

Marwan.aziz@su.edu.krd

Designing Models for Addressing and Naming

Guidelines for Addressing and Naming

- Use a structured model for addressing and naming
- Assign addresses and names hierarchically
- Decide in advance if you will use
 - Central or distributed authority for addressing and naming
 - Public or private addressing
 - Static or dynamic addressing and naming

Advantages of Structured Models for Addressing & Naming

- It makes it easier to
 - Read network maps
 - Operate network management software
 - Recognize devices in protocol analyzer traces
 - Meet goals for usability
 - Design filters on firewalls and routers
 - Implement route summarization

Public IP Addresses

- Managed by the Internet Assigned Numbers Authority ([IANA](#))
- Users are assigned IP addresses by Internet service providers (ISPs).
- ISPs obtain allocations of IP addresses from their appropriate Regional Internet Registry (RIR)

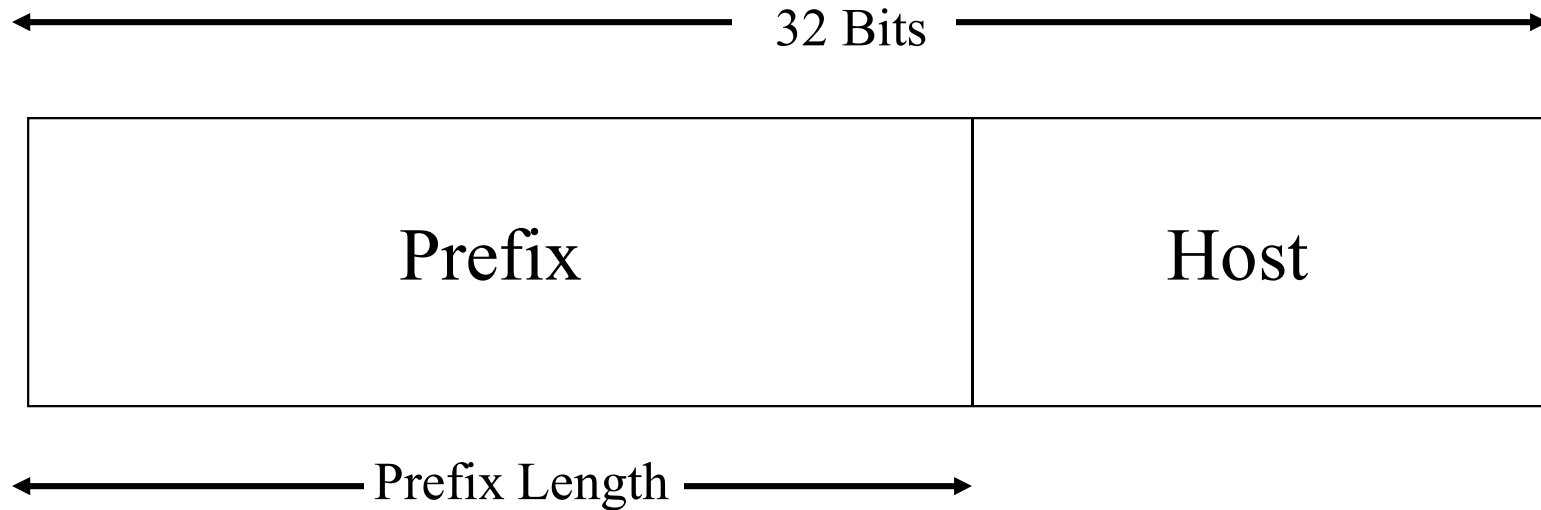
Private Addressing

- 10.0.0.0 – 10.255.255.255
- 172.16.0.0 – 172.31.255.255
- 192.168.0.0 – 192.168.255.255

Criteria for Using Static Vs. Dynamic Addressing

- The number of end systems
- The likelihood of needing to renumber
- The need for high availability
- Security requirements
- The importance of tracking addresses
- Whether end systems need additional information
 - (DHCP can provide more than just an address)

The Two Parts of an IP Address



Prefix Length

- An IP address is accompanied by an indication of the prefix length
 - Subnet mask
 - /Length
- Examples
 - 192.168.10.1 255.255.255.0
 - 192.168.10.1/24

Subnet Mask

- 32 bits long
- Specifies which part of an IP address is the network/subnet field and which part is the host field
 - The network/subnet portion of the mask is all 1s in binary.
 - The host portion of the mask is all 0s in binary.
 - Convert the binary expression back to dotted-decimal notation for entering into configurations.
- Alternative
 - Use slash notation (for example /24)
 - Specifies the number of 1s

Subnet Mask Example

- 11111111 11111111 11111111 00000000
- What is this in slash notation?
- What is this in dotted-decimal notation?

Another Subnet Mask Example

- 11111111 11111111 11110000 00000000
- What is this in slash notation?
- What is this in dotted-decimal notation?

One More Subnet Mask Example

- 11111111 11111111 11111000 00000000
- What is this in slash notation?
- What is this in dotted-decimal notation?

Designing Networks with Subnets

- Determining subnet size
- Computing subnet mask
- Computing IP addresses

Addresses to Avoid When Subnetting

- A node address of all ones (broadcast)
- A node address of all zeros (network)
- A subnet address of all ones (all subnets)
- A subnet address of all zeros (confusing)
 - Cisco IOS configuration permits a subnet address of all zeros with the **ip subnet-zero** command

Practice

- Network is 172.16.0.0
- You want to divide the network into subnets.
- You will allow 600 nodes per subnet.
- What subnet mask should you use?
- What is the address of the first node on the first subnet?
- What address would this node use to send to all devices on its subnet?

More Practice

- Network is 172.16.0.0
- You have eight LANs, each of which will be its own subnet.
- What subnet mask should you use?
- What is the address of the first node on the first subnet?
- What address would this node use to send to all devices on its subnet?

One More

- Network is 192.168.55.0
- You want to divide the network into subnets.
- You will have approximately 25 nodes per subnet.
- What subnet mask should you use?
- What is the address of the last node on the last subnet?
- What address would this node use to send to all devices on its subnet?

IP Address Classes

- Classes are now considered obsolete
- But you have to learn them because
 - Everyone in the industry still talks about them!
 - You may run into a device whose configuration is affected by the classful system

Classful IP Addressing

| Class | First Few Bits | First Byte | Prefix Length | Intent |
|-------|----------------|------------|---------------|---------------------|
| A | 0 | 1-126* | 8 | Very large networks |
| B | 10 | 128-191 | 16 | Large networks |
| C | 110 | 192-223 | 24 | Small networks |
| D | 1110 | 224-239 | NA | IP multicast |
| E | 1111 | 240-255 | NA | Experimental |

*Addresses starting with 127 are reserved for IP traffic local to a host.

Division of the Classful Address Space

| Class | Prefix Length | Number of Addresses per Network |
|-------|---------------|---------------------------------|
| A | 8 | $2^{24}-2 = 16,777,214$ |
| B | 16 | $2^{16}-2 = 65,534$ |
| C | 24 | $2^8-2 = 254$ |

Finding the classes in binary and dotted-decimal notation

| | First byte | Second byte | Third byte | Fourth byte |
|---------|---------------|----------------|---------------|----------------|
| Class A | 0 | | | |
| Class B | 10 | | | |
| Class C | 110 | | | |
| Class D | 1110 | | | |
| Class E | 1111 | | | |

a. Binary notation

| | First byte | Second byte | Third byte | Fourth byte |
|---------|---------------|----------------|---------------|----------------|
| Class A | 0–127 | | | |
| Class B | 128–191 | | | |
| Class C | 192–223 | | | |
| Class D | 224–239 | | | |
| Class E | 240–255 | | | |

b. Dotted-decimal notation

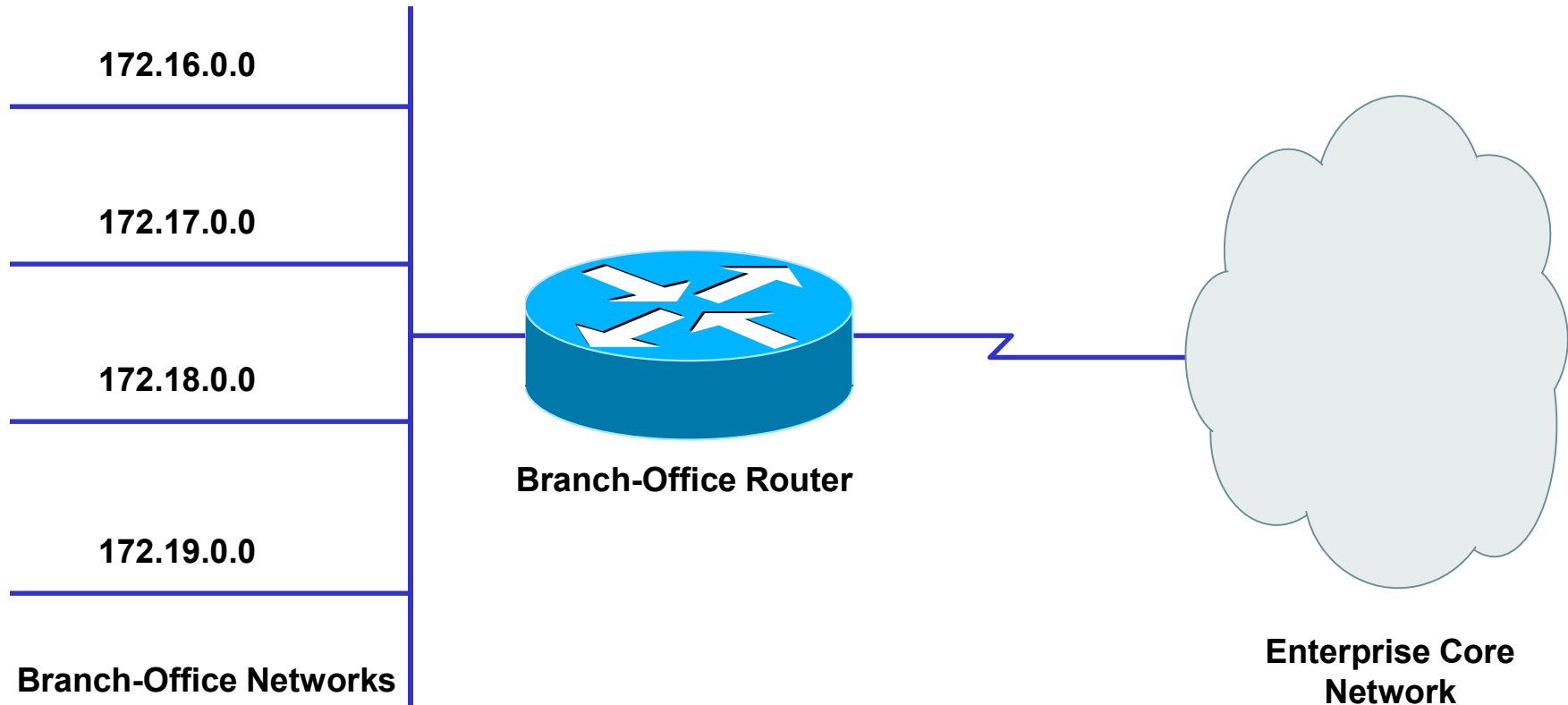
Classful IP is Wasteful

- Class A uses 50% of address space
- Class B uses 25% of address space
- Class C uses 12.5% of address space
- Class D and E use 12.5% of address space

Classless Addressing

- Prefix/host boundary can be anywhere
- Less wasteful
- Supports route summarization
 - Also known as
 - Aggregation
 - Supernetting
 - Classless routing
 - Classless inter-domain routing (CIDR)
 - Prefix routing

Supernetting



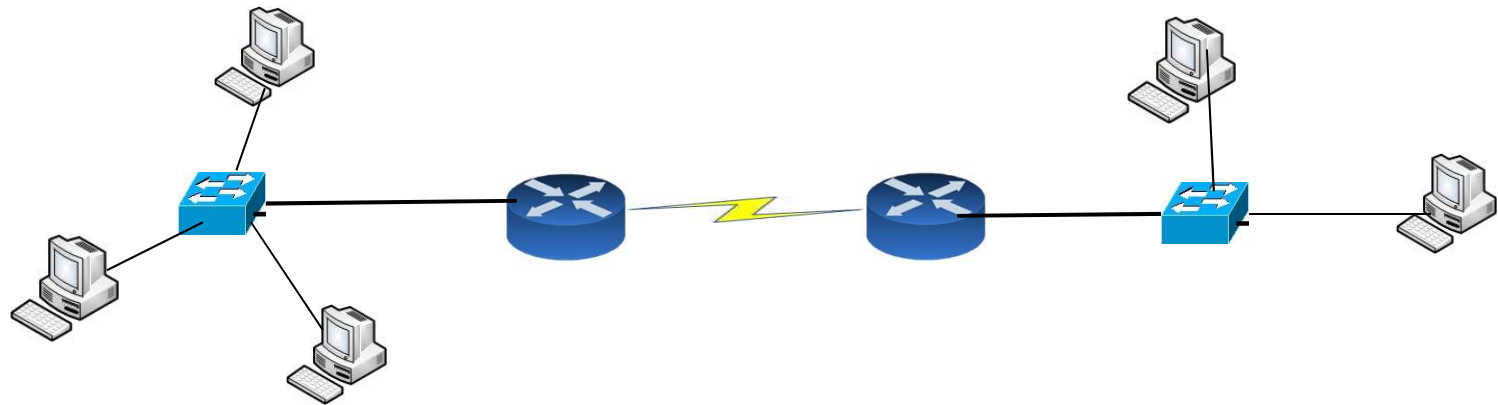
- Move prefix boundary to the left
- Branch office advertises 172.16.0.0/14

172.16.0.0/14 Summarization

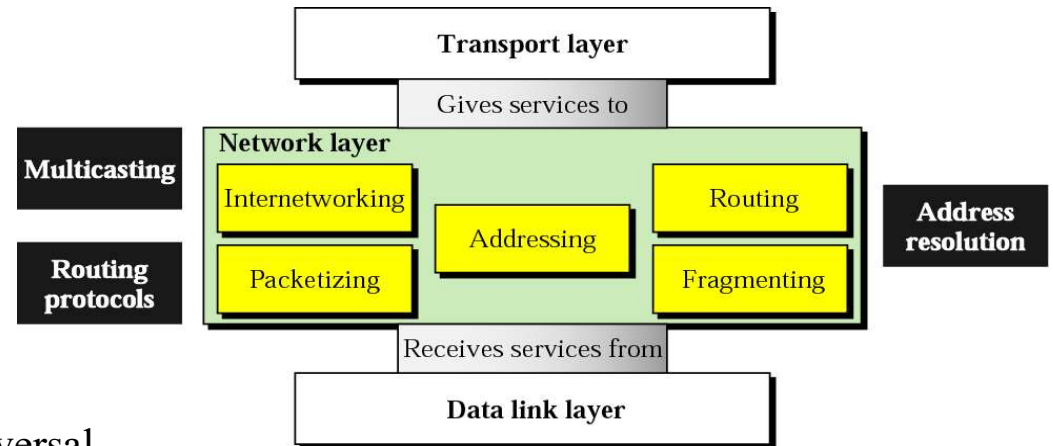
| Second Octet in Decimal | Second Octet in Binary |
|--------------------------------|-------------------------------|
| 16 | 00010000 |
| 17 | 00010001 |
| 18 | 00010010 |
| 19 | 00010011 |

What is Routing

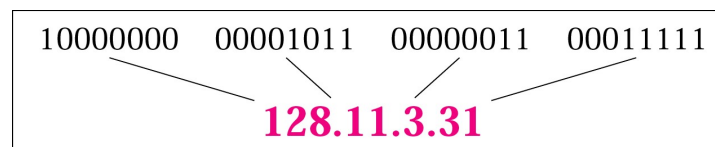
- ❑ The process of moving packets between IP-Based Networks.



IPv4 Addresses



- An IP address is a 32-bits long
- The IP addresses are unique and universal
- The address space of IPv4 is 2^{32} or 4,294,967,296
- Binary notation: 01110101 10010101 00011101 00000010
- Dotted-decimal notation: 117.149.29.2
- Four Octet (Byte) Address
- Can be one of Three Different Classes
- When Combined with a Subnet Mask, Defines a network and Host portion
- Operates on Layer three of OSI Model



Example

- Change the following IP addresses from binary notation to dotted-decimal notation.
 - a. 10000001 00001011 00001011 11101111
 - b. 11111001 10011011 11111011 00001111
- We replace each group of 8 bits with its equivalent decimal number and add dots for separation:
 - a. 129.11.11.239
 - b. 249.155.251.15

Simple way to Work with Decimal & Binary

○ Decimal : 255 = 128 64 32 16 8 4 2 1
Binary : 1 1 1 1 1 1 1 1

○ Decimal : 210 = 128 64 32 16 8 4 2 1
Binary :

○ Decimal : 120 = 128 64 32 16 8 4 2 1
Binary :

○ Decimal 180 Binary ?

○ Decimal 41 Binary ?

○ Binary 00110110 Decimal ?

○ Binary 10010110 Decimal ?

Subnetting

- Divide a large block of addresses into several contiguous groups and assign each group to smaller networks called subnets
- Increase the number of 1s in the mask

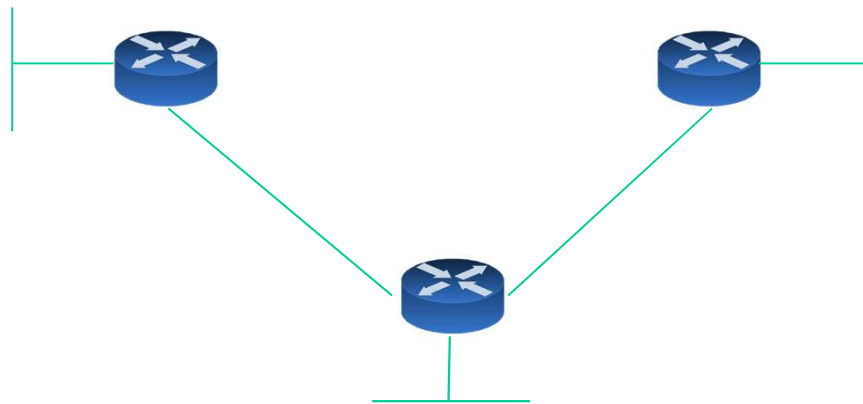
There are three styles of subnetting examples

- 1) Subnetting when given a required number of networks
- 2) Subnetting when given a required number of clients
- 3) Given an IP address & Subnet Mask, finding original network range (reverse engineering a subnet problem)

- 1) Subnetting when given a required number of networks

Example 1:

A service provider has given you the Class C network range 216.21.5.0 , Your company must break the network into 5 separate subnets.



Example 1: 216.21.5.0 5 Subnets

Step 1) Determine the number of subnets and convert to binary

$$\begin{array}{cccccccc} & 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ 5 = & 0 & 0 & 0 & 0 & 0 & \underline{1} & \underline{0} & \underline{1} \end{array}$$

Step 2) Reserve required bits in subnet mask and find the incremental value

subnet mask **255.255.255.0** = 11111111.11111111.11111111.00000000

-The binary value of 5 subnets tell us that we need at least 3 network bits to satisfy this requirement

$$5 = 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \underline{1} \quad \underline{0} \quad \underline{1} \quad \mathbf{3 \text{ Bits}}$$

Then convert 3 of client bits (0) to network bits (1)

$$\begin{array}{ccccccc} 11111111 & . & 11111111 & . & 11111111 & . & \underline{111}00000 \\ = & & 255 & & 255 & & 255 & & 224 \end{array}$$




The subnet mask of each IP in our divided network is **255.255.255.224**

Example 1: 216.21.5.0 5 Subnets

Step 3) Use increment to find network ranges

11111111 . 11111111 . 11111111 . 11(1)00000

 128 64 32

- Start with your given network address and add your increment to the subnetted octet: then fill in your end ranges, which is the last possible IP address before you start the next range

216.21.5.0 - 216.21.5.31

216.21.5.32 - 216.21.5.63

216.21.5.64 - 216.21.5.95

216.21.5.96 - 216.21.5.127

216.21.5.128 - 216.21.5.159

Note: Remember the first and last address from each range (network / broadcast IP) are unusable

Example 2: Your Company would like to break the Class B private IP address range 172.16.0.0 into 60 different subnets

Step 1) Determine the number of subnets and convert to binary

$$60 = 00 \underline{111100} \quad \mathbf{6 \text{ Bits}}$$

Step 2) Reserve required bits in subnet mask and find incremental value

The binary value of 60 subnets tells us that we need at least 6 network bits to satisfy this requirement (since you cannot get the number 60 with any less than 6 bits – 111100)

original subnet mask is 255.255.0.0 (Class B subnet)
11111111.11111111.00000000.00000000



11111111.11111111.11111100.00000000 = 255.255.252.0

Example 2: 172.16.0.0 60 Subnets

Step 3) Use increment to find network ranges

11111111 . 11111111 . 11111(1)00 . 00000000



4

- Start with your given network address and add your increment to the subnetted octet: then fill in your end ranges, which is the last possible IP address before you start the next range

172.16.0.0 - 172.16.3.255

172.16.4.0 - 172.16.7.255

172.16.8.0 - 172.16.11.255

172.16.12.0 - 172.16.15.255

172.16.16.0 - 172.16.19.255

.... etc

Solve these Problems

- **200.1.1.0, Class (C) break into 40 networks**
- **199.9.10.0, Class (C) break into 14 networks**
- **170.50.0.0, Class (B) break into 1000 networks**
- **12.0.0.0, Class (A) break into 25 networks**

2) Subnetting when given a required number of (hosts) clients

Example 1: A service provider has given you the
Class C
network range 209.50.1.0. Your company must
break the
network into as many subnets as possible as long as
there are *at least 50 clients per network*.

Step 1) Determine the number of clients and convert to binary

In this example, the binary representation of 50 = 00110010

Style 2) Example 1: 209.50.1.0 50 Clients

Step 2) Reserve required bits in subnet mask and find incremental value

The binary value of 50 clients tells us that we need at least 6 client bits to satisfy this requirement (since you cannot get the number 50 with any less than 6 bits – 110010) subnet mask is 255.255.255.0 (Class C subnet)

11111111.11111111.11111111.00000000

We must ensure 6 of the client bits (0) remain client bits (save the clients!) in order to satisfy the requirements. All other bits can become network bits:

New Mask = 11111111.11111111.11111111.11000000

note the 6 client bits that we have saved

convert the mask back to decimal 255.255.255.192

Style 2) Example 1: 209.50.1.0 50 Clients

Our increment bit is the last possible network bit, converted back to a binary number:

New Mask = 11111111.11111111.11111111.1(1)000000 bit with the parenthesis is your increment bit. If you convert this bit to a decimal number, it becomes the number 64

Step 3) Use increment to find network ranges

Start with your given network address and add your increment to the subnetted octet:

209.50.1.0 – 209.50.1.63

209.50.1.64 – 209.50.1.127

209.50.1.128 – 209.50.1.191

209.50.1.192 – 209.50.1.255

Example 2: Your Company would like to break the Class B private IP address range 172.16.0.0 into as many subnets as possible, provided that they can get at least 300 clients per subnet

Step 1 :

binary representation of 300 = 100101100

Step 2:

value of 300 clients tells us that we need at least 9 client

255.255.0.0 = 11111111.11111111.00000000.00000000

We must ensure 9 of the client bits (0) remain client bits (save the clients!)

New Mask = 11111111.11111111.11111111 0.00000000

255.255.254.0

Example 2: 172.16.0.0 300 clients

Step 3:

New Mask = 11111111.11111111.111111(1)0.0000000 – bit with the parenthesis is your increment bit. If you convert this bit to a decimal number, it becomes the number 2

Start with your given network address and add your increment to the subnetted octet:

172.16.0.0 – 172.16.1.255

172.16.2.0 – 172.16.3.255

172.16.4.0 – 172.16.5.255

etc...

Solve these Problems

- **200.1.1.0, Class (C) break into networks of 40 hosts each**
- **199.9.10.0, Class (C) break into networks of 12 hosts each**
- **170.50.0.0, Class (B) break into networks of 1000 hosts**
- **12.0.0.0, Class (A) break into networks of 100 hosts**

- 3) Subnetting when given an IP address & Subnet mask,
then find the original range (Reverse Engineering)

Example 1:

You have a host which its IP address
192.168.1.124 and its subnet 255.255.255.224,
identify the original range of (the subnet)
addresses that this IP address belong to.

When reverse engineering a problem, all you need to do is break the
subnet mask back into binary and find the increment that was used.

$$255.255.255.224 = 11111111.11111111.11111111.11100000$$

3) Reverse Engineering

Example 1:

- As before, the last possible network bit is your increment. In this case, the increment is 32
- Use this increment to find the network ranges until you pass the given IP address:

192.168.1.0 – 192.168.1.31

192.168.1.32 – 192.168.1.63

192.168.1.64- 192.168.95

192.168.1.96 – 192.168.1.127

192.168.1.128

(Reverse Engineering)

Example 2:

You have a host which its IP address 192.168.1.58 and its subnet 255.255.255.240, identify the original range of (the subnet) addresses that this IP address belong to.

When reverse engineering a problem, all you need to do is break the subnet mask back into binary and find the increment that was used.

$$255.255.255.224 = 11111111.11111111.11111111.11110000$$

In this case, the increment is 16

(Reverse Engineering)

Example 2:

- Using this increment (16) to find the network ranges until you pass the given IP address:

192.168.1.0 - 192.168.1.15

192.168.1.16 – 192.168.1.31

192.168.1.32 – 192.168.1.47

192.168.1.48 – 192.168.1.63 (*192.168.1.58 belongs to this range*)

Tip: what if its given a GW for that host 192.168.1.34 ?

Exception

- Example, if I were asked to break a network into 8 subnets, you would assume it would take four bits since 8 in binary is:
0 0 0 0 1 0 0 0
- However, you can achieve this requirement with only three bits since 0-7 is really 8 numbers (0, 1, 2, 3, 4, 5, 6, 7). If you work out the subnetting problem by reserving only three bits, you will get exactly 8 subnets.
- The same thing happens when breaking into networks for every “precise” binary number: **2, 4, 8, 16, 32, 64, 128.**
- So how do you avoid this issue? You can always subtract 1 from the number of networks required. For example, If you are asked to break a range into 16 subnets, figure it out for 15...and so on.

Exception

- There is a similar rule for finding the number of bits to reserve for hosts.
- For example, if we were asked to break a network into subnets that can hold up to 31 hosts, you would assume it would take 5 bits since 31 in binary is: 0 0 0 1 1 1 1.
- However, when you work out the problem you will find that you only get 30 hosts per subnet (1 IP address short). The same thing happens with every “full” binary number: **3, 7, 15, 31, 63, 127**. So how do you avoid this issue? You can always add 1 to the number of hosts required.
- For example, If you are asked to break a range into 63 subnets, figure it out for 64...and so on.

Exception

- Exception gets somewhat technical... so to summarize:
- When subnetting based on the number of networks, SUBTRACT 1 from the number
- When subnetting based on the number of hosts per network, ADD 1 to the number