ΕΠΛ 602:Foundations of Internet Technologies

Internet Protocols (IP, TCP, UDP), DNS,

Lecture Outline

- Main points of Lecture 3
- The Internet Protocols: IP, TCP, UDP
- DNS (name services)

Main Points

Networks

Network: a set of devices connected through communication links

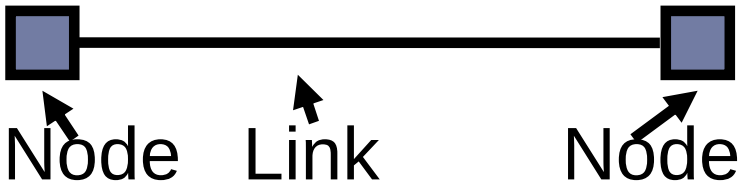
Communication Subsystem: hardware and software components that provide the communication facilities for the distributed system

- Transmission media (wire, fibre, wireless channels)
- Hardware devices (routers, switches, bridges, hubs, repeaters, network interfaces)
- Software (protocols stacks, communication handlers and drivers)

Subnet is a collection of nodes that may all be reached on the same physical network

Internet – internetworking among subnets

Simple Network



- Node: computer
  - End host: general-purpose computer, cell phone, PDA
  - Network node: switch or router
- Link: physical medium connecting nodes
  - Twisted pair: the wire that connects to telephones
  - Coaxial cable: the wire that connects to TV sets
  - Optical fiber: high-bandwidth long-distance links
  - Space: propagation of radio waves, microwaves, …

Types of networks

- **Personal Area Networks (PANs)**

connect various digital devices of a user by a low-cost, low-energy network

WPANs

Bluetooth, infra-red for PDAs (10-30m)

- **Local Area Networks (LANs)**

Floor/building wide: computers connected by a single communication medium segments connected by switches or hubs - Ethernet (broadcast)

WLANs

within buildings, several variants of IEEE 802.11 (WiFi)

- **Wide Area Networks (WANs)**

world-wide, different organizations: set of communication circuits linking a set of dedicated computers (routers)

WWANs

Mobile phones based on GSM wide area connection to the Internet

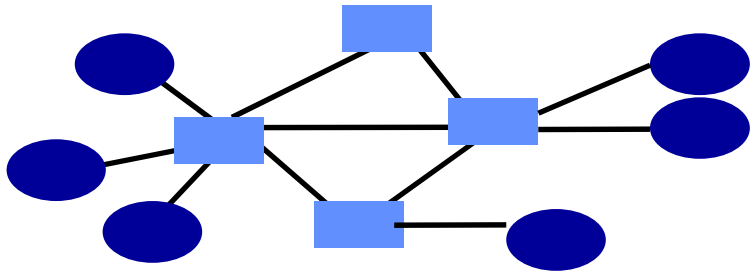
- **Metropolitan Area Networks (MANs)**

City-wide, distances up to 50 kms: *DSL* uses ATM switches located in telephone exchanges to route digital data to the subscriber over twisted pairs of & *Cable modem* analogue signaling on cable television networks up to 1.5 Mbps

Types of networks

	Example	*Range*	*Bandwidth (Mbps)*	*Latency (ms)*
Wired:				
LAN	Ethernet	1–2 kms	10–10,000	1–10
WAN	IP routing	worldwide	0.010–600	100–500
MAN	ATM	2–50 kms	1–600	10
Internetwork	Internet	worldwide	0.5–600	100–500
Wireless:				
WPAN	Bluetooth (IEEE 802.15.1)	10–30m	0.5–2	5–20
WLAN	WiFi (IEEE 802.11)	0.15–1.5 km	11–108	5–20
WMAN	WiMAX (IEEE 802.16)	5–50 km	1.5–20	5–20
WWAN	3G phone	cell: 1–-5	348–14.4	100–500

Beyond Directly-Connected Networks



- Switched network
  - End hosts at the edge
  - Network nodes that switch traffic
  - Links between the nodes
- Multiplexing
  - Many end hosts communicate over the network
  - Traffic shares access to the same links

In a nutshell

Before a **message** (a logical unit of information) is transmitted is subdivided into **packets** (transmission unit) of restricted length

Simplest form: sequence of bits of restricted length + addressing information (source, destination)

Packet switching (store-and-forward)
Just forward packets from the source to their destination
At each switching node a computer:
o Packets are queued in a buffer and transmitted when the link is available

Communication is *asynchronous* – messages arrive at their destination after a delay that varies

Routing

Required in all networks but LAN

Routing Algorithm: decide which out-going link to forward the packet

- for circuit switching, the route is determined during the circuit setup time
- for packet switching, each packet is routed independently

Adaptive routing: the best route for communication between two points on the network is re-evaluated periodically

Routing algorithm:

- Determine the route for each packet
- Update its knowledge of the network (traffic monitoring and detection of configuration changes or failures)

Both distributed – based on local information, in a hop-by-hop basis

Routing Table

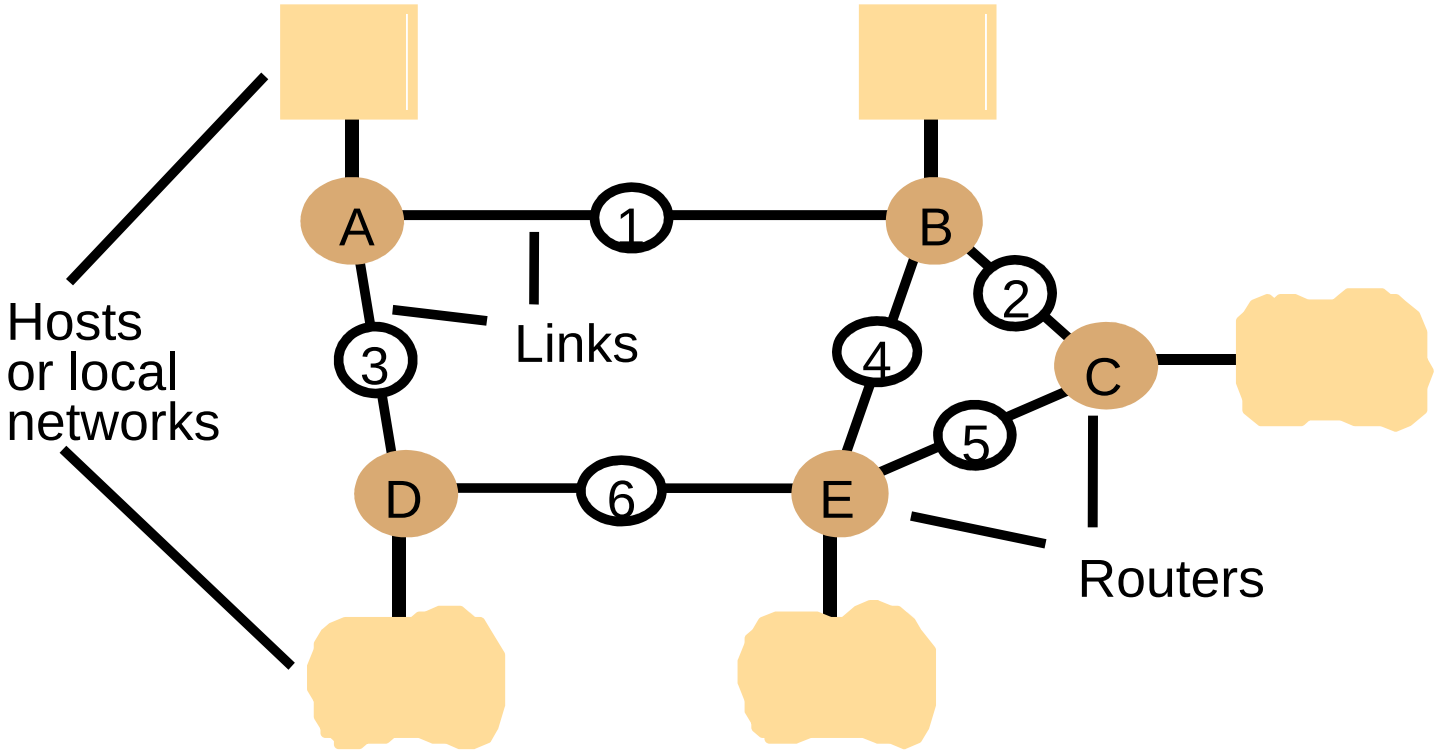
a record for each destination

fields: outgoing link, cost (e.g. hop count)

Path finding in graphs

Routing

- Router example



Routing

Router Information Protocol (RIP)

"Bellman-Ford distance vector" algorithm

- Sender: send table summary periodically (every 30sec in the Internet) or changes to neighbors
- Receiver:

  Consider *A* receives a table from *B*, *A* updates
  - *A* -> *B* -> … -> *X*:
    - *A* updates--*B* has more up-to-date (authoritative) info
  - *A* -> not *B* -> … -> *X*:
    - Does routing via *B* have a lower cost?
  - *B* -> … -> *X*:
    - *A* does not know *X*
  - *[B* -> *A* -> … -> *X]*:
    - *A* doesn't update--*A* has more up-to-date info
- Faulty link, cost is infinity

Sidenote
How to propagate updates?
❖ Push
❖ Pull (or on demand)
(+) many clients
(-) intrusive

Routing

RIP-1 (RFC 1058)

More recent algorithms

- more information, not just neighbors (e.g., actual bandwidth)
- link-state algorithms, each node responsible for finding the optimum route
  - A database at each node that represent all or a substantial portion of the network
  - Optimum routes to the destinations in the database

Protocols: Networks

A specification of:

1. The *sequence* of the messages to be exchanged
2. The *format* of the data in the messages

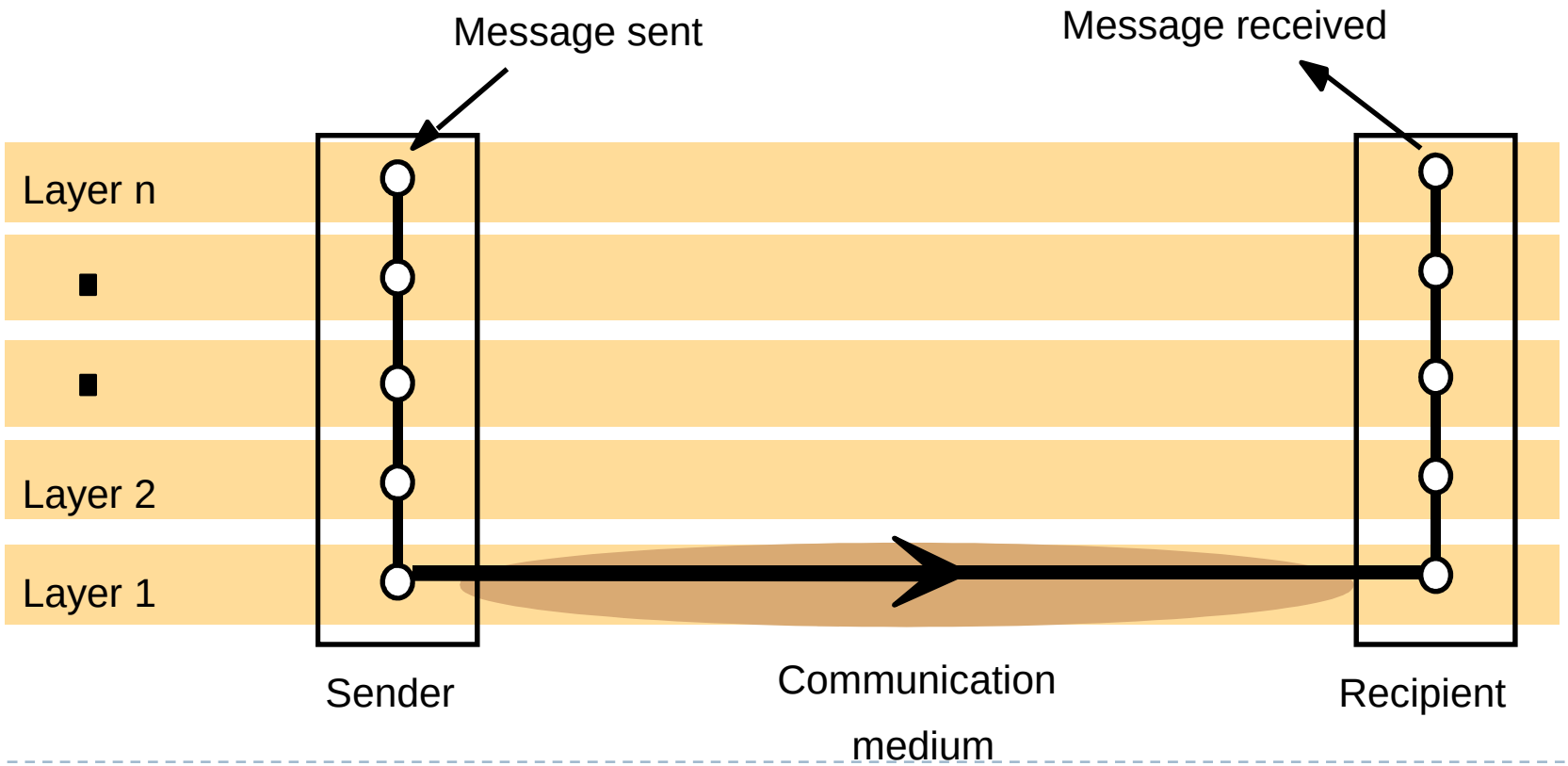
Implemented by a pair of software modules

Enables separate software components to be developed independently and implemented differently

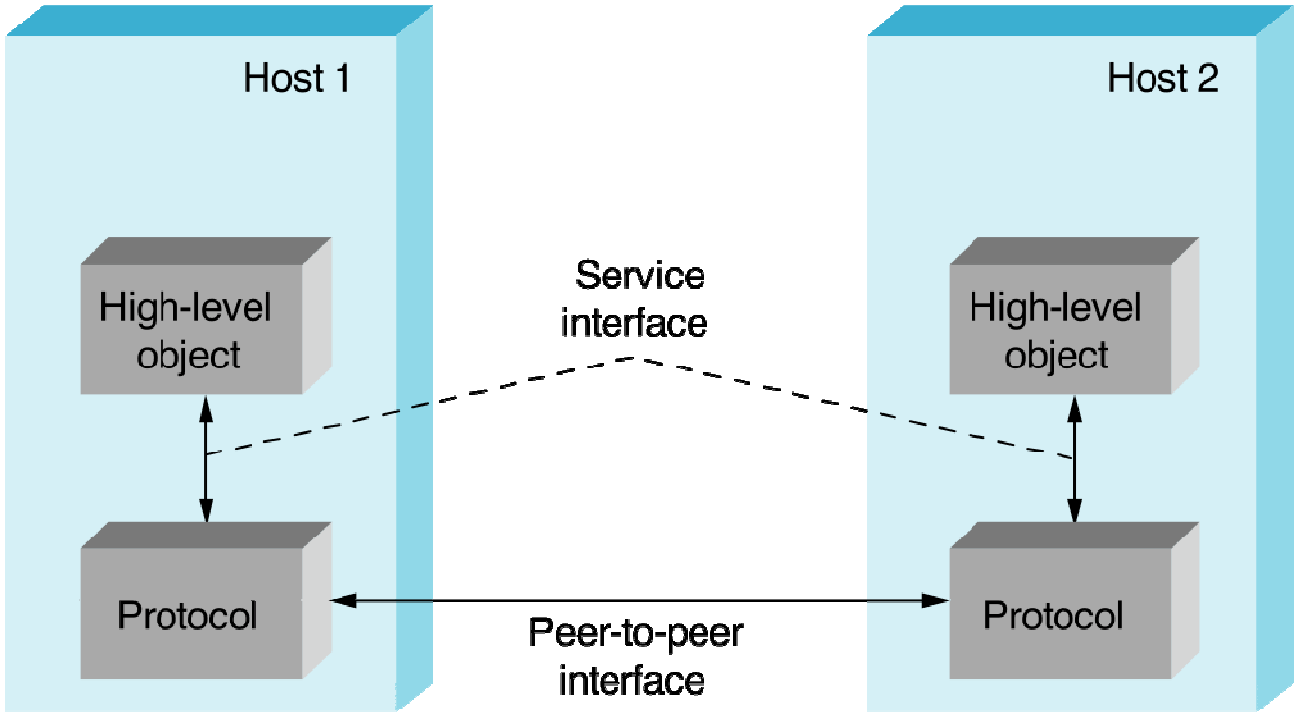
Protocols: Layers

Flow of data: each module appears to communicate directly with the module at the same level in the other computer (called peers)

Each layer by local procedure calls to the layers above and below it



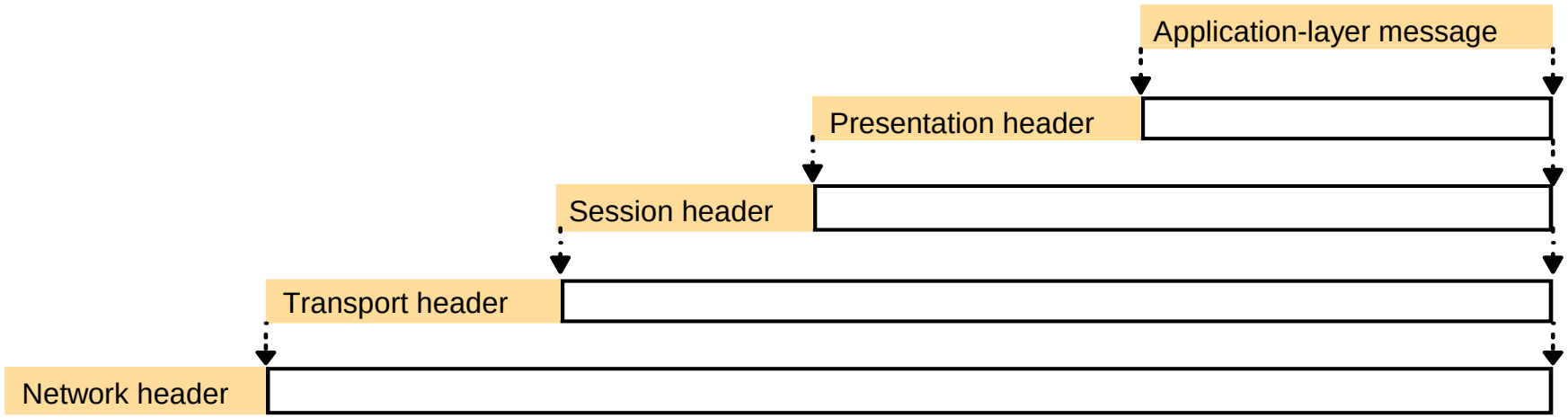
Protocols: Layers



Protocols: Layers

Encapsulation

At the sending side, each layer accepts items at the specified format



Protocols: Layers

Protocol suite or Protocol stack: a complete set of protocols reflecting the layered structure

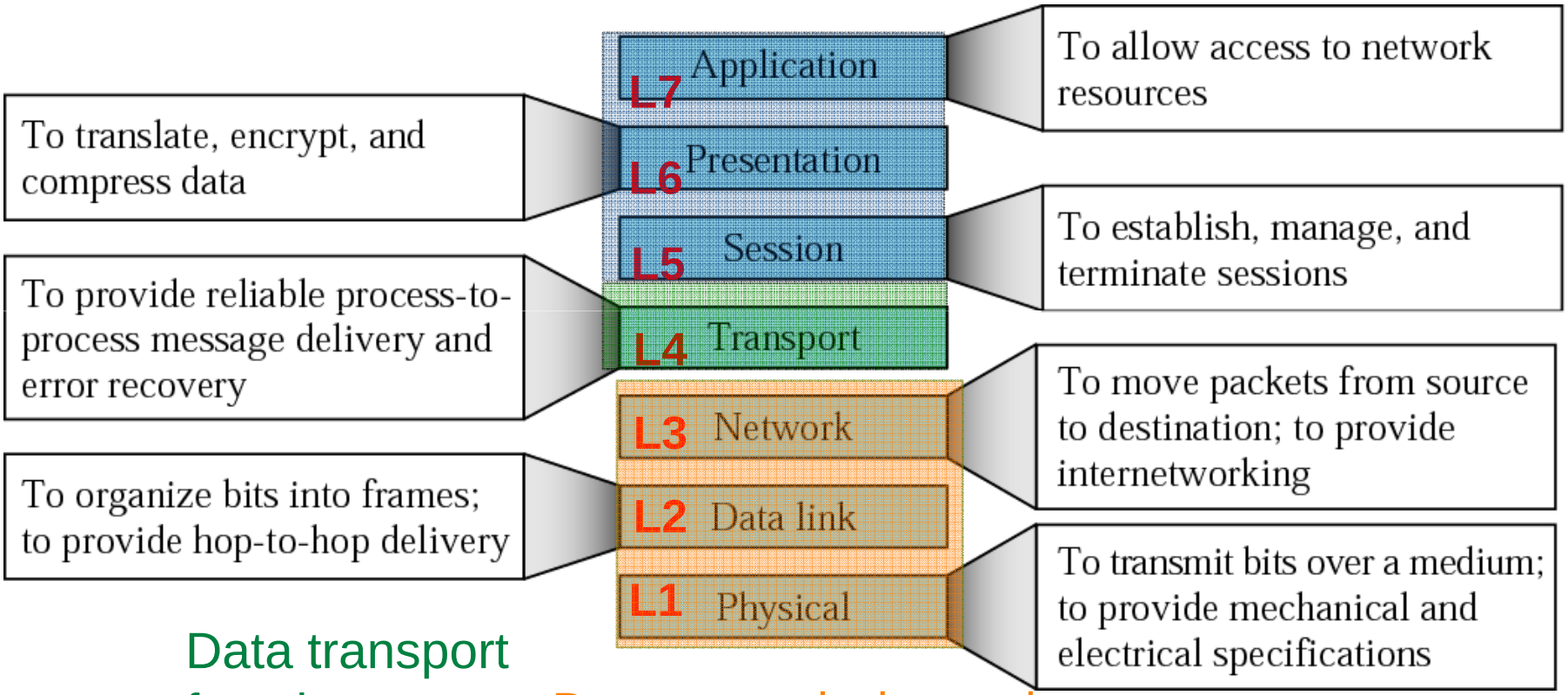
ISO OSI

- Seven layers that define the functions of data communication protocols.

- A layer does not define a single protocol

Protocols: OSI Layers

Application-related function



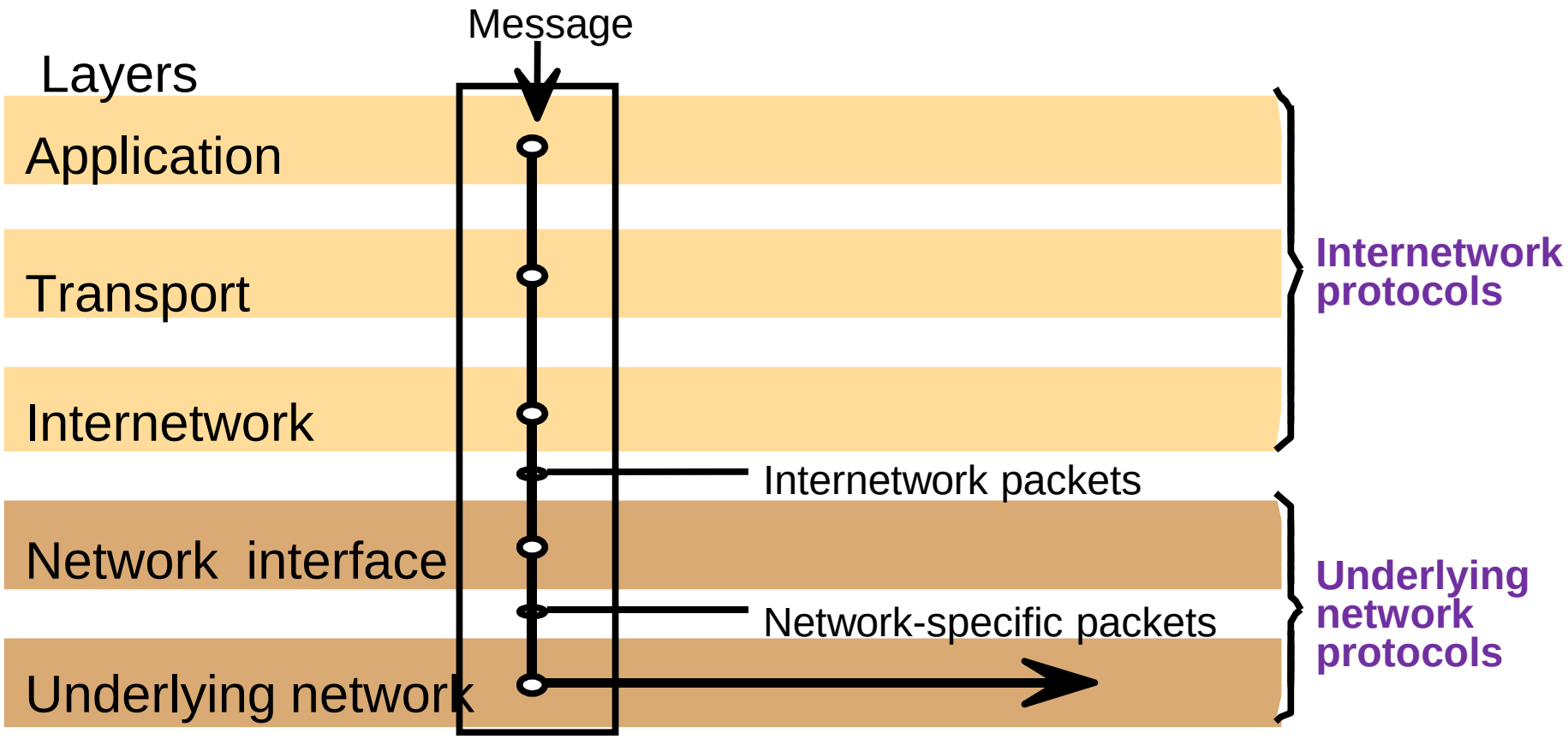
Data transport function

Data transmission and Routing functions

Protocols: OSI Layers (details)

Layer	Description	Examples
Application	Protocols that are designed to meet the communication requirements of specific applications, often defining the interface to a service.	**HTTP**, FTP , SMTP, CORBA IIOP
Presentation	Protocols at this level transmit data in a network representation that is independent of the representations used in individual computers, which may differ. Encryption is also performed in this layer, if required.	Secure Sockets (SSL),**CORBA** Data Rep.
Session	At this level *reliability and adaptation* are performed, such as detection of failures and automatic recovery.	
Transport	This is *the lowest level at which messages (rather than packets) are handled*. *Messages are addressed to communication ports attached to processes*, Protocols in this layer may be connection-oriented or connectionless.	**TCP**, UDP
Network	Transfers *data packets between computers in a specific network*. In a WAN or an internetwork this involves the generation of a route passing through routers. In a single LAN no routing is required.	**IP,** ATM virtual circuits
Data link	Responsible for transmission of *packets between nodes that are directly connected by a physical link*. In a WAN transmission is between pairs of routers or between routers and hosts. In a LAN it is between any pair of hosts.	Ethernet MAC, ATM cell transfer, PPP
Physical	The circuits and hardware that drive the network. It transmits *sequences of binary data* by analogue signalling, using amplitude or frequency modulation of electrical signals (on cable circuits), light signals (on fibre optic circuits) or other electromagnetic signals (on radio and microwave circuits).	Ethernet base-band signalling, ISDN

Protocols: Layers



Internet layers

Application = application + presentation
Transport = transport + session

Internetworking

Integrate many subnets

1. A unified *internetworking addressing scheme* that enables packets to be addressed to any host connected to any subnet (**IP addresses**)

2. A *protocol* defining the format of internetwork packets and giving rules according to which they are handled (**IP Protocol**)

3. *Internetworking components* that route packets to their destinations (**Internet routers**)

Internetworking: network nodes

- Intranet
  - many elements in one administrative domain
- Internet
  - collection of interconnected networks, across administrative domains.
- Host
  - a computer on the net

Internetworking: network nodes

- Router
  - a host that routes packets from one link to another *at **IP level***.
  - Often dedicated with no applications.
  - Maintain routing tables.
  - Linked to each other by direct connections or may be interconnected through subnets
- Switch
  - perform similar function to routers but for *local networks* (at the data link layer) only.
  - They interconnect separate ***Ethernets***, routing the incoming packets to the appropriate outgoing network.
  - By making temporary hardware connections between two ports (or store and forward) [switches do not exchange info with each other]

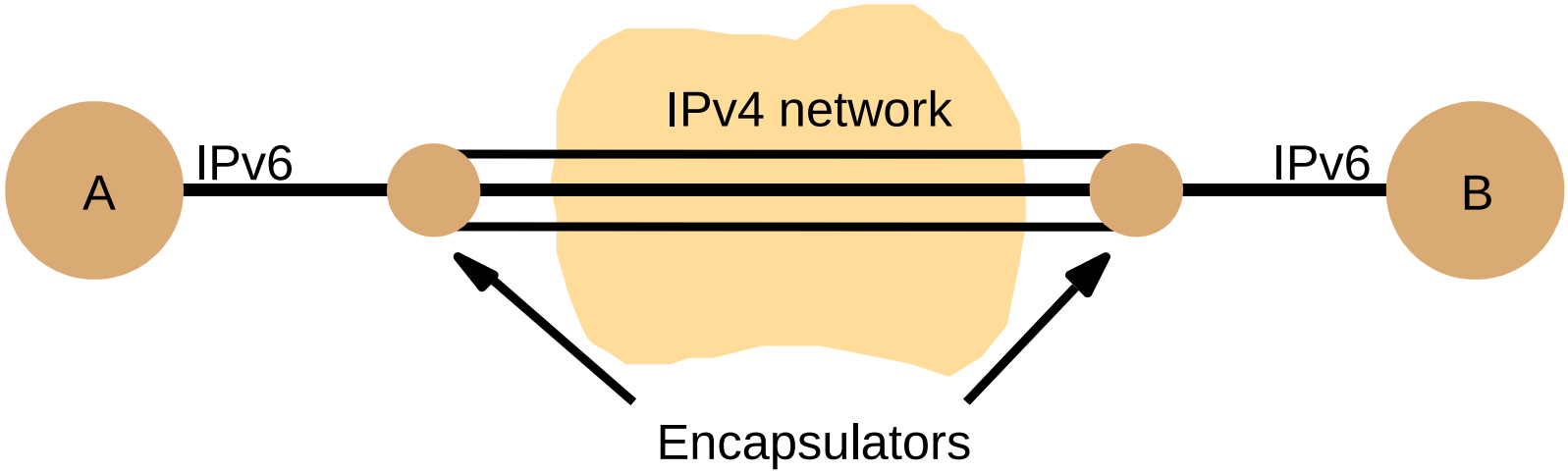
Internetworking: network nodes

- Hubs: (extend the segment of a LAN)
  - connect several computers together in a network, extend segments of Ethernet, and allow various computers to communicate with each other. Have a number of sockets to which a host computer may be connected (to support many segments)
- Bridges:
  - links different LANs together in a WAN, could be routers as well
- Gateways:
  - connect different types of networks. Can be configured to enable security. Used by networks as their primary link to Internet.
- Repeaters:
  - signal amplifiers.
- Network Access Points (NAPs):
  - connect regional midlevel networks to each other.

Internetworking: Routers

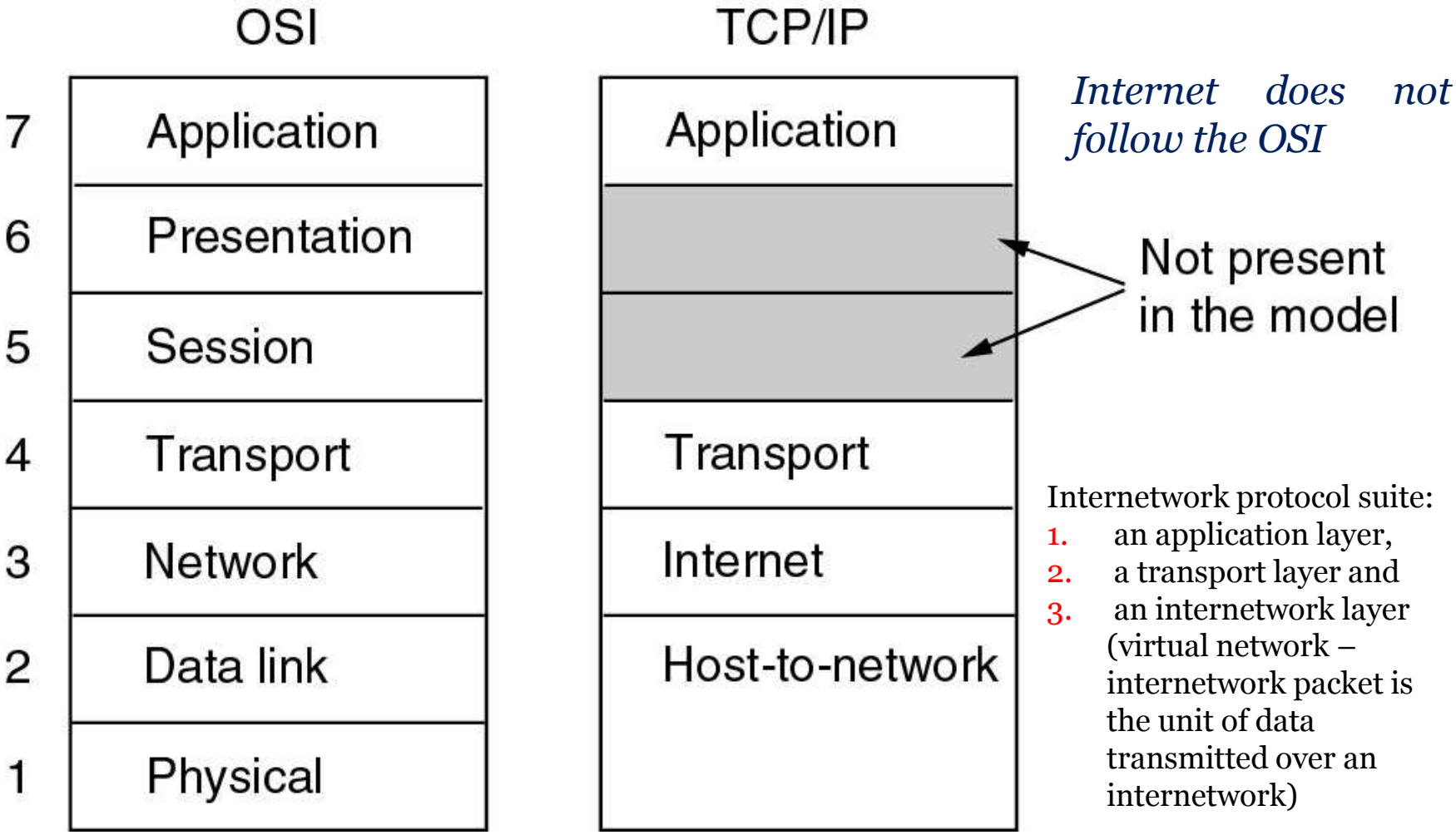
- Tunneling
  - a protocol tunnel is a software layer that transmits packets communicate through an "alien" protocol
  - “Hide” in the payload

(A, B islands of IPv6, at the boundaries they are encapsulated inside IPv4) *IPv6 traffic using IPv4 protocols*



Internet Protocols

Internet Protocols



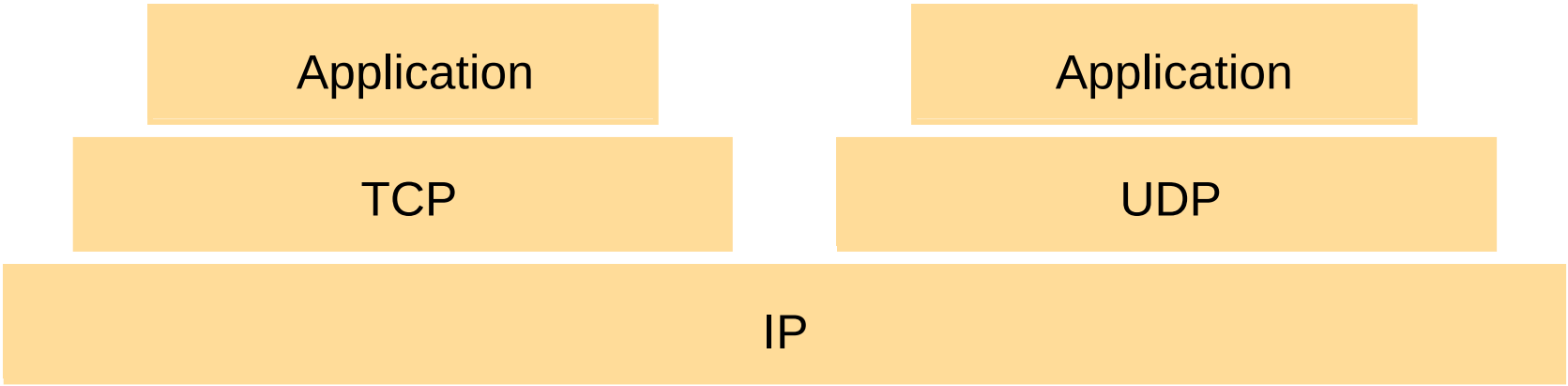
Internet does not follow the OSI

Internetwork protocol suite:

1. an application layer,
2. a transport layer and
3. an internetwork layer (virtual network – internetwork packet is the unit of data transmitted over an internetwork)

Internet Protocols

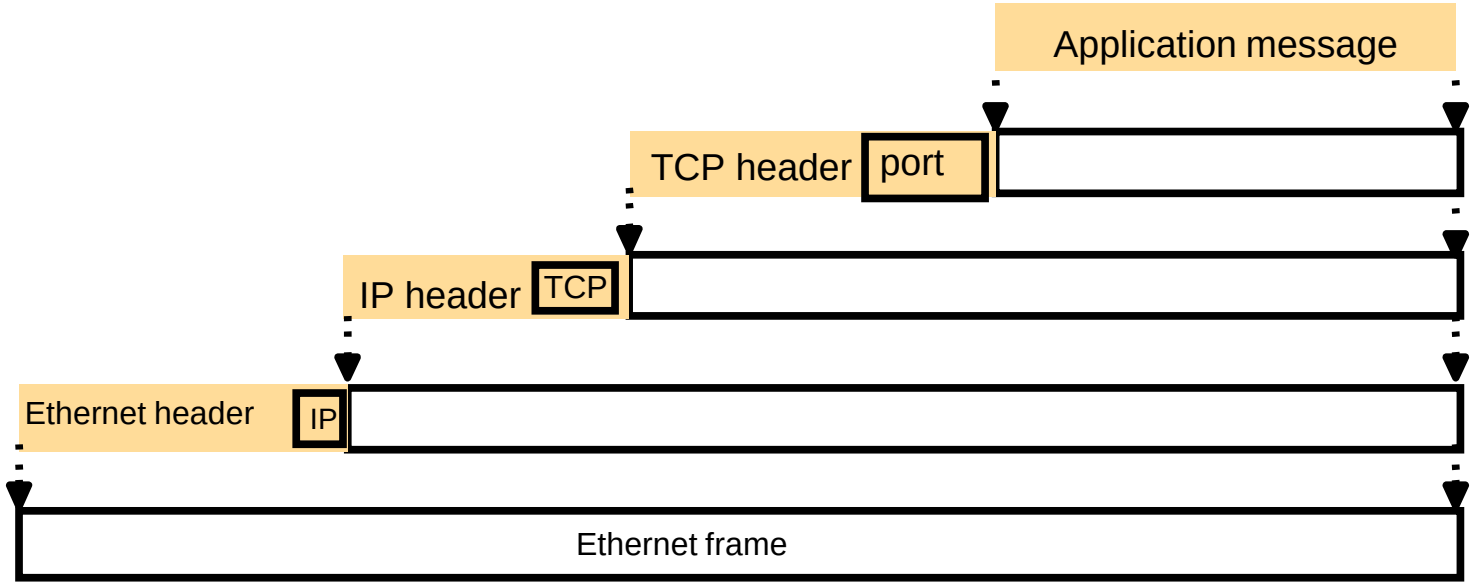
Transport protocols: TCP (reliable connection-oriented) UDP (datagram protocol that does not guarantee reliable delivery)
IP underlying "*network" protocol*



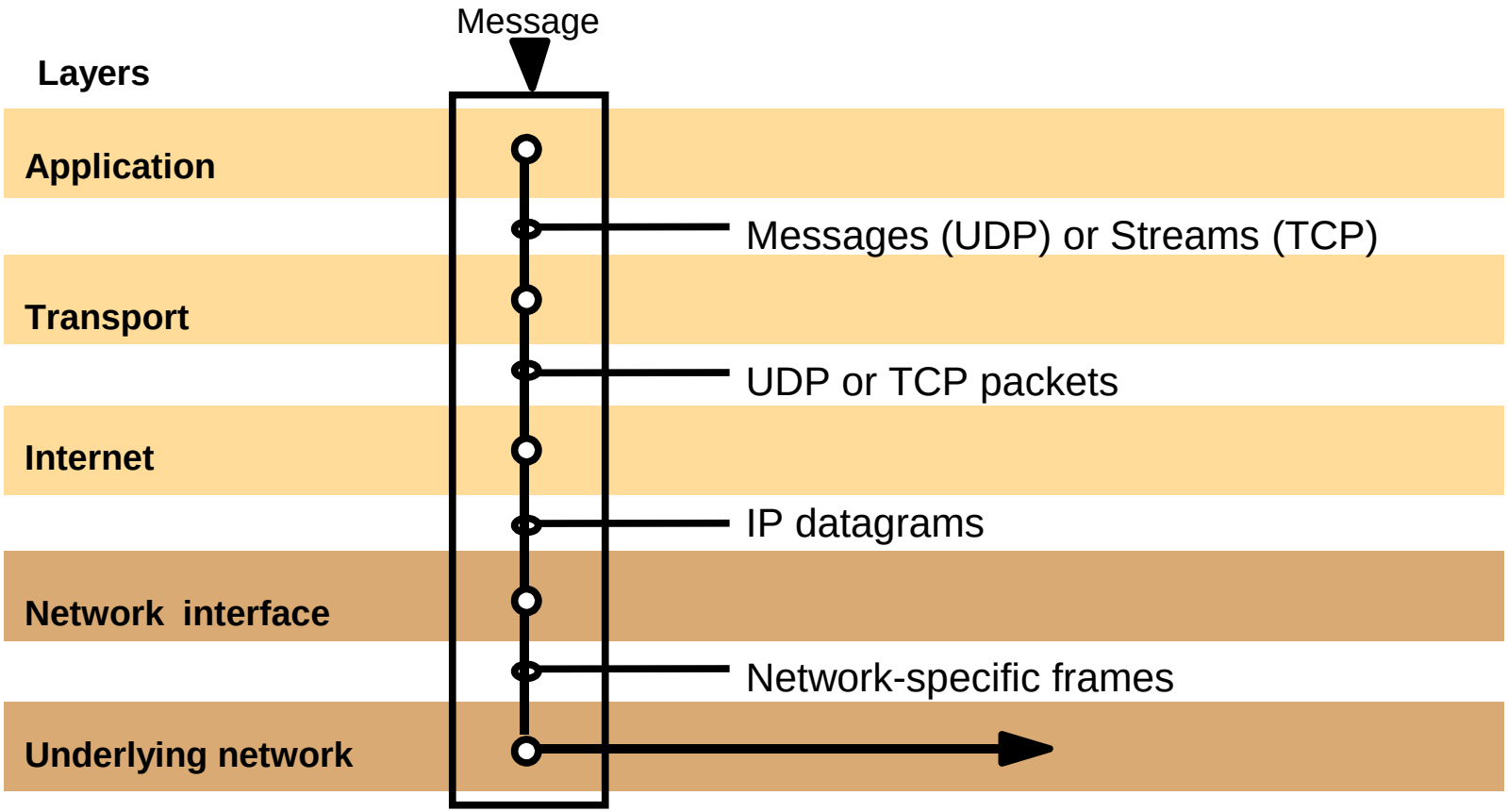
(inter) Network-layer packets consist of a header and a data field *HOST TO HOST*
Maximum transfer unit (MTU) [MTU in IP 64 kbytes]

Transport layer
Deliver messages to destinations with transport addresses *PROCESS TO PROCESS*
Transport address: network address + a port number
– ports are software-defined destination points at a host computer attached to processes

Internet Protocols: encapsulation

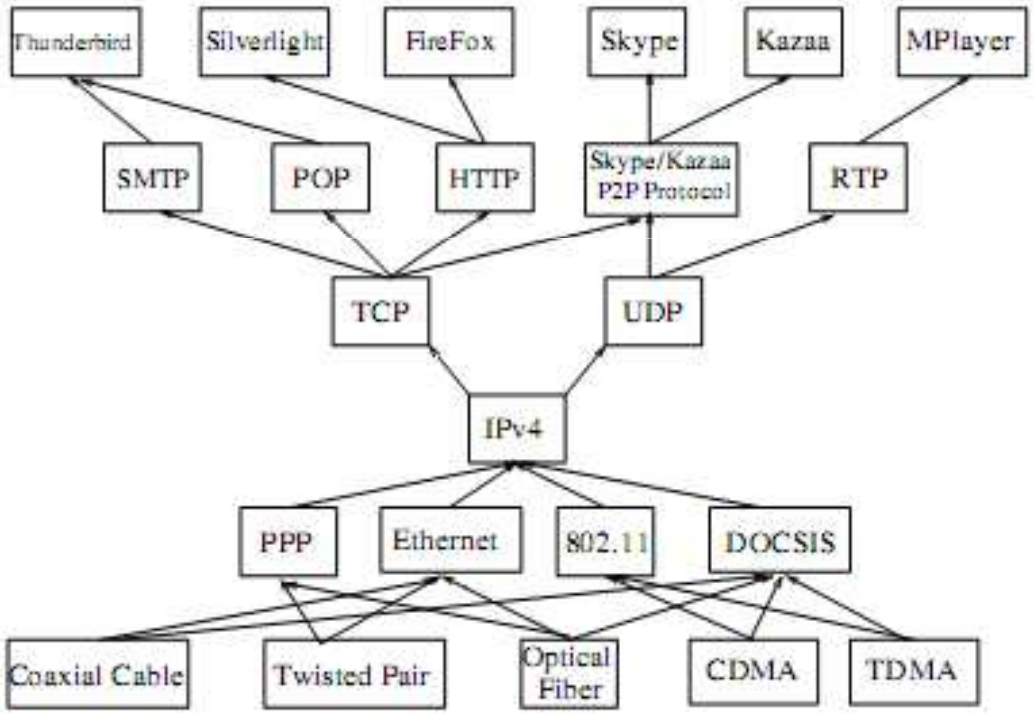


Internet Protocols

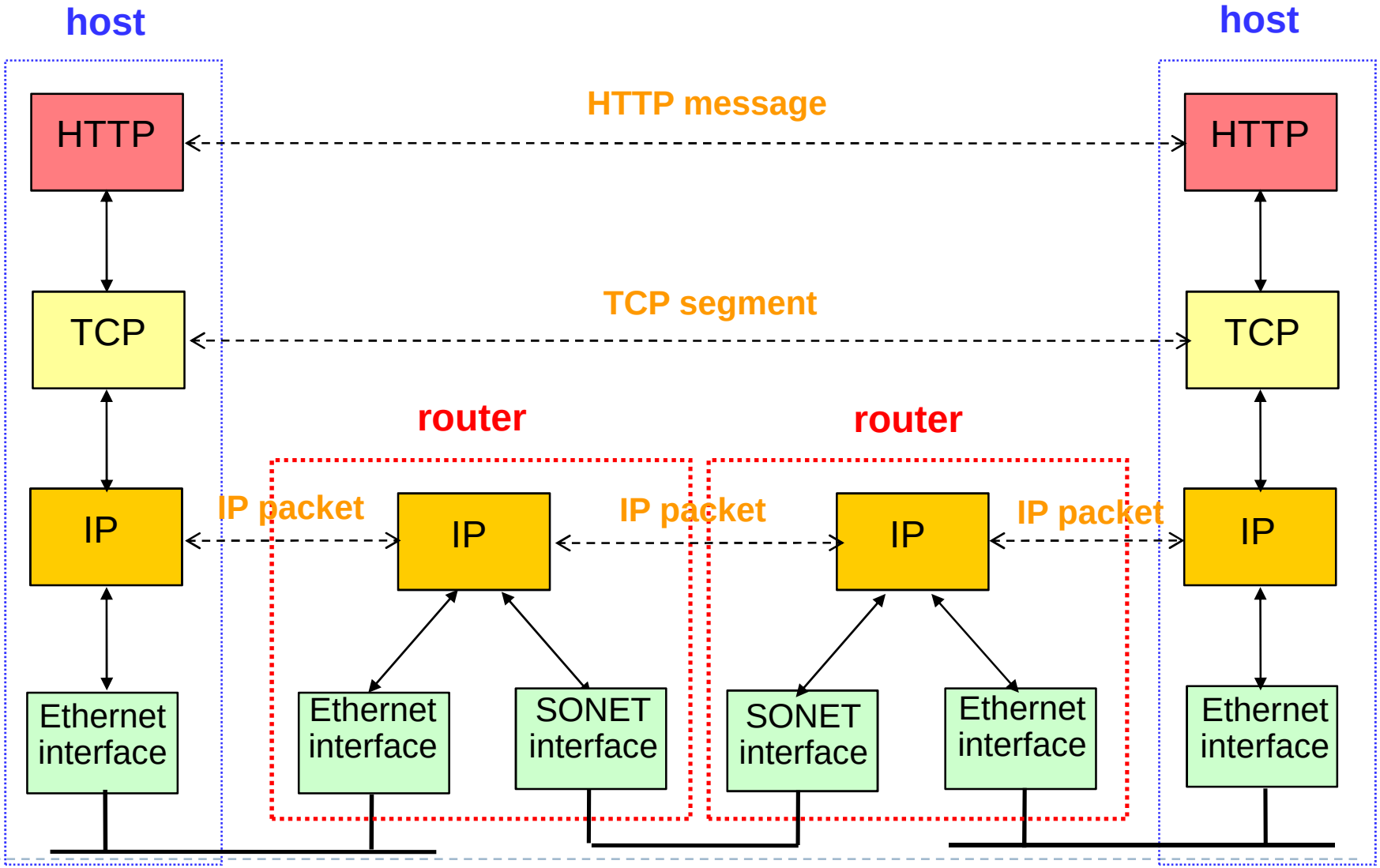


Internet Protocols

the Hourglass Architecture



IP Suite: End Hosts vs. Routers



Internet Protocols

- The popularity of TCP/IP is due to a number of important features:
  - *Open protocol standards*: freely available and developed independently from any computer hardware or OS.
  - Independence from specific physical network hardware.
  - A common addressing scheme.
  - Standardized high-level protocols.
- TCP/IP standards and protocols are published publicly as *Requests for Comments* (RFCs).

Lecture Outline

- Main points of Lecture 3
- The Internet Protocols:
  - IP,
  - UDP,
  - TCP
- DNS (name services)

IP Overview

Internetworking

Integrate many subnets (possibly different technologies: e.g., Etherner, ATM, DSL)

1. A unified *internetworking addressing scheme* that enables packets to be addressed to any host connected to any subnet (**IP addresses**)

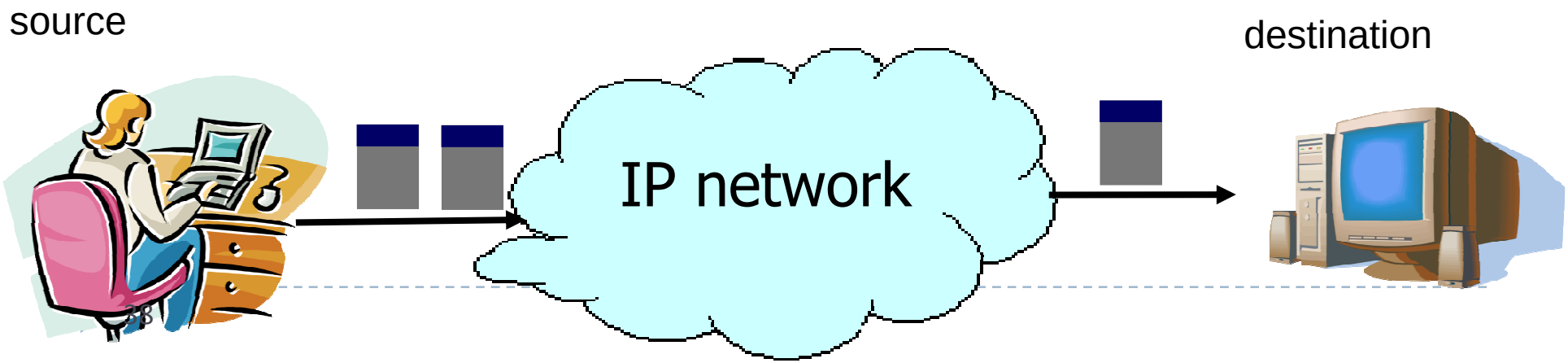
2. A *protocol* defining the format of internetwork packets and giving rules according to which they are handled (**IP Protocol**)

3. *Internetworking components* that route packets to their destinations using the technology of the underlying subnets (**IP Routing**)

Datagram packet
one-shot, no initial set up
different routes, out of order

IP Service: Best-Effort Packet Delivery

- Packet switching
  - Divide messages into a sequence of packets
  - Headers with source and destination address
- Best-effort delivery
  - Packets may be lost
  - Packets may be corrupted
  - Packets may be delivered out of order



IP Service Model: Why Packets?

- Data traffic is bursty
  - Do not waste reserved bandwidth
- Better multiplexing
  - Different transfers share access to the same links
- Better resource allocation (memory, buffers)
- Communication continue despite failures (on another path)

IP Service Model: Why Best-Effort?

- No need to reserve bandwidth and memory
- No network congestion control (beyond chocking)
- No need to do error detection & correction
- No need to synchronize (e.g., remember from one packet to next)

IP Addressing

Internet Protocols: IP addressing

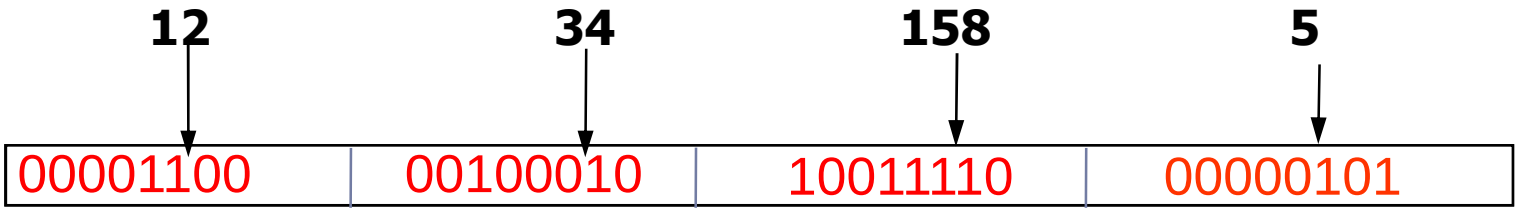
❖ Identify every connected host

It must be

- Universal
- Efficient in its use of address space
  - 2^{32} or 4 billion
- Lead to a flexible and efficient routing scheme

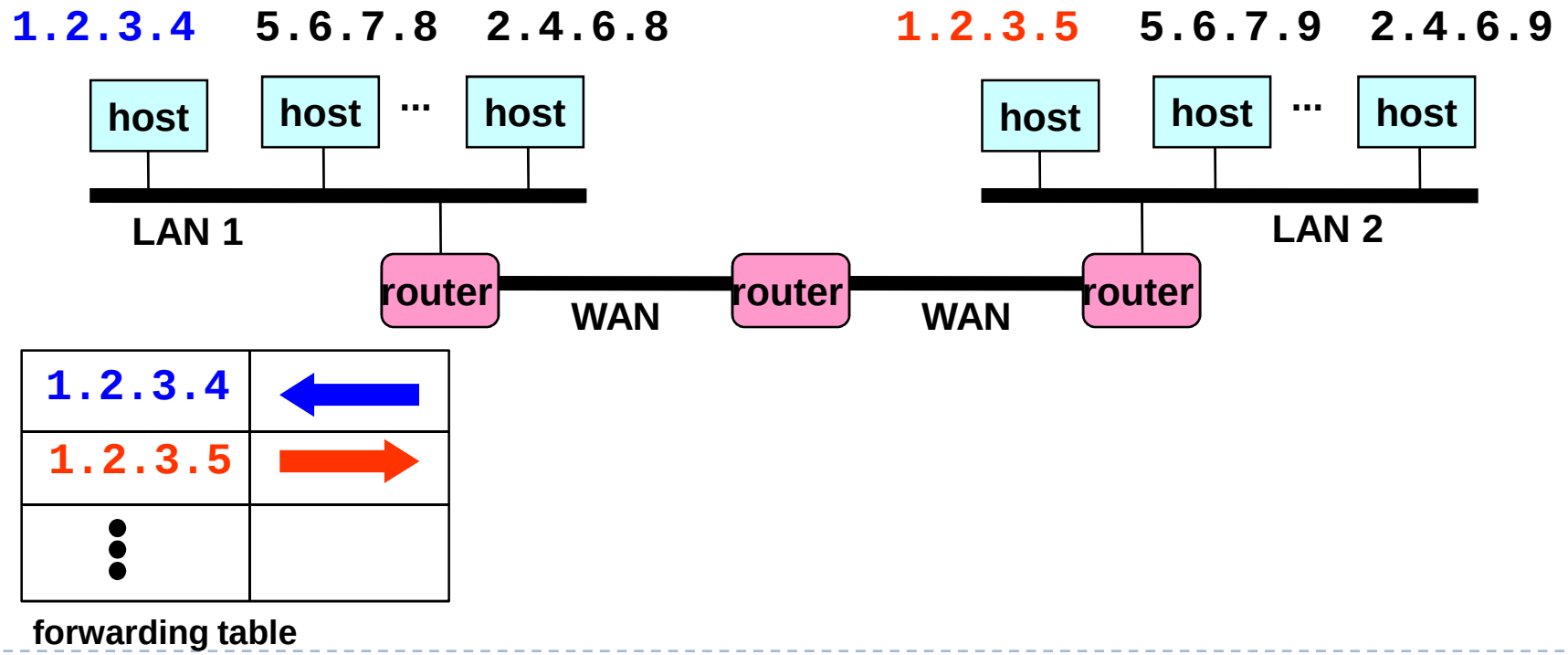
IP version 4

- A unique 32-bit number
- Identifies an interface (on a host, on a router, …)
- Represented in dotted-quad notation



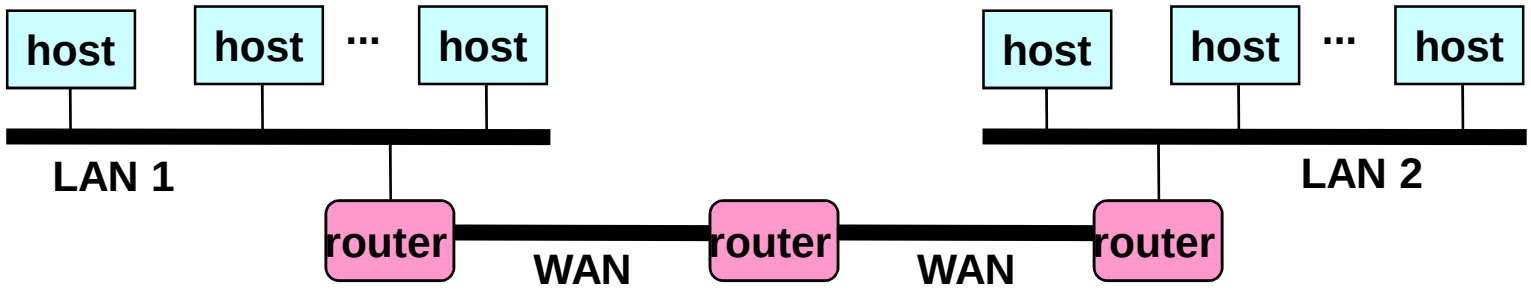
Internet Protocols: IP addressing (scalability)

- Suppose hosts had arbitrary addresses
  - Then every router would need a lot of information
  - …to know how to direct packets toward the host



Internet Protocols: IP addressing (scalability)

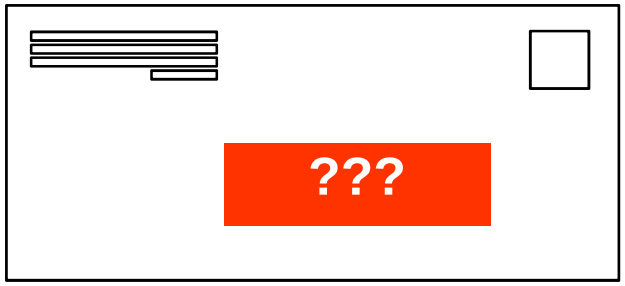
- The Internet is an "inter-network"
  - Used to connect *networks* together, not *hosts*
  - Needs a way to address a network (i.e., group of hosts)



LAN = Local Area Network
WAN = Wide Area Network

Internet Protocols: IP addressing (scalability)

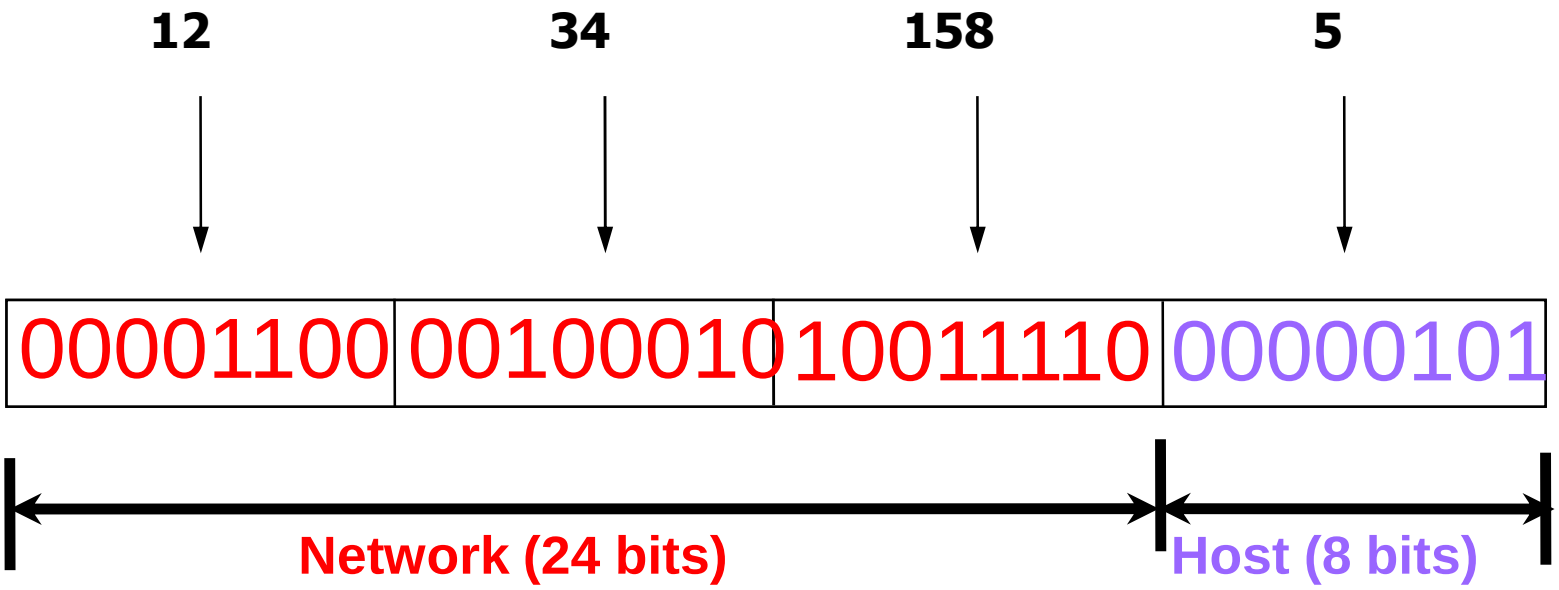
- Addressing in the U.S. mail
  - Zip code: 08540
  - Street: Olden Street
  - Building on street: 35
  - Room in building: 306
  - Name of occupant: Jennifer Rexford



- Forwarding the U.S. mail
  - Deliver letter to the post office in the zip code
  - Assign letter to mailman covering the street
  - Drop letter into mailbox for the building/room
  - Give letter to the appropriate person

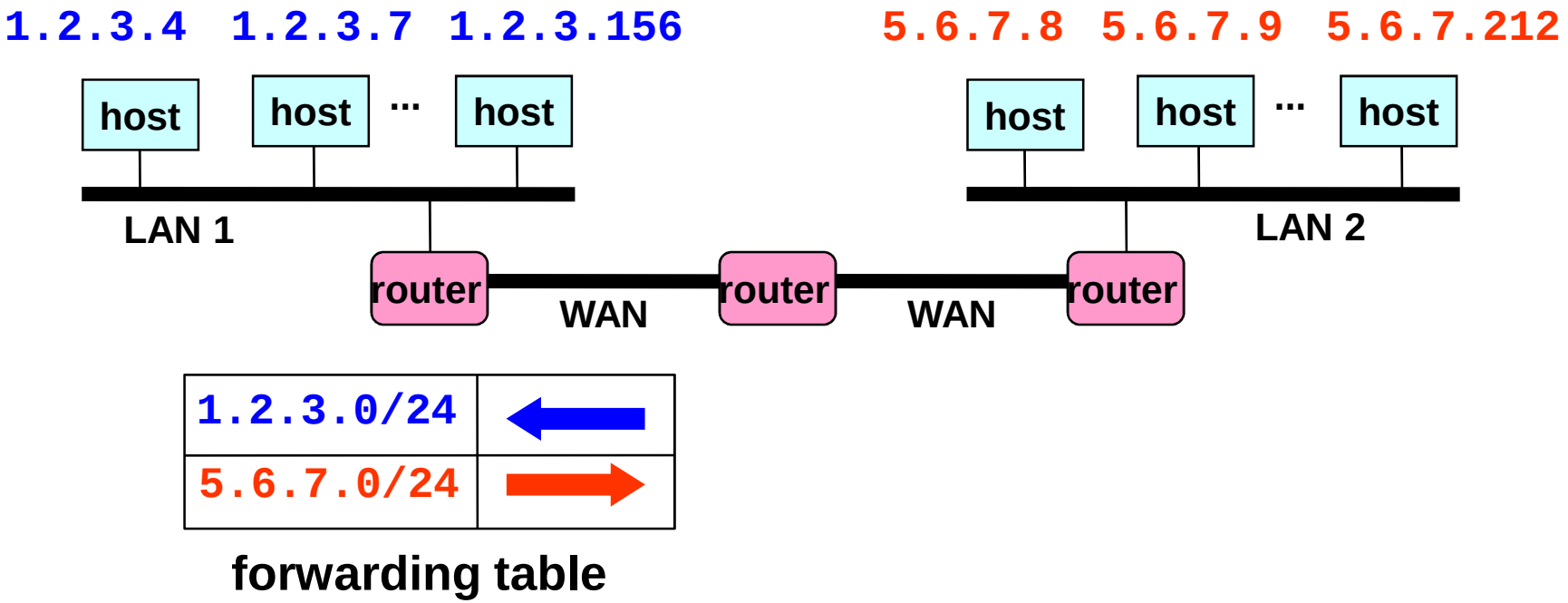
Internet Protocols: IP addressing (prefixes)

- Divided into **network** & **host portions** (left and right)
- 12.34.158.0/24 is a 24-bit prefix with 2^8 addresses



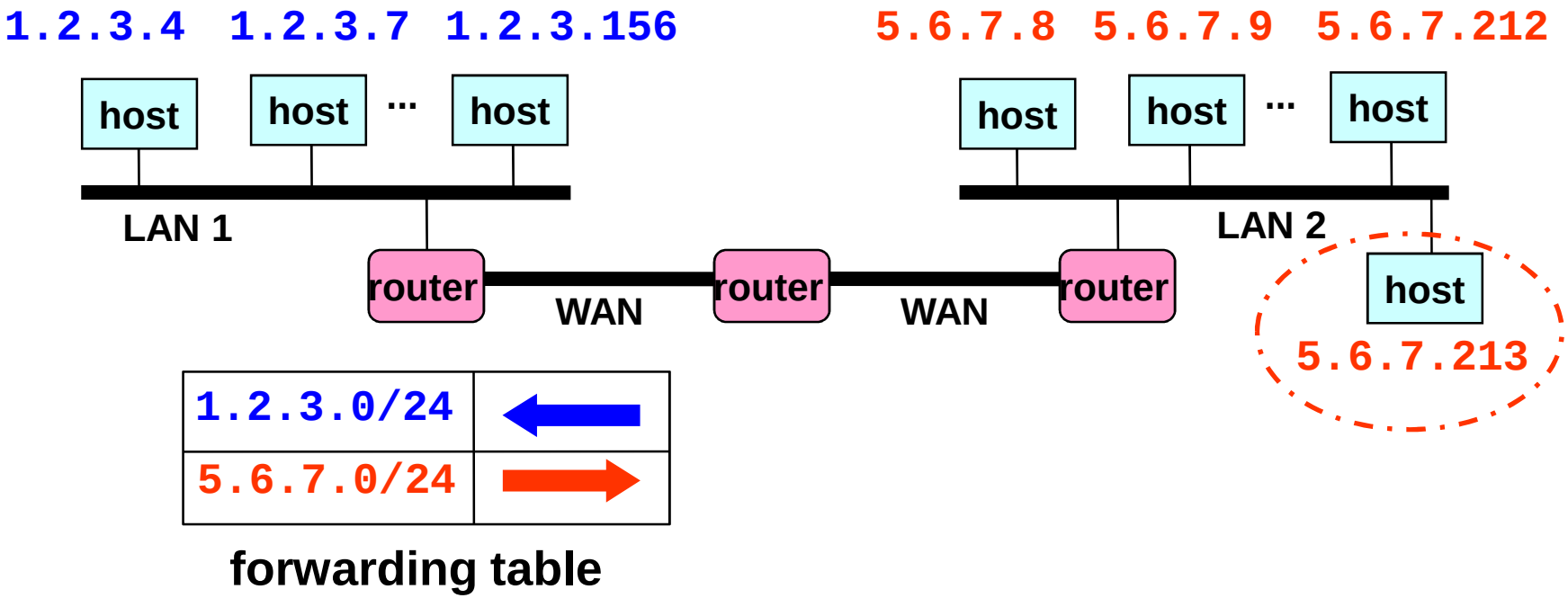
Internet Protocols: IP addressing (prefixes)

- Improved scalability: Number related hosts from a common subnet
  - 1.2.3.0/24 on the left LAN
  - 5.6.7.0/24 on the right LAN



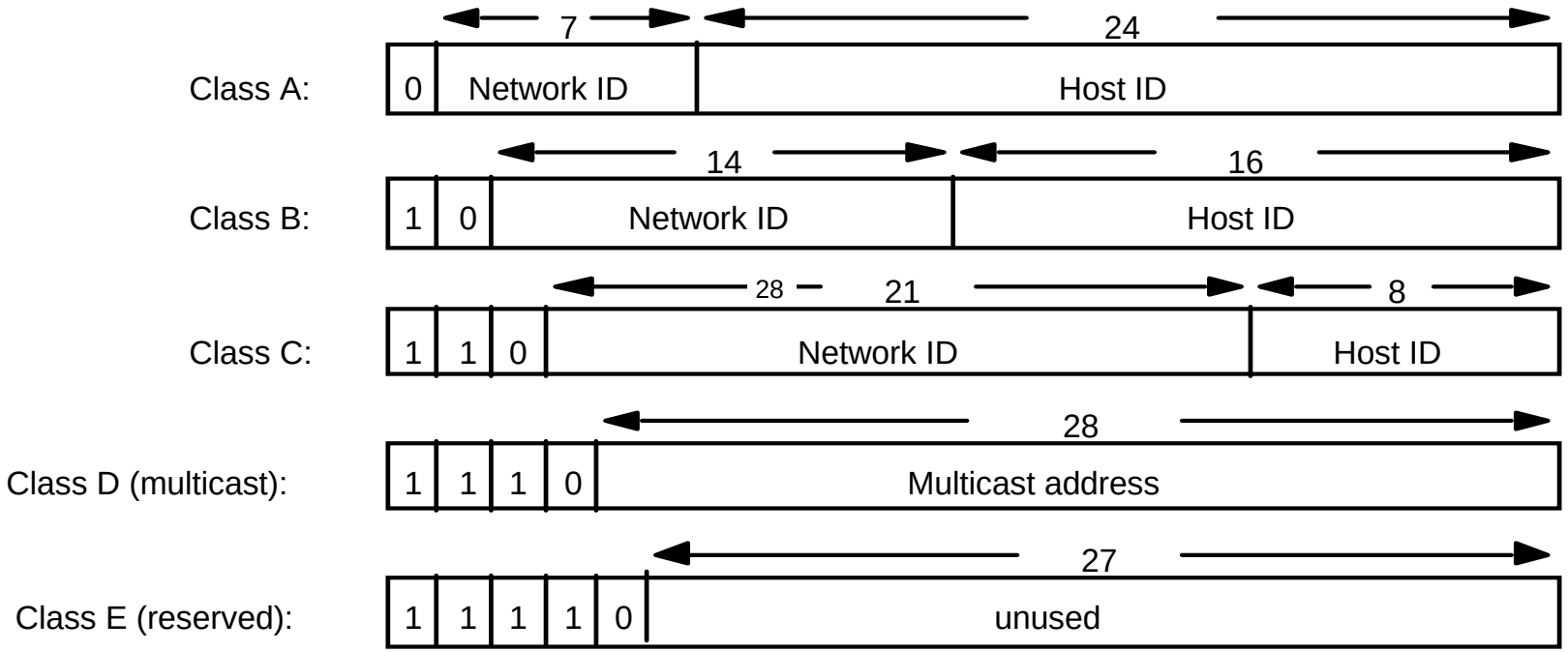
Internet Protocols: IP addressing (prefixes)

- Easy to add new hosts - no need to update the routers
  - E.g., adding a new host 5.6.7.213 on the right
  - Doesn't require adding a new forwarding entry



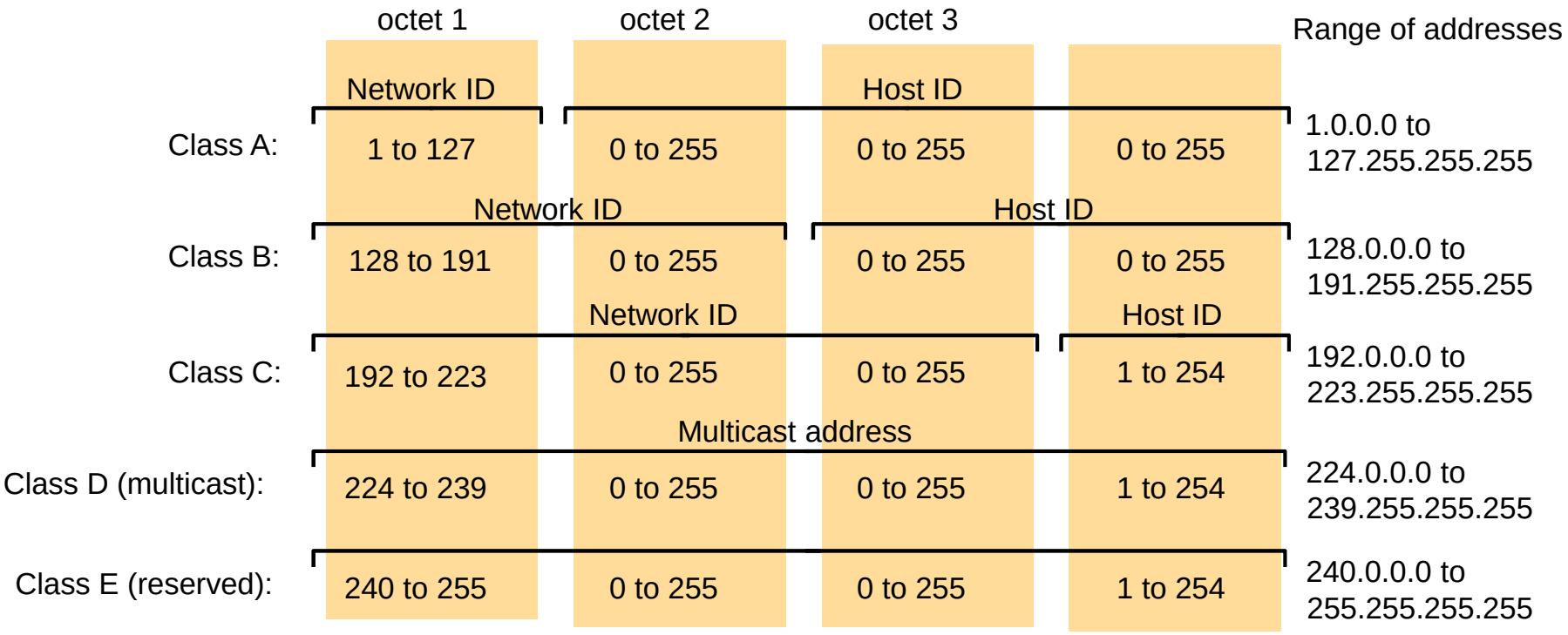
Internet Protocols: IP addressing

How to divide the 32 bits into:
Network Identifier (identifies one sub-network) +
Host identifier (identifies the host connection to that network)



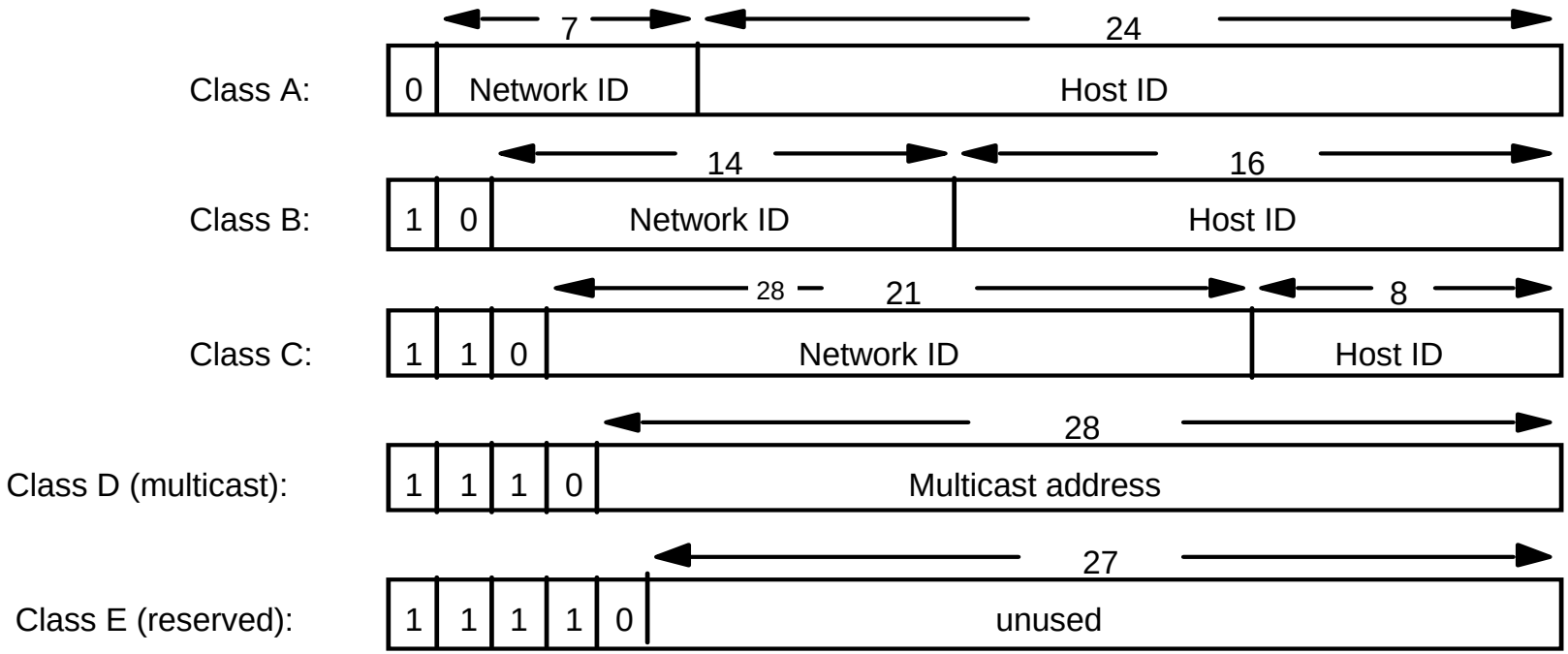
Internet Protocols: IP addressing

32 bits:
Written as sequence of four decimal bytes, one byte - octet



Internet Protocols: IP addressing

Class A 2^{24} hosts in each subnet, for very large networks
Class B 255 hosts
Class C fewer



Internet Protocols: IP addressing

- In the older days, only fixed allocation sizes
  - Class A: 0*
    - Very large /8 blocks (e.g., MIT has 18.0.0.0/8)
  - Class B: 10*
    - Large /16 blocks (e.g,. Princeton has 128.112.0.0/16)
  - Class C: 110*
    - Small /24 blocks (e.g., AT&T Labs has 192.20.225.0/24)
  - Class D: 1110*
    - Multicast groups
  - Class E: 11110*
    - Reserved for future use

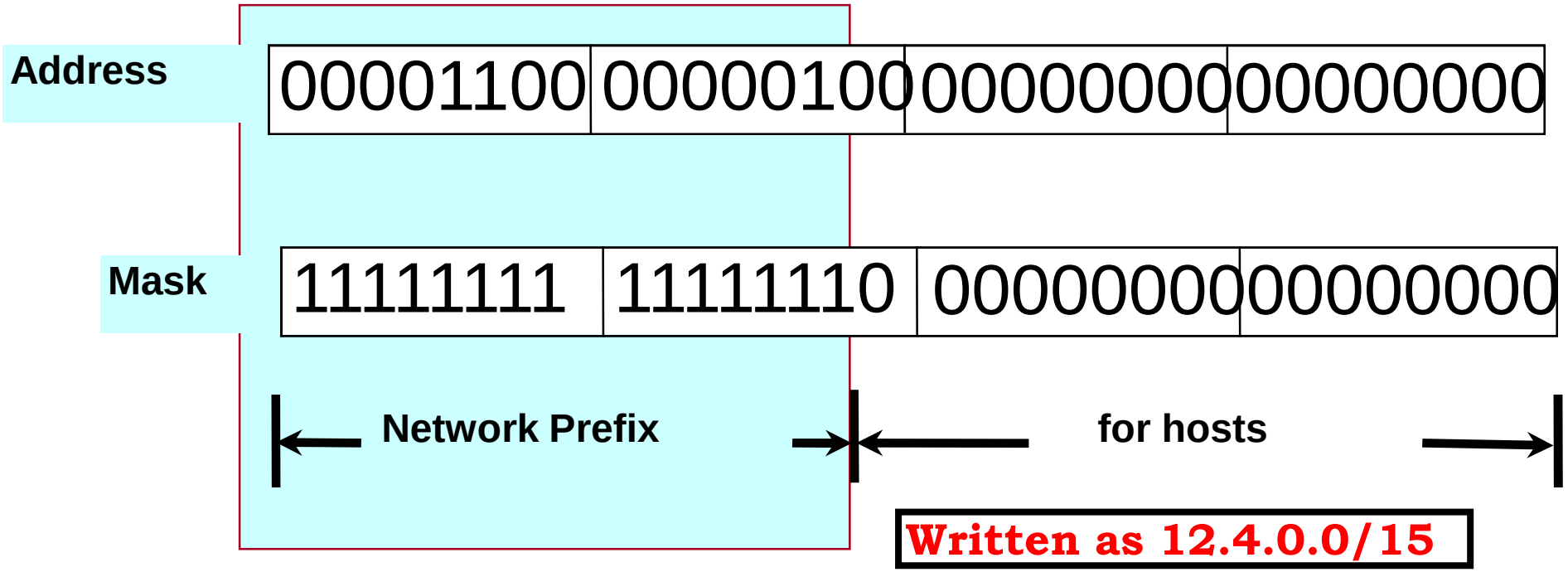
Internet Protocols: IP addressing (CIDR)

- Classless interdomain routing (CIDR)
  - Shortage of Class B networks
    - Allocate a batch of contiguous Class C addresses to a subnet requiring more than 255 addresses (also, subdivide Class B)
  - **Add a mask field** (bit pattern) to indicate bits for network portion that is compared with the routing table entry
  - 138.73.59.32/22 [subnet: first 22 bits; host: 10 bits]

Internet Protocols: IP addressing (CIDR)

Use two 32-bit numbers to represent a network.
Network number = IP address + Mask

IP Address : 12.4.0.0 **IP Mask: 255.254.0.0**

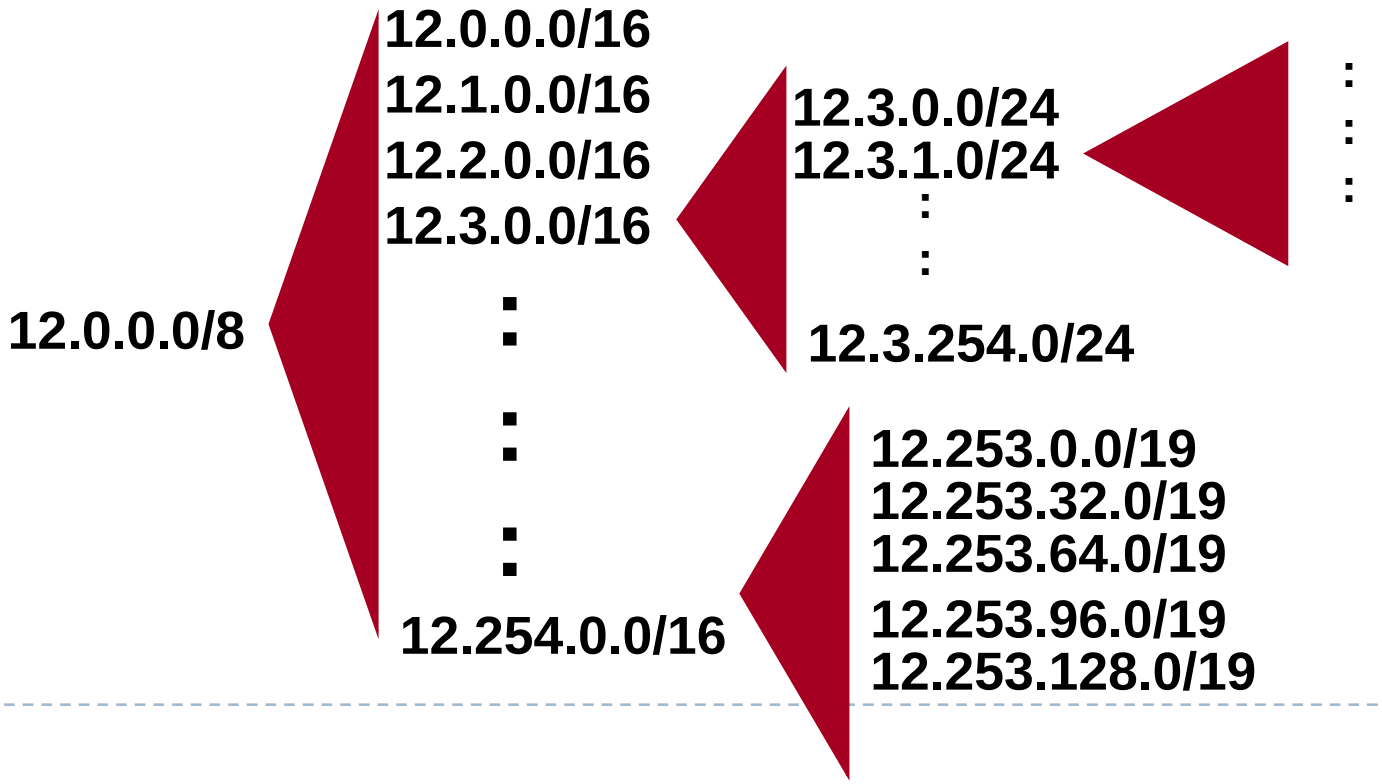


Internet Protocols: IP addressing (CIDR)

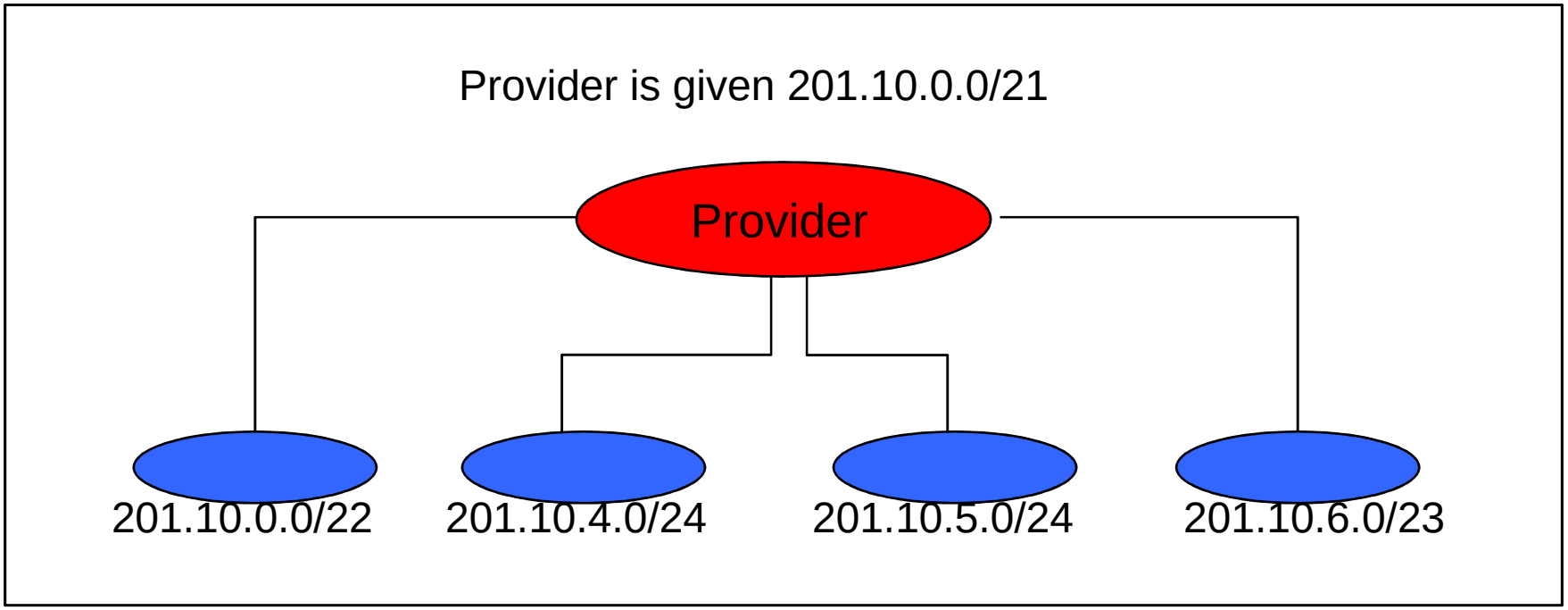
Hierarchical Address Allocation:

Prefixes are key to Internet scalability

- Address allocated in contiguous chunks (prefixes)
- Routing protocols and packet forwarding based on prefixes
- Today, routing tables contain ~150,000-200,000 prefixes

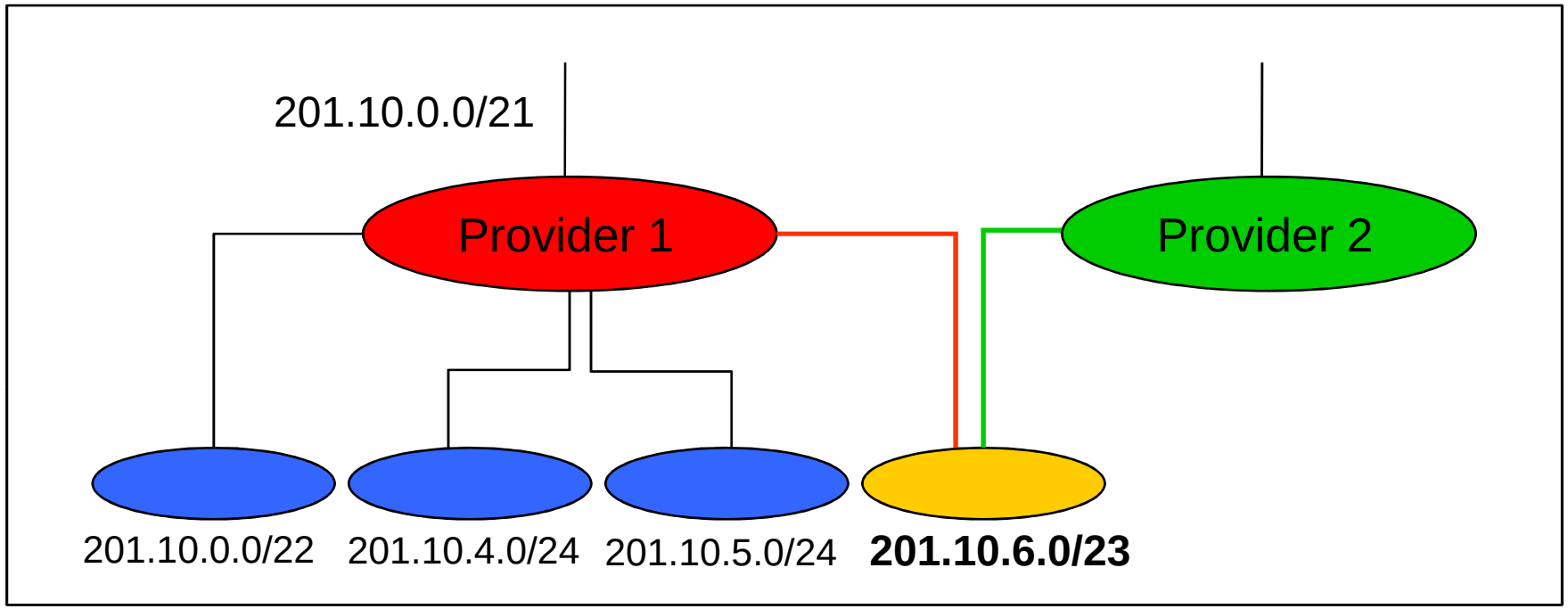


Internet Protocols: IP addressing (CIDR)



Routers in the rest of the Internet just need to know how to reach 201.10.0.0/21. The provider can direct the IP packets to the appropriate customer.

Internet Protocols: IP addressing (CIDR)



Multi-homed customer with 201.10.6.0/23 has two providers. Other parts of the Internet need to know how to reach these destinations through *both* providers.

❖ Any computer connected to more than one network must have a separate address on each

Internet Protocols: IP addressing

- Separation of control
  - Prefix: assigned to an institution
  - Addresses: assigned to nodes by the institution
- Who assigns prefixes?
  - Internet Corp. for Assigned Names and Numbers **(IANA)**
    - Allocates large blocks to Regional Internet Registries
  - Regional Internet Registries (RIRs)
    - E.g., ARIN (American Registry for Internet Numbers)
    - Allocated to ISPs and large institutions in a region
  - Internet Service Providers (ISPs)
    - Allocate address blocks to their customers
    - Who may, in turn, allocate to their customers…

Internet Protocols: IP Addressing (NAT)

- *Not all computers and devices need to be assigned globally unique (i.e., registered) IP addresses*

- Computers attached to a local network and access to NAT-enable routers to redirect incoming UDP/TCP packets for them – no register address

NAT-enabled routers maintain an address translation table (ATT)

- Router has a "global" IP address from ISP
- Each machine has a "local" IP address via a Dynamic Host Configuration Protocol (DHCP)

Internet Protocols: IP Addressing (NAT)

UDP or TCP packet from the internal network to a computer outside (machine->router->outside)

NAT Router

1. Receives the packet and saves the source IP address and port number to an available slot in the ATT
2. Replaces the *source address* in the packet with *its own IP address* and the *source port* with a *virtual port number* that indexes the table slot containing the sending computer's address information
3. Forwards the modified packet

UDP or TCP packet from an external computer (replies using the router's IP address and virtual port) (outside->router->machine)

Router

1. Uses the destination port number to access a slot in the ATT
2. Replaces the destination address and port with those stored in the slot and forwards the packet

Internet Protocols: IP Addressing (NAT)

Address Translation Table (ATT)

Maintains (cache) the mapping – a timer (if not accessed expires)

Ok, if as clients to external services, such as web services

Local machines as servers?

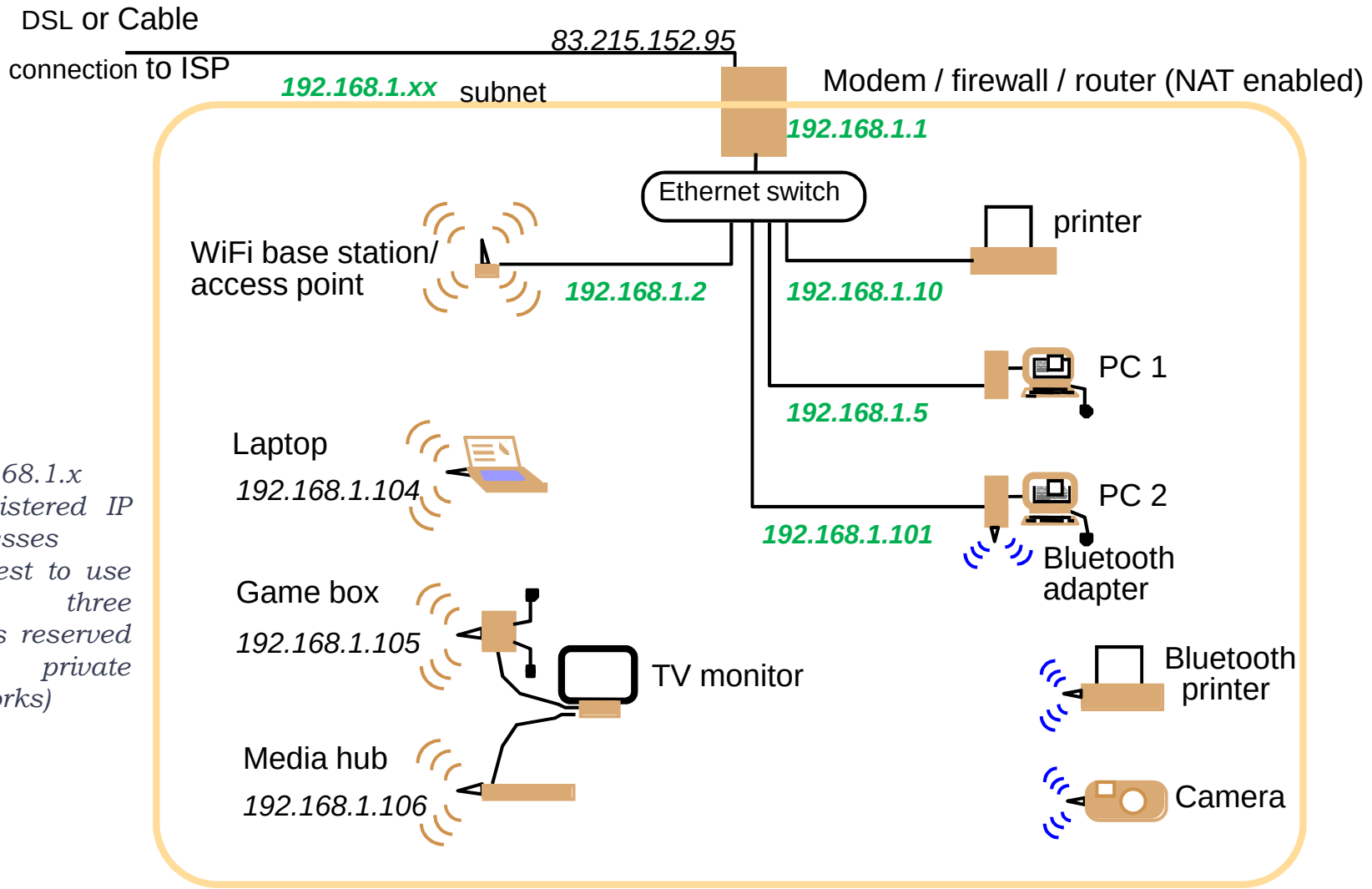
Manually configured to forward all of the incoming requests on a *given port* to one particular internal computer

- Fixed internal addr and port #
- Fixed entry in the table

All packets to the port on the router are forwarded to the internal addr and port # in the entry

Ok, if only one computer offers a service on any given port

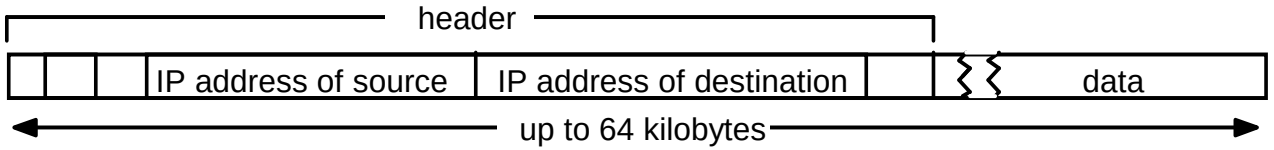
Internet Protocols: IP Addressing (NAT)



192.168.1.x unregistered IP addresses (suggest to use from three blocks reserved for private networks)

Internet Protocols: The IP Protocol

Transmits datagrams from one host to another, if necessary via intermediate routers

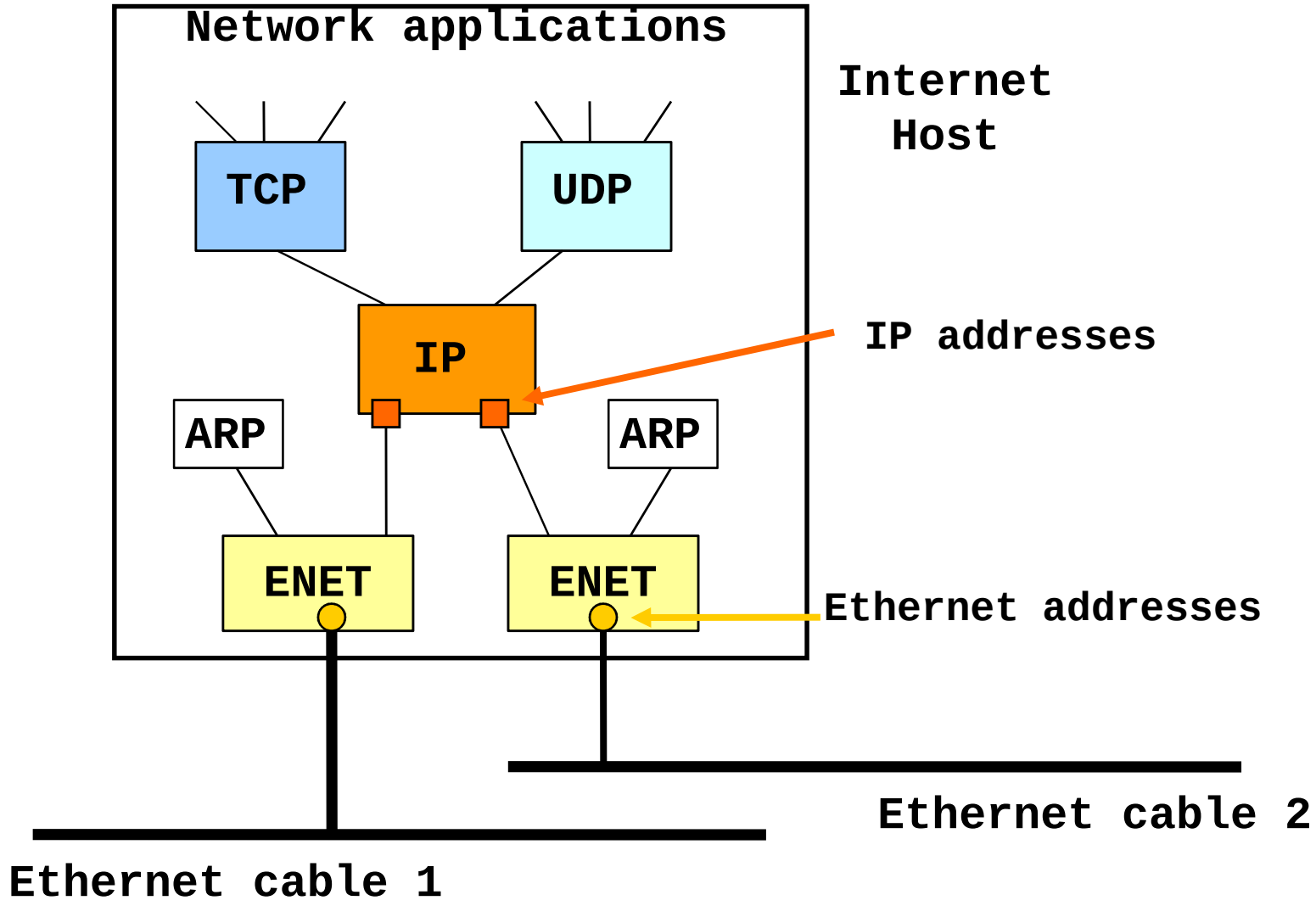


Provides only header checksum
Unreliable or best effort (packets may be lost, duplicated, delayed or delivered out of order)

If IP datagram is longer than MTU, it is broken into network packets

❖ *Inserts a "physical" network address attained through address resolution*

IP Protocol


Network applications
Internet
Host
TCP
UDP
IP
IP addresses
ARP
ARP
ENET
ENET
Ethernet addresses
Ethernet cable 2
Ethernet cable 1

Internet Protocols: The IP Protocol (APR)

Address Resolution Protocol (APR)

converts *Internet addresses* to *network addresses* for a specific underlying network (link-layer or physical addresses)

e.g., 32-bit Internet addresses to 48-bit Ethernet (MAC) addresses

Network topology dependent

- If hosts connected directly to Internet packet switches (no translation)
- Some LANs allow network addresses to be assigned dynamically to hosts chosen to match the id portion of the IP address
- For Ethernets

Internet Protocols: The IP Protocol (APR)

Used to translate IP addresses to Ethernet addresses.

- Translation done *only for outgoing IP packets*: this is when the IP header and the Ethernet header are created.
- Translation is performed with a table look-up in an ARP Table; stored (cached) in memory and contains a row for each computer:

IP address	Ethernet address
223.1.2.1	08-00-39-00-2F-C3
223.1.2.3	08-00-5A-21-A7-22
223.1.2.4	08-00-10-99-AC-54

Each ARP looks up the IP in the cache

- If in the cache, ok
- Else, transmit an Ethernet broadcast packet on the local Ethernet + cache the reply

ARP broadcasts are needed only when a computer is newly connected to the local Ethernet

Internet Protocols: The IP Protocol (APR)

- The ARP table is necessary because the IP address and Ethernet address are selected independently:
  - the IP address is selected by the network manager based on the location of the computer on the Internet.
  - the Ethernet address is selected by the manufacturer based on the Ethernet address space licensed by the manufacturer.

- Each host has separate ARP tables for each of its Ethernet interfaces.

Internet Protocols: The IP Protocol

IP spoofing: address can be stolen (source address is not authenticated)

Denial of service: many ping requests to a large number of computers at several sites with the IP address of a target computer as source

Internet Protocols: IP Routing

- RIP-1: discussed previously (distance-vector based)
- RIP-2: CIDR (classless), better multicast routing, authentication of RIP packets to prevent attacks at the routers
- link-state algorithms (each node constructs a map (graph) of the network): e.g., open shortest path first (OSPF)
- Observed: average latency of IP packets peaks at 30-seconds intervals [RIP updates are processed before IP]
  - because 30-second RIP update intervals, locked steps
  - random interval between 15-45 seconds for RIP update

Internet Protocols: IP Routing

- large routing table size
  - all destinations!!

Solution 1:

Topological grouping of IP addresses

map ip to geographical location

Solution 2:

- default route: store a subset, default to a single link for unlisted destinations
- (as long as key routers (those closest to the backbone links) have relatively complete table)

Internet Protocols: IP Routing

- The topological map of the Internet is divided conceptually into regions under a single administrative control, each of which is called an Autonomous System and contains a group of network IDs.
- Routing **inside an AS** is completely hidden from the rest of the Internet.
- Routes **between ASs** are computed in terms of AS hops: lists of intermediary ASs from the source AS to the destination AS.
  - All outside networks that belong to the same AS share the same route, expressed as the list of intermediary ASs.
  - To route to arbitrary ASs on the Internet, a router need only know the next-hop router to every AS, rather than to individual destinations.

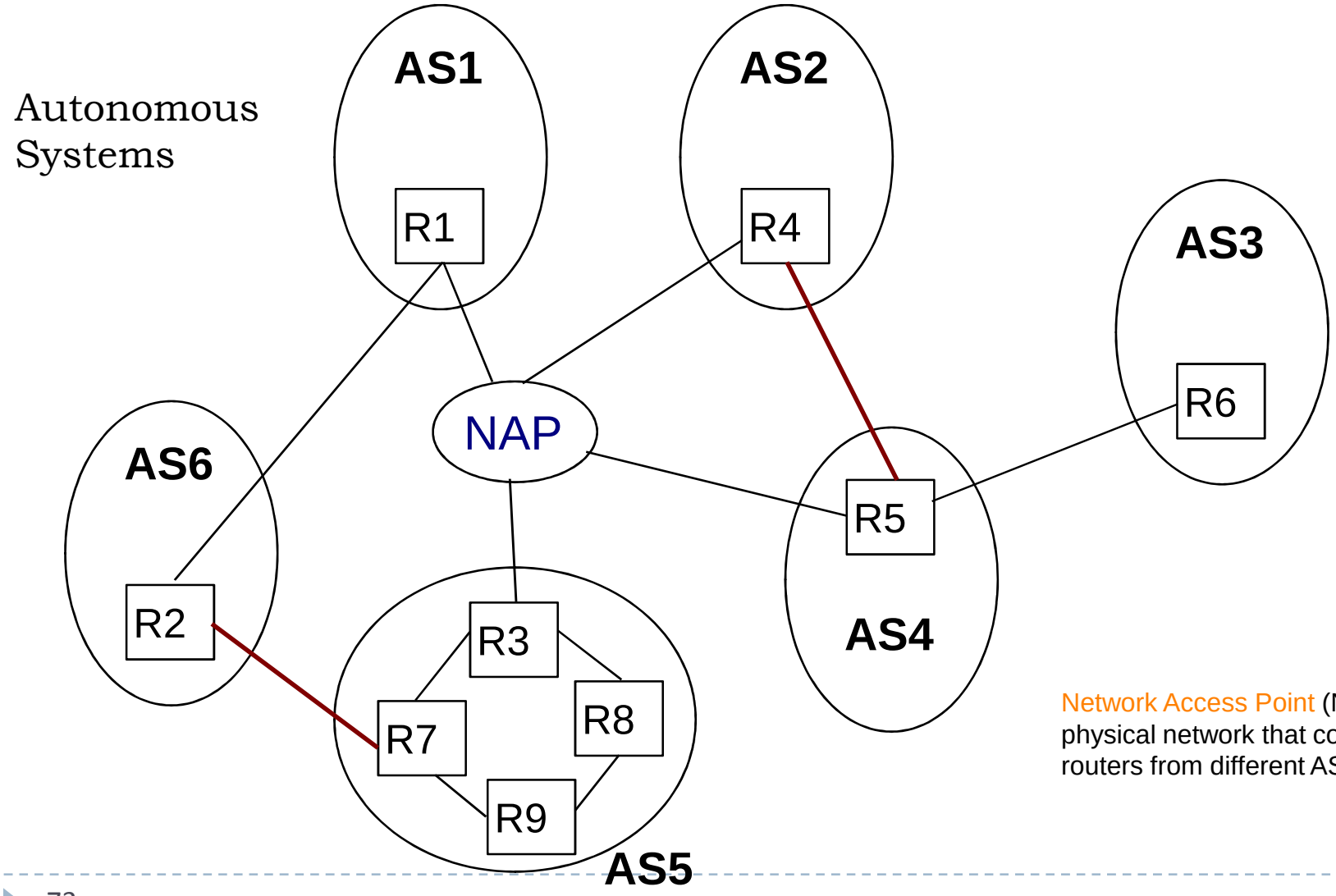
Internet Protocols: IP Routing

- Every AS has a backbone area

- The collection of routers that connect non-backbone areas to the backbone and the links that interconnect those routers are called the *backbone of the network*
- The links in the backbone are usually of high bandwidth and replicated or reliability

This hierarchic structure is a conceptual one primarily for resource management and maintenance

Internet Protocols: IP Routing



Network Access Point (NAP): a physical network that connects routers from different ASs

Internet Protocols: IP Routing

- Under complete control of AS owner
- Small ASs have only one router, which does the internal and external routing (e.g., AS1, AS2, AS3, AS4, AS6)
- Larger ASs may have more than one routers:
  - some of them are *border routers* and deal with outgoing or incoming traffic to the AS (e.g., AS5)
  - for internal routing the Open Shortest Path First (OSPF) protocol is used

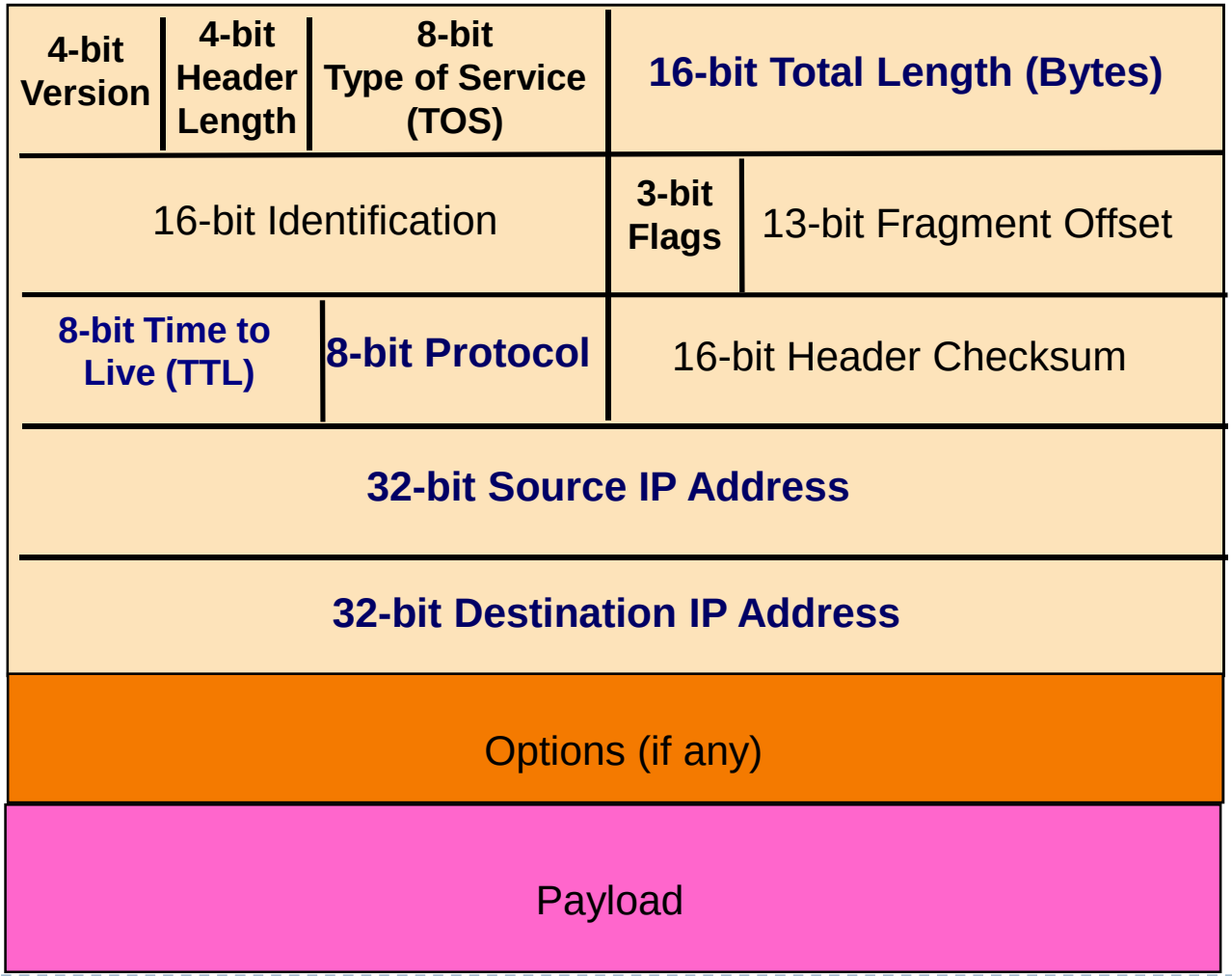
Internet Protocols: IP Routing

Routing between ASs

- Done according to the Border Gateway Protocol (BGP):
  - Each router advertises reachability information to neighbor routers for:
    - all networks within its AS and
    - for outside networks reachable via its AS (for transit ASs)
  - Reachability information includes a list of reachable networks and performance cost expressed as the number of hops to the destination
  - An AS can set routing policies that determines which reachability information is advertised to which routers

IP packet

IP Packet Structure



IP Packet

- Version number (4 bits)
  - Indicates the version of the IP protocol
    - Necessary to know what other fields to expect
    - Typically "4" (for IPv4), and sometimes "6" (for IPv6)
- Header length (4 bits)
  - Number of 32-bit words in the header
    - Typically "5" (for a 20-byte IPv4 header)
    - Can be more when "IP options" are used
- Type-of-Service (8 bits)
  - Allow packets to be treated differently based on needs
    - E.g., low delay for audio, high bandwidth for bulk transfer

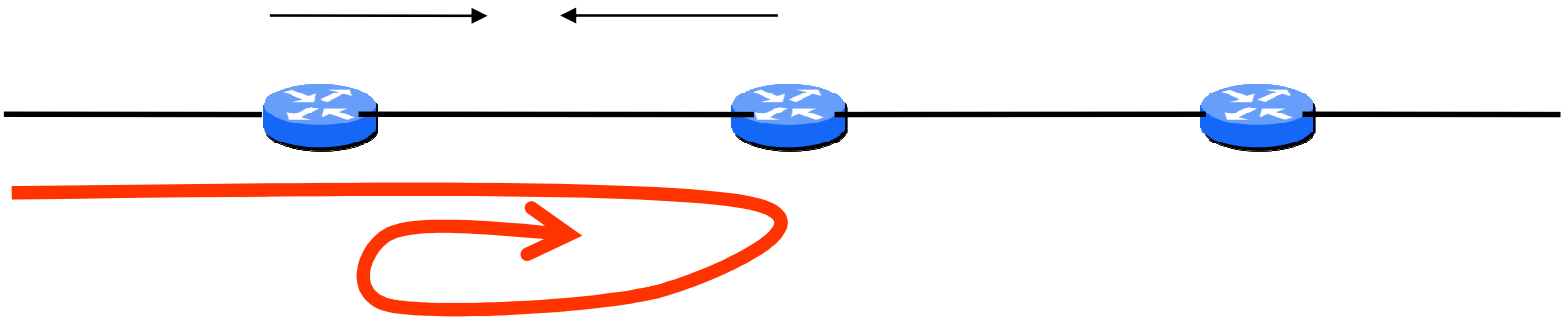
IP Packet

- Total length (16 bits)
  - Number of bytes in the packet
    - Maximum size is 63,535 bytes (2^{16} -1) … though underlying links may impose harder limits
- Fragmentation information (32 bits)
  - Packet identifier, flags, and fragment offset
    - Supports dividing a large IP packet into fragments in case a link cannot handle a large IP packet
- Time-To-Live (8 bits)
  - Used to identify packets stuck in forwarding loops and eventually discard them from the network

IP Packet

Time-to-Live (TTL) Field

- Potential robustness problem
  - Forwarding loops can cause packets to cycle forever
  - Confusing if the packet arrives much later



- Time-to-live field in packet header
  - TTL field decremented by each router on the path
  - Packet is discarded when TTL field reaches 0 and "time exceeded" message is sent to the source

IP Packet

Example Traceroute:
Berkeley to CNN

Hop number, IP address, DNS name

1	169.229.62.1	inr-daedalus-0.CS.Berkeley.EDU
2	169.229.59.225	soda-cr-1-1-soda-br-6-2
3	128.32.255.169	vlan242.inr-202-doecev.Berkeley.EDU
4	128.32.0.249	gigE6-0-0.inr-666-doecev.Berkeley.EDU
5	128.32.0.66	qsv-juniper--ucb-gw.calren2.net
6	209.247.159.109	POS1-0.hsipaccess1.SanJose1.Level3.ne
7	*	?
8	64.159.1.46	?
9	209.247.9.170	pos8-0.hsa2.Atlanta2.Level3.net
10	66.185.138.33	pop2-atm-P0-2.atdn.net
11	*	?
12	66.185.136.17	pop1-atl-P4-0.atdn.net
13	64.236.16.52	www4.cnn.com

No response from router

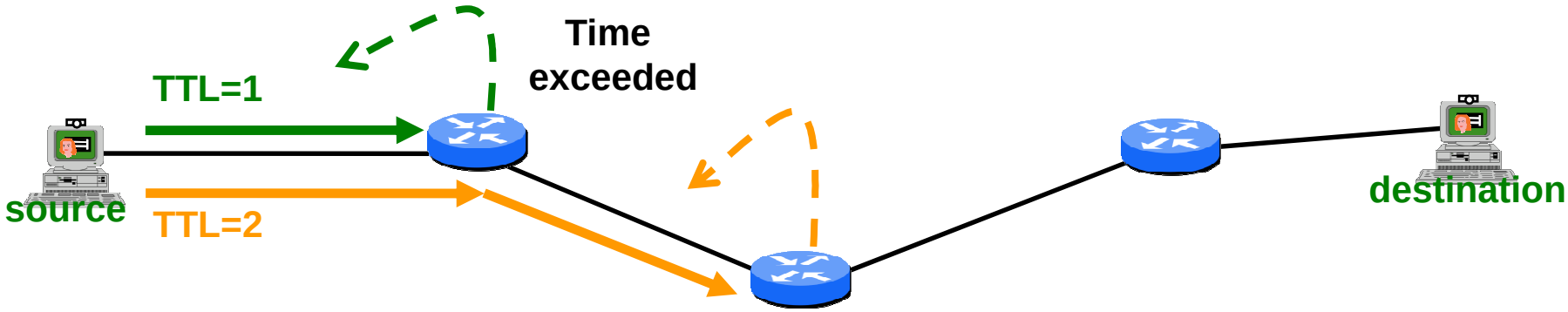
No name resolution

Try Running Traceroute Yourself

- On UNIX machine
  - Traceroute
  - E.g., "traceroute www.cnn.com" or "traceroute 12.1.1.1"
- On Windows machine
  - Tracert
  - E.g., "tracert www.cnn.com" or "tracert 12.1.1.1"
- Common uses of traceroute
  - Discover the topology of the Internet
  - Debug performance and reachability problems

IP Packet

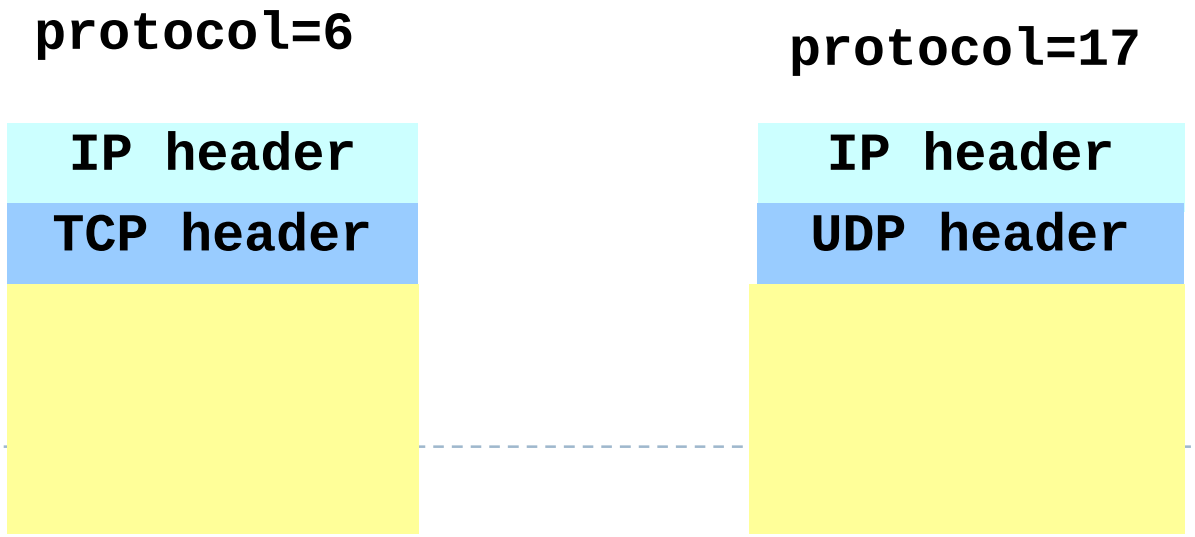
- Traceroute tool exploits this TTL behavior



Send packets with TTL=1, 2, … and record source of "time exceeded" message

IP Packet

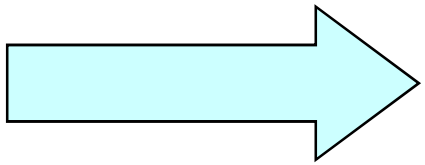
- Protocol (8 bits)
  - Identifies the higher-level protocol
    - E.g., "6" for the Transmission Control Protocol (TCP)
    - E.g., "17" for the User Datagram Protocol (UDP)
  - Important for demultiplexing at receiving host
    - Indicates what kind of header to expect next



IP Packet

- Checksum (16 bits)
  - Sum of all 16-bit words in the IP packet header
  - If any bits of the header are corrupted in transit
  - … the checksum won't match at receiving host
  - Receiving host discards corrupted packets
    - Sending host will retransmit the packet, if needed

```
  134
+ 212
-----
= 346
```



```
  134
+ 216
-----
= 350
```

Mismatch!

IP Packet

- Two IP addresses
  - Source IP address (32 bits)
  - Destination IP address (32 bits)
- Destination address
  - Unique identifier for the receiving host
  - Allows each node to make forwarding decisions
- Source address
  - Unique identifier for the sending host
  - Recipient can decide whether to accept packet
  - Enables recipient to send a reply back to source

Lecture Outline

- Main points of Lecture 3
- The Internet Protocols:
  - IP,
  - UDP,
  - TCP
- DNS (name services)

Transport Layer: UDP and TCP

Internet protocols (Transport Layer)

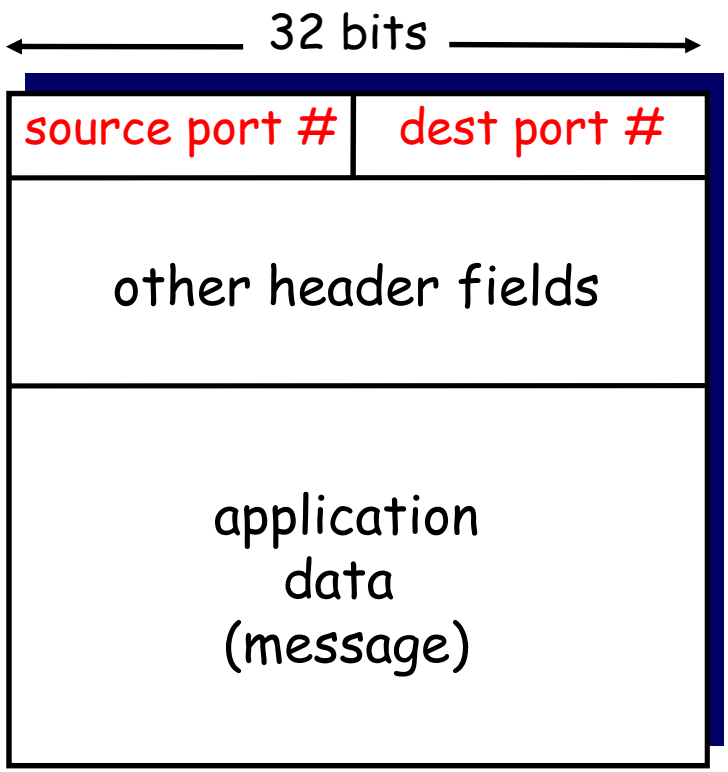
- Application layer
  - Communication for specific applications
  - *E.g., HyperText Transfer Protocol (HTTP), File Transfer Protocol (FTP), Network News Transfer Protocol (NNTP)*
- Transport layer
  - Communication between *processes* (process to process)
  - Relies on network layer and serves the application layer [*Sender*: breaks application messages into segments, and passes to network layer *Receiver*: reassembles segments into messages, passes to application layer]
  - *E.g., TCP and UDP*
- Network layer
  - Logical communication between *nodes* (host to host)
  - Hides details of the link technology
  - *E.g., IP*

Internet Transport Protocols

- Datagram messaging service (UDP)
  - No-frills extension of "best-effort" IP
- Reliable, in-order delivery (TCP)
  - Connection set-up
  - Discarding of corrupted packets
  - Retransmission of lost packets
  - Flow control
  - Congestion control

Internet protocols (Transport Layer)

- **Port numbers** are used for addressing messages to process within a particular computer and are valid only within that computer
- A port number is a 16bit integer
- Once an IP packet has been delivered to the destination host, the TCP/UDP layer software dispatches it to a process via a specific port at that host

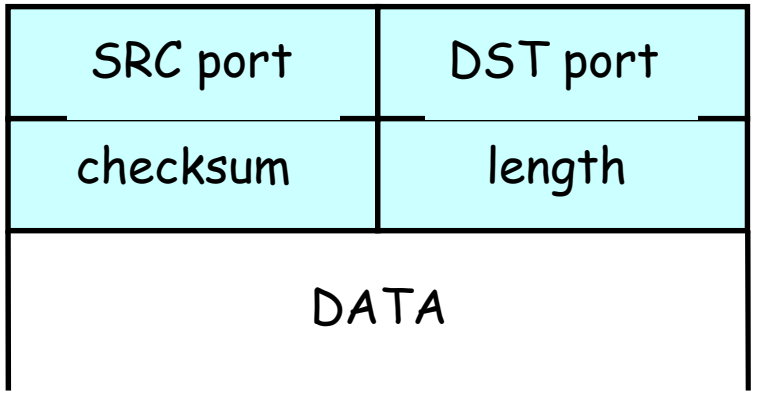


TCP/UDP segment format

Internet protocols: UDP

User Datagram Protocol (UDP)

- IP plus port numbers to support (de)multiplexing
- Optional error checking on the packet contents (if checksum field non-zero, receiver computes a check value on the packet content, if no match, the packet is dropped)



Internet protocols: UDP

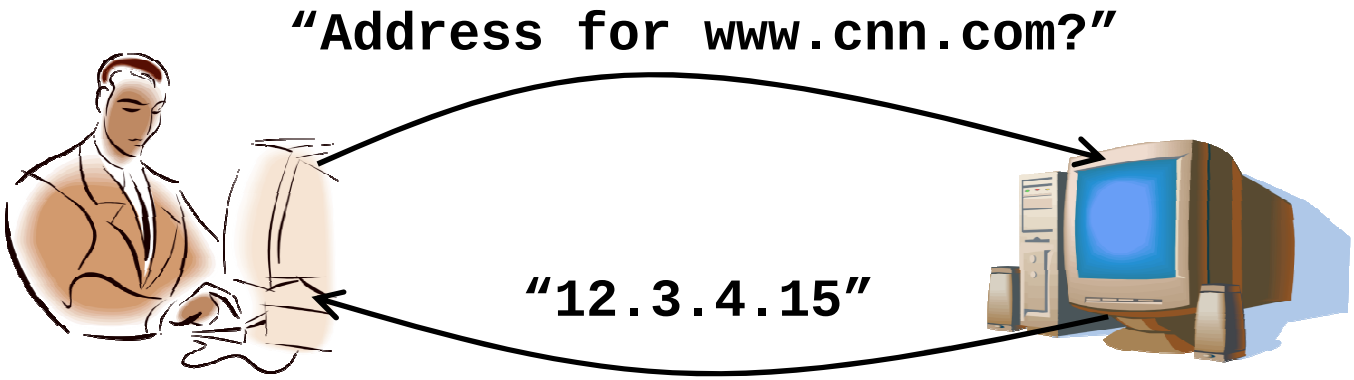
- Lightweight communication between processes
  - Avoid overhead and delays of ordered, reliable delivery
  - Send messages to and receive them from a socket

  - no guarantee of delivery (checksum is optional)
  - max of 64 bytes, same as IP
  - no setup costs, no segments, no administrative acknowledge messages

Internet protocols: UDP

- Multimedia streaming
  - Retransmitting lost/corrupted packets is not worthwhile (By the time the packet is retransmitted, it's too late (e.g., telephone calls, video conferencing, gaming)

- Simple query protocols like Domain Name System
  - Overhead of connection establishment is overkill
  - Easier to have application retransmit if needed



Internet protocols: TCP

Transmission Control Protocol (TCP)

Reliable delivery of arbitrarily long sequences of byte via a *stream-based* abstraction

Connection-oriented: *before* any data transfer, the sender and the receiver must cooperate in the establishment of a bidirectional communication channel

End-to-end agreement (intermediate nodes and routers have no knowledge)

TCP: Basic concepts

Connection oriented

- Explicit set-up and tear-down of TCP session
- State maintained at both hosts

Stream-of-bytes service

- Sends and receives a stream of bytes, not messages (packets)
- Division of data into datagrams, headers etc are invisible to the application above

Reliable, in-order delivery

- Checksums to detect corrupted data
- Acknowledgments & retransmissions for reliable delivery
- Sequence numbers to detect losses and reorder data

Full duplex transfer

- Allows data transfer between two applications in both directions at the same time

TCP: Basic concepts

Flow control

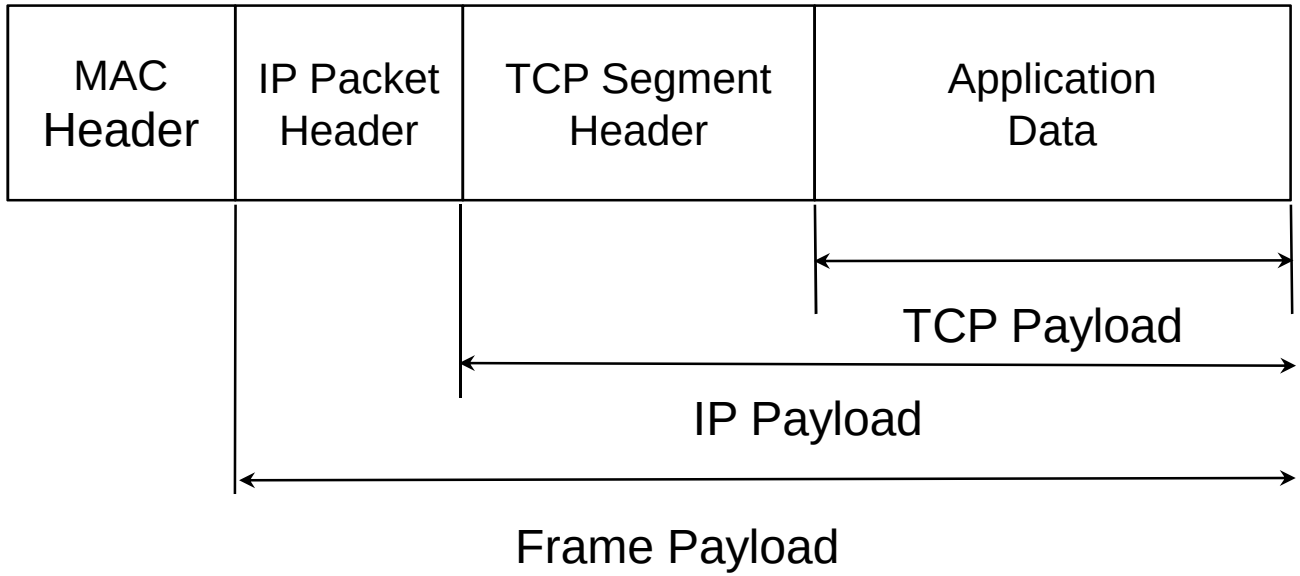
- The sender takes care not to overwhelm the receiver or the intervening nodes (overflow their buffer space)

Congestion control

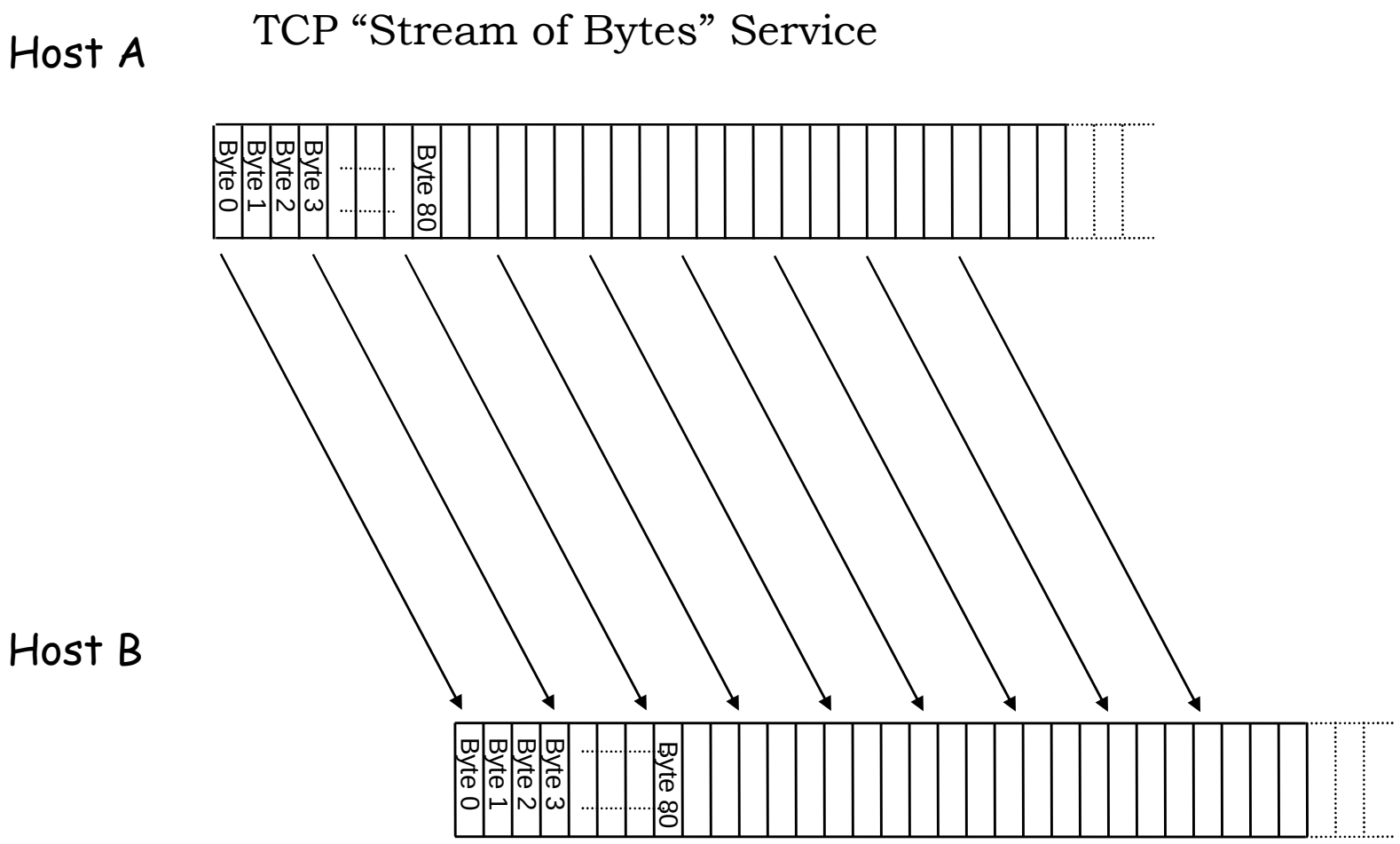
- Regulates the rate at which the network can transfer the data

TCP: Segments and sequencing

- TCP breaks the data stream into ***segments***, which the network layer encapsulates into IP datagrams.



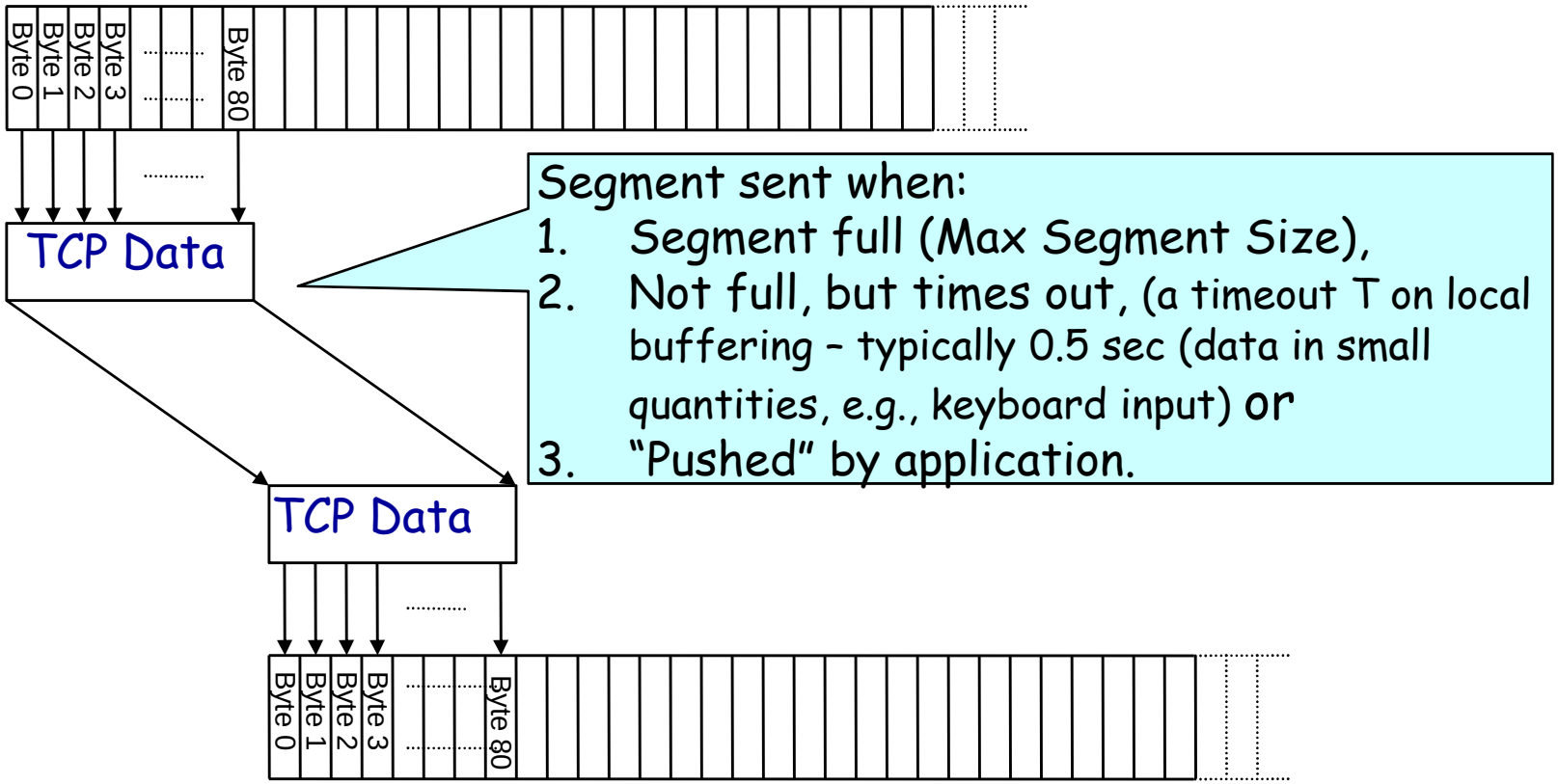
TCP: Segments and sequencing



TCP: Segments and sequencing

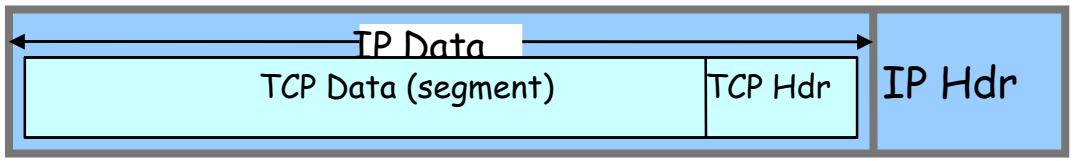
…Emulated Using TCP "Segments"

Host A



Host B

TCP: Segments and sequencing



- IP packet
  - No bigger than Maximum Transmission Unit (MTU)
    - E.g., up to 1500 bytes on an Ethernet
- TCP packet
  - IP packet with a TCP header and data inside
    - TCP header is typically 20 bytes long
- TCP segment
  - No more than *Maximum Segment Size (MSS)* bytes
    - E.g., up to 1460 consecutive bytes from the stream

TCP: Segments and sequencing

A **sequence number** is assigned to each TCP segment

- The *receiver* uses the sequence number to order the received segments before placing them in the input queue for the receiving process
  - No segments can be placed in the input stream until all lowered-numbered segments have been received and placed in the stream
  - Out of order segments are held in buffer

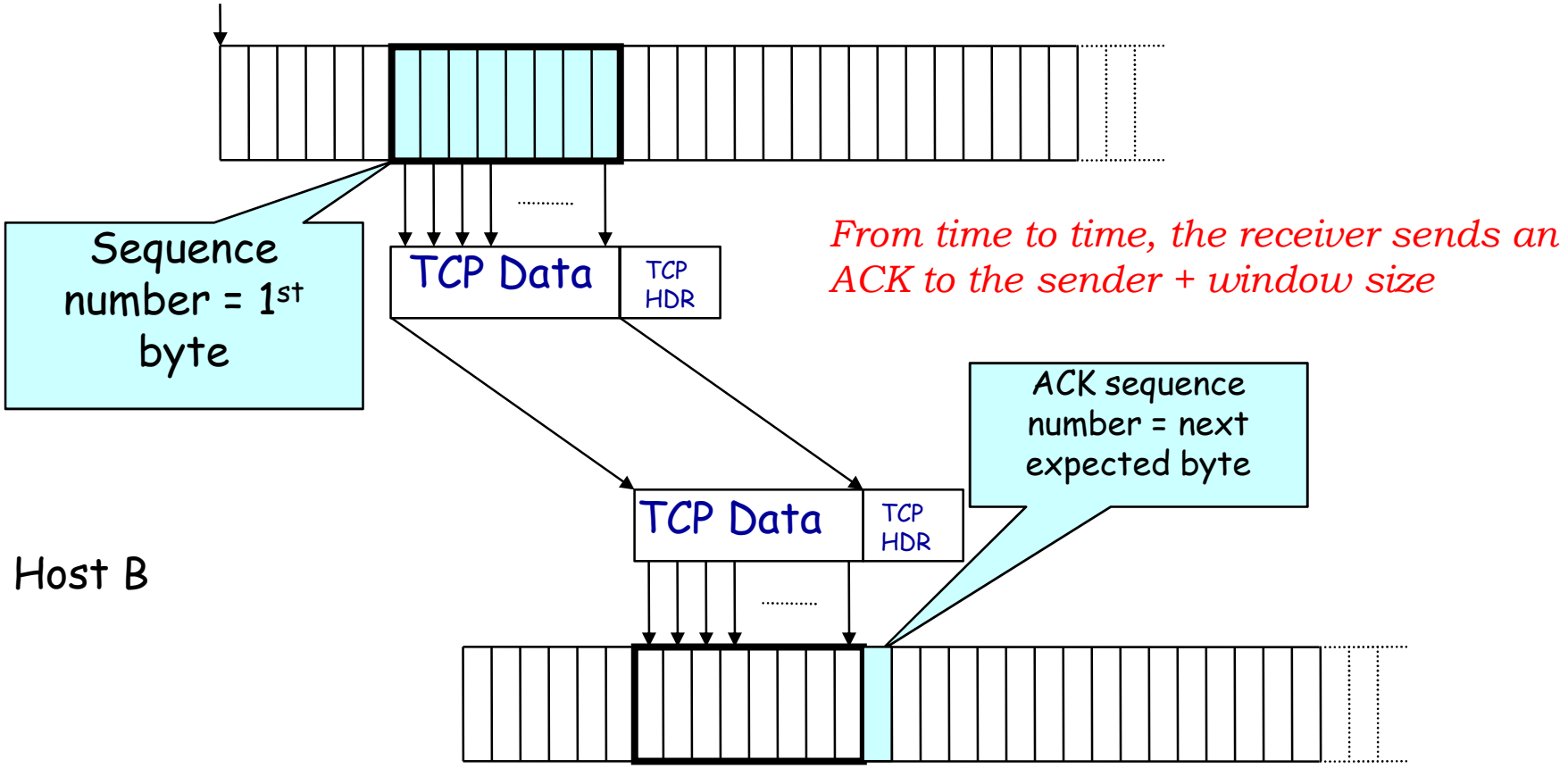
Sequence number = the byte number within the stream for the first byte in the segment

TCP: Segments and sequencing

Host A

ISN (initial sequence number)

sequence number = the byte number within the stream for the first byte in the segment
Used by the receiver



TCP: Segments and sequencing

How to choose the Initial Sequence Number (ISN)?

(i.e., sequence number for the very first byte)

Why not a de facto ISN of 1?

- It introduces the possibility of segments from different connections getting mixed up.

1. Suppose we established a TCP connection and sent a segment containing bytes 1 through 30.
2. A problem with the internetwork caused this segment to be delayed, and eventually, the TCP connection itself to be terminated.
3. We then started up a new connection (same IP and port) and again used a starting sequence number of 1.
4. As soon as this new connection was started, however, the old segment with bytes labeled 1 to 30 showed up.
5. The other device would erroneously think those bytes were part of the *new* connection.

Each TCP device, at the time a connection is initiated, chooses a 32-bit *ISN* for the connection. How?

TCP: Segments and sequencing

Traditionally, each device chose the ISN from a timed counter

- Counter initialized to 0 when TCP started up and
- Then its value increased by 1 every 4 microseconds until it reached the largest 32-bit value possible (4,294,967,295) at which point it "wrapped around" to 0 and resumed incrementing.
- Any time a new connection is set up, the ISN was taken from the current value of this timer.

Since it takes over 4 hours to count from 0 to 4,294,967,295, this virtually assured that each connection will not conflict with any previous ones.

But, it makes ISNs predictable.

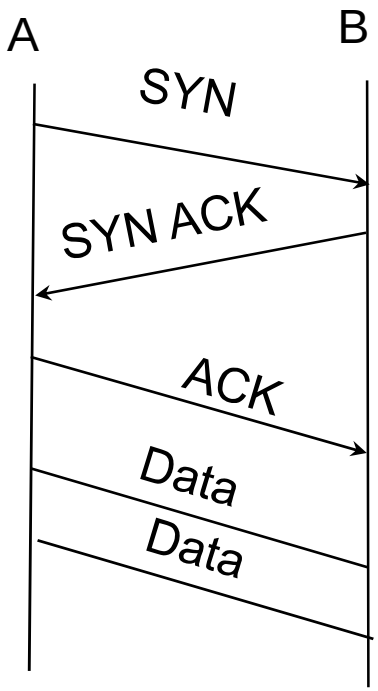
A malicious person write code to analyze ISNs and then predict the ISN of a subsequent TCP connection based on the ISNs used in earlier ones.

Implementations now use a random number in their ISN selection process.

TCP: 3-way handshake

Three-way handshake to establish connection

1. Host A sends a SYN (open) to the host B
2. Host B returns a SYN acknowledgment (SYN ACK)
3. Host A sends an ACK to acknowledge the SYN ACK

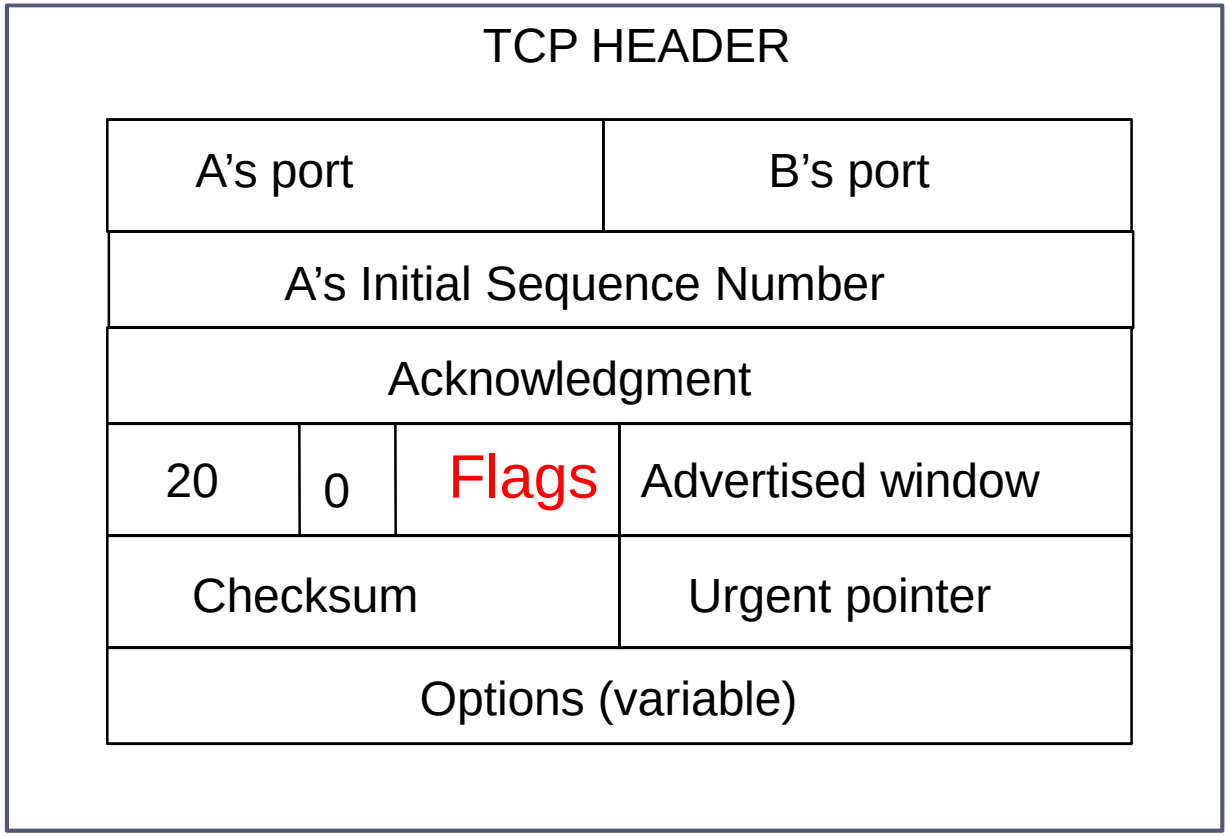


Each host tells its ISN to the other host.

TCP: 3-way handshake

STEP 1

Flags: SYN
FIN
RST
PSH
URG
ACK

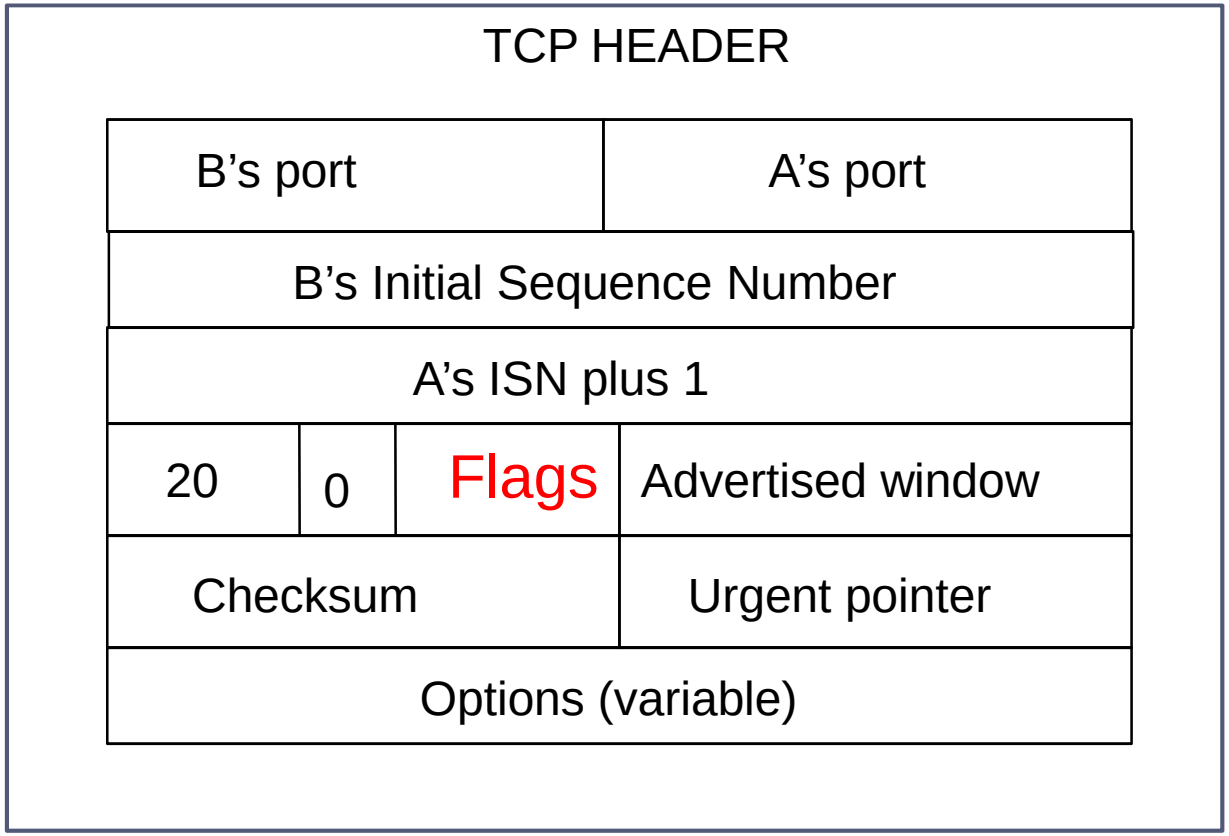


A tells B it wants to open a connection…

TCP: 3-way handshake

STEP 2

Flags: SYN
FIN
RST
PSH
URG
ACK



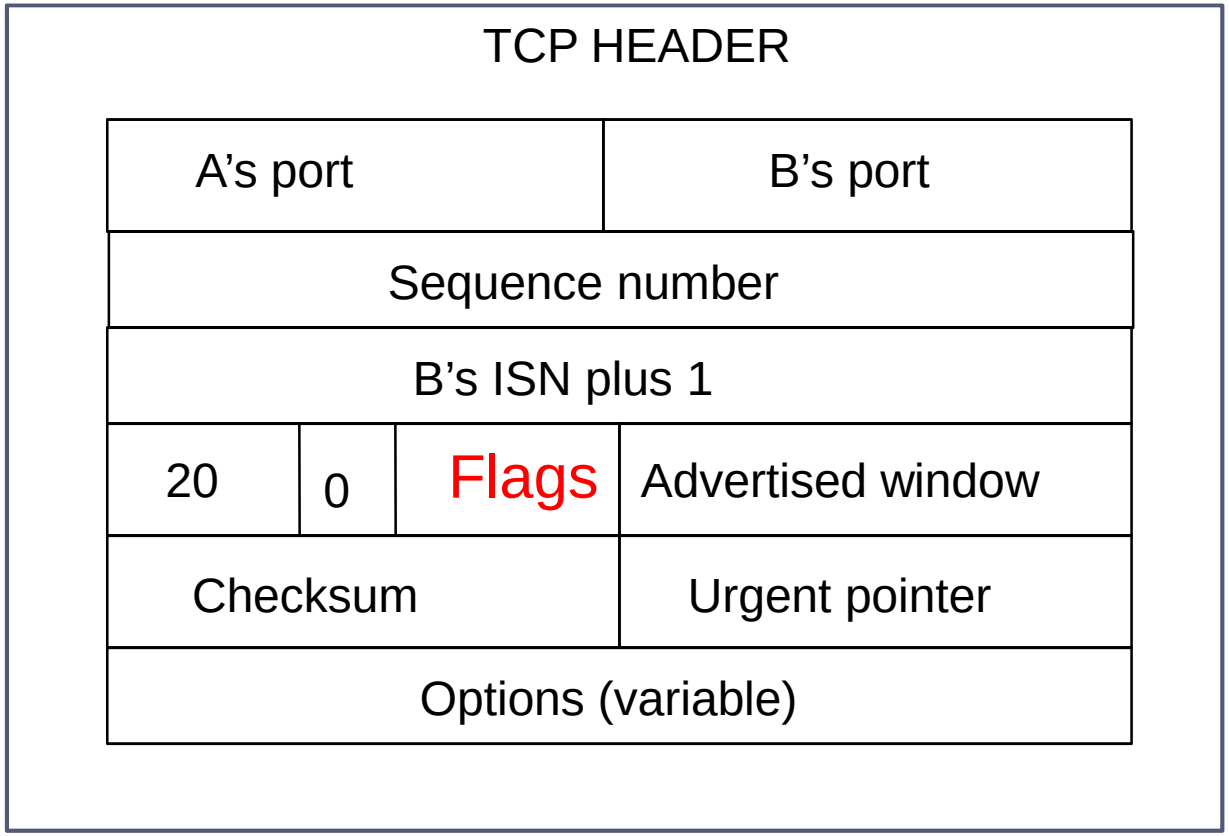
B tells A it accepts, and is ready to hear the next byte…

… upon receiving this packet, A can start sending data

TCP: 3-way handshake

STEP 3

Flags: SYN
FIN
RST
PSH
URG
ACK



A tells B it is okay to start sending

… upon receiving this packet, B can start sending data

TCP: 3-way handshake

What if the SYN Packet Gets Lost?

- Suppose the SYN packet gets lost
  - Packet is lost inside the network, or
  - Server rejects the packet (e.g., listen queue is full)
- Eventually, no SYN-ACK arrives
  - Sender sets a timer and wait for the SYN-ACK
  - … and retransmits the SYN-ACK if needed
- How should the TCP sender set the timer?
  - Sender has no idea how far away the receiver is
  - Hard to guess a reasonable length of time to wait
  - Some TCPs use a default of 3 or 6 seconds

TCP: 3-way handshake

SYN Loss and Web Downloads

- User clicks on a hypertext link
  - Browser creates a socket and does a "connect"
  - The "connect" triggers the OS to transmit a SYN
- If the SYN is lost…
  - The 3-6 seconds of delay may be very long
  - The user may get impatient
  - … and click the hyperlink again, or click "reload"
- User triggers an "abort" of the "connect"
  - Browser creates a new socket and does a "connect"
  - Essentially, forces a faster send of a new SYN packet!
  - Sometimes very effective, and the page comes fast

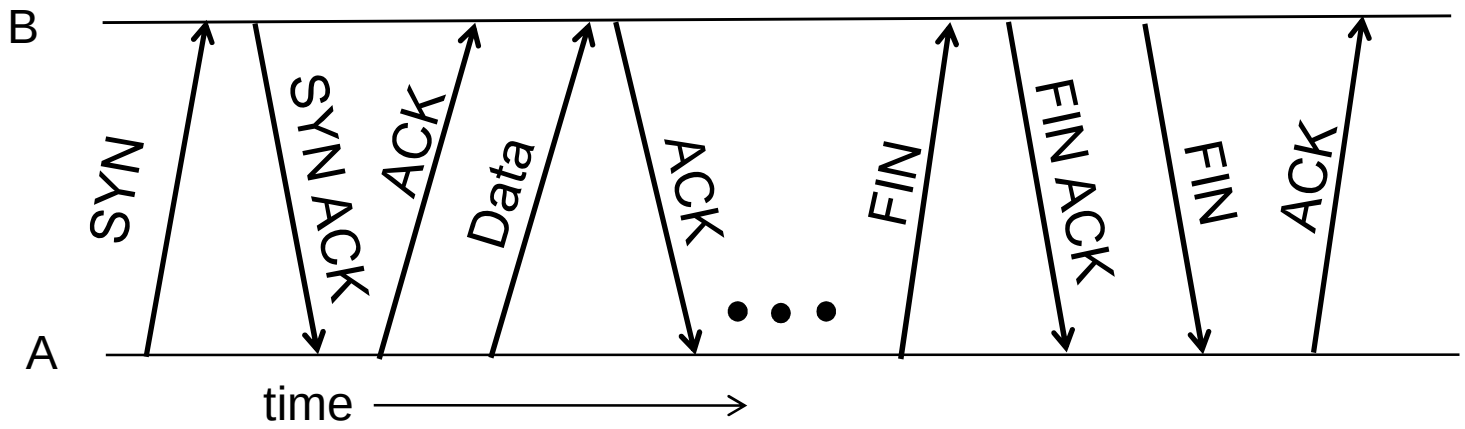
TCP: 3-way handshake

Performance implications

- TCP performance is suboptimal for transfers of small Web objects:
  - nearly all TCP implementations send the first data after the handshake is completed
  - this adds an additional *round-trip time (RTT)* to the application-layer data transfer

TCP: 3-way handshake

Tearing Down the Connection



- Closing the connection
  - Finish (FIN) to close and receive remaining bytes
  - And other host sends a FIN ACK to acknowledge
  - Reset (RST) to close and not receive remaining bytes

TCP: 3-way handshake

- Sending a FIN: close()
  - Process is done sending data via the socket
  - Process invokes "close()" to close the socket
  - Once TCP has sent all of the outstanding bytes then TCP sends a FIN

Receiving a FIN: EOF

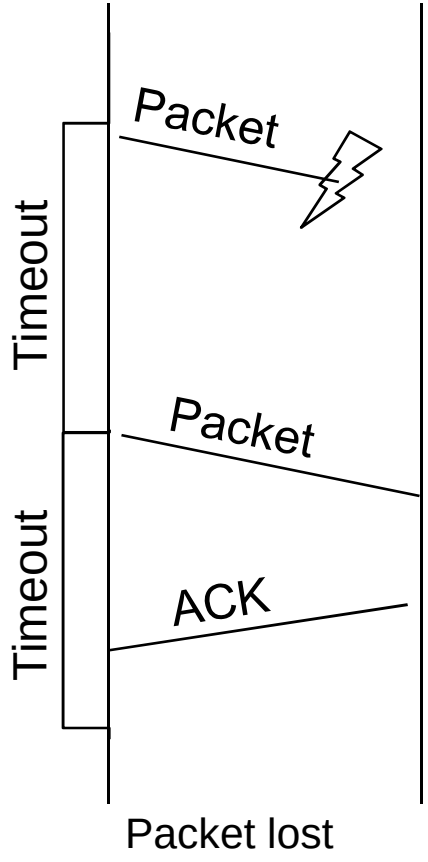
- Process is reading data from the socket
- Eventually, the attempt to read returns an EOF

TPC: Retransmissions

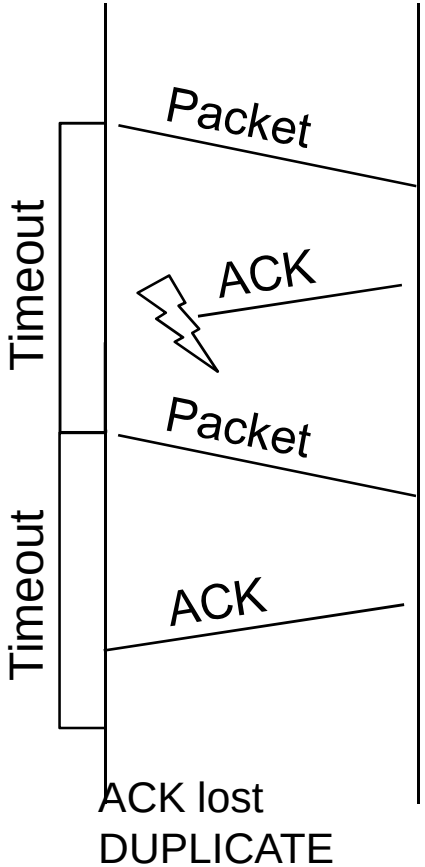
- A mechanism to deal with unreliability (loss of packets)
- TCP sender *retransmits segments that have not been acknowledged* by the receiver within a certain time after the initial transmission (timeout)
- The timeout value is adapted according to previously observed RTT (round trip time) and RTT variance between two communicating hosts
- Initial timeout value set to a predefined value

TPC: Retransmissions

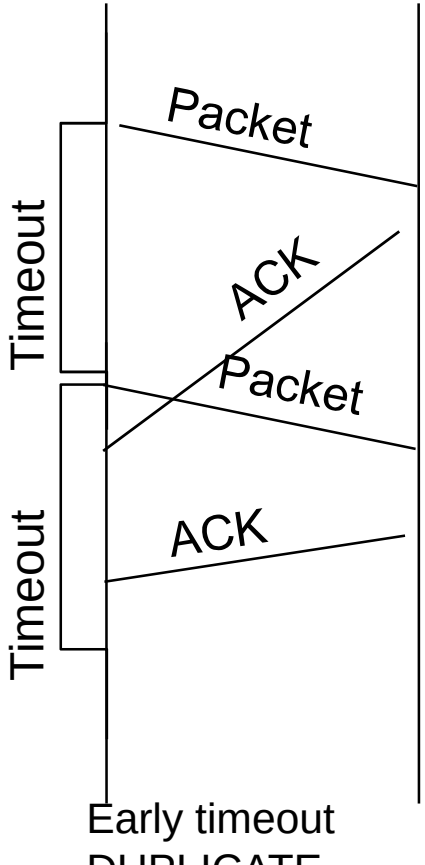
Reasons for Retransmission



Packet lost



ACK lost
DUPLICATE
PACKET



Early timeout
DUPLICATE
PACKETS

TCP: Flow control

Flow control limits the transmission rate to a rate that the receiver can absorb the data

How?

1. TCP makes the sender wait for an ACK after transmitting a certain amount of data

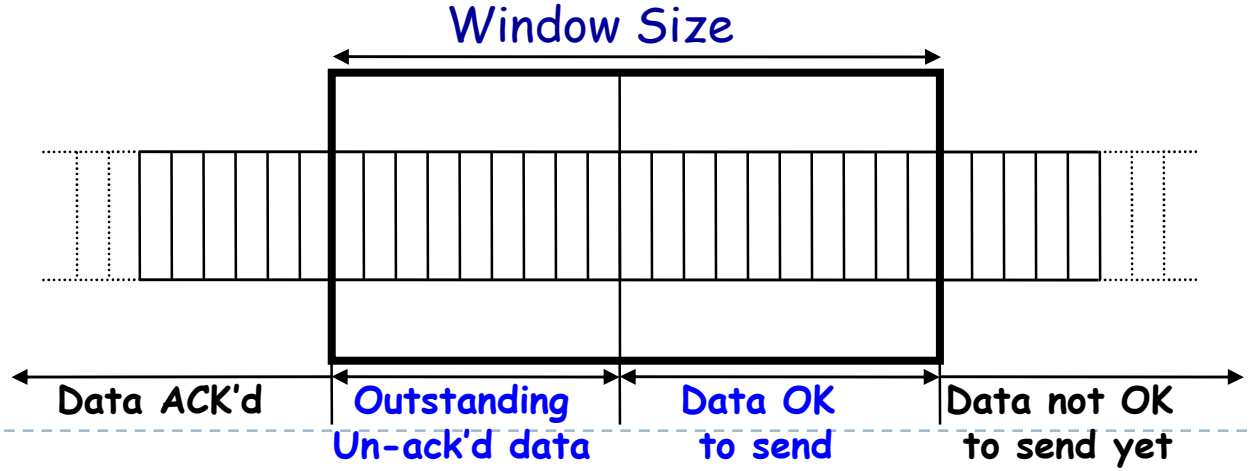
- To decide how much data to send before waiting, TCP uses the ***sliding window*** protocol

2. With every ACK segment, the receiver advertises a window in bytes, using the WINDOW header field
   - *this window size indicates how many more bytes the receiver is able to accept into its buffers*

TCP: Flow control

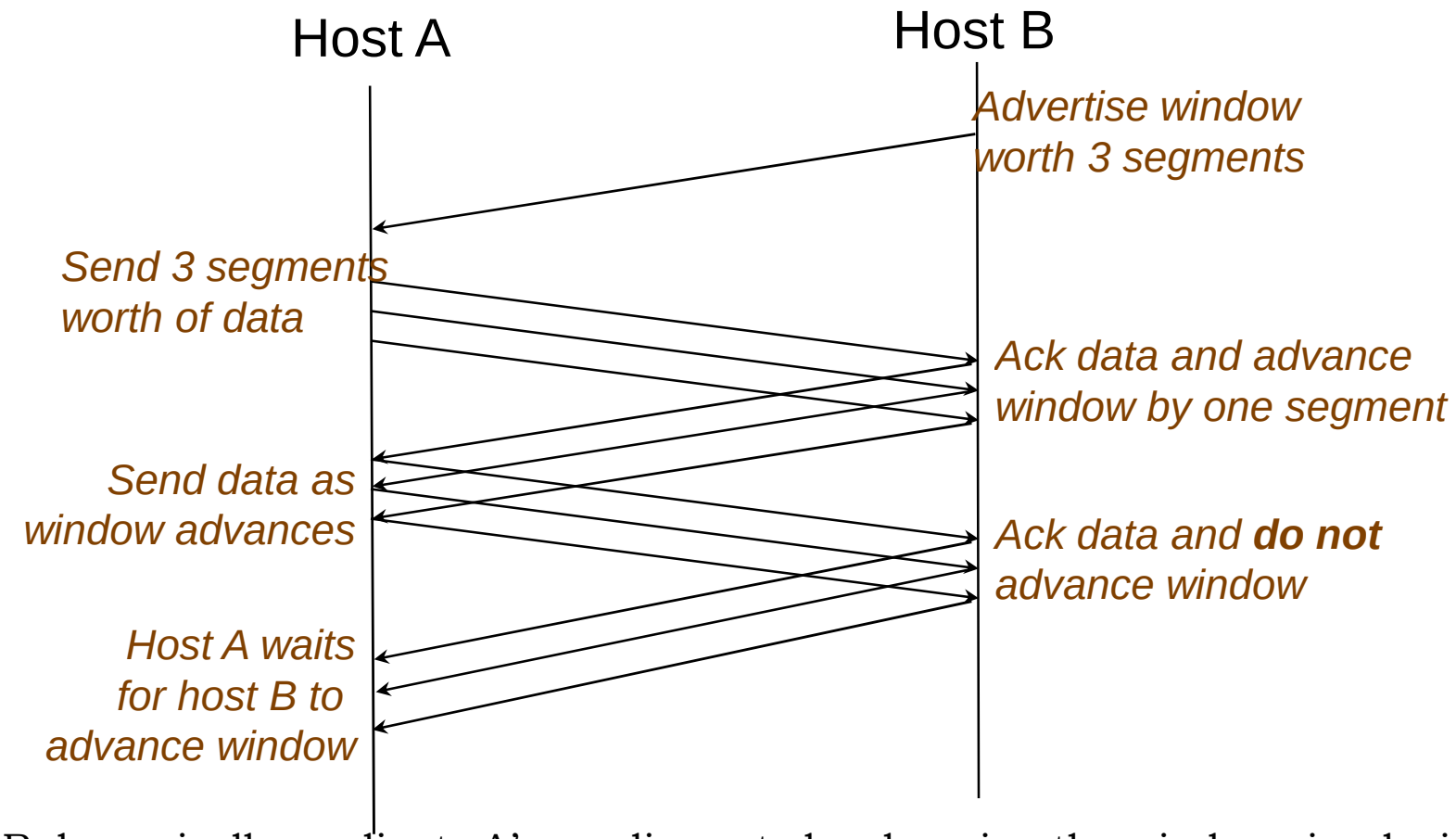
Receiver Buffering

- Window size
  - Amount that can be sent without acknowledgment
  - Receiver needs to be able to store this amount of data
- Receiver advertises the window to the sender
  - Tells the sender the amount of free space left
  - and the sender agrees not to exceed this amount



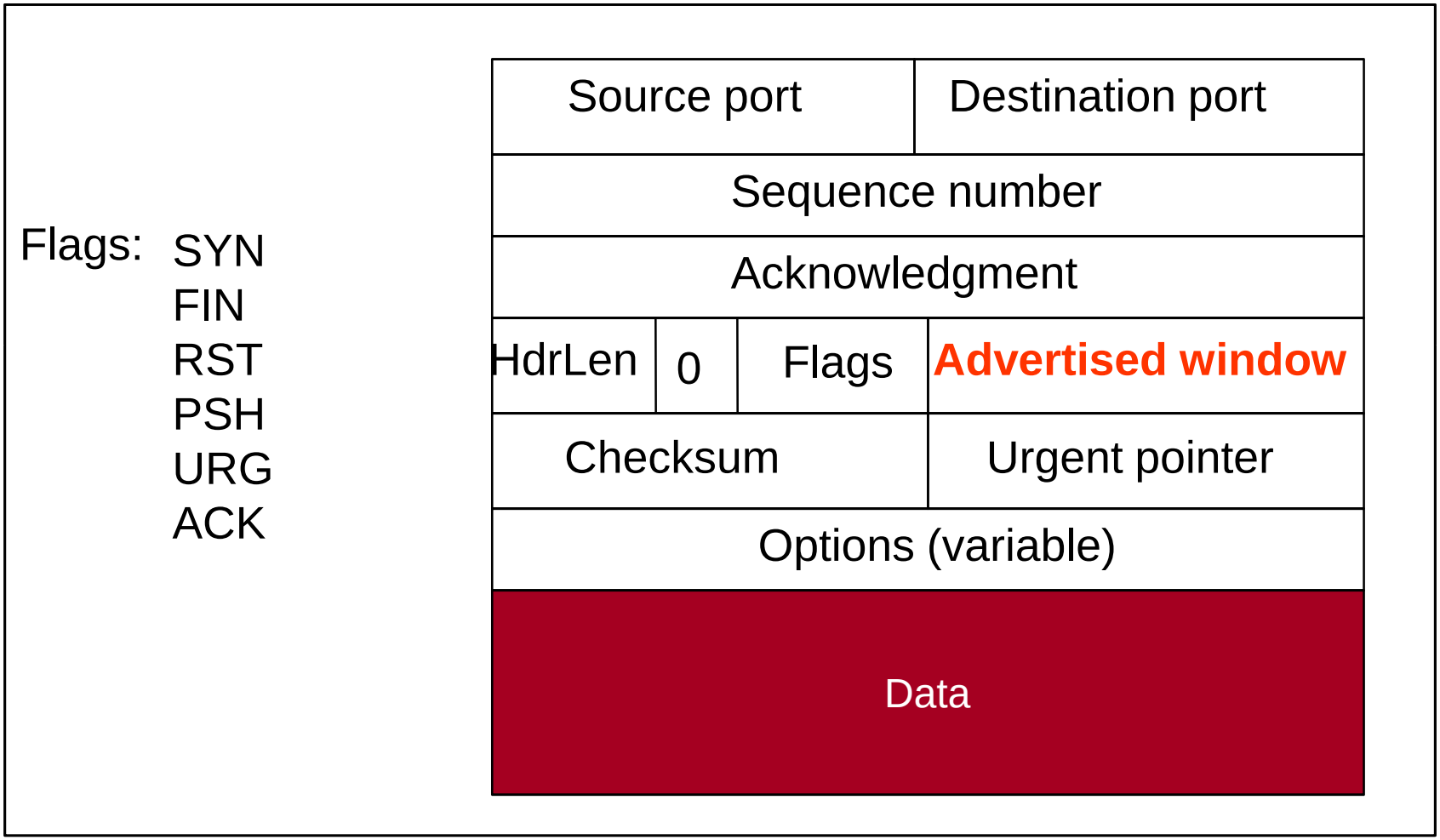
TCP: Flow control

Example of TCP sliding window



B dynamically readjusts A's sending rate by changing the window size during transmission

TCP: Flow control



TCP congestion control

- Applied in order to avoid packet loss and re-transmissions that further increase the congestion.
  - Timeout or dup ACKs
- TCP congestion control mechanism:
  - Congestion window (cwnd): specifies the *number of unacknowledged segments* one host may send to the other
- Basic idea:
  - sender monitors the loss of packets; when this occurs, the sender reduces quickly (multiplicatively) its transmission rate.
  - if there are no lost packets, sender increases slowly its transmission rate – $segsize^2$ / cwnd
  - linear increase, multiplicative decrease algorithm

Slow start mechanism

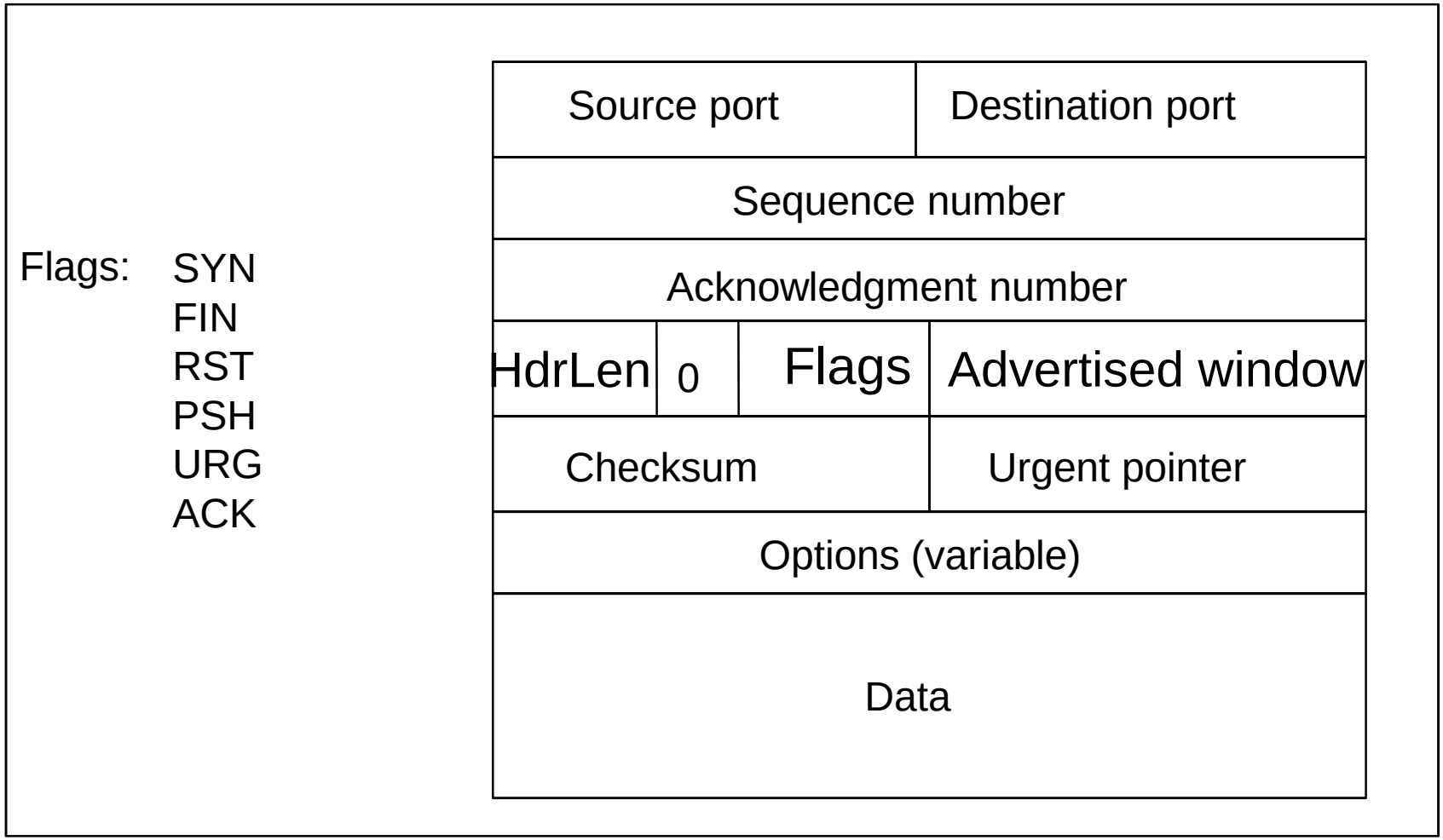
- Slow start and Congestion avoidance are implemented together
- Start with a small transfer rate (allow only two unacknowledged segments)
- Initially, increase the congestion window by one segment for each acknowledged segment:
  - this leads to an exponential increase of transfer rate
- If a packet is lost, reduce drastically the transfer rate by *reducing the congestion window by half*
- When receiving ACKs, increase cwnd
  - If *cwnd < half the window of congestion,* do slow start
  - else, do congestion avoidance

Performance implications

- The throughput of one large TCP transfer is usually higher than the throughput of many short ones, with the exception of very short connections that fit into the initial congestion window of two segments (~3KB)

- Short TCP transfers are affected mostly by the TCP handshake overhead, which cannot be amortized over the life of a connection

TCP Header


Flags:
SYN
FIN
RST
PSH
URG
ACK
Source port
Destination port
Sequence number
Acknowledgment number
HdrLen
0
Flags
Advertised window
Checksum
Urgent pointer
Options (variable)
Data

TCP Header Fields

- Source and Destination port numbers: for multiplexing/de-multiplexing at the hosts
- Sequence number: the offset of a segment in a byte stream (the sequence number of the first segment byte in the byte stream)
- Acknowledgment number: confirms that the sender of the segment has received all bytes up to this number from the other host (valid when the ACK bit is set)
- Code bits field (flags): deal with connection management and urgency of the content of the segment (SYN, FIN, RST, ACK)
- Window field: used for flow control

Internet protocols: TCP

- arbitrarily long sequence
- connection-oriented
- sequencing of segments
- flow control: acknowledgement includes "window size" (amount of data) for sender to send before next ack
- interactive service: higher frequency of buffer flush, send when deadline reached or buffer reaches MTU
- retransmission of lost packets
- buffering of incoming packets to preserve order and flow
- checksum on header and data

Lecture Outline

- Main points of Lecture 3
- The Internet Protocols:
  - IP,
  - UDP,
  - TCP
- DNS (name services)

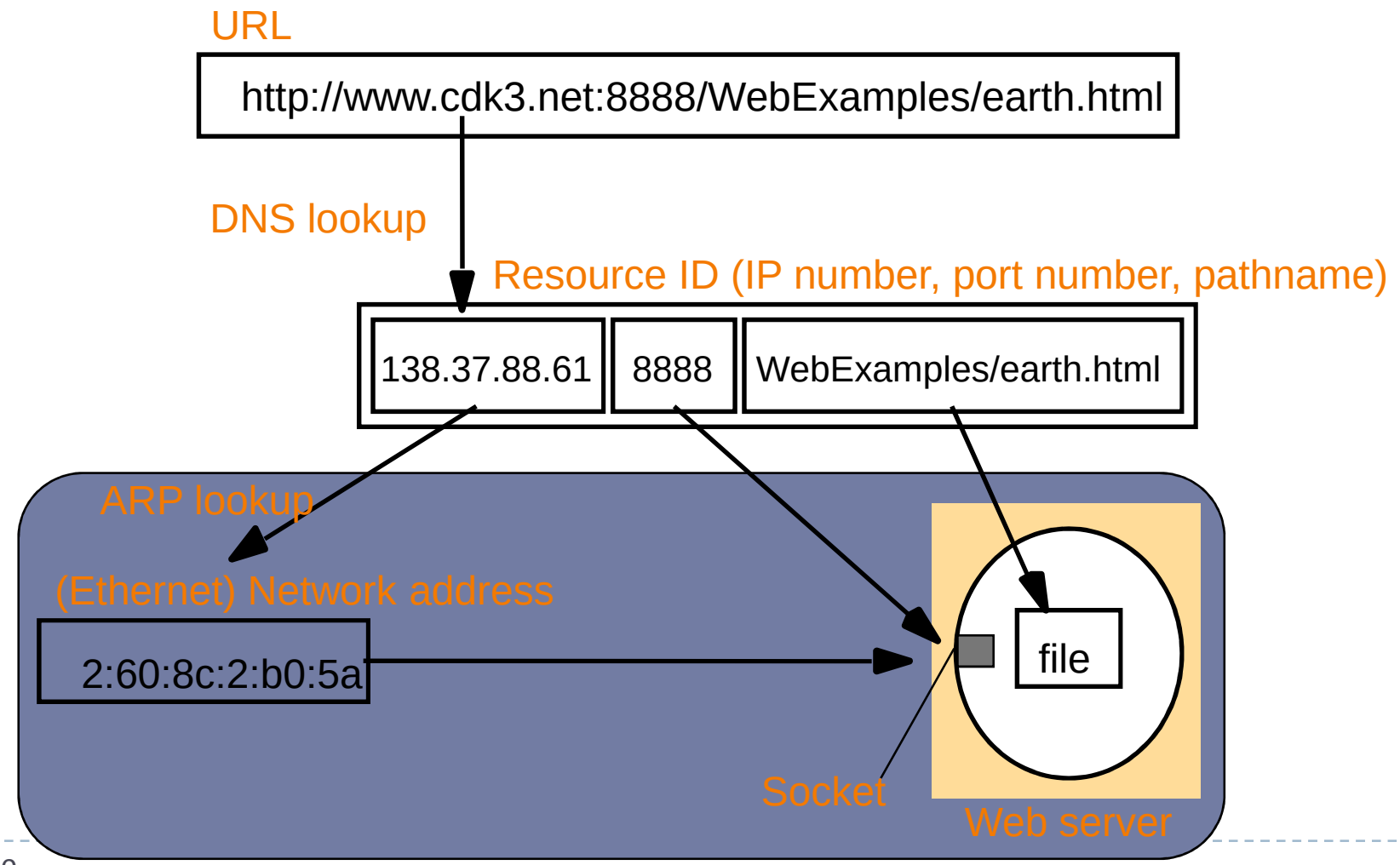
Name Services: DNS

Names, addresses and other attributes

- Resources or objects (web pages, files, nodes, etc) are accessed using *identifier, names, or addresses*
  - The term *identifier* is sometimes used to refer to names that are interpreted only by programs
  - A *name* is human-readable value (usually a string) that can be resolved to an identifier or address
  - *Address* identifies the location of the resource rather than the resource itself
  - Other descriptive attributes (values of some property associated with the an object (address may consider a key such attribute)

▪ A name is resolved when it is translated into data about the named resource or object by a name service
▪ Binding: the association between an object and a resource

Names, addresses and other attributes



Names, addresses and other attributes

Currently, different name systems are used for each type of resource:

resource	***name***	***identifies***
File	pathname	file within a given file system
Process	process id	process on a given computer
Port	port number	IP port on a given computer

Names, addresses and other attributes

- *Uniform Resource Identifiers* (URIs) offer a general solution for any type of resource.

- There two main classes:
  - URL Uniform Resource Locator
    - typed by the protocol field (http, ftp, nfs, etc.)
    - part of the name is service-specific
    - resources cannot be moved between domains
  - URN Uniform Resource Name
    - requires a universal resource name lookup service - a DNS-like system for all resources

Name services

A *name service* stores information about a collection of textual names in the form of binding between the names and the attributes of the entities they denote, such as users, computers, services and objects

The collection is subdivided into one or more *naming contexts*: individual subsets of the binding that are managed as a unit

Main operation: *Name resolution* (look up attributes for a given name)

Name spaces

Name space: the collection of all valid names recognized by a particular service

Follow a syntax (valid vs invalid)

Hierarchical vs Flat name space

(+) more manageable

each part is resolved relative to a separate context of relatively small size and the same name may be used with different names in different contexts (e.g., filesystem, URLs, etc)

(+) different contexts may be managed by different people or organizations

(+) trust

(+) updates, allow restructure of subtrees

(+) potentially infinite, so a system may grow indefinitely

Name spaces

Alias: a name defined to denote the same information as another name

e.g., symbolic links in file systems
URL shorteners in Twitter posts, e.g., http://bit.ly/ctqjvH

Talk this Friday (Feb 10), 15:00-16:00
Demetris Antoniades, "**We.b: The web of short URLs**"

Naming domain: a name space for which there exists a single administrative authority for assigning names within it

Name resolution

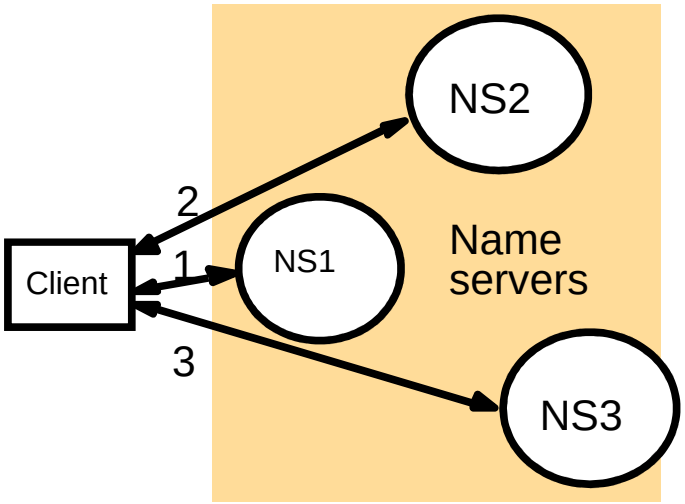
Data is partitioned into servers according to its domain

The process of locating naming data from more than one name server in order to resolve a name is called *navigation*

Iterative navigation vs recursive navigation

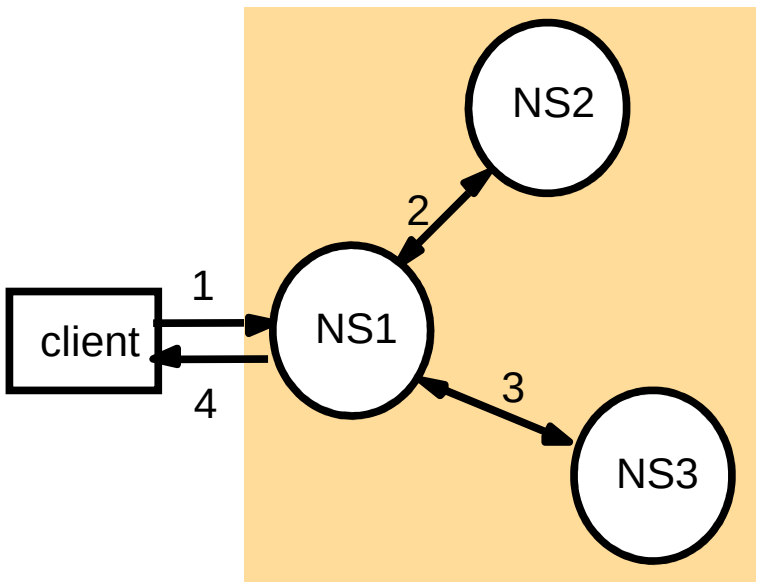
Multicast navigation

Name resolution

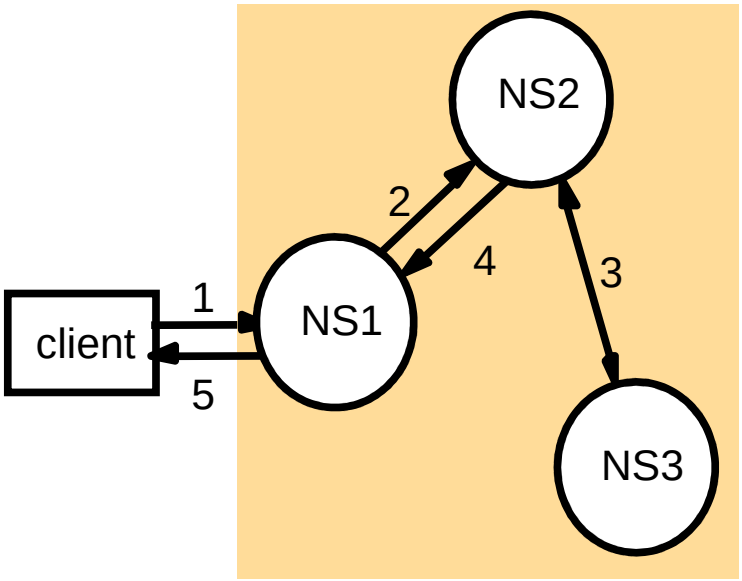


A client iteratively contacts name servers NS1–NS3 to resolve a name

Name resolution



Non-recursive
server-controlled



Recursive
server-controlled

A name server NS1 communicates with other name servers on behalf of a client

Name resolution

Caching

Client name resolution software and servers maintain a cache of the results of previous name resolutions

- Eliminate load from high-level name servers (root)
- Availability
- Communication cost

The Domain Name System (DNS)

Internet supports a scheme for the use of symbolic names for hosts and networks

Domain Name System: host names -> IP addresses

Organized into a naming hierarchy

Hierarchy reflect organizational structure

Independent of the physical layer

The Domain Name System (DNS)

- Host names
  - Mnemonic name appreciated by humans
  - Variable length, alpha-numeric characters
  - Provide little (if any) information about location
  - Examples: www.cnn.com and ftp.eurocom.fr
- IP addresses
  - Numerical address appreciated by routers
  - Fixed length, binary number
  - Hierarchical, related to host location
  - Examples: 64.236.16.20 and 193.30.227.161

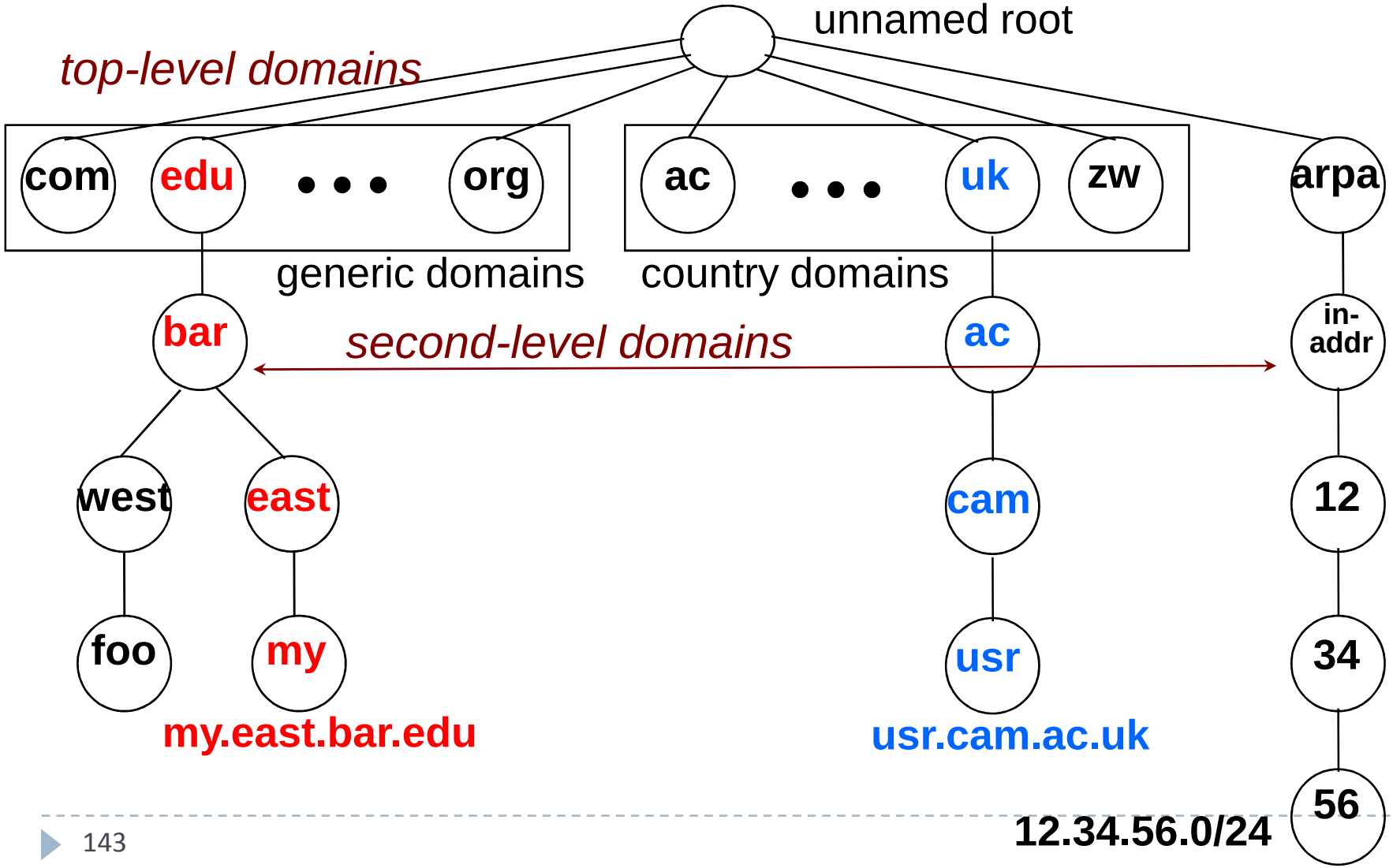
The Domain Name System (DNS)

Separating Naming and Addressing

- Names are easier to remember
  - www.cnn.com vs. 64.236.16.20
- Addresses can change underneath
  - Move www.cnn.com to 64.236.16.20
  - E.g., renumbering when changing providers
- Name could map to multiple IP addresses
  - www.cnn.com to multiple replicas of the Web site
- Map to different addresses in different places
  - Address of a nearby copy of the Web site
  - E.g., to reduce latency, or return different content
- Multiple names for the same address
  - E.g., aliases like ee.mit.edu and cs.mit.edu

The Domain Name System (DNS)

Distributed Hierarchical Database



The Domain Name System (DNS)

- Central server
  - One place where all mappings are stored
  - All queries go to the central server
- Many practical problems
  - Single point of failure
  - High traffic volume
  - Distant centralized database
  - Single point of update
  - Does not scale

Need a distributed, hierarchical collection of servers

DNS name servers

The DNS database is *distributed* across a **logical network of servers**
Each server holds **part** of the naming database (primarily data from its local domain)

Client-server with multiple servers
DNS Client, called resolver

- Accepts queries
- Formats them to messages according to the DNS protocol
- Communicates with one or more name servers (list in order of preference)

Request-reply protocol + UDP

- Queries for hosts in the local domain are satisfied by servers within that domain
- Each server also records the domain names and addresses of other name servers to satisfy queries outside the domain

DNS name servers: partition

DNS naming data are divided into *zones*

Name servers maintain data for one or more zones

Authoritative data: data that can be relied upon as being up-to-date

A zone contains
- Attribute data for names in its domain except any subdomains administered by lower-level authorities
- Name and addresses of at least two name servers that provide *authoritative data* for the zone
- The names of name servers that hold authoritative data for delegated subdomains
- Zone-management parameters (e.g., for caching)

Each zone must be replicated authoritatively in at least two servers
Primary (master) server and secondary servers

DNS name servers

Data are cached with a time-to-live value

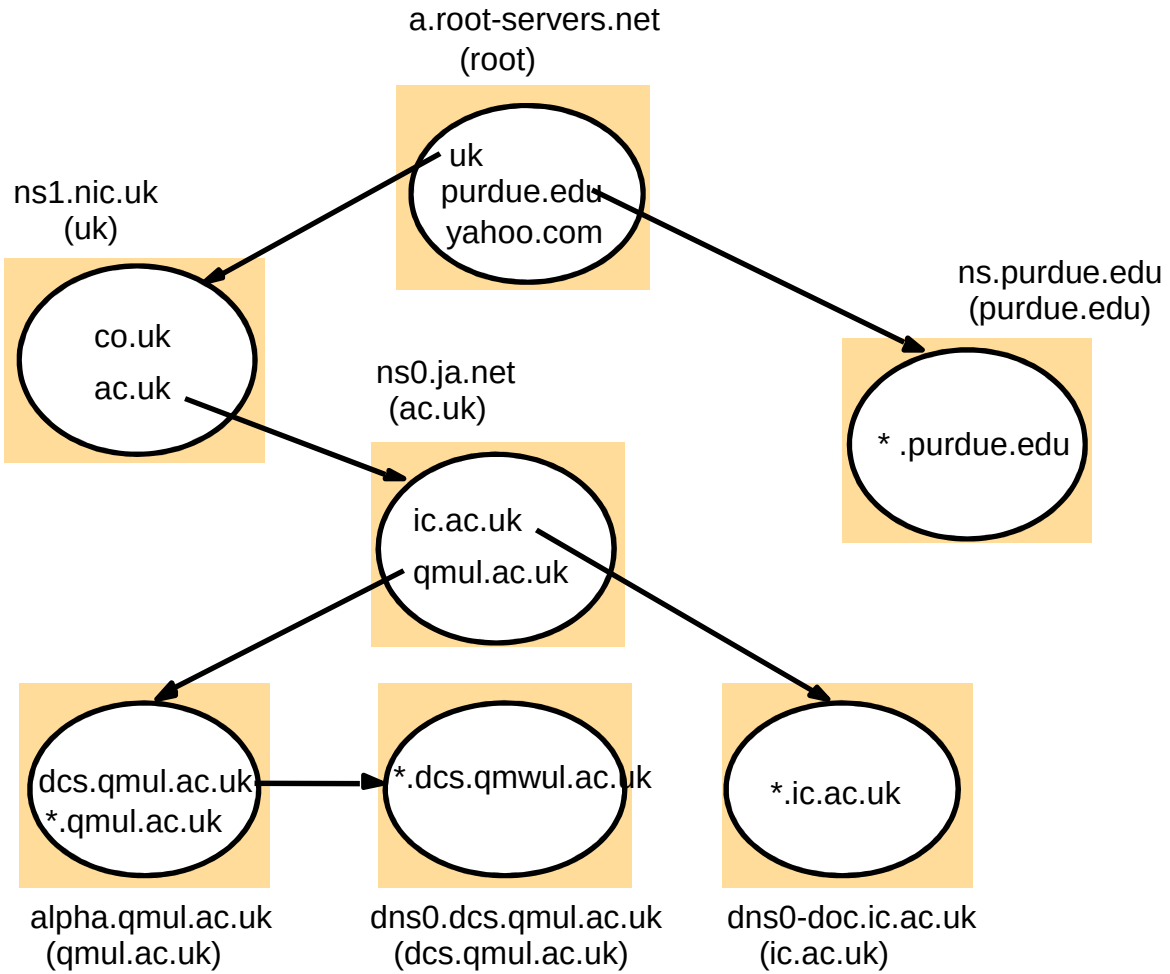
When it expires, contact the server

All DNS servers store the address of one or more root name servers

The Domain Name System (DNS)

Note: Name server names are in italics, and the corresponding domains are in parentheses.

Arrows denote name server entries



The Domain Name System (DNS)

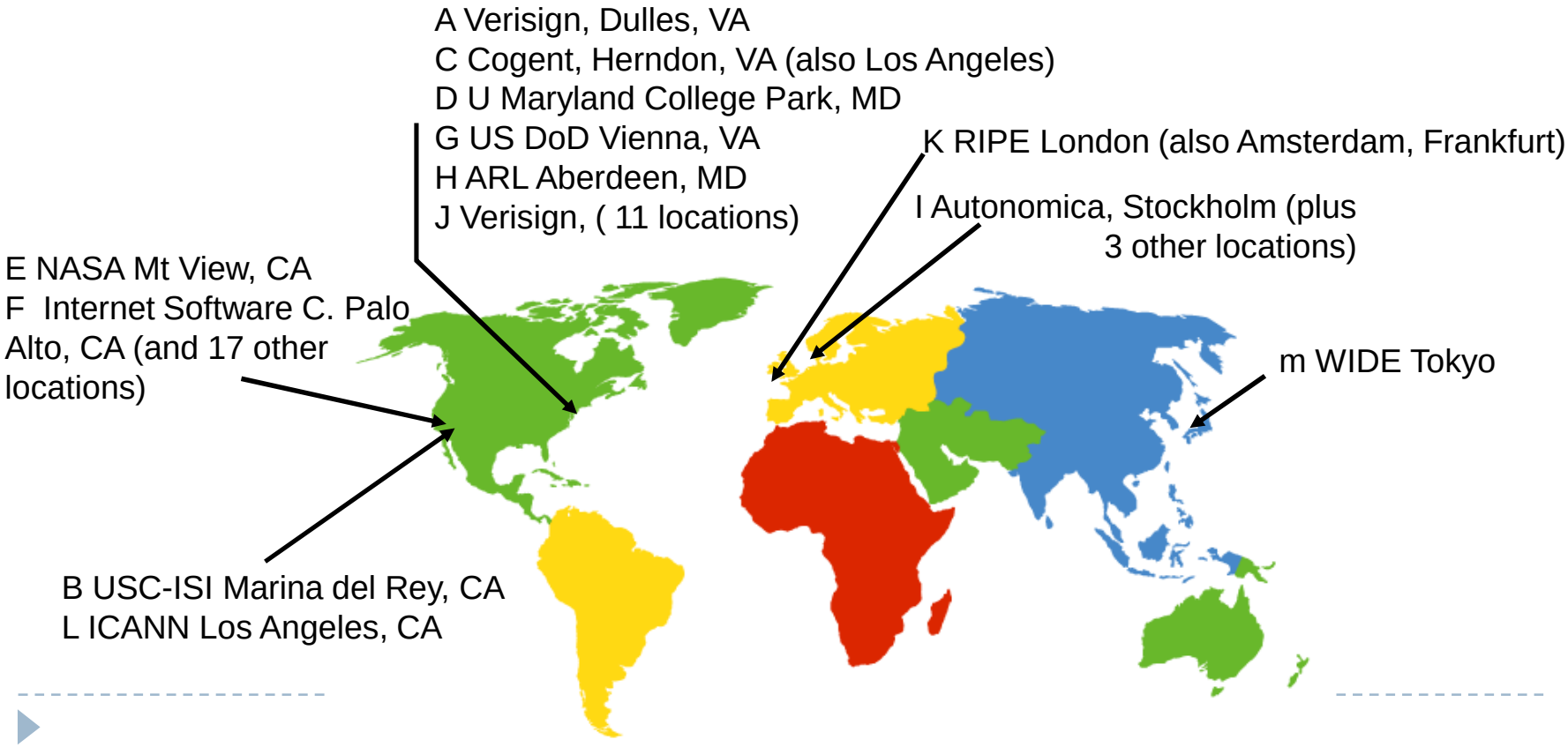
- Properties of DNS
  - Hierarchical name space divided into zones
  - Distributed over a collection of DNS servers
- Hierarchy of DNS servers
  - Root servers
  - Top-level domain (TLD) servers
  - Authoritative DNS servers
- Performing the translations
  - Local DNS servers
  - Resolver software

The Domain Name System (DNS)

- Top-level domain (TLD) servers
  - Generic domains (e.g., com, org, edu)
  - Country domains (e.g., uk, fr, ca, jp)
  - Typically managed professionally
    - Network Solutions maintains servers for "com"
    - Educause maintains servers for "edu"
- Authoritative DNS servers
  - Provide public records for hosts at an organization
  - For the organization's servers (e.g., Web and mail)
  - Can be maintained locally or by a service provider

The Domain Name System (DNS)

- 13 DNS root servers (see http://www.root-servers.org/)
- Labeled A through M



The Domain Name System (DNS)

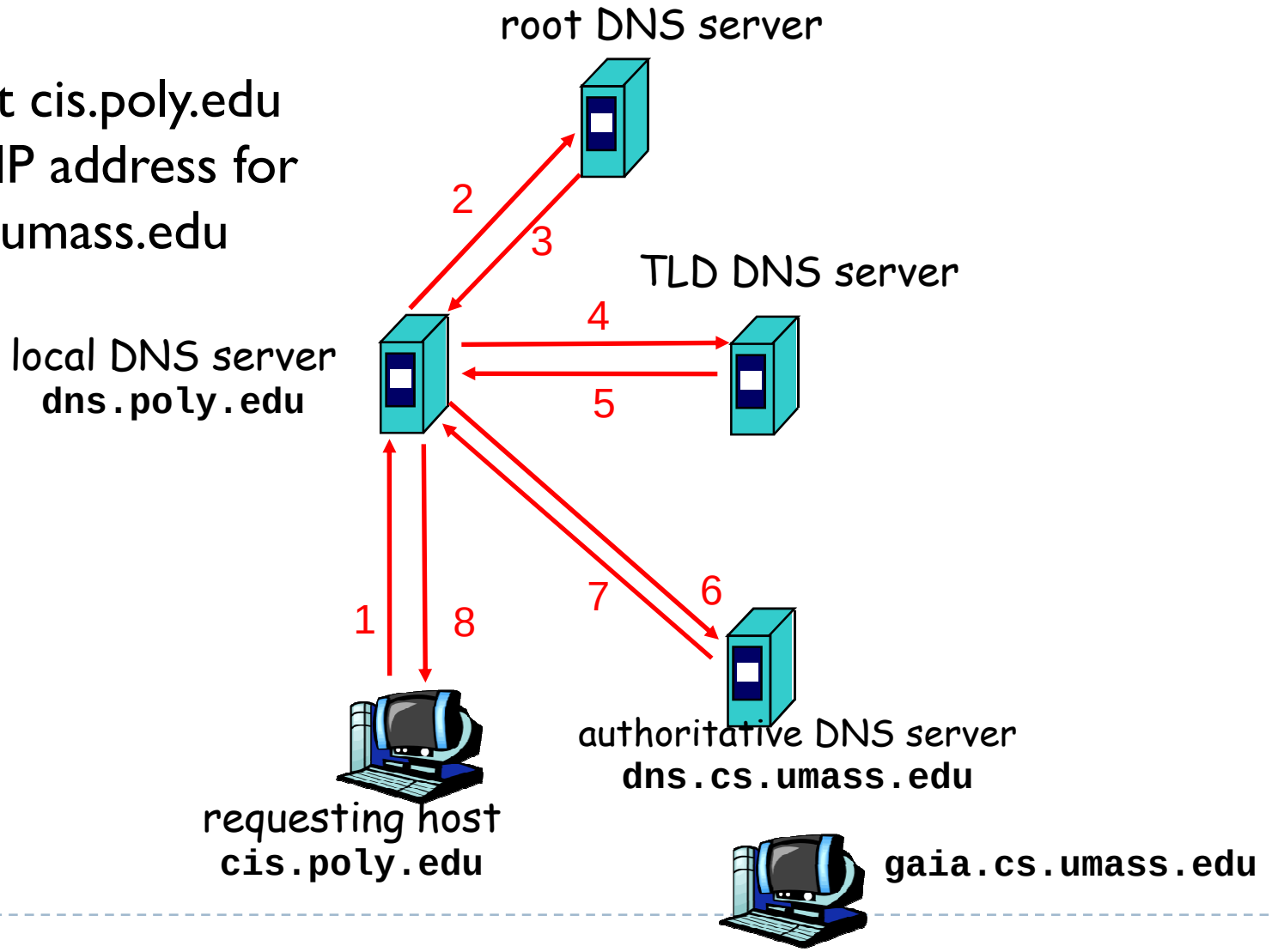
- A distributed naming database
- Name structure reflects administrative structure of the Internet
- Rapidly resolves domain names to IP addresses
  - exploits caching heavily
  - typical query time ~100 milliseconds
- Scales to millions of computers
  - partitioned database
  - caching
- Resilient to failure of a server
  - replication

Using DNS

- Local DNS server ("default name server")
  - Usually near the end hosts who use it
  - Local hosts configured with local server (e.g., /etc/resolv.conf) or learn the server via DHCP
- Client application
  - Extract server name (e.g., from the URL)
  - Do *gethostbyname()* to trigger resolver code
- Server application
  - Extract client IP address from socket
  - Optional *gethostbyaddr()* to translate into name

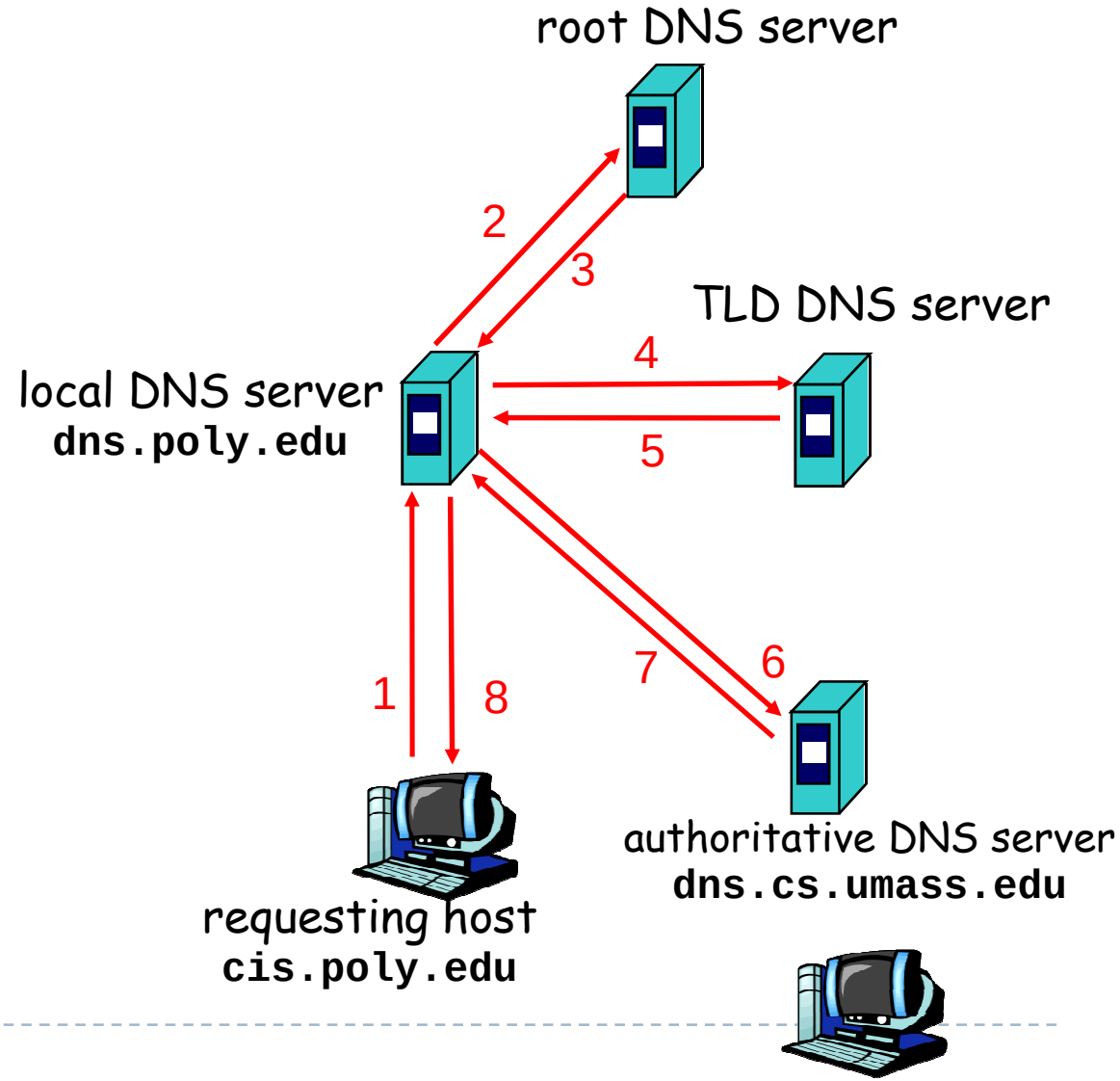
Example

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



Recursive vs. Iterative Queries

- Recursive query
  - Ask server to get answer for you
  - E.g., request 1 and response 8
- Iterative query
  - Ask server who to ask next
  - E.g., all other request-response pairs (2-7)



DNS Caching (more)

- Performing all these queries takes time
  - And all this before the actual communication takes place
    - E.g., 1-second latency before starting Web download
- Caching can substantially reduce overhead
  - The top-level servers very rarely change
  - Popular sites (e.g., www.cnn.com) visited often
  - Local DNS server often has the information cached
- How DNS caching works
  - DNS servers cache responses to queries
  - Responses include a "time to live" (TTL) field
  - Server deletes the cached entry after TTL expires

Negative Caching

- Remember things that don't work
  - Misspellings like www.cnn.comm and www.cnnn.com
  - These can take a long time to fail the first time
  - Good to remember that they don't work
  - … so the failure takes less time the next time around

Reliability

- DNS servers are replicated
  - Name service available if at least one replica is up
  - Queries can be load balanced between replicas
- UDP used for queries
  - Need reliability: must implement this on top of UDP
- Try alternate servers on timeout
  - Exponential backoff when retrying same server
- Same identifier for all queries
  - Don't care which server responds

Inserting Resource Records into DNS

- Example: just created startup "FooBar"
- Register foobar.com at Network Solutions
  - Provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts two RRs into the com TLD server:
    - (foobar.com, dns1.foobar.com, NS) – domain server
    - (dns1.foobar.com, 212.212.212.1, A) – IP address
- Put in authoritative server dns1.foobar.com
  - Type A record for www.foobar.com
  - Type MX record for foobar.com

Playing With Dig on UNIX

- Dig program
  - Allows querying of DNS system
  - Use flags to find name server (NS)
  - Disable recursion so that operates one step at a time

```
unix> dig +norecurse @a.root-servers.net NS www.cs.princeton.edu

;; AUTHORITY SECTION:
edu.                 2D IN NS        L3.NSTLD.COM.
edu.                 2D IN NS        D3.NSTLD.COM.
edu.                 2D IN NS        A3.NSTLD.COM.
edu.                 2D IN NS        E3.NSTLD.COM.
edu.                 2D IN NS        C3.NSTLD.COM.
edu.                 2D IN NS        G3.NSTLD.COM.
edu.                 2D IN NS        M3.NSTLD.COM.
edu.                 2D IN NS        H3.NSTLD.COM.
```

DNS Query in Web Download

- User types or clicks on a URL
  - E.g., http://www.cnn.com/2006/leadstory.html
- Browser extracts the site name
  - E.g., www.cnn.com
- Browser calls gethostbyname() to learn IP address
  - Triggers resolver code to query the local DNS server
- Eventually, the resolver gets a reply
  - Resolver returns the IP address to the browser
- Then, the browser contacts the Web server
  - Creates and connects socket, and sends HTTP request

Multiple DNS Queries

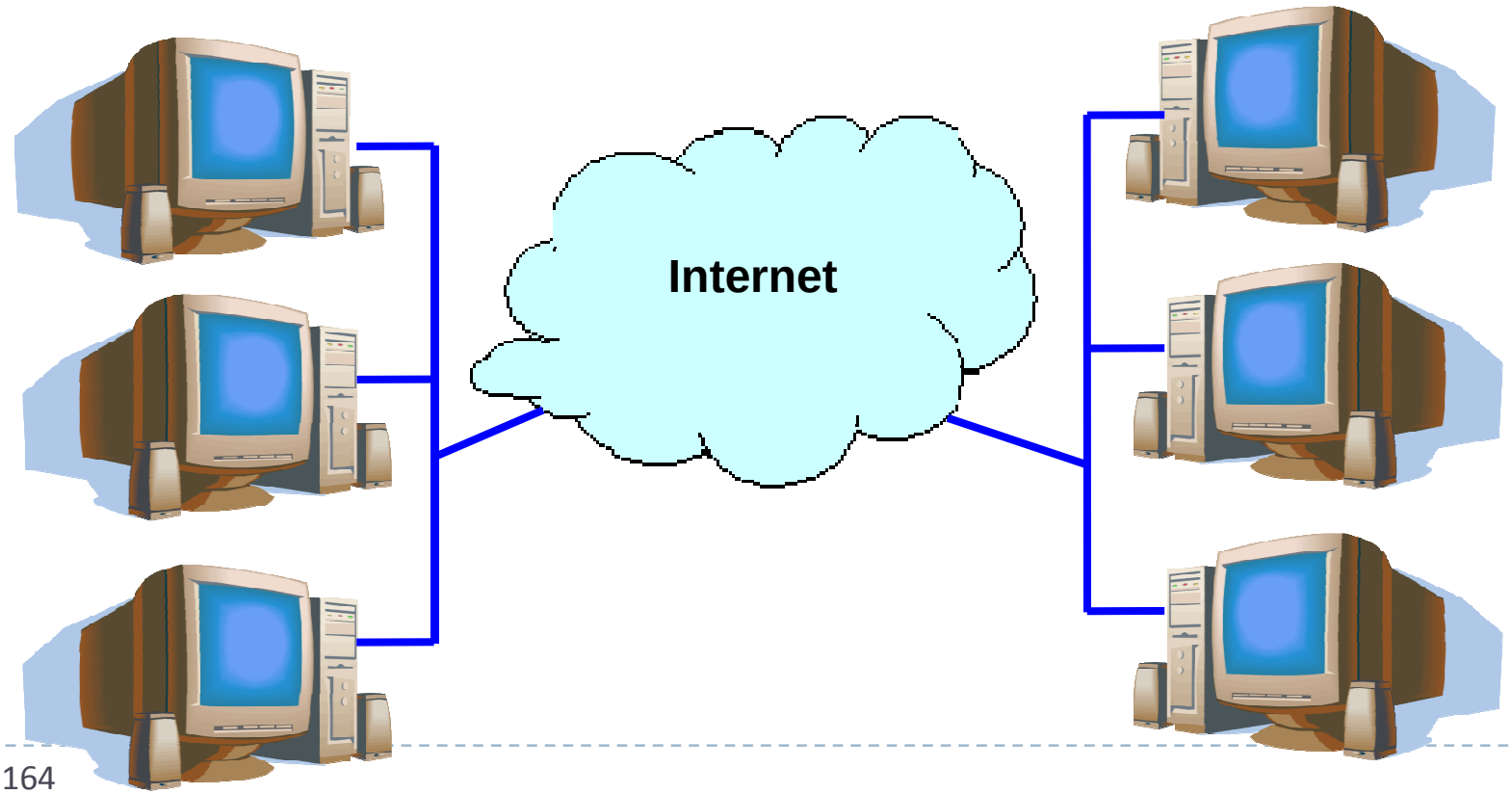
- Often a Web page has embedded objects
  - E.g., HTML file with embedded images
- Each embedded object has its own URL
  - … and potentially lives on a different Web server
  - E.g., http://www.myimages.com/image1.jpg
- Browser downloads embedded objects
  - Usually done automatically, unless configured otherwise
  - Requires learning the address for www.myimages.com

When are DNS Queries Unnecessary?

- Browser is configured to use a proxy
  - E.g., browser sends all HTTP requests through a proxy
  - Then, the proxy takes care of issuing the DNS request
- Requested Web resource is locally cached
  - E.g., cache has http://www.cnn.com/2006/leadstory.html
  - No need to fetch the resource, so no need to query
- Browser recently queried for this host name
  - E.g., user recently visited http://www.cnn.com/
  - So, the browser already called *gethostbyname()* and may be locally caching the resulting IP address

Web Server Replicas

- Popular Web sites can be easily overloaded
  - Web site often runs on multiple server machines



Directing Web Clients to Replicas

- Simple approach: different names
  - www1.cnn.com, www2.cnn.com, www3.cnn.com
  - But, this requires users to select specific replicas
- More elegant approach: different IP addresses
  - Single name (e.g., www.cnn.com), multiple addresses
  - E.g., 64.236.16.20, 64.236.16.52, 64.236.16.84, …
- Authoritative DNS server returns many addresses
  - And the local DNS server selects one address
  - Authoritative server may vary the order of addresses

Clever Load Balancing Schemes

- Selecting the "best" IP address to return
  - Based on server performance
  - Based on geographic proximity
  - Based on network load
  - …
- Example policies
  - Round-robin scheduling to balance server load
  - U.S. queries get one address, Europe another
  - Tracking the current load on each of the replicas

Challenge: What About DNS Caching?

- Problem: DNS caching
  - What if performance properties change?
  - Web clients still learning old "best" Web server until the cached information expires
- Solution: Small Time-to-Live values
  - Setting artificially small TTL values so replicas picked based on fresh information
- Disadvantages: abuse of DNS?
  - Many more DNS request/response messages
  - Longer latency in initiating the Web requests

Questions?