1.995992	0.0045	—	—
10.1.1.1	4782	10.1.1.11	80	10	1,314	5	802	5	512	2.725665	0.0053	1,219 k	778 k
10.1.1.1	3892	10.1.1.12	80	10	1,314	6	868	4	446	3.653683	0.0192	360 k	185 k
10.1.1.4	34814	10.1.1.1	80	37	6,009	21	3,640	16	2,369	0.000000	68.6763	424	275

As we can see, the traffic is distributed by their weight. the first 4th requests are attended by www1 and the last one is attended by www2.
(hay 5 de www1 pero deberían ser 4 xd).

Exercise 1.6— Now we are going to test an encrypted http connection, i.e. an https connection.

1. To do so, first we need to set up a certificate. Thus, build a self-signed certificate according to the instructions provided in the theory section. If you're in a hurry, you can use the testing certificate provided by the ssl package, located at:

[hacerlo !](#)

//from www

openssl genrsa -des3 -out mycakey.pem 2048 #to generate key

openssl req -new -x509 -days 2000 -key mycakey.pem -out mycacert.pem

#generate certificate

openssl x509 -in mycacert.pem -text -noout #check the certificate

```

generate keyinx/sites-available# openssl genrsa -des3 -out mycakey.pem 2048 #to g
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
Enter pass phrase for mycakey.pem:
Verifying - Enter pass phrase for mycakey.pem:
em -out mycacert.pem available# openssl req -new -x509 -days 2000 -key mycakey.p
Enter pass phrase for mycakey.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:spain
string is too long, it needs to be less than 2 bytes long
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:BARCELONA
Locality Name (eg, city) []:HOSPI
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UPC
Organizational Unit Name (eg, section) []:TELEM
Common Name (eg, YOUR name) []:Mclovin
Email Address []:mclovin@gmail.com

```

openssl genrsa -out myserverkey.pem 2048 #server key pair

chmod 400 myserverkey.pem #give permission cz its not encrypted

openssl req -new -key myserverkey.pem -out myservercert.csr #creat the certificate

(here in common name= www.example.com)

openssl x509 -req -in myservercert.csr -CA mycacert.pem -CAkey mycakey.pem

-CAcreateserial -days 360 -out myservercert.pem

#to sign the certificate

rm myservercert.csr #remove the certificate

2. Generate a new configuration for the https protocol in the virtualhost `www.example.com` served in the machine `www`. You can use the default configuration as the base for the new configuration. The key elements that should be included in the server section are:

Enable the new configuration and try to access to `https://www.example.com`. Remember to restart the nginx server each time you change any piece of configuration. Analyze the traffic exchanged between the browser and the server and comment the result

```
//from www
```

```
cd /etc/nginx/sites-available
```

```
cp default www.example.com
```

```
nano www.example.com
```

```
/etc/ssl/certs/ssl-cert-snakeoil.pem
```

```
/etc/ssl/private/ssl-cert-snakeoil.key
```

```
server {
    listen      433 ssl;
    server_name www.example.com;

    ssl_certificate      mycacert.pem;
    ssl_certificate_key  mycaker.pem;

    location / {
        root   /var/www;
        index  index.html index.htm;
    }

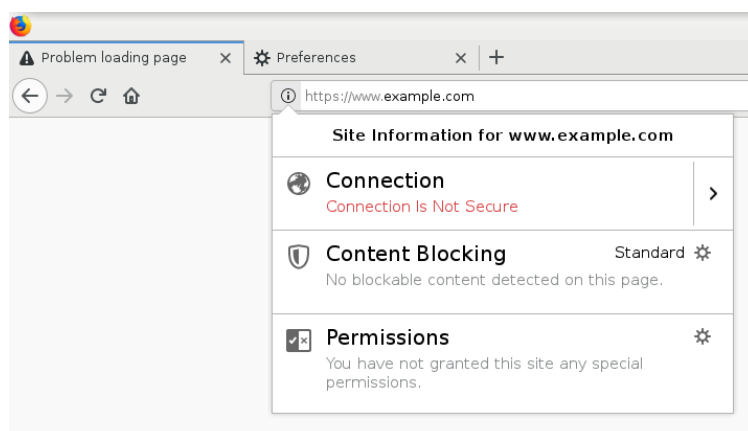
    location ~ ^/cgi/ {
        # Disable gzip (it makes scripts feel slower since they have to complet$
        # before getting gzipped)
    }
}
```

```
cd /etc/nginx/sites-enabled/
```

```
ln -s ../sites-available/www.example.com
```

```
www:/etc/nginx/sites-available# cd /etc/nginx/sites-enabled/
www:/etc/nginx/sites-enabled# ln -s ../sites-available/www.example.com
www:/etc/nginx/sites-enabled# ls
default  www.example.com
```

```
/etc/init.d/nginx start
```



Se debería conectar y en whireshark tendríamos q ver el protocolo TSL

3. Now we want to redirect all the requests received in the virtualhost `www.example.com`, from the protocol `http` to the protocol `https`. With that, we want to force that all the requests to the virtual host `www.example.com` are carried by an encrypted `https` connection instead of the regular `http` one, independently of what protocol the user typed in the browser when the request was made. To do so, first create the following configuration and name it as `https-redirection`:

Then disable the default configuration and enable the `https-redirection` one. From the host, generate a request to the `http://www.example.com` URL and describe the result. Describe the message exchange. Where does the redirection actually happen? Comment the results.

Exercise 1.7– Now we are going to test an `http2` connection. Get default `https` configuration and create a new one supporting the `http2` protocol. To support the `http2` protocol, just add the keyword `http2` next to the `ssl` keyword, as:

After that, enable the configuration and disable the regular `HTTPS` configuration. Restart the `nginx` server so that the changes become active, and send a request to the `https://www.example.com` URL. Check the captured data. How does the client inform the server that it supports the `HTTP2` protocol? Look at the `ALPN` section of the Client Hello TLS packet. Comment the results