Final Project

Name: _____

Exam time: 2:30 pm – 4:30 pm, Dec. 11, 2020. Late submission will result in either 0 pt or late penalty.

Total: 24 pts

1 Cisco Packet Tracer: Supernetting and Routing Table

We've learned the subnetting concept in class. In this project, we're going to learn the subnetting concept and routing table, and then realize it through Packet Tracer.

First, let's take a look at the following tutorial on supernetting.

Supernetting Tutorial: - Supernetting Explained with Examples

1.1 Supernetting Example

Given two subnetworks 140.113.1.0/24 and 140.113.2.0/24. To generate a single entry (supernetwork) which will cover the two subnetworks, we find the **longest prefix matching** between the two.

140.113.1.0/24 → [10001100 01110001 000000]01 00000000

140.113.2.0/24 → [10001100 01110001 000000]10 00000000

Comparing these two subnetworks, we see the leading 22 bits are exactly the same.

So, the supernetwork will have the save leading 22 bits as the network portion, while the rest of the bits are the host portion.

→ 140.113.0.0/22 (supernetwork)

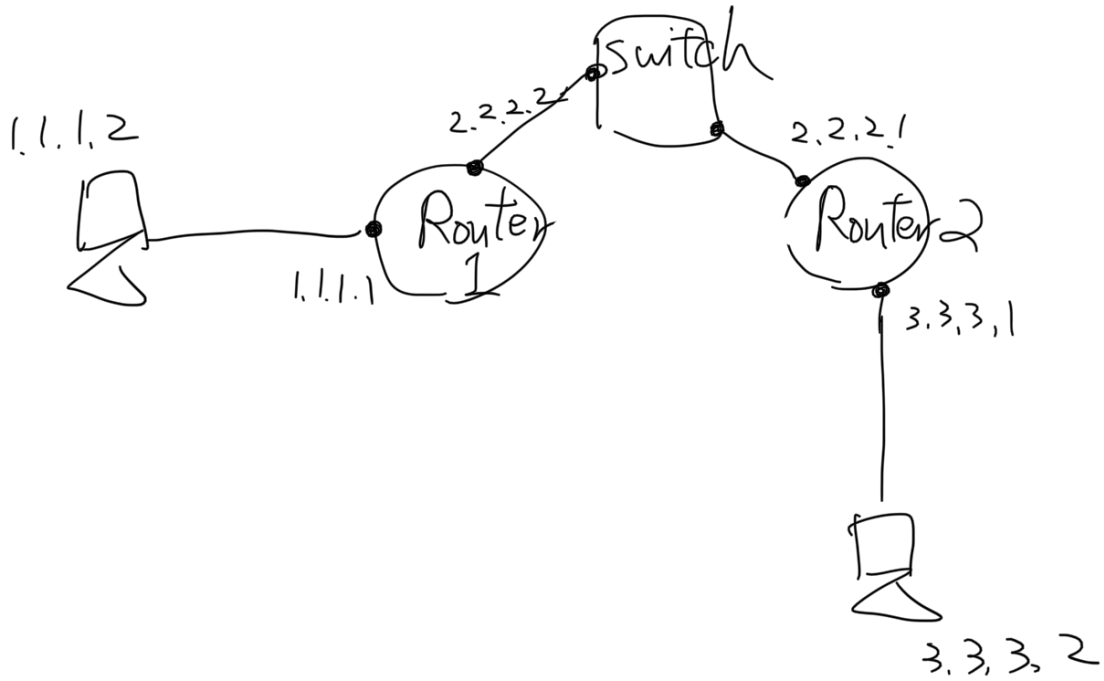
If we further analyze the two subnetworks and the supernetwork, we can confirm that the supernetwork actually covers all the IP addresses in the two subnetworks.

(1) 140.113.1.0/24 → has IP addresses 140.113.1.0 – 140.113.1.255
(2) 140.113.2.0/24 → has IP addresses 140.113.2.0 – 140.113.2.255
(3) 140.113.0.0/22 → has IP addresses 140.113.0.0 – 140.113.3.255

1.2 Routing Table

Routing table tells the router where to forward the packets.

Given a network below.



We assume both Router 1 and 2 have two interfaces connecting to two networks.

Router 1: 1.1.1.0/24 and 2.2.2.0/24

Router 2: 2.2.2.0/24 and 3.3.3.0/24

In order to route packets from 1.1.1.0/24 network to 3.3.3.0/24 network, the two machines and the routers need to set up their network, respectively.

- Machine 1 (address 1.1.1.2): It needs to set up the IP address

1.1.1.2, subnet mask 255.255.255.0, and gateway 1.1.1.1

- Machine 2 (address 3.3.3.2): It needs to set up the IP address 3.3.3.2, subnet mask 255.255.255.0, and gateway 3.3.3.1
- Router 1 (interface FastEthernet 0/0): IP address 1.1.1.1, subnet mask 255.255.255.0
- Router 1 (interface FastEthernet 1/0): IP address 2.2.2.2, subnet mask 255.255.255.0
- Router 1 (ROUTING - Static): Network 3.3.3.0, Mask 255.255.255.0, Next Hop 2.2.2.1
- Router 2 (interface FastEthernet 0/0): IP address 2.2.2.1, subnet mask 255.255.255.0
- Router 2 (interface FastEthernet 1/0): IP address 3.3.3.1, subnet mask 255.255.255.0
- Router 2 (ROUTING - Static): Network 1.1.1.0, Mask 255.255.255.0, Next Hop 2.2.2.2

Please read **Gateway-RoutingTable.pdf** for the meaning of the gateway and the entry in the routing table.

The complete example and setting can be found in **RoutingExample.pkt**

You may check the videos provided in the "Helpful Resources", especially the four videos on routing table.

1.3 TODO

Given the networks in **FinalProject.pkt**. Properly configure all the machines and routers using the supernetting concept.

Expected outcome: You should be able to send the ping messages from any one machine to any other machines.

You may refer to the Midterm Lab Video for partial instructions on how to set up the Packet Tracer.

1.4 Helpful Resources

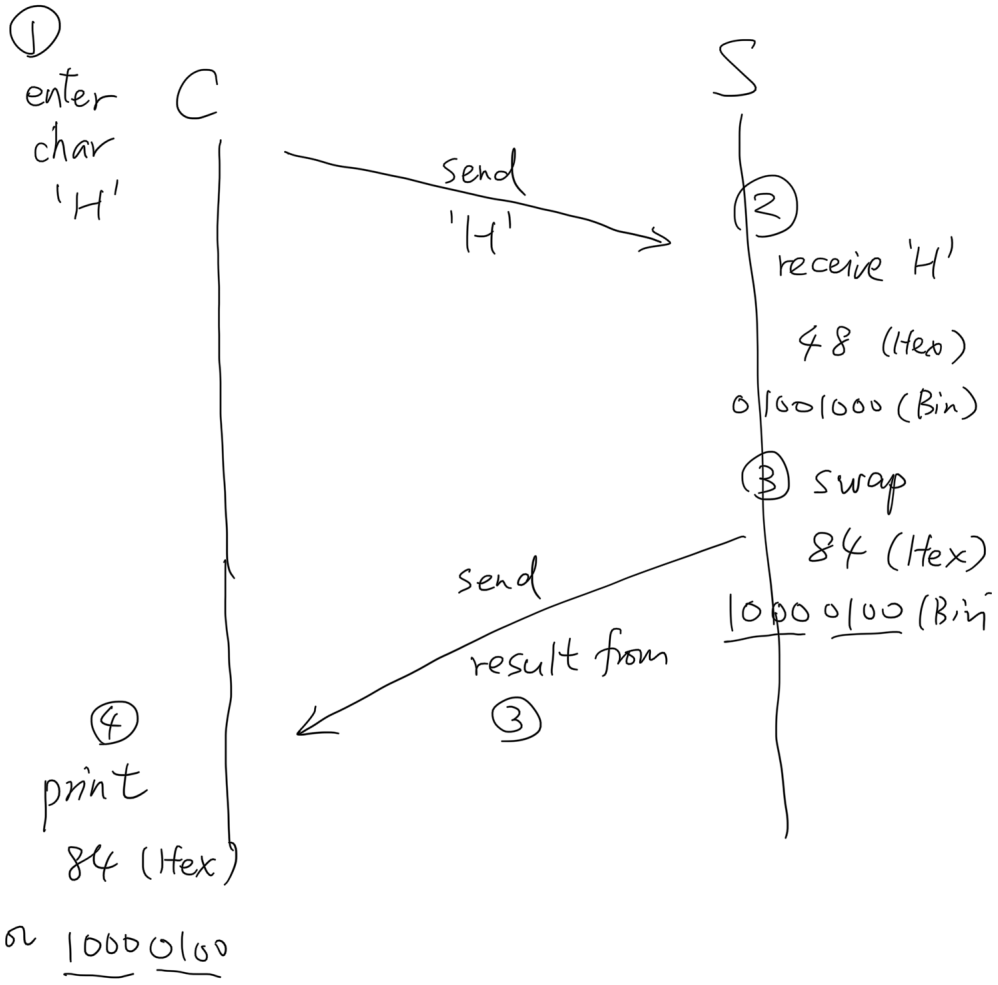
- IP Addressing - Part 1 (Binary Numbering Review)
- IP Addressing - Part 2 (Basic Subnetting)
- IP Addressing - Part 3 (Advanced Subnetting)
- Configuring 3 Router Network with Static Connections
- Routing Tables
- IP Route Table
- Static and Dynamic Routing

2 Socket Programming

We learned fundamental socket programming and checksum algorithm in class. In this project, we develop a simple socket program which will do the following.

1. (Client) Prompt the user to enter a character (assuming that we have 8-bit character in ASCII)
2. (Client) Send this character through either TCP or UDP socket to the server
3. (Server) Once receiving the character, swap the upper and lower 4-bit blocks.
4. (Server) Send the result from the previous step back to the client
5. (Client) Once receiving the packet, print out the result (either in binary or hexadecimal)

2.1 Example



2.2 Discussions

In the checksum() algorithm, we use operations like (value) & 0xFFFF.

We know that this bitwise AND operation will allow us to take only the least significant 16 bits from the value.

For example, if the value is 67890 (Decimal) or 1 0000 1001 0011 0010 (Binary),

```
(value) & 0xFFFF =

  1 [0000 1001 0011 0010]
& 0 1111 1111  1111  1111
======================
=  0 [0000 1001 0011 0010] (result)
```

When you compare the 16 bits in the brackets, they are exactly the same. In other words, (value & 0xFFFF) will get rid of anything leading bits (by making them all 0's) and keep only the right-most 16 bits.

2.3 Approach

- Set up the client and server socket programs (already provided in my slides and GitHub repo)
- Write the swap() function, which will accept a character and swap the leading and trailing 4 bits
- You can modify the 0xFFFF part so you can get individually the leading and trailing 4 bits