The Web, DHCP, NTP and NAT[1]

HARDNESS : 9/10

University of Tehran
School of Electrical and Computer Engineering

دانشگاه تهران

دانشکده‌ی مهندسی برق و کامپیوتر

Computer Network Lab

آزمایشگاه شبکه‌های کامپیوتری

Professor:
Dr. Ahmad Khonsari
دکتر احمد خونساری
a_khonsari@ut.ac.ir

Amir Haji Ali Khamseh'i
امیر حاجی‌علی‌خمسه‌ء
khamse@ut.ac.ir

Reza Sharifnia
رضا شریف نیا
Reza.sharifnia@ut.ac.ir

Muhammad Borhani
محمد برهانی
borhani.m@ut.ac.ir

AmirAhmad Khordadi
امیراحمد خردادی
a.a.khordadi@ut.ac.ir

Sina Kashipazha
سینا کاشی‌پزها
sina_kashipazha@ut.ac.ir

Hadi Safari
هادی صفری
hadi.safari@ut.ac.ir

December 13, 2021

۲۲ آذر ۱۴۰۰

[1]S. Panwar, S. Mao, J.-dong Ryoo, and Y. Li, “The Web, DHCP, NTP and NAT,” in TCP/IP Essentials: A Lab-Based Approach, Cambridge: Cambridge University Press, 2004, pp. 159–186.

Objectives

- The HyperText Transfer Protocol (HTTP) and the Apache web server.
- The Common Gateway Interface (CGI) and other web-server app (like PHP, Ruby, Python, . . .).
- The Dynamic Host Configuration Protocol (DHCP).
- The Network Time Protocol (NTP).
- The Network Address Translator (NAT) and the Port Address Translator (PAT).
- An introduction to socket programming.

Part I

HTTP Exercises

For the exercises in this section, the network topology is simple with two hosts are connected to a single network segment with IP addresses, i.e. from 128.238.66.100 to 128.238.66.101. You can use Figure 3.server.gui.

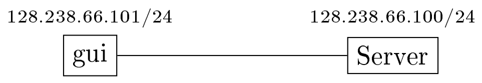

Figure 1: One gui and one terminal host (Server) connected directly. (Figure 3.server.gui)

1 HTTP Server

Examine the various configuration directives used and the corresponding settings[1].

Start the Apache server on *Server* by executing:

Server's Console

```
service apache2 start
```

In order to check if the server is working properly, you may start a *Mozilla* web browser to download the test page at `http://server.netlab/`.

Note: You should open *gui*'s Console. This will open a VNC page in web browser of your operating system (by default, Mozilla Firefox). Then, from the start menu of the GUI, open *Mozilla Firefox* and enter the *Server*'s IP address in the URL box:

gui's web browser

```
http://128.238.66.100/
```

Then, execute the following command in *Server* to list the process IDs of the `apache2` processes started:

Server's Console

```
pgrep apache2
```

Open the Multi-Processing Module config file with a text editor and study its various configurations and their applications by executing:

[1] `/etc/apache2/apache2.conf`

Server's Console

```
nano /etc/apache2/mods-available/mpm_prefork.conf
```

Note: To search for a string in `nano`, use `Ctrl+W`

Save the output and the configuration file for the lab report.

Report

1. How many `apache2` processes were started? Which one was the master server, and which ones were the child servers? Use `htop` in the **tree** mode to see process hierarchy by executing:

   Server's Console

   ```
   htop --tree
   ```

   Justify your answer using the `apache2` configurations file. Explain your answer with values of *Startservers, MinSpareServers, MaxSpareServers, MaxRequestWorkers, ServerLimit, MaxconnectionsPerChild*
2. What is the purpose of initiating multiple `apache2` processes?

2 HTTP Request

Execute `wireshark` to capture packets between the *gui* host and the *server* host.

Login to the *server* host's web server by executing the following command in the *gui*'s terminal:

gui's Auxiliary Console

```
telnet 128.238.66.100 80
```

In the telnet console, type the following HTTP request line by line:

gui's Auxiliary Console

```
 GET /index.html HTTP/1.0
 From: netlab@h0
 User-Agent: HTTPTool/1.0
```

Note that you need to press the *Return* key to input the last line, which is blank (you should press the *Enter* key twice). When the `telnet` process is terminated, save the output for your lab report.

Terminate `wireshark` and analyze the captured HTTP packets. Print and save the HTTP request and response.

Save the HTTP response's data part into a file, named `index.html`. Use *Mozilla* to view the file.

Report

1. Submit the HTTP request and response, including the start-lines and all the headers.

3 HTTP `Keep-Alive`

By default, *Apache* server supports persistent connections. Before this exercise, check the `KeepAlive` directive in the server configuration file to make sure it is turned on as `KeepAlive On`. To do this, execute the following command and use `Ctrl+W` to search for `KeepAlive`:

Server's Console

```
nano /etc/apache2/apache2.conf
```

Execute `tcpdump` or run `wireshark` to capture packets between the *gui* host and the *server* host.

Server's Auxiliary Console

```
tcpdump host 128.238.66.100 and 128.238.66.101        # or run wireshark
```

Enter the following URL in the *gui* NoVNC's web browser to download the HTML file consisting a line of text, an embedded picture, and a hyperlink:

gui's web browser

```
http://128.238.66.100/try1.html
```

In *Server*, disable the *Apache* server persistent connections by changing the value of `KeepAlive` option to `Off` from the configuration file (as mentioned above). To save the file after modifying it, press `Ctrl+X`, confirm it by entering `Y` and press enter. Then restart `apache2` service by running following command:

Server

```
service apache2 restart
```

Use `tcpdump` or run `wireshark` and print the HTTP requests and responses for the lab report.

Use *Mozilla* in *gui*'s to reload the `try1.html` file. Use *Ctrl+F5* to ignore cache.

Use `wireshark` and print the HTTP requests and responses for the lab report.

Report

1. When you browsed the `try1.html` file for the first time, how many HTTP requests were sent? Which files were requested? How many TCP connections were used?
2. Answer the above questions for when you browsed the `try1.html` file for the second time.
3. What is the purpose of using persistent connections?

4 HTTP Submit

Execute `wireshark` to capture packets between the *gui* host and the *server* host.

Use *Mozilla* in the (*gui*) to download the following file, which is an *HTML* form, from the server.

gui's web browser

```
http://128.238.66.100/try2.html
```

Fill a text string, such as *netlab*, into the text field in the form and click the submit button in the form.

When the server response is received, terminate `wireshark`.

Examine how LAMP (for php) or CGI (for perl) works, and identify the data string sent to the server. Save the HTTP request containing the data string for lab report.

Report

1. Submit the data string sent to the server.

Part II

DHCP Exercises

For the exercises in this section, the network topology is given in Figure 1.3, where all the hosts are connected to a single network segment.

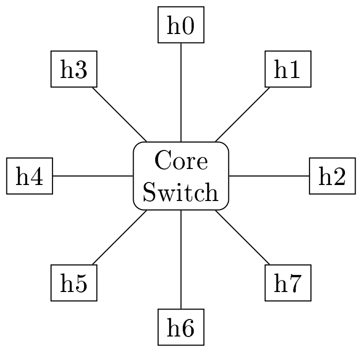

Figure 2: A single segment network (Figure 1.3)

5 DHCP Server

In this exercise, we use *h1* as the DHCP server. Configure the DHCP server by opening the configuration file[2] with a text editor such as `nano`. To do this, run the following command:

h_1's Console

```
nano /etc/dhcp/dhcpd.conf
```

Replace it with the configuration shown in Table 8.3. (*Note:* to do this, in `nano` text editor, hold `Ctrl+K` until all lines are cleared. Then, copy the given configuration to this file and save it). Then, do the following:

1. Change MAC address of *h5* in server configuration file (Table 8.3) by your *h5* MAC address and save to *dhcpd.conf*.

   Note: to obtain MAC address of *h5* run the following command:

   h_5's Console

   ```
   ifconfig
   ```

2. Set IP of *h1* to 128.238.66.100.

   h_1's Console

   ```
   ifconfig eth0 128.238.66.100/24
   ```

3. Start the DHCP server on *h1* in the foreground and working in the debugging mode:

   h_1's Console

   ```
   dhcpd -d -f
   ```

4. Execute `tcpdump -exn -nn -s 100` or `wireshark` to capture the DHCP messages in the network segment.

[2] /etc/dhcp/dhcpd.conf

h_1's Auxiliary Console

```
tcpdump -exn -nn -s 100       # or run wireshark
```

5. Then do the following to enable DHCP for the Ethernet interface on *h2*.

   h_2's Console

   ```
   ifconfig eth0 0
   dhclient eth0
   ```

   Note: Running `ifconfig eth0 0` clear IP of *eth0* on `h2`.

   When *h2* is successfully reconfigured, execute `ifconfig -a` or `ip a` to display its network interface configurations and execute `netstat -rn` or `ip r` to display its routing table.

   h_2's Console

   ```
   ifconfig -a
   ```

   or

   h_2's Console

   ```
   ip a
   ```

   Then,

   h_2's Console

   ```
   netstat -rn
   ```

   or

   h_2's Console

   ```
   ip r
   ```

   Save the outputs for the lab report.

6. Then, repeat 5 for *h3*.
7. Repeat 5 for *h4*.
8. Repeat 5 for *h5*.

Terminate `wireshark`. Print out the DHCP messages for the lab report.

Save the DHCP server output on *h1* for the lab report.

```
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 128.238.66.255;
option routers 128.238.66.1;
#option domain-name-servers 128.238.2.38, 128.238.3.21;
#option domain-name "netlab.ut.ac.ir";

subnet 128.238.66.0 netmask 255.255.255.0 {
    range 128.238.66.111 128.238.66.112;
}

host h5 {
    hardware ethernet xx:xx:xx:xx:xx:xx;    #Change this with h5's MAC address
    fixed-address 128.238.66.110;
}
```

Code 1: A DHCP server configuration file (Table 8.3)

Report

1. Compare the DHCP operation captured by `wireshark` and that shown by the DHCP server output. Explain how DHCP works.
2. Did *h2* and *h3* successfully obtain a set of new parameters? Compare the `ifconfig` and `netstat` output with the parameters carried in the corresponding DHCP messages.
3. Answer the above question for *h4*. Explain why *h4* failed.
4. Answer the above question for *h5*. Explain why *h5* succeeded.

Part III

NTP exercises

Before proceeding to the next exercise, reboot the hosts and set network topology is given in Figure 1.3 and set hosts IP address from 128.238.66.100 to 128.238.66.107.

Table 1: The IP addresses of the hosts (Table 1.2)

Host	IP Address	Subnet Mask
h0	128.238.66.100	255.255.255.0
h1	128.238.66.101	255.255.255.0
h2	128.238.66.102	255.255.255.0
h3	128.238.66.103	255.255.255.0
h4	128.238.66.104	255.255.255.0
h5	128.238.66.105	255.255.255.0
h6	128.238.66.106	255.255.255.0
h7	128.238.66.107	255.255.255.0

6 ** Local Date

Execute `date` to display the system time of your host(for example, *h0*). Display the manual page of `date`, and study its options and usages. Try the following `date` commands:

h_0's Console

```
date
# after display the system time
man date
# after display the manual page of date
date --date='2 days ago'
# after display the date
date --date='3 months 2 days'
# after display the date
date --set='+3 minutes'
# after display the date
date -r /etc/network/interfaces
```

Note: The sentences after # are descriptions and not need to enter them in *h0*'s Console.

For the last command, any other file can be used instead of `/etc/network/interfaces` [3].

Report

1. Submit the `date` outputs you saved. Explain the use of the commands.

7 Remote Date

While `tcpdump -n -nn -ex host h0.netlab and h1.netlab` or `wireshark` is running, execute `rdate -p h1.netlab` on the *h0* to display the system time of the *h1* machine. Repeat the above `rdate` command, but use the `-u` option.

h_0's Auxiliary Console

```
tcpdump -n -nn -ex host 128.238.66.100 and 128.238.66.101   # or run wireshark
```

h_0's Console

```
rdate -p 128.238.66.101
```

Now, repeat the above `rdate` command but use the `-u` option:

h_0's Console

```
rdate -p 128.238.66.101 -u
```

Save the `wireshark` or `tcpdump` outputs for the lab report.

Report

1. What port numbers were used by the remote machine? What port numbers were used by the local host?
2. How many bytes of data were returned by the remote time server, both in the UDP case and in the TCP case?
3. What TCP header options were used?

8 ** NTP Sync

In this exercise, we start the NTP server daemon on *h0* and use NTP to synchronize all the other hosts to *h0*. Initially, study the NTP configuration file[4] in *h0*.

h_0's Console

```
nano /etc/ntp.conf
```

Now, start the NTP server on *h0* by runnig elow command:

h_0's Console

```
service ntp start
```

To determine the status of the NTP server, execute the following command:

[3] This file is chosen because it is available in every host.
[4] `/etc/ntp.conf`

h_0's Console
```
service ntp status
```

Use `tcpdump -ex -n -nn host h1.netlab and h0.netlab` or `wireshark` to capture packets between the *h1* host and the *h0*.

h_1's Auxiliary Console
```
tcpdump -ex -n -nn host 128.238.66.101 and 128.238.66.100 # or run wireshark
```

Execute `ntpdate -d -v 128.238.66.100` to synchronize the *h1* host with *h0*.

h_1's Console
```
ntpdate -d -v 128.238.66.100
```

Study the output of this command and save the `ntpdate` and the `wireshark` outputs for the lab report.

Report

1. Which port does the NTP server use? Justify your answer using the `wireshark` output.

9 ** NTP Server

Keep the NTP server running on *h0*. Run `tcpdump` (or use `wireshark`) in any other host (e.g *h1*) to capture the NTP messages between *h1* and *h0* by executing the following command:

h_1's auxiliary console
```
tcpdump -exn -nn host 128.238.66.101 and 128.238.66.100   # or run wireshark
```

Edit the `/etc/ntp.conf` in *h1* using a text editor such as `nano`:

h_1's Console
```
nano /etc/ntp.conf
```

Add the following line to the end of the config file, and save it by pressing `Ctrl+X`, `Y` and enter respectively.

h_1's NTP config
```
server 128.238.66.100
```

Start the NTP client on *h1*, by executing:

h_1's Console
```
service ntp start
```

Wait for several minutes. Then terminate the `tcpdump` or `wireshark` program. Analyze the captured NTP packets. Print one of the NTP packets for the lab report.

Execute the following command in *h1* to get NTP server list:

h_1's Console
```
ntpq -p
```

Finally, execute `ntptrace` in *h1* to show the client/server relation of NTP:

h_1's Console
```
ntptrace
```

Report

1. Submit the NTP packet captured. List the fields and their values.
2. What was the rate at which NTP queries were sent by the client?
3. Which stratum was your host in? Which stratum was the NTP server in?

Part IV

NAT Exercises

For the exercises in this section, we use a network setting as shown in Figure 8.7. The lower subnet is a private network where the hosts are assigned with the Class A addresses with the 10.0.0.0/8 prefix. The upper subnet represents the Internet. The hosts, i.e. *h0* and *h1* are assigned with public IP addresses. Router 1 is used as the stub router, which performs address or port translation for the private network.

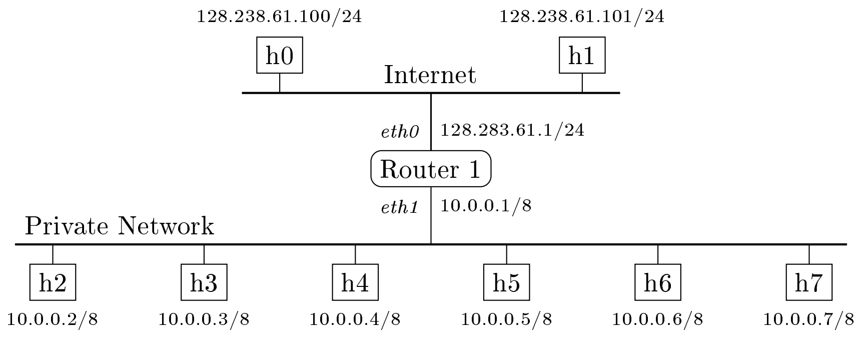

Figure 3: The network configuration for the NAT exercises (Figure 8.7)

10 NAT

Connect the hosts and Router 1 as shown in Figure 8.7. Then set the IP address and the network mask of your host as shown in the figure. Open R1's Console and run the commands shown in Table 8.5 line by line. Note that there is a static translation that maps 10.0.0.7, or *h7*, to 128.238.61.104.

R_1's Console

```
R1# conf term
R1(config)# ip nat pool mypool 128.238.61.102 128.238.61.103 netmask 255.255.255.0
R1(config)# ip nat inside source list 8 pool mypool
R1(config)# ip nat inside source static 10.0.0.7 128.238.61.104
R1(config)# int f0/0
R1(config-if)# ip address 128.238.61.1 255.255.255.0
R1(config-if)# no shut
R1(config-if)# ip nat outside
R1(config-if)# exit
R1(config)# int f0/1
R1(config-if)# ip address 10.0.0.1 255.0.0.0
R1(config-if)# no shut
R1(config-if)# ip nat inside
R1(config-if)# exit
R1(config)# access-list 8 deny host 10.0.0.7
R1(config)# access-list 8 permit 10.0.0.0 0.0.0.255
R1(config)# exit
```

Code 2: Figure 8.7 (Table 8.5)]NAT Router Configuration in Figure 8.7 (Table 8.5)

Afterward, execute the following command to display the current router configuration:

R_1's Console

```
R1# write term
```

Execute `show ip nat translations` in the router login window to display the translation table. Save the output for the lab report.

R_1's Console

```
R1# show ip nat translations
```

Save both outputs for the lab report.

Report

1. How many entries were there in the translation table? Why?

11 NAT Visibility

Keep the login session to the router running. Run `tcpdump` or `wireshark` on two side of the router (for example, *h0* and *h2*).

h_0 and h_2's Console

```
tcpdump -exn -nn              # or run wireshark
```

Before any host in the private network sends any packet out, `ping` an inside host (e.g. *h5*) from an outside host (e.g. *h1*). You may try to `ping` following IPs. Can you `ping` these IP addresses from *h1*?

h_1's Auxiliary Console

```
ping 10.0.0.5
ping 128.238.61.102
ping 128.238.61.103
ping 128.238.61.104
```

Let an inside host send packets to an outside host, e.g. from *h5*, execute:

h_5's Console

```
ping 128.238.61.100
```

Can you `ping` *h5* from an outside host now? Why? Which IP address should be used in the `ping` command in order to `ping` *h5*? (for example, run below command).

h_1's Auxiliary Console

```
ping 10.0.0.5
```

Execute the following command in the router console window to display the translation table. Save the output for the lab report:

R_1's Console

```
R1# show ip nat translations
```

Report

1. Answer the above questions. Use the saved translation table to justify your answers.
2. Compare the IP header of the ICMP query captured in the private network with that of the same ICMP query captured in the upper subnet, list their differences. Explain how NAT works.
3. In addition to the IP address, what else was changed in the ICMP query packet?

12 ** NAT Table

Keep the login session to the router running. Execute the following command or `wireshark` to capture ICMP messages on two subnet:

h_1 and h_2's Auxiliary Console

```
tcpdump -enx -s 100 ip proto 1          # or run wireshark
```

Execute the following command on *h2* to generate an ICMP port unreachable error:

h_2's Console

```
socket -i -u -n1 128.238.61.101 8888
```

Print the ICMP error message for the lab report.

Execute the following command in the router console window to display the translation table and save the output for the lab report

R_1's Console

```
R1# show ip nat translations
```

Report

1. Analyze the IP headers, the ICMP headers, and the ICMP payloads of the ICMP port unreachable errors captured in the private network and in the public network from the first experiment. Explain how ICMP error was handled by the NAT router.

13 ** NAT and PAT

Reboot the router to restore its default configuration(In GNS3, *stop* the router and then *start* it again). Then, configure the router to use PAT, as given in Table 8.6.

R$_1$'s Console

```
R1# conf term
R1(config)# ip nat inside source list 8 interface f0/0 overload
R1(config)# ip nat inside source static tcp 10.0.0.7 80 128.238.61.1 80
R1(config)# int f0/0
R1(config-if)# ip address 128.238.61.1 255.255.255.0
R1(config-if)# no shut
R1(config-if)# ip nat outside
R1(config-if)# exit
R1(config)# int f0/1
R1(config-if)# ip address 10.0.0.1 255.0.0.0
R1(config-if)# no shut
R1(config-if)# ip nat inside
R1(config-if)# exit
R1(config)# access-list 8 deny host 10.0.0.7
R1(config)# access-list 8 permit 10.0.0.0 0.0.0.255
```

Code 3: Figure 8.7 (Table 8.6)]PAT Router Configuration in Figure 8.7 (Table 8.6)

Now all the hosts in the private network use the same IP address 128.238.61.1. However, note that there is a static translation that maps *h7*'s port 80 to 128.238.61.1 port 80.

Execute `tcpdump` or `wireshark` on both subnets, same as before.

Generate traffic between the inside and outside hosts. Examine the `tcpdump` or `wireshark` output to see how PAT works.

Start the Apache web server on *h7*.

h$_7$'s Console

```
service apache2 start
```

Also, run the following command on an outside host (e.g. *h0*).

h$_0$'s Console

```
curl http://128.238.61.1
```

Execute the following command in the router's console window to display and then save the translation table:

R$_1$'s Console

```
R1# show ip nat translations
```

Report

1. From the `wireshark` data, explain how PAT worked, both for a dynamic translation and a static translation.
2. With PAT, can you have two web servers in the private network? If not, why? If yes, explain how this can be done.

Part V

** Socket Programming Exercises

14 Socket Programming

For the exercises in this section, use a simple network topology with two terminal hosts, as shown in Figure 6.

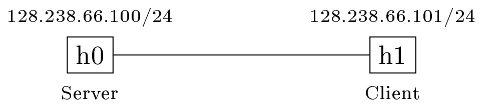

Figure 4: Two simple hosts connected directly.

Examine the UDP socket programs `/home/netlab/code/UDPserver.c` and `/home/netlab/code/UDPclient.c` to learn how to write a UDP socket program. First, navigate to the codes directory:

h_0's Console

```
cd /home/netlab/code
```

h_1's Console

```
cd /home/netlab/code
```

Now open the files with a text editor:

h_0's Console

```
nano UDPserver.c
```

h_1's Console

```
nano UDPclient.c
```

You can compile the C programs by using the following commands. Compile output available is at `/usr/local/bin/` directory:

h_0's Console

```
gcc -o UDPserver UDPserver.c -lnsl
```

h_1's Console

```
gcc -o UDPclient UDPclient.c -lnsl
```

Start `tcpdump` or `wireshark` to capture packets between these two hosts.

On the *h0*, start the UDP server by executing: (server-port can be any port unused by the system)

h_0's Console

```
./UDPserver server-port
```

Then, start the UDP client on *h1* by running:

h_1's Console

```
./UDPclient 128.238.66.100 server-port a-message
```

Where the `server-port` is same as before and `a-message` is any string. You may execute the UDP client program on other hosts to connect to the same UDP server.

Terminate `tcpdump`, examine its output and compare the output with the UDP server and client outputs.

Repeat the above experiments, but now use the `TCPserver.c` and `TCPclient.c` instead of `UDPserver.c` and `UDPclient.c`.

15 Socket Options

Study the applications of `setsockopt` function in socket programming by reading the documents or manual pages available in internet.

Examine the `netspy` and `netspyd` source code in Appendix C.2 of reference book to see how to create a multicast socket and how to set the TTL value for the packets.

16 FTP Socket Programming

This is an optional exercise on socket programming. Or, it can be assigned as a take-home project for extra credits. Note that familiarity with C (or C++) programming is required.

Problem

Examine the message exchanges of FTP. Write a FTP client program which takes a file name as input, and upload the file to a standard FTP server on a remote machine.

Hints

- First you need to set up the control connection to Port 21 of the remote machine, using a TCP socket.
- When the control connection is established, you need to exchange FTP commands with the remote FTP server, as given in Table 5.1 of reference book.
- You can first run the following commands to list all the FTP commands:

  h_1's Console

  ```
  telnet 128.238.66.100 21
  help
  ```

  Also, you can try the commands out in the `telnet` window, e.g. use the following commands to send the user ID and the password to the FTP server:

  h_1's telnet

  ```
  USER netlab
  PASS netlab
  ```

  To terminate the `telnet` session, type `QUIT`.
- In your program, these messages should be sent to the FTP server by calling the `send()` function of the local TCP socket.
- Also your program needs to parse the server responses (some examples are given in Table 5.2 of reference book) to find out the status of the previous FTP command.
- The FTP data connection should be established using the `PORT` command (see Chapter 5 of reference book).