

Computer Networks Lab

Week 2

Vishal R

PES1UG19CS571

I Section

February 7, 2021

Understanding Persistent and Non-Persistent HTTP Connections

1 Apache Server Configurations

Creation of client-server architecture required to setup two Virtual Machines, where one acts as a client and another acts as a server.

To create the server, **Apache** was installed and configured on one of the Virtual Machines. This Virtual Machine will act as server.

1.1 Installing Apache.

Apache was installed by typing the command `$ sudo apt-get install apache2`. This will install **Apache** on the Virtual Machine and will automatically start the server for us.

We can check if the server has been successfully started or not by typing the command `$ sudo systemctl status apache2` or `$ sudo service apache2 status` in the terminal.

```
vishalr@ubuntu:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2021-02-06 20:59:28 PST; 1h 11min ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 25581 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
  Main PID: 25585 (apache2)
    Tasks: 55 (limit: 2882)
   Memory: 5.8M
   CGroup: /system.slice/apache2.service
           └─25585 /usr/sbin/apache2 -k start
             └─25586 /usr/sbin/apache2 -k start
               └─25587 /usr/sbin/apache2 -k start

Feb 06 20:59:28 ubuntu systemd[1]: Starting The Apache HTTP Server...
Feb 06 20:59:28 ubuntu systemd[1]: Started The Apache HTTP Server.
vishalr@ubuntu:~$
```

Figure 1: Shows the current status of apache.

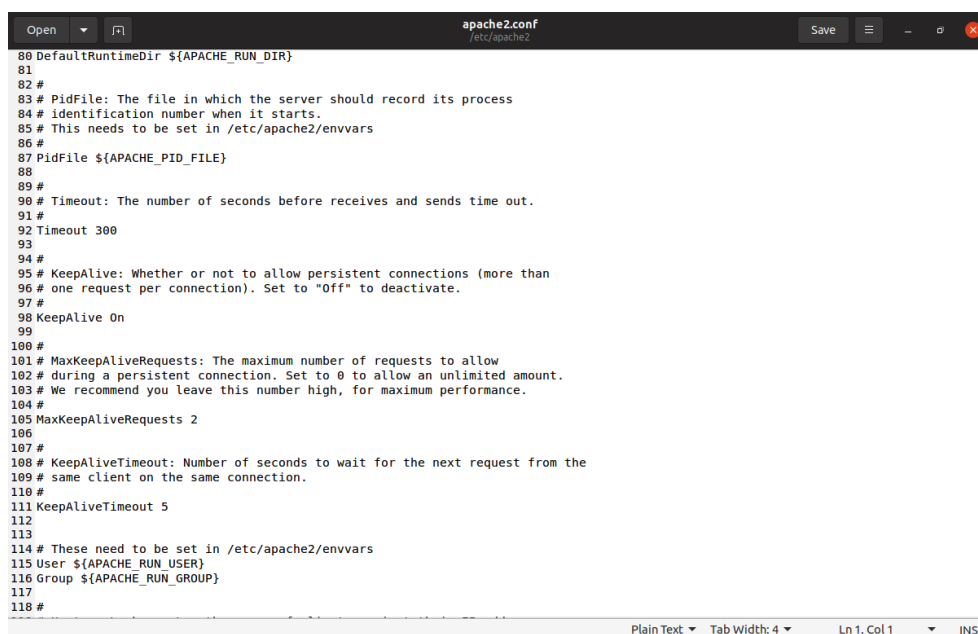
1.2 Additional Apache server configurations.

apache2.conf in /etc/apache2/ was opened using `$ sudo gedit /etc/apache2/apache2.conf`.

Note : This file can also be opened using `$ sudo nano /etc/apache2/apache2.conf`

Following changes were made to setup our server for this lab..

- KeepAlive option was set. (i.e value was made ON).
- MaximumKeepAliveRequests was set to 2.



```
Open  apache2.conf  Save  [Icons]
80 DefaultRuntimeDir ${APACHE_RUN_DIR}
81
82 #
83 # PidFile: The file in which the server should record its process
84 # identification number when it starts.
85 # This needs to be set in /etc/apache2/envvars
86 #
87 PidFile ${APACHE_PID_FILE}
88
89 #
90 # Timeout: The number of seconds before receives and sends time out.
91 #
92 Timeout 300
93
94 #
95 # KeepAlive: Whether or not to allow persistent connections (more than
96 # one request per connection). Set to "Off" to deactivate.
97 #
98 KeepAlive On
99
100 #
101 # MaxKeepAliveRequests: The maximum number of requests to allow
102 # during a persistent connection. Set to 0 to allow an unlimited amount.
103 # We recommend you leave this number high, for maximum performance.
104 #
105 MaxKeepAliveRequests 2
106
107 #
108 # KeepAliveTimeout: Number of seconds to wait for the next request from the
109 # same client on the same connection.
110 #
111 KeepAliveTimeout 5
112
113
114 # These need to be set in /etc/apache2/envvars
115 User ${APACHE_RUN_USER}
116 Group ${APACHE_RUN_GROUP}
117
118 #
```

Figure 2: Shows the changes made in apache2.conf file

1.3 Setting the IP Addresses of both client and server Virtual Machines.

A new IP Address was set on both client and server Virtual Machines. These IP Addresses were set manually by going into 'edit connections' and manually entering IP Address for client and server. This was done in order to avoid any fluctuations that occur while assigning IP address.

Note : IP Address can also be set by using the command `$ sudo ip addr add ip-address dev interface-name`.

1.3.1 Assigning IP Address in Server Machine.

IP address assigned to the server is 10.0.9.71/24 as per my SRN.

The screenshot shows the 'Wired' connection settings in NetworkManager. The 'IPv4' tab is active, displaying the 'Automatic (DHCP)' method. Under the 'Addresses' section, a table lists the assigned IP address '10.0.9.71' with a netmask of '24'. The 'DNS' section shows the 'Automatic' toggle is enabled.

Figure 3: Manually setting IP address on server machine.

We can verify this by typing the command `$ sudo ip addr show` in the terminal.

```
vishalr@ubuntu:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:a9:22:d8 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 10.0.9.71/24 brd 10.0.9.255 scope global noprefixroute ens33
        valid_lft forever preferred_lft forever
    inet6 fd15:4ba5:5a2b:1008:661f:b710:2b75:144a/64 scope global temporary dynamic
        valid_lft 592223sec preferred_lft 73777sec
    inet6 fd15:4ba5:5a2b:1008:9c14:3ddf:2966:f3ef/64 scope global dynamic mngtppaddr noprefixroute
        valid_lft 2591951sec preferred_lft 604751sec
    inet6 fe80::db93:c4b:f387:18f8/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
vishalr@ubuntu:~$
```

Figure 4: Verifying if IP Address has been set successfully on server machine.

1.3.2 Assigning IP Address in Client Machine.

IP address assigned to the server is 10.0.9.38/24 as per my serial number.

The screenshot shows the NetworkManager configuration window for a wired connection. The 'IPv4' tab is active. The 'IPv4 Method' is set to 'Manual'. The 'Addresses' table contains one entry: Address '10.0.9.38' and Netmask '24'. The 'DNS' toggle is turned on to 'Automatic'.

Figure 5: Manually setting IP address on client machine.

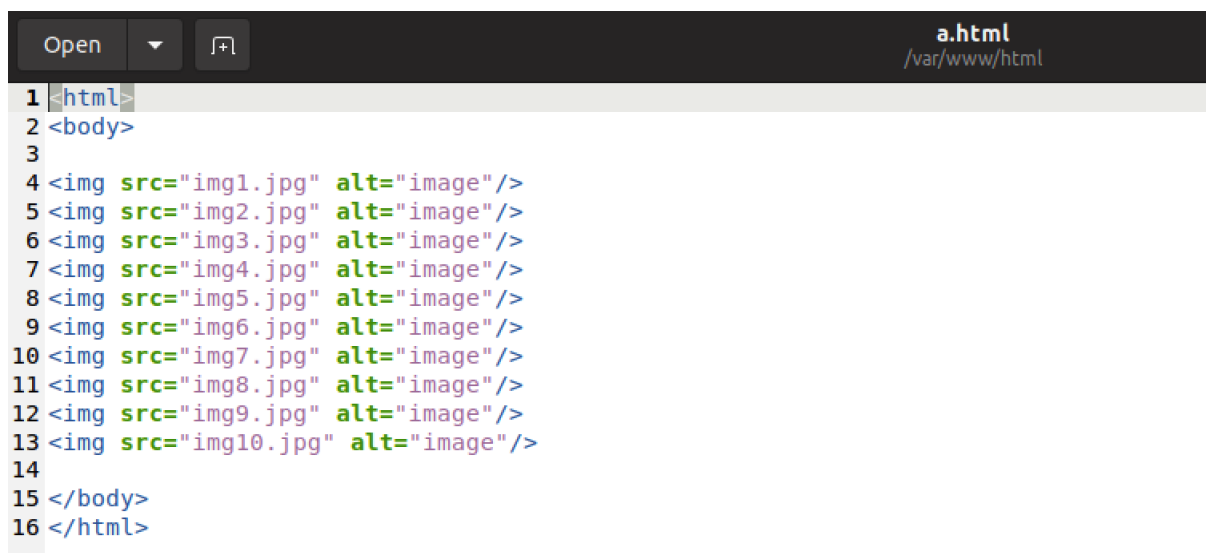
We can verify this by typing the command `$ sudo ip addr show` in the terminal.

```
vishal@ubuntu:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:8f:95:84 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 10.0.9.38/24 brd 10.0.9.255 scope global noprefixroute ens33
        valid_lft forever preferred_lft forever
    inet6 fd15:4ba5:5a2b:1008:5393:ccdf:7898:73dc/64 scope global temporary dynamic
        valid_lft 592028sec preferred_lft 73230sec
    inet6 fd15:4ba5:5a2b:1008:f2b1:28a:88ef:1731/64 scope global dynamic mngtmpaddr noprefixroute
        valid_lft 2591938sec preferred_lft 604738sec
    inet6 fe80::b2a9:ab36:a543:56a4/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
vishal@ubuntu:~$
```

Figure 6: Verifying if IP Address has been set successfully on client machine.

1.4 Creating a web page in server machine

A new .html file was created in the directory `/var/www/html/` and the following contents were added into it.



```

1 <html>
2 <body>
3
4 
5 
6 
7 
8 
9 
10 
11 
12 
13 
14
15 </body>
16 </html>

```

Figure 7: Contents of a.html.

1.5 Adding 10 images into the same directory.

10 images were downloaded from internet and were placed in the same directory where a.html is present. Images are of sizes 1.0 mb - 1.5 mb .

The images were renamed to (img1.jpg, img2.jpg, img3.jpg, ..., img10.jpg).

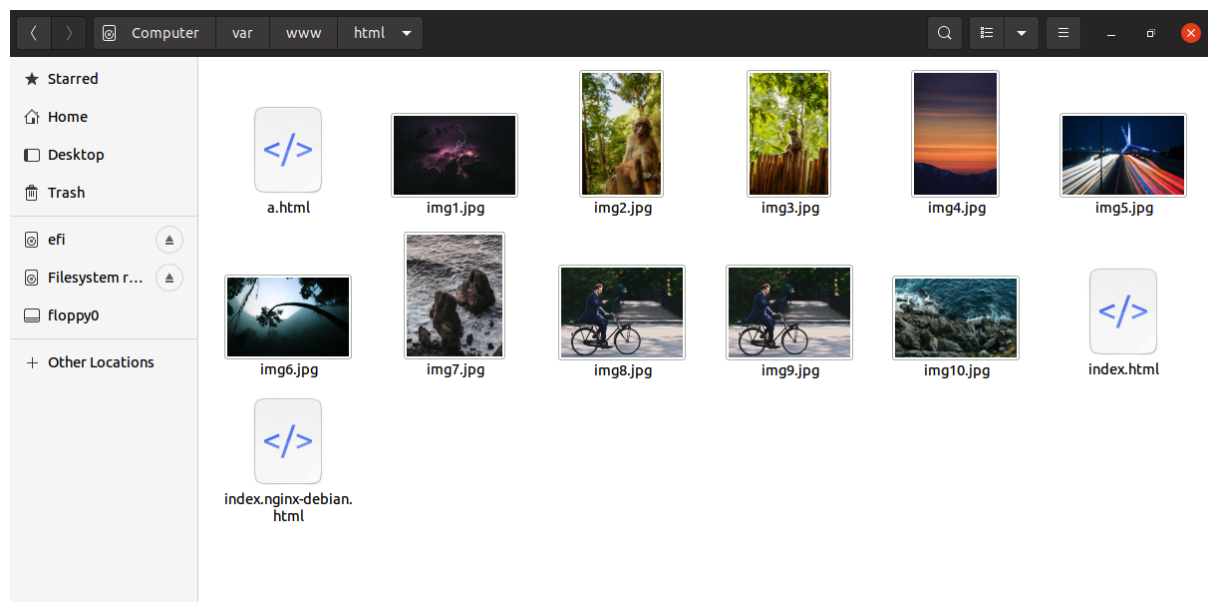
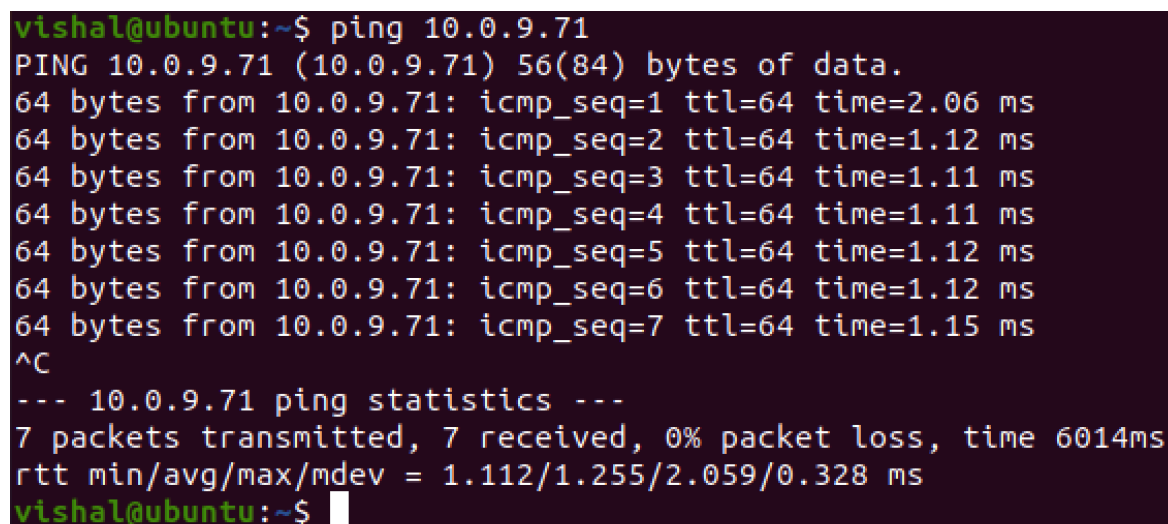


Figure 8: Contents of the folder /var/www/html/.

1.6 Testing connectivity between client and server

To test if client and server can communicate with each other, we can ping the server from the client using the command `ping 10.0.9.71`.



```
vishal@ubuntu:~$ ping 10.0.9.71
PING 10.0.9.71 (10.0.9.71) 56(84) bytes of data.
64 bytes from 10.0.9.71: icmp_seq=1 ttl=64 time=2.06 ms
64 bytes from 10.0.9.71: icmp_seq=2 ttl=64 time=1.12 ms
64 bytes from 10.0.9.71: icmp_seq=3 ttl=64 time=1.11 ms
64 bytes from 10.0.9.71: icmp_seq=4 ttl=64 time=1.11 ms
64 bytes from 10.0.9.71: icmp_seq=5 ttl=64 time=1.12 ms
64 bytes from 10.0.9.71: icmp_seq=6 ttl=64 time=1.12 ms
64 bytes from 10.0.9.71: icmp_seq=7 ttl=64 time=1.15 ms
^C
--- 10.0.9.71 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6014ms
rtt min/avg/max/mdev = 1.112/1.255/2.059/0.328 ms
vishal@ubuntu:~$
```

Figure 9: Checking connectivity between client and server using ping.

We can see that we are receiving packets from server hence we can conclude that both client and server can communicate with each other.

After all these steps, we are done with configuring the apache server and now we can start analysing persistent and non-persistent connections.

2 Non-Persistent HTTP Connection

A Non-persistent connection is a type of HTTP connection which is closed after the server sends the requested object to client. Hence, if the client requests more than one object, the connection is opened and closed for each object that the client requests for.

2.1 Setting up Non-Persistent HTTP Connection

To setup a non-persistent connection, few configurations must be changed in the browser's config files on the client side.

For this experiment, I have used Firefox browser.

In Firefox, we will type `about:config` in the search bar.

We will search for 'persistent' in the filter options. We then set the following options in the browser.

- `max-persistent-connections-per-server` was set to 0.
- `persistent-settings` was set to false.

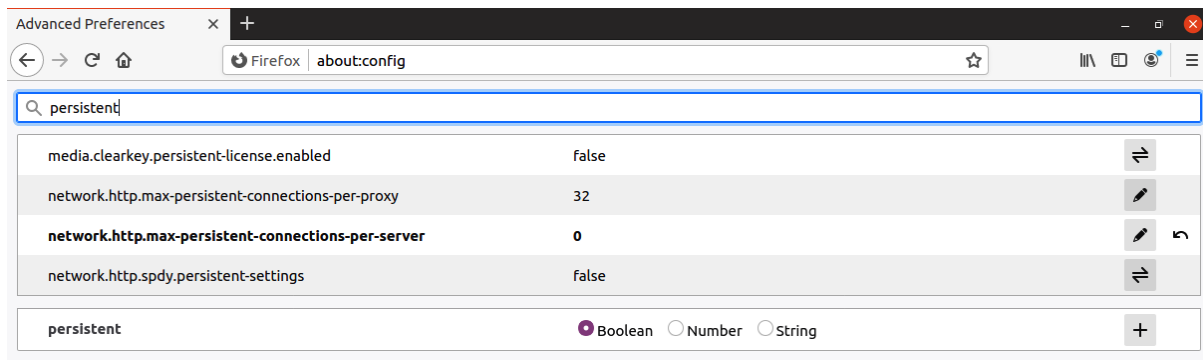


Figure 10: Configuring the browser for non-persistent connections.

Now, we can request for the `html` file we created in server from the client.

2.2 Make GET request from the Client to Server

We can request for the `a.html` file we created on server from the client by typing `http://10.0.9.71/a.html` in the browser search bar. This will load the web page from the server. Before making the GET request, we will open Wireshark and monitor 'any' interface for HTTP Packets using the `http` filter option.

The screenshot shows a Wireshark packet capture on the 'any' interface, filtered by 'http'. The packet list shows the following data:

No.	Time	Source	Destination	Protocol	Length	Info
4	0.001162052	10.0.9.38	10.0.9.71	HTTP	410	GET /a.html HTTP/1.1
6	0.003827781	10.0.9.71	10.0.9.38	HTTP	444	HTTP/1.1 200 OK (text/html)
22	0.399022561	10.0.9.38	10.0.9.71	HTTP	356	GET /img1.jpg HTTP/1.1
190	0.472774837	10.0.9.71	10.0.9.38	HTTP	14657	HTTP/1.1 200 OK (JPEG JFIF image)
192	0.473490618	10.0.9.38	10.0.9.71	HTTP	356	GET /img2.jpg HTTP/1.1
253	0.514213245	10.0.9.71	10.0.9.38	HTTP	16728	HTTP/1.1 200 OK (JPEG JFIF image)
255	0.514719295	10.0.9.38	10.0.9.71	HTTP	356	GET /img3.jpg HTTP/1.1
328	0.556551850	10.0.9.71	10.0.9.38	HTTP	11658	HTTP/1.1 200 OK (JPEG JFIF image)
330	0.564836189	10.0.9.38	10.0.9.71	HTTP	356	GET /img4.jpg HTTP/1.1
409	0.587192736	10.0.9.71	10.0.9.38	HTTP	24271	HTTP/1.1 200 OK (JPEG JFIF image)
411	0.610375016	10.0.9.38	10.0.9.71	HTTP	356	GET /img5.jpg HTTP/1.1
461	0.645193009	10.0.9.71	10.0.9.38	HTTP	5274	HTTP/1.1 200 OK (JPEG JFIF image)
463	0.651575495	10.0.9.38	10.0.9.71	HTTP	356	GET /img6.jpg HTTP/1.1
538	0.711284346	10.0.9.71	10.0.9.38	HTTP	20727	HTTP/1.1 200 OK (JPEG JFIF image)
540	0.894068908	10.0.9.38	10.0.9.71	HTTP	356	GET /img7.jpg HTTP/1.1
601	0.964016858	10.0.9.71	10.0.9.38	HTTP	27781	HTTP/1.1 200 OK (JPEG JFIF image)
605	1.025767736	10.0.9.38	10.0.9.71	HTTP	356	GET /img8.jpg HTTP/1.1
683	1.053879698	10.0.9.71	10.0.9.38	HTTP	2201	HTTP/1.1 200 OK (JPEG JFIF image)
686	1.115886360	10.0.9.38	10.0.9.71	HTTP	356	GET /img9.jpg HTTP/1.1
745	1.156638273	10.0.9.71	10.0.9.38	HTTP	9441	HTTP/1.1 200 OK (JPEG JFIF image)
747	1.167691176	10.0.9.38	10.0.9.71	HTTP	357	GET /img10.jpg HTTP/1.1
798	1.207097867	10.0.9.71	10.0.9.38	HTTP	412	HTTP/1.1 200 OK (JPEG JFIF image)
800	1.234004717	10.0.9.38	10.0.9.71	HTTP	359	GET /favicon.ico HTTP/1.1
801	1.234980070	10.0.9.71	10.0.9.38	HTTP	397	HTTP/1.1 404 Not Found (text/html)

Figure 11: Packet Capture in Wireshark.

We can calculate the total time taken to load all the 10 images by taking the time difference between the two highlighted packets in the figure.

Note : The last packet highlighted is wrong. Please consider packet above it.

Total Time : $1.207097867 - 0.001162052 = 1.205935815$ s.

3 Persistent HTTP Connections

In non-persistent connections, the server will close the connection after the requested object is sent to client. With Persistent connections, the server leaves the connection open and hence retrieving more objects from server need not require opening another connection.

3.1 Setting up Persistent HTTP Connections

To setup a persistent connection, few configurations must be changed in the browser's config files on the client side.

In firefox, we will type `about:config` in the search bar.

We will search for 'persistent' in the filter options. We then set the following options in the browser.

- `max-persistent-connections-per-server` was set to `<number>`.
- `persistent-settings` was set to `true`.

Here `<number>` can be any integer which denotes the maximum number of persistent connections for a server.

3.2 2 Persistent HTTP Connection

For 2 persistent HTTP connection, we will set `<number>` to 2 as shown below.

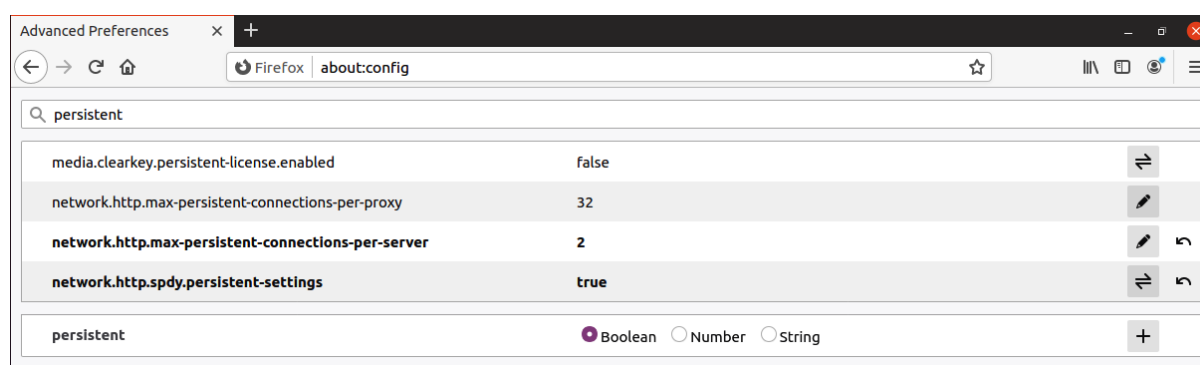


Figure 12: Configuring web browser for 2 persistent connections.

Again can request for the `a.html` file we created on server from the client by typing `http://10.0.9.71/a.html` in the browser search bar. This will load the web page from the server.

Before making the GET request, we will open wireshark and monitor 'any' interface for HTTP Packets using the `http` filter option.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.001480212	10.0.9.38	10.0.9.71	HTTP	410	GET /a.html HTTP/1.1
6	0.003449974	10.0.9.71	10.0.9.38	HTTP	444	HTTP/1.1 200 OK (text/html)
30	0.258315780	10.0.9.38	10.0.9.71	HTTP	356	GET /img1.jpg HTTP/1.1
121	0.296067280	10.0.9.38	10.0.9.71	HTTP	356	GET /img2.jpg HTTP/1.1
282	0.342662044	10.0.9.71	10.0.9.38	HTTP	28312	HTTP/1.1 200 OK (JPEG JFIF image)
317	0.359925770	10.0.9.71	10.0.9.38	HTTP	20449	HTTP/1.1 200 OK (JPEG JFIF image)
319	0.363268684	10.0.9.38	10.0.9.71	HTTP	356	GET /img3.jpg HTTP/1.1
382	0.410992756	10.0.9.38	10.0.9.71	HTTP	356	GET /img4.jpg HTTP/1.1
477	0.469083667	10.0.9.71	10.0.9.38	HTTP	4418	HTTP/1.1 200 OK (JPEG JFIF image)
484	0.479368732	10.0.9.71	10.0.9.38	HTTP	8343	HTTP/1.1 200 OK (JPEG JFIF image)
486	0.485470998	10.0.9.38	10.0.9.71	HTTP	356	GET /img5.jpg HTTP/1.1
612	0.555123276	10.0.9.38	10.0.9.71	HTTP	356	GET /img6.jpg HTTP/1.1
703	0.592997947	10.0.9.71	10.0.9.38	HTTP	15410	HTTP/1.1 200 OK (JPEG JFIF image)
706	0.593418798	10.0.9.38	10.0.9.71	HTTP	356	GET /img7.jpg HTTP/1.1
806	0.667354296	10.0.9.71	10.0.9.38	HTTP	1717	HTTP/1.1 200 OK (JPEG JFIF image)
839	0.723605694	10.0.9.71	10.0.9.38	HTTP	13487	HTTP/1.1 200 OK (JPEG JFIF image)
841	0.847191938	10.0.9.38	10.0.9.71	HTTP	356	GET /img8.jpg HTTP/1.1
933	0.937436481	10.0.9.38	10.0.9.71	HTTP	356	GET /img9.jpg HTTP/1.1
1036	0.965306923	10.0.9.71	10.0.9.38	HTTP	2201	HTTP/1.1 200 OK (JPEG JFIF image)
1045	0.977517221	10.0.9.71	10.0.9.38	HTTP	23921	HTTP/1.1 200 OK (JPEG JFIF image)
1049	1.018338979	10.0.9.38	10.0.9.71	HTTP	357	GET /img10.jpg HTTP/1.1
1139	1.087106537	10.0.9.71	10.0.9.38	HTTP	13444	HTTP/1.1 200 OK (JPEG JFIF image)
1141	1.144477772	10.0.9.38	10.0.9.71	HTTP	359	GET /favicon.ico HTTP/1.1
1142	1.145943568	10.0.9.71	10.0.9.38	HTTP	397	HTTP/1.1 404 Not Found (text/html)

Figure 13: Packet capture in Wireshark.

We can calculate the total time taken to load all the 10 images by taking the time difference between the two highlighted packets in the figure.

Total Time : $1.087106537 - 0.001480212 = 1.085626325$ s.

3.3 4 Persistent HTTP Connection

For 4 persistent HTTP connection, we will set <number> to 4 as shown below.

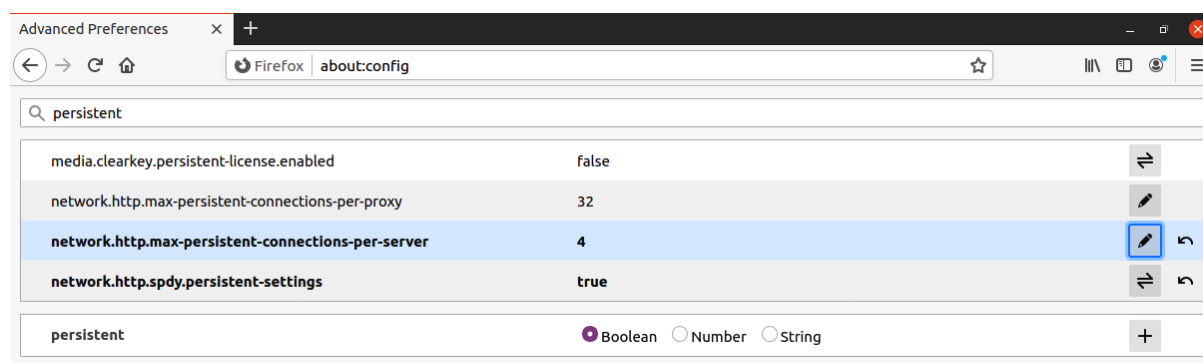


Figure 14: Configuring web browser for 4 persistent connections.

We will make GET request to same url again. Before making the GET request, we will open wireshark and monitor 'any' interface for HTTP Packets using the `http` filter option.

No.	Time	Source	Destination	Protocol	Length	Info
6	0.008082075	10.0.9.38	10.0.9.71	HTTP	410	GET /a.html HTTP/1.1
8	0.009197382	10.0.9.71	10.0.9.38	HTTP	444	HTTP/1.1 200 OK (text/html)
26	0.419398213	10.0.9.38	10.0.9.71	HTTP	356	GET /img1.jpg HTTP/1.1
128	0.483351537	10.0.9.38	10.0.9.71	HTTP	356	GET /img2.jpg HTTP/1.1
145	0.486467703	10.0.9.38	10.0.9.71	HTTP	356	GET /img4.jpg HTTP/1.1
168	0.490766469	10.0.9.38	10.0.9.71	HTTP	356	GET /img3.jpg HTTP/1.1
438	0.550799184	10.0.9.71	10.0.9.38	HTTP	19624	HTTP/1.1 200 OK (JPEG JFIF image)
508	0.559593165	10.0.9.38	10.0.9.71	HTTP	356	GET /img5.jpg HTTP/1.1
804	0.621074706	10.0.9.71	10.0.9.38	HTTP	2551	HTTP/1.1 200 OK (JPEG JFIF image)
872	0.636311646	10.0.9.71	10.0.9.38	HTTP	11658	HTTP/1.1 200 OK (JPEG JFIF image)
879	0.637262286	10.0.9.71	10.0.9.38	HTTP	7417	HTTP/1.1 200 OK (JPEG JFIF image)
892	0.644696077	10.0.9.38	10.0.9.71	HTTP	356	GET /img6.jpg HTTP/1.1
893	0.646972960	10.0.9.38	10.0.9.71	HTTP	356	GET /img7.jpg HTTP/1.1
1018	0.686226220	10.0.9.71	10.0.9.38	HTTP	5274	HTTP/1.1 200 OK (JPEG JFIF image)
1073	0.720982240	10.0.9.38	10.0.9.71	HTTP	356	GET /img8.jpg HTTP/1.1
1194	0.788027600	10.0.9.38	10.0.9.71	HTTP	356	GET /img9.jpg HTTP/1.1
1363	0.838224884	10.0.9.71	10.0.9.38	HTTP	5097	HTTP/1.1 200 OK (JPEG JFIF image)
1401	0.852496618	10.0.9.71	10.0.9.38	HTTP	10591	HTTP/1.1 200 OK (JPEG JFIF image)
1419	0.867603719	10.0.9.71	10.0.9.38	HTTP	3165	HTTP/1.1 200 OK (JPEG JFIF image)
1422	0.871894950	10.0.9.38	10.0.9.71	HTTP	357	GET /img10.jpg HTTP/1.1
1531	0.942043293	10.0.9.71	10.0.9.38	HTTP	753	HTTP/1.1 200 OK (JPEG JFIF image)
1560	0.977125748	10.0.9.71	10.0.9.38	HTTP	6204	HTTP/1.1 200 OK (JPEG JFIF image)
1562	1.067096174	10.0.9.38	10.0.9.71	HTTP	359	GET /favicon.ico HTTP/1.1
1564	1.069196759	10.0.9.71	10.0.9.38	HTTP	397	HTTP/1.1 404 Not Found (text/html)

Figure 15: Packet capture in Wireshark.

We can calculate the total time taken to load all the 10 images by taking the time difference between the two highlighted packets in the figure.

Total Time : $0.977125748 - 0.008082075 = 0.969043673$ s.

3.4 6 Persistent HTTP Connection

For 6 persistent HTTP connection, we will set <number> to 6 as shown below.

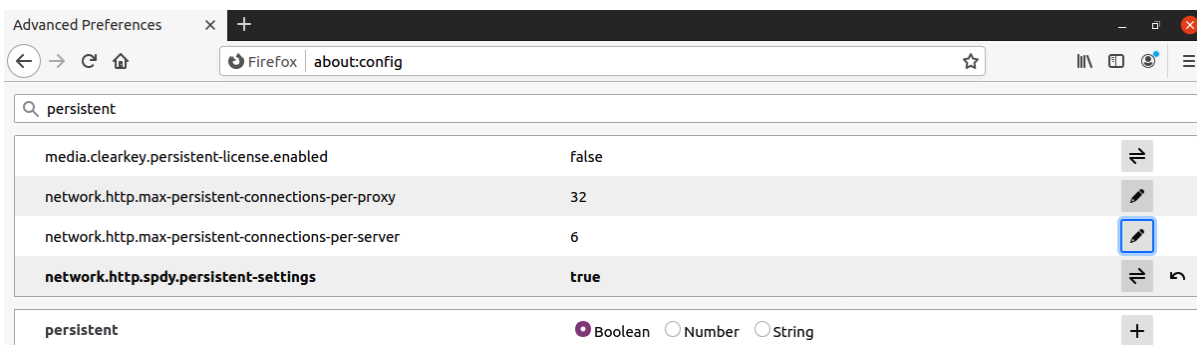


Figure 16: Configuring web browser for 6 persistent connections.

We will make GET request to same url again. Before making the GET request, we will open wireshark and monitor 'any' interface for HTTP Packets using the http filter option.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.000994922	10.0.9.38	10.0.9.71	HTTP	410	GET /a.html HTTP/1.1
6	0.004106615	10.0.9.71	10.0.9.38	HTTP	444	HTTP/1.1 200 OK (text/html)
30	0.189490258	10.0.9.38	10.0.9.71	HTTP	356	GET /img1.jpg HTTP/1.1
99	0.228770973	10.0.9.38	10.0.9.71	HTTP	356	GET /img2.jpg HTTP/1.1
108	0.228953251	10.0.9.38	10.0.9.71	HTTP	356	GET /img3.jpg HTTP/1.1
109	0.229052466	10.0.9.38	10.0.9.71	HTTP	356	GET /img4.jpg HTTP/1.1
120	0.232046595	10.0.9.38	10.0.9.71	HTTP	356	GET /img5.jpg HTTP/1.1
121	0.232221756	10.0.9.38	10.0.9.71	HTTP	356	GET /img6.jpg HTTP/1.1
514	0.303134825	10.0.9.71	10.0.9.38	HTTP	29760	HTTP/1.1 200 OK (JPEG JFIF image)
606	0.319189007	10.0.9.71	10.0.9.38	HTTP	24098	HTTP/1.1 200 OK (JPEG JFIF image)
846	0.338325003	10.0.9.71	10.0.9.38	HTTP	2551	HTTP/1.1 200 OK (JPEG JFIF image)
925	0.359518272	10.0.9.71	10.0.9.38	HTTP	7417	HTTP/1.1 200 OK (JPEG JFIF image)
940	0.353848468	10.0.9.71	10.0.9.38	HTTP	11658	HTTP/1.1 200 OK (JPEG JFIF image)
954	0.363447844	10.0.9.38	10.0.9.71	HTTP	356	GET /img7.jpg HTTP/1.1
955	0.364504562	10.0.9.71	10.0.9.38	HTTP	19279	HTTP/1.1 200 OK (JPEG JFIF image)
973	0.369555313	10.0.9.38	10.0.9.71	HTTP	356	GET /img8.jpg HTTP/1.1
1003	0.382851773	10.0.9.71	10.0.9.38	HTTP	269	HTTP/1.1 200 OK (JPEG JFIF image)
1045	0.392321965	10.0.9.38	10.0.9.71	HTTP	356	GET /img9.jpg HTTP/1.1
1139	0.432769445	10.0.9.38	10.0.9.71	HTTP	357	GET /img10.jpg HTTP/1.1
1198	0.456984878	10.0.9.71	10.0.9.38	HTTP	39508	HTTP/1.1 200 OK (JPEG JFIF image)
1230	0.491155775	10.0.9.71	10.0.9.38	HTTP	10889	HTTP/1.1 200 OK (JPEG JFIF image)
1239	0.503004952	10.0.9.71	10.0.9.38	HTTP	15233	HTTP/1.1 200 OK (JPEG JFIF image)
1244	0.656451425	10.0.9.38	10.0.9.71	HTTP	359	GET /favicon.ico HTTP/1.1
1246	0.658493148	10.0.9.71	10.0.9.38	HTTP	397	HTTP/1.1 404 Not Found (text/html)

Figure 17: Packet capture in Wireshark.

We can calculate the total time taken to load all the 10 images by taking the time difference between the two highlighted packets in the figure.

Total Time : $0.503004952 - 0.000994922 = 0.50201003$ s.

3.5 10 Persistent HTTP Connection

For 10 persistent HTTP connection, we will set <number> to 10 as shown below.

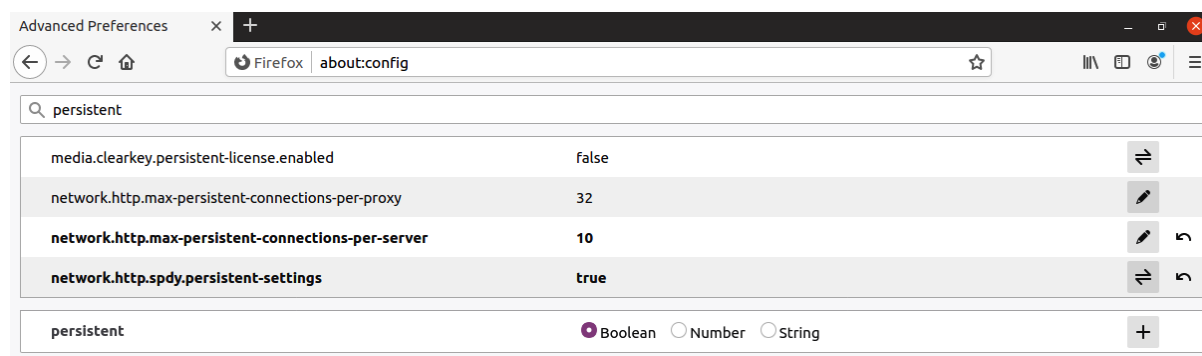


Figure 18: Configuring web browser for 10 persistent connections.

We will make GET request to same url again. Before making the GET request, we will open wireshark and monitor 'any' interface for HTTP Packets using the `http` filter option.

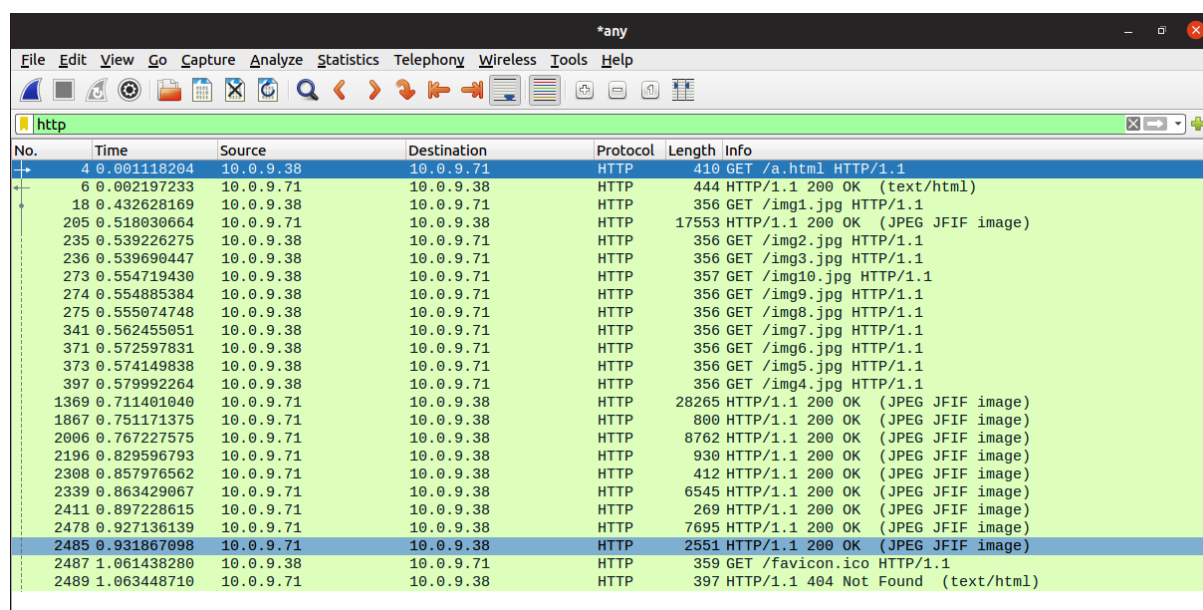


Figure 19: Packet capture in Wireshark.

We can calculate the total time taken to load all the 10 images by taking the time difference between the two highlighted packets in the figure.

Total Time : $0.931867098 - 0.001118204 = 0.930748894$ s.

4 Observations

Total time calculated is the time difference between the first **GET** request made to server and the last response from the server. By default, time displayed by wireshark is in seconds. Hence we will use seconds itself. Total time here indicates the total load time, basically the amount of time required to load all resources in the web page.

We calculated the total time difference for Non-persistent connection and for 2,4,6,10 persistent connections.

Connection Type	Total Time
Non-Persistent	1.205935815 s
2 Persistent	1.085626325 s
4 Persistent	0.969043673 s
6 Persistent	0.50201003 s
10 Persistent	0.930748894 s

We can notice that when using a non-persistent connection, there is a high load time where as if when we switched over to persistent connection, we were able to reduce the load time. The main reason is that in non-persistent connection, the connection gets closed after a response from the server. Thus additional time is required to open a new connection and hence we see high load time in non-persistent connection.

Persistent connection have lower load time because the connection is left open and additional operations like pipelining and parallelism will be performed thus having higher throughput.

From the table above, we can note that the **optimal number of persistent connection is 6** as it has the least load time. When we increased to 10 persistent connections, load time increased and it increases more as we increase the number of persistent connections further. This is because of decrease in throughput of each connection.

By default, most web browsers use 6 persistent connections and increasing it further will not yield better results.