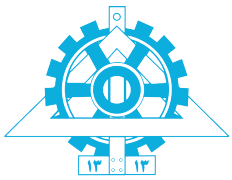
UDP and Its Applications[1]

HARDNESS : 7/10



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

October 24, 2021

۲ آبان ۱۴۰۰

[1]S. Panwar, S. Mao, J.-dong Ryoo, and Y. Li, "UDP and its applications," in TCP/IP Essentials: A Lab-Based Approach, Cambridge: Cambridge University Press, 2004, pp. 100–110.

Objectives

- Study **socket** as a traffic generator, in terms of its features and command line options.
- Study the User Datagram Protocol.
- IP fragmentation.
- MTU and path MTU discovery.
- UDP applications, using the Trivial File Transfer Protocol as an example.
- Compare UDP with TCP, using TFTP and the File Transfer Protocol (FTP).

Part I

Using the `socket` Program

In this lab , you will not need to a router, only two hosts and one hub to connect two host together as show in Figure 5.0 and Table 5.0

Table 1: Host IP addresses for Figure 5.0

$Host_A$		$Host_B$	
Name	IP Address	Name	IP Address
h0	128.238.61.100/24	h1	128.238.61.101/24

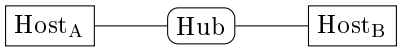


Figure 1: Simple router experiment (Figure 5.0)

1 Socket Operation

Use the following commands in *h0* and *h1* as client and server host to observe the basic operation of `socket` [1] and `echo` service.

h_0's Console

```
socket -u 128.238.61.101 echo
```

h_1's Console

```
socket -s 5555
```

h_0's Console

```
socket -i -n3 -w2048 128.238.61.101 5555
```

Note: in general, this command is like `socket` -i -n3 -w2048 *server-host* 5555.

Report

1. Explain the operation of each command.

[1] Basic command is `sock` use alternative `socket` (rename of `sock`)

2 socket -h

Study various options associated with the `socket` program. A brief list of options can be displayed by typing `socket`. More detailed discussion on socket can be found in Appendix C of [5] in the reference book.

h_0's Console

```
socket
```

3 Segment Size

While running `tcpdump -nv` on *your-host* or *remote-host* (or run Wireshark), execute the following command with different values of size (i.e. the size of the datagram).

h_0's Console

```
tcpdump -nv
```

h_0's Auxiliary console

```
socket -u -i -n1 -w100 128.238.61.101 echo
```

Note: in general, this command is like `socket` -u -i -n1 -w *size server-host* echo.

The `-u` option is used to send UDP datagrams rather than TCP segments.

Increase *size* (i.e. the size of the datagram) until fragmentation occurs.

Use `netstat -in` to find out the MTU of the Ethernet interface.

h_0's Auxiliary console

```
netstat -in
```

Report

1. What is the maximum value of *size* for which the UDP datagram can be sent without IP fragmentation? Justify your answer with the `netstat` output.

4 Datagram Fragmentation

Capture the data packets generated by the following command, while run `tcpdump src host` ***your-host*** on *h1*.

h_1's Console

```
tcpdump src host 128.238.61.100
```

h_0's Console

```
socket -u -i -n1 -w10000 128.238.61.101 echo
```

Save the `tcpdump` output for the lab report.

Report

1. Explain the `tcpdump` output in terms of the IP header fields that are used in fragmentation.
2. When IP fragmentation occurs, only the first fragment has the UDP header. How do you verify this fact from the `tcpdump` output?

5 Maximum Datagram Size

While running `tcpdump`, execute the following command with different values of size (for example 10000),

h_0's Auxiliary Console

```
tcpdump
```

h_0's Console

```
socket -u -i -n1 -w10000 128.238.61.101 echo
```

Note: in general, this command is like `socket` -u -i -n1 -w *size server-host* echo.

in order to find out the maximum size of a UDP datagram that the system can send or receive, even when fragmentation is allowed.

Report

1. What is the maximum size of user data in a UDP datagram that the system can send or receive, even when fragmentation is allowed?

Part II

Path MTU Discovery Exercise

6 Discover path MTU

Connect the routers and the hosts as shown in Figure 5.5 Change the IP addresses of all hosts accordingly.(*Note:* There is no need to change the IP address in $Fig-4\_10$ that you downloaded from github). Note that the router IP addresses are the same as their default. Also you need to add route for all hosts to other subnets.

h_0's Console

```
ip route add 128.238.62.0/24 dev eth0
```

h_1's Console

```
ip route add 128.238.61.0/24 dev eth0
```

Change the `MTU` of the *ethernet1* interfaces of R_1 to 500 bytes.

R1#

```
config term
R1(config)# int f0/1
R1(config-if)# ip unreachables ! enables the router to send ICMP unreachable errors
R1(config-if)# ip mtu 500
! this command set ip layer mtu size
! the real packet has 20 byte ip header (or more depend on options value)
R1(config-if)# mtu 500 ! may be not supported (can see: mtu ?)
! this command set ethernet layer mtu size
```

Note: the sentences written after **!** are comments and do not need to be runned.

Test connectivity by `ping`ing hosts in the other subnets. After you can reach the hosts in the other subnets, run `tcpdump -nx` on your host.

h_0's Console

```
ping 128.238.62.101
! After you can reach the hosts in the other subnets, press Ctrl+Z to stop ping command.
```

h_0's Auxiliary Console

```
tcpdump -nx
```

Start a UDP socket server on *remote-host*, using `socket -u -s 5555` on *h1*.

h_1's Console

```
socket -u -s 5555
```

Then run the socket client from *h0*:

h_0's Console

```
socket -i -u -n10 -w1200 -p5 128.238.62.101 5555
```

Table 2: Router and Host IP addresses for Figure 5.5 (Table 5.5)

Router		$Host_A$		$Host_B$	
eth0	eth1	Name	IP Address	Name	IP Address
128.238.61.1/24	128.238.62.1/24	h0	128.238.61.101/24	h1	128.238.62.101/24

Figure 2: The network setup for Path MTU Discovery Exercise (Figure 5.5/Figure 4.10)

Observe the DF bit of the first datagram and that of the following datagrams. Save the `tcpdump` output for your lab report.

Report

1. Explain the operation of path MTU discovery based on the `tcpdump` outputs saved.
2. Which ICMP message is used in path MTU discovery? Give the decimal value of each field of the captured ICMP message.
3. What is the MTU of the destination network of the UDP datagram? Verify your answer using both the ICMP message and the IP fragmentation trace saved.

Part III

Exercises with FTP and TFTP

Use first network (section Using the `socket` Program) topology (Figure 5.0 or Figure 1.3) for this exercise. We will study the performance of FTP and TFTP for file transfer between two machines. By transferring the same file using these two protocols, we can compare the operations and performances of UDP and TCP.

Four files (`large.dum`, `med.dum`, `small.dum` and `thin.dum`) with random contents are stored in the `/home/netlab`[2] directory of each host in the lab. We will use the `get` command to retrieve files from a remote host.

[2] We change original path (`/tftpboot`) to `/home/netlab` to be same as ftp user path.

When FTP is used, you need to change directory to `/home/netlab/` by `cd /home/netlab` before retrieving the file. If you don't know how to use `tftp`, refer to its manual page.

7 TFTP and FTP

In order to compare the transfer rates of FTP and TFTP, we will retrieve all large, med, small and thin files from a remote server using FTP and TFTP, respectively.

First run the following `tcpdump` command:

h_0's Console

```
tcpdump host 128.238.61.100 and 128.238.61.101
```

Note: You can use the redirect operator, `>`, to save the `tcpdump` output into a text file for large `tcpdump` outputs and read it with `less`, `grep`, `nano` or `vim`. For example you can change previous command into:

h_0's Console

```
tcpdump host 128.238.61.100 and 128.238.61.101 > output.dump
```

Now you should repeat this scenario for all files (`large.dum`, `med.dum`, `small.dum` and `thin.dum`).

h_0's Auxiliary Console

```
$ ls /etc/xinetd.d/    ! see services in xinetd
$ tftp host
tftp> get $filename    ! thin, small, med, large
tftp> quit
$ ftp host             ! Enter user and password -> netlab and netlab
ftp> ls
ftp> get $filename     ! thin, small, med, large
ftp> quit
```

Note: For example, we run below commands for file *small.mud*. You should repeat this commands for other files.

h_0's Auxiliary Console

```
ls /etc/xinetd.d/
tftp 128.238.61.101
tftp> get small.dum
tftp> quit
ftp 128.238.61.101 # Enter user and password -> netlab and netlab
ftp> ls
ftp> get small.dum
ftp> quit
```

Also, from the `ftp` window, record the transfer rate (time) displayed.

Report

1. Examining the saved `tcpdump` output. Identify the starting and ending time of actual data transfer. Don't include the time spent establishing the TCP connection. Calculate the time spent for data transfer.
2. Compare the time with the value displayed in `ftp` window. Are they consistent? If there exists any significant difference, what might be the reason?
3. Now, from the second session, carefully determine the starting and ending time of data transfer for the `tftp` program.
4. Compare the time with the value displayed in `tftp` window. Are they consistent? If there exists any significant difference, what might be the reason?

5. By comparing the actual data transfer times of `ftp` and `tftp`, which of these two is faster, and why?

8 TFTP Analysis

Capture the packets that are exchanged during a `tftp` session for the `/home/netlab/small.dum` file between *h0* and *h1*, using:

h_0's Console

```
tcpdump -x host 128.238.61.100 and 128.238.61.101
```

h_0's Auxiliary Console

```
tftp 128.238.61.101
tftp> get small.dum
tftp> quit
```

Observe the protocol in action. Analyze various types of TFTP messages. Save `tcpdump` output for the lab report.

Report

1. List all the different types of packets exchanged during the `tftp` session. Compare them with the TFTP message format in Figure 5.3 of reference book.

   Why does the server's port number change?

2. In most cases, `tftp` service is restricted.[3] Why is `tftp` service not generally available to users? At least write two problem of `tftp` protocol.

3. In section 5, we found the maximum size of a UDP datagram in your machine. With `tftp`, which uses UDP, we transferred a file larger than the maximum UDP datagram size. How do you explain this?

9 FTP Small File

Repeat the above experiment, but use `ftp`. Capture a trace of the packets exchanged when downloading the `/home/netlab/small.dum` file using `ftp`.

Examine the port numbers used.

h_0's Console

```
tcpdump -x host 128.238.61.100 and 128.238.61.101
```

h_0's Auxiliary Console

```
ftp 128.238.61.101    # Enter user and password -> netlab and netlab
ftp> get small.dum
ftp> quit
```

Report

1. How many well-known port numbers were used? Which machine used the well-known port numbers? What were the other machine's port numbers?

2. As can be seen from the `tcpdump` output, FTP involves two different connections, `ftp-control` and `ftp-data`. Why are two different connections used, instead of one connection?

[3] This is not the case in our lab, where we deliberately enabled the `tftp` service and use it as a tool to study the UDP protocol.

10 FTP Debug

Run `ftp` in *your-host* using the debug mode: `ftp -d remote-host`.

h_0's Console

```
ftp -d 128.238.61.101      # Enter user and password -> netlab and netlab
```

After logging into the remote host, type `dir /home/netlab/small.dum` in the `ftp` window.

h_0's Console

```
ftp> dir /home/netlab/small.dum
```

Then type `quit` to terminate the `ftp` session, and save the `ftp` window output.

h_0's Console

```
ftp> quit
```

Report

1. Submit what you saved in this exercise, explaining each line of the output. Explain how the `PORT` command works. Which connection, the control connection or the data connection, did the server send the response (the `LIST` output) on?