CS 315: Computer Networks Lab
Spring 2022-23, IIT Dharwad
Assignment-6
Wireshark: UDP & Socket Programming
February 07, 2023

Lab Instructions

- Please leave your bags on the Iron shelf near the SP16 entrance.
- Login to the Ubuntu OS on your machine. The login credentials are as follows:
    - Username: user
    - Password: 123456
- Mark your attendance in the attendance sheet before leaving the lab.
- Handle the lab resources with utmost care.
- Please go through the following exercises in today's lab.
- It is recommended that you complete all the following exercises during the lab slot itself.
- If you face any difficulties, please feel free to seek help online or from your peers or TAs.
- After finishing all exercises, please carry your solutions with you (via email/pen drive) for future reference, and delete the files from the desktop.

Introduction

In this lab, we'll take a quick look at the UDP transport protocol. UDP is a streamlined, no-frills protocol. Because UDP is simple and sweet, we'll be able to cover it pretty quickly in this lab. At this stage, you should be a Wireshark expert. Thus, we are not going to spell out the steps as explicitly as in earlier labs. In particular, we are not going to provide example screenshots for all the steps.

Part-1: Wireshark UDP

Start capturing packets in Wireshark and then do something that will cause your host to send and receive several UDP packets. It's also likely that just by doing nothing (except capturing packets via Wireshark) that some UDP packets sent by others will appear in your trace. In particular, the Domain Name System (DNS) protocol typically sends DNS query and response messages inside of UDP, so it's likely that you'll find some DNS messages (and therefore UDP packets) in your trace.

Specifically you can try out the `nslookup` command, which invokes the underlying DNS protocol, which in turn will send UDP segments from/to the host issuing the nslookup. `nslookup` is available in most Microsoft, Apple IOS, and Linux operating systems. To run `nslookup` you just type the `nslookup` command on the command line in a DOS window, Mac IOS terminal window, or Linux shell. Below Figure is a screenshot of running `nslookup` on the

Linux command line on the newworld.cs.umass.edu host located in the CS Department at the University of Massachusetts (UMass) campus, to display the IP address of www.nyu.edu.

```
sysad@sysad-OptiPlex-7080:~$ nslookup www.nyu.edu
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
www.nyu.edu     canonical name = d1q5ku5vnwkd2k.cloudfront.net.
Name:   d1q5ku5vnwkd2k.cloudfront.net
Address: 13.227.138.56
Name:   d1q5ku5vnwkd2k.cloudfront.net
Address: 13.227.138.109
Name:   d1q5ku5vnwkd2k.cloudfront.net
Address: 13.227.138.2
Name:   d1q5ku5vnwkd2k.cloudfront.net
Address: 13.227.138.7
Name:   d1q5ku5vnwkd2k.cloudfront.net
Address: 2600:9000:2154:6a00:1:f7e2:cb00:93a1
Name:   d1q5ku5vnwkd2k.cloudfront.net
Address: 2600:9000:2154:d000:1:f7e2:cb00:93a1
Name:   d1q5ku5vnwkd2k.cloudfront.net
Address: 2600:9000:2154:c000:1:f7e2:cb00:93a1
Name:   d1q5ku5vnwkd2k.cloudfront.net
Address: 2600:9000:2154:c200:1:f7e2:cb00:93a1
```

We don't need to go into any more details about `nslookup` or DNS, as we're just interested in getting a few UDP segments into Wireshark, and we promised this lab would be short!

After starting packet capture on Wireshark, run `nslookup` for a hostname that you haven't visited for a while. Then stop packet capture, set your Wireshark packet filter so that Wireshark only displays the UDP segments sent and received at your host. Pick the first UDP segment and expand the UDP fields in the details window.

Answer the following questions:

1. Select the first UDP segment in your trace. What is the packet number of this segment in the trace file? What type of application-layer protocol message is being carried in this UDP segment? Look at the details of this packet in Wireshark. How many fields are there in the UDP header? What are the names of these fields?
2. By consulting the displayed information in Wireshark's packet content field for this packet, what is the length (in bytes) of each of the UDP header fields?
3. The value in the Length field is the length of what? Verify your claim with your captured UDP packet.

4. What is the maximum number of bytes that can be included in a UDP payload? (Hint: the answer to this question can be determined by your answer to 2. above)
5. What is the largest possible source port number? (Hint: see the hint in 4.)
6. What is the protocol number for UDP? Give your answer in decimal notation. To answer this question, you'll need to look into the Protocol field of the IP datagram containing this UDP segment.
7. Examine the pair of UDP packets in which your host sends the first UDP packet and the second UDP packet is a reply to this first UDP packet. (Hint: for a second packet to be sent in response to a first packet, the sender of the first packet should be the destination of the second packet). What is the packet number of the first of these two UDP segments in the trace file? What is the packet number of the second of these two UDP segments in the trace file? Describe the relationship between the port numbers in the two packets.

Part-2: Socket Programming

In this assignment, you'll write a client that will use sockets to communicate with a server that you will also write.

Here's what your client and server should do: Your client should first generate a random between 1 and 100, open a TCP socket to your server and send a message containing (i) a string containing your (client) name and (ii) the generated integer value and then wait for a server reply.

Your server will create a string containing its(server) name and then begin accepting connections from clients. On receipt of a client message, your server should

i. print (display) the client's name (extracted from the received message) and the server's name
ii. itself generate a random integer between 1 and 100 (it's fine for the server to use the same number all the time) and display the client's number, its number, and the sum of those numbers
iii. send its name string and the server-generated integer value back to the client
iv. if your server receives an integer value that is out of range, it should terminate after releasing any created sockets. You can use this to shut down your server.

Programming Languages and Operating Systems: You may write your programs in C, C++ or Python on any platform (Liux/unix, Mac, Windows)

Part-3: Socket Programming: WebServer

In this lab, you will learn the basics of socket programming for TCP connections in Python: how to create a socket, bind it to a specific address and port, as well as send and receive a HTTP packet. You will also learn some basics of HTTP header format.

You will develop a web server that handles one HTTP request at a time. Your web server should accept and parse the HTTP request, get the requested file from the server's file system, create an HTTP response message consisting of the requested file preceded by header lines, and then send the response directly to the client. If the requested file is not present in the server, the server should send an HTTP "404 Not Found" message back to the client.

Running the Server: Put an HTML file (e.g., HelloWorld.html) in the same directory that the server is in. Run the server program. Determine the IP address of the host that is running the server (e.g., *10.250.65.188*). Now, open a browser and provide the corresponding URL. For example: *http://10.250.65.188:6789/HelloWorld.html*

'HelloWorld.html' is the name of the file you placed in the server directory. Note also the use of the port number after the colon. You need to replace this port number with whatever port you have used in the server code. In the above example, we have used the port number 6789. The browser should then display the contents of HelloWorld.html. If you omit ":6789", the browser will assume port 80 and you will get the web page from the server only if your server is listening at port 80.

Then try to get a file that is not present at the server. You should get a "404 Not Found" message.

Submission Details

- **Part-1:** Write your answers wireshark UDP in a single doc/tex file, and submit its PDF named after your IIT Dharwad roll number, which contains all answers (with screenshots, if necessary).
- **Part-2:** Submit the both server and client files with your roll number prefixed as <Roll_number>_server.py <Roll_number>_server.py
- **Part-3:** Submit a directory containing your HTML file and <Roll_number>_webserver.py