Lecture 5: TCP Congestion Control. Network Layer. Part 1

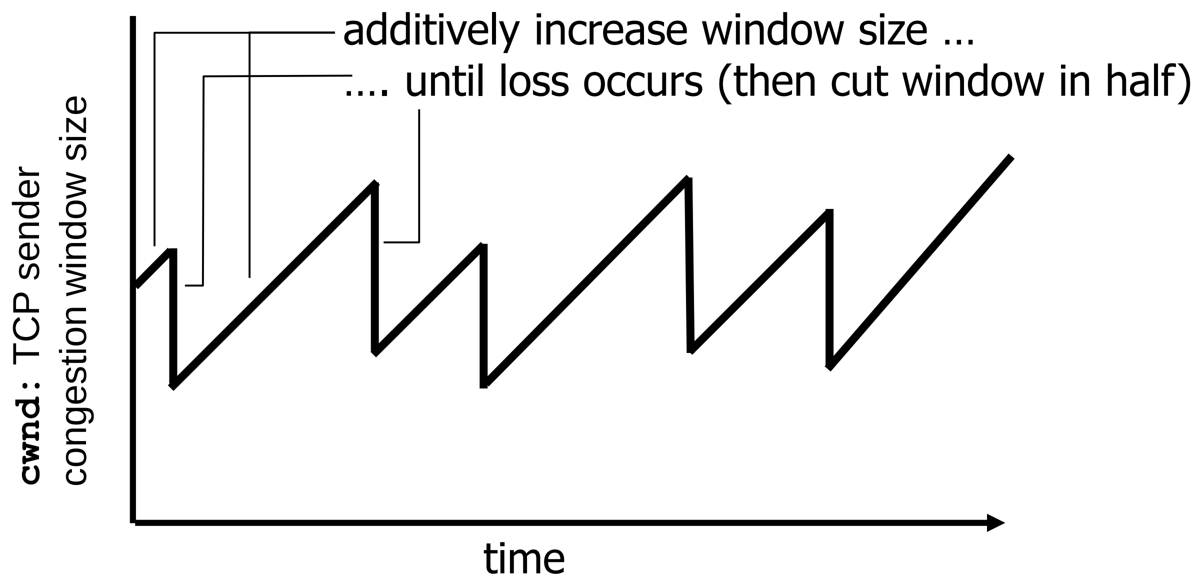
Acknowledgement: Materials presented in this lecture are predominantly based on slides from:

- *Computer Networking: A Top Down Approach,* J. Kurose, K. Ross, 7th ed., 2017, Addison-Wesley, Chapter 4
- *Business Data Communications and Networking,* J. Fitzgerald, A. Dennis, 10/11th ed., 2013, John Wiley & Sons, Chapter 5

TCP congestion control summary:
Additive Increase, Multiplicative Decrease (AIMD)

- *approach:* sender increases transmission rate (congestion window size **`cwnd`**), probing for usable bandwidth, until loss occurs
  - *additive increase:* increase **`cwnd`** (congestion window size) by 1 MSS (maximum segment size) every RTT until loss detected
  - *multiplicative decrease*: cut **`cwnd`** in half after loss

AIMD saw tooth behavior: probing for bandwidth



TCP congestion control fundamentals

- TCP congestion control is based on the idea that **each sender limits the rate** at which it sends traffic into its connection as a function of perceived network congestion.
  – If a TCP sender perceives that there is little congestion on the path between itself and the destination, then the TCP sender increases its send rate;
  – If the sender perceives that there is congestion along the path, then the sender reduces its send rate.
- The rate is controlled by means of the **congestion window (`cwnd`)** variable
- The congestion window imposes a constraint on the rate at which a TCP sender can send traffic into the network.
- Specifically, the amount of unacknowledged data at a sender may not exceed the minimum of **`cwnd`** and **`rwnd`,** that is:

`LastBytesent – LastByteAcked < min{cwnd, rwnd}`

Adjusting the transmission rate

- If the receive window **`rwnd`** is large enough so that we can ignore now the flow control, the above simplifies to

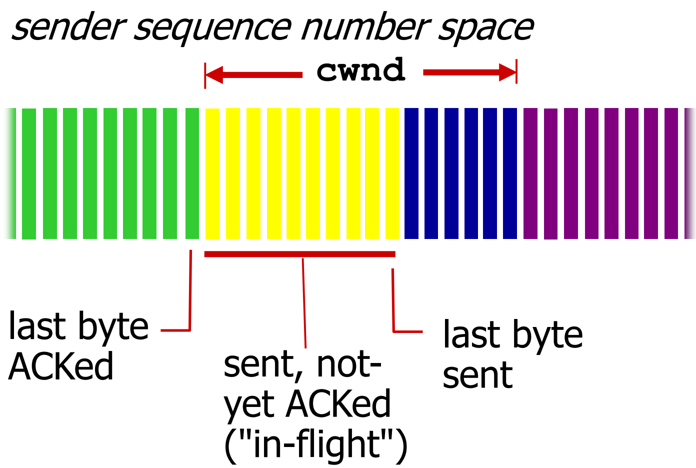
`LastBytesent – LastByteAcked < cwnd`

- Hence, he amount of unacknowledged data at the sender is solely limited by **`cwnd`**.
- If we assume that at the beginning of every **`RTT`**, the sender can send **`cwnd`** bytes then at the end of the **`RTT`** the sender receives acknowledgments for the data.
- Thus the sender's send rate is roughly

$$rate \approx \frac{cwnd}{RTT} \text{ bytes/sec}$$

- By adjusting the value of **`cwnd`**, the sender can therefore **adjust the rate** at which it sends data into its connection.

Congestion Window and Transmission rate



TCP sending rate:

- *roughly:* send **`cwnd`** bytes, wait RTT for ACKs, then send more bytes

$$\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

- sender limits transmission:

`LastByteSent - LastByteAcked < cwnd`

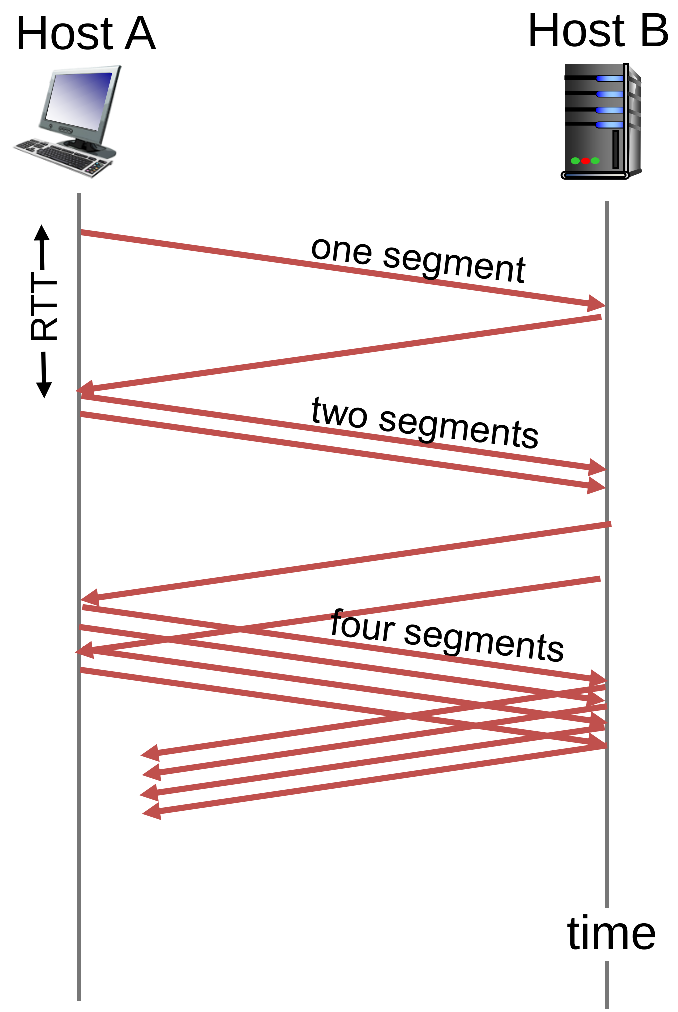
- **cwnd** is dynamic – function of **perceived** network congestion
- A **TCP sender perceives** that there is **congestion** on the path between itself and the destination by detecting a **"loss event":**
  - the occurrence of a **timeout** or
  - the receipt of **three duplicate ACKs**

TCP congestion-control algorithm

- TCP congestion-control algorithm (RFC 5681) was first described in [Jacobson 1988 and 1990]
- Four intertwined congestion control algorithms:
  - slow start,
  - congestion avoidance,
  - fast retransmit,
  - fast recovery.
- The RFC 5681 document also
  - specifies how TCP should begin transmission after a relatively long idle period, as well as
  - discusses various acknowledgment generation methods.

TCP Slow Start

- when connection begins, increase rate exponentially until first loss event:
  - initially **cwnd** = 1 MSS
  - **double cwnd** every RTT
  - done by incrementing **cwnd** for every ACK received
- initial rate is slow but ramps up exponentially fast
- When the congestion is detected, **cwnd** is typically **halved,** and the **congestion-avoidance mode** is entered



TCP: switching from slow start to CA

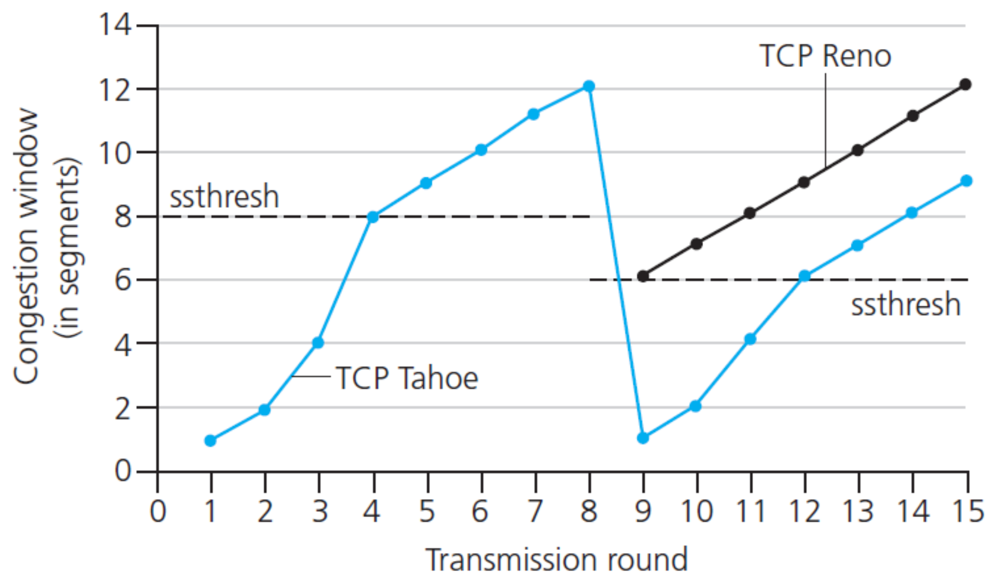
Q: when should the exponential increase switch to linear?

A: when **cwnd** gets to 1/2 of its value before a loss event

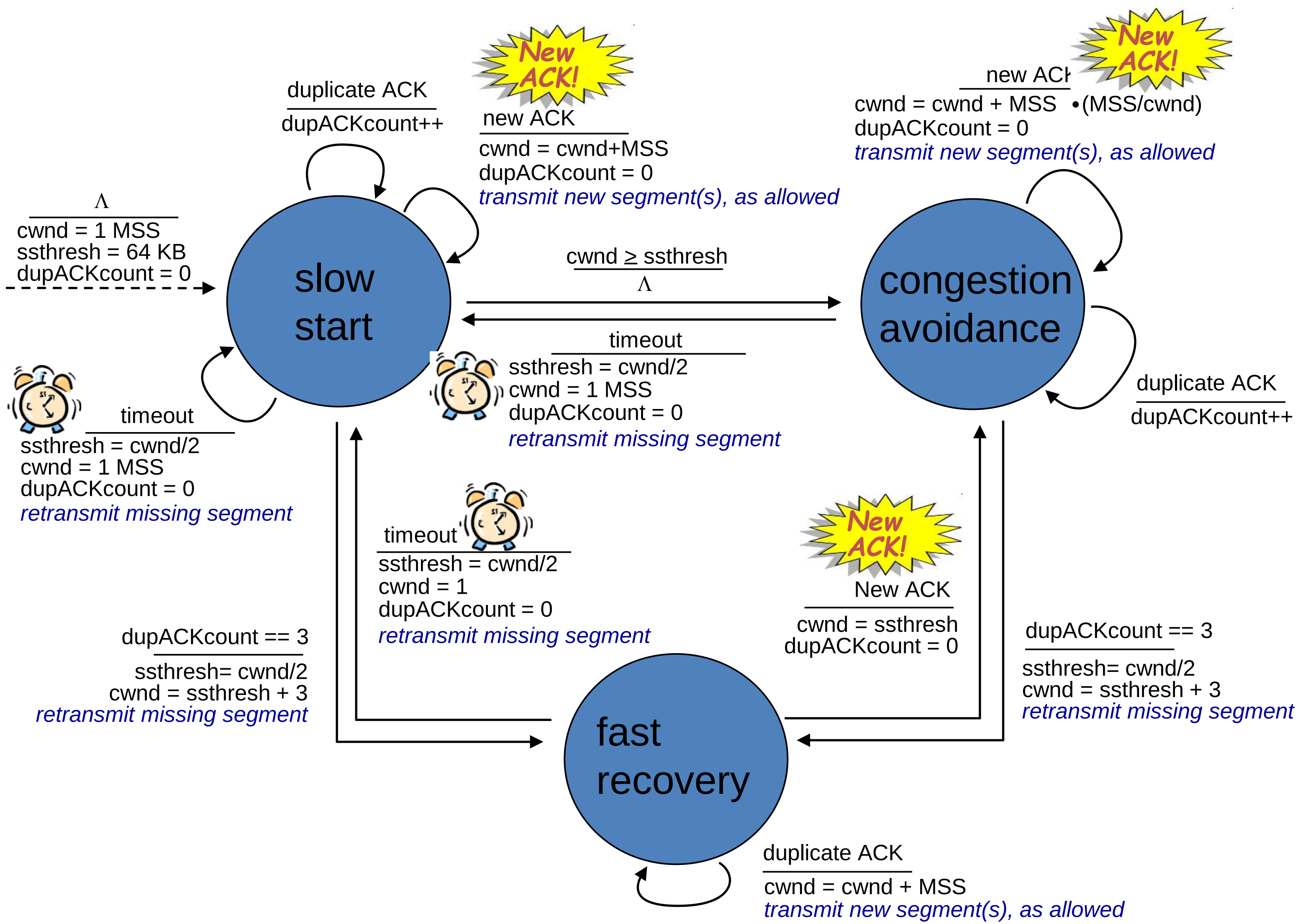
Implementation:

- variable **ssthresh**
- on loss event, **ssthresh** is set to 1/2 of **cwnd** just before loss event

Fast recovery



TCP Congestion Control Flow Chart



Comments to the TCP Congestion Control Flow Chart

- Three basic states:
  - slow start,
  - congestion avoidance,
  - fast recovery.
- Adjustable state variables:
  - cwnd
  - ssthresh
  - dupACKcount
- Input signals:
  - new ACK
  - duplicate ACK
  - dupACKcount == 3
  - timeout
  - cwnd > ssthresh

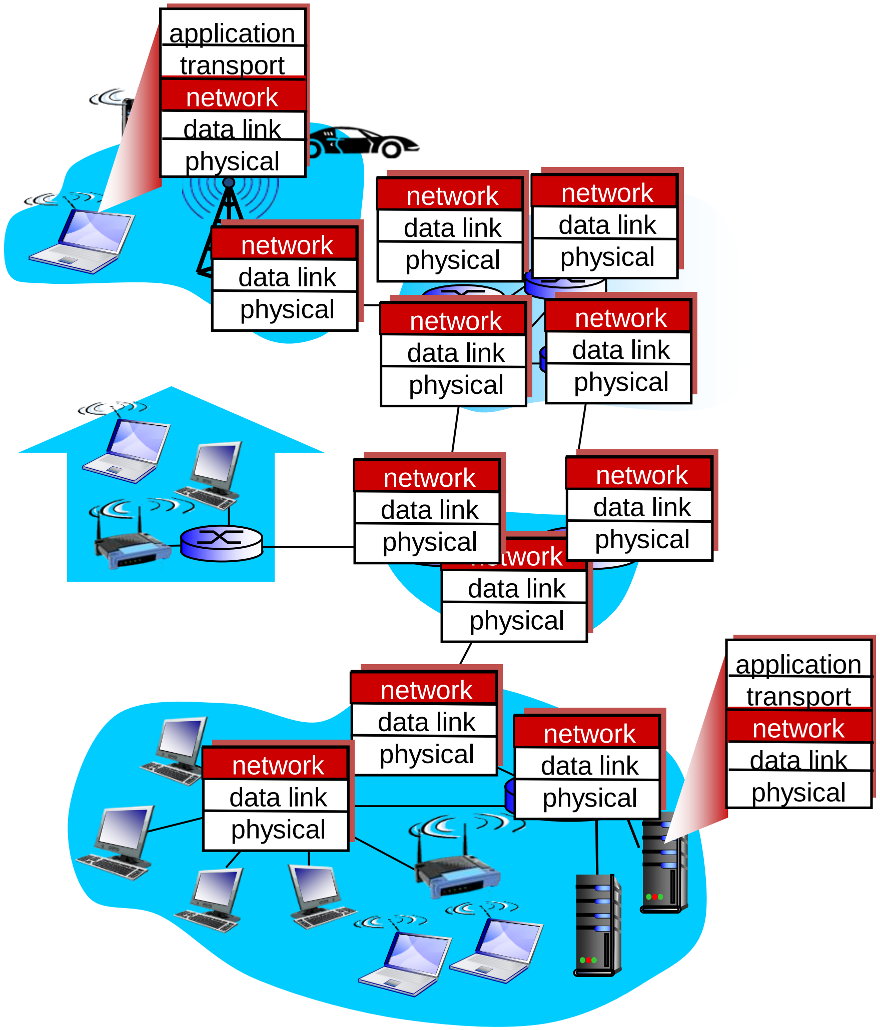
Lecture 5: Network Layer. Part 1

Outline

- General description of functions:
  - addressing
  - routing
  - forwarding
- Structure of the IPv4 packet
- Internet and IPv4 addressing
- Subnets
- Classless InterDomain Routing (CIDR)
- Allocation of the IP addresses. ICANN.
- DHCP protocol
- Address resolution
- DNS

Network layer: General Functions

- transports segments from sending to receiving host
- on sending side encapsulates segments into datagrams aka IP packets
- on receiving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- router examines header fields in all IP packets passing through it

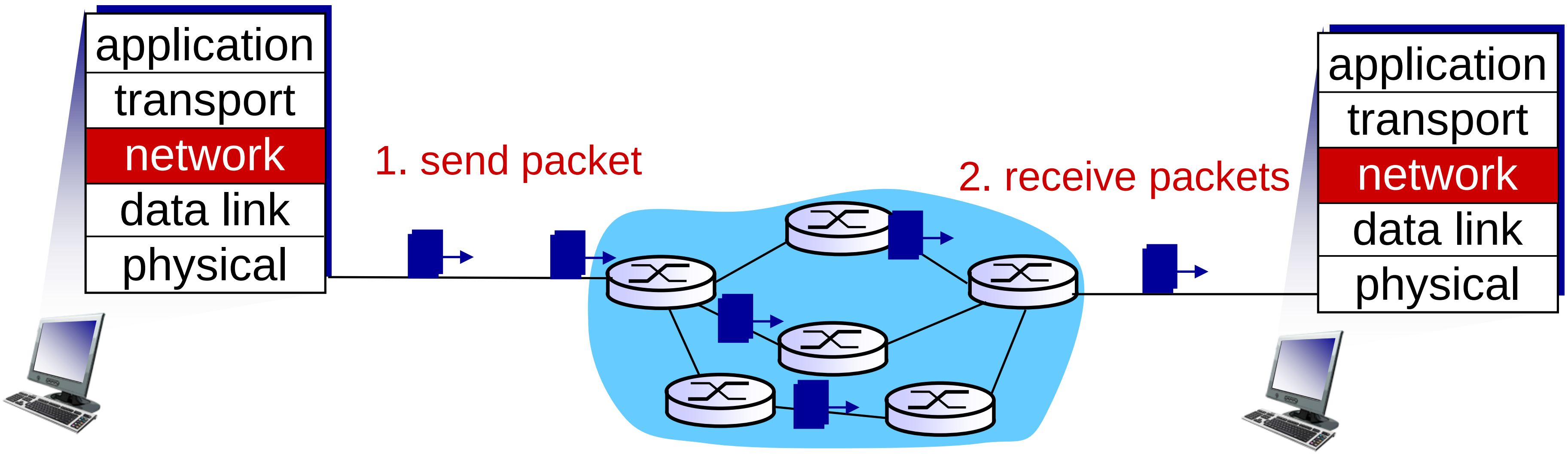


Key network-layer functions

- *Addressing:*
  - Three levels of addressing*: URLs, IP addresses, Physical/MAC addresses*
  - Need for the address resolution procedures to translate between addresses
- *routing:* determine route taken by packets from source to destination
- *forwarding:* local function at the router (similar to the switch forwarding) – move packets from router’s input to appropriate router output

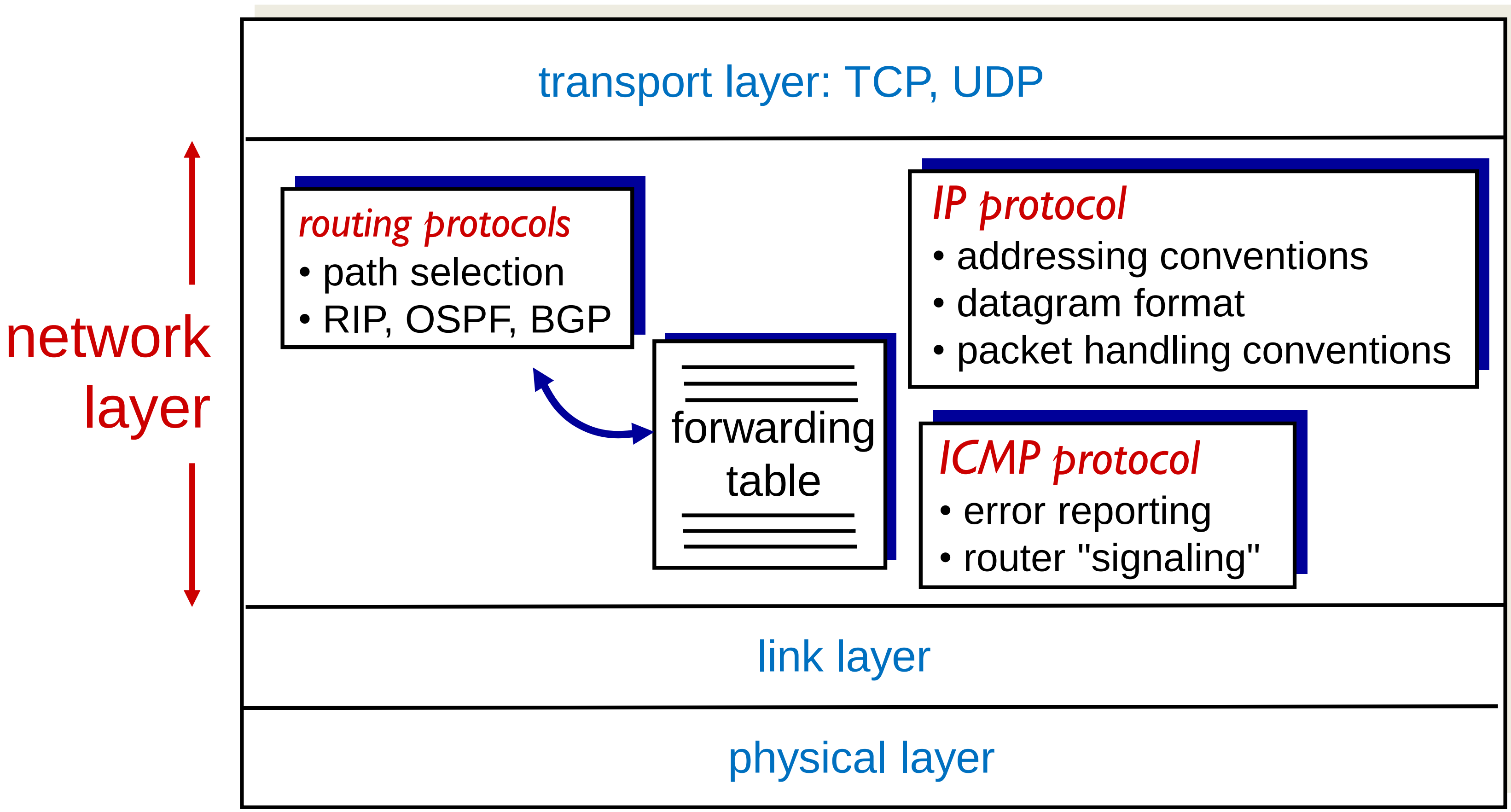
IP networks

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets forwarded using destination host address



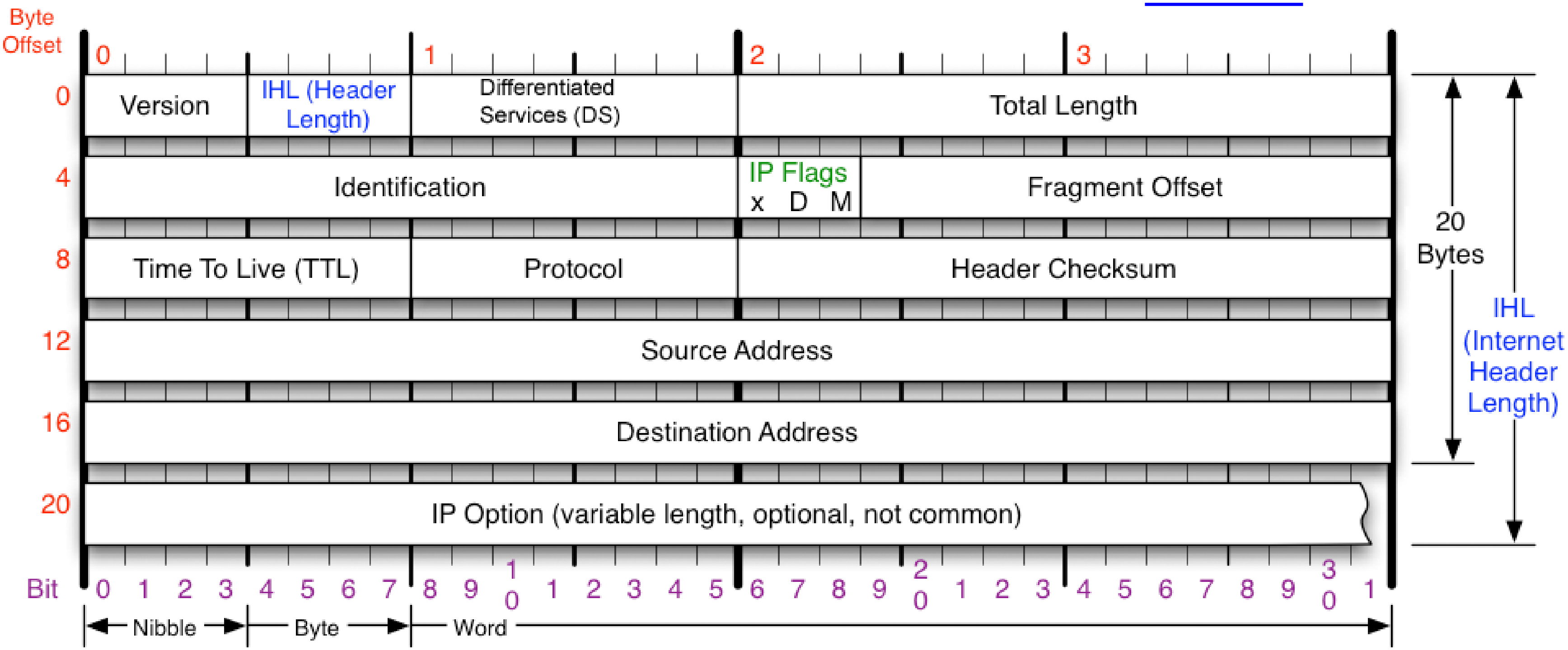
The Internet network layer

host, router network layer functions:



IPv4 Header

RFC 791



- **Version:** 4 or 6
- **DiffServ** (DS): recently changed to match IPv6
- **Total Length:** 16-bit, header + data,
  min 20 bytes, max 65,535 bytes

Read in Wikipedia

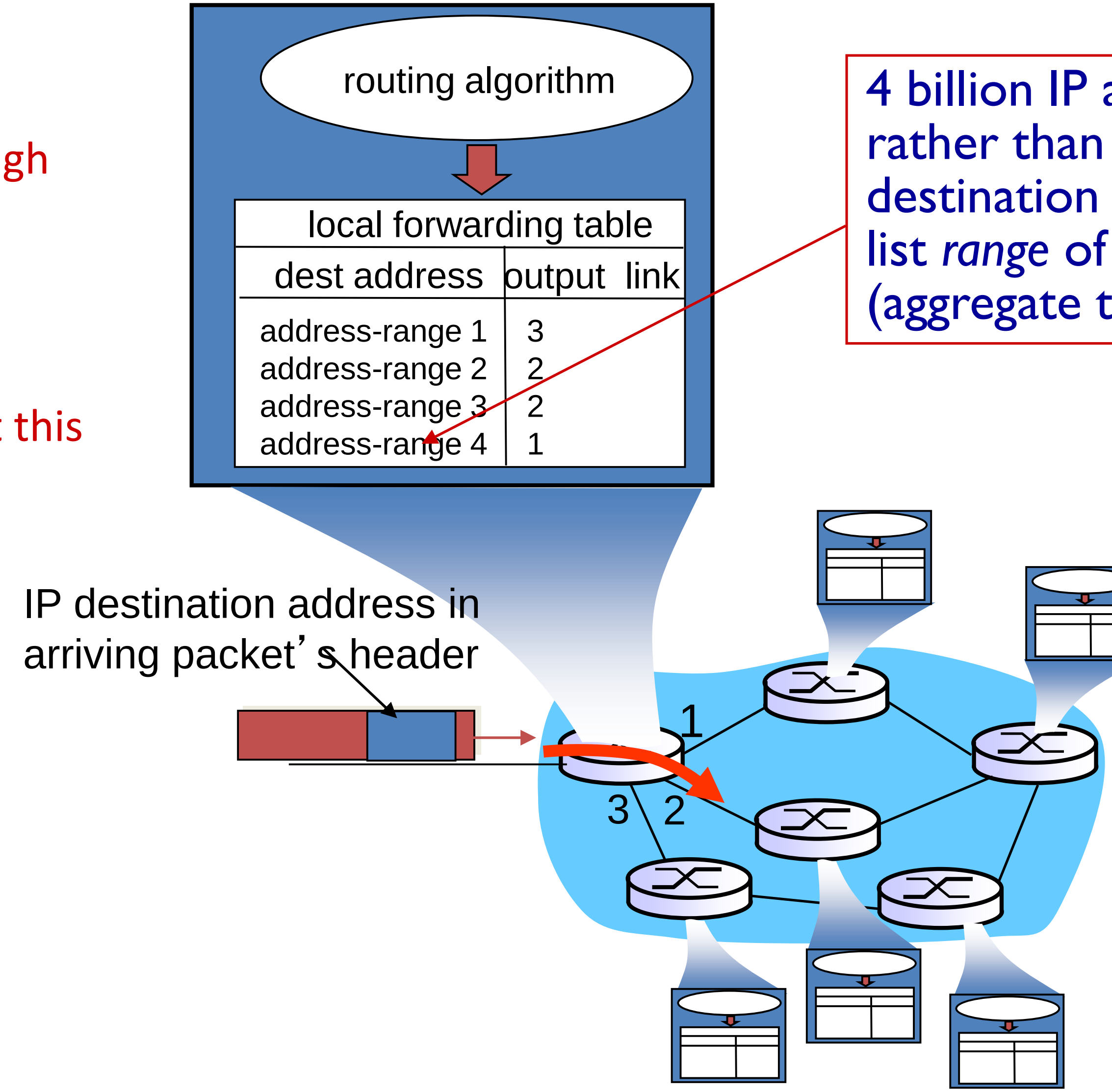
- **Identification**
- **IP Flags: x, D, M**
- **Fragment Offset**
- **Protocol** used in the data portion (TCP, UDP,)

- **Header Checksum**
- **Time to live (TTL)** aka **Hop limit:** maximum number of routers that the packet can be sent through.
- **Source and Destination Addresses:** 32-bit (4-byte) IP addresses that identify the sender and receiver of the packet

Datagram forwarding table

routing algorithm determines end-end-path through network

forwarding table determines local forwarding at this router



4 billion IP addresses, so rather than list individual destination address list *range* of addresses (aggregate table entries)

Internet Addresses

- Managed by ICANN
  - Internet Corporation for Assigned Names and Numbers
  - Manages the assignment of both IP and application layer name space (domain names)
    - Both assigned at the same time and in groups
    - Manages some domains directly (e.g., .com, .org, .net) and
    - Authorizes private companies to become domain name **registrars** as well
- Example: Indiana University
  - URLs that end in .indiana.edu and iu.edu
  - IPv4 addresses in the 129.79.x.x range (where x is any number between 0 and 255)
- .monash.edu has IPv4 in the 130.194.x.x range

IPv4 Addresses

- 4 byte (32 bit) addresses
  - Strings of 32 binary bits
- Dotted decimal notation
  - Used to make IP addresses easier to understand for human readers
  - Breaks the address into four bytes and writes the digital equivalent for each byte
- Example: 128.192.56.1

1 0 0 0 0 0 0 0	**1 1 0 0 0 0 0 0**	**0 0 1 1 1 0 0 0**	**0 0 0 0 0 0 0 1**

Classfull (old) and Classless (CIDR) Addressing

- In the old days of the Internet, addresses used to be assigned by class.
- A class A address was one for which the organization received a fixed first byte and could allocate the remaining three bytes.
- A class B address has the first two bytes fixed, and the organization can assign the remaining two bytes.
- A class C address has the first three bytes fixed with the organization able to assign the last byte, which provides about 250 addresses.
- Addresses are no longer assigned in this way and most network vendors are no longer using the terminology.
- The newer terminology **(CIDR – Classless InterDomain Routing)** is classless addressing in which a slash is used to indicate the address range (it is also called **slash notation**).
- For example **128.192.59.0/24** means the first 24 bits (3 bytes) are fixed, and the organization can allocate the last byte (8 bits).

IP addressing: CIDR

CIDR: Classless InterDomain Routing

- **subnet** portion of address of arbitrary length
- address format: a.b.c.d/x, where x is the number of bits in subnet portion of address

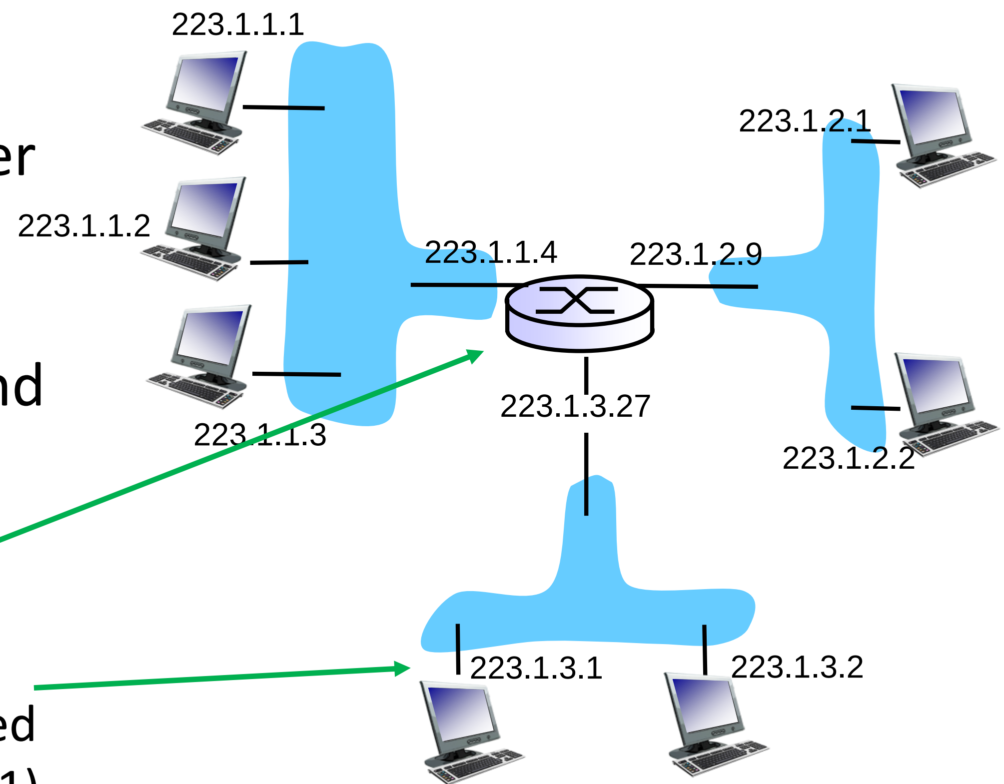
subnet part | host part

11001000 00010111 00010000 00000000

200.23.16.0/23

IPv4 addressing: more comments

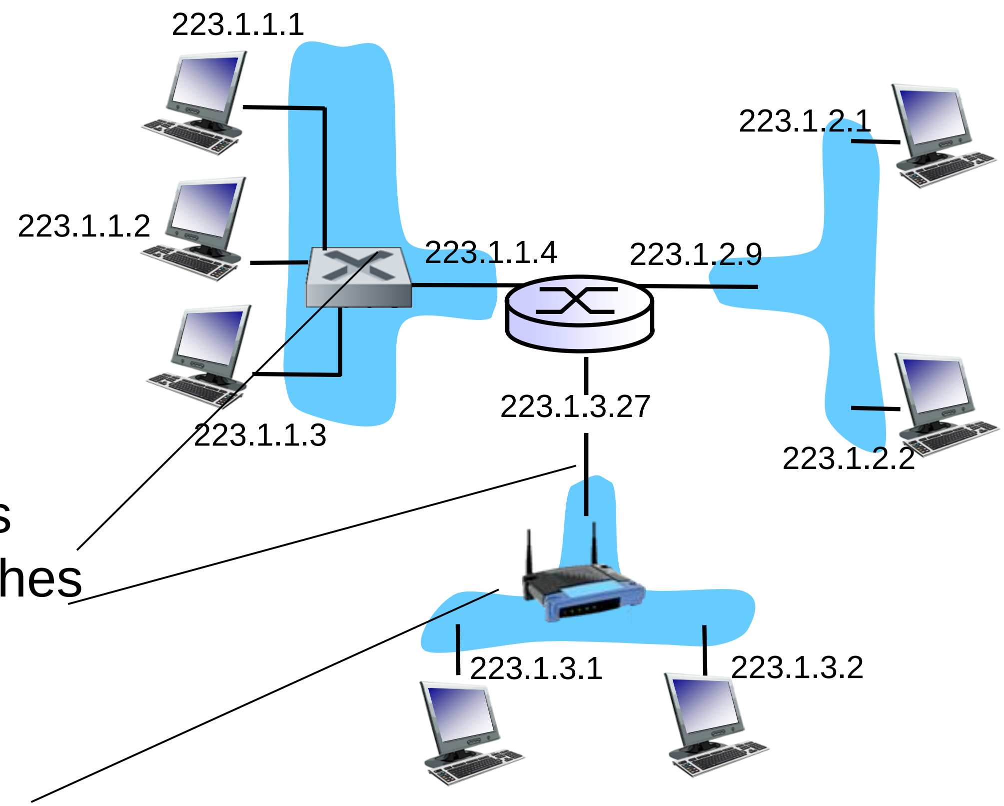
- *IP address:* 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - Routers typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
- ***IP addresses associated with each interface***



$$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$$

IP addressing: introduction

Q: how are interfaces actually connected?



A: **wired Ethernet** interfaces connected by Ethernet switches

A: **wireless WiFi** interfaces connected by WiFi access point and an Ethernet switch

Subnets

What is a subnet ?

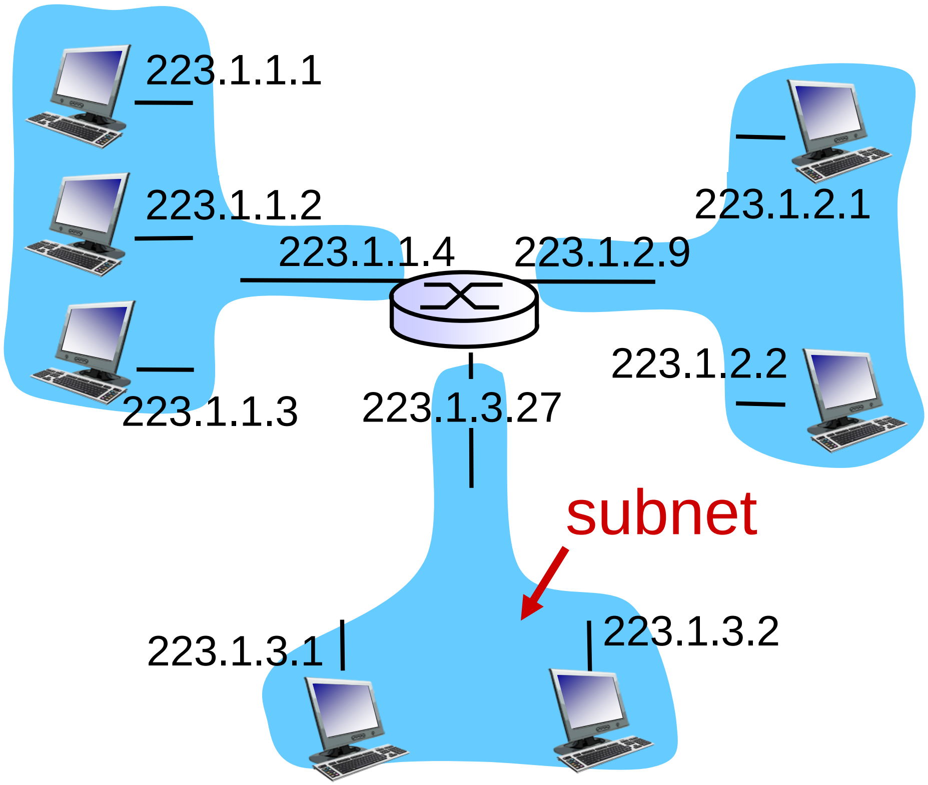
– A group of computers sharing the more significant part of their IP addresses

IP address in the subnet

– **subnet part** – more significant bits

– **host part** - less significant bits

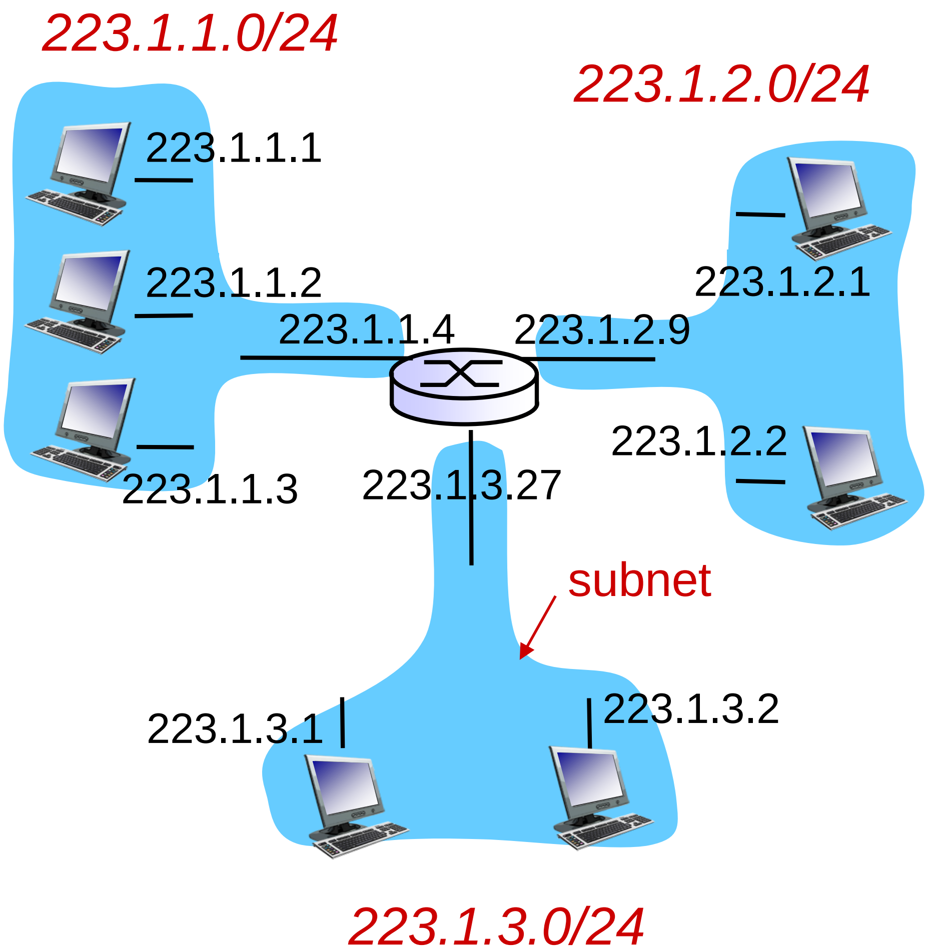
- Most **typically**, computers connected to an **Ethernet switch** form a subnet aka **LAN**



network consisting of 3 subnets

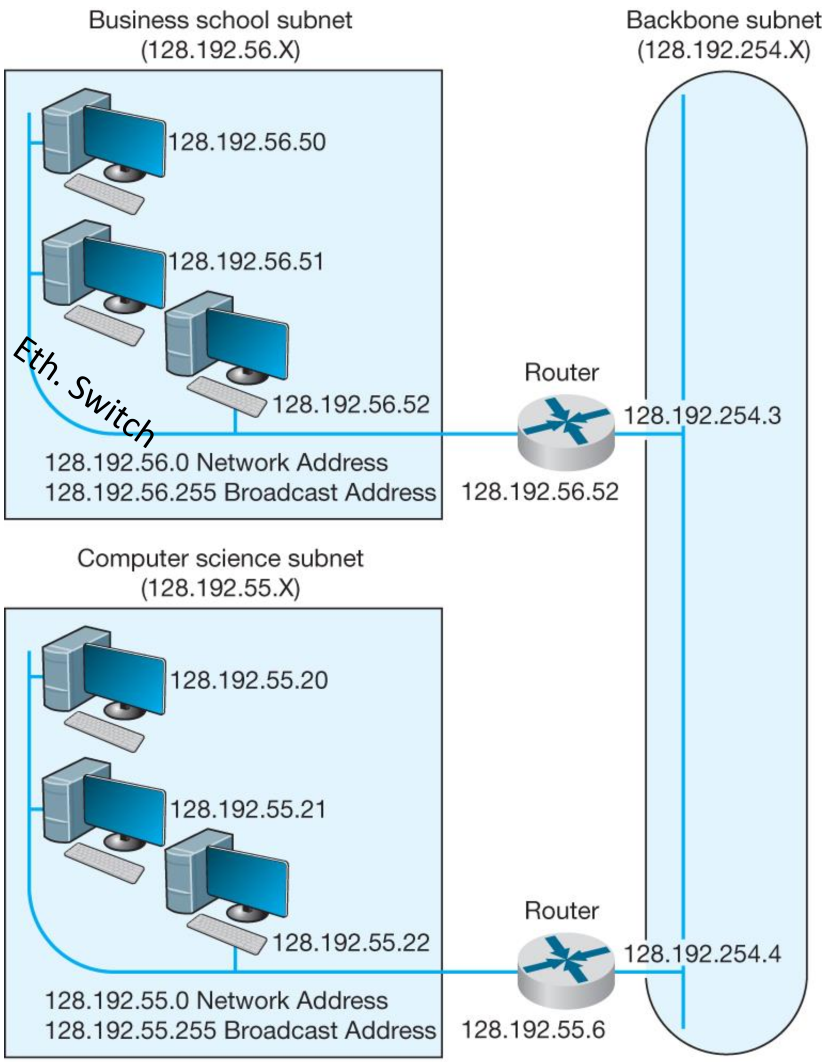
Subnets

- Note three subnets specified by 24 most significant bits of the IP addresses:
  - *223.1.1.0/24*
  - *223.1.2.0/24*
  - *223.1.3.0/24*
- Remaining 8 bits specify the host (computer), e.g. 223.1.**3**.1 (subnet **3**, computer **1**)
- Subnet mask: 255.255.255.0



subnet mask: /24

Subnets: Another Example



- Note three subnets:
  - Business school subnet (128.192.56.x)
  - Computer science subnet (128.192.55.x)
  - **Backbone subnet** (128.192.254.x)
- Note that routers have two IP addresses
- Computers in the subnet (LAN) are **responsible for routing** to computers in the same subnet

Subnet Masks

- Used to make it easier to separate the subnet part of the address from the host part.
- Used to **identify computers belonging to the same subnet**
- Example
  - Subnet: 149.61.10.x
  - Subnet mask: 255.255.255.000, or /24, or in binary

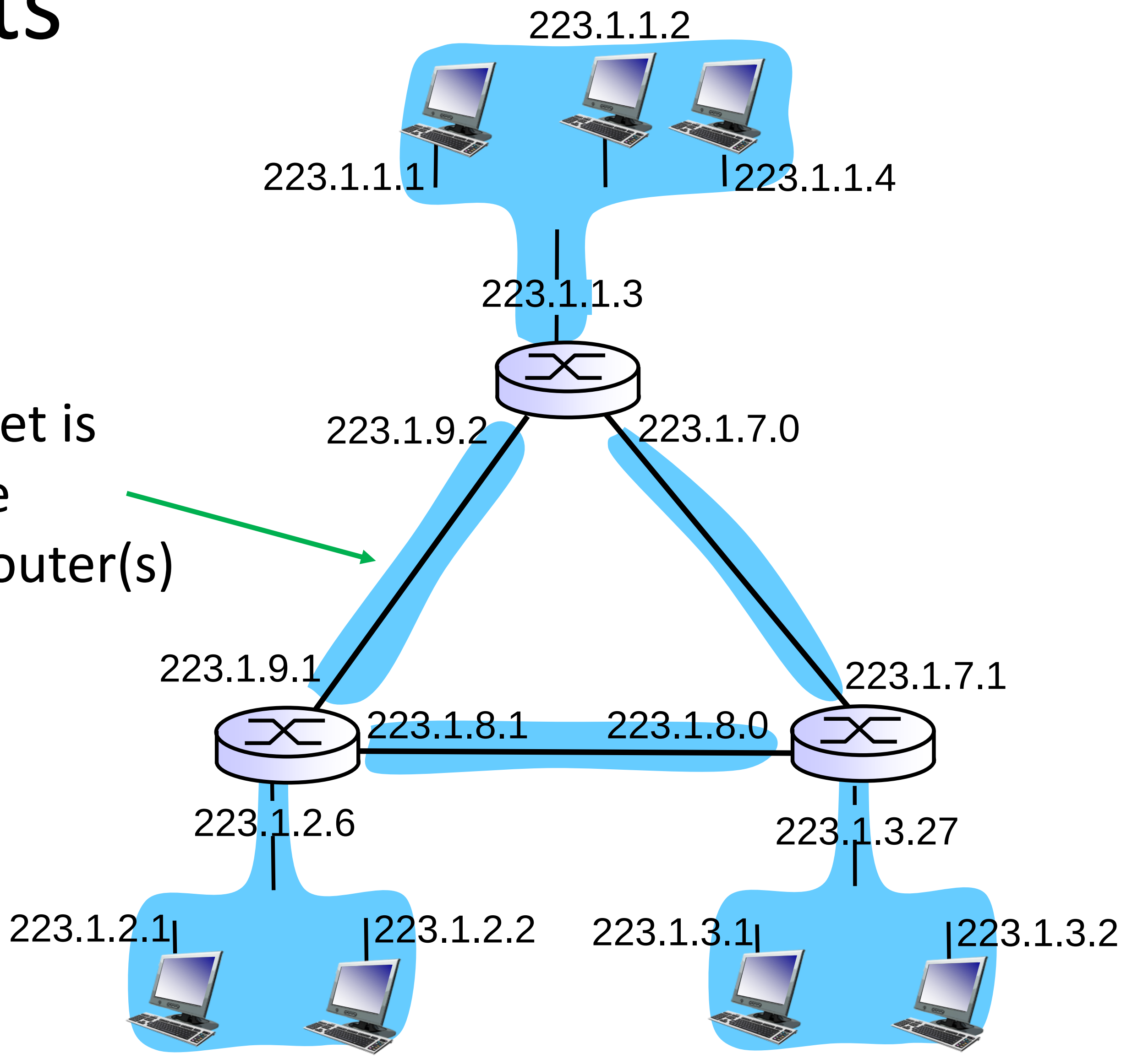
    11111111.11111111.11111111.00000000
- Example
  - Subnets: 149.61.10.0-63, (6 bits for the subnet) /26
  - Subnet mask 255.255.255.192 or, in binary:

    11111111.11111111.11111111.11000000 /26

Subnets

how many?

Note that the subnet is associated with the **interface** not the router(s)
223.1.9.0/24



Special-use IP addresses

Range	Description	Reference
0.0.0.0/8	Current network (only valid as source address)	RFC 6890
10.0.0.0/8	Private network	RFC 1918
100.64.0.0/10	Shared Address Space	RFC 6598
127.0.0.0/8	Loopback	RFC 6890
169.254.0.0/16	Link-local	RFC 3927
172.16.0.0/12	Private network	RFC 1918
192.0.0.0/24	IETF Protocol Assignments	RFC 6890
192.0.2.0/24	TEST-NET-1, documentation and examples	RFC 5737
192.88.99.0/24	IPv6 to IPv4 relay	RFC 3068
192.168.0.0/16	Private network	RFC 1918
198.18.0.0/15	Network benchmark tests	RFC 2544
198.51.100.0/24	TEST-NET-2, documentation and examples	RFC 5737
203.0.113.0/24	TEST-NET-3, documentation and examples	RFC 5737
224.0.0.0/4	IP multicast (former Class D network)	RFC 5771
240.0.0.0/4	Reserved (former Class E network)	RFC 1700
255.255.255.255	Broadcast	RFC 919

Private IPv4 addresses

Private networks addresses, three rangers:

10.0.0.0/8 , 172.16.0.0/12 , 192.168.0.0/16

- These ranges are not routable outside of private networks,
- Private machines cannot directly communicate with public networks.
- They can do so through network address translation (discussed later)
- Typically used in small LANs/subnet not hosting servers.

Link-Local addresses

- 169.254.0.0/16 are valid on links (such as a local network segment or point-to-point connection) connected to a host.
- These addresses are not routable.
- Like private addresses, these addresses cannot be the source or destination of packets traversing the internet.
- These addresses are primarily used for address autoconfiguration when a host cannot obtain an IP address from a DHCP server or other internal configuration methods.
- We used 169.254… addresses when setting up the **wireless ad-hoc networks**
- Microsoft created an implementation called Automatic Private IP Addressing (APIPA), which was deployed on millions of machines and became a de facto standard.

Examples

```
Wireless LAN adapter Wi-Fi: (at the R&D Apartments)
Connection-specific DNS Suffix . :
Description . . . . . . . . . . . : Intel(R) Dual Band Wireless-AC 7260
Physical Address. . . . . . . . . : 5C-51-4F-ED-39-FB
IPv4 Address. . . . . . . . . . . : 192.168.100.9(Preferred)
Default Gateway . . . . . . . . . : 192.168.100.1
DHCP Server . . . . . . . . . . . : 192.168.100.1
DNS Servers . . . . . . . . . . . : 192.168.100.1 ; 198.41.0.4
```

Wireless ad-hoc network:

Node 1:

Interface: 169.254.87.51 --- 0xd

```
Internet Address Physical Address Type
    169.254.10.113  3c-77-e6-18-fa-0b dynamic
    169.254.255.255 ff-ff-ff-ff-ff-ff static
    224.0.0.22      01-00-5e-00-00-16 static
    224.0.0.252     01-00-5e-00-00-fc static
    239.255.255.250 01-00-5e-7f-ff-fa static
    255.255.255.255 ff-ff-ff-ff-ff-ff static
```

DHCP: Dynamic Host Configuration Protocol

goal: allow a host to *dynamically* obtain its IP address from the network DHCP server when it joins network

– can renew its lease on address in use

– allows reuse of addresses (only hold address while connected)

– support for mobile users who want to join network

DHCP overview: RFC 2131

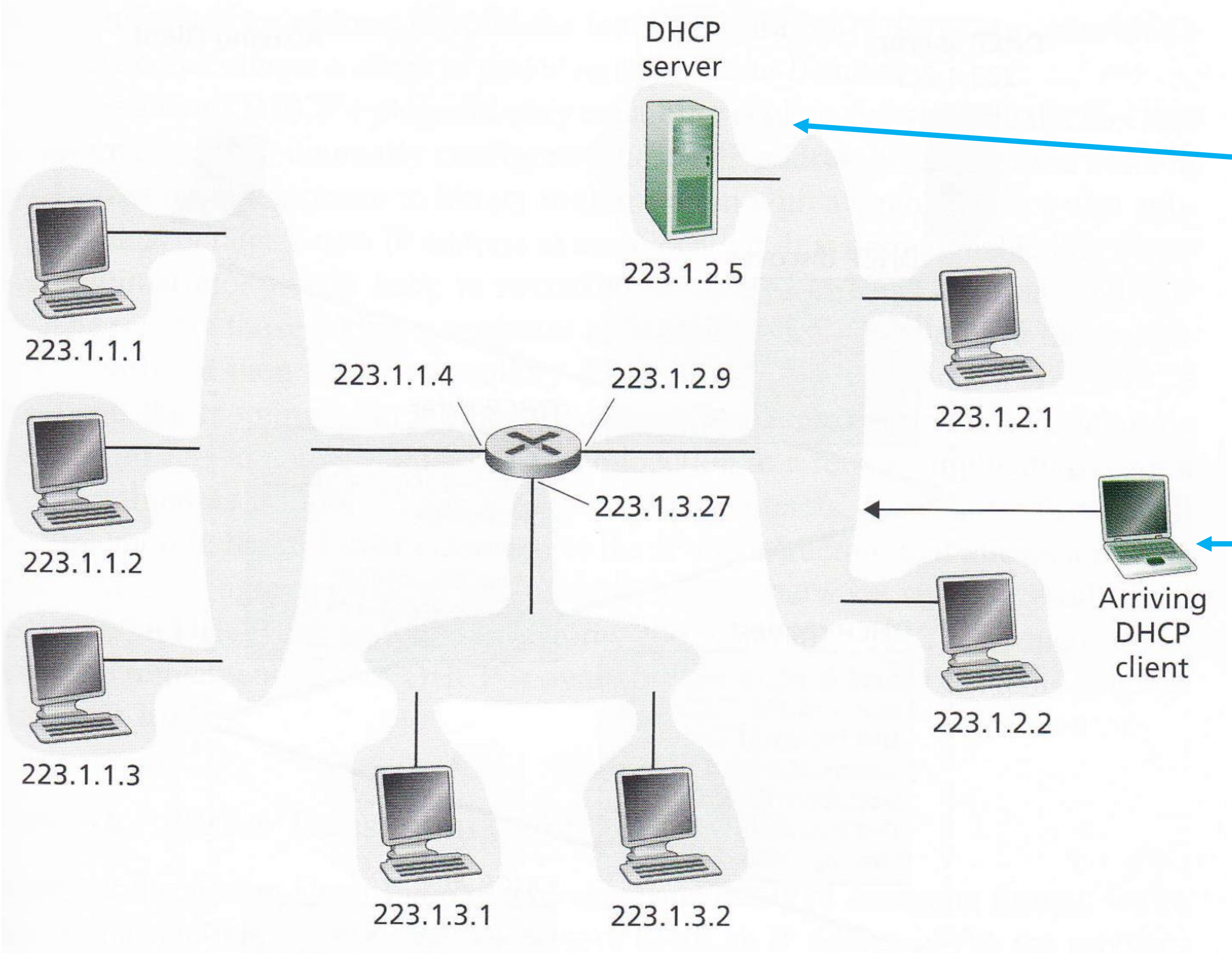
– host broadcasts "DHCP discover" msg [optional]

– DHCP server responds with "DHCP offer" msg [optional]

– host requests IP address: "DHCP request" msg

– DHCP server sends address: "DHCP ack" msg

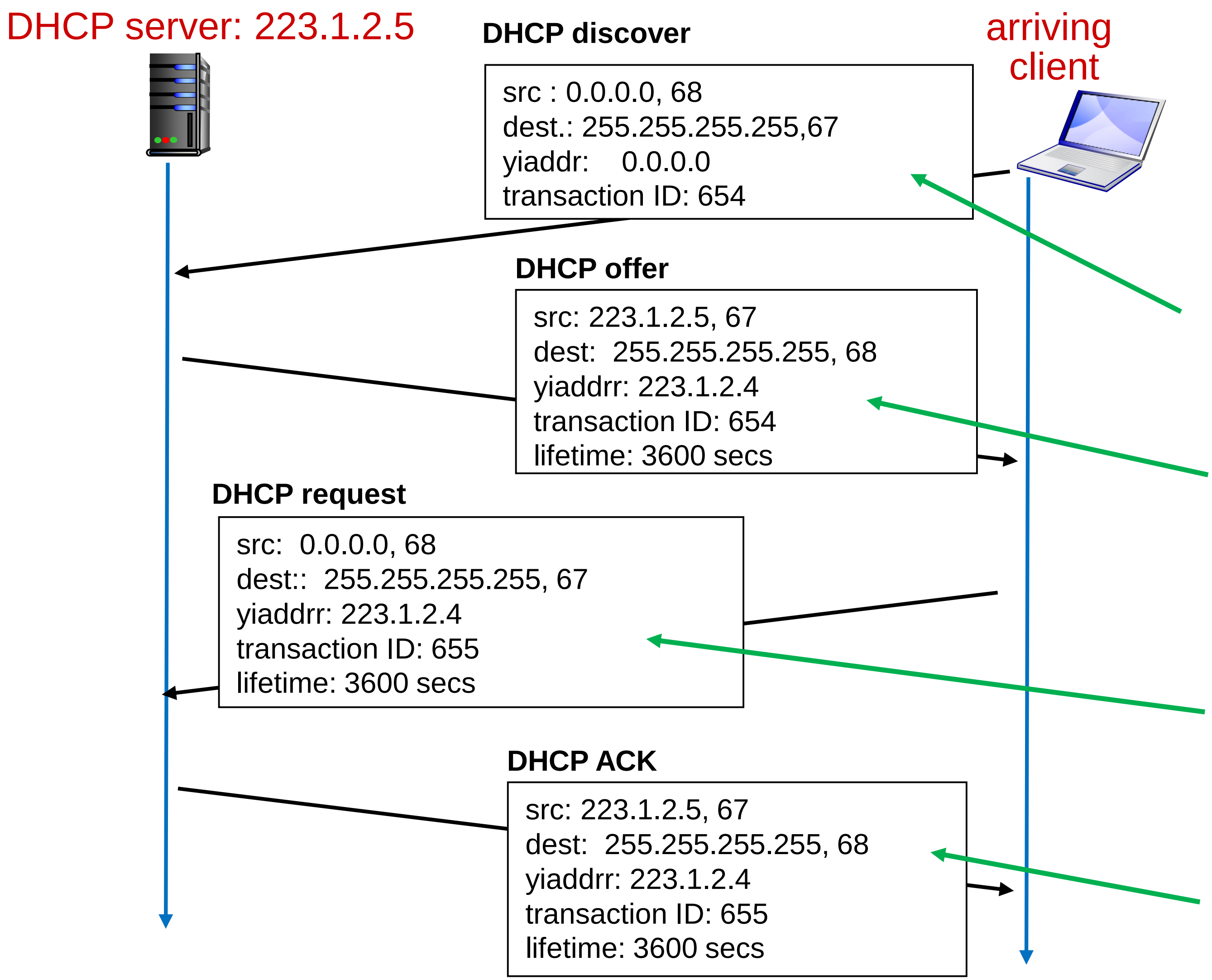
DHCP example



Note

- DHCP server Typically build into a router/ switch
- Arriving client

DHCP client-server scenario



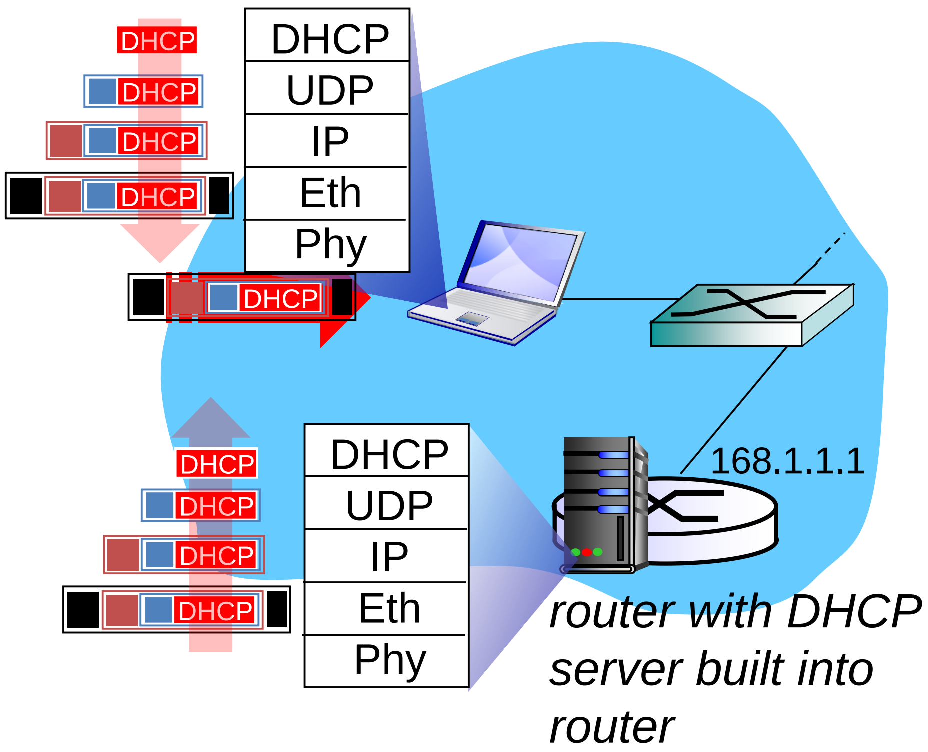
The arriving client does not know any address. The UDP **discovery** msg is sent to:
All ones, port 67

DHCP **offer**s 223.1.2.4

The client agrees to the offer sending DHCP **request**

DHCP sends **ACK**

DHCP: more than IP address



- connecting laptop/client needs its IP address, **addr** of first-hop router, **addr** of DNS server:
- Sends the DHCP **discovery request** broadcast on LAN, received at router running DHCP server
- After **DHCP offer** and **DHCP request** DHCP server formulates DHCP ACK received by the client containing , in general:

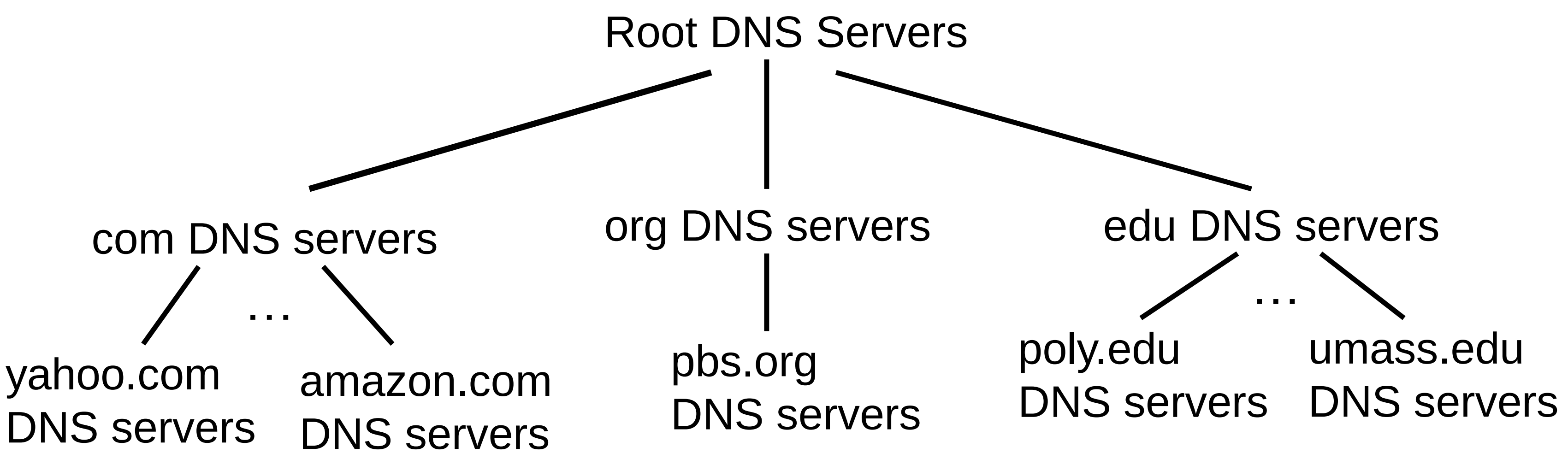
- Client's new IP address
- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

Address Resolution

1. Server Name Resolution
   - Translating destination **host's domain name** to its corresponding **IP address**
   - www.yahoo.com is resolved to → 203.84.216.121
   - Uses one or more Domain Name Service (DNS) **servers** to resolve the address
   - Corresponding addresses are maintained in the **address table(s)**
2. Data Link Layer Address Resolution
   - Identifying the MAC address of the next node (that packet must be forwarded to)
   - Uses Address Resolution Protocol (ARP)

DNS: a distributed, hierarchical database

RFC 1034 Domain Names - Concepts and Facilities
RFC 1035 Domain Names - Implementation and Specification. Plus many updates



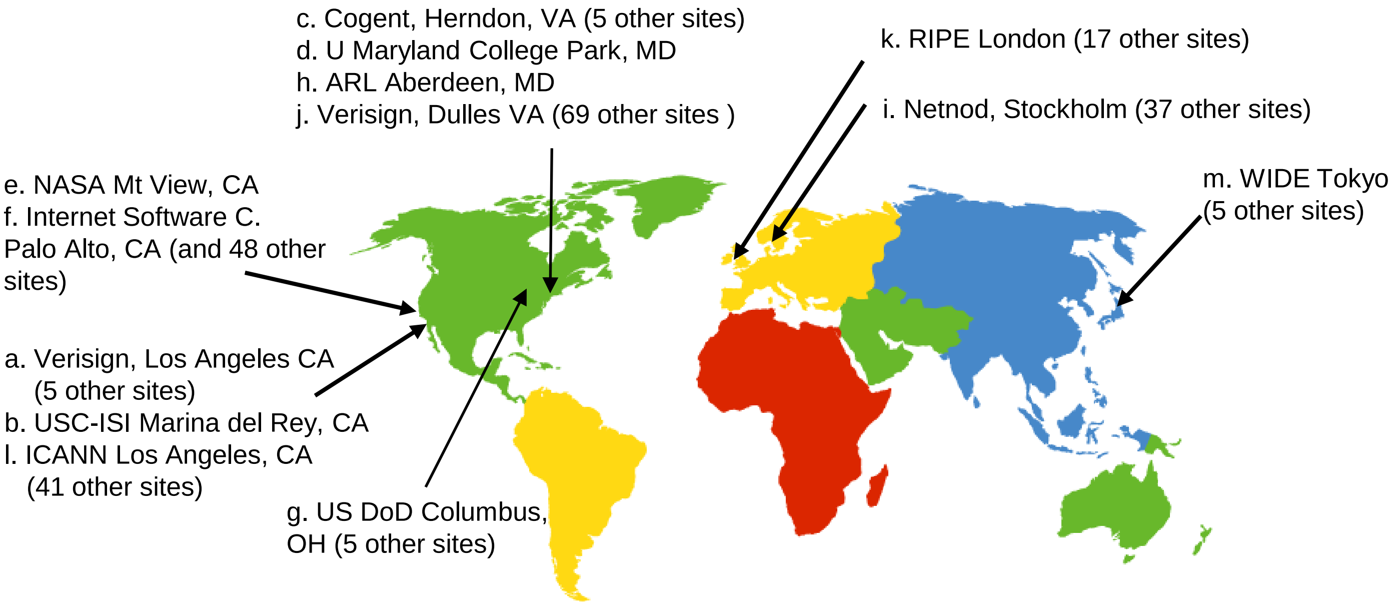
*client wants IP for **www.amazon.com**; 1st approx:*

- client queries **root server** to find **.com** DNS server
- client queries .com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get **IP address** for www.amazon.com

DNS: root name servers

- contacted by local name server that can not resolve name
- root name server:
  - contacts **authoritative name server** if name mapping not known
  - gets mapping
  - returns mapping to local name server

13 root name "servers" worldwide



Compare with:
Root Name Servers

TLD, authoritative servers

top-level domain (TLD) servers:

- responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g.: cn, uk, fr, ca, jp
- **Network Solutions** maintains servers for .com TLD
- **Educause** for .edu TLD

authoritative DNS servers:

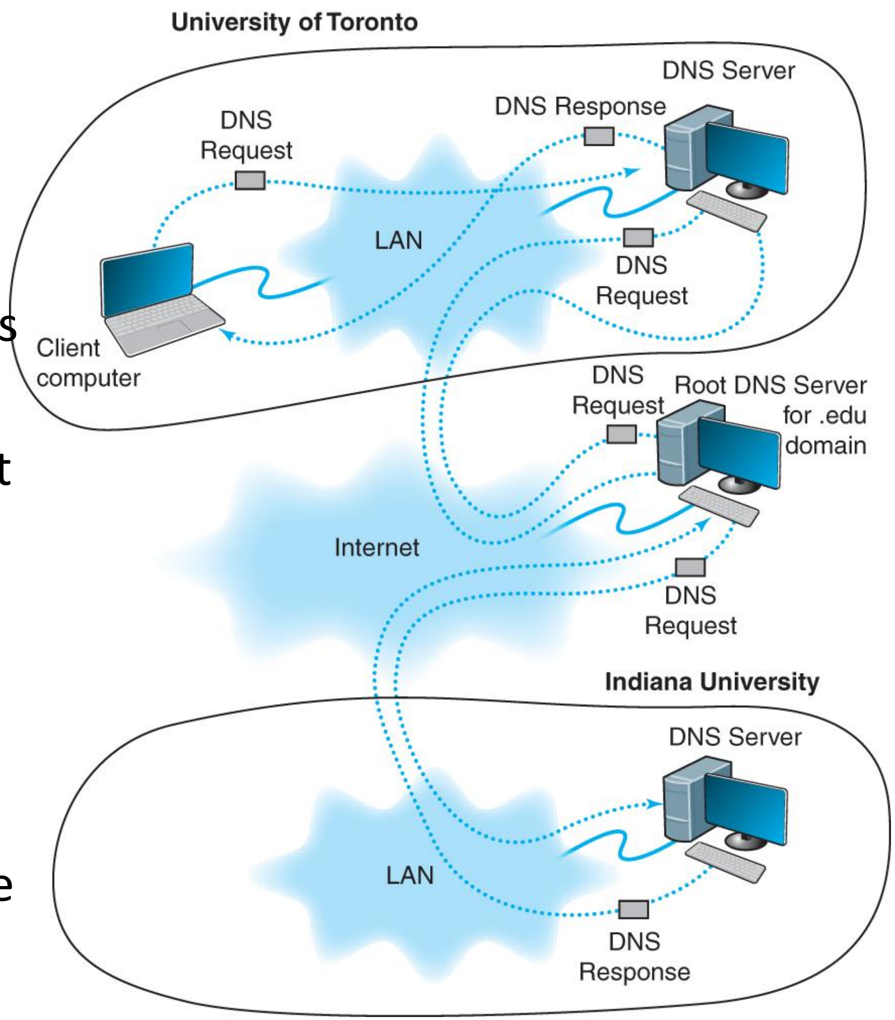
- Organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
- can be maintained by organization or service provider

Local DNS name server

- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one
  - also called "default name server"
- when host makes DNS query, query is sent to its **local DNS server**
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts as proxy, forwards query into hierarchy

How DNS Works. Simplified Example

1. A client at Toronto asks for a web page on Indiana University's server
2. URL => IP not in his address table
3. Sends DNS request to the **local DNS** server
4. URL => IP not available
5. Local server sends DNS request to the **.edu root server**



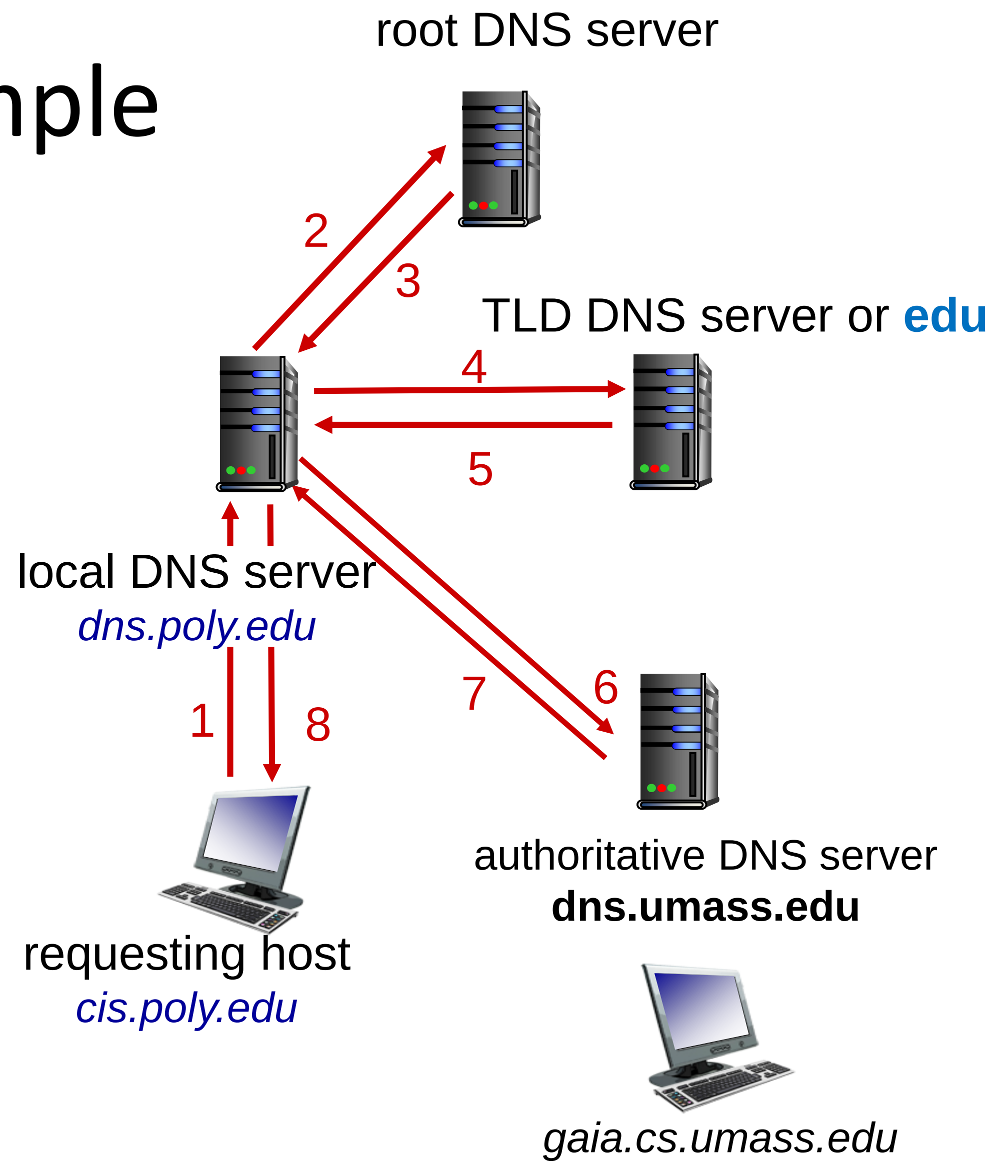
5. Root DNS server knows the IP of the Indiana DNS servers and asks for the IP for the requested URL
6. Indiana DNS server sends the IP back to the .edu root server
7. The .edu root sends the IP back to the **local DNS** server
8. Local DNS server sends the IP back to the client computer

DNS name resolution example

- host at **cis.poly.edu** wants IP address for **gaia.cs.umass.edu**

iterated query:

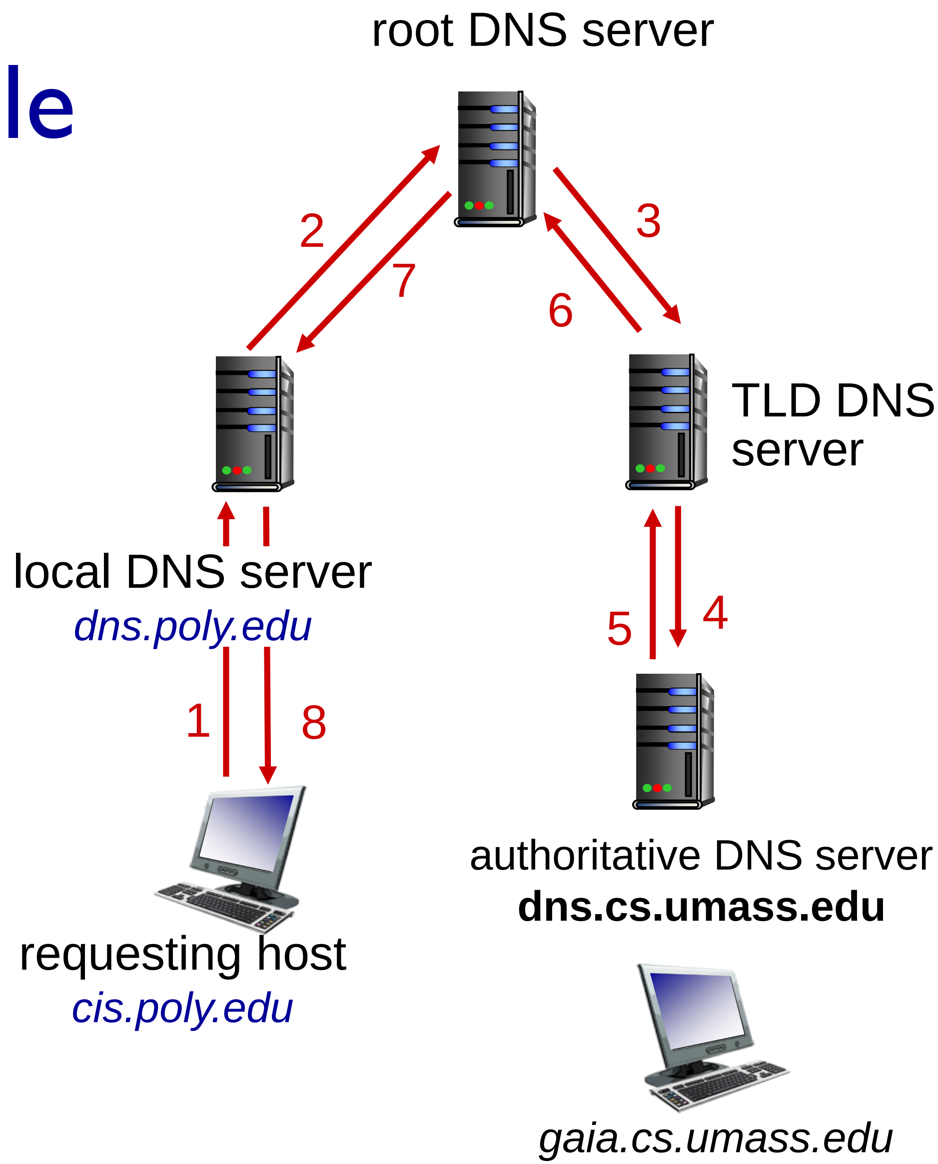
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS name resolution example

recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



DNS: caching, updating records

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time (TTL)
  - TLD servers typically cached in local name servers
    - thus root name servers not often visited
- cached entries may be *out-of-date* (best effort name-to-address translation!)
  - if name host changes IP address, may not be known Internet-wide until all TTLs expire
- update/notify mechanisms proposed in IETF standard

  RFC 2136: Dynamic Updates in the Domain Name System (DNS UPDATE)

DNS records

DNS: distributed db storing resource records (RR)

RR format: **`(name, value, type, ttl)`**

type=A

- **`name`** is hostname
- **`value`** is IP address

type=NS

- **`name`** is domain (e.g., foo.com)
- **`value`** is hostname of authoritative name server for this domain

type=CNAME

- **`name`** is alias name for some "canonical" (the real) name
- **`www.ibm.com`** is really **`servereast.backup2.ibm.com`**
- **`value`** is canonical name

type=MX

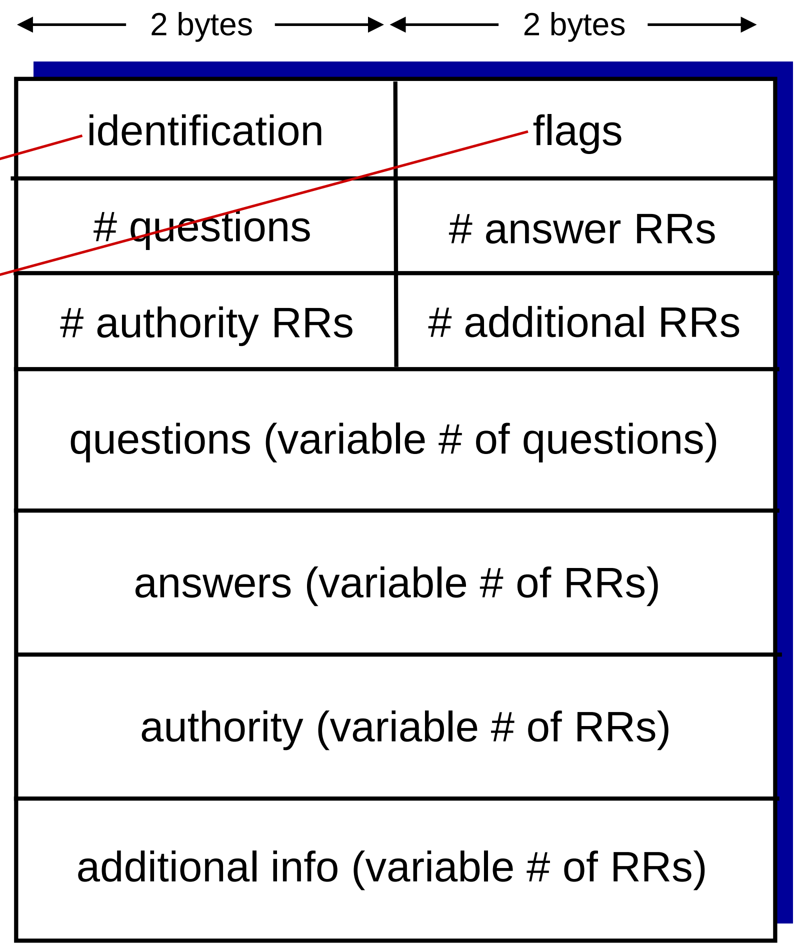
- **`value`** is name of mailserver associated with **`name`**

DNS protocol, messages

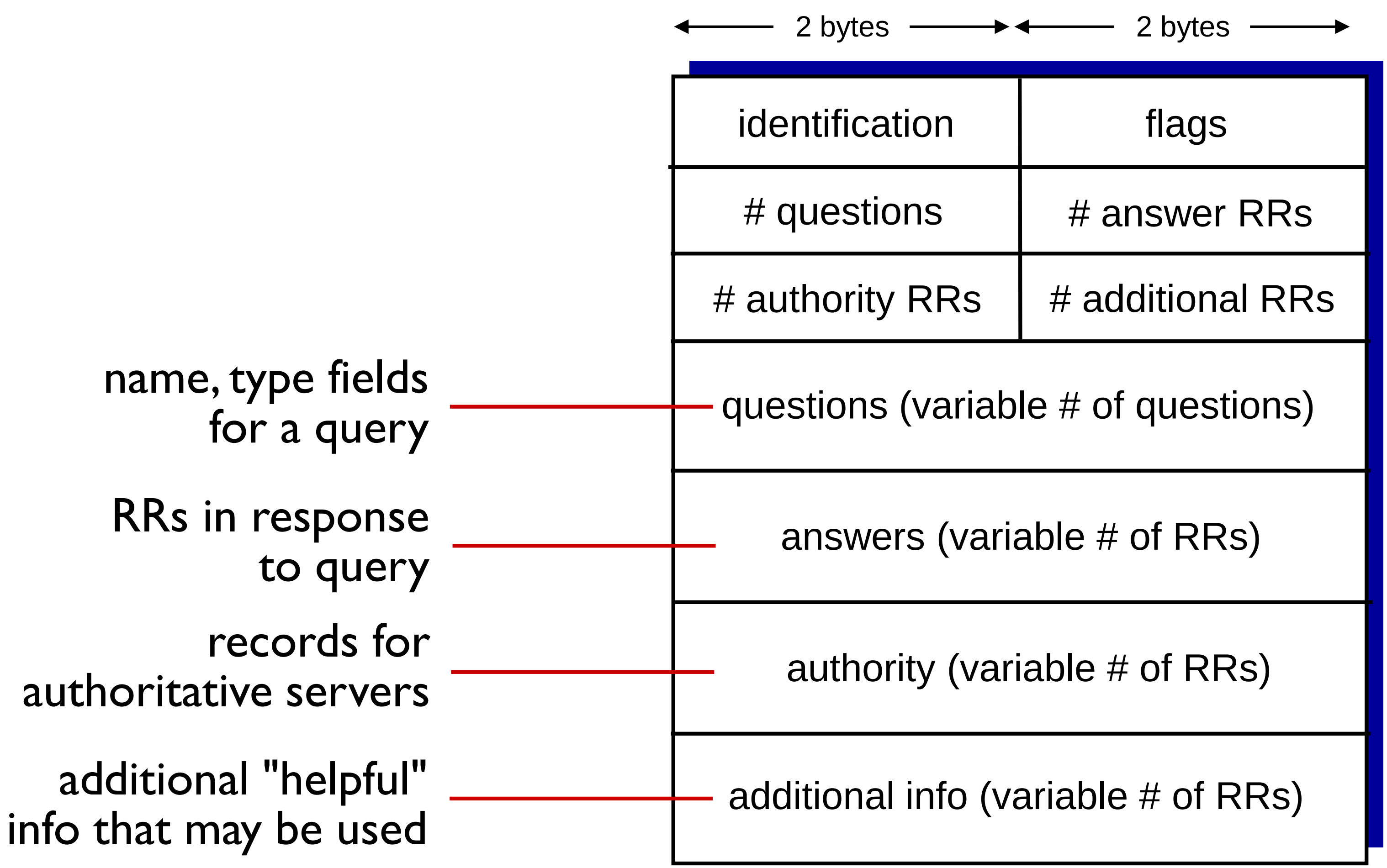
- *query* and *reply* messages, both with same *message format*

msg header

- identification: 16 bit # for query, reply to query uses the same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative



DNS protocol, messages



Inserting records into DNS

- example: new startup "Network Utopia"
- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts two RRs into .com TLD server:
    `(networkutopia.com, dns1.networkutopia.com, NS)`
    `(dns1.networkutopia.com, 212.212.212.1, A)`
- create authoritative server type A record for www.networkuptopia.com; type MX record for networkutopia.com