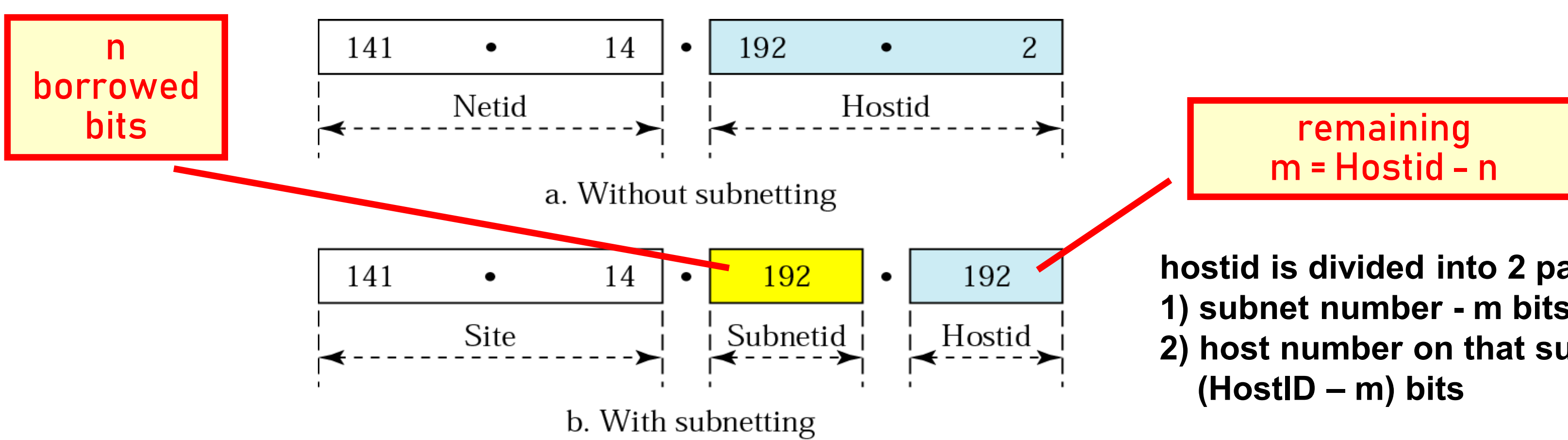
# Subnetting (cont.)

**Subnetted Network** – network divided into several smaller subnetworks each having its own (sub)network address

- internally, each subnetwork is recognized by its subnetwork address; <u>to the rest of the Internet all subnetworks still appear as a single network</u>

**Organization of Address Space in a Subnetted Network**

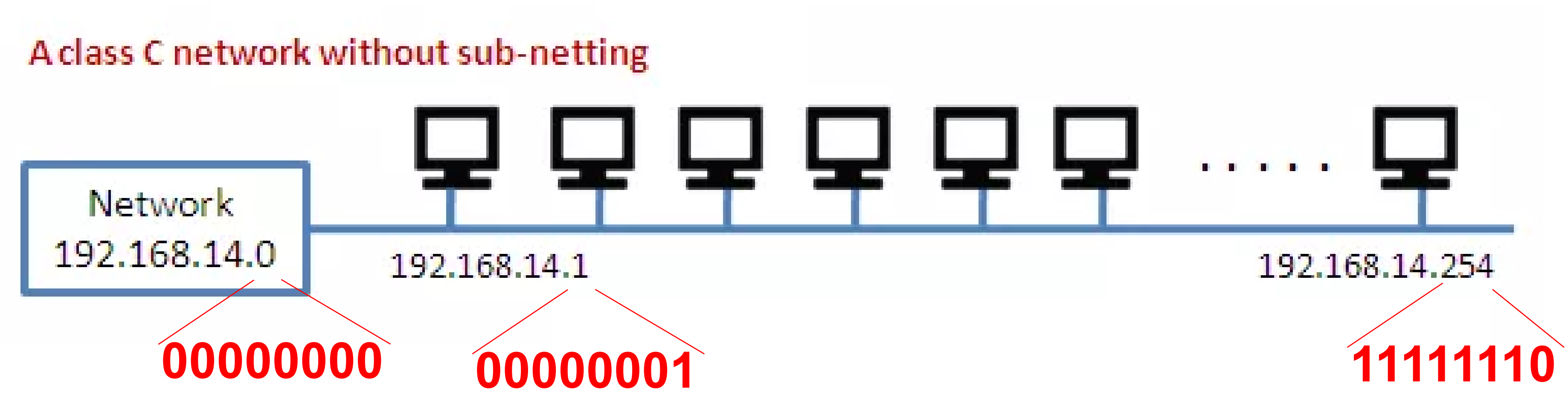
- a number of HostID bits are borrowed (used) for subnet identification
- with n borrowed bits, $2^n$ subnets can be created
- <u>number of hosts in each subnet: $2^{Hostid - n}$</u>




hostid is divided into 2 parts:
1) subnet number - m bits
2) host number on that subnet - (HostID – m) bits

# Subnetting (cont.)

**Example** **[ Subnetting example ]**

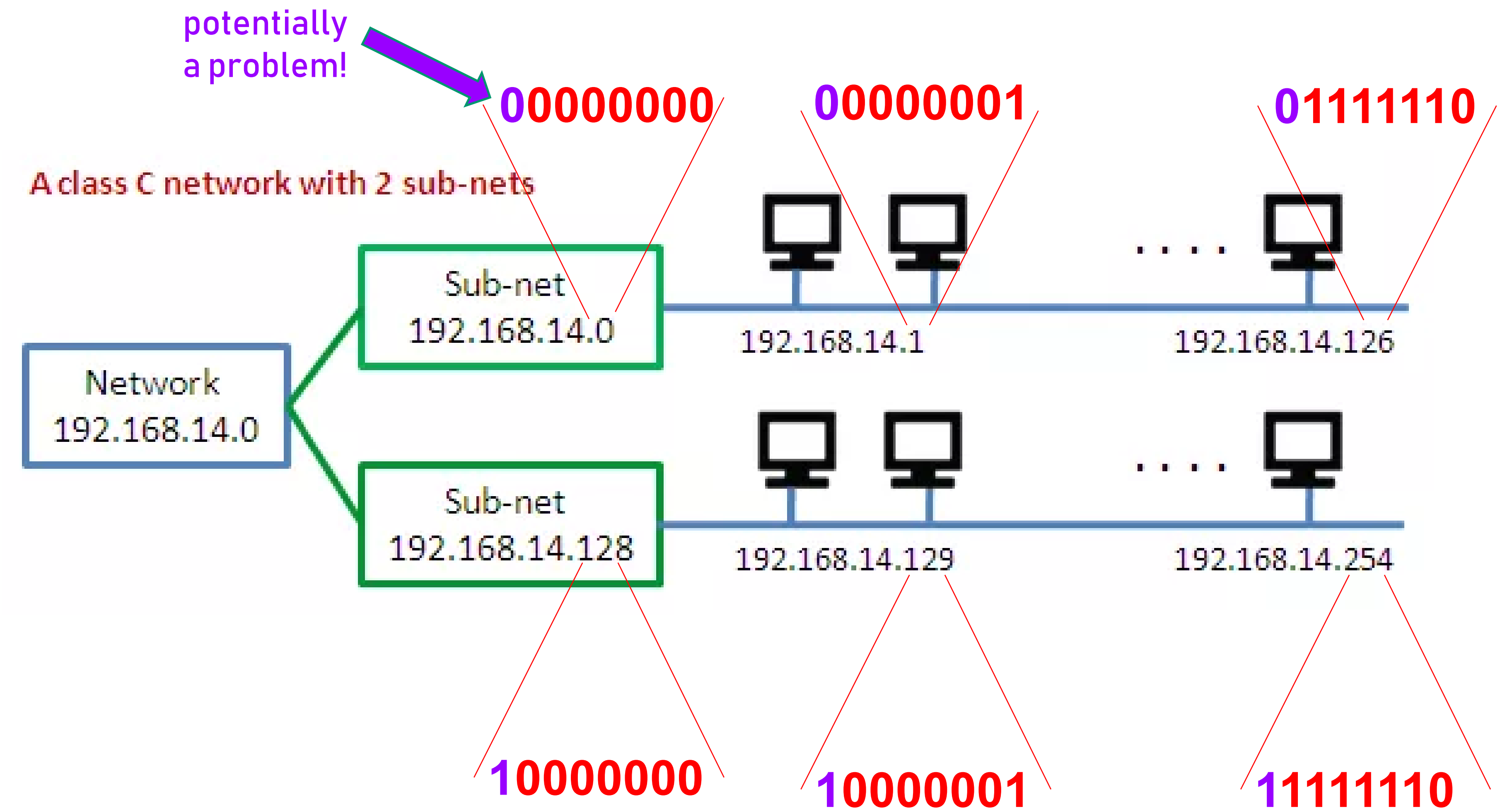



**Assume we want to split this network into 2 equal size segments.**

# Subnetting (cont.)

**Example** **[ Subnetting example (cont.) ]**

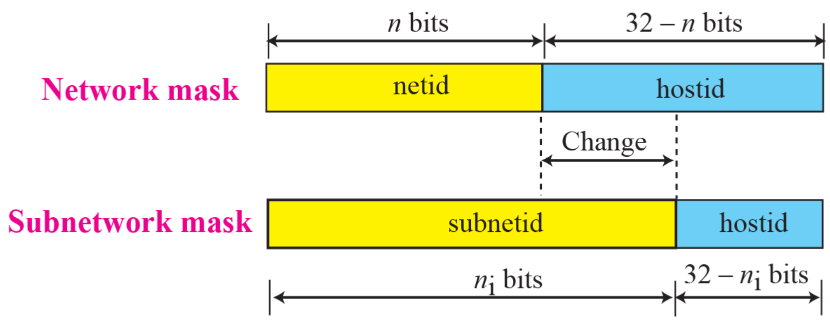
**What will be the 1st and what the last address in each subnetwork ??**

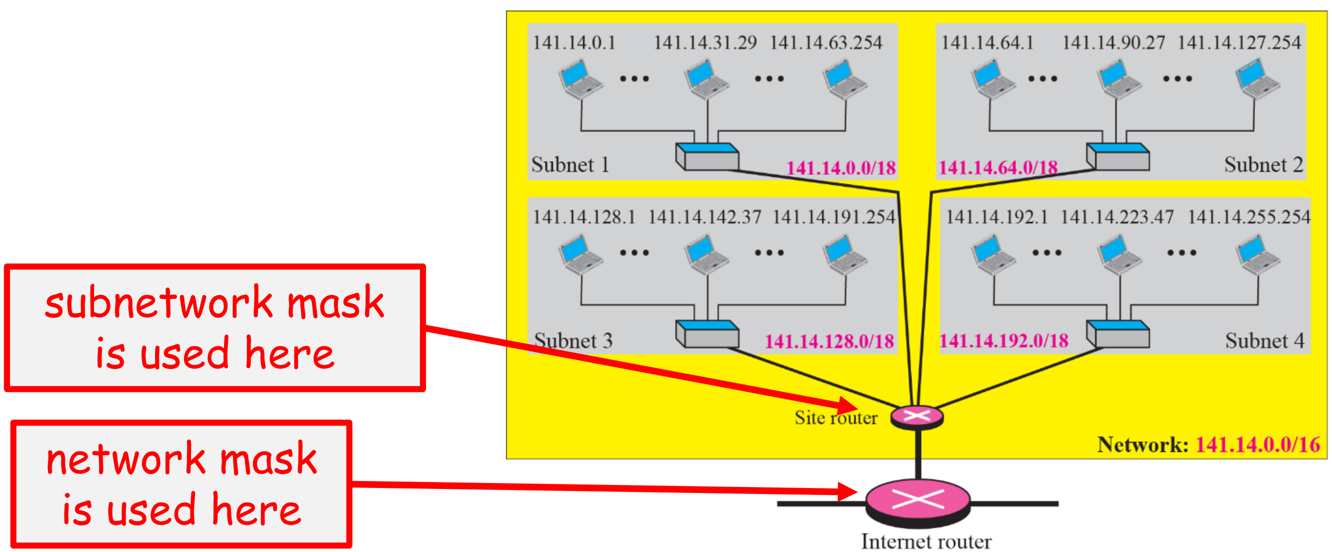


# Subnetting (cont.)

**Subnet Mask** – **32-bit binary number, used by <u>inside routers</u> in a subnetted network – helps extract the subnet address when AND-ed with an IP address from the block**

- **# of 1-s in subnet mask > # of 1-s in default mask**



# Subnetting (cont.)

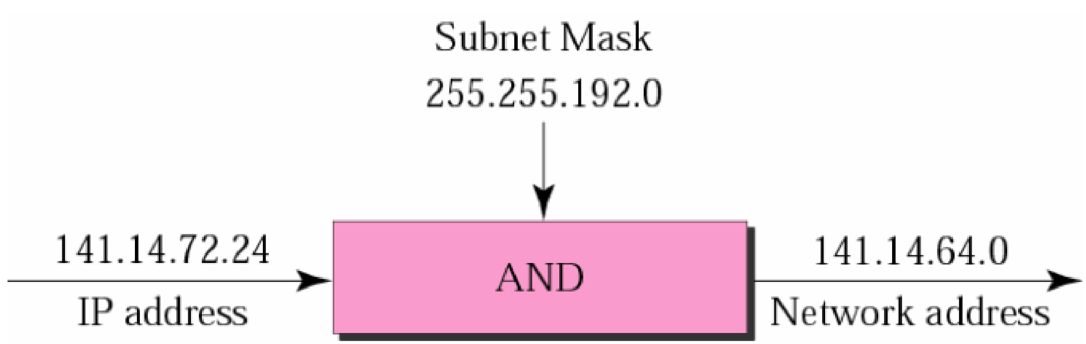
**Example** **[ subnet mask ]**

**What is the subnetwork address if the destination address is 141.14.72.24 and the subnet mask is 255.255.192.0 ?**

**Solution:**

| | | | | |
|---|---|---|---|---|
| **Address 1** | 10001101 | 00001110 | 01001000 | 00011000 |
| **Subnet Mask** | 11111111 | 11111111 | 11000000 | 00000000 |
| **Subnet** | 10001101 | 00001110 | 01000000 | 00000000 |

**The subnetwork address is 141.14.64.0.**

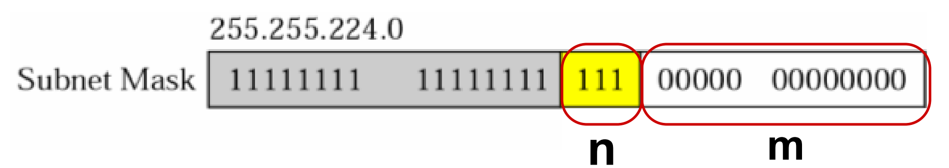


# Subnetting (cont.)

## Subnet Design Considerations

**(how to chose n?)**

- **How many subnets does the organization need today & how many it will need in the future?** (should be < $2^n$)
- **How many hosts are there on the organization's largest subnet today; how many there will be in the future?** (should be <$2^m$)




## Subnet Design Procedure

**(1) take the maximum number of subnets required and round up to the nearest <u>higher</u> power of two**

- **e.g., if a organization needs 9 subnets, the network administr. will have to round up to $2^4$=16**
- **if 9 subnets are required today, but 8 more will have to be added in two years, it might be wise to allow for more growth and select $2^5$ (32) as the max number of subnets**

**(2) make sure that there are enough host addresses for the organization's largest subnet**

- **if the largest subnet needs to support 50 host addresses today, $2^5$ (32) will not provide enough host address space so the network administrator will have to round up to $2^6$ (64)**

# Subnetting (cont.)

## Example [ subnetting ]

**A customer has been given IP add. block 128.100.0.0/16 (a Class B add. block) for his company. He requires 3 separate networks with the maximum possible number of host connections on each network. How should he subnet his network?**

**Solution:**

**The first two octets 128.100 are fixed due to '/16' mask (i.e., a Class B block). Therefore we have the last two octets to play with.**

### Possibility 1

**Let us just use the first 2 bits for a subnet IDs:**

| Octet 1 | Octet 2 | Octet 3 | Octet 4 |
|---|---|---|---|
| 10000000 | 01100100 | 00000000 | 00000000 |
| 128. | 100. | 0. | 0 |

**The possible combinations for these two bits are:**

| | | | | |
|---|---|---|---|---|
| 00 = 0 | -> | subnet IP: 128.100.0.0, | broadcast IP: 128.100.63.255 | |
| 01 = 64 | -> | subnet IP: 128.100.64.0, | broadcast IP: 128.100.127.255 | |
| 10 = 128 | -> | subnet IP: 128.100.128.0, | broadcast IP: 128.100.191.255 | |
| 11 = 192 | -> | subnet IP: 128.100.192.0, | broadcast IP: 128.100.255.255 | |

potentially a problem!

# Subnetting (cont.)

**However all 1's and all 0's cannot be used as subnet IDs! (RFC 950 rules out '11' and '00' as useable subnet IDs, since these combined with all 1-s and all 0-s hostid are special addresses.)**
**Hence, we would be left with only two subnets instead of the 3 we required.**

## Possibility 2

**Let us use an extra bit in octet 3 for subnet IDs.**

| Octet 1 | Octet 2 | Octet 3 | Octet 4 |
|---|---|---|---|
| 10000000 | 01100100 | 00000000 | 00000000 |
| 128. | 100. | 0. | 0 |

**The possible combinations for these three bits are:**

| | |
|---|---|
| ~~000 = 0~~ | ~~-> 128.100.0.0~~ |
| 001 = 32 | -> 128.100.32.0 |
| 010 = 64 | -> 128.100.64.0 |
| 011 = 96 | -> 128.100.96.0 |
| 100 = 128 | -> 128.100.128.0 |
| 101 = 160 | -> 128.100.160.0 |
| 110 = 192 | -> 128.100.192.0 |
| ~~111 = 224~~ | ~~-> 128.100.224.0~~ |

# Subnetting (cont.)

**As before all 1's and all 0's are not permitted for subnets, therefore we are left with 6 possible subnets ($2^3$ - 2):**

| | |
|---|---|
| 001 = 32 | -> 128.100.32.0 |
| 010 = 64 | -> 128.100.64.0 |
| 011 = 96 | -> 128.100.96.0 |
| 100 = 128 | -> 128.100.128.0 |
| 101 = 160 | -> 128.100.160.0 |
| 110 = 192 | -> 128.100.192.0 |

**This leaves the rest of the bits (from power 13 downwards) in octet 3 and all the bits in octet 4 to construct the individual host addresses.**

**An example of a host address in subnet 128.100.192.0 would be: 128.100.194.23. On first inspection it would appear that address 128.100.194.23 has nothing to do with the subnet 128.100.192.0. However, by looking a little more closely at the final two octets of the host address, we can see that this host is indeed a part of the given subnet.**

| Octet 1 | Octet 2 | Octet 3 | Octet 4 | |
|---|---|---|---|---|
| 10000000<br>128. | 01100100<br>100. | 11100000<br>192. | 00000000<br>0 | **Subnet 6** |
| 10000000<br>128. | 01100100<br>100. | 11100010<br>194. | 00010111<br>23 | **Host on Subnet 6** |

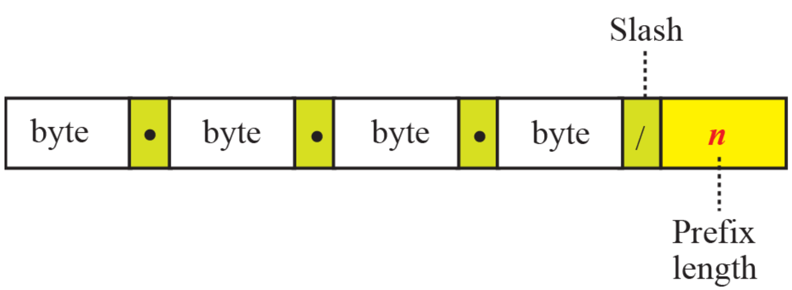
# Classless Addressing

**Classless Addressing (introduced in 1996)** – known as CIDR "Classless InterDomain Routing" addressing – removes class privileges to compensate for address depletion

- with CIDR, ISPs can subdivide their big address space & give blocks of varying sizes to their customers (2, 4, 8, 16, etc.)
- in classful addressing, all created blocks were of the same size ($2^{24}$, $2^{16}$, or $2^{8}$ hosts)

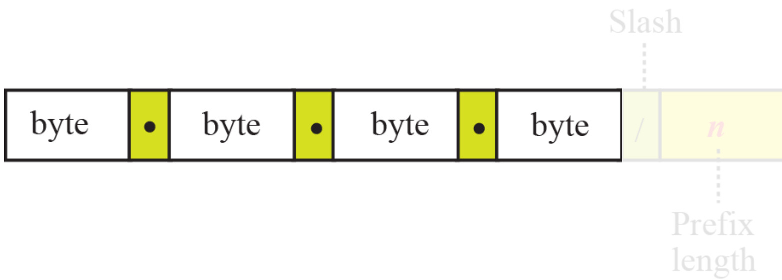
**CIDR Address Format** – in classless addressing, the division between Netid (prefix) and Hostid (sufix) can occur at any arbitrary bit/location

- 'slash' address format: a.b.c.d/x, where x is # of bits in network portion of address and/or # of 1's in the network mask (aka prefix length)

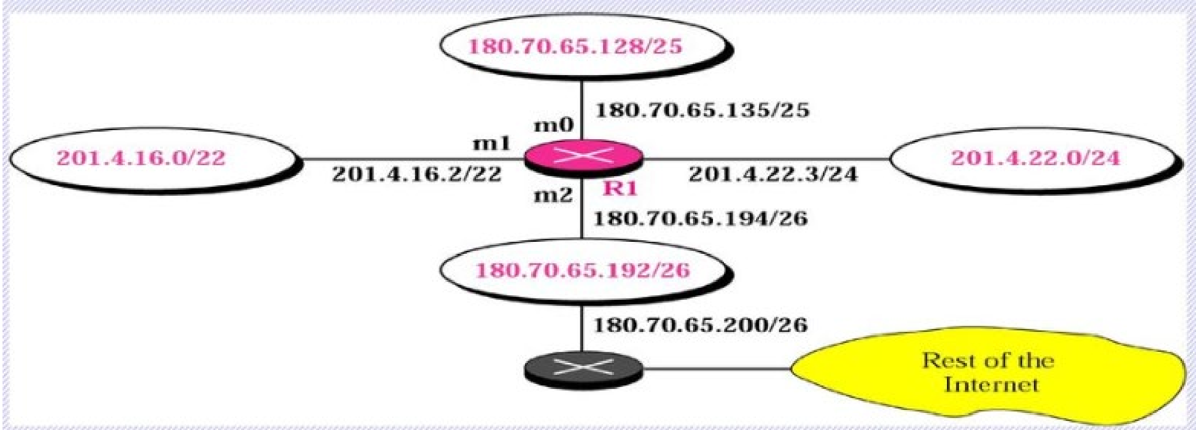


# Classless Addressing (cont.)

## How Does CIDR Work?




**<u>CIDR PROBLEM</u>: Routers can no longer rely on the first few bits to determine the size of prefix.**

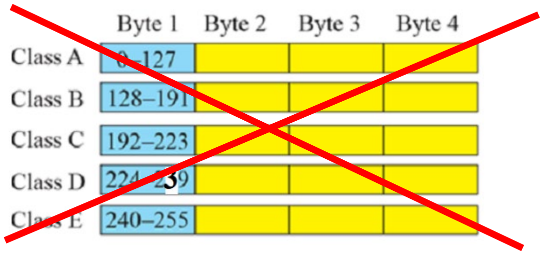



| *Mask* | *Network Address* | *Next Hop* | *Interface* |
|---|---|---|---|
| /26 | 180.70.65.192 | - | m2 |
| /25 | 180.70.65.128 | - | m0 |
| /24 | 201.4.22.0 | - | m3 |
| /22 | 201.4.16.0 | .... | m1 |
| Default | Default | 180.70.65.200 | m2 |

**Routers now rely on (i.e., need to receive) <u>network advertisements</u> containing their slash address format.**

# Classless Addressing (cont.)

**Blocks in Classless Addresses** – **in classless addressing, we need to know one of the addresses in the block <u>and the prefix length</u> in order to define/identify the respective block/network**




## Example [ classless addressing ]

**In classless addressing, an address itself cannot define the block the address belongs to. For example, the address 230.8.24.56 can belong to many blocks of addresses …**

how big is the respective network???

| | |
|---|---|
| **230.8.24.56** | **11100110 00001000 00011000 00111000** |

**Potential blocks of addresses:**

| | | | | |
|---|---|---|---|---|
| **230.8.24.56 / 16** | **Block:** | **230.8.0.0** | **to** | **230.8.255.255** |
| **230.8.24.56 / 20** | **Block:** | **230.8.16.0** | **to** | **230.8.31.255** |
| **230.8.24.56 / 26** | **Block:** | **230.8.24.0** | **to** | **230.8.24.63** |

# Classless Addressing (cont.)

**Example** **[ network address in classless addressing ]**

**What is the network address if one of its addresses is: 167.199.170.82/27.**

**Solution:**

**Prefix length 27 ⇒ keep first 27 bits and change the remaining 5 bits to 0s.**

**The last byte is 01010010.**

**By changing last 5 bits to 0s, we get 01000000 = 64.**

**So, network address = 167.199.170.64/27.**

**Example** **[ network address in classless addressing ]**

**What is the network address if one of its addresses is: 202.78.5.34/22.**

**Solution:**

**Prefix length 22 ⇒ keep first 22 bits and change the remaining 10 bits to 0s.**

**The last two bytes are 00000101 00100000.**

**By changing last 10 bits to 0s, we get 00000100 00000000 = 4.0**

**So, network address = 202.78.4.0/22 .**

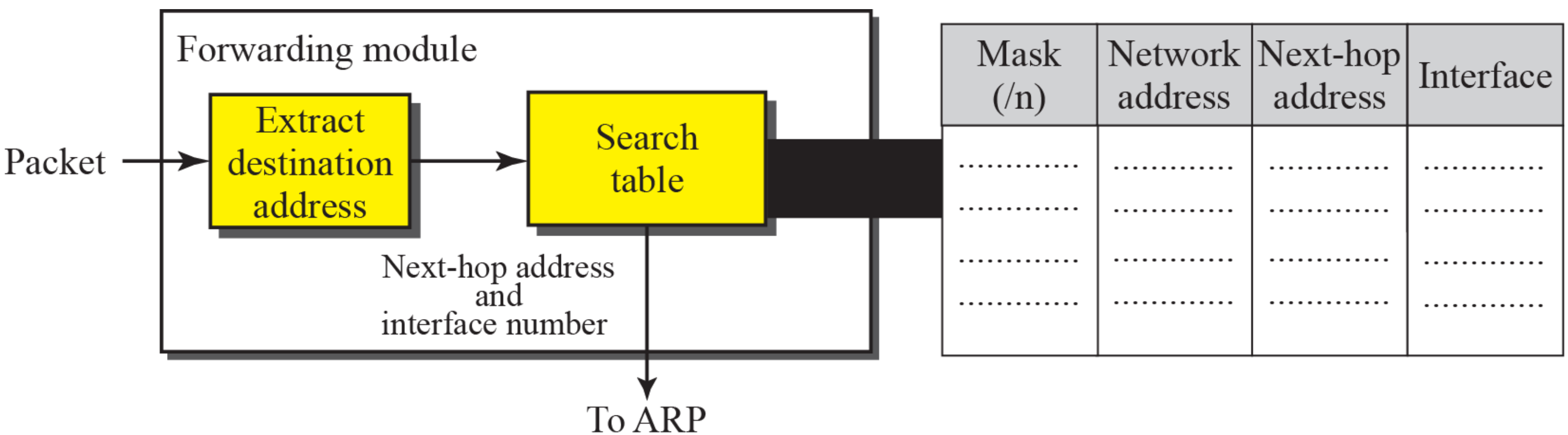
# Classless Addressing (cont.)

**CIDR Routing** – unfortunately, in CIDR addressing, the destination address gives no clue about the respective network address

- a packet does not carry the destination IP address in the slash notation ☹
- network address extraction must be done at the same time as table searching:

step 1) apply a mask /n to the destination address - with progressively decreasing value n!!!

step 2) check if the resultant network address is in the table

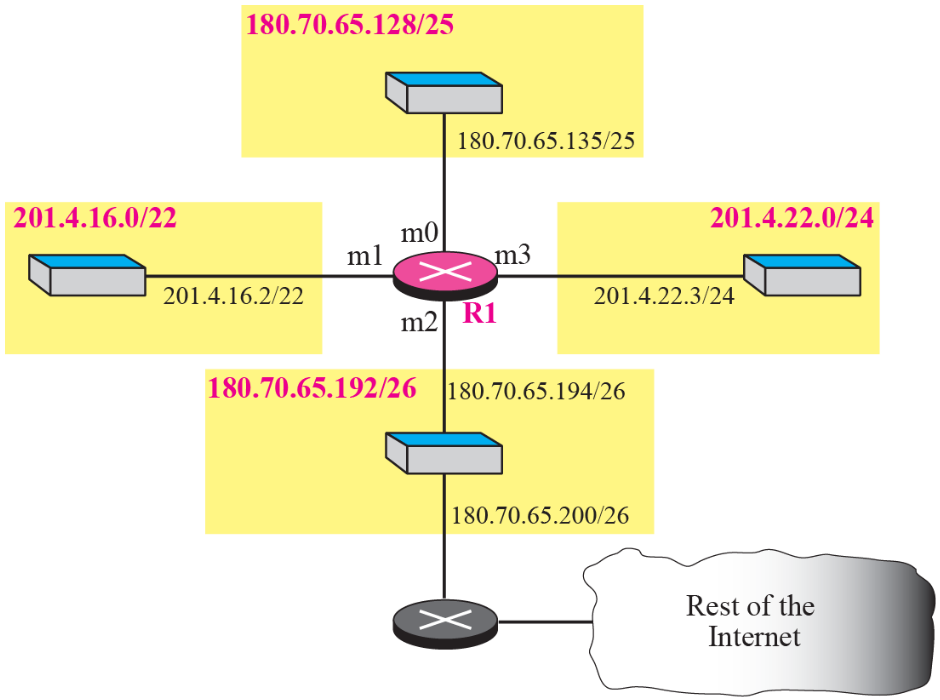



From "TCP/IP Protocol Suite" by B. Forouzan, 4/e, pp. 169

# Classless Addressing (cont.)

**Example** **[ classless routing ]**

**Explain the forwarding process if a packet arrives at R1 with the destination address 180.7.65.140.**




| *Mask* | *Network Address* | *Next Hop* | *Interface* |
|---|---|---|---|
| /26 | 180.70.65.192 | - | m2 |
| /25 | 180.70.65.128 | - | m0 |
| /24 | 201.4.22.0 | - | m3 |
| /22 | 201.4.16.0 | .... | m1 |
| Default | Default | 180.70.65.200 | m2 |

**Steps:**

1) **Apply mask /26 to destination address. Result: 180.7.65.128. Not valid (not in tables).**

2) **Apply mask /25 to destination address. Result: 180.7.65.128. Valid (in tables)! Outgoing port: m0.**

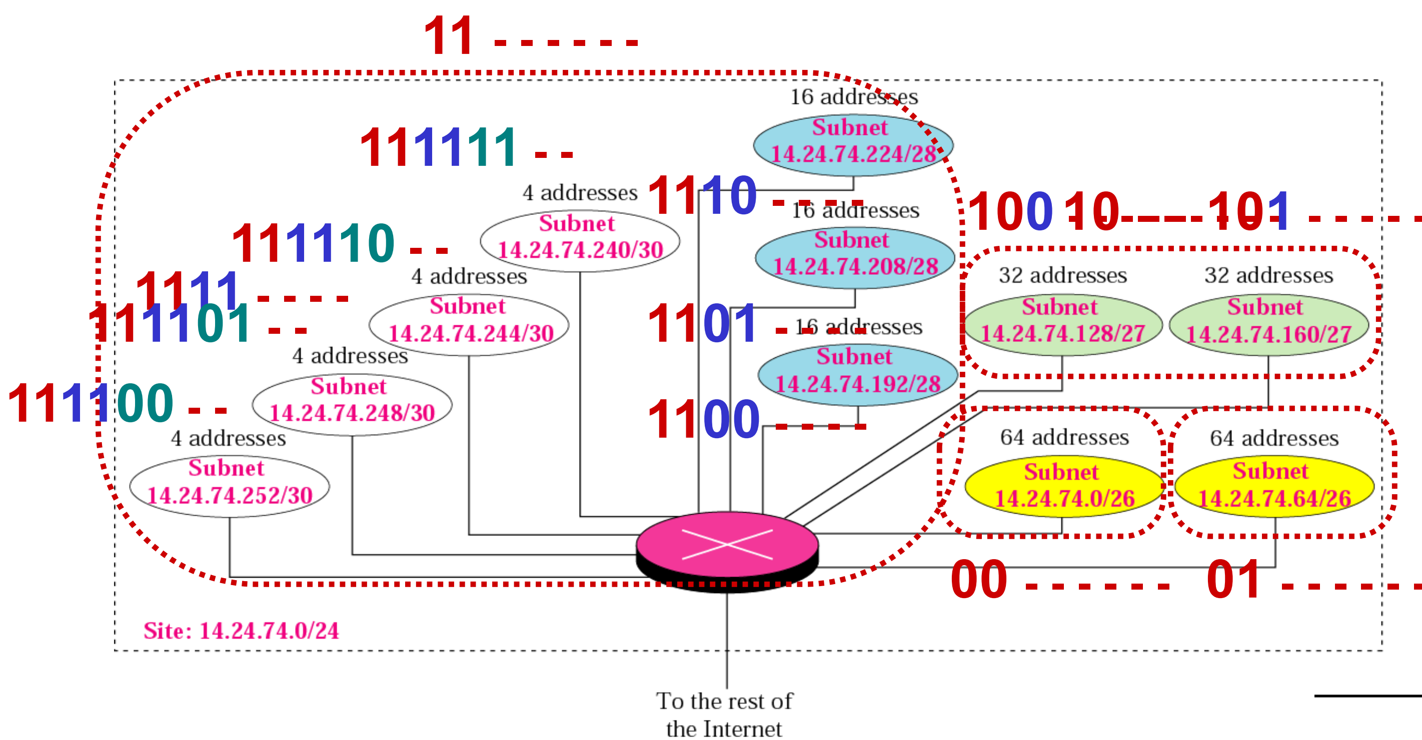
# Classless Addressing (cont.)

## Example [ subnetting in classless networks ]

**An organization is granted a block of addresses with the beginning address: 14.24.74.0/24. There are $2^{32-24}$=256 addresses in this block. The organization needs to have 11 subnets, with 256 host in total, as shown below:**

- **2 subnets, each with 64 addresses** – need 6-bit long hostIDs!
- **2 subnets, each with 32 addresses** – need 5-bit long hostIDs!
- **3 subnets, each with 16 addresses** – need 4-bit long hostIDs!
- **4 subnets, each with 4 addresses** – need 2-bit long hostIDs!

**Design the subnets. Assume all 0-s and all 1-s subnet ID are allowed.**



# Classless Addressing (cont.)

$$14.24.74.00/24 = \underbrace{00001110\ \ 00011000\ \ 01001010}_{/24\ \text{bits}}\ /\ \underbrace{00000000}_{\text{remaining 8 bits}}$$

**1) With first 2 out of 8 available bits, we can create 4 networks (i.e. 4 blocks of addresses) each with 64 host. We use the first of the two blocks for the first two subnets.**

6 bits for host IDs

| | | | | | | |
|---|---|---|---|---|---|---|
| **Subnet 1:** | **14.24.74.00/26** | **=** | **00001110** | **00011000** | **01001010** | **00000000** |
| **Subnet 2:** | **14.24.74.64/26** | **=** | **00001110** | **00011000** | **01001010** | **01000000** |
| unused 1: | 14.24.74.128/26 | = | 00001110 | 00011000 | 01001010 | **10000000** |
| unused 2: | 14.24.74.192/26 | = | 00001110 | 00011000 | 01001010 | **11000000** |

**2) We use the third block of 64 addresses (unused 1) for the next two subnets, each with 32 hosts.**

5 bits for host IDs

| | | | | | | |
|---|---|---|---|---|---|---|
| **Subnet 3:** | **14.24.74.128/27** | **=** | **00001110** | **00011000** | **01001010** | **10000000** |
| **Subnet 4:** | **14.24.74.160/27** | **=** | **00001110** | **00011000** | **01001010** | **10100000** |

**3) We split the fourth block of 64 addresses (unused 2) into 4 sub-blocks, each with each with 16 hosts.**

4 bits for host IDs

| | | | | | | |
|---|---|---|---|---|---|---|
| **Subnet 5:** | **14.24.74.192/28** | **=** | **00001110** | **00011000** | **01001010** | **11000000** |
| **Subnet 6:** | **14.24.74.208/28** | **=** | **00001110** | **00011000** | **01001010** | **11010000** |
| **Subnet 7:** | **14.24.74.224/28** | **=** | **00001110** | **00011000** | **01001010** | **11100000** |
| unused 3: | 14.24.74.224/28 | = | 00001110 | 00011000 | 01001010 | **11110000** |

# Classless Addressing (cont.)

4) **We use the last available sub-block for the last four subnets, each with 4 addresses.**

**2 bits for host IDs**

| | | | | | | |
|---|---|---|---|---|---|---|
| **Subnet 8:** | **14.24.74.240/30 =** | **00001110** | **00011000** | **01001010** | **11110000** |
| **Subnet 9:** | **14.24.74.244/30 =** | **00001110** | **00011000** | **01001010** | **11110100** |
| **Subnet 10:** | **14.24.74.248/30 =** | **00001110** | **00011000** | **01001010** | **11111000** |
| **Subnet 11:** | **14.24.74.252/30 =** | **00001110** | **00011000** | **01001010** | **11111100** |

---

**Note: The advantages of classless addressing come at certain price!**

**From:** **http://www.ietf.org/internet-drafts/draft-ietf-grow-rfc1519bis-02.txt**

"*With the change from classful network numbers to classless prefixes, it is not possible to infer the network **mask** from the initial bit pattern of an **IPv4** address. This has implications for how routing information is stored and propagated. Network masks or prefix lengths must be explicitly carried in routing protocols …*

*Similarly, routing and forwarding tables in layer-3 network equipment must be organized to store both prefix and prefix length or **mask**. Equipment which organizes its routing/forwarding information according to legacy class A/B/C network/subnet conventions cannot be expected to work correctly …* "

# Obtaining IP Address and Masks

**Which IP address info should be known to host?**

– each computer attached to the Internet must have the following information

- its own IP address [host IP address]
- its subnet mask
- the IP address of a router
- the IP address of a DNS server

**How does host get its IP address?**

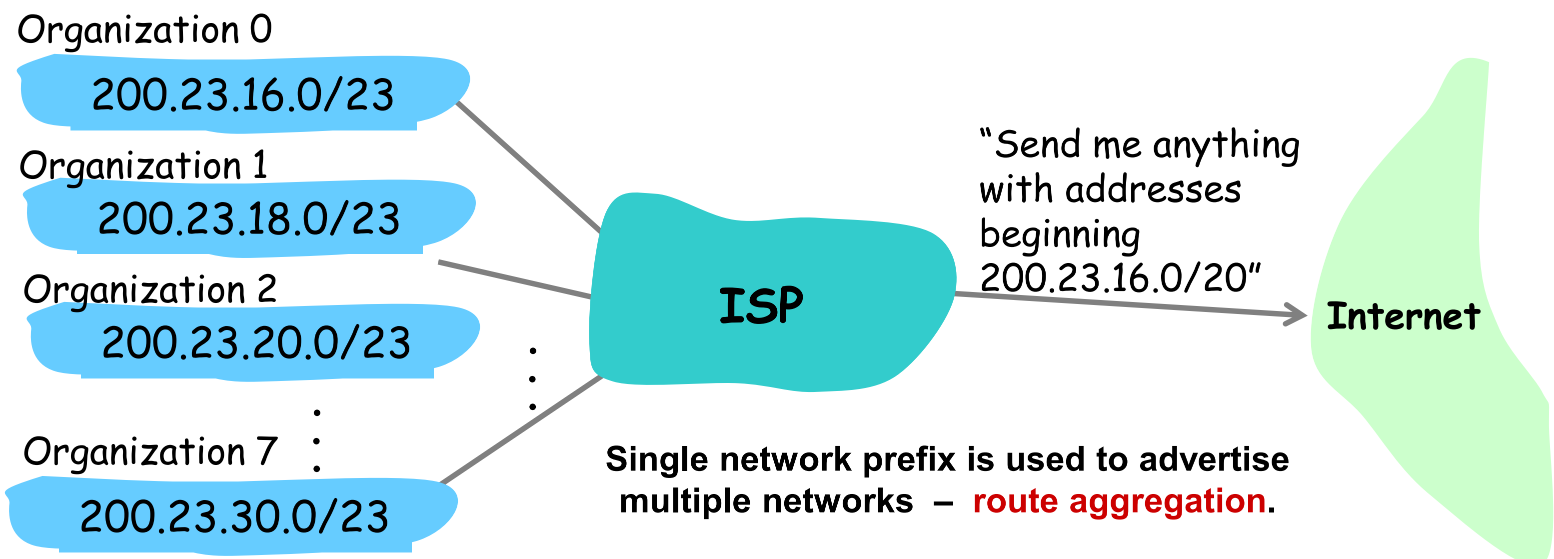
(1) **manual configuration**: a system administrator manually configures the IP address into the host (typically in a file)

- Wintel: *control-panel->network->configuration->tcp/ip->properties*, or simply type *ipconfig*
- UNIX: */etc/rc.config*

(2) **Dynamic Host Configuration Protocol (DHCP)**: host obtains an IP address automatically, as well as additional information such as the address of its first-hop router and the address of its DNS server

# Obtaining IP Address (cont.)

**How does network get its IP address?** – **to obtain a block of IP addresses network administrator must first contact its ISP**




**Single network prefix is used to advertise multiple networks – route aggregation.**

**How does ISP get a block of IP addresses?** – **apply to Internet Corporation for Assigned Names and Numbers (ICANN) – international authority for managing the IP address space**

**(see: http://www.icann.org/general/)**