Introduction to Network Programming

Venkat Arun

In this assignment, we will learn how to build a simple networked application and give you hands-on experience with exploring the networks around you. This will introduce you to some common tools that, should you choose to become a professional computer scientist or engineer, you are likely to use often. We hope that what you learn here will encourage you to have fun and experiment with networks outside of class as well—while remaining ethical and not causing harm to others, of course.

This assignment assumes you are using a Linux machine. To that end, we will provide you with access to CloudLab machines, which you can set up by following the guide here: `https://www.cs.utexas.edu/~venkatar/`. You may choose to forego CloudLab and use a Linux installation on your laptop or a virtual machine instead. This assignment can also be completed using macOS or Windows WSL, though you may need to do some additional exploration to get everything working. This is part of the fun. A future assignment will assume you have Linux installed (specifically, to use the mahimahi.mit.edu network emulator), so we recommend setting that up as soon as possible.

TLDR: You need two machines capable of running common UNIX tools. These can be two CloudLab servers, two personal computers (e.g., yours and your teammate's), or one personal computer and one CloudLab server. Just remember that CloudLab is a limited resource, so please provision a server only when you are actively using it.

1 Looking at the interfaces in your computer

A computer usually has multiple network interfaces, which can be either physical or virtual. Physical interfaces correspond to ways for the computer to send packets to another computer. On your personal computer, you likely have an interface for WiFi and another for an ethernet (wired) port. Any packets sent to the WiFi interface, for instance, will be sent to the WiFi router, which will then forward them wherever necessary. This interface is usually named something like `wlp6s0` or `wlan`. You can list all interfaces using the command ifconfig. Run this command and try to understand its output. Use an LLM (like ChatGPT) if necessary. I used the prompt "explain the following output from ifconfig to me" in the default and free version of `perplexity.ai`. It did a good job of explaining most things but omitted what the `enp6s0` interface did. Asking this explicitly fixed the issue.

There will also be another interface called "localhost" or "lo," which is a virtual interface representing the local computer: any packets sent via that interface will appear on the same machine. You may have additional virtual interfaces. For instance, if you are using a VPN, all packets will be sent to that virtual interface and picked up by the VPN program running on your computer. These will then be encrypted and encapsulated into a connection to the VPN server.

In your report, include a screenshot of the output from your computer and an explanation of why each interface is configured the way it is. While we allow, and even encourage, the use of LLMs for understanding the output, you should write the report yourself with a much less detailed summary than what LLMs usually provide (you just need to describe what the interfaces do). More importantly, you are responsible for understanding everything in the assignment. We may quiz you at our discretion. Plus, we've designed this assignment to teach you skills that will be useful very frequently in your career. This is likely to be time well spent.

2 A simple chat "application"

NetCat We will use a command line utility called "netcat", abbreviated as "nc". Before connecting to a web server, let us first get two instances of `nc` to talk to each other. To do that, open two terminals on the same computer.

1. In one terminal, run `nc -l 0.0.0.0 8000`. The `-l` makes it "listen" for connections. This is what web servers do: wait for someone to connect to them. The next argument makes it listen to connections coming in from all interfaces. If you want it to only listen to one interface, you can specify the IP address of that interface instead. The `8000` refers to what port it should listen on. A port is a number between 0-65535 ($2^{16} - 1$). While IP addresses specify which computer a packet should be sent to, ports tell the operating system which application inside a computer should receive it. By convention, we use ports above 1024 for ad-hoc applications. Lower port numbers are reserved for special applications. For example, port 80 is used for HTTP-based web servers. Upon running this command, `nc` will do nothing and just wait. Let it wait and move to the other terminal.

2. In this terminal, run `nc 127.0.0.1 8000`. This makes it initiate a new connection to the specified address, causing it to connect to the other `nc` instance. The IP address `127.0.0.1` is a special address reserved for sending packets to the localhost interface. Now, whatever you type in one `nc` instance will appear in the other. Note: the text is only sent after you press "enter"

You have just created the simplest possible chat application! However, communicating within the same machine is not very useful. The next important step is to run `nc` on two *different* machines. Use `ifconfig` to find what IP

address to use. In your report, explain the steps you had to take to make this happens. You are likely to run into issues. Use internet searches or LLMs to debug them.

Here are two issues you might run into because internet service providers usually do not assign public IP addresses to personal computers. This means that they can create connections (i.e. what the second `nc` command did), but not accept (listen for) them. Depending on your setup, you need to do the following.

If you are using one CloudLab machine and one personal computer, run the server (i.e. the command with the `-l` flag) on the server remote machine. If both machines are personal computers, it is most likely to work only if you are connected to the same WiFi, or connected directly with each other using an ethernet cable.

3 Downloading a web page

Run `netcat example.com 80` and send the following text:

```
GET / HTTP/1.1
Host: example.com
```

This text is an HTTP (Hyper-Text Transfer Protocol) request. The protocol specifies how web browsers communicate with web servers. You can also modify the request to access other websites. When you send this text (press "enter" once more after the text to indicate the end of the header to the server), you should get the following response from the server that includes an HTTP response (first 14 lines) header followed by HTML that the browser uses to display the web page. You need not understand everything in this response.

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Age: 68437
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Thu, 15 Aug 2024 18:34:01 GMT
Etag: "3147526947"
Expires: Thu, 22 Aug 2024 18:34:01 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: ECAcc (lac/5591)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 1256

<!doctype html>
<html>
<head>
```

```
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans",

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 2em;
        background-color: #fdfdff;
        border-radius: 0.5em;
        box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
        color: #38488f;
        text-decoration: none;
    }
    @media (max-width: 700px) {
        div {
            margin: 0 auto;
            width: auto;
        }
    }
    </style>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You may use this
    domain in literature without prior coordination or asking for permission.</p>
    <p><a href="https://www.iana.org/domains/example">More information...</a></p>
</div>
</body>
</html>
```

In your report, specify what request you'd use to access `google.com`. You should get a very large response in return, because the web page includes a lot

of code. However, the concept remains the same.

What request will you use to perform a google search (say, you search for the word "hello")? Here are two hints to get you started. First, the simplified URL for that search is `www.google.com/search?q=hello`. If you put this into the browser, you will see the search page. Second, the path is indicated by the `/` field in the first line of the request we sent to example.com. You may need to use the internet searches and/or LLMs for this. Here, the path is `/search?q=hello`.

Note, this may not work for all websites because some servers expect a two-character line ending (carriage return + line feed) whereas many terminal emulators use only a single character (line feed). In this case, the servers will return a response saying "Bad Request". If this happens, you can press Ctrl-V + Ctrl-M before pressing "enter". In Python, you can write `'\r\n'` in python. However, you need not do this for example.com and google.com. We tested. This is the kind of annoying detail that is important to get things working in practice. Personally, I never remember them and always look them up. Fixing such issues is an important skill to learn as an engineer.

You may have noticed that in the previous segment, we had to use IP addresses whereas we got to use nice names like "example.com" and "google.com" here. This is because `nc` automatically translates such "domain names" to IP addresses. You can do this manually using the command "nslookup example.com". This will ask the Domain Name System (DNS) for the IP address corresponding to the domain name. DNS is a world-wide directory that translates between domain names and IP addresses. Once you get the IP address, you can use that instead of the domain name in `nc`. In your report, include the IP addresses for the two websites.

4 Download a web page using Python sockets

You do not need `nc` to create connections. You can also create them programmatically using the sockets API. Given below is a small python snippet that connects to a given IP address, sends a message and receives a response. Modify it to download and print example.com:

```
import socket

def send_message(ip_address, port, message):
    try:
        # Create a socket object
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            # Connect to the server
            s.connect((ip_address, port))

            # Send the message
            s.sendall(message.encode('utf-8'))
```

```
            # Receive the response
            response = s.recv(10000)

            return response.decode('utf-8')
    except Exception as e:
        return f"An error occurred: {e}"

# Example usage
response = send_message("127.0.0.1", 8000, "Hello world")
print(response)
```

For extra 20% credit, you can build an HTTP server in a very similar way. For instance, try to create one that displays the example.com web page and open it using a web browser.