Lab 02 - Networking Warmup - Part 1

Objectives. In this lab, you will learn how to perform some tasks over the Internet by hand. We expect this lab to take you between 1 and 2 hours to complete.

1 Set up GNU/Linux on your computer

SENG 4220's programming assignments and final project require the GNU/Linux operating system and a recent C++ compiler that supports the C++ 2017 standard. For this part of the course, we use a virtual machine provided by Prof. Keith Winstein from Stanford University. All required programs are already installed on this virtual machine. Please download this VM image and follow this instruction to set up this virtual machine on your VirtualBox.

2 Networking by Hand

Let's get started with using the network. You are going to do two tasks by hand: retrieving a Web page (just like a Web browser) and sending an email message (like an email client). Both of these tasks rely on a networking abstraction called a reliable bidirectional byte stream: you'll type a sequence of bytes into the terminal, and the same sequence of bytes will eventually be delivered, in the same order, to a program running on another computer (a server). The server responds with its own sequence of bytes, delivered back to your terminal.

2.1 Fetch a Web page

1. In a Web browser, visit http://cs144.keithw.org/hello and observe the result.
2. Now, you'll do the same thing the browser does, by hand.
   (a) On your VM, open a terminal and run `telnet cs144.keithw.org http`. This tells the telnet program to open a reliable byte stream between your computer and another computer (named cs144.keithw.org), and with a particular service running on that computer: the "http" service, for the Hyper-Text Transfer Protocol, used by the World Wide Web [1].
   If your computer has been set up properly and is on the Internet, you will see:
   ```
   Trying 104.196.238.229...
   Connected to cs144.keithw.org.
   Escape character is '^]'.
   ```
   (b) Type `GET /hello HTTP/1.1`. This tells the server the path part of the URL.
   (c) Type `Host: cs144.keithw.org`. This tells the server the host part of the URL.
   (d) Type `Connection: close`. This tells the server that you are finished making requests, and it should close the connection as soon as it finishes replying.
   (e) Hit the Enter key one more time. This sends an empty line and tells the server that you are done with your HTTP request.
   (f) If all went well, you will see the same response that your browser saw, preceded by HTTP headers that tell the browser how to interpret the response.

[1] The computer's name has a numerical equivalent (104.196.238.229, an Internet Protocol v4 address), and so does the service's name (80, a TCP port number). We'll talk more about these later.

Task 1

Now that you know how to fetch a Web page by hand, show us you can! Use the above technique to fetch the URL https://pages.cpsc.ucalgary.ca/ sina.keshvadi1/hello . You will receive a secret number. Take a screenshot of your work and write the secret key in your report.

2.2 Send yourself an email

Now that you know how to fetch a Web page, it's time to send an email message, again using a reliable byte stream to a service running on another computer.

1. Open your VM's terminal (make sure you are on TRU's network), run `telnet 104.47.75.228 smtp`. The "smtp" service refers to the Simple Mail Transfer Protocol, used to send email messages. If all goes well, you will see a success message with code 220.

2. First step: identify your computer to the email server. Type `HELO TRU`. Wait to a message with code 250.

3. Next step: who is sending the email? Type `MAIL FROM: YourEmail@tru.ca` . Replace YourEmail with your TRU email address. If all goes well, you will see "250 Sender ok".

4. Next: who is the recipient? For starters, try sending an email message to yourself. Type `RCPT TO: YourEmail@tr.ca` . Replace YourEmail with your own TRU email address. If all goes well, you will see "250 Recipient ok."

5. It's time to upload the email message itself. Type `DATA` to tell the server you're ready to start. If all goes well, you will see "354 End data with <CR><LF>.<CR><LF>".

6. Now you are typing an email message to yourself. First, start by typing the headers that you will see in your email client. Leave a blank line at the end of the headers.
   `From: YourEmail@tru.ca`
   `To: YourEmail@tru.ca`
   `Subject: Hello from SENG 4220!`

7. Type the body of the email message — anything you like. When finished, end with a dot on a line by itself: `.` . Expect to see something like: "250 Message accepted for delivery".

8. Type QUIT to end the conversation with the email server. Check your inbox and spam folder to make sure you got the email.

Task 2

Now that you know how to send an email by hand to yourself, try sending one to a friend or lab partner and make sure they get it [2]. Finally, show us you can send one to us. Use the above technique to send an email, from yourself, to webidsina@gmail.com .

2.3 Listening and Connecting

You've seen what you can do with telnet: a client program that makes outgoing connections to programs running on other computers. Now it's time to experiment with being a simple server: the kind of program that waits around for clients to connect to it.

1. In one terminal window, run `netcat -v -l -p 9090` on your VM.

[2]It's possible to give a phony "from" address. Electronic mail is a bit like real mail from the postal service, in that the accuracy of the return address is (mostly) on the honor system. You can write anything you like as the return address on a postcard, and the same is largely true of email. Please do not abuse this — seriously. With engineering knowledge comes responsibility! Sending email with a phony "from" address is commonly done by spammers and criminals so they can pretend to be somebody else.

2. Leave netcat running. In another terminal window, run `telnet localhost 9090` (also on your VM).
3. If all goes well, the netcat will have printed something like “Connection from localhost 53500 received!”.
4. Now try typing in either terminal window—the netcat (server) or the telnet (client). Notice that anything you type in one window appears in the other, and vice versa.
5. In the netcat window, quit the program by typing `Ctrl - C`. Notice that the telnet program immediately quits as well.

Task 3

Take a screenshot of your work and attach it to your report.

2.4 Traceroute

Traceroute is a Internet diagnostic tool for displaying the route (path) and measuring transit delays of packets across an Internet Protocol (IP) network. The history of each hop is recorded as the round-trip time (RTT), and the domain name or IP address of the router is resolved for each hop. Traceroute is typically run from a command line interface (CLI) on a computer. The basic syntax for the command is `traceroute [hostname or IP address]`. For example, to trace the route to google.com, you would run the command `traceroute google.com`.

It’s worth noting that, depending on the network architecture, some hops may not be visible, and the traceroute may not reach the final destination. In this case, the traceroute will stop after a certain number of hops and will display “* * *” in place of the missing information.

Task 4

Use traceroute to check the path to the following machines that run these Web services:

1. www.tru.ca
2. www.amazon.ca
3. www.book.jp

2.5 Whois

WHOIS is a query and response protocol that is widely used for querying databases that store the registered users or assignees of an Internet resource, such as a domain name, an IP address block, or an autonomous system, but is also used for a wider range of other information. For example, `whois tru.ca` will return the WHOIS information for the domain “tru.ca”, including the registrant’s name, contact information, and the dates of registration and expiration.

Task 5

1. Open a terminal on your VM.
2. For this task, you can use any domain name you would like, such as “tru.ca”. Type the command `whois tru.ca` and press enter.
3. Look at the WHOIS information that is returned. The WHOIS information will include details such as the registrant’s name, contact information, and the dates of registration and expiration.
4. Look for the “Registrar” section in the WHOIS information. This will tell you the company that is responsible for registering the domain.

5. Look for the “Name Servers” section in the WHOIS information. This will tell you the DNS servers that are associated with the domain.
6. Repeat the process for different domains, compare the information you get and try to find similarities or differences.
7. Attach an screenshot of your work to your report.

3 Implement a Text-based Web Browser

In the previous lab, we used telnet to fetch a Web page. In this section, we implement a C++ program to fetch a Web page. The source code is provided. Please download the web.cpp and follow the instructions:

1. Compile web.cpp using `g++ web.cpp -o web`
2. Run web using `./web`
3. Read the source code to understand how this app works. Ask questions if you have any.
4. Comment Example 1 in the code (lines 31 and 32), and un-comment Example 2. Compile and run again.
5. Try example 3.
6. Try another Website like amazon.ca.

Task 6

Discuss with your friend and answer the following questions:

1. Imagine the time that the Web only uses HTTP. Are you able to implement a Web browser? If yes, how? If not, why?
2. What other application you can develop with this idea of writing your own applciation to work with Web servers? Give me one example.
3. Why was your observation when you opened amazon.ca?

What is HTTPs?

Now you are so interested in Web, here are some notes about the HTTPs.

1. Identical to HTTP, except request and response messages are transmitted using SSL (Secure Sockets Layer) or its successor TLS (Transport Layer Security). (we will cover these topics in Web security)
2. HTTPS is used automatically for any URL beginning with “https:” instead of “http:”.
3. What HTTPS does for you:
   (a) The request and response messages are transmitted between the browser and server in encrypted form.
   (b) This prevents snoopers on the network from accessing private information in the messages, such as passwords or credit card numbers.
   (c) A certificate exchange allows the browser to identify the server it is communicating with. HTTPS doesn’t help the server to identify the browser.
4. HTTPS does not guarantee that the browser and server can trust each other. You just know that no-one else is listening.

5. HTTPS requires additional server setup: must create a certificate that identifies the server to the browser.
6. In designing Web applications you must use HTTPS whenever possible (SAFE DEFAULT).

4 Implement a Echo Web Server

In the previous section, you created a simple Web browser. Your program was a client program that made a connection to a Web server that is listening on port 80. In this section, you will implement a Server that is listening to your desire port number and provides service to other clients. The source code is provided. Please download the server.cpp and follow these instructions:

1. Compile server.cpp using `g++ server.cpp -o server`.
2. Run server using `./server 4220` (you can choose any number between 1000 to 56535).
3. Open another terminal.
4. Use telnet to connect to your server `telnet 127.0.0.1 4220` (if you chose another port number, use your port number instead of 4220).
5. Type message in Telnet and press Enter. Check the result on both terminals.
6. Try to read the server source code and ask questions if you have any.

Task 7

1. Are you able to implement a server that translates English to Spanish using this server? If yes, explain how, if no, explain why? (one paragraph is enough)
2. If you had enough time and resources, what interesting application do you want to implement by using this client/server architecture? (one paragraph is enough)
3. Is this echo communication secure?
4. Change the server to make each message UPPERCASE and return it back to the client.

Good Luck!