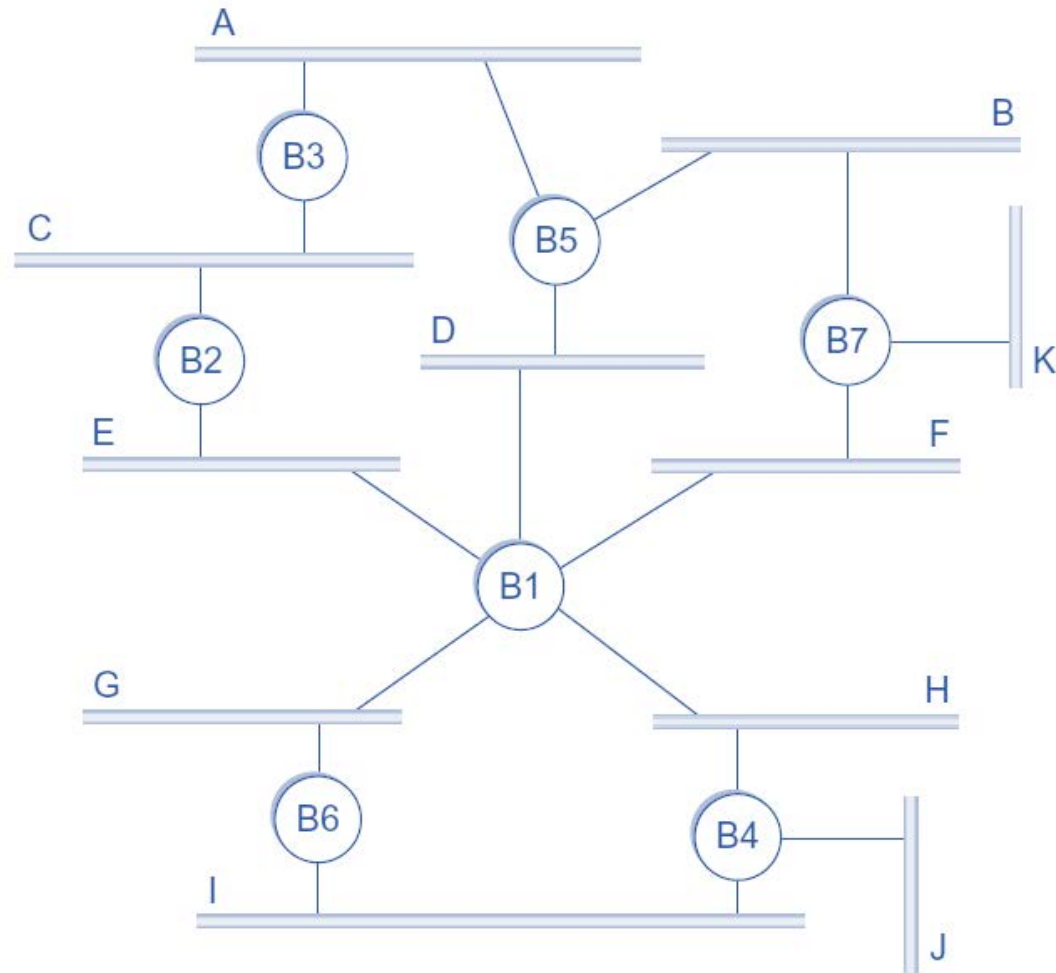




# CS120: Computer Networks

## **Lecture 9. Internet Protocol**

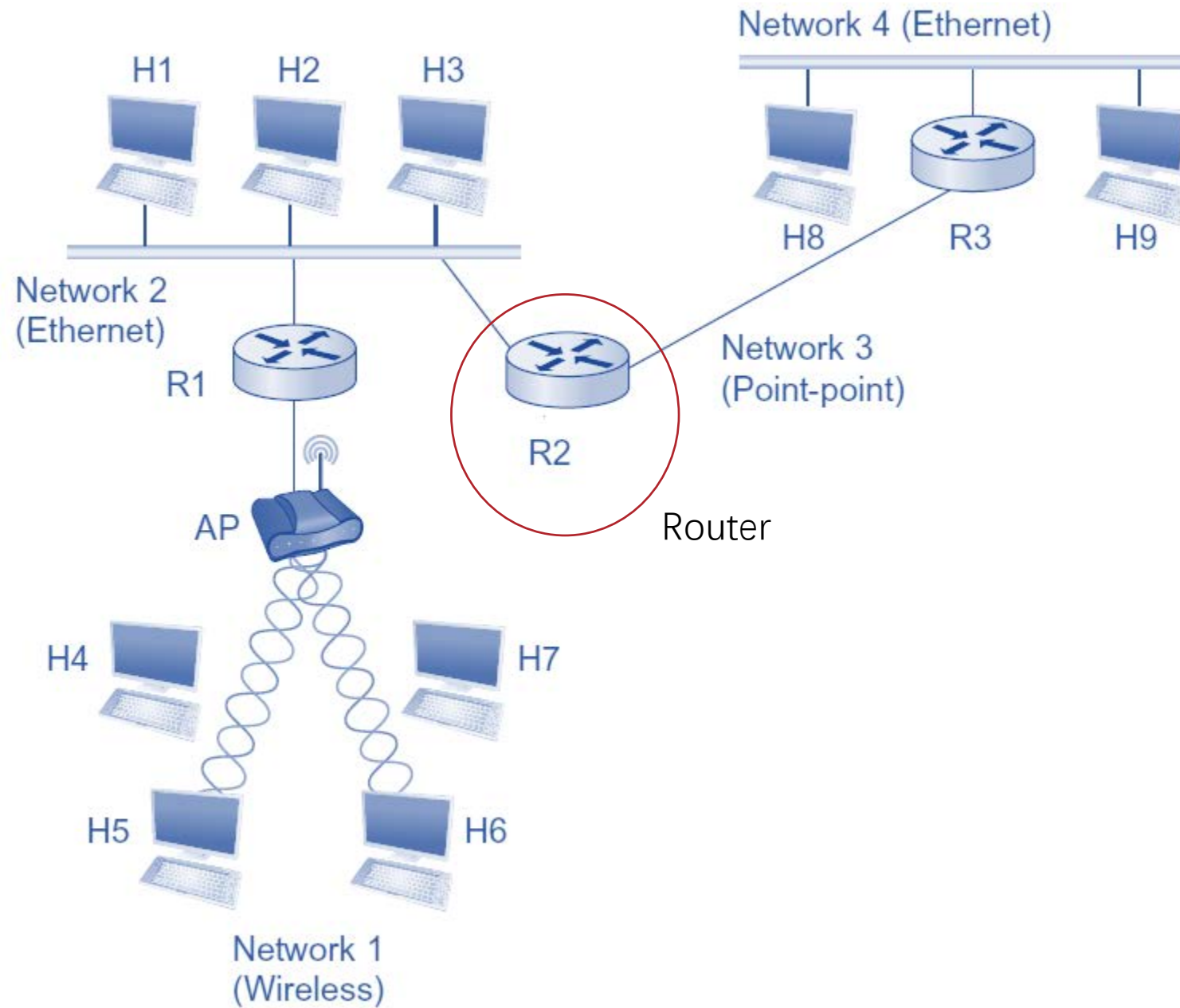
Zhice Yang



How to Further Extend the Network ?

# Limitation of Extended Ethernet

- Addressing Scalability
  - Spanning Tree does not scale
    - Large network
      - Switches store too many forwarding entries
      - Huge broadcasting overhead
- Network Heterogeneity
  - Cannot communicate with other networks
    - Cannot addressing nodes in other networks



# Internet Protocol (IP)

- Goal:
- Scalable Addressing Scheme
  - Support Heterogeneous Networks
- Service Model: Datagram (Connectionless)
  - Packets can be lost
  - Packets can be delivered out of order
  - Duplicate copies of a packet can be delivered
  - Packets can be delayed for a long time

# Outline

- IP Addressing
  - IP Address
  - Subnet
  - Routing Aggregation
  - IP Distribution: DHCP
  - IP and Switching: ARP
- IP Packet
  - Fragmentation

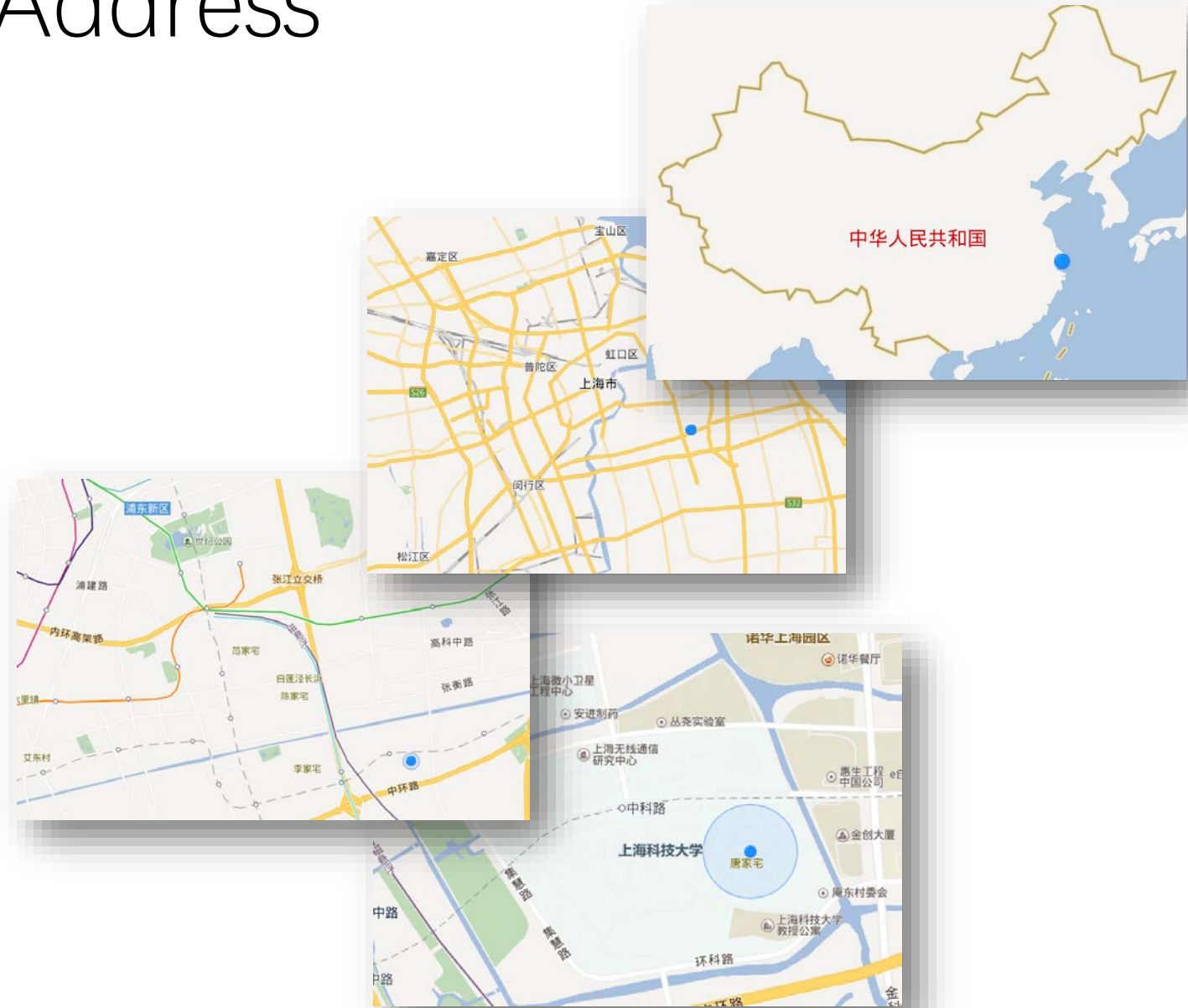
# Addressing in Postal Service

- NAME => Ethernet MAC Address
  - Unique
  - but less informative in finding route to deliver
- In practice we use: Location Address + NAME



# Hierarchical Address

- China
- Shanghai
- Pudong
- ShanghaiTech





- IP Address: 32-bit identifier for host or router ports
  - Globally unique (original goal)
  - Hierarchical: network + host (original goal)



# IP Address

- Dot notation
  - 10.3.2.4
  - 128.96.33.15
  - 192.12.69.77

10000000 01100000 00100010 00001111

128. 96. 33. 15

# Assigning IP Address

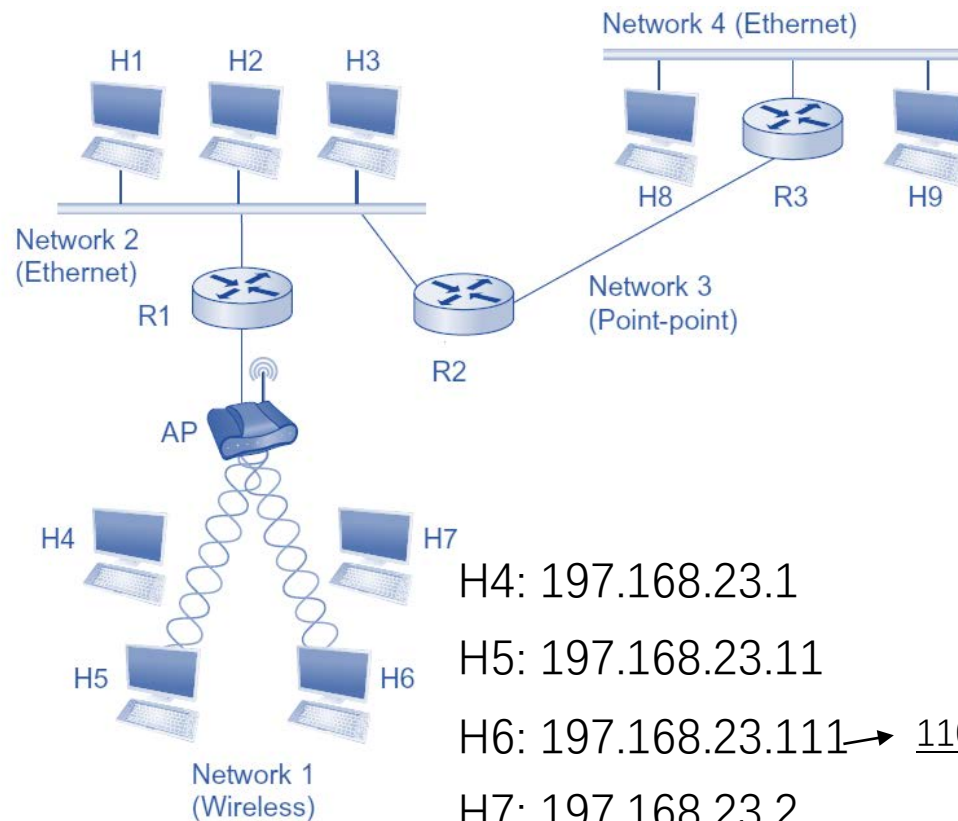
- Each host has a unique IP address
- Hosts in the same physical network have the same network part

H1: 200.155.11.5

H2: 200.155.11.3

H3: 200.155.11.2

11001000.10011011.00001011.XXXXXXXXXX



H8: 210.168.1.10

H9: 210.168.1.200

11010010.10101000.00000001.XXXXXXXXXX

H4: 197.168.23.1

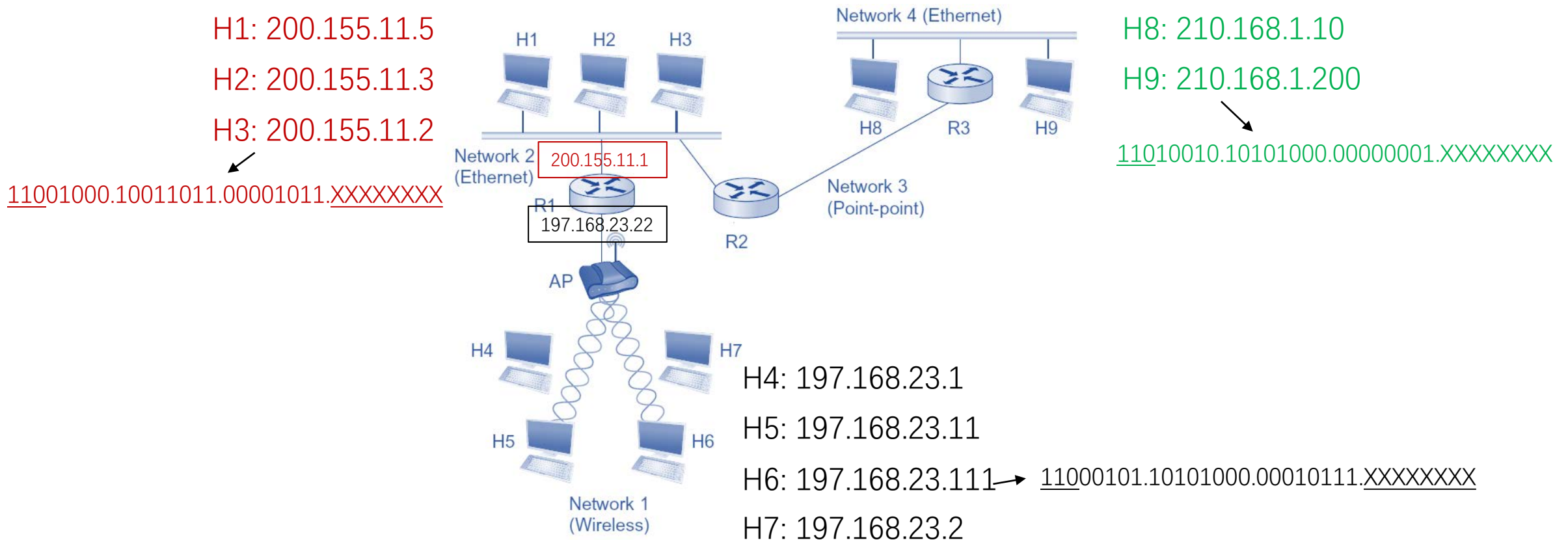
H5: 197.168.23.11

H6: 197.168.23.111 → 11000101.10101000.00010111.XXXXXXXXXX

H7: 197.168.23.2

# Assigning IP Address

- Each router contains multiple network interfaces
- Each port has the IP address of the connected network



# Forwarding with IP Address

- Each router maintains a table
  - if IP.network == Connected network
    - forward to the host (How? ARP: IP->MAC)
  - if IP.network != Connected network
    - forward to some router

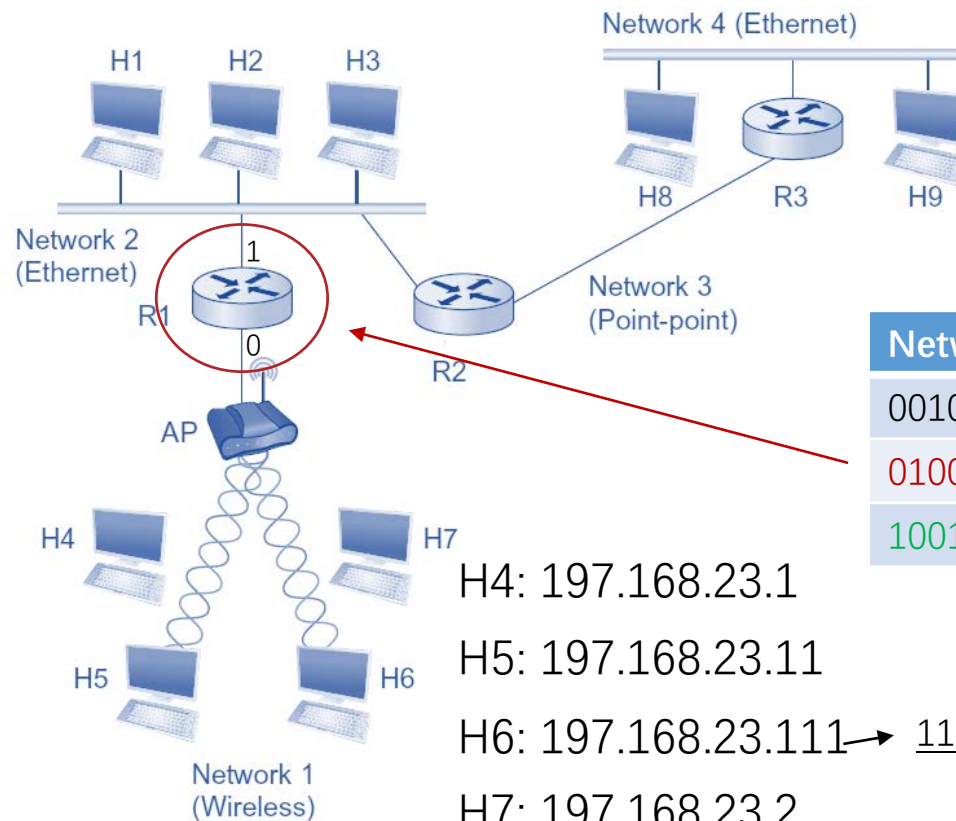
NO need to check the host part

H1: 200.155.11.5

H2: 200.155.11.3

H3: 200.155.11.2

11001000.10011011.00001011.XXXXXXXXXX



H8: 210.168.1.10

H9: 210.168.1.200

11010010.10101000.00000001.XXXXXXXXXX

NetworkNum	NextHop
00101.10101000.00010111	Interface 0
01000.10011011.00001011	Interface 1
10010.10101000.00000001	R2

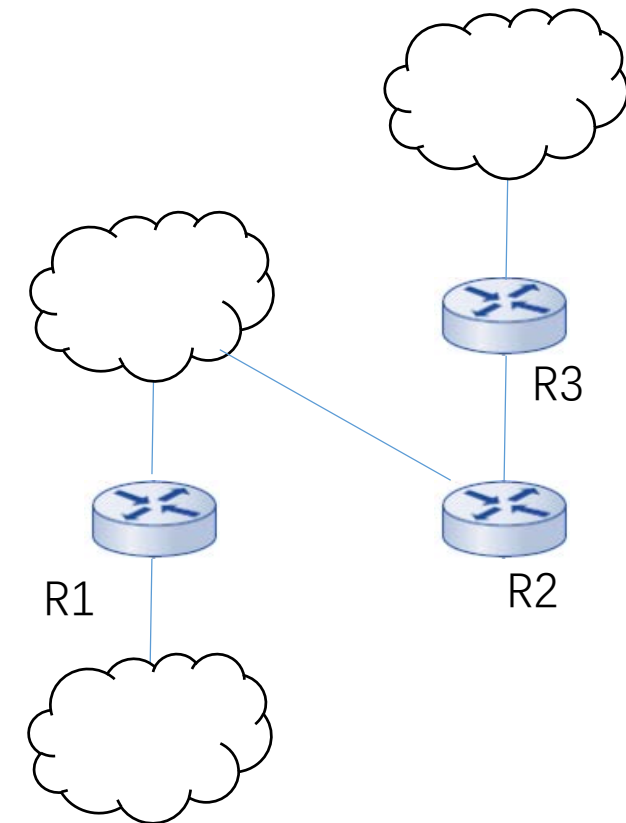
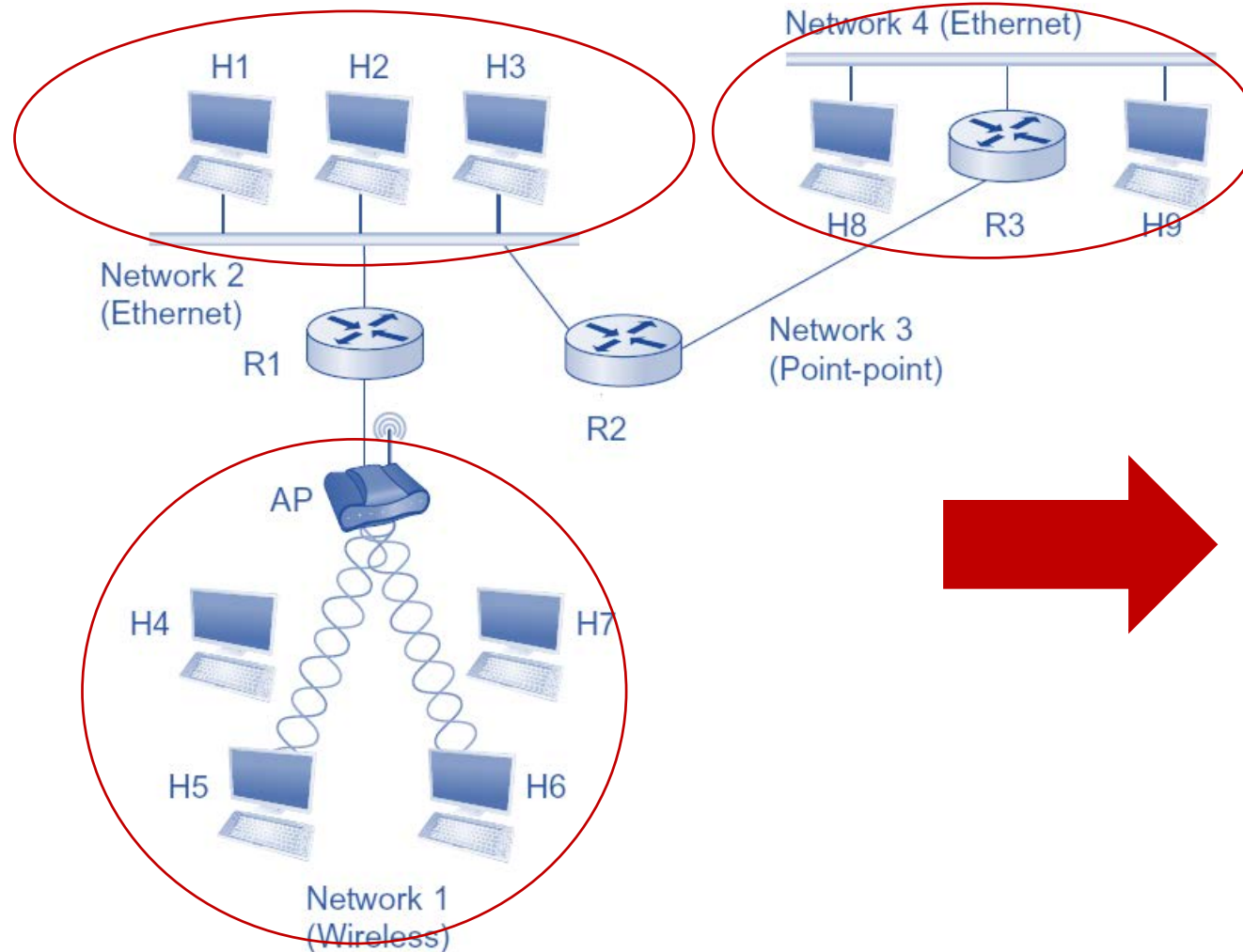
H4: 197.168.23.1

H5: 197.168.23.11

H6: 197.168.23.111 → 11000101.10101000.00010111.XXXXXXXXXX

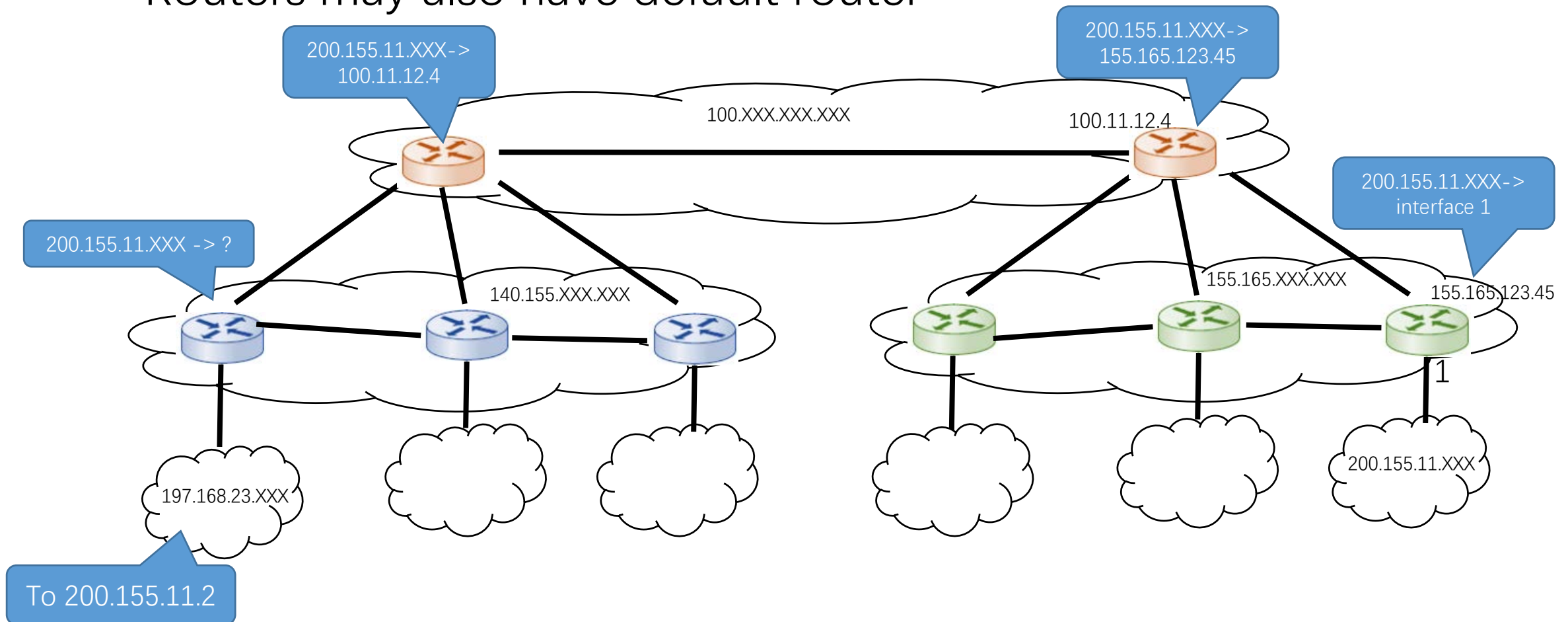
H7: 197.168.23.2

# Forwarding with IP Address



# Forwarding with IP Address

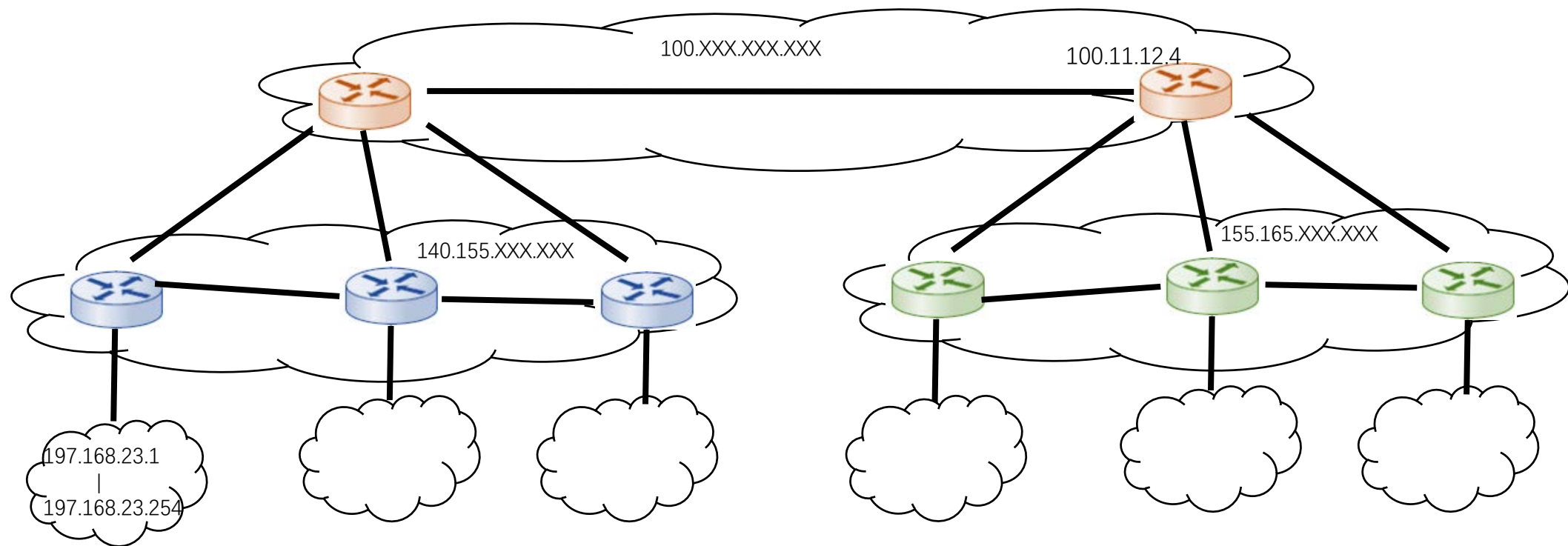
- Each host has a default router
- Routers may also have default router



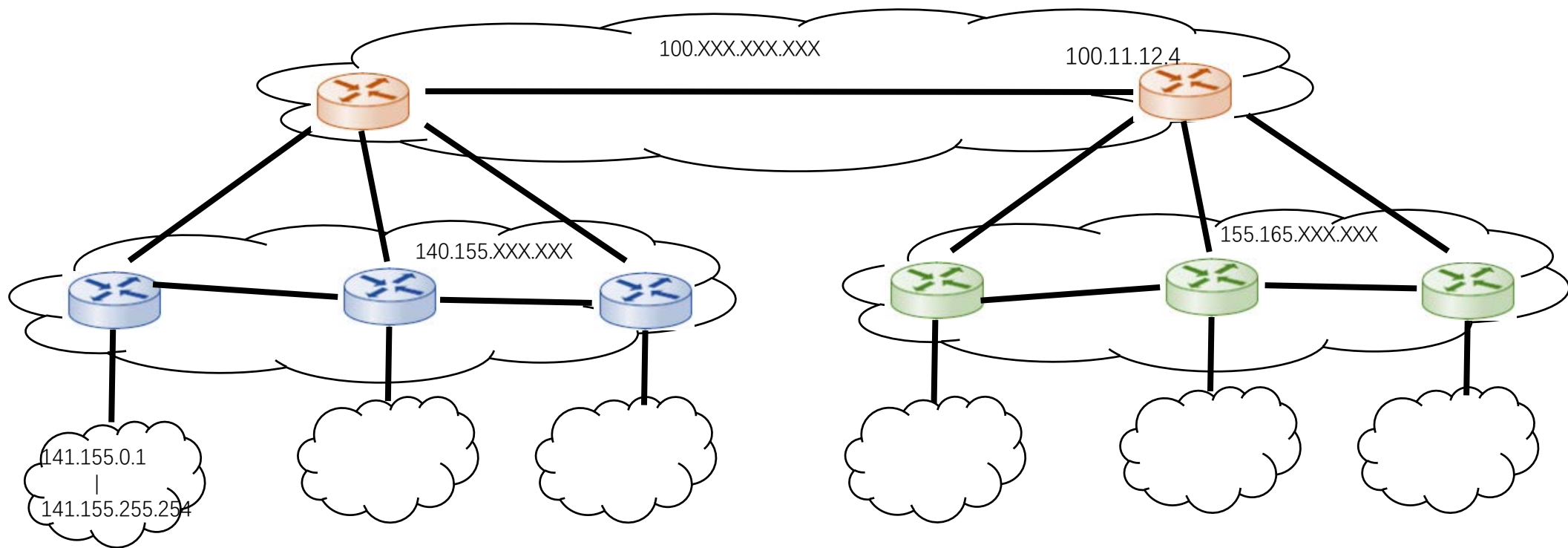
# Class Addressing

- Limitation
  - Address utilization is not efficient
    - 255 hosts
      - Class C: not enough
      - Class B: too many addresses are wasted
  - Forwarding table is still large
    - Proportional to the number of networks





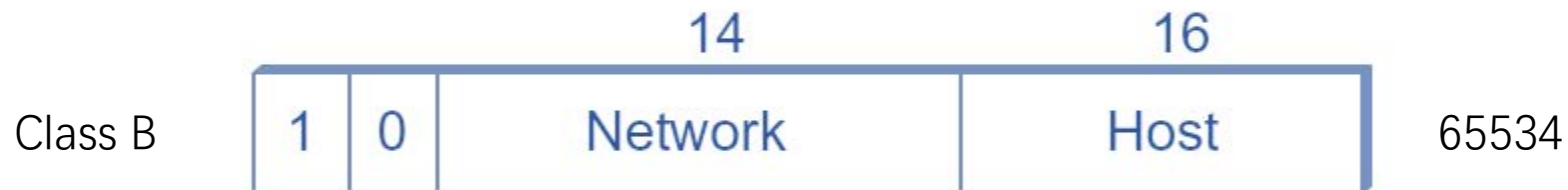
255 hosts  
Not Enough!



Not Efficient

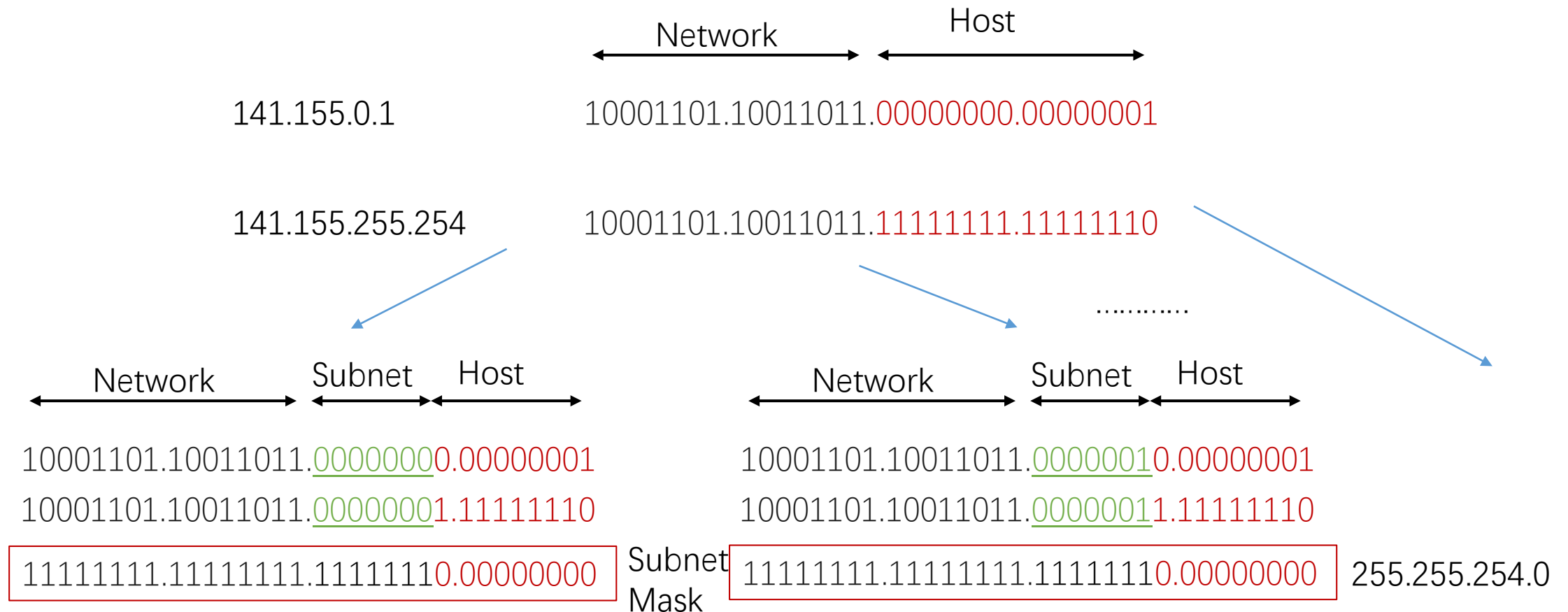
# IP Address

- IP Address: 32-bit identifier for host, router ports
  - Globally unique (original goal)
  - Hierarchical: network + host (original goal)



# Subnet Mask

- “and” IP address with network mask to determine the Subnet

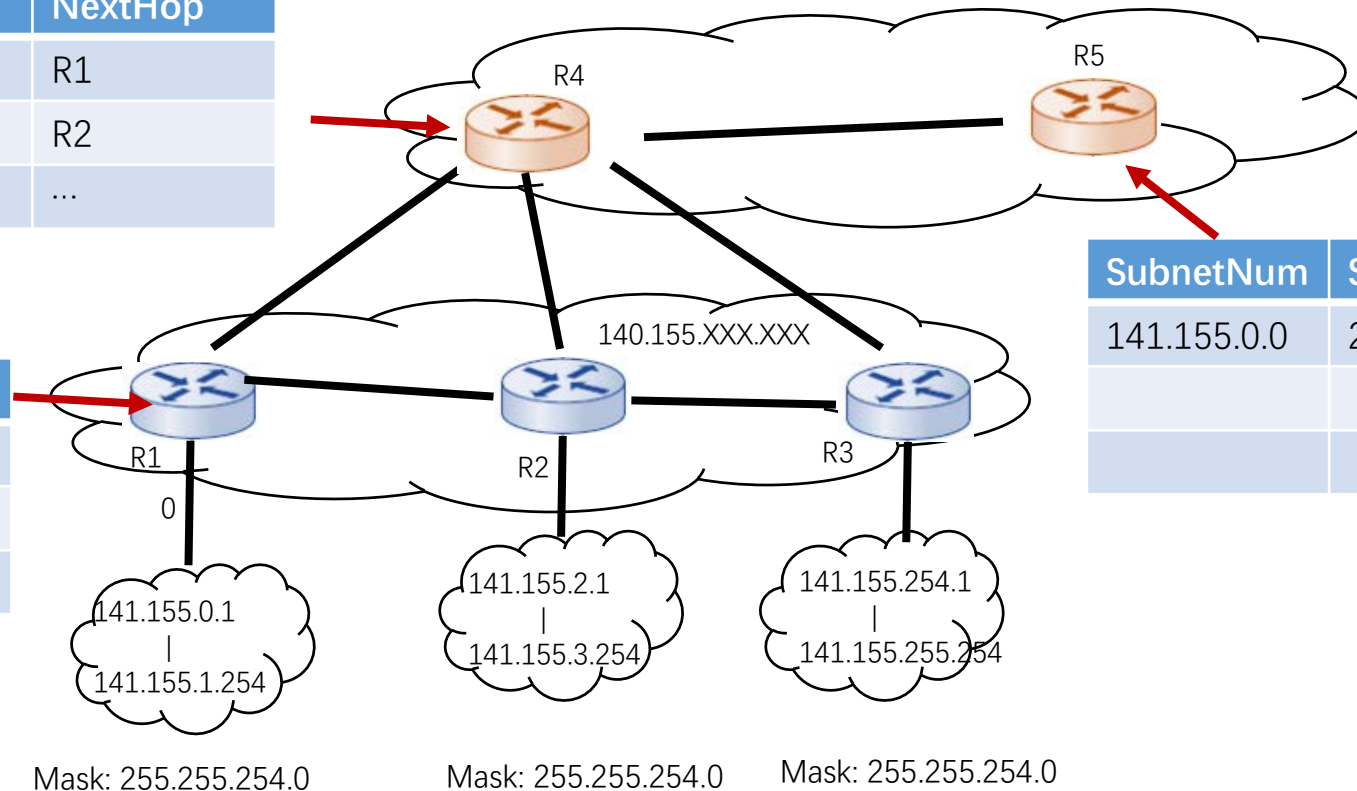


# Subnet Mask

- Divide a Large Network Address into Small Subnet Addresses

SubnetNum	SubnetMask	NextHop
141.155.0.0	255.255.254.0	R1
141.155.2.0	255.255.254.0	R2
...	...	...

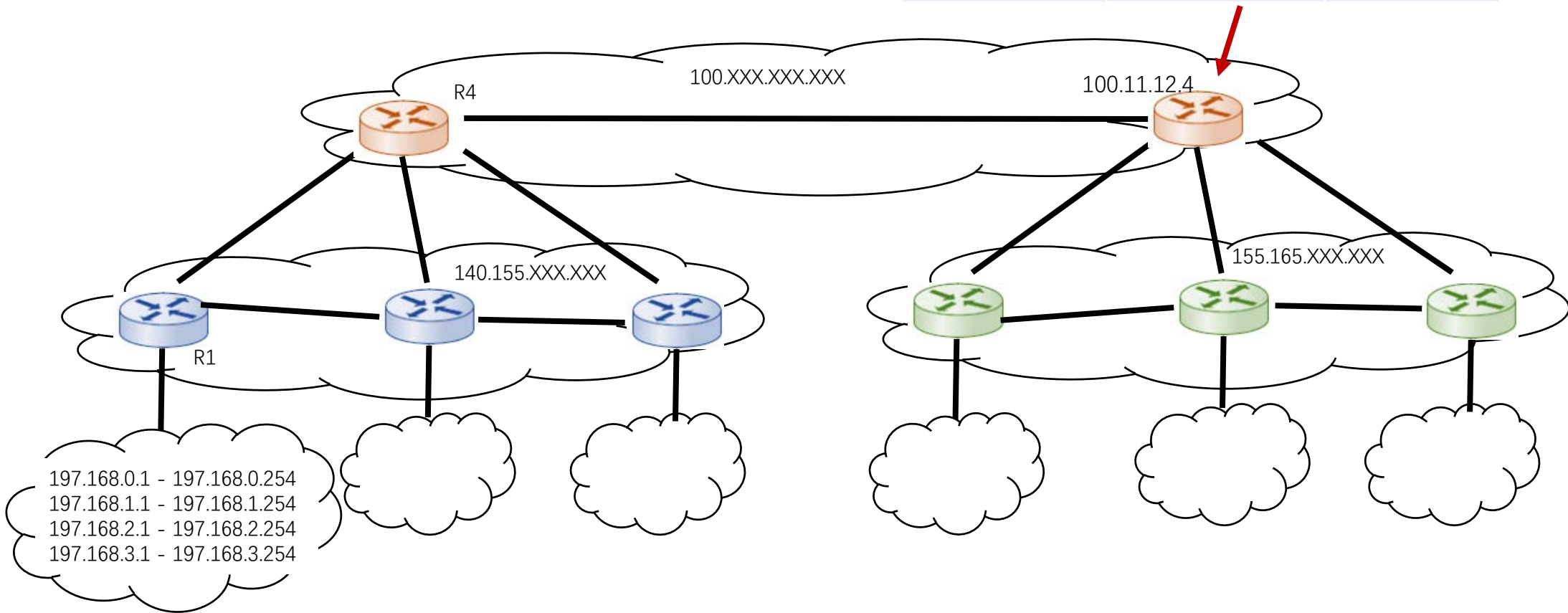
SubnetNum	SubnetMask	NextHop
141.155.0.0	255.255.254.0	Interface 0
141.155.2.0	255.255.254.0	R2
...	...	...



SubnetNum	SubnetMask	NextHop
141.155.0.0	255.255.0.0	R4

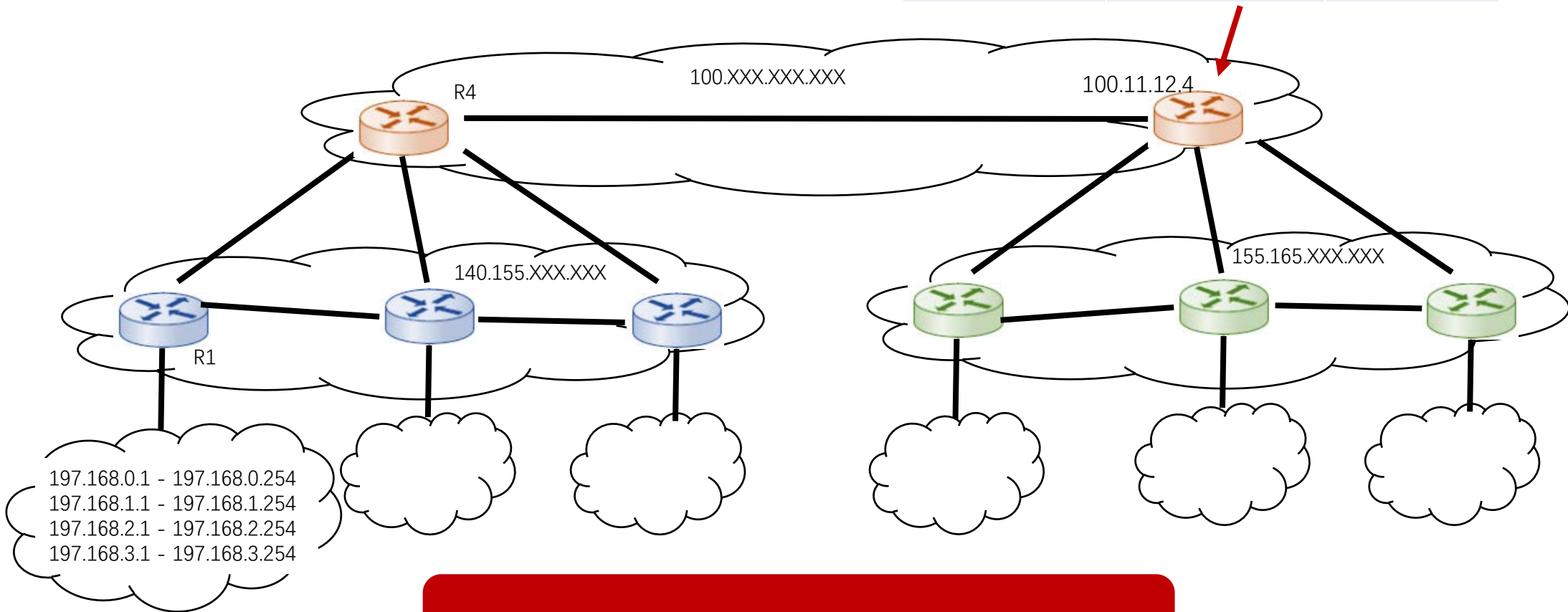
# Route Aggregation

SubnetNum	SubnetMask	NextHop
197.168.0.0	255.255.255.0	R4
197.168.1.0	255.255.255.0	R4
197.168.2.0	255.255.255.0	R4
197.168.3.0	255.255.255.0	R4



# Route Aggregation

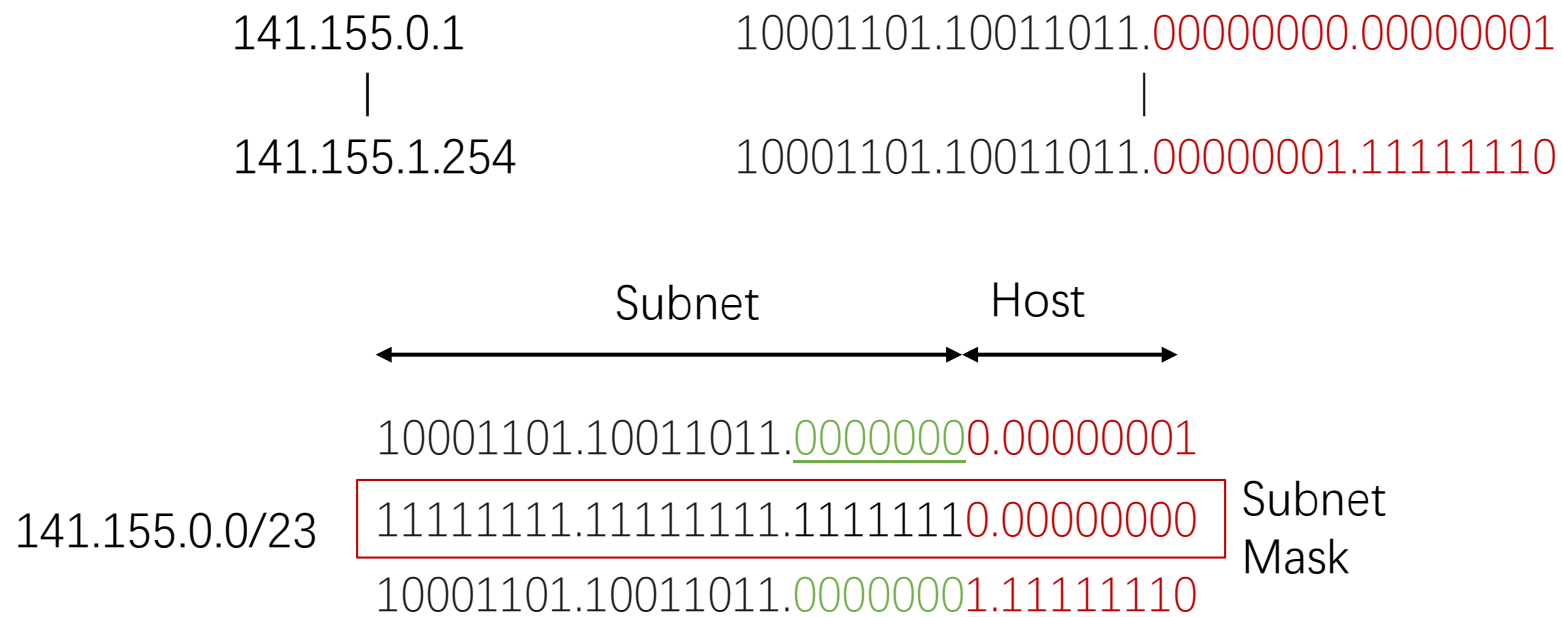
SubnetNum	SubnetMask	NextHop
197.168.0.0	255.255.252.0	R4



the Size of Forwarding Table is Reduced

# Classless InterDomain Routing (CIDR)

- Subnet portion of address is of arbitrary length
- Address format: a.b.c.d/x, where x is # bits in subnet portion of address

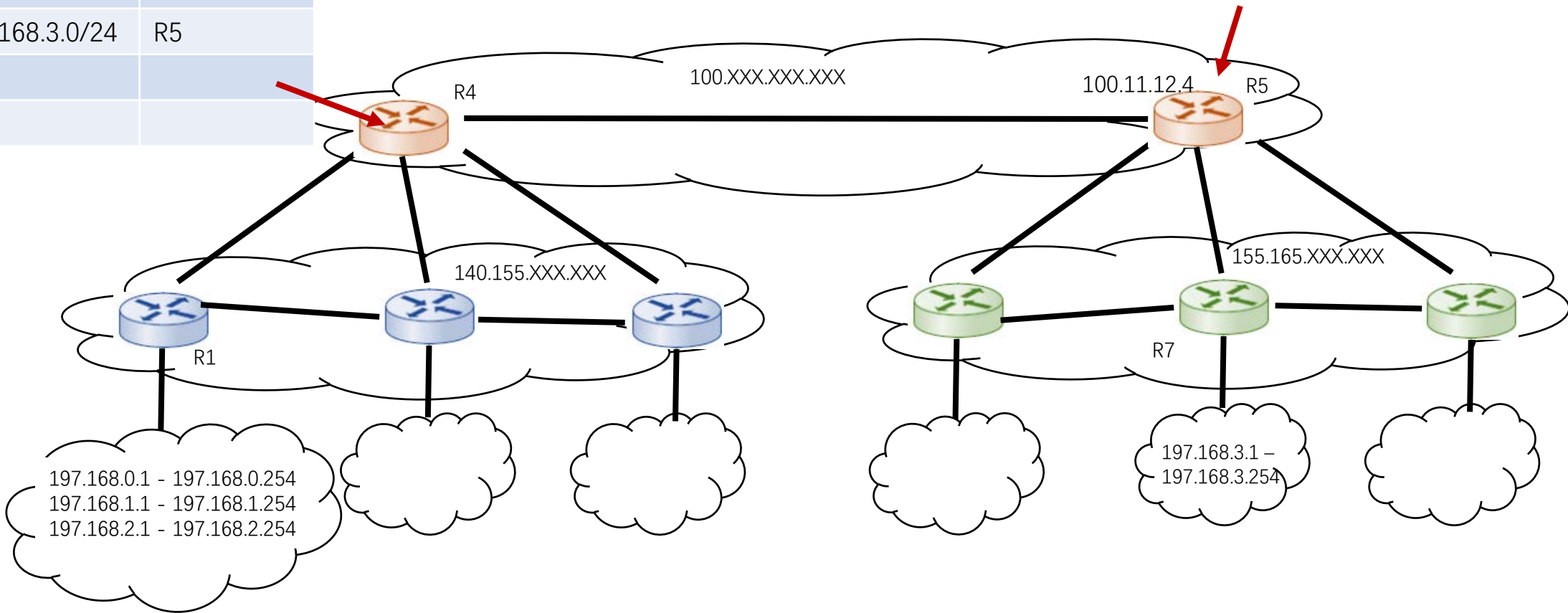




# Specific Routes ?

SubnetNum	NextHop
197.168.0.0/22	R1
197.168.3.0/24	R5

SubnetNum	NextHop
197.168.0.0/22	R4
197.168.3.0/24	R7



# Longest Prefix Matching

- When looking for forwarding table entry for given destination address, use longest address prefix that matches destination address.

SubnetNum	NextHop	
197.168.0.0/22	R4	11000101.10101000.000000**.*****
197.168.3.0/24	R7	11000101.10101000.00000011.*****
197.168.4.0/22	R9	11000101.10101000.000001**.*****

11000101.10101000.00000011.11010111



R7

11000101.10101000.00000111.11010111



R9

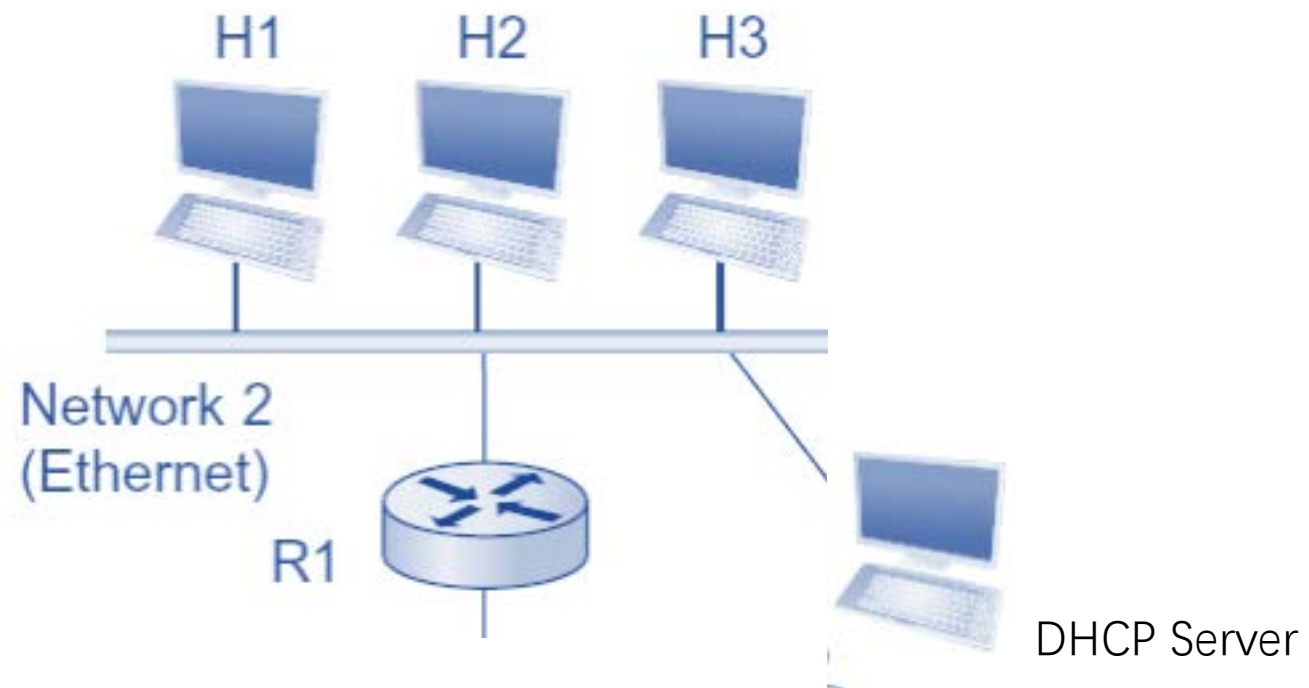
# How to Assign IP Addresses ?

- Hard-coded
- Dynamic Host Configuration Protocol (DHCP)
  - Dynamically get IP address from network server

# Dynamic Host Configuration Protocol (DHCP)

- Goal: allow host to dynamically obtain its IP address from network server when it joins the network
  - Reuse IP addresses
    - Release IP of unconnected host, e.g. power-off
    - Support for mobile hosts who want to join the network

# DHCP



# DHCP

DHCP Server  
223.1.2.1



discover

```
src MAC: MAC of client
dest MAC: FF:FF:FF:FF:FF:FF
src IP: 0.0.0.0
dest. IP: 255.255.255.255
yiaddr: 0.0.0.0
```

Client



offer

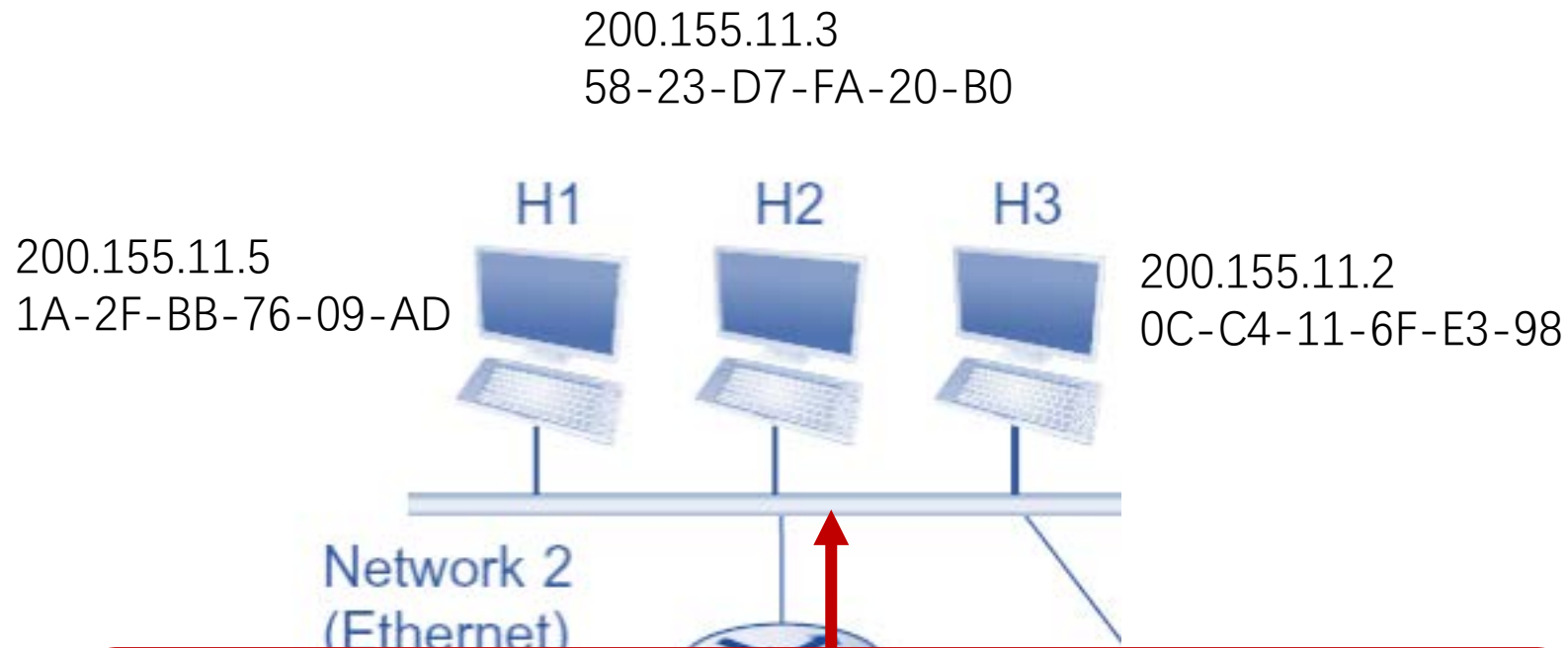
```
src MAC: MAC of server
dest MAC: MAC of client
src IP: 223.1.2.1
dest. IP : 255.255.255.255
yiaddr: 223.1.2.4
```

request

```
src MAC: MAC of client
dest MAC: FF:FF:FF:FF:FF:FF
src IP: 0.0.0.0
dest. IP : 255.255.255.255
ciaddr: 223.1.2.4
```

ack

```
src MAC: MAC of server
dest MAC: MAC of client
src : 223.1.2.1
dest.: 255.255.255.255
yiaddr: 223.1.2.4
```



How to Determine the Interface's MAC Address, Knowing its IP address?

# Address Resolution Protocol (ARP)


- A wants to send datagrams to B
  - if B's IP address is in the same subnet and B's MAC address not in A's ARP table
    - A broadcasts ARP query packet, containing B's IP address
    - B receives ARP packet, replies to A with its (B's) MAC address
      - Frame is sent to A's MAC address (unicast)
    - A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)



# Address Resolution Protocol (ARP)

- ARP table: each IP node (host, router) on LAN has table IP/MAC address mappings for some LAN nodes
  - < IP address; MAC address; TTL >
- TTL (Time To Live)
  - Time after which address mapping will be forgotten

200.155.11.3; 58-23-D7-FA-20-B0  
200.155.11.5; 1A-2F-BB-76-09-AD  
200.155.11.2; 0C-C4-11-6F-E3-98



Addressing: from one  
LAN to Another

# Demo

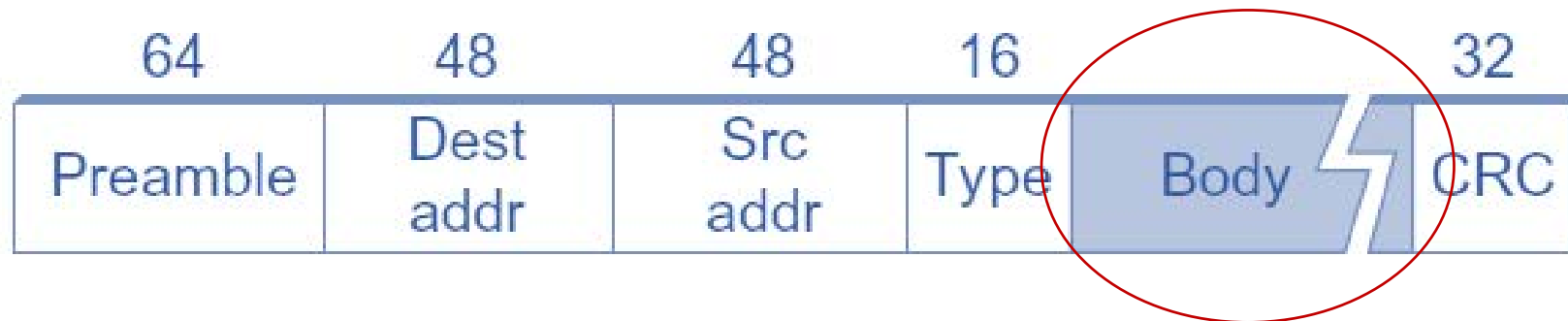
- DHCP
  - Four handshake messages
    - `ipconfig /release`
    - `ipconfig /renew`
- ARP
  - Show arp table: `arp -a`
- Forwarding Table
  - Show Forwarding Table: `route print`

# Outline

- IP Addressing
  - IP Address
  - Subnet
  - Routing Aggregation
  - IP Distribution: DHCP
  - IP and Switching: ARP
- IP Packet
  - Fragmentation

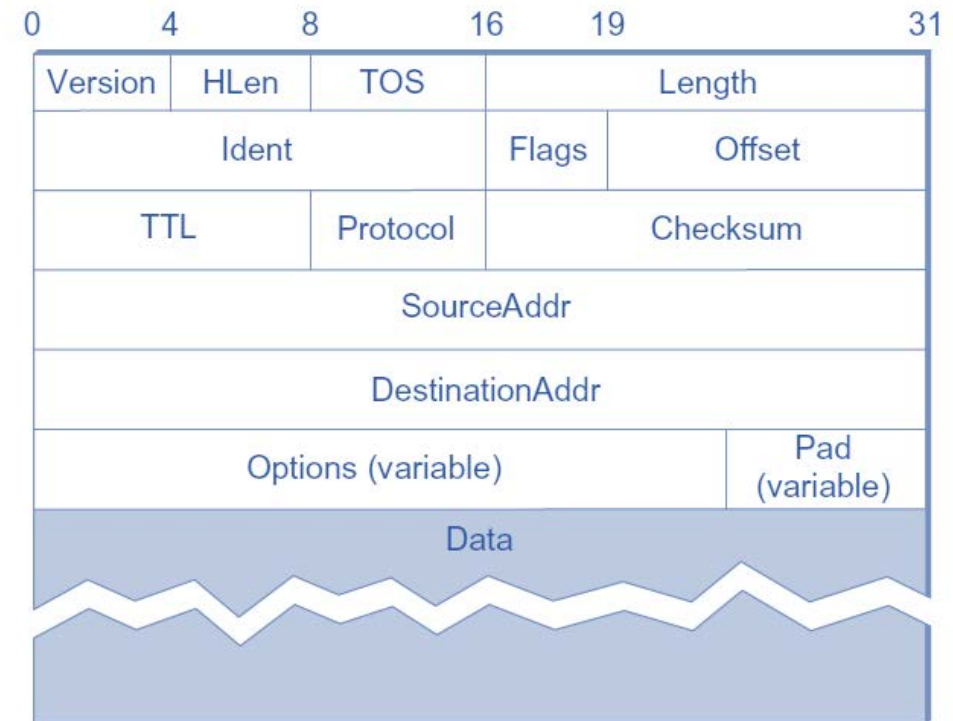
# Ethernet Frame

- Type
  - IPV4, ARP, RoCE, etc.
  - Length
- Body 46-1500 B
- CRC 32
- NO ACK



# Packet Format

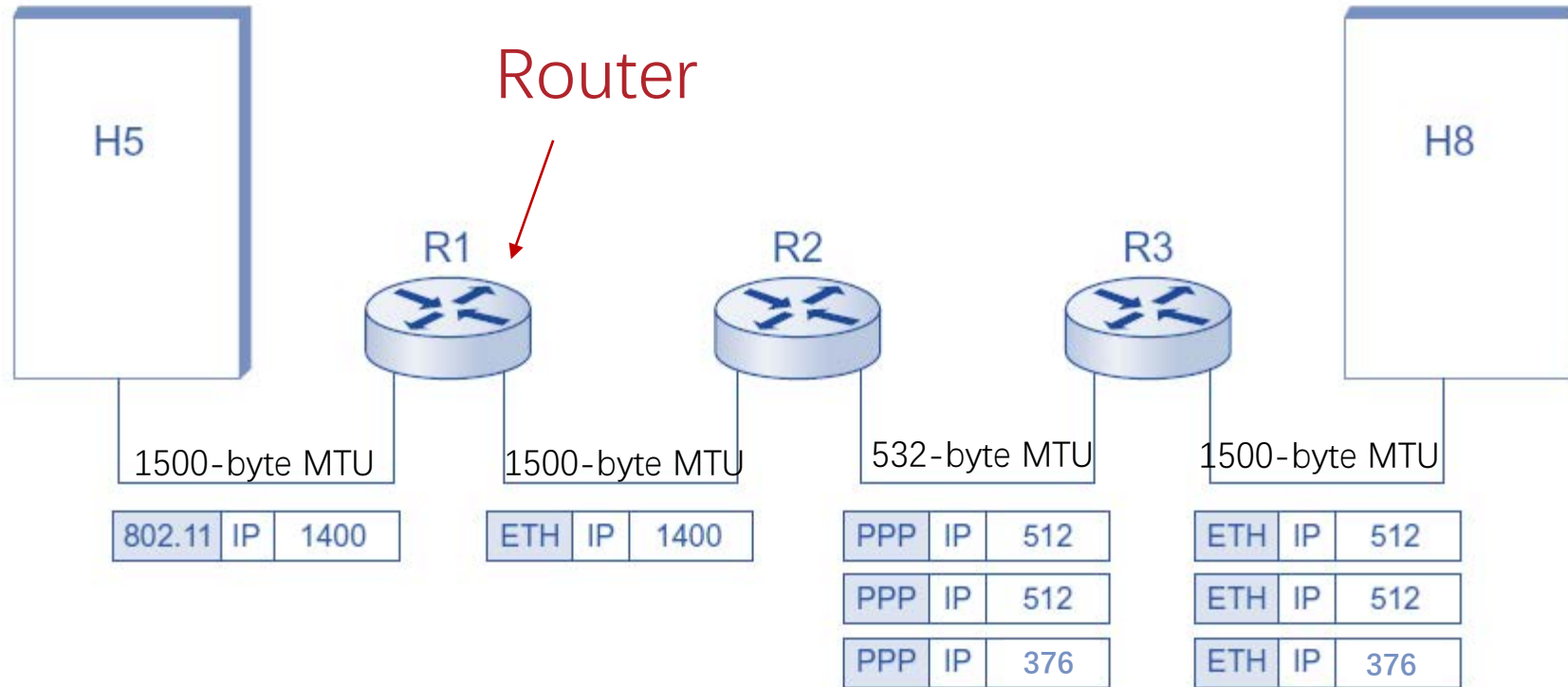
- Version (4): currently 4
- Hlen (4): number of 32-bit words in header
- TOS (8): type of service
- Length (16): number of bytes in this datagram
- Ident (16): used by fragmentation
- Flags/Offset (16): used by fragmentation
- TTL (8): number of hops this datagram has traveled
- Protocol (8): demux key (TCP=6, UDP=17)
- Checksum (16): of the header only
- DestAddr & SrcAddr (32)



# Fragmentation and Reassembly

- Network links have MTU (max.transfer size) - largest possible link-level frame
  - Different link types, different MTUs
- Large datagram is divided (“fragmented”) within net
  - One datagram becomes several datagrams
  - “Reassembled” only at final destination

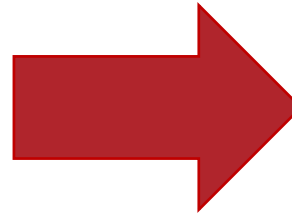
# Fragmentation and Reassembly





Identify the group of the fragments

Start of header				
Ident = x			0	Offset = 0
Rest of header				
1400 data bytes				



Start of header				
Ident = x			1	Offset = 0
Rest of header				
512 data bytes				

Start of header				
Ident = x			1	Offset = 64
Rest of header				
512 data bytes				

Start of header				
Ident = x			0	Offset = 128
Rest of header				
376 data bytes				

# Reference

- Textbook 3.2