CS 176A: Homework #2

Important notes:

- Parts 1 (written questions) and 2 (wireshark) of the assignment are to be completed independently.
- Part 3 (programming) may be completed with one partner. If you need help finding a partner, use the Canvas homework 2 discussion.

Turn-in instructions:

- submit a PDF of your answers to "Homework 2 Written Assignment" on Gradescope
- submit your code to "Homework 2 Programming Assignment" on Gradescope

Part 1: Answer the following questions (30 points)

1. Note: to answer these questions, it may be helpful for you to first read through the Wireshark lab.
(a) What is a **whois** database?
(b) Use a whois database on the Internet to lookup ucsb.edu. Who is in charge of the UCSB network? What are the names of the UCSB name servers? When did UCSB obtain its original DNS entry? Which whois database did you use to find this information?
(c) Use nslookup to find a Web server that has multiple IP addresses (do not use an example from class). List the name of that web server and the addresses it maps to.
(d) Does the UCSB web server have multiple IP addresses?
(e) On a machine in CSIL, type the command "ip addr show". What IP address does your machine have? (Hint: it is the address that starts with 128.111). Include the name of the machine you are on so we can check your answer (i.e. csil-01.cs.ucsb.edu)

2. Install and compile the Python programs TCPClient and UDPClient on one host and TCPServer and UDPServer on another host (the code is available in section 2.7 of the textbook).
(a) Suppose you run TCPClient before running TCPServer. What happens? Why?
(b) Suppose you run UDPClient before you run UDPServer. What happens? Why?
(c) What happens if you use different port numbers for the client and server sides?

3. Suppose that in UDPClient.py, after we create the socket, we add the line:
clientSocket.bind((‘’, 5432))

Will it become necessary to change UDPServer.py? What are the port numbers for the sockets in UDPClient and UDPServer? What were they before making this change?

Part 2: Wireshark Lab (30 points total)
Complete the Wireshark Lab in the attached handout. The turn-in requirements are listed on the handout. Note there is a set of questions to answer in each part.

Part 3: Programming Assignment (40 points total)
Complete the programming assignment detailed on the attached handout to build TCP and UDP clients and servers.

Wireshark Lab: DNS

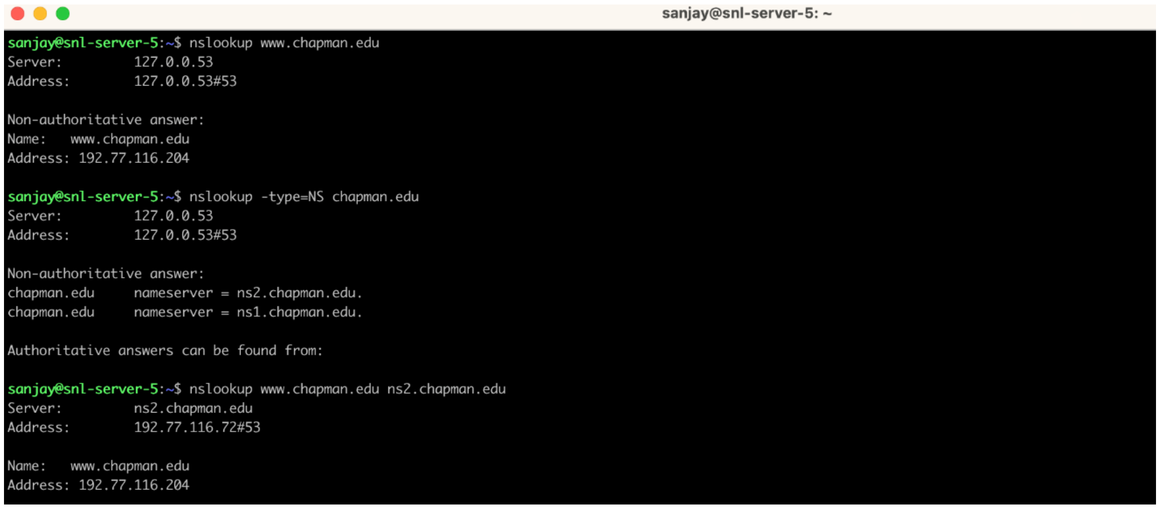
As described in Section 2.4 of the textbook, the Domain Name System (DNS) translates hostnames to IP addresses, fulfilling a critical role in the Internet infrastructure. In this lab, we'll take a closer look at the client side of DNS. Recall that the client's role in the DNS is relatively simple – a client sends a *query* to its local DNS server, and receives a *response* back. As shown in Figures 2.19 and 2.20 in the textbook, much can go on within the network, invisible to the DNS clients, as the hierarchical DNS servers communicate with each other to either recursively or iteratively resolve the client's DNS query. From the DNS client's standpoint, however, the protocol is quite simple – a query is formulated to the local DNS server and a response is received from that server.

Before beginning this lab, you'll probably want to review DNS by reading Section 2.4 of the text. In particular, you may want to review the material on **local DNS servers**, **DNS caching**, **DNS records and messages**, and the **TYPE field** in the DNS record.

1. nslookup

In this lab, we'll make use of the nslookup tool, which is available in most Linux/Unix and Microsoft platforms today. To run nslookup in Linux/Unix, you just type **nslookup** on the command line. To run it in Windows, open the Command Prompt and run nslookup on the command line. Read the man page for nslookup to learn about all of its functionalities.

In its most basic operation, the nslookup tool allows the host running the tool to query any specified DNS server for a DNS record. The queried DNS server can be a root DNS server, a top-level-domain DNS server, an authoritative DNS server, or an intermediate DNS server (see the textbook for definitions of these terms). To accomplish this task, nslookup sends a DNS query to the specified DNS server, receives a DNS reply from that same DNS server, and displays the result.



```
sanjay@snl-server-5: ~
sanjay@snl-server-5:~$ nslookup www.chapman.edu
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   www.chapman.edu
Address: 192.77.116.204

sanjay@snl-server-5:~$ nslookup -type=NS chapman.edu
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
chapman.edu     nameserver = ns2.chapman.edu.
chapman.edu     nameserver = ns1.chapman.edu.

Authoritative answers can be found from:

sanjay@snl-server-5:~$ nslookup www.chapman.edu ns2.chapman.edu
Server:         ns2.chapman.edu
Address:        192.77.116.72#53

Name:   www.chapman.edu
Address: 192.77.116.204
```

The above screenshot shows the results of three independent nslookup commands (displayed in the Windows Command Prompt). In this example, the client host is located on the campus of UCSB, where the default behavior is to use the local DNS resolver running on port 53 of localhost (127.0.0.1). When running nslookup, if no DNS server is specified, then nslookup sends the query to the local DNS resolver. Consider the first command:

```
nslookup www.chapman.edu
```

In words, this command is saying "please send me the IP address for the host www.chapman.edu". As shown in the screenshot, the response from this command provides two pieces of information: (1) the name and IP address of the DNS server that provides the answer; and (2) the answer itself, which is the host name and IP address of www.chapman.edu. Although the response came from the local DNS resolver it is quite possible that this local DNS resolver iteratively contacted several other DNS servers to get the answer, as described in Section 2.5 of the textbook.

Now consider the second command:

```
nslookup –type=NS chapman.edu
```

In this example, we have provided the option "-type=NS" and the domain "chapman.edu". This causes nslookup to send a query for a type-NS record to the local DNS resolver. In words, the query is saying, "please send me the host names of the authoritative DNS for chapman.edu". (When the –type option is not used, nslookup uses the default, which is to query for type A records.) The answer, displayed in the above screenshot, first indicates the DNS server that is providing the answer (which is the local DNS resolver along with two Chapman nameservers. Each of these servers is indeed an authoritative DNS server for the hosts on the Chapman campus. However, nslookup also indicates that the answer is "non-authoritative," meaning that this answer came from the cache of some server rather than from an authoritative Chapman DNS server.

Now finally consider the third command:

```
nslookup www.chapman.edu ns2.chapman.edu
```

In this example, we indicate that we want the query sent to the DNS server ns2.chapman.edu rather than to the local DNS resolver. Thus, the query and reply transaction takes place directly between our querying host and ns2.chapman.edu. In this example, the DNS server ns2.chapman.edu provides the IP address of the host www.chapman.edu.

Now that we have gone through a few illustrative examples, you are perhaps wondering about the general syntax of nslookup commands. The syntax is:

```
nslookup –option1 –option2 host-to-find dns-server
```

In general, nslookup can be run with zero, one, two or more options. And as we have seen in the above examples, the dns-server is optional as well; if it is not supplied, the query is sent to the default local DNS server.

Now that we have provided an overview of nslookup, it is time for you to test drive it yourself. Do the following:

1) Run nslookup to obtain the IP address of a Web server in Asia.
2) Run nslookup to determine the authoritative DNS servers for a university in Europe.
3) Run nslookup so that one of the DNS servers obtained in Question 2 is queried for the mail servers for Yahoo! mail.

Note: you do not need to turn in anything to "prove" you did this. The above three questions are simply for your own practice.

2. Tracing DNS with Wireshark

Part 2.1: Now that we are familiar with nslookup, we're ready to examine the operation of DNS. Let's first examine the DNS packets that are generated by ordinary Web-surfing activity.

- Open Wireshark and load the file dns-wireshark-trace-1. Enter "ip.addr ==128.111.5.234" into the filter. This is the IP address on which the trace was captured. This filter removes all packets that neither originate nor are destined to the capturing host. The packet capture was generated by opening a web browser, and then visiting the web page: http://www.ietf.org. Once the page was loaded, packet capture was stopped.

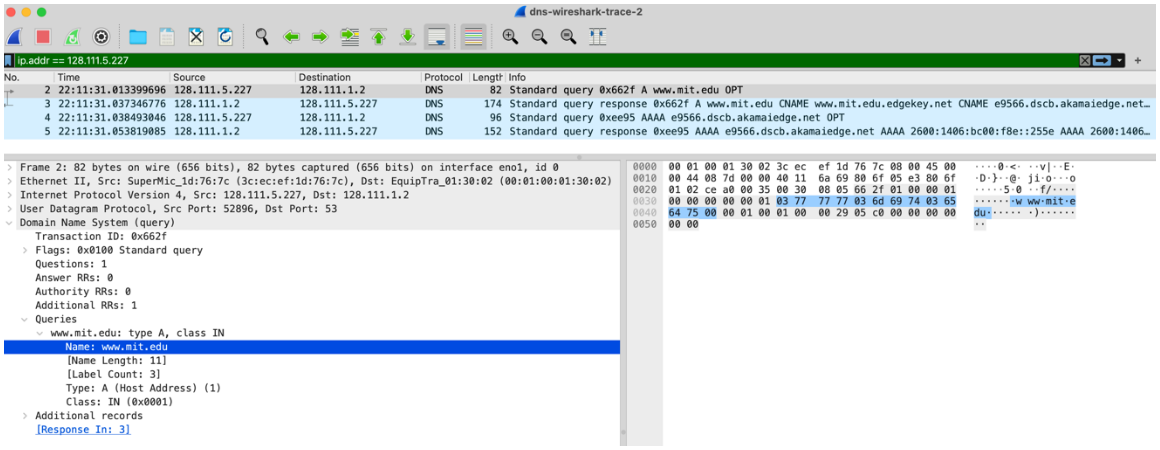
Answer the following questions:

1. Locate the DNS query and response messages. Are they sent over UDP or TCP?
2. What is the destination port for the DNS query message? What is the source port of DNS response message?
3. To what IP address is the DNS query message sent? This is the IP address of a local DNS server.
4. Examine the DNS query message. What "Type" of DNS query is it? Does the query message contain any "answers"?
5. This web page contains images. Before retrieving each image, does the host issue new DNS queries?

Part 2.2: Now let's examine with nslookup.

- Load dns-wireshark-trace-2. This trace was created by performing an nslookup on www.mit.edu from a host with an IP address of 128.111.5.227.

The trace should look like the following:



```
ip.addr == 128.111.5.227

No.  Time               Source         Destination    Protocol Length Info
  2  22:11:31.013399696 128.111.5.227  128.111.1.2    DNS       82 Standard query 0x662f A www.mit.edu OPT
  3  22:11:31.037346776 128.111.1.2    128.111.5.227  DNS      174 Standard query response 0x662f A www.mit.edu CNAME www.mit.edu.edgekey.net CNAME e9566.dscb.akamaiedge.net…
  4  22:11:31.038493046 128.111.5.227  128.111.1.2    DNS       96 Standard query 0xee95 AAAA e9566.dscb.akamaiedge.net OPT
  5  22:11:31.053819085 128.111.1.2    128.111.5.227  DNS      152 Standard query response 0xee95 AAAA e9566.dscb.akamaiedge.net AAAA 2600:1406:bc00:f8e::255e AAAA 2600:1406…
```

```
> Frame 2: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface eno1, id 0
> Ethernet II, Src: SuperMic_1d:76:7c (3c:ec:ef:1d:76:7c), Dst: EquipTra_01:30:02 (00:01:00:01:30:02)
> Internet Protocol Version 4, Src: 128.111.5.227, Dst: 128.111.1.2
> User Datagram Protocol, Src Port: 52896, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x662f
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 1
  v Queries
    v www.mit.edu: type A, class IN
        Name: www.mit.edu
        [Name Length: 11]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  > Additional records
    [Response In: 3]
```

```
0000  00 01 00 01 30 02 3c ec  ef 1d 76 7c 08 00 45 00   ····0·<· ··v|··E·
0010  00 44 08 7d 00 00 40 11  6a 69 80 6f 05 e3 80 6f   ·D·}··@· ji·o···o
0020  01 02 ce a0 00 35 00 30  08 05 66 2f 01 00 00 01   ·····5·0 ··f/····
0030  00 00 00 00 00 01 03 77  77 77 03 6d 69 74 03 65   ·······w ww·mit·e
0040  64 75 00 00 01 00 01 00  00 29 05 c0 00 00 00 00   du······ ·)······
0050  00 00                                              ··
```

We see from the above screenshot that nslookup actually sent two DNS queries and received two DNS responses. For the purpose of this assignment, in answering the following questions, ignore the second pair of query/response. You should instead focus on the first query and response messages.

1. What is the destination port for the DNS query message? What is the source port of the DNS response message?
2. To what IP address is the DNS query message sent?
3. Examine the DNS response message. How many "answers" are provided? What do each of these answers contain?

Part 2.3: Now load the following file: dns-wireshark-trace-3. This trace was generated by issuing the following command from a host with an IP address of 128.115.227:

```
nslookup -type=NS mit.edu
```

Answer the following question:

1. Examine the DNS response message. What MIT nameservers does the response message provide? Does this response message also provide the IP addresses of the MIT nameservers?

CS176A - Programming Assignment for Homework 2

Objective: There are a number of objectives to this assignment. The first is to introduce you to sockets through a simple socket programming assignment. Second, because you can use the Internet to look for examples, this assignment will help you see just how many network programming aids are available via the web. And finally, having just a bit of practical experience will put a lot of the protocol concepts we learn into perspective.

Assignment: The goal of this assignment is to implement a TCP client and server, and a UDP client and server (for a total of four different programs). **You must use the C programming language**. Your TCP or UDP client/server will communicate over the network and exchange data.

The server will start in *passive* mode listening on a specified port for a transmission from a client. Separately, the client will be started and will contact the server on a given IP address and port number that must be entered via the command line. The client will pass the server a string consisting of a sequence of characters. If the string contains anything but numbers, the server will respond with "Sorry, cannot compute!" and close the connection. If the string contains all numbers, the individual digits will be added together and returned as a string (see below for an example). If the server receives a string of numbers, it will (1) add the digits together, (2) send the value back to the client, and (3) does not close the connection unless the response is a single digit. The server repeats this process by adding the digits of the new number together and sending the response to the client until there is only one digit remaining in the sum. The server sends that final one digit sum, and then closes the connection. The maximum string length you need to accommodate is 128 characters. We will not test negative numbers. See below for the exact output.

We will provide you with a skeleton makefile for compiling your programs. Your code MUST work with the makefile you turn in, **and it MUST work on Gradescope**. **If it works on your own personal machine but not on Gradescope, you will not receive credit for the program.**

Examples

Server Details

Assume that you started a server on machine 128.111.49.44, listening to port number 32000. The syntax to start the server should look like the following:

```
csil-machine1> server 32000
```

- In this example, “server” should be replaced by one of the names given below in the Submission section (e.g. server_c_udp).
- The server should input exactly one command-line argument which is the port number to listen on.
- The server should accept connections on the **localhost** IP address for auto-grading purposes. To do this in C, you can use INADDR_LOOPBACK or INADDR_ANY for the IP address when calling bind.
- The server should not produce any output but should close the connection after interacting with a client.

- After responding to a client, the server should continue to listen on the port to receive more numbers from the client. The server should be able to handle receiving multiple numerical transmissions from the client.

Client Details

The syntax to start the client should look like the following:

```
csil-machine2>./client 128.111.49.44 32000
```

- In this example, “client” should be replaced by one of the names given below in the Submission section (e.g. client_c_udp).
- The IP address and port number are those which the server is listening on and which the client should use to connect to the server. Your program should input them as command-line arguments in the order shown above.

Client Input/Output for Non-Numeric Example

```
csil-machine2> ./client 128.111.49.44 32000
Enter string: Here is a string with letters.
From server: Sorry, cannot compute!
csil-machine2>
```

Client Input/Output for Numeric Example

```
csil-machine2> ./client 128.111.49.44 32000
Enter string: 123456789101234567891012345678910
From server: 138
From server: 12
From server: 3
csil-machine2>
```

Note: The communication is unidirectional. This means that the client sends a string once per session and then awaits server response(s). The client does NOT need to send back the updated sum to the server. The server iterates through all the sum until it sends back a single-digit response.

After each successful (single-digit sum) or unsuccessful (“sorry, cannot compute”) response, the server closes the connection, but keeps the socket open for other incoming requests. For TCP, this means closing the current session. UDP, however, is connection-less and hence requires no explicit termination.

Submission

You must use C for this assignment. You must turn in exactly four programs (all headers, etc. should be included in the one file), a makefile, and a README file. The files should be:

- Server in C using UDP (file name to turn in: server_c_udp.c)
- Client in C using UDP (file name to turn in: client_c_udp.c)
- Server in C using TCP (file name to turn in: server_c_tcp.c)
- Client in C using TCP (file name to turn in: client_c_tcp.c)
- Makefile (completed, with the appropriate file names added)
- README (file name to turn in: README.txt)

The assignment will be submitted to Gradescope and autograded. Upload **all** of the above files to Gradescope. Your Makefile should output executables named appropriately so that the autograder can run them using the following commands:

- ./server_c_udp <port>
- ./client_c_udp <ip> <port>
- ./server_c_tcp <port>
- ./client_c_tcp <ip> <port>

Make sure to include a README to indicate how your program works. In case your program is not fully functional, indicate what the problem might be. <u>In the README, be sure to indicate the name of your partner, if you had one. Please make a **group submission** on gradescope if you had a partner.</u>

NOTE: Pay attention to all of these directions carefully. Ensure all outputs including spacing and newlines match the examples above. An automated checker may be used and if there are any deviations, you will lose points!

Grading Guidelines

You may use pieces of code from the Internet or your textbook to help you do this assignment (e.g. basic socket code). However, this is just like citing a passage from a book, so if you copy code, you **must** cite it. To do this, put a comment at the beginning of your code that explains exactly what you have copied, who originally wrote it, and where it came from.

Below is a breakdown of points for this assignment. In addition to correctness, part of the points count towards how well code is written and documented. NOTE: good code/documentation does not imply that more is better. The goal is to be efficient, elegant and succinct!

- 8 pts: UDP client
- 8 pts: TCP client
- 8 pts: UDP server
- 8 pts: TCP server
- 8 pts: Documentation/Proper References/working makefile