Designing subnets

Fraida Fund

31 MARCH 2017 on education, routing

Your task in this experiment is to set up subnets in a few small LANs to meet given design requirements.

It should take about 60-120 minutes to run this experiment, *after* you work out what part of the address space to assign to each LAN.

You can run this experiment on GENI or on CloudLab. Refer to the testbed-specific prerequisites listed below.

GENI-specific instructions: Prerequisites

To reproduce this experiment on GENI, you will need an account on the GENI Portal, and you will need to have joined a project. You should have already uploaded your SSH keys to the portal and know how to log in to a node with those keys.

Cloudlab-specific instructions: Prerequisites

To reproduce this experiment on Cloudlab, you will need an account on Cloudlab, you will need to have joined a project, and you will need to have set up SSH access.

Background

Computer networks are often divided into subnets, to reduce the size of the broadcast domain or to create organizational, administrative and security boundaries in the network.

The following excerpt from William Stallings, "Data and Computer Communications" explains:

The concept of subnet was introduced to address the following requirement. Consider an internet that includes one or more WANs and a number of sites, each of which has a number of LANs. We would like to allow arbitrary complexity of interconnected LAN structures within an organization while insulating the overall internet against explosive growth in network numbers and routing complexity. One approach to this problem is to assign a single network number to all of the LANs at a site. From the point of view of the rest of the internet, there is a single network at that site, which simplifies addressing and routing. To allow the routers within the site to function properly, each LAN is assigned a subnet number. The host portion of the internet address is partitioned into a subnet number and a host number to accommodate this new level of addressing.

Within the subnetted network, the local routers must route on the basis of an extended network number consisting of the network portion of the IP address and the subnet number. The bit positions containing this extended network number are indicated by the address mask. The use of the address mask allows the host to determine whether an outgoing datagram is destined for a host on the same LAN (send directly) or another LAN (send datagram to router). It is assumed that some other means (e.g., manual configuration) are used to create address masks and make them known to the local routers.

Table 18.3a shows the calculations involved in the use of a subnet mask. Note that the effect of the subnet mask is to erase the portion of the host field that refers to an actual host on a subnet. What remains is the network number and the subnet number. Figure 18.8 shows an example of the use of subnetting. The figure shows a local complex consisting of three LANs and two routers. To the rest of the internet, this complex is a single network with a Class C address of the form 192.228.17.x, where the leftmost three octets are the network number and the rightmost octet contains a host number x. Both routers R1 and R2 are configured with a subnet mask with the value 255.255.255.224 (see Table 18.3a). For example, if a datagram with the destination address 192.228.17.57 arrives at R1 from either the rest of the internet or from LAN Y, R1 applies the subnet mask to determine that this address refers to subnet 1, which is LAN X, and so forwards the datagram to LAN X. Similarly, if a datagram with that destination address arrives at R2 from LAN Z, R2 applies the mask and then determines from its forwarding database that datagrams destined for subnet 1 should be forwarded to R1. Hosts must also employ a subnet mask to make routing decisions.

The default subnet mask for a given class of addresses is a null mask (Table 18.3b), which yields the same network and host number as the non-subnetted address.

(a) Dotted decimal and binary representations of IP address and subnet masks

	Binary Representation	Dotted Decimal
IP address	11000000.11100100.00010001.00111001	192.228.17.57
Subnet mask	11111111.11111111.11111111.11100000	255.255.255.224
Bitwise AND of address and mask (resultant network/subnet number)	11000000.11100100.00010001.00100000	192.228.17.32
Subnet number	11000000.11100100.00010001.001	1
Host number	00000000.00000000.00000000.00011001	25

(b) Default subnet masks

	Binary Representation	Dotted Decimal
Class A default mask	11111111.00000000.00000000.00000000	255.0.0.0
Example Class A mask	11111111.11000000.00000000.00000000	255.192.0.0
Class B default mask	11111111.11111111.00000000.00000000	255.255.0.0
Example Class B mask	11111111.11111111.11111000.00000000	255.255.248.0
Class C default mask	11111111.11111111.11111111.00000000	255. 255. 255.0
Example Class C mask	11111111.11111111.11111111.11111100	255. 255. 255.252

Table 18.3 from Stallings, "Data and Computer Communications"

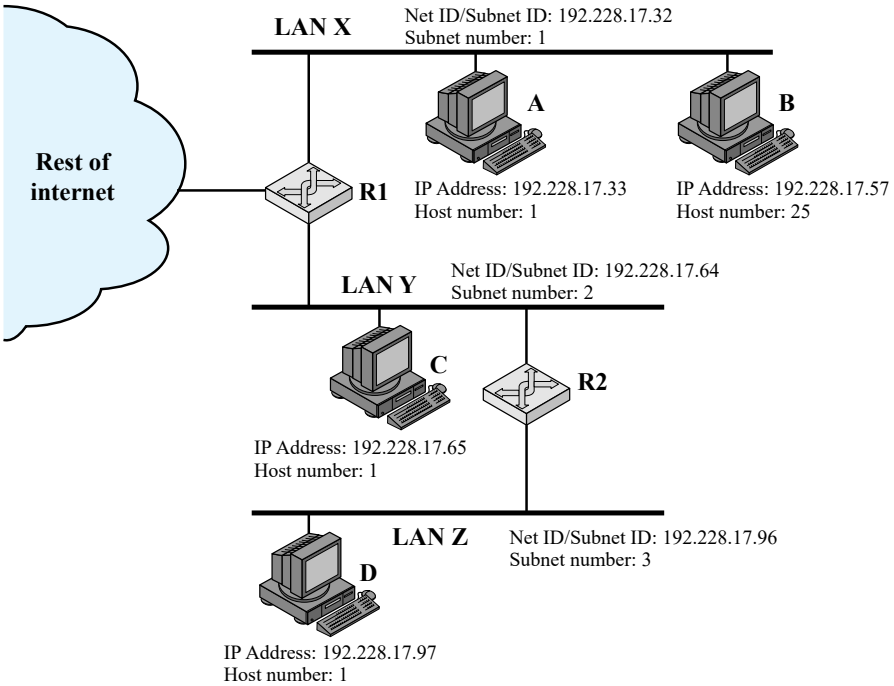


Figure 18.8 from Stallings, "Data and Computer Communications"

When designing subnets, you typically have some constraints that you must satisfy:

- You will have a limited IP space available.
- You will need to make each subnet large enough to support the number of hosts it is expected to have.
- You will need to think about how traffic will be routed between subnets.

As an example, consider a scenario where we have to design subnets to partition the IP address space 10.189.24.0/24 (i.e., the range from 10.189.24.0-10.189.24.255) among three LANs, using subnets. Furthermore, we are told that LAN A and LAN B must support 10 hosts each (not including the gateway), and LAN C must support 100 hosts (not including the gateway). In this section, we'll work out the solution to this problem, and show how we realize our design on GENI; in the next section, you'll have to solve a similar problem on your own.

The first step is to identify the smallest subnet mask that can support the number of hosts required for each LAN.:

- To be able to address 100 hosts on LAN C, we need 7 bits to be used for the host part of the address ($2^7 = 128$. One address will be reserved for the gateway, one address is the network address, and one address is the broadcast address, leaving 125 addresses for the other hosts on the network.) Thus, LAN C should have a subnet mask of 255.255.255.128, which leaves the 7 rightmost bits free for the host address:

```
11111111.11111111.11111111.10000000
```

- To be able to address 10 hosts each on LAN A and LAN B, we need 4 bits to be used for the host part of the address ($2^4 = 16$. One address will be reserved for the gateway, one address is the network address, and one address is the broadcast address, leaving 13 addresses for the other hosts on the network.) Thus, LAN A and LAN B should have a subnet mask of 255.255.255.240, which leaves the 4 rightmost bits free for the host address:

```
11111111.11111111.11111111.11110000
```

Now we are ready to begin allocating addresses to subnets. First, we'll convert the 10.189.24.0/24 address range from dotted decimal notation to binary:

```
00001010.10111101.00011000.00000000
```

This will be the network address for the *first* subnet we allocate. It's usually easier to begin by allocating the largest subnet first, so we will start with LAN C.

We've already determined the network address (10.189.24.0) and the subnet mask (255.255.255.128) for LAN C. Next, we'll compute the broadcast address for the subnet. This is the bitwise OR of the network address and the inverse of the subnet mask:

```
00001010.10111101.00011000.00000000
00000000.00000000.00000000.01111111
-----------------------------------
00001010.10111101.00011000.01111111
```

which is 10.189.24.127 in dotted decimal notation. The highest IP address that can be assigned to a host in the network is 10.189.24.126 (1 less than the broadcast address). The smallest IP address that can be assigned to a host in the network is 10.189.24.1 (1 more than the network address), and by convention this will be assigned to the gateway. That leaves the range 10.189.24.2-10.189.24.126 for other hosts in the network.

Next, we'll turn our attention to LAN B, which must support 10 hosts. We've already used up some of our IP address space, so we are now left with the range 10.189.24.128-10.189.24.255 to work with. The address at the lower end of this range 10.189.24.128, will be the network address for the next subnet. In binary, this is:

```
00001010.10111101.00011000.10000000
```

We've already determined that the subnet mask for LAN B should be 255.255.255.240. Next, we'll compute the broadcast address for the subnet. This is the bitwise OR of the network address and the inverse of the subnet mask:

```
00001010.10111101.00011000.10000000
00000000.00000000.00000000.00001111
-----------------------------------
00001010.10111101.00011000.10001111
```

or 10.89.124.143 in dotted decimal notation. The highest IP address that can be assigned to a host in the network is 10.189.24.142 (1 less than the broadcast address). The smallest IP address that can be assigned to a host in the network is 10.189.24.129 (1 more than the network address), and by convention this will be assigned to the gateway. That leaves the range 10.189.24.130-10.189.24.142 for other hosts in the network.

Last, we will allocate addresses for LAN A. We've used up the lower part of our allocated address space, through 10.189.24.143, so we are now left with the range 10.189.24.144-10.189.24.255 to work with. The address at the lower end of this range 10.189.24.144, will be the network address for the next subnet. In binary, this is:

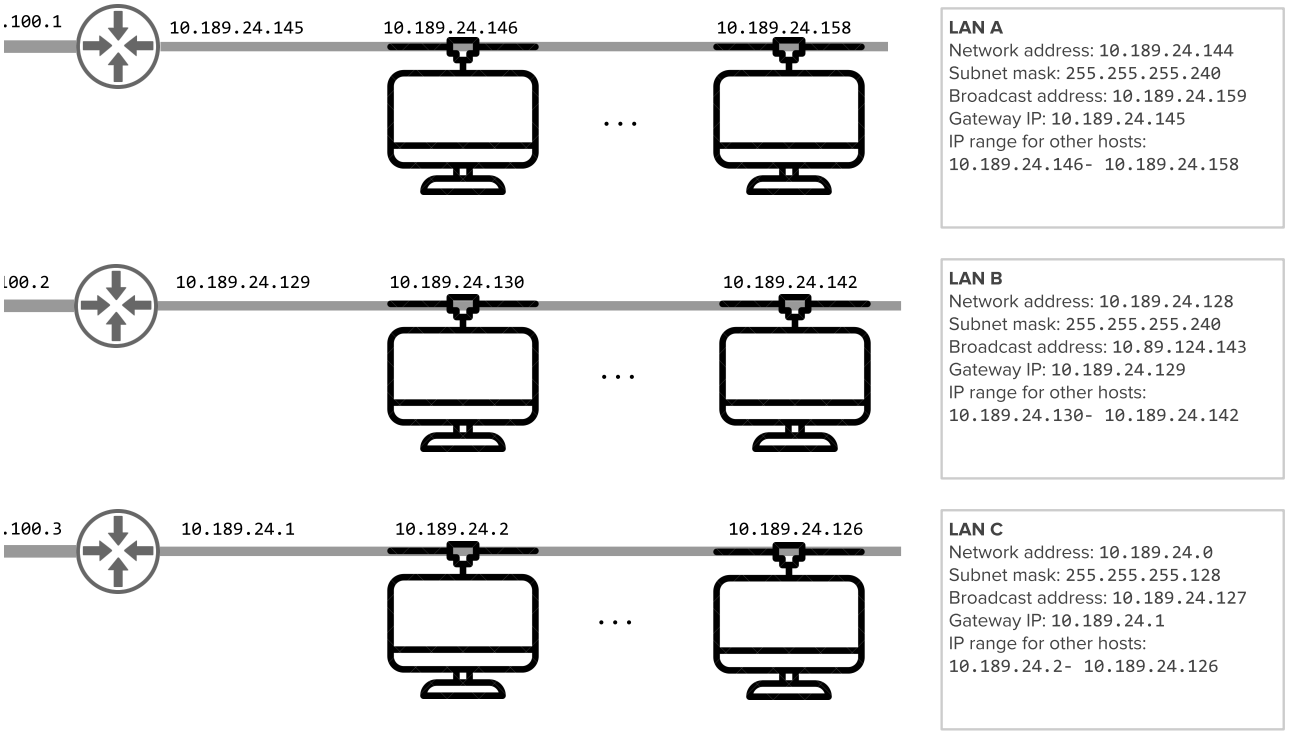
```
00001010.10111101.00011000.10001000
```

We've already determined that the subnet mask for LAN A should be 255.255.255.240. Next, we'll compute the broadcast address for the subnet. This is the bitwise OR of the network address and the inverse of the subnet mask:

```
00001010.10111101.00011000.10010000
00000000.00000000.00000000.00001111
-----------------------------------
00001010.10111101.00011000.10011111
```

or 10.89.124.159 in dotted decimal notation. The highest IP address that can be assigned to a host in the network is 10.189.24.158 (1 less than the broadcast address). The smallest IP address that can be assigned to a host in the network is 10.189.24.145 (1 more than the network address), and by convention this will be assigned to the gateway. That leaves the range 10.189.24.146-10.189.24.158 for other hosts in the network.

The following diagram shows our overall network design:



(we assume the routers all have a second interface connected to a routing network, 10.10.100.0/24, as in the experiment below.)

We use `ifconfig` to assign IP addresses and netmasks to each host and router on a GENI topology that mimics this setup:

After setting up these interfaces, hosts within the same LAN can reach one another, but not between LANs. To enable connectivity between LANs, we added routes as follows, and verified that a host in LAN A can reach all of the other hosts:

Run my experiment

For this experiment, you will reserve a topology that includes three routers (A, B, and C) and two hosts connected to each router. The routers will already be configured with IP addresses (in the 10.10.100.0/24 subnet) on the link that connects the routers to one another. However, it will be up to you to design subnets for the small LAN connected to each router.

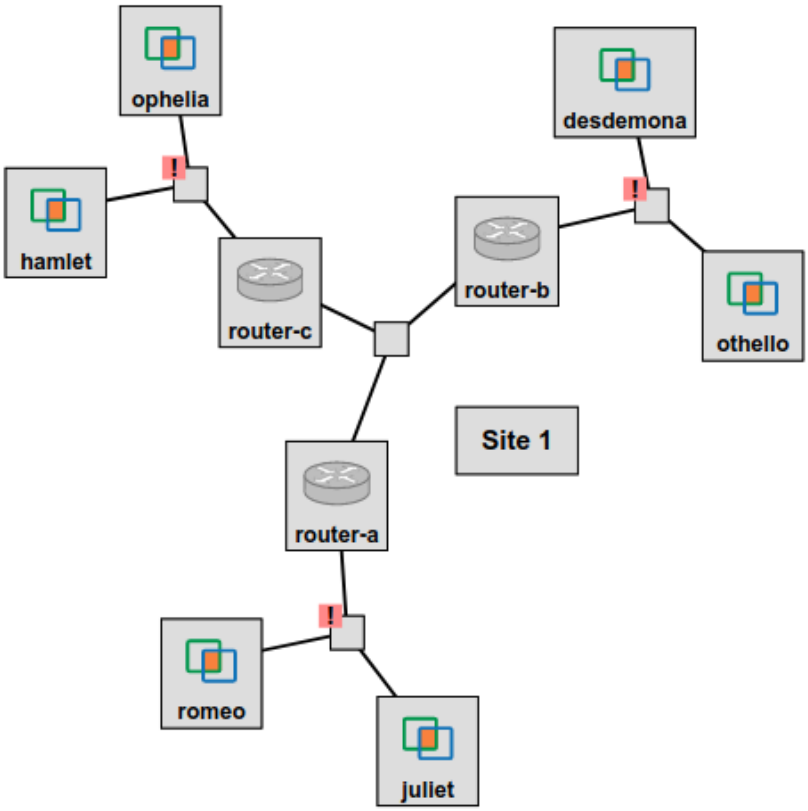
Follow the instructions for the testbed you are using (GENI or Cloudlab) to reserve the resources and log in to each of the hosts in this experiment.

GENI-specific instructions: Reserve resources

To reserve these resources on GENI, create a new slice on GENI. Click on “Add Resources”, and load the RSpec from the following URL:

https://git.io/Jthun

This will load the following topology onto your canvas:



Then, select an InstaGENI site to bind to, and reserve your resources.

Your topology may have some red warning indicators on the LANs. This is not a problem - it's just a warning that some IP addresses are duplicated in the topology. In this case, that's intentional (these interfaces are assigned an address of "0.0.0.0", which results in them having no IPv4 address.)

Click on "Site 1" and choose an InstaGENI site to bind to, then reserve your resources. Wait for your nodes to boot up (they will turn green in the canvas display on your slice page in the GENI portal when they are ready). Then, use the details given in the GENI Portal to SSH into each node.

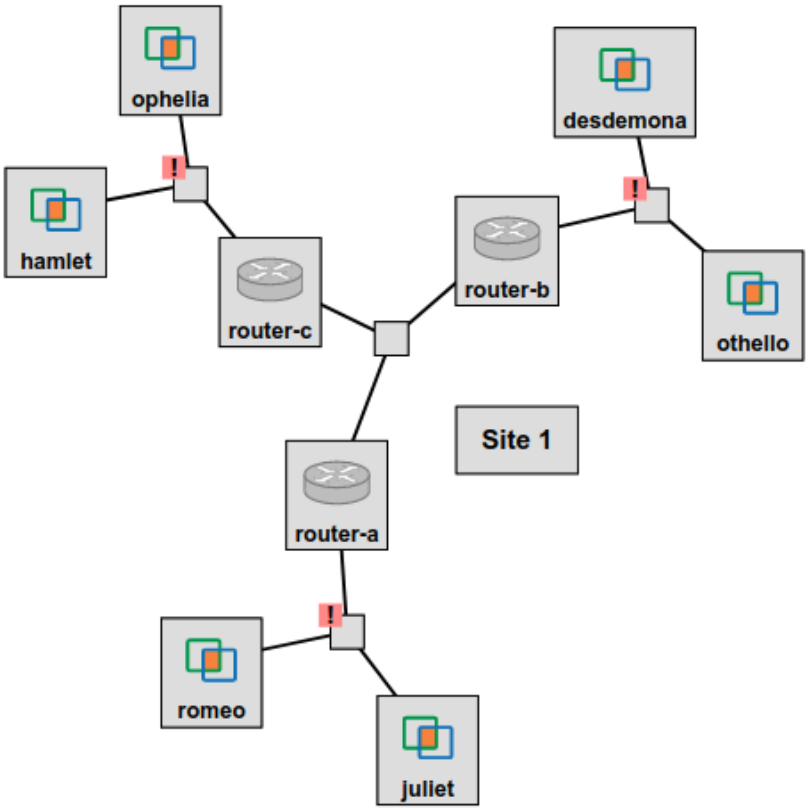
When you have logged in to each node, continue to the Challenge: design subnets section.

Cloudlab-specific instructions: Reserve resources

To reserve resources on Cloudlab, open this profile page:

https://www.cloudlab.us/p/nyunetworks/education?refspec=refs/heads/design_subnets

This profile instantiates the following topology:



Click "next", then select the Cloudlab project that you are part of and a Cloudlab cluster with available resources. (This experiment is compatible with any of the Cloudlab clusters.) Then click "next", and "finish".

Wait until all of the sources have turned green and have a small check mark in the top right corner of the "topology view" tab, indicating that they are fully configured and ready to log in. Then, click on "list view" to get SSH login details for the client, router, and server hosts. Use these details to SSH into each.

When you have logged in to each node, continue to the Challenge: design subnets section.

Challenge: design subnets

Your challenge is to design subnets for these LANs, with the following design constraints:

- The available IP address space is the range 10.10.172.0-10.10.172.255. You may only assign IP addresses in this range.
- The LAN connected to router A must be able to support at least 50 hosts.
- The LAN connected to router B must be able to support at least 75 hosts.
- The LAN connected to router C must be able to support at least 20 hosts.

You must design and set up subnets to accommodate these requirements.

For each subnet, you should identify:

- the subnet mask.
- the network address. This will be the bitwise AND of the IP address of any host on the network, and the subnet mask.
- the smallest IP address that may be assigned to a host in the subnet (host number 1). This is the network address, plus 1.
- the broadcast address for the subnet. This is the bitwise OR of the network address and the inverse of the subnet mask.
- the highest IP address that may be assigned to a host in the subnet. This is the broadcast address, minus 1.

There is more than one correct solution - any solution that satisfies the requirements above is acceptable! Once you have made your design decisions, complete the following table:

Hosts	Subnet mask	Network address	Smallest host address	Highest host address	Broadcast address	Notes
LAN A: romeo and juliet						Should support at least 50 hosts
LAN B: othello and desdemona						Should support at least 75 hosts
LAN C: hamlet and ophelia						Should support at least 20 hosts

Implement your design on each LAN

Next, you are going to configure the network interface on each host and router on each of the three LANs, to implement the design that you described above.

In each LAN, assign an IP address and subnet mask to every host interface connected to the LAN. Assign the lowest IP address in the subnet (host number 1) to the LAN-facing interface of the router (this is a common convention); assign the highest IP address in the subnet to one host, and any other IP address in the subnet to the other host.

To assign an IP address and subnet mask on a host, use the following command:

```
sudo ifconfig IFACE IP netmask NETMASK
```

substituting an IP and NETMASK in dotted decimal notation, and an interface name (e.g. `eth1` or `eth2`) for IFACE.

At this point, each host in a LAN should be able to reach every other host in the *same* LAN. You can run

```
ping -c 5 IP
```

(substituting the destination IP address of the other host) to verify this. However, you will not be able to send and receive traffic between different LANs - for this, you'll need to add routing rules.

Add routing rules

On each host, add one or more routing rules to describe how to reach the other two LANs.

One way to do this is to add a route for *each* destination LAN, that looks something like this

```
sudo route add -net ADDRESS netmask NETMASK gw GATEWAY
```

where:

- ADDRESS is the network address of the *destination* LAN, in dotted decimal notation
- NETMASK is the subnet mask of the *destination* LAN, in dotted decimal notation
- GATEWAY is the IP address of the LAN-facing interface of the router on the *source* LAN, also in dotted decimal notation.

For example, if adding a rule on a host in LAN A to enable it to reach hosts on LAN C, you would use the network address of LAN C for ADDRESS, the netmask of LAN C for NETMASK, and the IP address of the LAN-facing interface of router A for GATEWAY.

As an alternative, you can take advantage of the *longest prefix matching* rule, and add just one route on each host for *all* traffic to the other two LANs:

```
sudo route add -net 10.10.172.0 netmask 255.255.255.0 gw GATEWA
```

where GATEWAY is the IP address of the LAN-facing interface of the router on the *source* LAN, in dotted decimal notation.

In addition to adding rules on every host, you will also need to add a rule on each router. For these rules, the GATEWAY address should be the IP address of the router for the *destination* LAN (on the WAN that connects the routers). For example, if setting up a rule on router A for routing to the C LAN, I will use the IP address 10.10.100.3 as the GATEWAY, since it is the IP address of router C on the WAN that connects the three routers.

If you make a mistake, you can delete a rule using the same syntax as when you added it, but replace the `add` with `del`. To see the rules that are already in place, use

```
route -n
```

When you've set up these rules correctly, every host in every LAN should be able to `ping` every host in every *other* LAN in your topology. You can also run

```
traceroute IP
```

or

```
mtr --report --no-dns IP
```

on any host to see the path it uses to reach a destination IP address.

Your `traceroute` or `mtr` will be unable to reach the intermediate routers because there are no routing rules set up for the 10.10.100.0/24 network. It will show those hops as `????` or `****`. If you'd like it to show the addresses of intermediate routers, you can add another rule on each (non-router) host:

```
sudo route add -net 10.10.100.0 netmask 255.255.255.0 gw GATEWA
```

substituting the LAN-facing IP address of the router on the *same* LAN where it says GATEWAY. Then, you'll be able to see the intermediate hops on the router network (10.10.100.0/24) in your `traceroute` output.

Clean up your resources

When you've finished, delete your resources on the GENI Portal to free them for other experimenters.

Notes

Exercise

Table of design choices: Complete the following table with the subnet design choices for your network:

Hosts	Subnet mask	Network address	Smallest host address	Highest host address	Broadcast address	Notes
LAN A: romeo and juliet						Should support at least 50 hosts
LAN B: othello and desdemona						Should support at least 75 hosts
LAN C: hamlet and ophelia						Should support at least 20 hosts

Network interface configuration: After you have configured the network interfaces and routes according to your subnet design, show the network interface configuration as follows:

- On each of the six hosts - romeo, juliet, hamlet, ophelia, othello, desdemona - show the output of `ifconfig eth1`.
- On each of the routers, show the output of `ifconfig`.

Routing within and between subnets: After you have configured the network interfaces and routes according to your subnet design, show the output of `route -n` on the "romeo" host. Annotate the output to indicate:

1. which rule will apply to traffic from romeo to juliet
2. which rule will apply to traffic from romeo to hamlet
3. which rule will apply to traffic from romeo to ophelia
4. which rule will apply to traffic from romeo to othello
5. which rule will apply to traffic from romeo to desdemona

If there are multiple matching rules, indicate the *one* that applies in each case according to the longest prefix matching rule.

Also, show the output of `traceroute` or `mtr` *from* "romeo" to *each* of the five other hosts (i.e. run the `traceroute` or `mtr` command five times, with a different destination IP address each time, and save the output each time). Please indicate which host is the destination in each case.