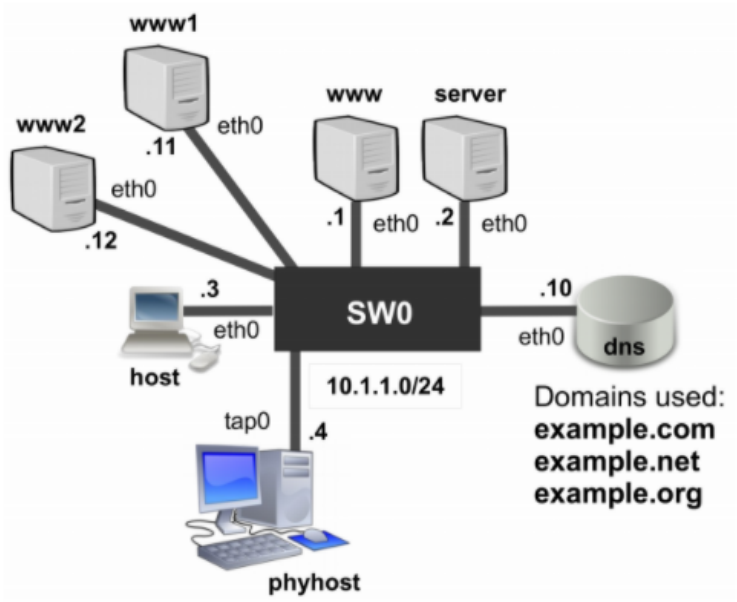


Exercise 1

```
phyhost$ simctl www-new start
```

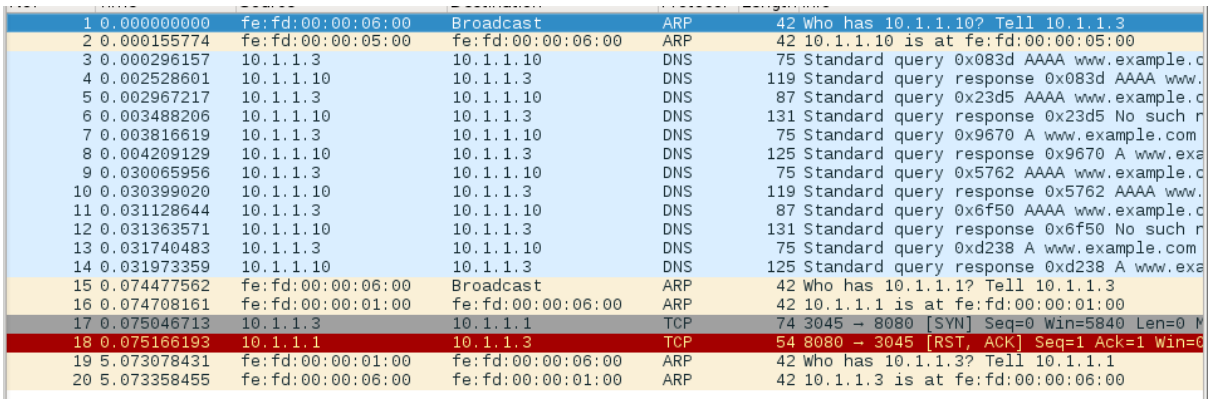
This scenario starts four virtual machines: host, www, server and dns. Each virtual machine has also two consoles (0 and 1) enabled.
First of all, we're going to have a look into the nginx configuration files. Remember that the nginx configuration files are stored in the /etc/nginx folder.

```
www:~# cd /etc/nginx/
www:/etc/nginx# ls
fastcgi.conf            mime.types           sites-available
fastcgi.conf.default    mime.types.default   sites-enabled
fastcgi_params          nginx.conf           uwsgi_params
fastcgi_params.default  nginx.conf.default   uwsgi_params.default
koi-utf                 scgi_params          win-utf
koi-win                 scgi_params.default
```

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.
host:~# lynx www.example.com:8080

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


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

lynx: Can't access startfile http://www.example.com:8080/
```

SimNet0:



1	0.000000000	fe:fd:00:00:06:00	Broadcast	ARP	42 Who has 10.1.1.10? Tell 10.1.1.3
2	0.000155774	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
3	0.000296157	10.1.1.3	10.1.1.10	DNS	75 Standard query 0x083d AAAA www.example.c
4	0.002528601	10.1.1.10	10.1.1.3	DNS	119 Standard query response 0x083d AAAA www.
5	0.002967217	10.1.1.3	10.1.1.10	DNS	87 Standard query 0x23d5 AAAA www.example.c
6	0.003488206	10.1.1.10	10.1.1.3	DNS	131 Standard query response 0x23d5 No such r
7	0.003816619	10.1.1.3	10.1.1.10	DNS	75 Standard query 0x9670 A www.example.com
8	0.004209129	10.1.1.10	10.1.1.3	DNS	125 Standard query response 0x9670 A www.exa
9	0.030065956	10.1.1.3	10.1.1.10	DNS	75 Standard query 0x5762 AAAA www.example.c
10	0.030399020	10.1.1.10	10.1.1.3	DNS	119 Standard query response 0x5762 AAAA www.
11	0.031128644	10.1.1.3	10.1.1.10	DNS	87 Standard query 0x6f50 AAAA www.example.c
12	0.031363571	10.1.1.10	10.1.1.3	DNS	131 Standard query response 0x6f50 No such r
13	0.031740483	10.1.1.3	10.1.1.10	DNS	75 Standard query 0xd238 A www.example.com
14	0.031973359	10.1.1.10	10.1.1.3	DNS	125 Standard query response 0xd238 A www.exa
15	0.074477562	fe:fd:00:00:06:00	Broadcast	ARP	42 Who has 10.1.1.1? Tell 10.1.1.3
16	0.074708161	fe:fd:00:00:01:00	fe:fd:00:00:06:00	ARP	42 10.1.1.1 is at fe:fd:00:00:01:00
17	0.075046713	10.1.1.3	10.1.1.1	TCP	74 3045 → 8080 [SYN] Seq=0 Win=5840 Len=0 M
18	0.075166193	10.1.1.1	10.1.1.3	TCP	54 8080 → 3045 [RST, ACK] Seq=1 Ack=1 Win=0
19	5.073078431	fe:fd:00:00:01:00	fe:fd:00:00:06:00	ARP	42 Who has 10.1.1.3? Tell 10.1.1.1
20	5.073358455	fe:fd:00:00:06:00	fe:fd:00:00:01:00	ARP	42 10.1.1.3 is at fe:fd:00:00:06:00

Which is the IP address associated to www.example.com? 10.1.1.1
Why the browser is not able to establish a TCP connection with the server?
Describe the DNS and TCP traffic captured.
DNS frames are correctly transmitted (10.1.1.3: host -> 10.1.1.10: dns server-> 10.1.1.1: www server) until the switch tries to reach, using TCP, the www server through port 8080.
So when the TCP petition is done, the server responds with an RST and ACK. It is not posible to connect with the port cause the one asigned to TCP is 80.

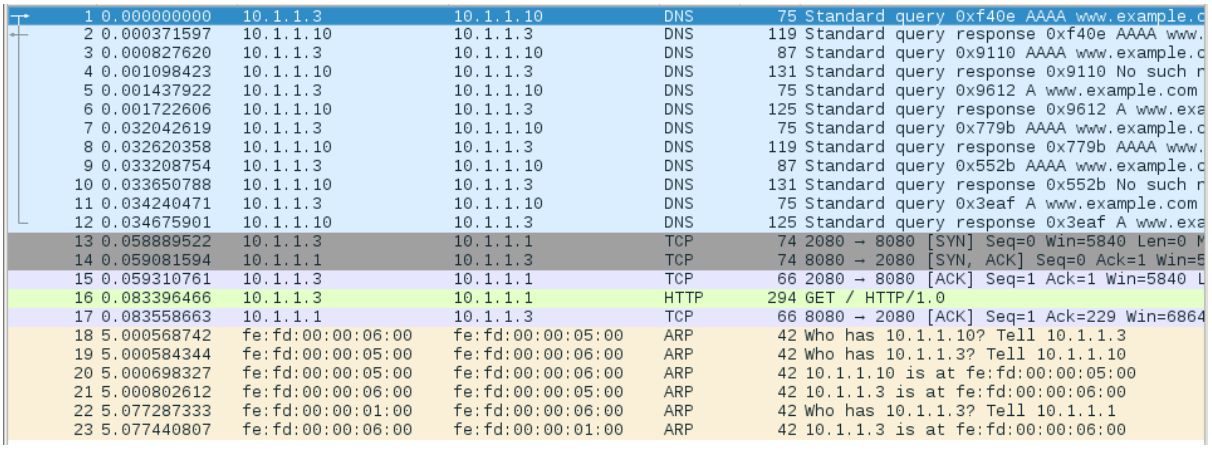
2) Capture the traffic on tap0 and repeat the previous experiment, but this time execute a netcat in the www machine listening on port 8080.
www:~# nc -l -p 8080

```
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
```

host:~# lynx www.example.com:8080

```
HTTP request sent; waiting for response.
```

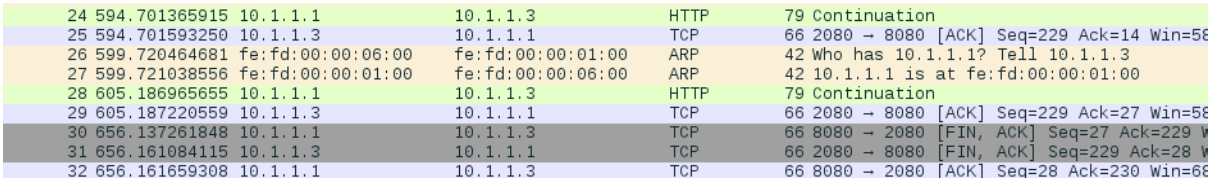
SimNet0:



1	0.000000000	10.1.1.3	10.1.1.10	DNS	75 Standard query 0xf40e AAAA www.example.c
2	0.000371597	10.1.1.10	10.1.1.3	DNS	119 Standard query response 0xf40e AAAA www.
3	0.000827620	10.1.1.3	10.1.1.10	DNS	87 Standard query 0x9110 AAAA www.example.c
4	0.001098423	10.1.1.10	10.1.1.3	DNS	131 Standard query response 0x9110 No such r
5	0.001437922	10.1.1.3	10.1.1.10	DNS	75 Standard query 0x9612 A www.example.com
6	0.001722606	10.1.1.10	10.1.1.3	DNS	125 Standard query response 0x9612 A www.exa
7	0.032042619	10.1.1.3	10.1.1.10	DNS	75 Standard query 0x779b AAAA www.example.c
8	0.032620358	10.1.1.10	10.1.1.3	DNS	119 Standard query response 0x779b AAAA www.
9	0.033208754	10.1.1.3	10.1.1.10	DNS	87 Standard query 0x552b AAAA www.example.c
10	0.033650788	10.1.1.10	10.1.1.3	DNS	131 Standard query response 0x552b No such r
11	0.034240471	10.1.1.3	10.1.1.10	DNS	75 Standard query 0x3eaf A www.example.com
12	0.034675901	10.1.1.10	10.1.1.3	DNS	125 Standard query response 0x3eaf A www.exa
13	0.058889522	10.1.1.3	10.1.1.1	TCP	74 2080 → 8080 [SYN] Seq=0 Win=5840 Len=0 M
14	0.059081594	10.1.1.1	10.1.1.3	TCP	74 8080 → 2080 [SYN, ACK] Seq=0 Ack=1 Win=5
15	0.059310761	10.1.1.3	10.1.1.1	TCP	66 2080 → 8080 [ACK] Seq=1 Ack=1 Win=5840 L
16	0.083396466	10.1.1.3	10.1.1.1	HTTP	294 GET / HTTP/1.0
17	0.083558663	10.1.1.1	10.1.1.3	TCP	66 8080 → 2080 [ACK] Seq=1 Ack=229 Win=6864
18	5.000568742	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
19	5.000584344	fe:fd:00:00:05:00	fe:fd:00:00:06:00	ARP	42 Who has 10.1.1.3? Tell 10.1.1.10
20	5.000698327	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
21	5.000802612	fe:fd:00:00:06:00	fe:fd:00:00:05:00	ARP	42 10.1.1.3 is at fe:fd:00:00:06:00
22	5.077287333	fe:fd:00:00:01:00	fe:fd:00:00:06:00	ARP	42 Who has 10.1.1.3? Tell 10.1.1.1
23	5.077440807	fe:fd:00:00:06:00	fe:fd:00:00:01:00	ARP	42 10.1.1.3 is at fe:fd:00:00:06:00

Which version of HTTP is using the browser? Version HTTP 1.0
Is the connection closed? The connection will be closed when the server finishes the tx.
Describe the DNS and HTTP traffic and kill the netcat to finish.

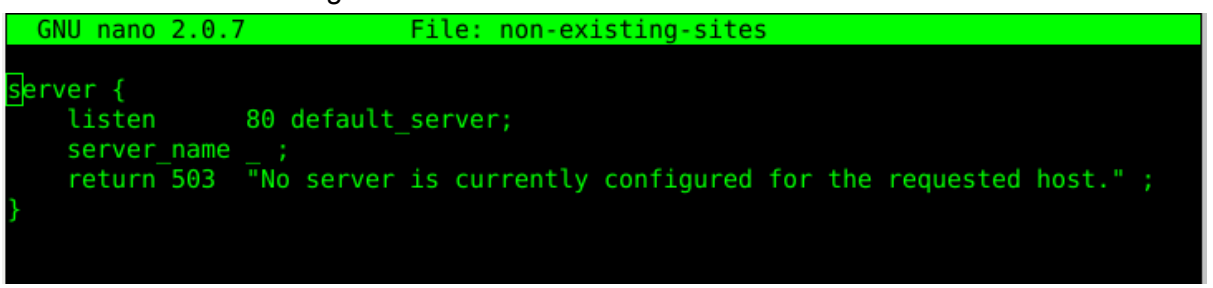
The traffic DNS is the expected. We can see the TCP traffic to establish the connection (SYN; SYN-ACK; ACK) and the ones which close it (FIN-ACK; FIN-ACK; ACK).



3) Let's take a look at the nginx web server in the www machine. How many websites are available? Is there any website enabled?
www:~# cd /etc/nginx/sites-available-> default, non-exististing-sites, balancer
www:~# nano default -> www.example.com-> index.html
www:~# cd /etc/nginx/sites-enabled-> default

4) Open the default configuration. What is the virtual host name(s) for this site? Where is the website content placed?
www:~# nano default
host(s): www.example.com
location: /var/www

5) Open the non-existing-sites configuration. What do you think the purpose of this site configuration is?
www:~# cd /etc/nginx/sites-enabled
www:~# nano non-existing-files



The purpose is to configure an error message 503 for the non existing sites.

6) Enable the non-existing-sites by typing, from the sites-enabled folder, the following command:
www:~# cd /etc/nginx/sites-enabled

```
ln -s ../sites-available/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 "/".
host:~# nc www.example.com 80
ó
host:~# nc 10.1.1.1 80
GET / HTTP/1.0
Which response do you obtain? Describe the HTTP traffic captured for the GET request.

```
host:~# nc www.example.com 80
GET / HTTP/1.0

HTTP/1.1 200 OK
Server: nginx/1.17.8
Date: Mon, 12 Apr 2021 15:55:41 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>
```

SimNet0:

1	0.000000000	10.1.1.3	10.1.1.10	DNS	75	Standard query 0x4aa9 A www.example.com
2	0.000344845	10.1.1.10	10.1.1.3	DNS	125	Standard query response 0x4aa9 A www.exa
3	0.003604948	10.1.1.3	10.1.1.1	TCP	74	3156 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS
4	0.003700255	10.1.1.1	10.1.1.3	TCP	74	80 → 3156 [SYN, ACK] Seq=0 Ack=1 Win=579
5	0.003793440	10.1.1.3	10.1.1.1	TCP	66	3156 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len
6	4.967406222	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
7	4.967740284	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
8	9.196463938	10.1.1.3	10.1.1.1	TCP	81	3156 → 80 [PSH, ACK] Seq=1 Ack=1 Win=584
9	9.196679676	10.1.1.1	10.1.1.3	TCP	66	80 → 3156 [ACK] Seq=1 Ack=16 Win=5792 Le
10	39.129920663	10.1.1.3	10.1.1.1	HTTP	67	GET / HTTP/1.0
11	39.130069812	10.1.1.1	10.1.1.3	TCP	66	80 → 3156 [ACK] Seq=1 Ack=17 Win=5792 Le
12	39.130376173	10.1.1.1	10.1.1.3	TCP	297	80 → 3156 [PSH, ACK] Seq=1 Ack=17 Win=57
13	39.130474002	10.1.1.3	10.1.1.1	TCP	66	3156 → 80 [ACK] Seq=17 Ack=232 Win=6912
14	39.130802356	10.1.1.1	10.1.1.3	HTTP	111	HTTP/1.1 200 OK (text/html)
15	39.130898233	10.1.1.3	10.1.1.1	TCP	66	3156 → 80 [ACK] Seq=17 Ack=277 Win=6912
16	39.131432901	10.1.1.1	10.1.1.3	TCP	66	80 → 3156 [FIN, ACK] Seq=277 Ack=17 Win=
17	39.132175076	10.1.1.3	10.1.1.1	TCP	66	3156 → 80 [FIN, ACK] Seq=17 Ack=278 Win=
18	39.132276763	10.1.1.1	10.1.1.3	TCP	66	80 → 3156 [ACK] Seq=278 Ack=18 Win=5792

The traffic HTTP is the petition from the host and the "OK" response from the server.

8) Now edit the default configuration and remove the "" from the server_name section.

```
 GNU nano 2.0.7                File: 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;
```

Reload nginx conf:

www:~# /etc/init.d/nginx reload
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? default and non-existing
The response is a 503 code (the one for non-existing-sites)

```
host:~# nc www.example.com 80
GET / HTTP/1.0

HTTP/1.1 503 Service Temporarily Unavailable
Server: nginx/1.17.8
Date: Tue, 20 Apr 2021 10:55:25 GMT
Content-Type: application/octet-stream
Content-Length: 57
Connection: close

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

Thats cause we should put the host’s name after the get. But theres nothing so the server uses the non-existing-sites file. In the previous case as we had “” in the hosts names it is not needed to put anything else.

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.
host:~# nc www.home.com 80

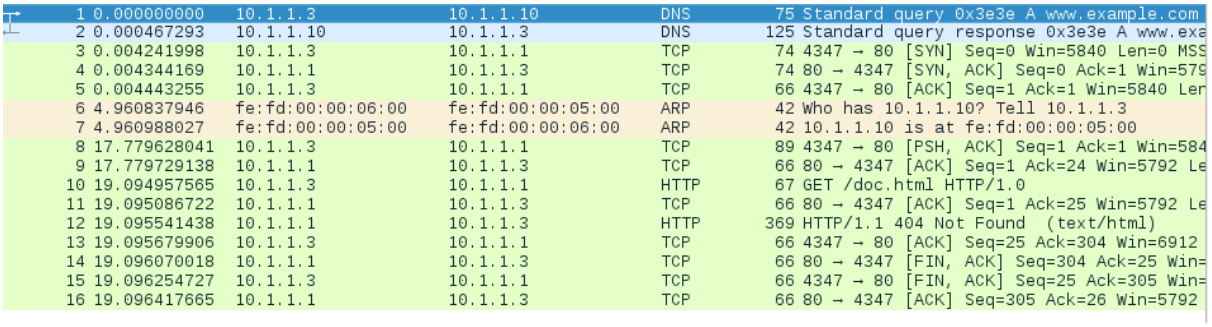
```
host:~# nc www.example.com 80
GET / HTTP/1.1
host: www.home.com

HTTP/1.1 503 Service Temporarily Unavailable
Server: nginx/1.17.8
Date: Tue, 20 Apr 2021 11:03:33 GMT
Content-Type: application/octet-stream
Content-Length: 57
Connection: keep-alive

No server is currently configured for the requested host.
```

Obviously we receive an 503 code cause the host is not in the conf.

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.
www:~# /etc/init.d/apache2 start
host:~# nc www.example.com 80
GET /doc.html HTTP/1.0

```
host:~# nc www.example.com 80
GET /doc.html HTTP/1.0

HTTP/1.1 404 Not Found
Server: nginx/1.17.8
Date: Mon, 12 Apr 2021 16:29:29 GMT
Content-Type: text/html
Content-Length: 153
Connection: close

<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.17.8</center>
</body>
</html>
```

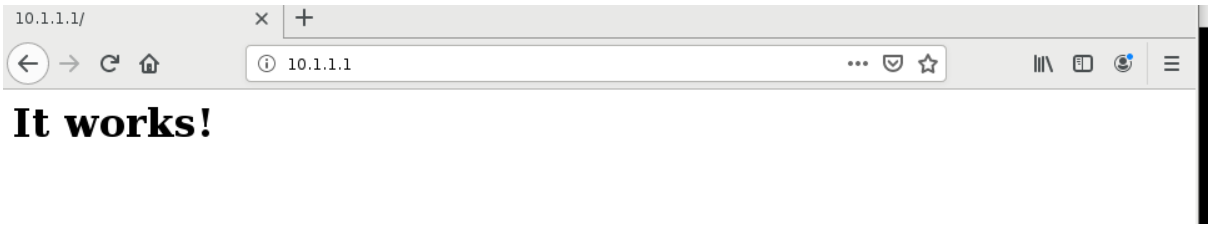
SimNet0:

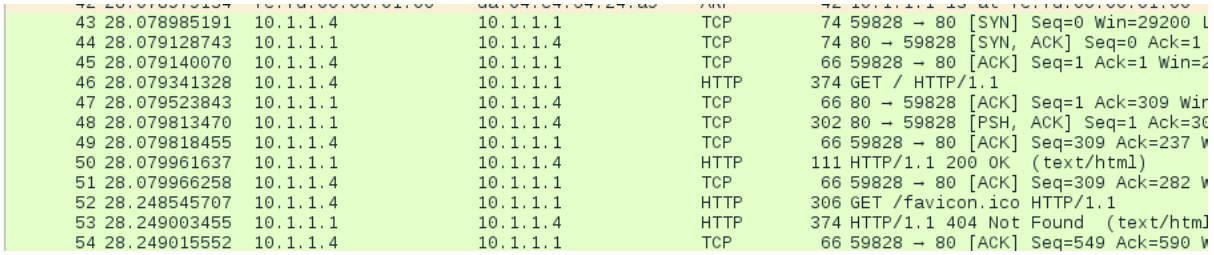


1	0.000000000	10.1.1.3	10.1.1.10	DNS	75 Standard query 0x3e3e A www.example.com
2	0.000467293	10.1.1.10	10.1.1.3	DNS	125 Standard query response 0x3e3e A www.exa
3	0.004241998	10.1.1.3	10.1.1.1	TCP	74 4347 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS
4	0.004344169	10.1.1.1	10.1.1.3	TCP	74 80 → 4347 [SYN, ACK] Seq=0 Ack=1 Win=579
5	0.004443255	10.1.1.3	10.1.1.1	TCP	66 4347 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len
6	4.960837946	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
7	4.960988027	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
8	17.779628041	10.1.1.3	10.1.1.1	TCP	89 4347 → 80 [PSH, ACK] Seq=1 Ack=1 Win=584
9	17.779729138	10.1.1.1	10.1.1.3	TCP	66 80 → 4347 [ACK] Seq=1 Ack=24 Win=5792 Le
10	19.094957565	10.1.1.3	10.1.1.1	HTTP	67 GET /doc.html HTTP/1.0
11	19.095086722	10.1.1.1	10.1.1.3	TCP	66 80 → 4347 [ACK] Seq=1 Ack=25 Win=5792 Le
12	19.095541438	10.1.1.1	10.1.1.3	HTTP	369 HTTP/1.1 404 Not Found (text/html)
13	19.095679906	10.1.1.3	10.1.1.1	TCP	66 4347 → 80 [ACK] Seq=25 Ack=304 Win=6912
14	19.096070018	10.1.1.1	10.1.1.3	TCP	66 80 → 4347 [FIN, ACK] Seq=304 Ack=25 Win=
15	19.096254727	10.1.1.3	10.1.1.1	TCP	66 4347 → 80 [FIN, ACK] Seq=25 Ack=305 Win=
16	19.096417665	10.1.1.1	10.1.1.3	TCP	66 80 → 4347 [ACK] Seq=305 Ack=26 Win=5792

There is no resource like that in the server-> 404 Not Found

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.
phyhost:~# sudo ifconfig SimNet0 10.1.1.4/24 -> firefox &
10.1.1.1/ -> It works!



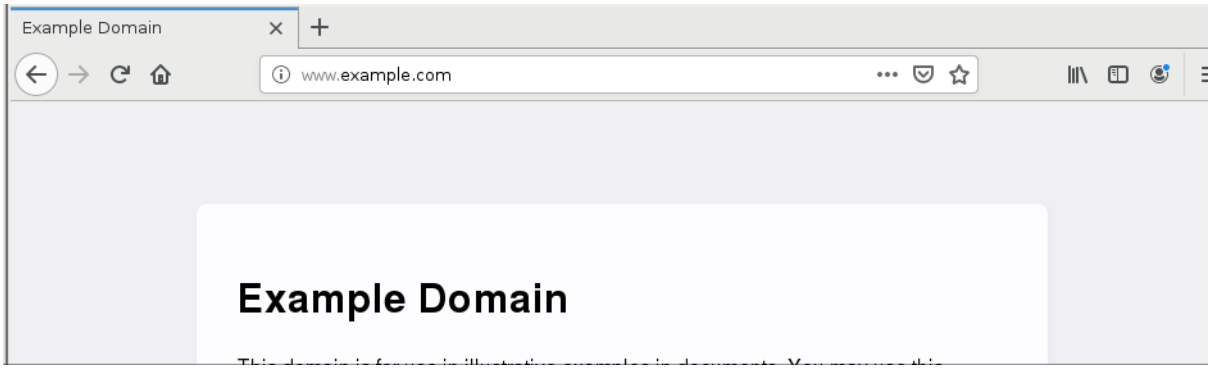


43	28.078985191	10.1.1.4	10.1.1.1	TCP	74 59828 → 80 [SYN] Seq=0 Win=29200 L
44	28.079128743	10.1.1.1	10.1.1.4	TCP	74 80 → 59828 [SYN, ACK] Seq=0 Ack=1
45	28.079140070	10.1.1.4	10.1.1.1	TCP	66 59828 → 80 [ACK] Seq=1 Ack=1 Win=2
46	28.079341328	10.1.1.4	10.1.1.1	HTTP	374 GET / HTTP/1.1
47	28.079523843	10.1.1.1	10.1.1.4	TCP	66 80 → 59828 [ACK] Seq=1 Ack=309 Wir
48	28.079813470	10.1.1.1	10.1.1.4	TCP	302 80 → 59828 [PSH, ACK] Seq=1 Ack=30
49	28.079818455	10.1.1.4	10.1.1.1	TCP	66 59828 → 80 [ACK] Seq=309 Ack=237 W
50	28.079961637	10.1.1.1	10.1.1.4	HTTP	111 HTTP/1.1 200 OK (text/html)
51	28.079966258	10.1.1.4	10.1.1.1	TCP	66 59828 → 80 [ACK] Seq=309 Ack=282 W
52	28.248545707	10.1.1.4	10.1.1.1	HTTP	306 GET /favicon.ico HTTP/1.1
53	28.249003455	10.1.1.1	10.1.1.4	HTTP	374 HTTP/1.1 404 Not Found (text/html
54	28.249015552	10.1.1.4	10.1.1.1	TCP	66 59828 → 80 [ACK] Seq=549 Ack=590 W

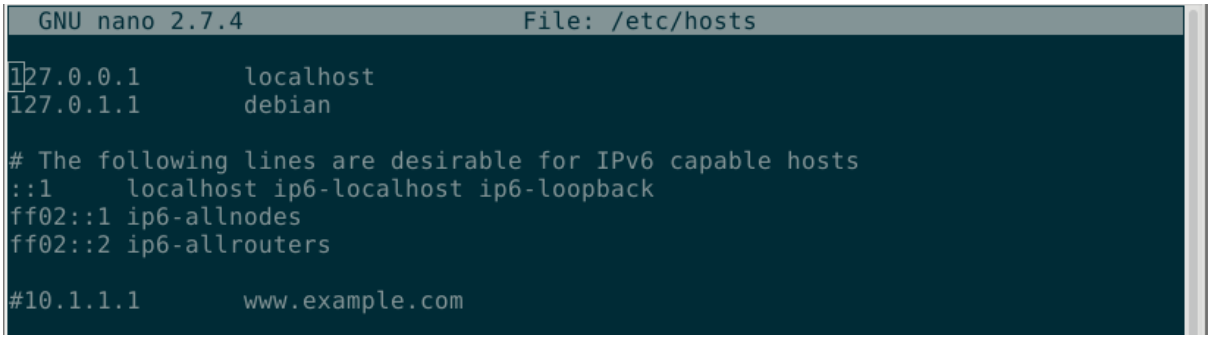
10.1.1.1/doc.html -> 404 Not Found


```
    71 39.782355811  10.1.1.4          10.1.1.1          HTTP       382 GET /doc.html HTTP/1.1
    72 39.782842886  10.1.1.1          10.1.1.4          HTTP       374 HTTP/1.1 404 Not Found  (text/html
    73 39.782856581  10.1.1.4          10.1.1.1          TCP         66 59828 → 80 [ACK] Seq=865 Ack=898 W
```

Can you use the name www.example.com from the phyhost? why? No.



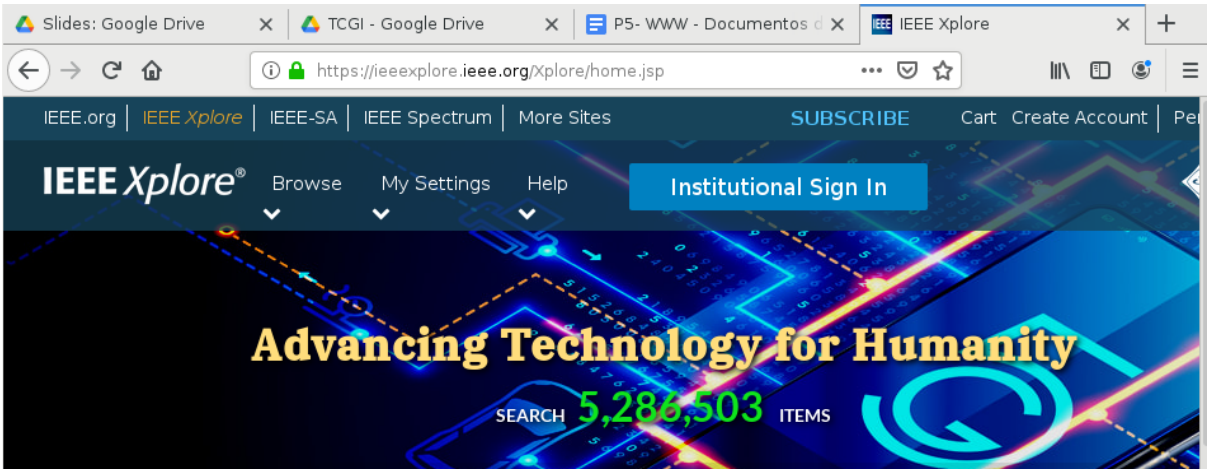
Propose a way to reach the www machine when typing www.example.com.
We should put the host in the file /etc/hosts from the phyhost.-> Uncomment last line.



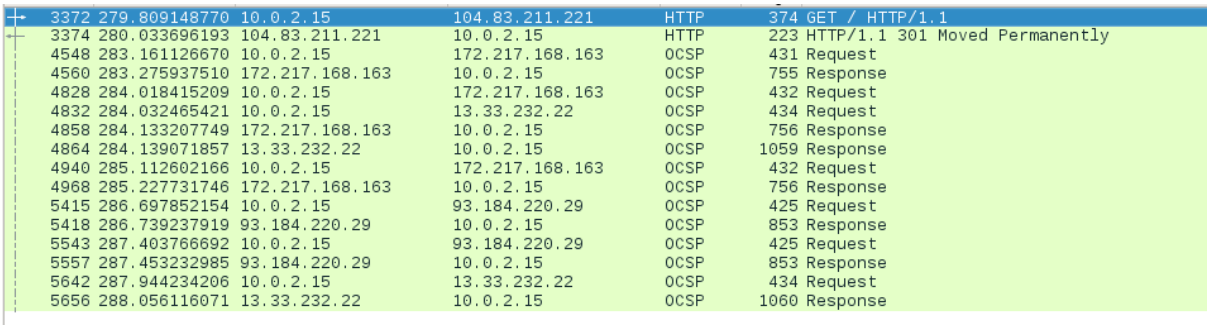
Exercise 2

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**.
   Describe the HTTP traffic captured. In particular, discuss the GET requests that you observe and the number of connections.



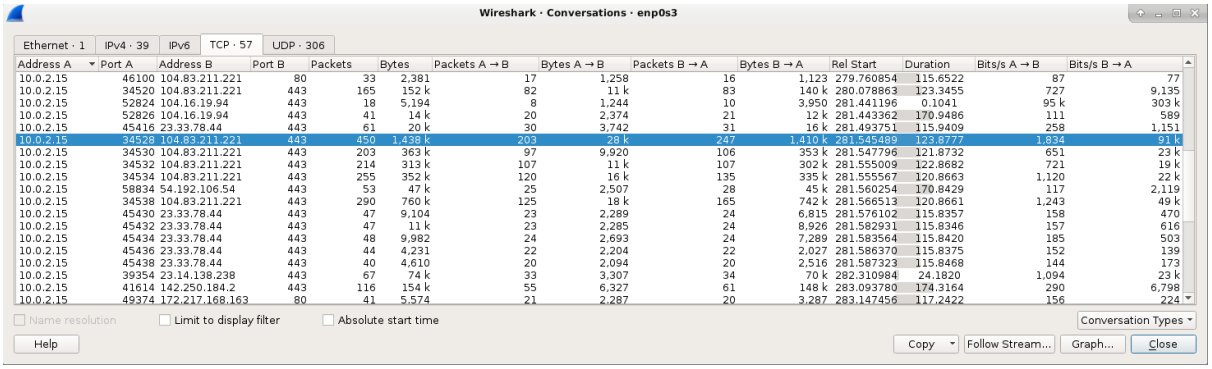
enp0s3 (visualization filter: http)



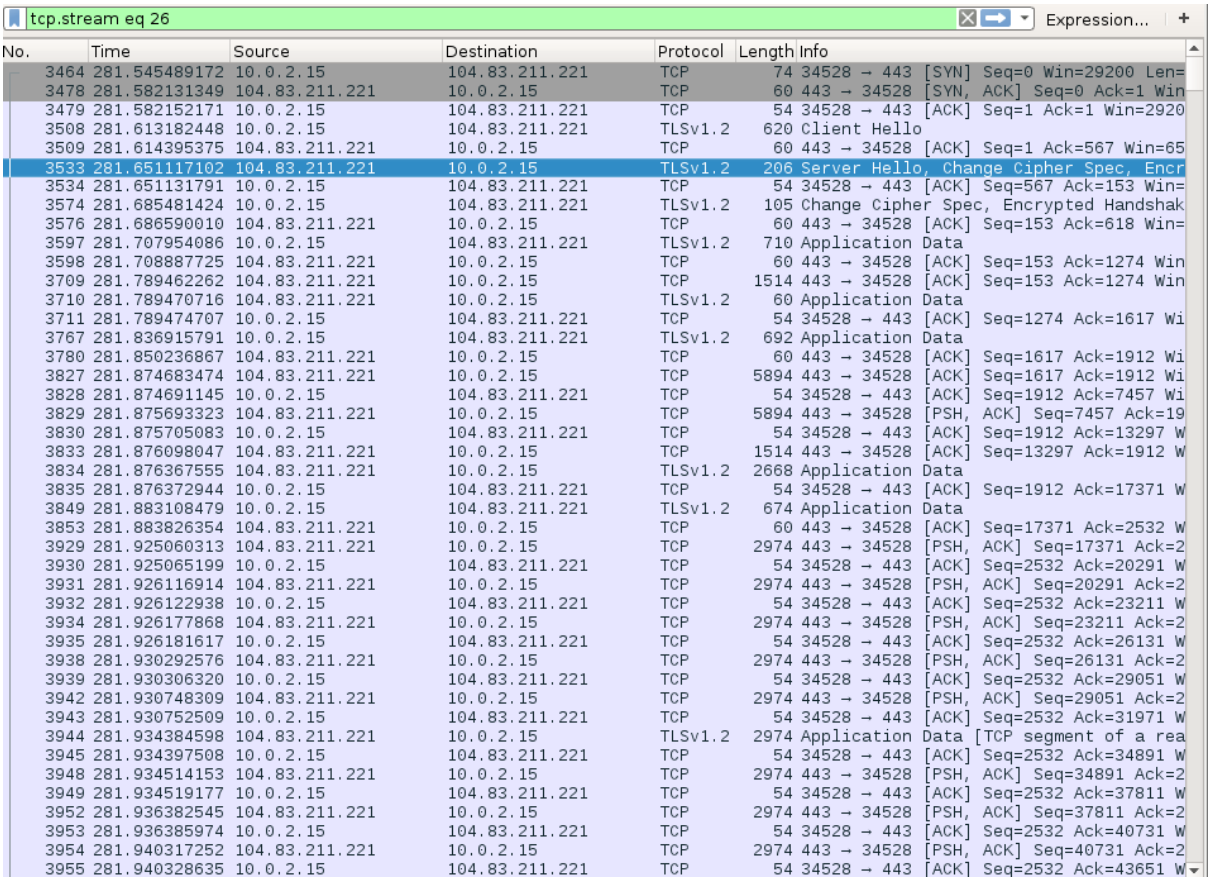
3372	279.809148770	10.0.2.15	104.83.211.221	HTTP	374 GET / HTTP/1.1
3374	280.033696193	104.83.211.221	10.0.2.15	HTTP	223 HTTP/1.1 301 Moved Permanently
4548	283.161126670	10.0.2.15	172.217.168.163	OCSP	431 Request
4560	283.275937510	172.217.168.163	10.0.2.15	OCSP	755 Response
4828	284.018415209	10.0.2.15	172.217.168.163	OCSP	432 Request
4832	284.032465421	10.0.2.15	13.33.232.22	OCSP	434 Request
4858	284.133207749	172.217.168.163	10.0.2.15	OCSP	756 Response
4864	284.139071857	13.33.232.22	10.0.2.15	OCSP	1059 Response
4940	285.112602166	10.0.2.15	172.217.168.163	OCSP	432 Request
4968	285.227731746	172.217.168.163	10.0.2.15	OCSP	756 Response
5415	286.697852154	10.0.2.15	93.184.220.29	OCSP	425 Request
5418	286.739237919	93.184.220.29	10.0.2.15	OCSP	853 Response
5543	287.403766692	10.0.2.15	93.184.220.29	OCSP	425 Request
5557	287.453232985	93.184.220.29	10.0.2.15	OCSP	853 Response
5642	287.944234206	10.0.2.15	13.33.232.22	OCSP	434 Request
5656	288.056116071	13.33.232.22	10.0.2.15	OCSP	1060 Response

To do this analysis easier, you can take one or combine both of the following approaches:

- 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.



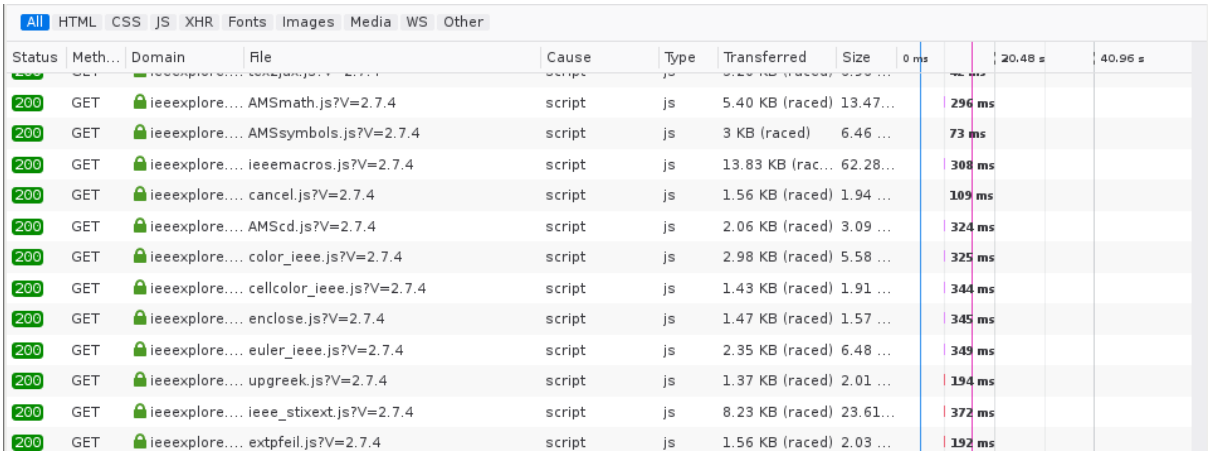
Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.2.15	46100	104.83.211.221	80	33	2,381	17	1,258	16	1,123	279.760854	115.6522	87	77
10.0.2.15	34520	104.83.211.221	443	165	152 k	82	11 k	83	140 k	280.078863	123.3455	727	9,135
10.0.2.15	52824	104.16.19.94	443	18	5,194	8	1,244	10	3,950	281.441196	0.1041	95 k	303 k
10.0.2.15	52826	104.16.19.94	443	41	14 k	20	2,374	21	12 k	281.443362	170.9486	111	589
10.0.2.15	45416	23.33.78.44	443	61	20 k	30	3,742	31	16 k	281.493751	115.9409	258	1,151
10.0.2.15	34528	104.83.211.221	443	450	1,438 k	203	28 k	247	1,410 k	281.545489	123.8777	1,834	91 k
10.0.2.15	34530	104.83.211.221	443	203	363 k	97	9,920	106	353 k	281.547796	121.8732	651	23 k
10.0.2.15	34532	104.83.211.221	443	214	313 k	107	11 k	107	302 k	281.555009	122.8682	721	19 k
10.0.2.15	34534	104.83.211.221	443	255	352 k	120	16 k	135	335 k	281.555567	120.8663	1,120	22 k
10.0.2.15	58834	54.192.106.54	443	53	47 k	25	2,507	28	45 k	281.560254	170.8429	117	2,119
10.0.2.15	34538	104.83.211.221	443	290	760 k	125	18 k	165	742 k	281.566513	120.8661	1,243	49 k
10.0.2.15	45430	23.33.78.44	443	47	9,104	23	2,289	24	6,815	281.576102	115.8357	158	470
10.0.2.15	45432	23.33.78.44	443	47	11 k	23	2,285	24	8,926	281.582931	115.8346	157	616
10.0.2.15	45434	23.33.78.44	443	48	9,982	24	2,693	24	7,289	281.583564	115.8420	185	503
10.0.2.15	45436	23.33.78.44	443	44	4,231	22	2,204	22	2,027	281.586370	115.8375	152	139
10.0.2.15	45438	23.33.78.44	443	40	4,610	20	2,094	20	2,516	281.587323	115.8468	144	173
10.0.2.15	39354	23.14.138.238	443	67	74 k	33	3,307	34	70 k	282.310984	24.1820	1,094	23 k
10.0.2.15	41614	142.250.184.2	443	116	154 k	55	6,327	61	148 k	283.093780	174.3164	290	6,798
10.0.2.15	49374	172.217.168.163	80	41	5,574	21	2,287	20	3,287	283.147456	117.2422	156	224



tcp.stream eq 26 Expression... +

No.	Time	Source	Destination	Protocol	Length	Info
3464	281.545489172	10.0.2.15	104.83.211.221	TCP	74	34528 → 443 [SYN] Seq=0 Win=29200 Len=
3478	281.582131349	104.83.211.221	10.0.2.15	TCP	60	443 → 34528 [SYN, ACK] Seq=0 Ack=1 Win
3479	281.582152171	10.0.2.15	104.83.211.221	TCP	54	34528 → 443 [ACK] Seq=1 Ack=1 Win=2920
3508	281.613182448	10.0.2.15	104.83.211.221	TLSv1.2	620	Client Hello
3509	281.614395375	104.83.211.221	10.0.2.15	TCP	60	443 → 34528 [ACK] Seq=1 Ack=567 Win=65
3533	281.651117102	104.83.211.221	10.0.2.15	TLSv1.2	206	Server Hello, Change Cipher Spec, Encr
3534	281.651131791	10.0.2.15	104.83.211.221	TCP	54	34528 → 443 [ACK] Seq=567 Ack=153 Win=
3574	281.685481424	10.0.2.15	104.83.211.221	TLSv1.2	105	Change Cipher Spec, Encrypted Handshak
3576	281.686590010	104.83.211.221	10.0.2.15	TCP	60	443 → 34528 [ACK] Seq=153 Ack=618 Win=
3597	281.707954086	10.0.2.15	104.83.211.221	TLSv1.2	710	Application Data
3598	281.708887725	104.83.211.221	10.0.2.15	TCP	60	443 → 34528 [ACK] Seq=153 Ack=1274 Win
3709	281.789462262	104.83.211.221	10.0.2.15	TCP	1514	443 → 34528 [ACK] Seq=153 Ack=1274 Win
3710	281.789470716	104.83.211.221	10.0.2.15	TLSv1.2	60	Application Data
3711	281.789474707	10.0.2.15	104.83.211.221	TCP	54	34528 → 443 [ACK] Seq=1274 Ack=1617 Wi
3767	281.836915791	10.0.2.15	104.83.211.221	TLSv1.2	692	Application Data
3780	281.850236867	104.83.211.221	10.0.2.15	TCP	60	443 → 34528 [ACK] Seq=1617 Ack=1912 Wi
3827	281.874683474	104.83.211.221	10.0.2.15	TCP	5894	443 → 34528 [ACK] Seq=1617 Ack=1912 Wi
3828	281.874691145	10.0.2.15	104.83.211.221	TCP	54	34528 → 443 [ACK] Seq=1912 Ack=7457 Wi
3829	281.875693323	104.83.211.221	10.0.2.15	TCP	5894	443 → 34528 [PSH, ACK] Seq=7457 Ack=19
3830	281.875705083	10.0.2.15	104.83.211.221	TCP	54	34528 → 443 [ACK] Seq=1912 Ack=13297 W
3833	281.876098047	104.83.211.221	10.0.2.15	TCP	1514	443 → 34528 [ACK] Seq=13297 Ack=1912 W
3834	281.876367555	104.83.211.221	10.0.2.15	TLSv1.2	2668	Application Data
3835	281.876372944	10.0.2.15	104.83.211.221	TCP	54	34528 → 443 [ACK] Seq=1912 Ack=17371 W
3849	281.883108479	10.0.2.15	104.83.211.221	TLSv1.2	674	Application Data
3853	281.883826354	104.83.211.221	10.0.2.15	TCP	60	443 → 34528 [ACK] Seq=17371 Ack=2532 W
3929	281.925060313	104.83.211.221	10.0.2.15	TCP	2974	443 → 34528 [PSH, ACK] Seq=17371 Ack=2
3930	281.925065199	10.0.2.15	104.83.211.221	TCP	54	34528 → 443 [ACK] Seq=2532 Ack=20291 W
3931	281.926116914	104.83.211.221	10.0.2.15	TCP	2974	443 → 34528 [PSH, ACK] Seq=20291 Ack=2
3932	281.926122938	10.0.2.15	104.83.211.221	TCP	54	34528 → 443 [ACK] Seq=2532 Ack=23211 W
3934	281.926177868	104.83.211.221	10.0.2.15	TCP	2974	443 → 34528 [PSH, ACK] Seq=23211 Ack=2
3935	281.926181617	10.0.2.15	104.83.211.221	TCP	54	34528 → 443 [ACK] Seq=2532 Ack=26131 W
3938	281.930292576	104.83.211.221	10.0.2.15	TCP	2974	443 → 34528 [PSH, ACK] Seq=26131 Ack=2
3939	281.930306320	10.0.2.15	104.83.211.221	TCP	54	34528 → 443 [ACK] Seq=2532 Ack=29051 W
3942	281.930748309	104.83.211.221	10.0.2.15	TCP	2974	443 → 34528 [PSH, ACK] Seq=29051 Ack=2
3943	281.930752509	10.0.2.15	104.83.211.221	TCP	54	34528 → 443 [ACK] Seq=2532 Ack=31971 W
3944	281.934384598	104.83.211.221	10.0.2.15	TLSv1.2	2974	Application Data [TCP segment of a rea
3945	281.934397508	10.0.2.15	104.83.211.221	TCP	54	34528 → 443 [ACK] Seq=2532 Ack=34891 W
3948	281.934514153	104.83.211.221	10.0.2.15	TCP	2974	443 → 34528 [PSH, ACK] Seq=34891 Ack=2
3949	281.934519177	10.0.2.15	104.83.211.221	TCP	54	34528 → 443 [ACK] Seq=2532 Ack=37811 W
3952	281.936382545	104.83.211.221	10.0.2.15	TCP	2974	443 → 34528 [PSH, ACK] Seq=37811 Ack=2
3953	281.936385974	10.0.2.15	104.83.211.221	TCP	54	34528 → 443 [ACK] Seq=2532 Ack=40731 W
3954	281.940317252	104.83.211.221	10.0.2.15	TCP	2974	443 → 34528 [PSH, ACK] Seq=40731 Ack=2
3955	281.940328635	10.0.2.15	104.83.211.221	TCP	54	34528 → 443 [ACK] Seq=2532 Ack=43651 W

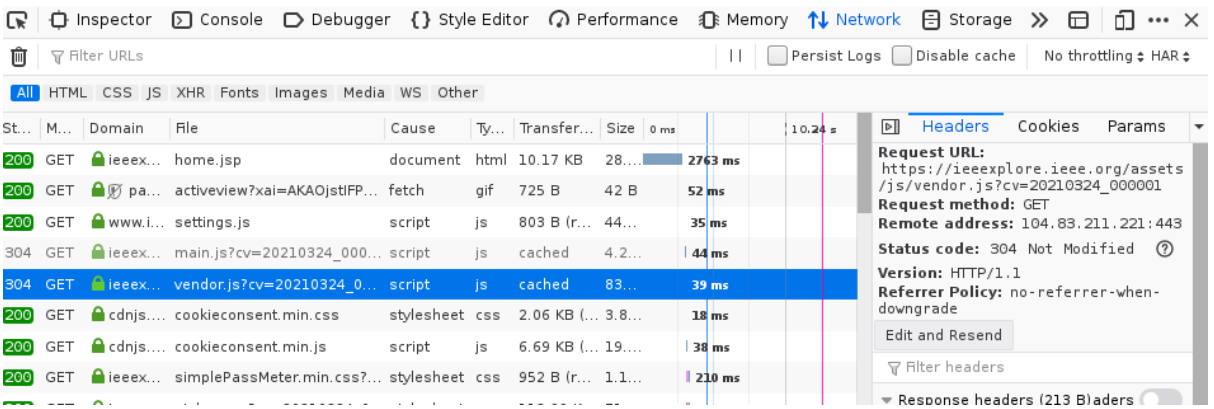
- Open the developer tools (e.g. push **control+shift+i**). A new section will show up in the bottom of the browser. There, one can analyze any matter related with the browser performance. Select the tab named **Network**, which is responsible for displaying the traffic being exchanged (e.g. when HTTP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



All HTML CSS JS XHR Fonts Images Media WS Other

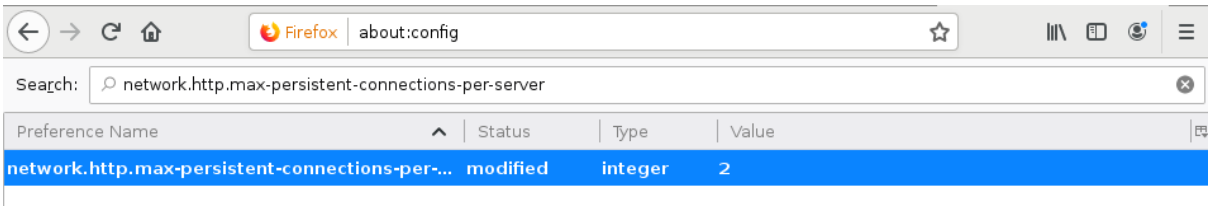
Status	Meth...	Domain	File	Cause	Type	Transferred	Size	0 ms	20.48 s	40.96 s
200	GET	ieeexplore....	AMSmath.js?V=2.7.4	script	js	5.40 KB (raced)	13.47...	296 ms		
200	GET	ieeexplore....	AMSsymbols.js?V=2.7.4	script	js	3 KB (raced)	6.46 ...	73 ms		
200	GET	ieeexplore....	ieeemacros.js?V=2.7.4	script	js	13.83 KB (rac...	62.28...	308 ms		
200	GET	ieeexplore....	cancel.js?V=2.7.4	script	js	1.56 KB (raced)	1.94 ...	109 ms		
200	GET	ieeexplore....	AMScd.js?V=2.7.4	script	js	2.06 KB (raced)	3.09 ...	324 ms		
200	GET	ieeexplore....	color_ieee.js?V=2.7.4	script	js	2.98 KB (raced)	5.58 ...	325 ms		
200	GET	ieeexplore....	cellcolor_ieee.js?V=2.7.4	script	js	1.43 KB (raced)	1.91 ...	344 ms		
200	GET	ieeexplore....	enclose.js?V=2.7.4	script	js	1.47 KB (raced)	1.57 ...	345 ms		
200	GET	ieeexplore....	euler_ieee.js?V=2.7.4	script	js	2.35 KB (raced)	6.48 ...	349 ms		
200	GET	ieeexplore....	upgreek.js?V=2.7.4	script	js	1.37 KB (raced)	2.01 ...	194 ms		
200	GET	ieeexplore....	ieee_stixext.js?V=2.7.4	script	js	8.23 KB (raced)	23.61...	372 ms		
200	GET	ieeexplore....	extpfeil.js?V=2.7.4	script	js	1.56 KB (raced)	2.03 ...	192 ms		

Now ask a second time for the index of the previously requested page (e.g. ieeexplore.ieee.org). Describe how this time HTTP caching works.
Note. To reproduce the experiment you have to remove the cache of firefox. You can do this clicking in ”clear your recent history“ in the menu Edit.Preferences.Privacy of firefox.

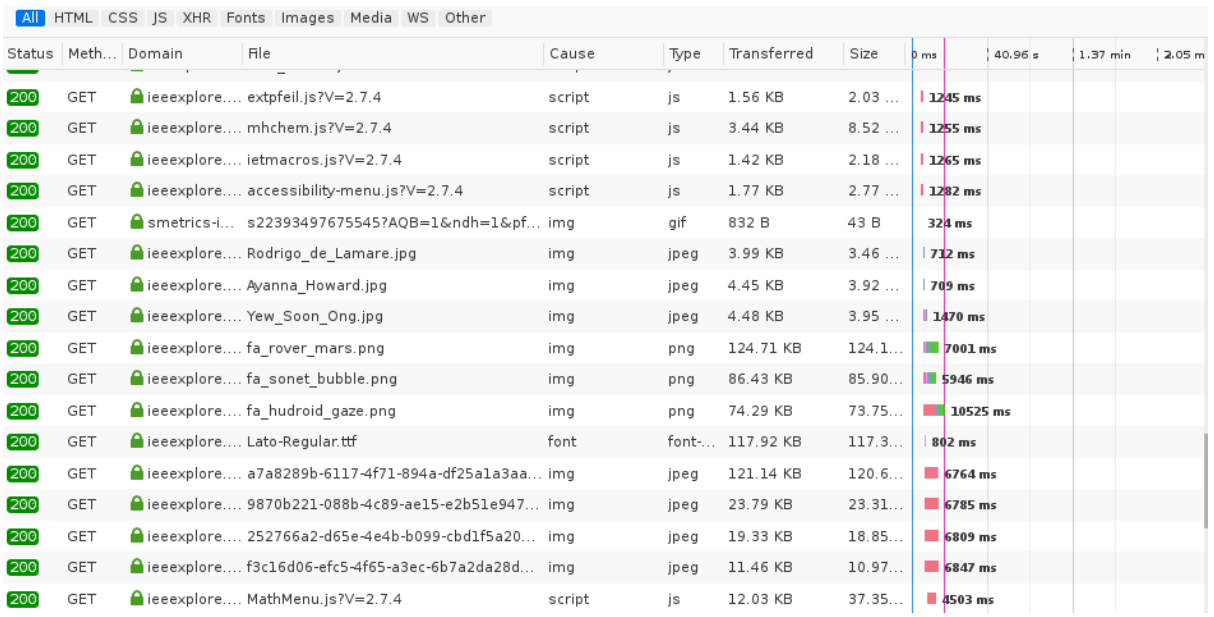


Now all the petitions have been cancelled-> The response has a 304 Not Modified code. Thats cause the data hasn't changed since the last petition.

2) Remove the cache of firefox and decrease its maximum number of persistent connections per server from 6 to 2. For this purpose, use the **about:config** string in the URL of firefox and then, search and modify the parameter **network.http.max-persistent-connections-per-server**.



From firefox request again for the index page of the complex webpage used above (e.g. ieeexplore.ieee.org).
Describe the HTTP traffic captured. In particular, comment the number of persistent connections that you observe now and the GET requests through each connection.
Note. When you finish the exercise, do not forget to set to 6 again the maximum number of persistent connections per server.



As we see, now the connections last more than before cause we have limited the persistent connections (to the same server) to 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”). -> /var/www

```
  GNU nano 2.0.7                File: 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;
    }
```

Note. You must create the “images” directory.
www:~# mkdir /var/www/images
www:~# cp /tmp/upc1.gif /tmp/upc2.gif /tmp/upc3.png /var/www/images/upc1.gif

Modify the HTML index of the server and create local hyperlinks to these images. Describe how you do it.

```
  GNU nano 2.0.7              File: index.html                    Modified

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

<img src="/images/upc1.gif">
<img src="/images/upc2.gif">
<img src="/images/upc3.png">

</body></html>
```

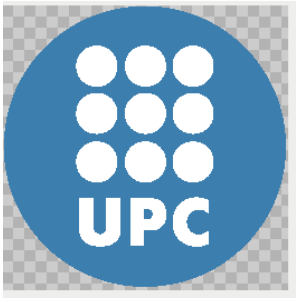
In order to see the changes it is needed to restart the daemon nginx:
www:~# /etc/init.d/nginx restart

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.
Explain how you do it.

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

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.1.1.4	10.1.1.1	TCP	74	52134 → 80 [SYN] Seq=0 Win=29200 Len=0 M
2	0.016163187	fe:fd:00:00:01:00	Broadcast	ARP	42	Who has 10.1.1.4? Tell 10.1.1.1
3	0.016174364	16:77:4d:db:9b:bb	fe:fd:00:00:01:00	ARP	42	10.1.1.4 is at 16:77:4d:db:9b:bb
4	0.016365411	10.1.1.1	10.1.1.4	TCP	74	80 → 52134 [SYN, ACK] Seq=0 Ack=1 Win=57
5	0.016381055	10.1.1.4	10.1.1.1	TCP	66	52134 → 80 [ACK] Seq=1 Ack=1 Win=29312 L
6	5.253946326	16:77:4d:db:9b:bb	fe:fd:00:00:01:00	ARP	42	Who has 10.1.1.1? Tell 10.1.1.4
7	5.254286182	fe:fd:00:00:01:00	16:77:4d:db:9b:bb	ARP	42	10.1.1.1 is at fe:fd:00:00:01:00
8	32.541402902	10.1.1.4	10.1.1.1	TCP	96	52134 → 80 [PSH, ACK] Seq=1 Ack=1 Win=29
9	32.541698577	10.1.1.1	10.1.1.4	TCP	66	80 → 52134 [ACK] Seq=1 Ack=31 Win=5792 L
10	37.543895967	fe:fd:00:00:01:00	16:77:4d:db:9b:bb	ARP	42	Who has 10.1.1.4? Tell 10.1.1.1
11	37.543908411	16:77:4d:db:9b:bb	fe:fd:00:00:01:00	ARP	42	10.1.1.4 is at 16:77:4d:db:9b:bb
12	37.765497504	16:77:4d:db:9b:bb	fe:fd:00:00:01:00	ARP	42	Who has 10.1.1.1? Tell 10.1.1.4
13	37.765872490	fe:fd:00:00:01:00	16:77:4d:db:9b:bb	ARP	42	10.1.1.1 is at fe:fd:00:00:01:00
14	47.286486971	10.1.1.4	10.1.1.1	HTTP	67	GET /images/upc1.gif HTTP/1.0
15	47.286835476	10.1.1.1	10.1.1.4	TCP	66	80 → 52134 [ACK] Seq=1 Ack=32 Win=5792 L
16	47.287111505	10.1.1.1	10.1.1.4	TCP	300	80 → 52134 [PSH, ACK] Seq=1 Ack=32 Win=5
17	47.287118908	10.1.1.4	10.1.1.1	TCP	66	52134 → 80 [ACK] Seq=32 Ack=235 Win=3033
18	47.287386751	10.1.1.1	10.1.1.4	TCP	1514	80 → 52134 [ACK] Seq=235 Ack=32 Win=5792
19	47.287392425	10.1.1.4	10.1.1.1	TCP	66	52134 → 80 [ACK] Seq=32 Ack=1683 Win=332
20	47.287545147	10.1.1.1	10.1.1.4	HTTP	746	HTTP/1.1 200 OK (GIF89a)
21	47.287549875	10.1.1.4	10.1.1.1	TCP	66	52134 → 80 [ACK] Seq=32 Ack=2363 Win=360
22	47.287823360	10.1.1.1	10.1.1.4	TCP	66	80 → 52134 [FIN, ACK] Seq=2363 Ack=32 Wi
23	47.287845810	10.1.1.4	10.1.1.1	TCP	66	52134 → 80 [FIN, ACK] Seq=32 Ack=2364 Wi
24	47.288280205	10.1.1.1	10.1.1.4	TCP	66	80 → 52134 [ACK] Seq=2364 Ack=33 Win=579

Edit appropriately the file response.http with mousepad or vi to obtain the original image upc1.gif.
www:~# vi response.http -> ¡IMPORTANT: do it with vi cause in images it reades chars that nano or other editor doesnt!
We should remove all the previous information and save starting from “GIF89” in a new file named image.gif.
Another option is to edit with nano and copy the file to a new one (image.gif). Then opened with vi and quit (typing “:quit”) and vuala cause vi is a fucking shit and i dont know how to use it.
Open image.gif manually)



After that, repeat the process using HTTP 1.1.
HTTP 1.1 needs also to be provided a host

```
telem@debian:~$ nc 10.1.1.1 80 > response.http
GET /images/upc1.gif HTTP/1.1
Host: 10.1.1.1
80
```

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?

Wget: the non interactive network downloader: puede ejecutarse en el background. Supports HTTP, HTTPS y FTP.
phyhost:~# wget 10.1.1.1/images/upc1.gif

```
telem@debian:~$ wget 10.1.1.1/images/upc1.gif
--2021-04-13 12:06:52--  http://10.1.1.1/images/upc1.gif
Connecting to 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

2021-04-13 12:06:52 (242 MB/s) - 'upc1.gif' saved [2128/2128]
```

In wireshark it appears the HTTP version used: 1.1

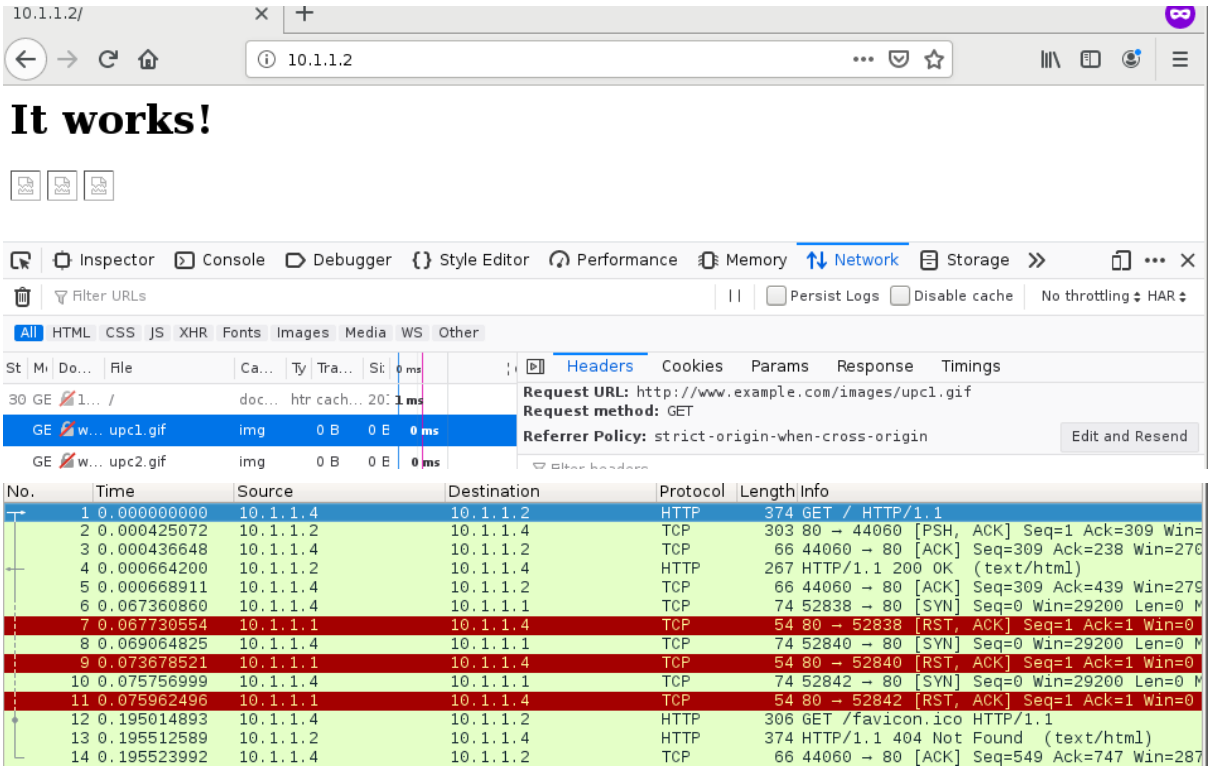
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.1.1.4	10.1.1.1	TCP	74	52632 → 80 [SYN] Seq=0 Win=29200 Len=0 M
2	0.021575521	fe:fd:00:00:01:00	Broadcast	ARP	42	Who has 10.1.1.4? Tell 10.1.1.1
3	0.021589552	16:77:4d:db:9b:bb	fe:fd:00:00:01:00	ARP	42	10.1.1.4 is at 16:77:4d:db:9b:bb
4	0.021830319	10.1.1.1	10.1.1.4	TCP	74	80 → 52632 [SYN, ACK] Seq=0 Ack=1 Win=57
5	0.021848261	10.1.1.4	10.1.1.1	TCP	66	52632 → 80 [ACK] Seq=1 Ack=1 Win=29312 L
6	0.026836925	10.1.1.4	10.1.1.1	HTTP	214	GET /images/upc1.gif HTTP/1.1
7	0.027135188	10.1.1.1	10.1.1.4	TCP	66	80 → 52632 [ACK] Seq=1 Ack=149 Win=5792
8	0.027438477	10.1.1.1	10.1.1.4	TCP	305	80 → 52632 [PSH, ACK] Seq=1 Ack=149 Win=
9	0.027445958	10.1.1.4	10.1.1.1	TCP	66	52632 → 80 [ACK] Seq=149 Ack=240 Win=303
10	0.027673956	10.1.1.1	10.1.1.4	TCP	1514	80 → 52632 [ACK] Seq=240 Ack=149 Win=579
11	0.027680995	10.1.1.4	10.1.1.1	TCP	66	52632 → 80 [ACK] Seq=149 Ack=1688 Win=33
12	0.027695764	10.1.1.1	10.1.1.4	HTTP	746	HTTP/1.1 200 OK (GIF89a)
13	0.027699379	10.1.1.4	10.1.1.1	TCP	66	52632 → 80 [ACK] Seq=149 Ack=2368 Win=36
14	0.033038495	10.1.1.4	10.1.1.1	TCP	66	52632 → 80 [FIN, ACK] Seq=149 Ack=2368 W
15	0.033378636	10.1.1.1	10.1.1.4	TCP	66	80 → 52632 [FIN, ACK] Seq=2368 Ack=150 W
16	0.033390526	10.1.1.4	10.1.1.1	TCP	66	52632 → 80 [ACK] Seq=150 Ack=2369 Win=36
17	5.079102026	16:77:4d:db:9b:bb	fe:fd:00:00:01:00	ARP	42	Who has 10.1.1.1? Tell 10.1.1.4
18	5.106000518	fe:fd:00:00:01:00	16:77:4d:db:9b:bb	ARP	42	10.1.1.1 is at fe:fd:00:00:01:00

6) Get a console on 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.
server:~# /etc/init.d/nginx start
server:~# nano /var/www/index.html

```
  GNU nano 2.0.7              File: index.html

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

<img src="http://www.example.com/images/upc1.gif">
<img src="http://www.example.com/images/upc2.gif">
<img src="http://www.example.com/images/upc3.png">

</body></html>
```

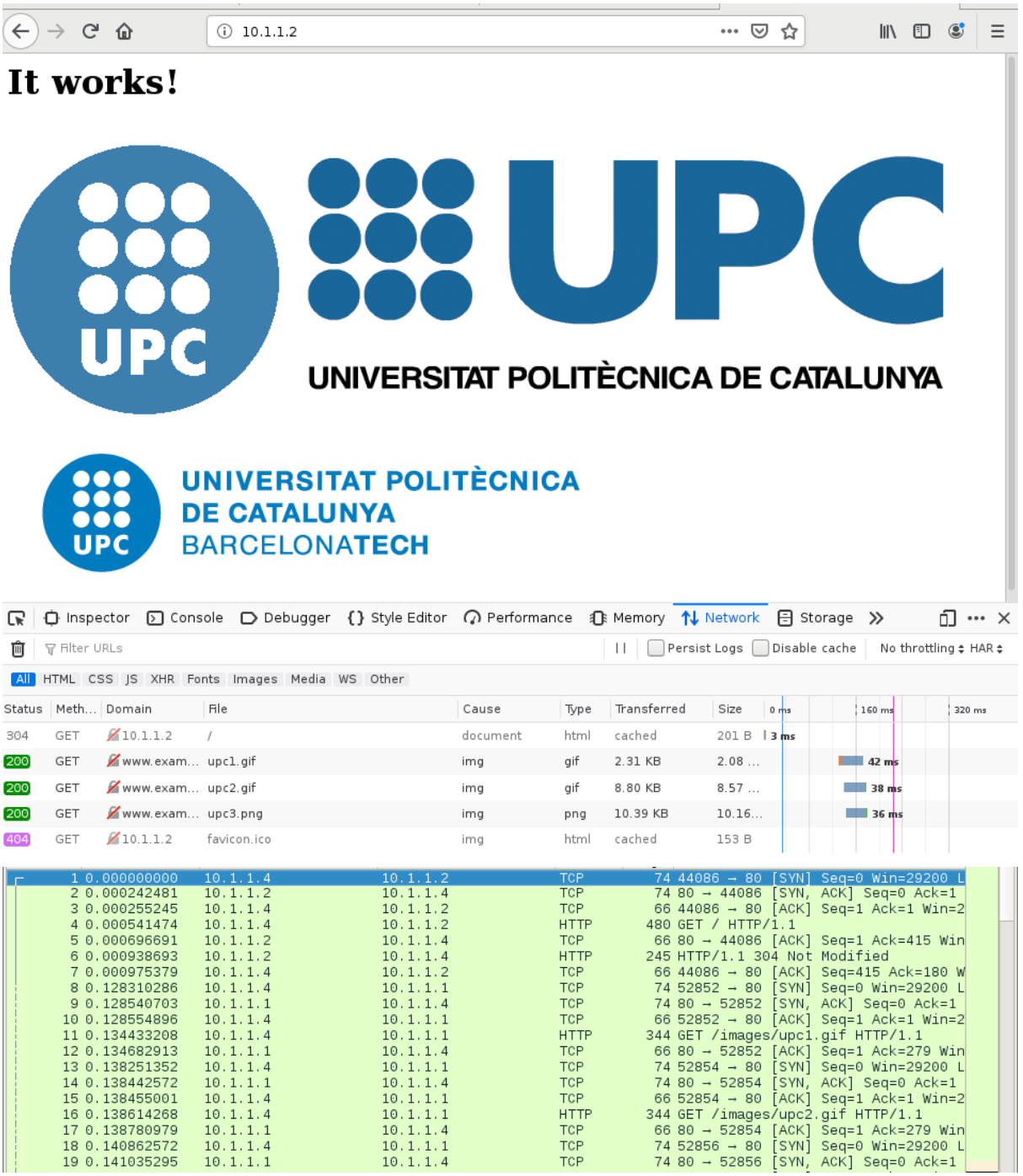
server:~# /etc/init.d/nginx restart

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 the 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).
www:~# /etc/init.d/nginx stop
server:~# ifconfig-> @IP= 10.1.1.2



The images cannot be loaded. Thats because the www daemon is closed. So when the html is loaded it cannot access to the http content in www.

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.
www:~# /etc/init.d/nginx start



It works perfectly cause now the browser (firefox) can access to the http content in 10.1.1.1

Exercise 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 and paste the content from the PDF, since that operation often includes unwanted extra characters.
   Describe the steps of your configuration.
   cd /www:~# nano /var/www/cgi-bin/datecgi.sh

TCGI slides 10.01.01
https://drive.google.com/drive/folders/1ugbAayGfGTtjjVX0rLdREecVnr-qNXcN?usp=sharing

From the pract pdf (Code 1.6):

```
 GNU nano 2.0.7          File: cgi-bin/datecgi.sh

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

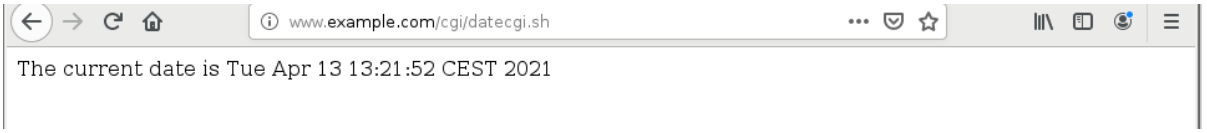
Need to give permissions:
www: ~# cd /var/www/cgi-bin
www:/var/www/cgi-bin# ls -l

```
www:/var/www/cgi-bin# ls -l
total 8
-rw-r--r-- 1 root     root     126 Apr 13 13:10 datecgi.sh
-rwx------ 1 www-data www-data 193 Feb 12  2020 example.sh
www:/var/www/cgi-bin#
```

www: ~# chown www-data:www-data datecgi.sh (autor:file group)
www: ~# chmod 700 datecgi.sh

```
www:/var/www/cgi-bin# chown www-data:www-data datecgi.sh
www:/var/www/cgi-bin# ls -l
total 8
-rw-r--r-- 1 www-data www-data 126 Apr 13 13:10 datecgi.sh
-rwx------ 1 www-data www-data 193 Feb 12  2020 example.sh
www:/var/www/cgi-bin# chmod 700 datecgi.sh
www:/var/www/cgi-bin# ls -l
total 8
-rwx------ 1 www-data www-data 126 Apr 13 13:10 datecgi.sh
-rwx------ 1 www-data www-data 193 Feb 12  2020 example.sh
```

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.



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.
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)
```

The 2 default locations are: “/” and "whatever cgi”.
So, when we ask for the URL http://www.example.com/cgi-bin/datecgi.sh the server would take location '/'-> root cause “cgi-bin” isnt “cgi”. By chance, in the directory /var/www/cgi-bin we have the script “datecgi.sh”. So the server reads the petition as if the user was asking for that file and not for the cgi interpretation.

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.
www: ~# cd /etc/nginx/sites-available/
www: ~# nano non-existing-sites

```
 GNU nano 2.0.7          File: non-existing-sites

server {
    listen      80 default_server;
    server_name _ ;
    return 503  "No server is currently configured for the requested host." ;
}
```

For the non existing sites the nginx is configured to listen in the default server. So, we are going to add and configure the location “/cgi-bin/” in order to return the error.
www: ~# nano /etc/nginx/sites-enabled/default

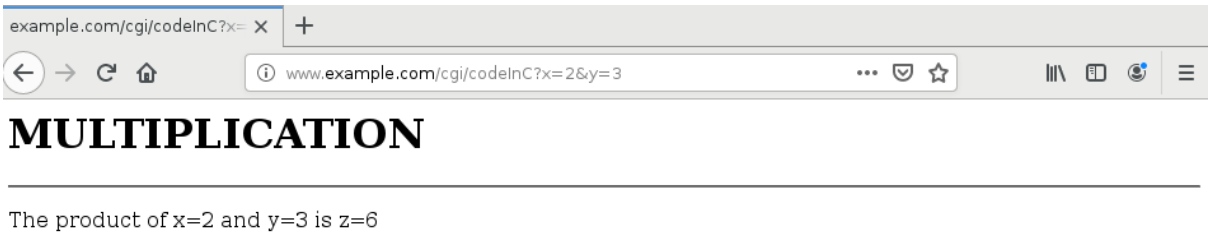
```
 GNU nano 2.0.7              File: default

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;
    }

    location ~ ^/cgi/ {
```

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.
www: ~# cd /var/www/cgi-bin/
www: ~# nano codeInC.c

```
 GNU nano 2.0.7            File: 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*$
                printf( "</body></html>\n");
        return 0;
}
                                [ Wrote 19 lines ]

www:/var/www/cgi-bin#
```

www: ~# gcc codeInC.c -o codeInC
www: ~# chown www-data:www-data codeInC
www: ~# chmod 700 codeInC
Restart the daemon and search in browser:

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



Exercise 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.
   www: ~# netstat -tnlp | grep nginx

```
www:~# netstat -tnlp | grep nginx
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
1171/nginx
```

   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:
   dns: ~# nano /etc/bind/db.example.net

```
  GNU nano 2.0.7            File: db.example.net

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

```
dns#/etc/init.d/bind9 restart
```

   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.
   phyhost: ~# ping www.example.com/www.example.net

```
host:~# ping www.example.net
PING www.example.net (10.1.1.1) 56(84) bytes of data.
64 bytes from www1.example.com (10.1.1.1): icmp_seq=1 ttl=64 time=0.264 ms
64 bytes from www1.example.com (10.1.1.1): icmp_seq=2 ttl=64 time=0.455 ms
```

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:

```
www# cd /etc/nginx/sites-available/
www# cp default www.example.com
www# cp default www.example.net
```

Change the document root and server name keys in these new configurations. To do so, just edit the files or alternatively proceed with the following steps.
Change the document root in any key that uses it by typing:

```
www# sed -i "s/\/var\/www/\/var\/www\/com/g" www.example.com
www# sed -i "s/\/var\/www/\/var\/www\/net/g" www.example.net
```

Change now the server name by editing these files or just typing:

```
www# sed -i "s/server_name.*/server_name www.example.com;/" www.example.com
www# sed -i "s/server_name.*/server_name www.example.net;/" www.example.net
```

Open the text files and verify that both, document root and server name have been properly changed.

```
  GNU nano 2.0.7            File: 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;

        # Required to forbid default content browsing
        autoindex off;
```

```
  GNU nano 2.0.7            File: www.example.net

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.net;

    location / {
        root   /var/www/net;
        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;
```

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

```
www# cd /etc/nginx/sites-enabled/
www# rm default
```

Then, we need to enable the www.example.com and www.example.net virtual hosts. To do so we need to set a soft link to their configuration in the sites-available folder. Accordingly, type:

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

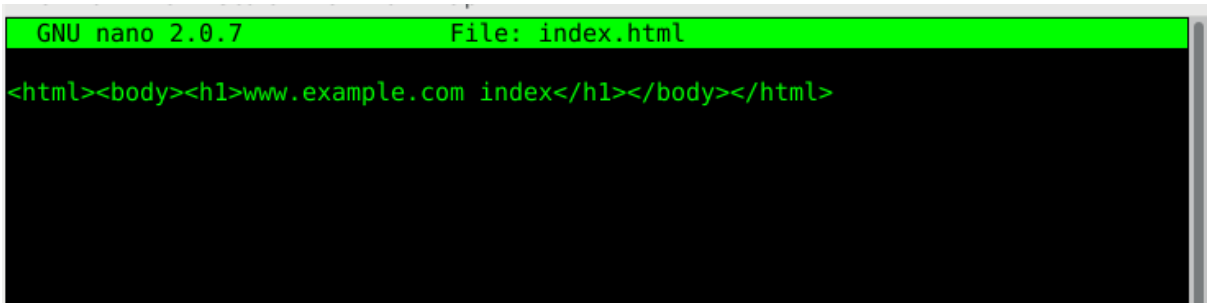
And reload the configuration of nginx:

```
www# /etc/init.d/nginx reload
```

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/ned respectively. These folders need to be created in advance by typing:

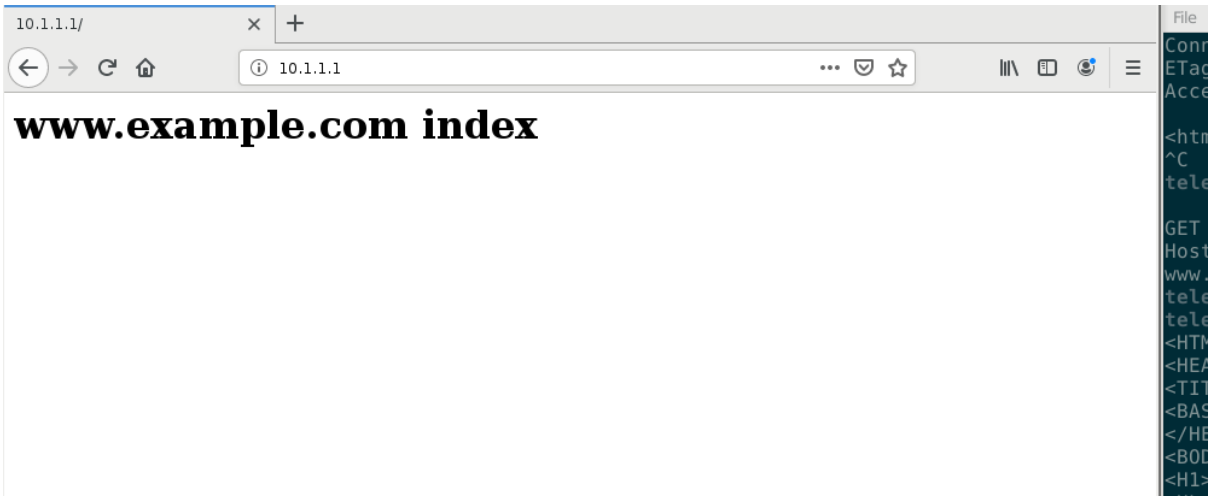
```
www# mkdir /var/www/com
www# mkdir /var/www/net
```

www:/var/www# cp index.html /var/www/com/index.html

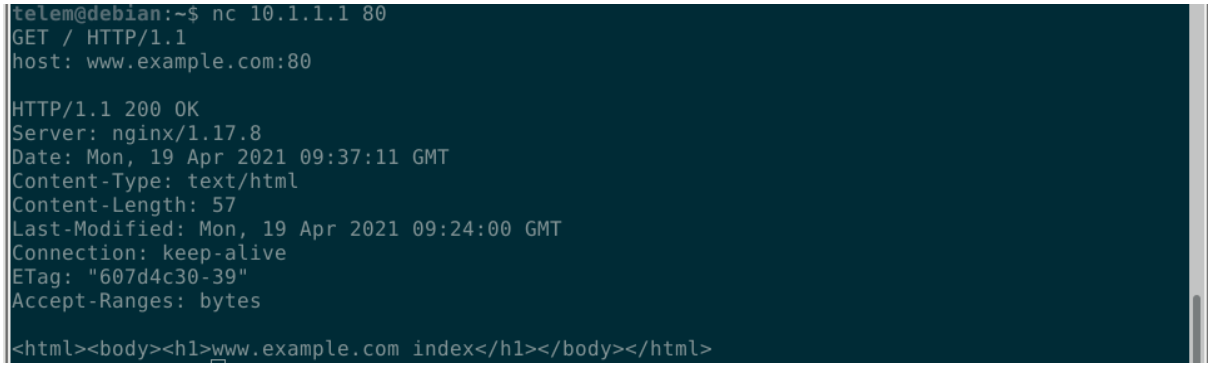


www:/var/www# cp index.html /var/www/net/index.html

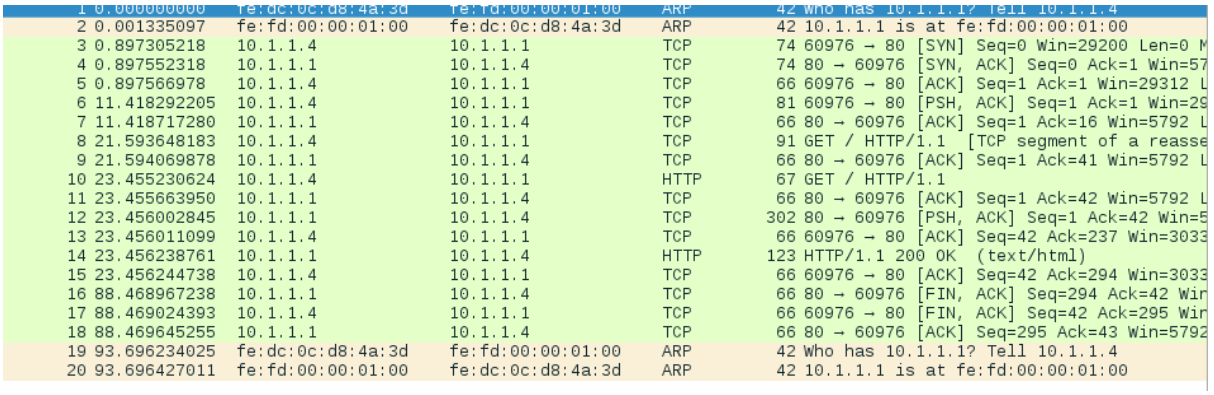
Capture on tap0 and describe how you test this configuration. Try also to connect directly with the IP address 10.1.1.1. Discuss the results.
From the phyhost we open the browser and search for the address www.example.com and www.example.net. We should see the html page described in index.html from each folder.
(Make sure phyhost has an @IP address assigned and server is restarted)



If we search the @IP address assigned it charges the first index.html (.com)
Or using nc (we can say to which host we wanna connect):



SimNet0:



1	0.000000000	fe:dc:0c:d8:4a:3d	fe:fd:00:00:01:00	ARP	42 Who has 10.1.1.1? Tell 10.1.1.4
2	0.001335097	fe:fd:00:00:01:00	fe:dc:0c:d8:4a:3d	ARP	42 10.1.1.1 is at fe:fd:00:00:01:00
3	0.897305218	10.1.1.4	10.1.1.1	TCP	74 60976 → 80 [SYN] Seq=0 Win=29200 Len=0 M
4	0.897552318	10.1.1.1	10.1.1.4	TCP	74 80 → 60976 [SYN, ACK] Seq=0 Ack=1 Win=57
5	0.897566978	10.1.1.4	10.1.1.1	TCP	66 60976 → 80 [ACK] Seq=1 Ack=1 Win=29312 L
6	11.418292205	10.1.1.4	10.1.1.1	TCP	81 60976 → 80 [PSH, ACK] Seq=1 Ack=1 Win=29
7	11.418717280	10.1.1.1	10.1.1.4	TCP	66 80 → 60976 [ACK] Seq=1 Ack=16 Win=5792 L
8	21.593648183	10.1.1.4	10.1.1.1	TCP	91 GET / HTTP/1.1 [TCP segment of a reasse
9	21.594069878	10.1.1.1	10.1.1.4	TCP	66 80 → 60976 [ACK] Seq=1 Ack=41 Win=5792 L
10	23.455230624	10.1.1.4	10.1.1.1	HTTP	67 GET / HTTP/1.1
11	23.455663950	10.1.1.1	10.1.1.4	TCP	66 80 → 60976 [ACK] Seq=1 Ack=42 Win=5792 L
12	23.456002845	10.1.1.1	10.1.1.4	TCP	302 80 → 60976 [PSH, ACK] Seq=1 Ack=42 Win=5
13	23.456011099	10.1.1.4	10.1.1.1	TCP	66 60976 → 80 [ACK] Seq=42 Ack=237 Win=3033
14	23.456238761	10.1.1.1	10.1.1.4	HTTP	123 HTTP/1.1 200 OK (text/html)
15	23.456244738	10.1.1.4	10.1.1.1	TCP	66 60976 → 80 [ACK] Seq=42 Ack=294 Win=3033
16	88.468967238	10.1.1.1	10.1.1.4	TCP	66 80 → 60976 [FIN, ACK] Seq=294 Ack=42 Win
17	88.469024393	10.1.1.4	10.1.1.1	TCP	66 60976 → 80 [FIN, ACK] Seq=42 Ack=295 Win
18	88.469645255	10.1.1.1	10.1.1.4	TCP	66 80 → 60976 [ACK] Seq=295 Ack=43 Win=5792
19	93.696234025	fe:dc:0c:d8:4a:3d	fe:fd:00:00:01:00	ARP	42 Who has 10.1.1.1? Tell 10.1.1.4
20	93.696427011	fe:fd:00:00:01:00	fe:dc:0c:d8:4a:3d	ARP	42 10.1.1.1 is at fe:fd:00:00:01:00

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

HTTP/1.1 200 OK
Server: nginx/1.17.8
Date: Mon, 19 Apr 2021 09:40:05 GMT
Content-Type: text/html
Content-Length: 57
Last-Modified: Mon, 19 Apr 2021 09:24:41 GMT
Connection: keep-alive
ETag: "607d4c59-39"
Accept-Ranges: bytes

<html><body><h1>www.example.net index</h1></body></html>
```

SimNet0:

```
    22 161.049067403 fe:fd:00:00:01:00    Broadcast           ARP        42 Who has 10.1.1.4? Tell 10.1.1.1
    23 161.049080409 fe:dc:0c:d8:4a:3d    fe:fd:00:00:01:00   ARP        42 10.1.1.4 is at fe:dc:0c:d8:4a:3d
    24 161.049314427 10.1.1.1             10.1.1.4            TCP        74 80 → 60978 [SYN, ACK] Seq=0 Ack=1 Win=
    25 161.049334046 10.1.1.4             10.1.1.1            TCP        66 60978 → 80 [ACK] Seq=1 Ack=1 Win=29312
    26 166.144264328 fe:dc:0c:d8:4a:3d    fe:fd:00:00:01:00   ARP        42 Who has 10.1.1.1? Tell 10.1.1.4
    27 166.144669126 fe:fd:00:00:01:00    fe:dc:0c:d8:4a:3d   ARP        42 10.1.1.1 is at fe:fd:00:00:01:00
    28 176.285477455 10.1.1.4             10.1.1.1            TCP        81 60978 → 80 [PSH, ACK] Seq=1 Ack=1 Win=
    29 176.285773994 10.1.1.1             10.1.1.4            TCP        66 80 → 60978 [ACK] Seq=1 Ack=16 Win=5792
    30 188.652288653 10.1.1.4             10.1.1.1            TCP        91 GET / HTTP/1.1  [TCP segment of a reas
    31 188.652703864 10.1.1.1             10.1.1.4            TCP        66 80 → 60978 [ACK] Seq=1 Ack=41 Win=5792
    32 197.794494545 10.1.1.4             10.1.1.1            HTTP       67 GET / HTTP/1.1
    33 197.794922153 10.1.1.1             10.1.1.4            TCP        66 80 → 60978 [ACK] Seq=1 Ack=42 Win=5792
    34 197.795238452 10.1.1.1             10.1.1.4            TCP       302 80 → 60978 [PSH, ACK] Seq=1 Ack=42 Win
    35 197.795247221 10.1.1.4             10.1.1.1            TCP        66 60978 → 80 [ACK] Seq=42 Ack=237 Win=30
    36 197.795470448 10.1.1.1             10.1.1.4            HTTP      123 HTTP/1.1 200 OK  (text/html)
    37 197.795476448 10.1.1.4             10.1.1.1            TCP        66 60978 → 80 [ACK] Seq=42 Ack=294 Win=30
    38 230.015112588 10.1.1.4             10.1.1.1            TCP        66 60978 → 80 [FIN, ACK] Seq=42 Ack=294 W
    39 230.015987118 10.1.1.1             10.1.1.4            TCP        66 80 → 60978 [FIN, ACK] Seq=294 Ack=43 W
    40 230.016003172 10.1.1.4             10.1.1.1            TCP        66 60978 → 80 [ACK] Seq=43 Ack=295 Win=30
```

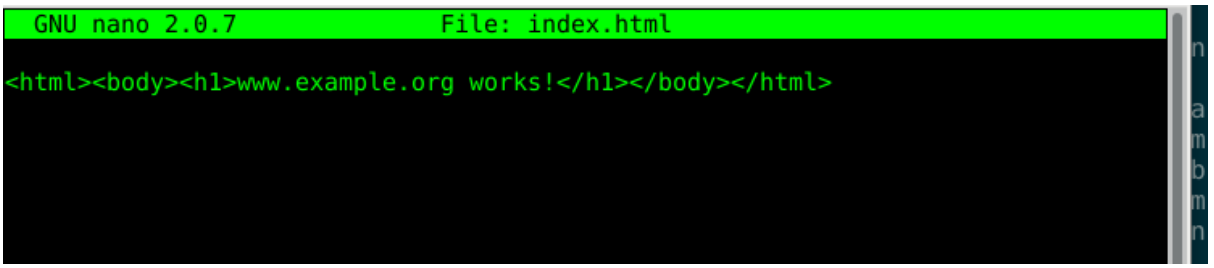
3) Now we are going to test a configuration where the domain www.example.org is translated by the dns server to two different IP addresses: 10.1.1.1 and 10.1.1.2. The bind9 server uses a round robin strategy for translating names with multiple IP addresses (a different IP address for each query in a 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.

```
  GNU nano 2.0.7            File: 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;
```

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

```
# 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
```

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?
It is useful if the www console is busy and we need to distribute the traffic. (the easy way)

Exercise 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.

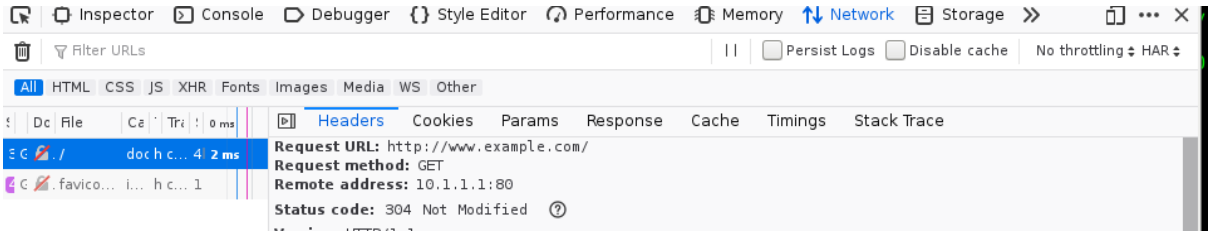
1) In the machine www, go to the sites-available folder and open the balancer configuration file. Check it out and figure out what each code section does.

```
 GNU nano 2.0.7                File: 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;
    }
}
```

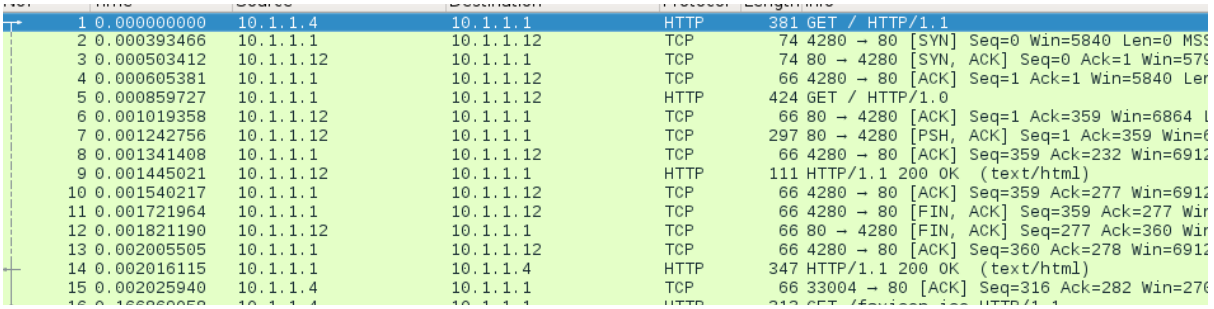
The server listens on port 80 and the domain is www.example.com.
When the user charges the main page (root /), the server would put the headers written and load the http://tcgi-app… url.
In the first two lines we see that this link is loaded by the www1 and www2 server. So it is configured a balancer that uses round robin. (first request to first server, second to 2 and consecutively).

2) Still in the machine www, set up the nginx configuration so that only the balancer configuration is run.
To do it so we need to delete the previous configurations and enable the balancer.
www: /etc/nginx/sites-enabled# rm www.example.org www.example.net www.example.com
www: /etc/nginx/sites-enabled# ln -s ../sites-available/balancer
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?

3) Start the nginx servers at the www1 and www2 machines. Try again to access the url www.example.com.
www1: # /etc/init.d/nginx start
www2: # /etc/init.d/nginx start



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.
It is attended by www but really it is www2 who cached the request.
SimNet0:

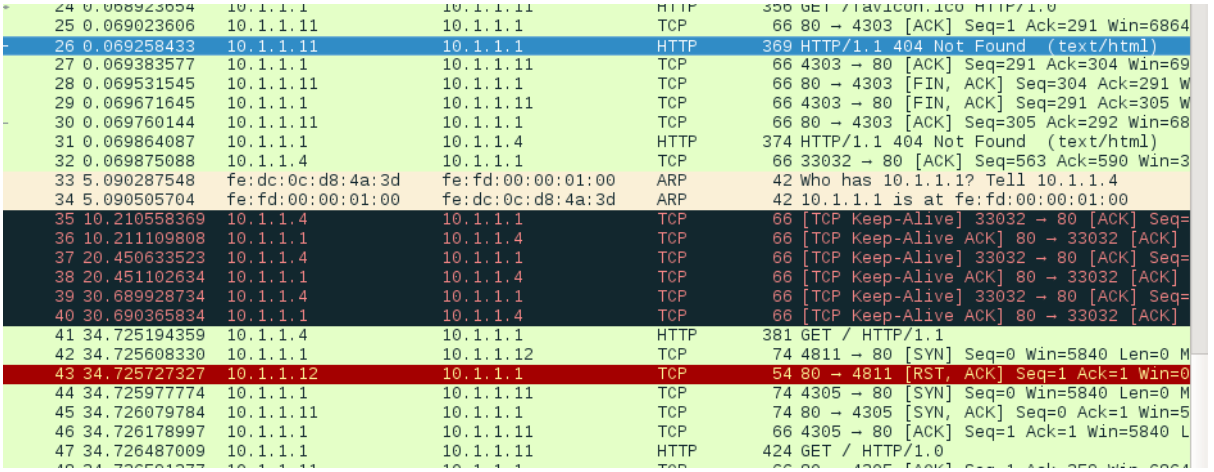


1	0.000000000	10.1.1.4	10.1.1.1	HTTP	381	GET / HTTP/1.1
2	0.000393466	10.1.1.1	10.1.1.12	TCP	74	4280 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS
3	0.000503412	10.1.1.12	10.1.1.1	TCP	74	80 → 4280 [SYN, ACK] Seq=0 Ack=1 Win=579
4	0.000605381	10.1.1.1	10.1.1.12	TCP	66	4280 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len
5	0.000859727	10.1.1.1	10.1.1.12	HTTP	424	GET / HTTP/1.0
6	0.001019358	10.1.1.12	10.1.1.1	TCP	66	80 → 4280 [ACK] Seq=1 Ack=359 Win=6864 L
7	0.001242756	10.1.1.12	10.1.1.1	TCP	297	80 → 4280 [PSH, ACK] Seq=1 Ack=359 Win=6
8	0.001341408	10.1.1.1	10.1.1.12	TCP	66	4280 → 80 [ACK] Seq=359 Ack=232 Win=6912
9	0.001445021	10.1.1.12	10.1.1.1	HTTP	111	HTTP/1.1 200 OK (text/html)
10	0.001540217	10.1.1.1	10.1.1.12	TCP	66	4280 → 80 [ACK] Seq=359 Ack=277 Win=6912
11	0.001721964	10.1.1.1	10.1.1.12	TCP	66	4280 → 80 [FIN, ACK] Seq=359 Ack=277 Win
12	0.001821190	10.1.1.12	10.1.1.1	TCP	66	80 → 4280 [FIN, ACK] Seq=277 Ack=360 Win
13	0.002005505	10.1.1.1	10.1.1.12	TCP	66	4280 → 80 [ACK] Seq=360 Ack=278 Win=6912
14	0.002016115	10.1.1.1	10.1.1.4	HTTP	347	HTTP/1.1 200 OK (text/html)
15	0.002025940	10.1.1.4	10.1.1.1	TCP	66	33004 → 80 [ACK] Seq=316 Ack=282 Win=276

4) Access again the url www.example.com. Do it several times. Discuss the results.
The request is attended by one of the servers each time.
5) Let's simulate a service failure by stopping the nginx server at the www2 machine.

```
www2# /etc/init.d/nginx stop
```

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.



25	0.069023606	10.1.1.11	10.1.1.1	TCP	66	80 → 4303 [ACK] Seq=1 Ack=291 Win=6864
26	0.069258433	10.1.1.11	10.1.1.1	HTTP	369	HTTP/1.1 404 Not Found (text/html)
27	0.069383577	10.1.1.1	10.1.1.11	TCP	66	4303 → 80 [ACK] Seq=291 Ack=304 Win=69
28	0.069531545	10.1.1.11	10.1.1.1	TCP	66	80 → 4303 [FIN, ACK] Seq=304 Ack=291 W
29	0.069671645	10.1.1.1	10.1.1.11	TCP	66	4303 → 80 [FIN, ACK] Seq=291 Ack=305 W
30	0.069760144	10.1.1.11	10.1.1.1	TCP	66	80 → 4303 [ACK] Seq=305 Ack=292 Win=68
31	0.069864087	10.1.1.1	10.1.1.4	HTTP	374	HTTP/1.1 404 Not Found (text/html)
32	0.069875088	10.1.1.4	10.1.1.1	TCP	66	33032 → 80 [ACK] Seq=563 Ack=590 Win=3
33	5.090287548	fe:dc:0c:d8:4a:3d	fe:fd:00:00:01:00	ARP	42	Who has 10.1.1.1? Tell 10.1.1.4
34	5.090505704	fe:fd:00:00:01:00	fe:dc:0c:d8:4a:3d	ARP	42	10.1.1.1 is at fe:fd:00:00:01:00
35	10.210558369	10.1.1.4	10.1.1.1	TCP	66	[TCP Keep-Alive] 33032 → 80 [ACK] Seq=
36	10.211109808	10.1.1.1	10.1.1.4	TCP	66	[TCP Keep-Alive ACK] 80 → 33032 [ACK]
37	20.450633523	10.1.1.4	10.1.1.1	TCP	66	[TCP Keep-Alive] 33032 → 80 [ACK] Seq=
38	20.451102634	10.1.1.1	10.1.1.4	TCP	66	[TCP Keep-Alive ACK] 80 → 33032 [ACK]
39	30.689928734	10.1.1.4	10.1.1.1	TCP	66	[TCP Keep-Alive] 33032 → 80 [ACK] Seq=
40	30.690365834	10.1.1.1	10.1.1.4	TCP	66	[TCP Keep-Alive ACK] 80 → 33032 [ACK]
41	34.725194359	10.1.1.4	10.1.1.1	HTTP	381	GET / HTTP/1.1
42	34.725608330	10.1.1.1	10.1.1.12	TCP	74	4811 → 80 [SYN] Seq=0 Win=5840 Len=0 M
43	34.725727327	10.1.1.12	10.1.1.1	TCP	54	80 → 4811 [RST, ACK] Seq=1 Ack=1 Win=0
44	34.725977774	10.1.1.1	10.1.1.11	TCP	74	4305 → 80 [SYN] Seq=0 Win=5840 Len=0 M
45	34.726079784	10.1.1.11	10.1.1.1	TCP	74	80 → 4305 [SYN, ACK] Seq=0 Ack=1 Win=5
46	34.726178997	10.1.1.1	10.1.1.11	TCP	66	4305 → 80 [ACK] Seq=1 Ack=1 Win=5840 L
47	34.726487009	10.1.1.1	10.1.1.11	HTTP	424	GET / HTTP/1.0

The second time trying to access the url it should be loaded by www2 but it sends a [RST,ACK], which is saying that the flag RST is set. (disconnected).
When we start the server in 1 and 2 it goes as planned:

```
    23 2.471713680   10.1.1.11          10.1.1.1           TCP        66 80 → 4980 [ACK] Seq=1 Ack=291 Win=6864
    24 2.471948654   10.1.1.11          10.1.1.1           HTTP      369 HTTP/1.1 404 Not Found  (text/html)
    25 2.472048235   10.1.1.1           10.1.1.11          TCP        66 4980 → 80 [ACK] Seq=291 Ack=304 Win=69
    26 2.472989213   10.1.1.11          10.1.1.1           TCP        66 80 → 4980 [FIN, ACK] Seq=304 Ack=291 W
    27 2.473146536   10.1.1.1           10.1.1.11          TCP        66 4980 → 80 [FIN, ACK] Seq=291 Ack=305 W
    28 2.473287420   10.1.1.11          10.1.1.1           TCP        66 80 → 4980 [ACK] Seq=305 Ack=292 Win=68
    29 2.473418657   10.1.1.1           10.1.1.4           HTTP      374 HTTP/1.1 404 Not Found  (text/html)
    30 2.473431389   10.1.1.4           10.1.1.1           TCP        66 33036 → 80 [ACK] Seq=563 Ack=590 Win=3
    31 5.870444909   10.1.1.4           10.1.1.1           HTTP      487 GET / HTTP/1.1
    32 5.870848474   10.1.1.1           10.1.1.12          TCP        74 4549 → 80 [SYN] Seq=0 Win=5840 Len=0 M
    33 5.870961497   10.1.1.12          10.1.1.1           TCP        74 80 → 4549 [SYN, ACK] Seq=0 Ack=1 Win=5
    34 5.871059855   10.1.1.1           10.1.1.12          TCP        66 4549 → 80 [ACK] Seq=1 Ack=1 Win=5840 L
    35 5.871504555   10.1.1.1           10.1.1.12          HTTP      530 GET / HTTP/1.0
    36 5.871593923   10.1.1.12          10.1.1.1           TCP        66 80 → 4549 [ACK] Seq=1 Ack=465 Win=6864
    37 5.890403843   10.1.1.12          10.1.1.1           HTTP      240 HTTP/1.1 304 Not Modified
    38 5.890561696   10.1.1.1           10.1.1.12          TCP        66 4549 → 80 [ACK] Seq=465 Ack=175 Win=69
    39 5.890738461   10.1.1.1           10.1.1.12          TCP        66 4549 → 80 [FIN, ACK] Seq=465 Ack=175 W
```

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

```
server www1.example.com weight=2
```

, 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 webpage several times. Discuss the results.

```
  GNU nano 2.0.7                File: balancer                          Modified

upstream tcgi-app {
    server www1.example.com weight=3;
    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;
    }
}
```

SimNet0:

http Expression...

No.	Time	Source	Destination	Protocol	Length	Info
3	1.343545500	10.1.1.4	10.1.1.1	HTTP	381	GET / HTTP/1.1
7	1.344419196	10.1.1.1	10.1.1.11	HTTP	424	GET / HTTP/1.0
11	1.344972037	10.1.1.11	10.1.1.1	HTTP	111	HTTP/1.1 200 OK (text/html)
16	1.347099472	10.1.1.1	10.1.1.4	HTTP	347	HTTP/1.1 200 OK (text/html)
18	1.459599809	10.1.1.4	10.1.1.1	HTTP	313	GET /favicon.ico HTTP/1.1
22	1.460444409	10.1.1.1	10.1.1.11	HTTP	356	GET /favicon.ico HTTP/1.0
24	1.460821441	10.1.1.11	10.1.1.1	HTTP	369	HTTP/1.1 404 Not Found (text/html)
29	1.461320692	10.1.1.1	10.1.1.4	HTTP	374	HTTP/1.1 404 Not Found (text/html)
31	8.571863917	10.1.1.4	10.1.1.1	HTTP	381	GET / HTTP/1.1
35	8.572743211	10.1.1.1	10.1.1.11	HTTP	424	GET / HTTP/1.0
39	8.573297219	10.1.1.11	10.1.1.1	HTTP	111	HTTP/1.1 200 OK (text/html)
42	8.573685271	10.1.1.1	10.1.1.4	HTTP	347	HTTP/1.1 200 OK (text/html)
46	8.660080549	10.1.1.4	10.1.1.1	HTTP	313	GET /favicon.ico HTTP/1.1
50	8.662234105	10.1.1.1	10.1.1.12	HTTP	356	GET /favicon.ico HTTP/1.0
52	8.663415537	10.1.1.12	10.1.1.1	HTTP	369	HTTP/1.1 404 Not Found (text/html)
55	8.663722623	10.1.1.1	10.1.1.4	HTTP	374	HTTP/1.1 404 Not Found (text/html)

The first 3 requests are attended by www1, the last one by www2.

Exercise 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:

```
certificate: /etc/ssl/certs/ssl-cert-snakeoil.pem
key: /etc/ssl/private/ssl-cert-snakeoil.key
```

First, we create a key pair for the CA:

$ openssl genrsa -des3 -out mycakey.pem 2048

• Note that the key pair is protected (with DES3) by a password.

```
www:/etc/nginx/sites-enabled# CD --
-bash: CD: command not found
www:/etc/nginx/sites-enabled# cd --
www:~# openssl genrsa -des3 -out mycakey.pem 2048
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:
www:~# openssl req -new -x509 -days 2000 -key mycakey.pem -out mycacert.pem
```

• Now, we create a certificate for the CA:

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

• You have to answer some questions about the certificate.

• You can check the data of the certificate with the command:

$ openssl x509 -in mycacert.pem -text -noout

```
www:~# openssl req -new -x509 -days 2000 -key mycakey.pem -out mycacert.pem
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]:SP
State or Province Name (full name) [Some-State]:Cat
Locality Name (eg, city) []:BCN
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UPC
Organizational Unit Name (eg, section) []:TCGI
Common Name (eg, YOUR name) []:Mireia
Email Address []:mireia
```

Then, you can create a "server key pair":

$ openssl genrsa -out myserverkey.pem 2048

• We need the private key in clear (note that we do not use the option -des3).

• So, since we are not encrypting the key, we must keep it with restricted access:

$ chmod 400 myserverkey.pem

```
www:~# openssl genrsa -out myserverkey.pem 2048
Generating RSA private key, 2048 bit long modulus
.....................+++
..............+++
e is 65537 (0x10001)
www:~# chmod 400 myserverkey.pem
```

Now, we create a certificate request .csr:

$ openssl req -new -key myserverkey.pem -out myservercert.csr

When filling out the form, you have to use a common name that is set to the IP address or Full domain name of the server.-> www.example.com

```
www:~# openssl req -new -key myserverkey.pem -out myservercert.csr
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]:SP
State or Province Name (full name) [Some-State]:Cat
Locality Name (eg, city) []:BCN
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UPC
Organizational Unit Name (eg, section) []:TCGI
Common Name (eg, YOUR name) []:www.example.com
Email Address []:mireia

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:pass
An optional company name []:
```

• Now, we use the CA key to sign the server certificate, which creates the file myserver.crt:

$ openssl x509 -req -in myservercert.csr -CA mycacert.pem -CAkey mycakey.pem -CAcreateserial \-days 360 -out myservercert.pem

Important: use www.example.com as DN.

```
Signature ok
subject=/C=SP/ST=Cat/L=BCN/O=UPC/OU=TCGI/CN=www.example.com/emailAddress=mireia
Getting CA Private Key
Enter pass phrase for mycakey.pem:
```

• Finally, we can remove the certificate request:

$ rm myservercert.csr

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.

www: /etc/nginx/sites-available# cp default www.example.com

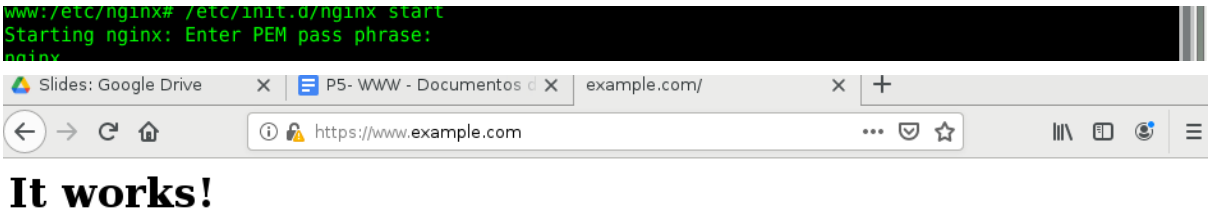
The key elements that should be included in the server section are:

```
server {
     listen 443 ssl;
     server_name www.example.com;

     ssl_certificate           path-to-ssl-certificate;
     ssl_certificate_key       path-to-ssl-key;

     ...
}
```

```
 GNU nano 2.0.7           File: www.example.com

erver {
    listen 443 ssl;
    server_name  www.example.com;

    ssl_certificate           mycacert.pem;
    ssl_certificate_key       mycakey.pem;

    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)
```

Enable the new configuration and try to access https://www.example.com.

www: /etc/nginx/sites-enabled# ln -s ../sites-available/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 results.



SimNet0:

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:

```
server {
    listen 80;

    server_name www.example.com;
    return 301 https://www.example.com$request_uri;
}
```

www: /etc/nginx/sites-available# nano https-redirection

Then disable the default configuration and enable the https-redirection one.
www: /etc/nginx/sites-enabled# rm default
www: /etc/nginx/sites-enabled# ln -s ../sites-available/https-redirection
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.
www:# /etc/init.d/nginx restart
host# lynx http://www.example.com

1	0.000000000	10.1.1.3	10.1.1.10	DNS	75 Standard query 0xd0fb AAAA www.example.c
2	0.000368307	10.1.1.10	10.1.1.3	DNS	119 Standard query response 0xd0fb AAAA www.
3	0.000774728	10.1.1.3	10.1.1.10	DNS	87 Standard query 0xceee AAAA www.example.c
4	0.001030807	10.1.1.10	10.1.1.3	DNS	131 Standard query response 0xceee No such r
5	0.001350414	10.1.1.3	10.1.1.10	DNS	75 Standard query 0x11de A www.example.com
6	0.001553142	10.1.1.10	10.1.1.3	DNS	125 Standard query response 0x11de A www.exa
7	0.025900841	10.1.1.3	10.1.1.1	TCP	74 3555 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS
8	0.026012196	10.1.1.1	10.1.1.3	TCP	74 80 → 3555 [SYN, ACK] Seq=0 Ack=1 Win=579
9	0.026127810	10.1.1.3	10.1.1.1	TCP	66 3555 → 80 [ACK] Seq=1 Ack=1 Win=5840 Len
10	0.049491754	10.1.1.3	10.1.1.1	HTTP	289 GET / HTTP/1.0
11	0.049660886	10.1.1.1	10.1.1.3	TCP	66 80 → 3555 [ACK] Seq=1 Ack=224 Win=6864 L
12	0.055859450	10.1.1.1	10.1.1.3	HTTP	429 HTTP/1.1 301 Moved Permanently (text/ht
13	0.055968259	10.1.1.3	10.1.1.1	TCP	66 3555 → 80 [ACK] Seq=224 Ack=364 Win=6912
14	0.056646473	10.1.1.1	10.1.1.3	TCP	66 80 → 3555 [FIN, ACK] Seq=364 Ack=224 Win
15	0.096442948	10.1.1.3	10.1.1.1	TCP	66 3555 → 80 [ACK] Seq=224 Ack=365 Win=6912
16	0.118221526	10.1.1.3	10.1.1.1	TCP	66 3555 → 80 [FIN, ACK] Seq=224 Ack=365 Win
17	0.118434672	10.1.1.1	10.1.1.3	TCP	66 80 → 3555 [ACK] Seq=365 Ack=225 Win=6864
18	2.185128249	10.1.1.3	10.1.1.10	DNS	75 Standard query 0x947f AAAA www.example.c
19	2.186226914	10.1.1.10	10.1.1.3	DNS	119 Standard query response 0x947f AAAA www.
20	2.187597615	10.1.1.3	10.1.1.10	DNS	87 Standard query 0x4217 AAAA www.example.c
21	2.190387076	10.1.1.10	10.1.1.3	DNS	131 Standard query response 0x4217 No such r
22	2.191399472	10.1.1.3	10.1.1.10	DNS	75 Standard query 0x3ca0 A www.example.com
23	2.192087950	10.1.1.10	10.1.1.3	DNS	125 Standard query response 0x3ca0 A www.exa
24	2.213826834	10.1.1.3	10.1.1.1	TCP	74 3085 → 443 [SYN] Seq=0 Win=5840 Len=0 MS
25	2.213971286	10.1.1.1	10.1.1.3	TCP	74 443 → 3085 [SYN, ACK] Seq=0 Ack=1 Win=57
26	2.214075473	10.1.1.3	10.1.1.1	TCP	66 3085 → 443 [ACK] Seq=1 Ack=1 Win=5840 Le
27	2.228289870	10.1.1.3	10.1.1.1	TLSv1	139 Client Hello
28	2.228428499	10.1.1.1	10.1.1.3	TCP	66 443 → 3085 [ACK] Seq=1 Ack=74 Win=5792 L
29	2.246855094	10.1.1.1	10.1.1.3	TLSv1	1325 Server Hello, Certificate, Server Hello
30	2.247011852	10.1.1.3	10.1.1.1	TCP	66 3085 → 443 [ACK] Seq=74 Ack=1260 Win=835
31	2.253178433	10.1.1.3	10.1.1.1	TLSv1	333 Client Key Exchange
32	2.290868371	10.1.1.1	10.1.1.3	TCP	66 443 → 3085 [ACK] Seq=1260 Ack=341 Win=68
33	2.291034194	10.1.1.3	10.1.1.1	TLSv1	333 Change Cipher Spec, Encrypted Handshake
34	2.291154446	10.1.1.1	10.1.1.3	TCP	66 443 → 3085 [ACK] Seq=1260 Ack=608 Win=79
35	2.292288626	10.1.1.1	10.1.1.3	TLSv1	125 Change Cipher Spec, Encrypted Handshake
36	2.335217001	10.1.1.3	10.1.1.1	TCP	66 3085 → 443 [ACK] Seq=608 Ack=1319 Win=83
37	4.997820864	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
38	4.997877841	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
39	7.462267891	10.1.1.3	10.1.1.1	TLSv1	343 Application Data
40	7.466337623	10.1.1.1	10.1.1.3	TLSv1	375 Application Data
41	7.466494602	10.1.1.3	10.1.1.1	TCP	66 3085 → 443 [ACK] Seq=885 Ack=1628 Win=83
42	7.466863073	10.1.1.1	10.1.1.3	TLSv1	103 Encrypted Alert
43	7.466976973	10.1.1.3	10.1.1.1	TCP	66 3085 → 443 [ACK] Seq=885 Ack=1665 Win=83
44	7.467992382	10.1.1.1	10.1.1.3	TCP	66 443 → 3085 [FIN, ACK] Seq=1665 Ack=885 W
45	7.507621998	10.1.1.3	10.1.1.1	TCP	66 3085 → 443 [ACK] Seq=885 Ack=1666 Win=83
46	7.535634098	10.1.1.3	10.1.1.1	TCP	66 3085 → 443 [FIN, ACK] Seq=885 Ack=1666 W
47	7.535784641	10.1.1.1	10.1.1.3	TCP	66 443 → 3085 [ACK] Seq=1666 Ack=886 Win=90

www.example.com -> 301 (redirection)->https (TLS frames)

Exercise 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:

```
server {
    listen 443 ssl http2;
    server_name www.example.com;

    ssl_certificate           path-to-ssl-certificate;
    ssl_certificate_key       path-to-ssl-key;

    ...
}
```

```
 GNU nano 2.0.7            File: www.example.com

server {
    listen       443 ssl http2;
    server_name  www.example.com;

    ssl_certificate           /etc/ssl/certs/ssl-cert-snakeoil.pem;
    ssl_certificate_key       /etc/ssl/private/ssl-cert-snakeoil.key;

    location / {
        root   /var/www;
```

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.
www: /etc/nginx/sites-enabled# rm https-redirection
www:# /etc/init.d/nginx restart
host:# lynx https://www.example.com

1	0.000000000	fe:fd:00:00:06:00	Broadcast	ARP	42 Who has 10.1.1.10? Tell 10.1.1.3
2	0.000151167	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
3	0.000274263	10.1.1.3	10.1.1.10	DNS	75 Standard query 0x0dd3 AAAA www.example.c
4	0.000629475	10.1.1.10	10.1.1.3	DNS	119 Standard query response 0x0dd3 AAAA www.
5	0.001143204	10.1.1.3	10.1.1.10	DNS	87 Standard query 0xb16a AAAA www.example.c
6	0.001380626	10.1.1.10	10.1.1.3	DNS	131 Standard query response 0xb16a No such r
7	0.001738732	10.1.1.3	10.1.1.10	DNS	75 Standard query 0xaf14 A www.example.com
8	0.001961124	10.1.1.10	10.1.1.3	DNS	125 Standard query response 0xaf14 A www.exa
9	0.023378237	10.1.1.3	10.1.1.1	TCP	74 4372 → 443 [SYN] Seq=0 Win=5840 Len=0 MS
10	0.036851472	fe:fd:00:00:01:00	Broadcast	ARP	42 Who has 10.1.1.3? Tell 10.1.1.1
11	0.037013718	fe:fd:00:00:06:00	fe:fd:00:00:01:00	ARP	42 10.1.1.3 is at fe:fd:00:00:06:00
12	0.037082316	10.1.1.1	10.1.1.3	TCP	74 443 → 4372 [SYN, ACK] Seq=0 Ack=1 Win=57
13	0.037201698	10.1.1.3	10.1.1.1	TCP	66 4372 → 443 [ACK] Seq=1 Ack=1 Win=5840 Le
14	0.084835481	10.1.1.3	10.1.1.1	TLSv1	139 Client Hello
15	0.084991421	10.1.1.1	10.1.1.3	TCP	66 443 → 4372 [ACK] Seq=1 Ack=74 Win=5792 L
16	0.099868110	10.1.1.1	10.1.1.3	TLSv1	582 Server Hello, Certificate, Server Hello
17	0.100000263	10.1.1.3	10.1.1.1	TCP	66 4372 → 443 [ACK] Seq=74 Ack=517 Win=6912
18	0.104333871	10.1.1.3	10.1.1.1	TLSv1	205 Client Key Exchange
19	0.135616883	10.1.1.1	10.1.1.3	TCP	66 443 → 4372 [ACK] Seq=517 Ack=213 Win=579
20	0.135950425	10.1.1.3	10.1.1.1	TLSv1	173 Change Cipher Spec, Encrypted Handshake
21	0.136306577	10.1.1.1	10.1.1.3	TCP	66 443 → 4372 [ACK] Seq=517 Ack=320 Win=579
22	0.141015407	10.1.1.1	10.1.1.3	TLSv1	125 Change Cipher Spec, Encrypted Handshake
23	0.185752639	10.1.1.3	10.1.1.1	TCP	66 4372 → 443 [ACK] Seq=320 Ack=576 Win=691
24	5.010544300	fe:fd:00:00:05:00	fe:fd:00:00:06:00	ARP	42 Who has 10.1.1.3? Tell 10.1.1.10
25	5.010826338	fe:fd:00:00:06:00	fe:fd:00:00:05:00	ARP	42 10.1.1.3 is at fe:fd:00:00:06:00
26	9.258765749	10.1.1.3	10.1.1.1	TLSv1	391 Application Data
27	9.266461600	10.1.1.1	10.1.1.3	TLSv1	375 Application Data
28	9.266574814	10.1.1.3	10.1.1.1	TCP	66 4372 → 443 [ACK] Seq=645 Ack=885 Win=798
29	9.267763903	10.1.1.1	10.1.1.3	TLSv1	103 Encrypted Alert
30	9.267876846	10.1.1.3	10.1.1.1	TCP	66 4372 → 443 [ACK] Seq=645 Ack=922 Win=798
31	9.268623208	10.1.1.1	10.1.1.3	TCP	66 443 → 4372 [FIN, ACK] Seq=922 Ack=645 Wi
32	9.319992792	10.1.1.3	10.1.1.1	TCP	66 4372 → 443 [ACK] Seq=645 Ack=923 Win=798
33	9.330351198	10.1.1.3	10.1.1.1	TCP	66 4372 → 443 [FIN, ACK] Seq=645 Ack=923 Wi
34	9.330498834	10.1.1.1	10.1.1.3	TCP	66 443 → 4372 [ACK] Seq=923 Ack=646 Win=686

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.
In the application layer protocol negotiation section (ALPN): next protocol h2 -> It supports http2