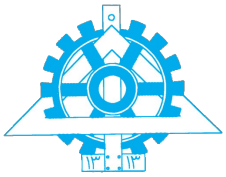
In the name of Allah

بسم الله‌الرحمن الرحیم

TCP and its Applications Laboratory Manual



University of Tehran
دانشگاه تهران

School of Electrical and Computer Engineering
دانشکده مهندسی برق و کامپیوتر

Computer Network Lab
آزمایشگاه شبکه‌های کامپیوتری

Dr. Ahmad Khonsari - احمد خونساری
a_khonsari@ut.ac.ir

Amir Haji Ali Khamseh'i - امیر حاجی علی خمسه‌ء
khamse@ut.ac.ir

Sina Kashi pazha - سینا کاشی پزها
sina_kashipazha@ut.ac.ir

Amirahmad Khordadi - امیر احمد خردادی
a.a.khordadi@ut.ac.ir

November 15, 2018

۲۴ آبان ۱۳۹۷

HTTP exercises 1

Study the Apache server configuration file (see Section 8.2.3). Examine the various configuration directives used and the corresponding settings.
Start the Apache server on your host. In order to check if the server is working properly, you may start a `Mozilla` web browser to download the test page at `http://localhost/`. Then, execute **pgrep httpd** to list the process IDs of the **httpd** processes started. Save the output and the configuration file for the lab report.

Report

1. How many httpd processes were started? Which one was the master server, and which ones were the child servers? Justify your answer using the httpd.conf file.

2. What is the purpose of initiating multiple httpd processes?

HTTP exercises 2

Execute **wireshark** to capture packets between your host and a remote host.
Login to the remote host's web server: **telnet** `10.0.0.2` **80**.
In the login console, type the following HTTP request line by line:

```
GET /usage/index.html HTTP/1.0
From: guest@your host
User-Agent: HTTPTool/1.0
```

Note that you need to type the `Return` key to input the last line, which is blank. 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.

HTTP exercises 3

By default, Apache server supports persistent connections. Before this exercise, the lab instructor should check the `KeepAlive` directive in the server configuration file to make sure it is turned on, as `KeepAlive` on.
Execute **wireshark** to capture packets between your host and a remote host.
Start `Mozilla` on your host. Go to menu `Edit/Preferences/Advanced/HTTP Networking`, and uncheck the Enable `Keep-Alive` checkbox to disable persistent connections.
Enter the URL `http://10.0.0.2/try1.html`, to download the HTML file consisting a line of text, an embedded picture, and a hyperlink. Use **wireshark** and print the HTTP requests and responses for the lab report.
Restart the **wireshark** program. Goto `Mozilla` menu `Edit/Preferences/Advanced/HTTP Networking`, and enable persistent connections by checking Enable `Keep-Alive`.
Use `Mozilla` to reload the `try1.html` file.
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?

HTTP exercises 4

Execute **wireshark** to capture packets between your host and a remote host.
Use `Mozilla` to download the `http://remote host/try2.htm` file, which is an `HTML` form, from the remote host.
Fill a text string, e.g., the name of the host being used, 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 CGI 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.

DHCP exercises 1 [1]

In this exercise, we use `guchi` as the DHCP server, with a configuration file shown in Table 8.3. Do the following.

1. Start the DHCP server on `guchi` in the foreground and working in the debugging mode: **/usr/sbin/dhcpd -d -f**.

2. Execute **wireshark** to capture the DHCP messages in the network segment.

3. Then do the following to enable DHCP for the Ethernet interface on `shakti`.
   Go to the system menu: `System Settings/Network`. In the `Network Configuration` dialog, choose the `Device` tab, and click on the `eth0` item. Next,click the Edit button to bring up the `Ethernet Device` dialog. In this dialog, check `Automatically obtain IP address settings with` and select `dhcp` from the following drop list. When the configuration is done, save the new configuration and then execute **/etc/init.d/network restart** to load the new configuration.
   When `shakti` is successfully reconfigured, execute **ifconfig -a** to display its network interface configurations and execute **netstat -rn** to display its routing table. Save the outputs for the lab report.

4. Then, repeat 3 for `vayu`.

5. Repeat 3 for agni.

6. Repeat 3 for apah.

Terminate **wireshark**.Print out the DHCP messages for the lab report.
Save the DHCP server output on `guchi` for the lab report.

Report

1. Compare the DHCP operation captured by **wireshark** and that shown by the DHCP server output. Explain how DHCP works.

2. Did `shakti` and `vayu` 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 agni. Explain why agni failed.

4. Answer the above question for apah. Explain why apah succeeded.

[1] For the exercises in this section, we use the same network setting as the one used in the previous exercises.

NTP exercises 1 [2]

Execute **date** to display the system time of your host. Display the manual page of **date**, and study its options and usages. Try the following **date** commands:
date --date='2 days ago'
date --date='3 months 2 days'
date --set='+3 minutes'
date -r file_name
You can choose any file in the current directory for the *file_ name* parameter.

Report

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

NTP exercises 2

While **wireshark** is running, execute **rdate -p** *remote_ host* to display the system time of the remote machine. Repeat the above **rdate** command, but use the **-u** option. Save the **wireshark** 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?

NTP exercises 3

In this exercise, we start the NTP server daemon on `shakti` and use NTP to synchronize all the other hosts to `shakti`. Study the NTP configuration file `/etc/ntp.conf` in `shakti` and in your host. If you are using another machine, you can telnet to shakti and display the `/etc/ntp.conf` file in the telnet window. Start the NTP server on `shakti` by: **/etc/init.d/ntpd start**. To determine the status of the NTP server, use **/etc/init.d/ntpd status**. Use **wireshark** to capture packets between your host and `shakti`. Execute **ntpdate -d -v 128.238.66.100** to synchronize your host to `shakti`. Study the output of this command. 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.

NAT exercises 1 [3]

Connect the hosts and Router 1 as shown in Fig. 8.7. Then set the IP address and the network mask of your host as shown in the figure. In addition, you need to add a default route in your host's routing table, using the router interface on your subnet as the default router. One student should **telnet** to the router and configure the router as shown in Table 8.5. Note that there is a static translation that maps 10.0.0.7, or `guchi`, to 128.238.61.104. Login to the router, execute **write term** to display the current router configuration. Execute **show ip nat translations** in the *Privileged EXEC* mode to display the translation table. Save both outputs for the lab report.

[2] Before proceeding to the next exercise, reboot the hosts to restore their original configurations.
[3] For the exercises in this section, we use a network setting as shown in Fig. 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., shakti and vayu are assigned with public IP addresses. Router 1 is used as the stub router, which performs address or port translation for the private network.

NAT Router Configuration in Fig. 8.7

```
ip nat pool mypool 128.238.61.102 128.238.61.103 netmask 255.255.255.0
ip nat inside source list 8 pool mypool

ip nat inside source static 10.0.0.7 128.238.61.104

interface ethernet 0
ip address 128.238.61.1 255.255.255.0
ip nat outside

interface ethernet 1
ip address 10.0.0.1 255.0.0.0
ip nat inside

access-list 8 deny host 10.0.0.7
access-list 8 permit 10.0.0.0 0.0.0.255
```

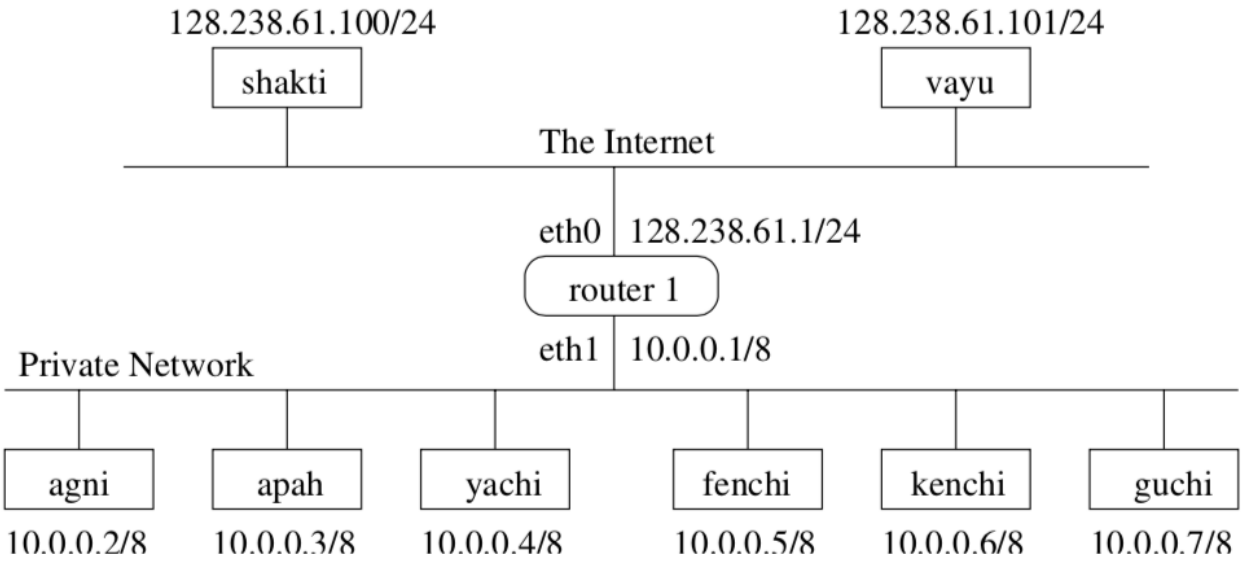


Figure 8.7. The network configuration for the NAT exercises.

Report

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

NAT exercises 2

Keep the login session to the router running. Execute **wireshark** on all the hosts.
Before any host in the private network send any packets out, ping an inside host (e.g., `fenchi`) from an outside host (e.g., **vayu**). You may try to ping 10.0.0.5, 128.238.61.102, 128.238.61.103, or 128.238.61.104. Can you ping these IP addresses?
Let an inside host send packets to an outside host, e.g., from `fenchi`, execute ping 128.238.61.100. Can you ping `fenchi` from an outside host now? Why? Which IP address should be used in the ping command in order to ping `fenchi`?

Execute show ip nat translations in the router login window to display the translation table. Save the output for the lab report.
Exchange the data you saved with a student in the other subnet.

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?

NAT exercises 3

Keep the login session to the router running. Execute **wireshark** to capture ICMP messages.
Execute **sock -i -u -n1 128.238.61.101 8888** on `agni` to generate an ICMP port unreachable error.
Print the ICMP error message for the lab report.
Execute **show ip nat translations** in the router login window to display the translation table. Save the output for the lab report. Exchange the data you saved with a student in the other subnet.

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.

NAT exercises 4

Reboot the router to restore its default configuration. Then, configure the router to use PAT, as given in 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 guchi's port 80 to 128.238.61.1 port 80.
Execute **wireshark** on all the hosts.
Generate traffic between the inside and outside hosts. Examine the **wireshark** output to see how PAT works. Start the Apache web server on `guchi`. Also, start the web browser `Mozilla` on an outside host (e.g., `shakti`), and enter the URL `http://128.238.61.1`. Save the **wireshark** output. Use **show ip nat translations** to display and then save the translation table. Exchange the data you saved with a student in the other subnet.

PAT Router Configuration in Fig. 8.7

```
ip nat inside source list 8 interface ethernet 0 overload

ip nat inside source static tcp 10.0.0.7 80 128.238.61.1 80

interface ethernet 0
ip address 128.238.61.1 255.255.255.0
ip nat outside

interface ethernet 1
ip address 10.0.0.1 255.0.0.0
ip nat inside

access-list 8 deny host 10.0.0.7
access-list 8 permit 10.0.0.0 0.0.0.255
```

Report

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

Socket programming exercises 1

Examine the UDP socket programs `/home/guest/UDPserver.c` and `/home/guest/UDPclient.c` to learn how to write a UDP socket program. Compile the C programs using `gcc -o UDPserver UDPserver.c -lnsl` and `gcc -o UDPclient UDPclient.c -lnsl`.
Start **wireshark** to capture packets from or to a remote host.
On the remote host, start the UDP server by **UDPserver server_port**. Then, start the UDP client on your host by **UDPclient remote_host server_port a_message**. You may execute the UDP client program on other hosts to connect to the same UDP server. Terminate **wireshark**, 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`.

Socket programming exercises 2

Execute **man setsockopt** to display the various socket options and how to set them. Examine the **netspy** and **netspyd** source code in Appendix C.2 to see how to create a multicast socket and how to set the TTL value for the packets.

Socket programming exercises 3

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 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.
- You can first run **telnet remote_host 21**, then type **help** to list all the FTP commands. Also, you can try the commands out in the **telnet** window, e.g. use **USER guest** to send the user ID and **PASS guest1** to send the password to the FTP server. 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) to find out the status of the previous FTP command.
- The FTP data connection should be established using the **PORT** command (see Chapter 5).