Problem 1

'traceroute' is a computer network diagnostic tool for displaying the route (path) and measuring transit delays of packets across an Internet Protocol (IP) network. In this problem, you will use the 'traceroute' command to understand how packets route to a destination.

1. Run *traceroute* command to find a route to 'ucla.edu'. How many hops are there in between your local host to the destination? Copy and paste the result on your console in the answer box. (If you are using Windows Command Prompt, then use '*tracert*' command instead.)

2. Run *traceroute* command to find a route to 'columbia.edu'. Copy and paste the result into the answer box.

3. Compare two results in terms of the number of hops and the delays.

Problem 1 continued on next page...

1. There are 5 hops from my local host to the 'ucla.edu' destination:

```
$ traceroute ucla.edu
traceroute to ucla.edu (128.97.27.37), 64 hops max, 52 byte packets
1 s-169-232-219-1 (169.232.219.1) 1.995 ms 1.564 ms 1.287 ms
2 169.232.12.210 (169.232.12.210) 1.313 ms 1.682 ms 1.484 ms
3 dr00f2.csb1--cr00f2.csb1.ucla.net (169.232.4.52) 1.861 ms 1.294 ms 1.799 ms
4 cr00f2.csb1--sr02f2.csb1.ucla.net (169.232.8.7) 1.788 ms 1.623 ms 1.595 ms
5 128.97.27.37 (128.97.27.37) 1.850 ms !Z 1.806 ms !Z 1.571 ms !Z
```

2. There are 18 hops from my local host to the 'columbia.edu' destination:

```
$ traceroute columbia.edu
traceroute to columbia.edu (128.59.105.24), 64 hops max, 52 byte packets
1 s-169-232-219-1 (169.232.219.1) 2.613 ms 1.711 ms 4.908 ms
2 169.232.12.210 (169.232.12.210) 6.967 ms 2.028 ms 2.437 ms
3 dr00f2.csb1--cr00f2.csb1.ucla.net (169.232.4.52) 1.606 ms 1.834 ms 3.207 ms
4 cr00f2.csb1--bd11f1.anderson.ucla.net (169.232.4.5) 1.931 ms 2.718 ms 2.892 ms
5 lax-hpr2--ucla-10ge.cenic.net (137.164.27.5) 4.672 ms 2.440 ms 2.303 ms
6 hpr-i2--lax-hpr2-r&e.cenic.net (137.164.26.201) 2.234 ms 4.374 ms 1.902 ms
7 ae-1.4079.rtsw.salt.net.internet2.edu (162.252.70.115) 15.273 ms 15.921 ms 18.562 ms
8 ae-5.4079.rtsw.kans.net.internet2.edu (162.252.70.144) 35.296 ms 35.311 ms 37.523 ms
9 ae-3.4079.rtsw.chic.net.internet2.edu (162.252.70.140) 45.644 ms 46.288 ms 45.609 ms
10 ae-2.4079.rtsw.eqch.net.internet2.edu (162.252.70.133) 45.657 ms 45.897 ms 48.649 ms
11 ae-1.4079.rtsw.clev.net.internet2.edu (162.252.70.130) 54.981 ms 55.093 ms 54.511 ms
12 buf-9208-i2-clev.nysernet.net (199.109.11.33) 59.421 ms 59.075 ms 58.744 ms
13 syr-9208-buf-9208.nysernet.net (199.109.7.193) 76.602 ms 75.143 ms 61.937 ms
14 nyc-9208-syr-9208.nysernet.net (199.109.7.162) 71.011 ms 69.302 ms 67.636 ms
15 columbia.nyc-9208.nysernet.net (199.109.4.14) 68.117 ms 71.961 ms 70.197 ms
16 cc-core-1-x-nyser32-gw-1.net.columbia.edu (128.59.255.5) 70.660 ms 70.908 ms 68.090 ms
17 cc-conc-1-x-cc-core-1.net.columbia.edu (128.59.255.210) 69.158 ms 69.367 ms 81.260 ms
18 caho.columbia.edu (128.59.105.24) 68.797 ms 68.890 ms 71.011 ms
```

3. The route from my local host to 'ucla.edu' has fewer hops than the route from my local host to 'columbia.edu' ($5 < 18$ respectively) and in general the average RTT time for a packet to reach the server and return to my computer is longer for most IP addresses in the 'colmbia.edu' route. In fact, some 'columbia.edu' RTT delays range from 30ms to 70ms while the 'ucla.edu' RTT delays never exceed 2 ms. These results are intuitive because UCLA is much closer to my local host compared to Columbia. UCLA's web server must be geographically near the school to minimize hops and delays from users located nearby accessing 'ucla.edu', and Columbia's web server must be geographically near Upper Manhattan, New York.

Problem 2

Host A is sending real-time voice over a packet-switched network. Host A converts analog voice to a digital 128 kbps bit stream on the fly. Host A then groups 1,600 bytes into a packet. Assume that the 1,600 bytes packet already includes all headers. There is one link between Hosts A and B; its transmission rate is 3 Mbps and its propagation delay is 20 msec. As soon as Host A gathers a packet, it sends it to Host B. As soon as Host B receives an entire packet, it converts the packets bits to an analog signal. How much time elapses from the time a bit is created (from the original analog signal at Host A) until the bit is decoded (as part of the analog signal at Host B)? In this problem, do not consider acknowledgement (response) from Host B.

The total elapsed time can be constructed from the following delays:
a = time it takes host A to create a 1,600 byte packet from a 128 kbps bit stream
b = time it takes host A to transmit an entire 1,6000 byte packet to host B
c = delay it takes for a bit to propagate from host A to host B by using the propagation delay of 20 ms

$a = \frac{(1{,}600 \times 8) bits}{(128 \times 1{,}000) \frac{bits}{second}} = 0.1s \times \frac{1{,}000ms}{1s} = 100ms$
$b = \frac{(1{,}600 \times 8) bits}{(3 \times 10^6) \frac{bits}{second}} = 0.004266s \times \frac{1{,}000ms}{1s} = 4.267ms$
$c = 20ms$

Total elapsed time $= a + b + c = 100ms + 4.267ms + 20ms = 124.267ms$

Problem 3

Tow hosts, A and B are separated by 20,000 kilometers and are connected by a direct link of $R = 2Mbps$. Suppose the propagation speed over the link is $2.5 * 10^8 meters/sec$.

1. Consider sending a file of 800,000 bits from Host A to Host B. Suppose the file is sent continuously as one large message. What is the maximum number of bits that will be in the link at any given time?
2. How long does it take to send the file, assuming it is sent continuously?
3. Suppose now the file is broken up into 20 packets with each packet containing 40,000 bits. Suppose that each packet is acknowledged by the receiver and the transmission time of an acknowledgment packet is negligible. Finally, assume that the sender cannot send a packet until the preceding one is acknowledged. How long does it take to send the file?

1. Let:
   p = the propagation delay
   d = the distance between A and B in meters
   s = the propagation speed over the link
   $p = \frac{d}{s} = \frac{20{,}000km \times \frac{1{,}000m}{1km}}{(2.5 \times 10^8)\frac{m}{s}} = 0.08s$
   The link can send $R = 2Mbps = (2 \times 10^6)\frac{bits}{s}$ at once
   So the max number of bits in the link at once is $p \times R = 0.08s \times (2 \times 10^6)\frac{bits}{s} = 160,000$ bits
   Since 800,000 bits > 160,000 bits, there will at most be 160,000 bits on the link at any given time
2. To send a file that contains $L = 800,000$ bits, there will be a transmission delay t and a propagation delay p, where $t = \frac{L}{R} = \frac{800{,}000bits}{(2*10^6)\frac{bits}{s}} = 0.4s$ and p is already computed from part 1.
   Thus, the total time to send the file $= t + p = 0.4s + 0.08s = 0.48s$
3. Each packet contains 40,000 bits. This means that the time needed to transmit a packet is
   $= \frac{40{,}000bits}{(2 \times 10^6)\frac{bits}{s}} = 0.02s$
   Each packet requires 0.02s to transmit, 0.08s to propagate, and another 0.08s for the acknowledgement packet to be propagated back.
   Thus, the file takes $(0.02s + 0.08s + 0.08s) \times 20 = 3.6s$ to be sent

Problem 4

Alice and Bob are working remotely on a course project and are using `git` as the version control software.

1. Is it true that one must have GitHub/GitLab account to use git?
2. What is(are) the command(s) to initialize a local git repository?
3. Do Alice and Bob both must initialize local git repository? If no, what are the alternative?
4. Let's consider that Alice modified the file `server.cpp`:
   (a) What git commands Alice needs to save modifications in the local git repository
   (b) What git commands Alice needs to upload saved modifications to GitHub
   (c) What git commands Bob needs to get Alice's changes and apply them to the local repository
5. Let's consider that both Alice and Bob modified the file `server.cpp` and Alice was first to successfully upload saved modifications (commit) to GitHub
   (a) Can Bob upload his changes without any additional actions? If no, why?
   (b) If actions needed, list git commands that Bob will need to use to share his modifications with Alice.

1. No, one can use `git` *without* a GitHub/GitLab account
2. `git init` : used to initialize a new local repository
3. No. Only one person needs to initialize a local `git` repository, and the other person can use the `git clone` command to clone the repository into their machine.
4. (a)
   ```
   git add server.cpp
   git commit -m "Some commit message"
   ```
   (b)
   ```
   git push
   ```
   (c) Bob can either perform a `git pull` or a `git fetch` followed by a `git merge`. These two alternatives are equivalent and allows Bob to get Alice's changes and apply them to the local repository.
5. (a) No, Bob must first perform a `git pull`. After pulling, if there are merge conflicts with the code (as in both Alice and Bob modified the same lines of code), Bob may have to manually fix the merge conflicts generated by `git` and then run a `git add <modified_files>` command followed by a `git commit`. Finally, once the changes are pulled in and no merge conflicts exist, Bob can upload his changes with a `git push` command.
   (b)
   ```
   git pull
   ```
   If there are merge conflicts, manually fix them.
   ```
   git add <modified_files>
   git commit -m "Some commit message"
   git push
   ```

Problem 5

You will learn some basic usages of `Vagrant` in your projects.

1. What is Vagrant mainly used for?
2. What is VirtualBox used for?
3. What is *Vagrantfile*?
4. List at least five commands you can use with Vagrant.

1. `Vagrant` builds machine environments in a single, smooth workflow. It is mainly used to automatically set up a virtual machine to run your code on so that you don't need to manually configure the set up process.
2. VirtualBox allows users to run multiple operating systems at the same time. This is advantageous to developers because extensive testing on different operating systems can ensure that all users have a smooth interaction with your code, regardless of their respective operating system of choice.
3. A `Vagrantfile` is a configuration file written in Ruby that details the project-specific set up details for your code, such as describing which virtual machines are required and how to configure these machines.
4. `vagrant ssh` : SSH into a virtual machine
   `vagrant up` : Start virtual machine
   `vagrant share` : Share virtual machine with a temporary, unique URL
   `vagrant halt` : Halt virtual machine
   `vagrant destroy` : Destroy your virtual machine