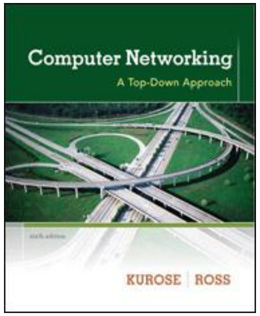
Chapter 0: COMPUTER NETWORKING
Part 1

Communications in Distributed Systems

Fundamentals and Grand Tour of Computer Networking

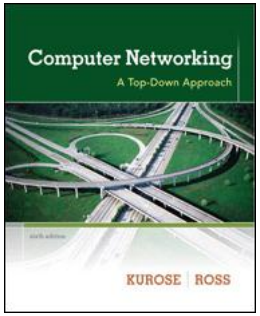


Thanks to the authors of the textbook [**KR**] for providing the base slides. I made several changes/additions.
These slides may incorporate materials kindly provided by Prof. Dakai Zhu.
So I would like to thank him, too.

Turgay Korkmaz
korkmaz@cs.utsa.edu

Chapter 0: Computer Networking

- Layered Protocols
- Grand tour of computer networking, the Internet
- Client-server paradigm,
- Socket Programming

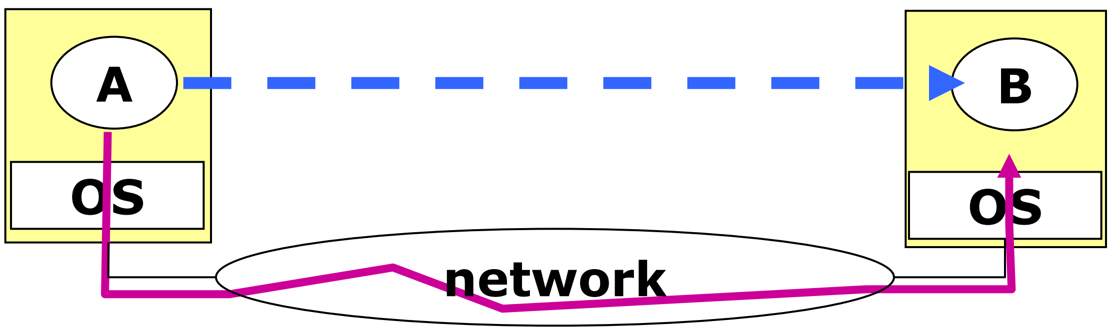


Objectives

- To understand how processes communicate (the heart of distributed systems)
- To understand computer networks and their layers
- To understand client-server paradigm and low-level message passing using **sockets** (part 2)

Fundamentals

How can A and B communicate?

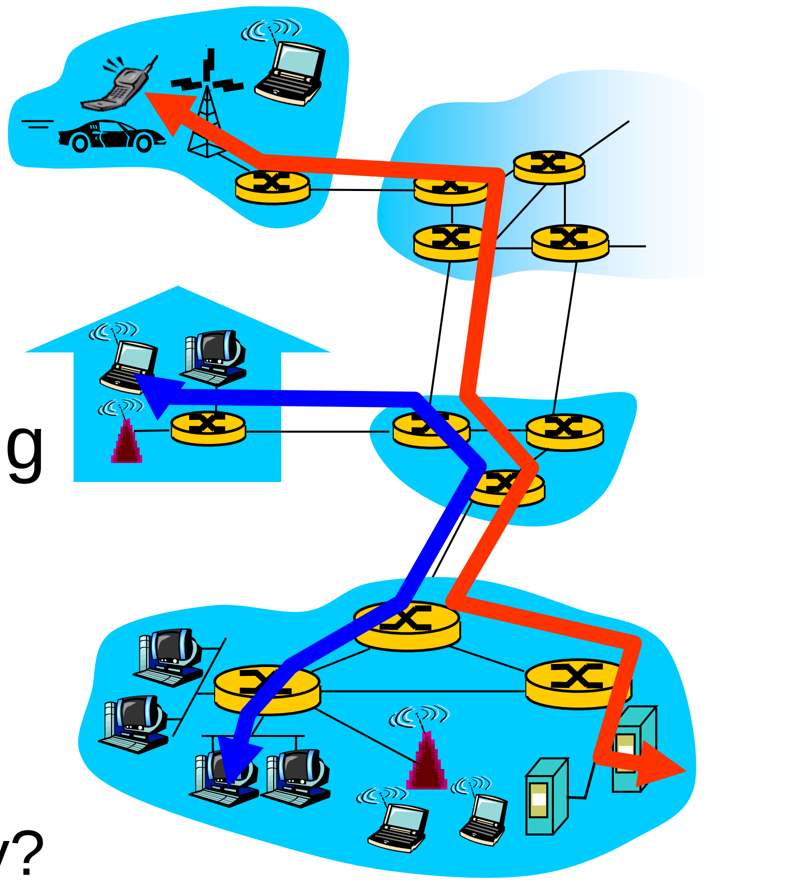


- Many different agreements (protocols) are needed at various levels
- **Application-level agreements** 
  - Bit representation to meaning of each message
- **Other-levels and agreements** 
  - How to actually transmit messages through a network
  - Addressing, performance, scalability, reliability, security

What's Network (the Internet)?

To learn more, take CS 6543

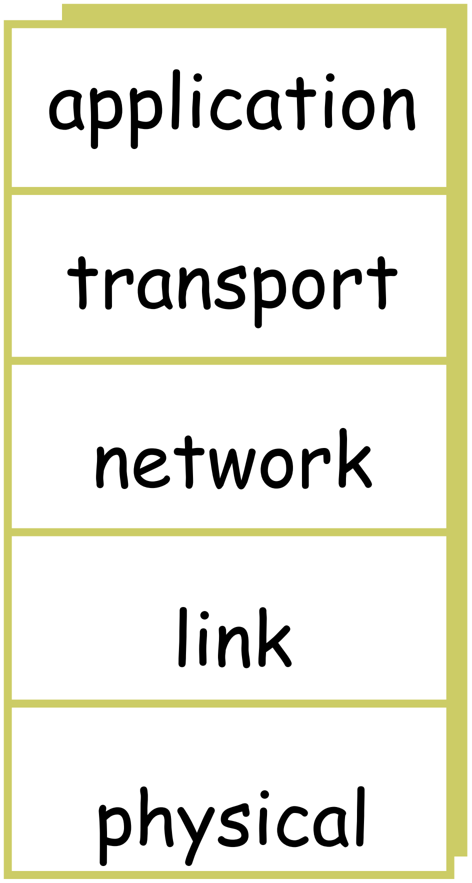
- Network of networks connecting millions of devices:
  - Hosts (end systems)
  - Links (fiber to satellite)
  - Routers and switches
- Collection of protocols providing communication services to distributed applications
- Networks are complex!
  - How can we deal with complexity?
  - Modular design, layering!



From Computer Networking by Kurose and Ross.

Internet protocol stack

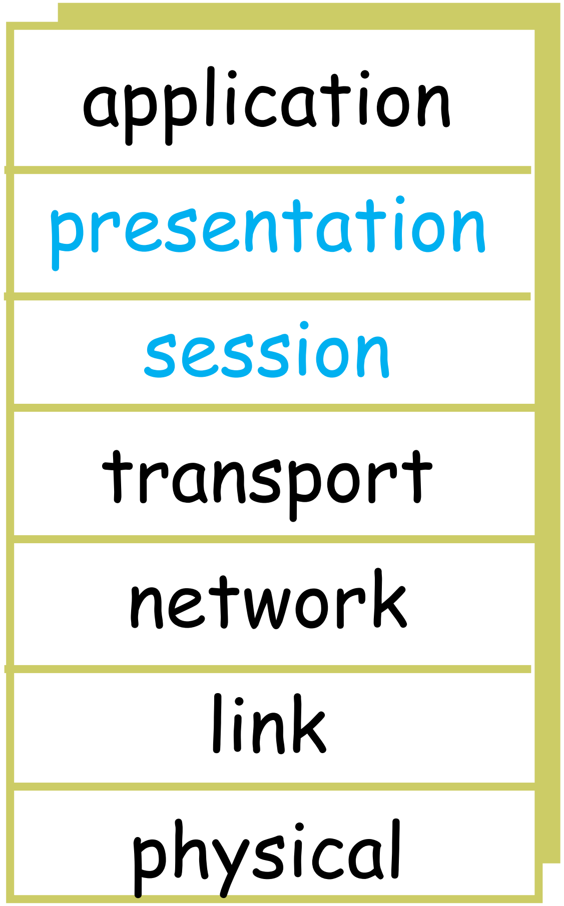
- application: Protocols that are designed to meet the communication requirements of specific applications, often defining the interface to a service. (FTP, SMTP, HTTP)
- transport: process-to-process data transfer (TCP, UDP)
- network: routing of datagrams from source to destination (IP, OSPF, BGP)
- link: data transfer between neighboring network elements (PPP, Ethernet)
- physical: transmission of bits on a link (electrical signals on cable, light signals on fibre or other electromagnetic signals on radio)



From Computer Networking by Kurose and Ross.

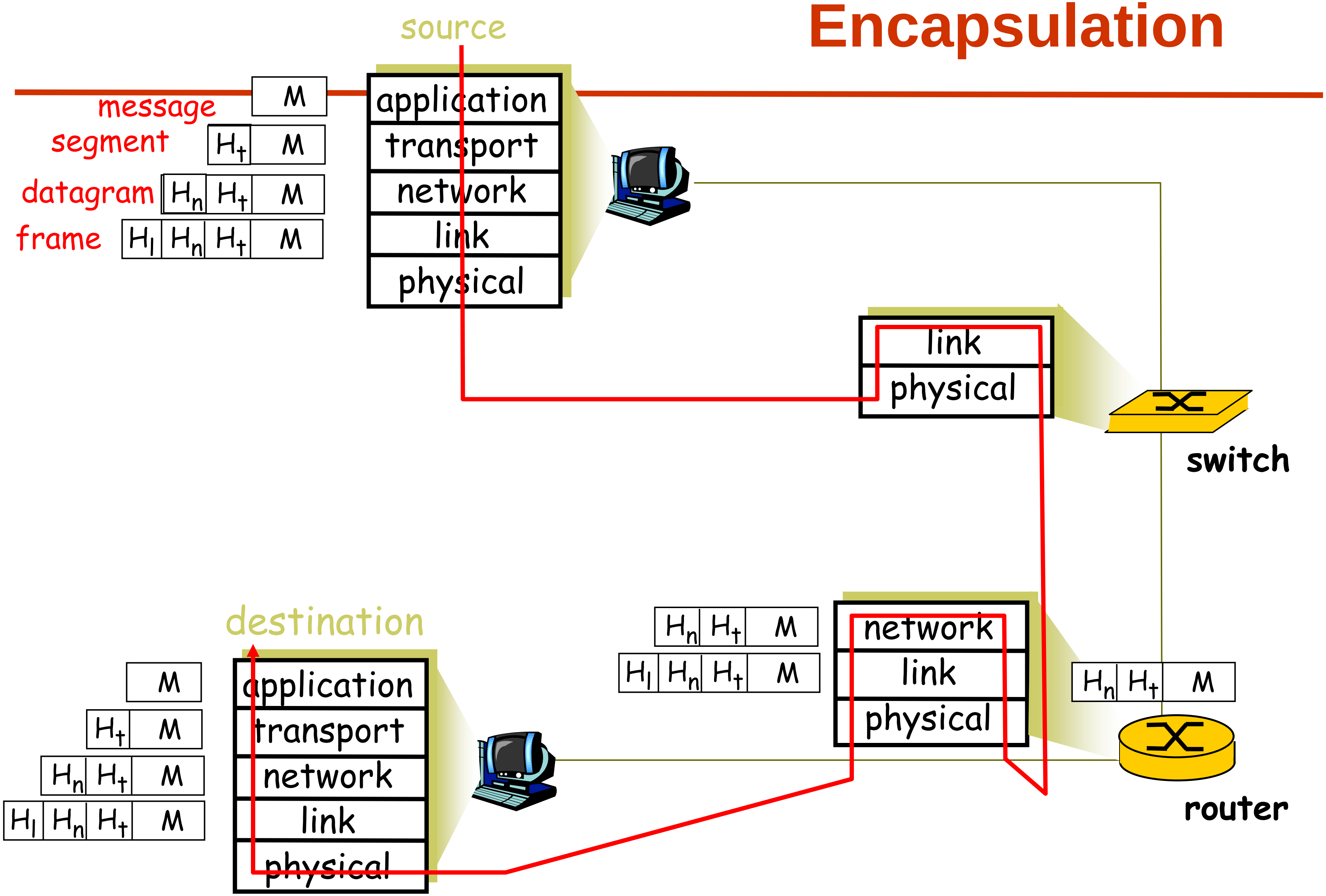
ISO/OSI reference model

- presentation: allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- *session:* synchronization, check pointing, recovery of data exchange
- Internet stack "missing" these layers!
  - these services, *if needed,* must be implemented in application
  - needed?



From Computer Networking by Kurose and Ross.

Encapsulation

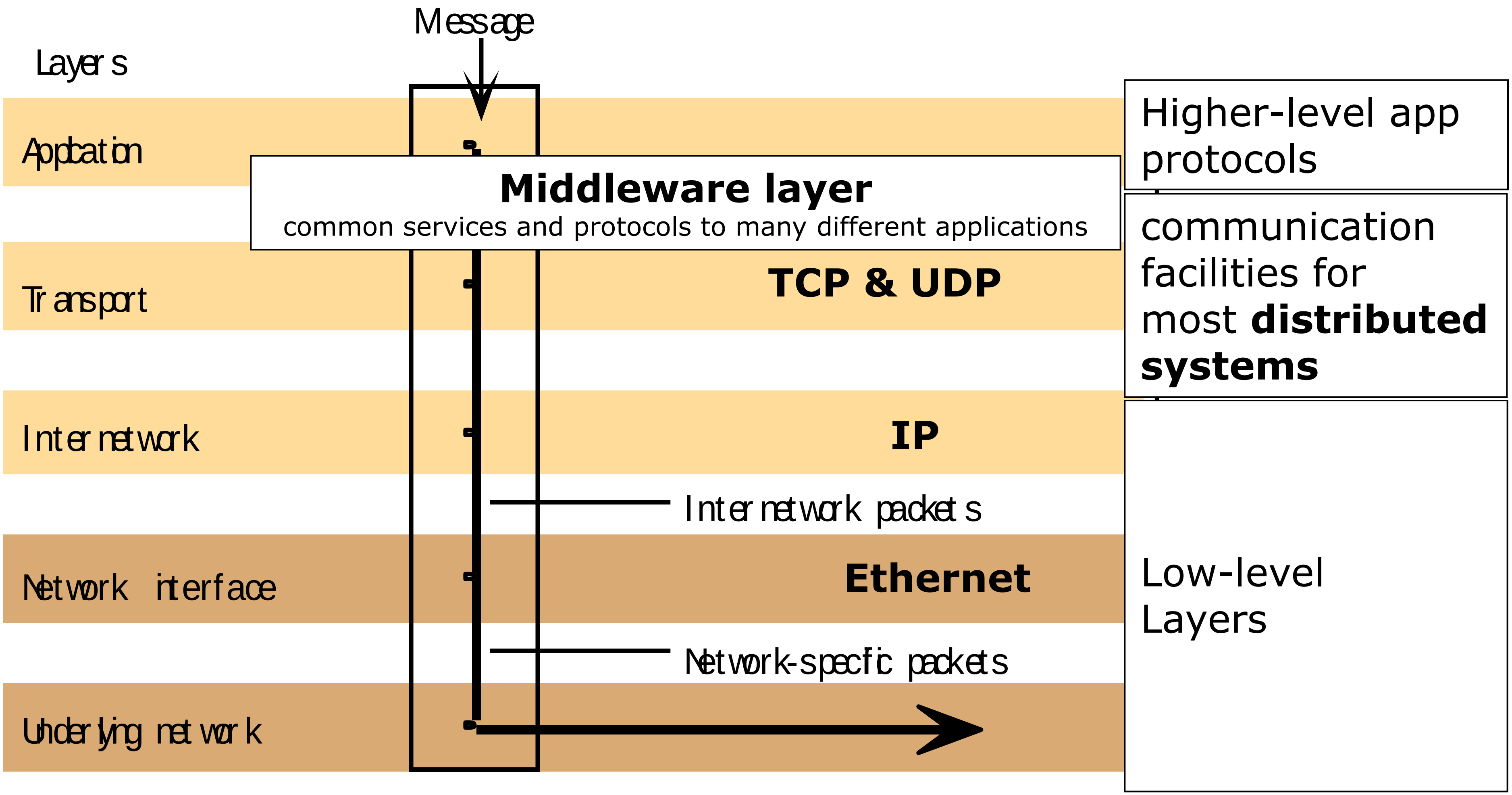


From Computer Networking by Kurose and Ross.

Why layering?

- Explicit structure allows identification, relationship of complex system's pieces
- Each layer
  - gets a service from the one below,
  - performs a specific task, and
  - provides a service to the one above
- Modularization eases maintenance and updating of system
  - We can change the implementation of a layer without affecting the rest of the system as long as the interfaces between the layer are kept the same!
- In some cases, layering considered harmful! Why?

Distributed Systems and Layer Structure



The transport layer and middleware layer provide the actual communication facilities for most distributed systems.

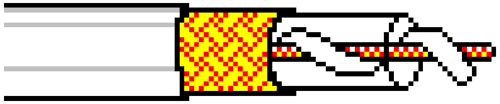
But before discussing their details, let us just review all the layers in a bottom-up fashion (more details are in CS 6543)

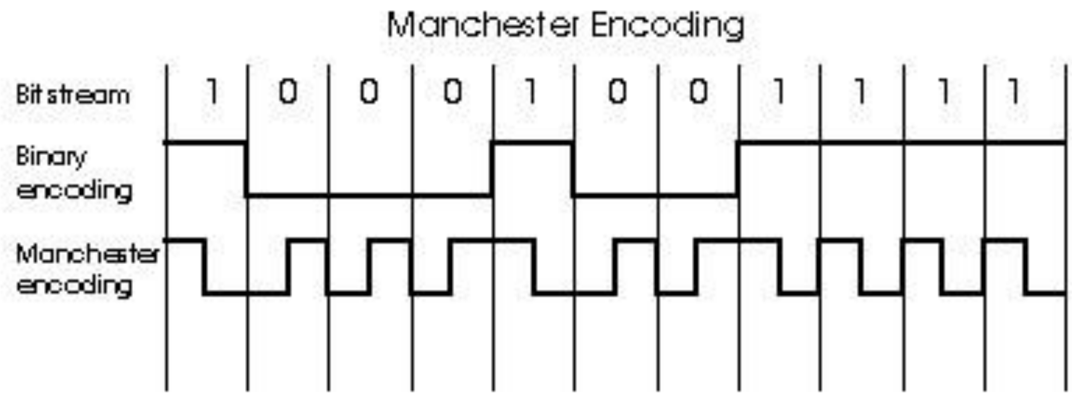
GRAND TOUR OF COMPUTER NETWORKING

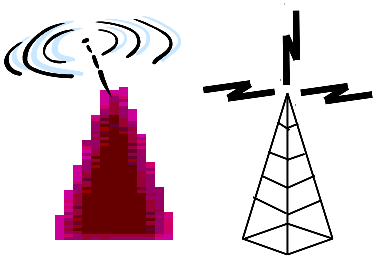
Physical Layer

Transmission of bits on a link

- electrical signals on cable,
- light signals on fibre
- electromagnetic signals on radio



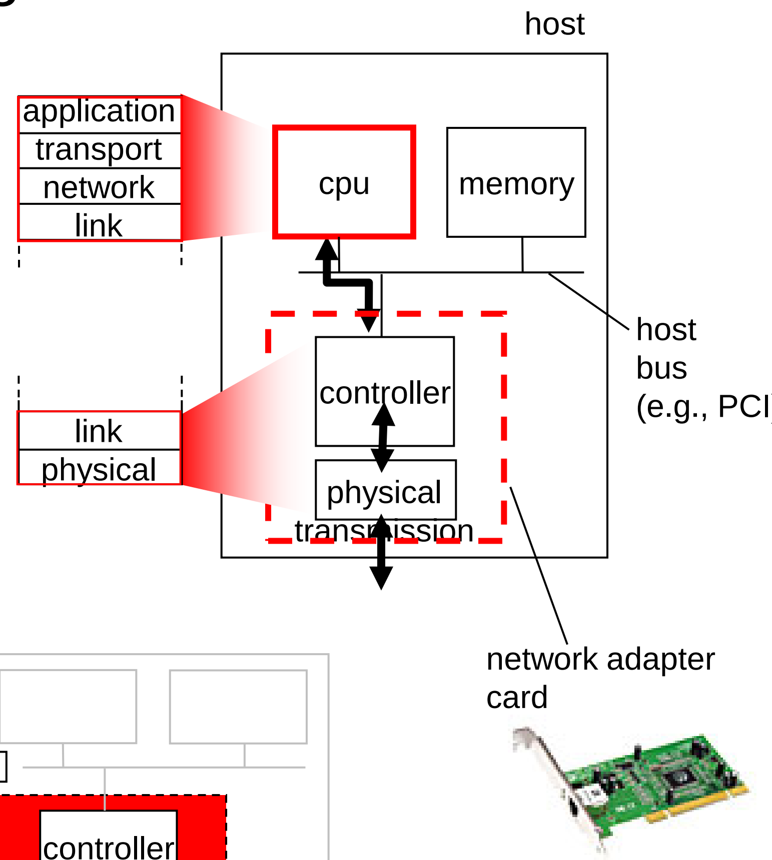


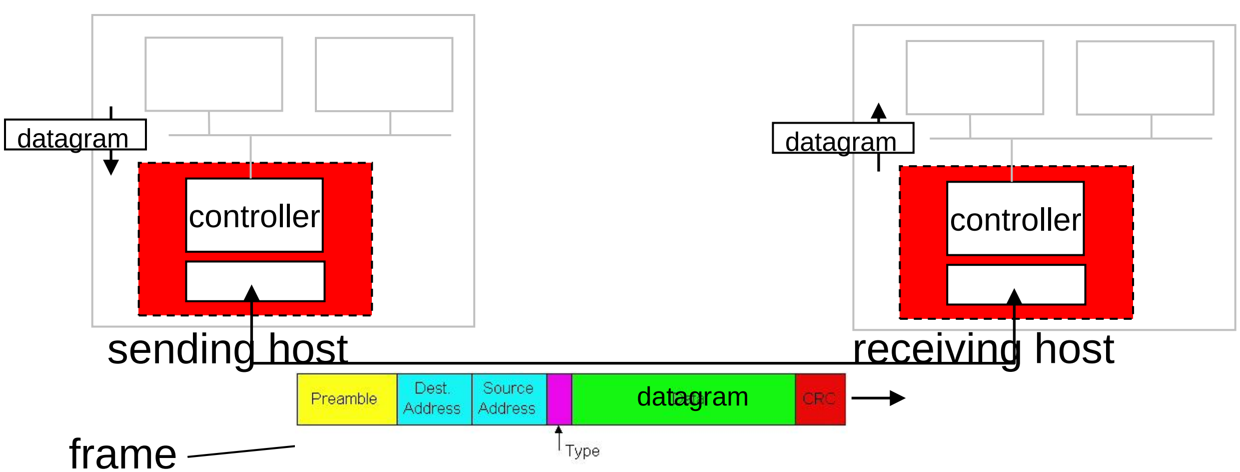


From Computer Networking by Kurose and Ross.

Link Layer

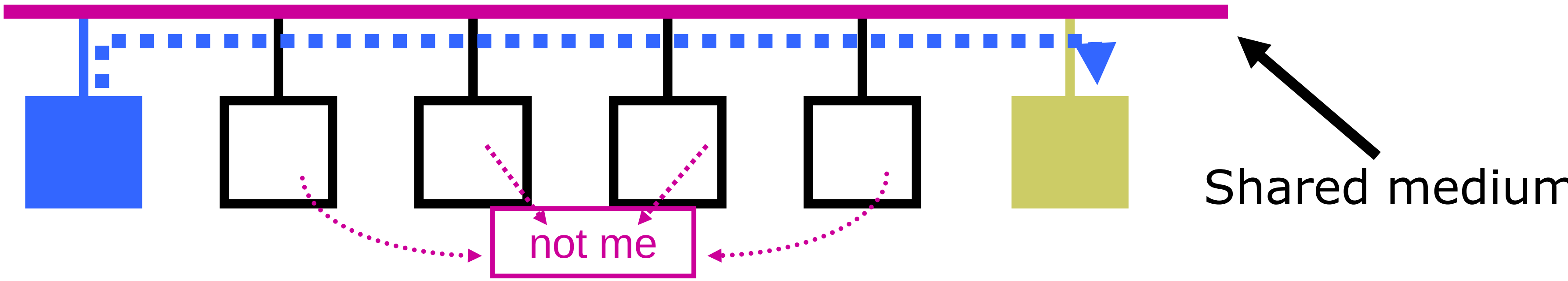
- Data transfer between neighboring network elements
  - Link layer services
    - Framing, error detection and correction…
  - Multiple access protocols
  - Link-layer Addressing
  - Ethernet
  - Link-layer switches
  - PPP





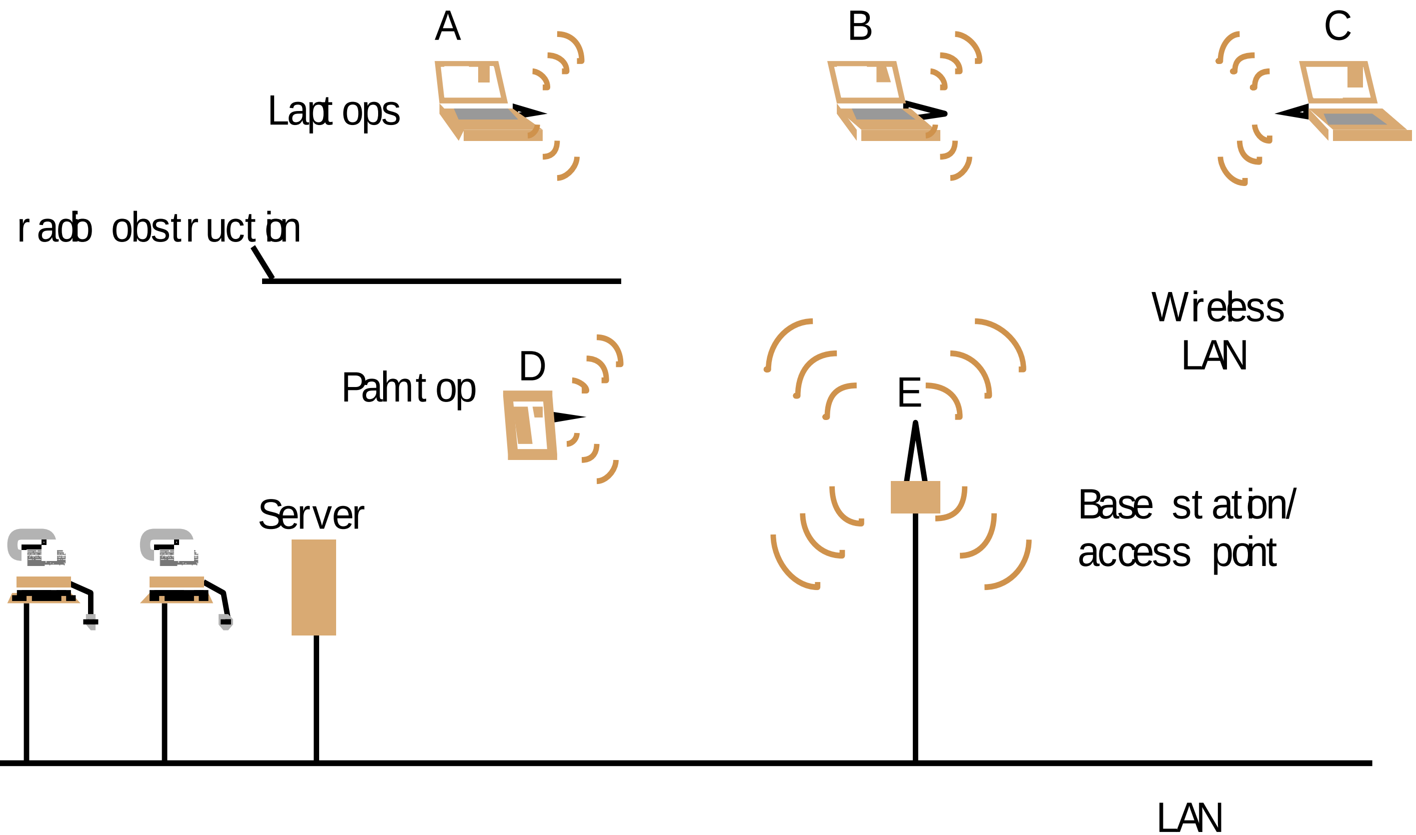
From Computer Networking by Kurose and Ross.

Link layer: Ethernet



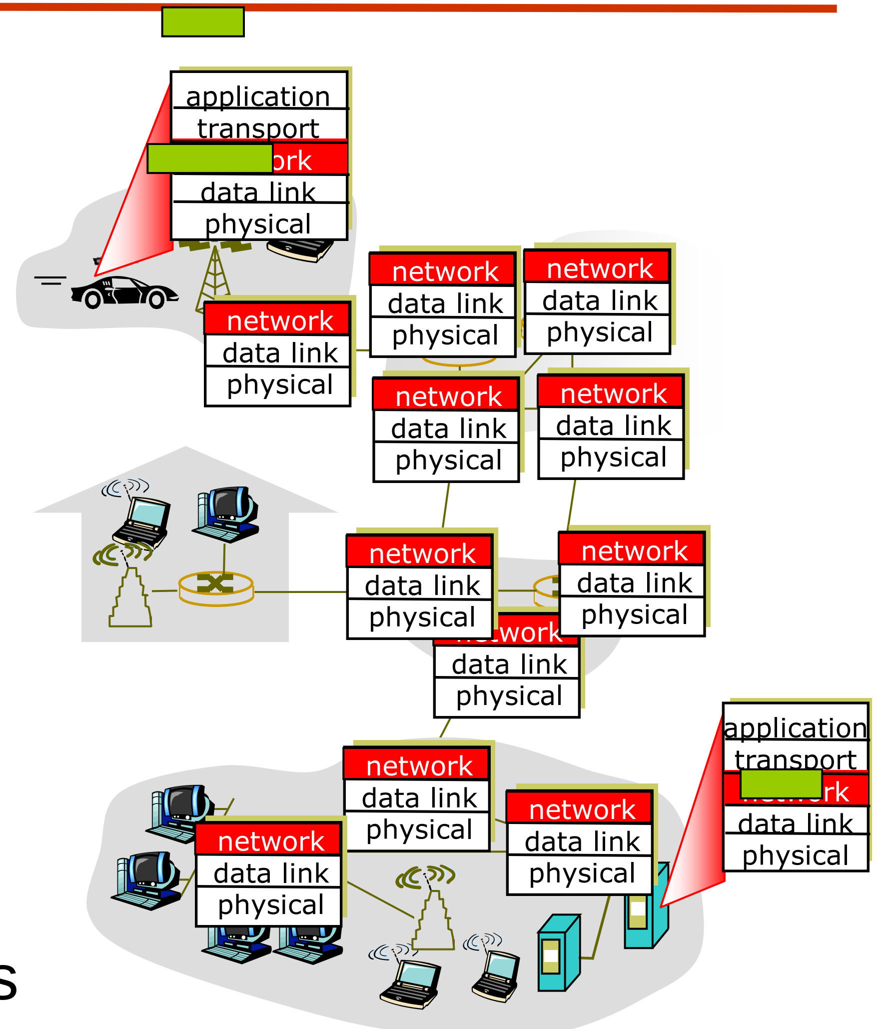
- **Shared medium**: Carrier Sensing Multi-Access.
  - CSMA/CD: collision detection
- Every Ethernet interface has a ***unique*** 48 bit address (a.k.a. *hardware address*).
  - Example: **`C0:B3:44:17:21:17`**
- Addresses are assigned to vendors by a central authority (IEEE to manufacturers)

Wireless LAN



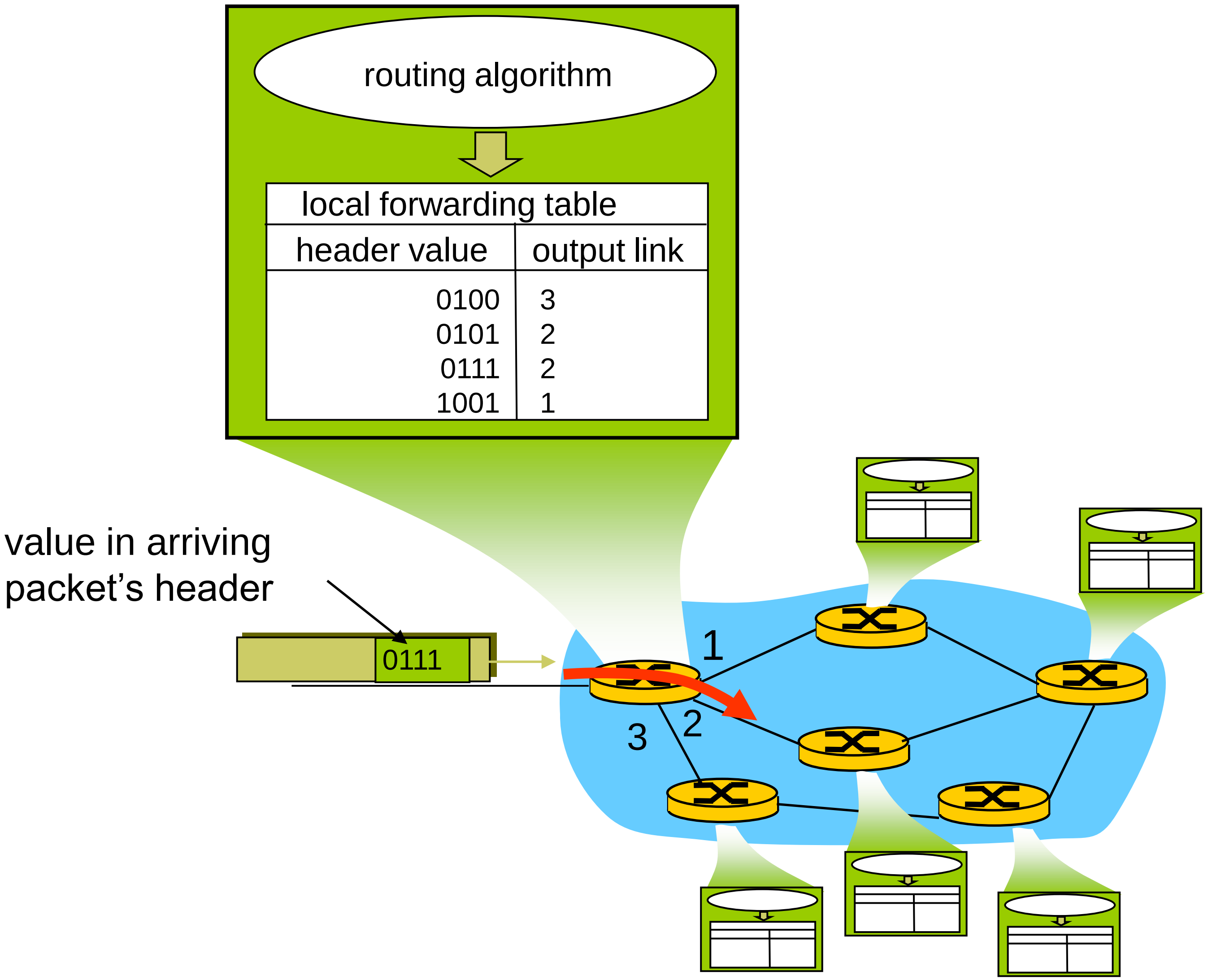
Network layer

- *[On sending side] :* Takes segments from transport layer and encapsulates them into datagrams
- Transports datagrams from sending to receiving host through the network
- *[On receiving side]:* Extracts segments from datagrams and delivers them to transport layer
- Routers examine header fields in all IP datagrams and forwards it to next node
- How to know the next node?



From Computer Networking by Kurose and Ross.

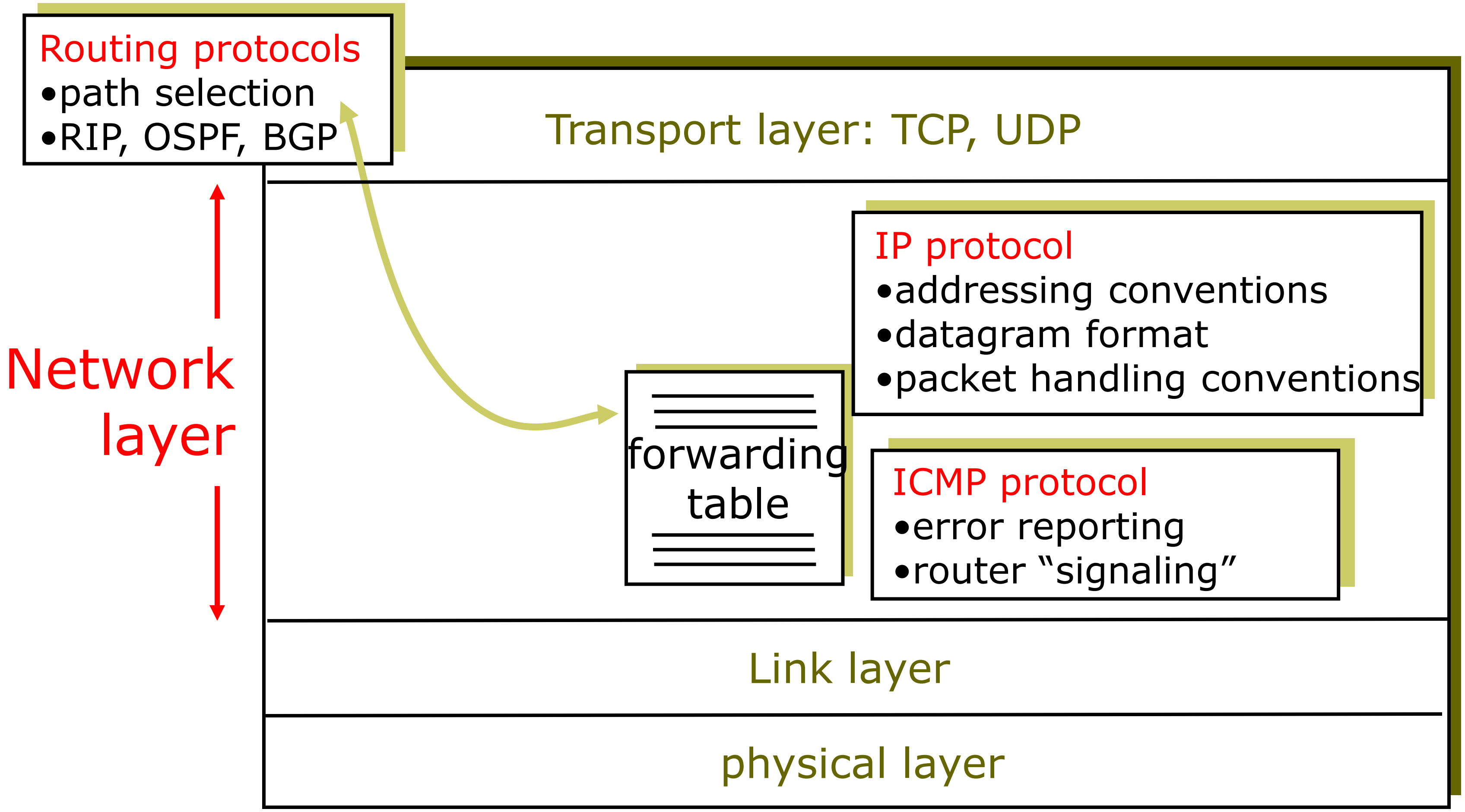
Forwarding Problem: Where to Send Next?



From Computer Networking by Kurose and Ross.

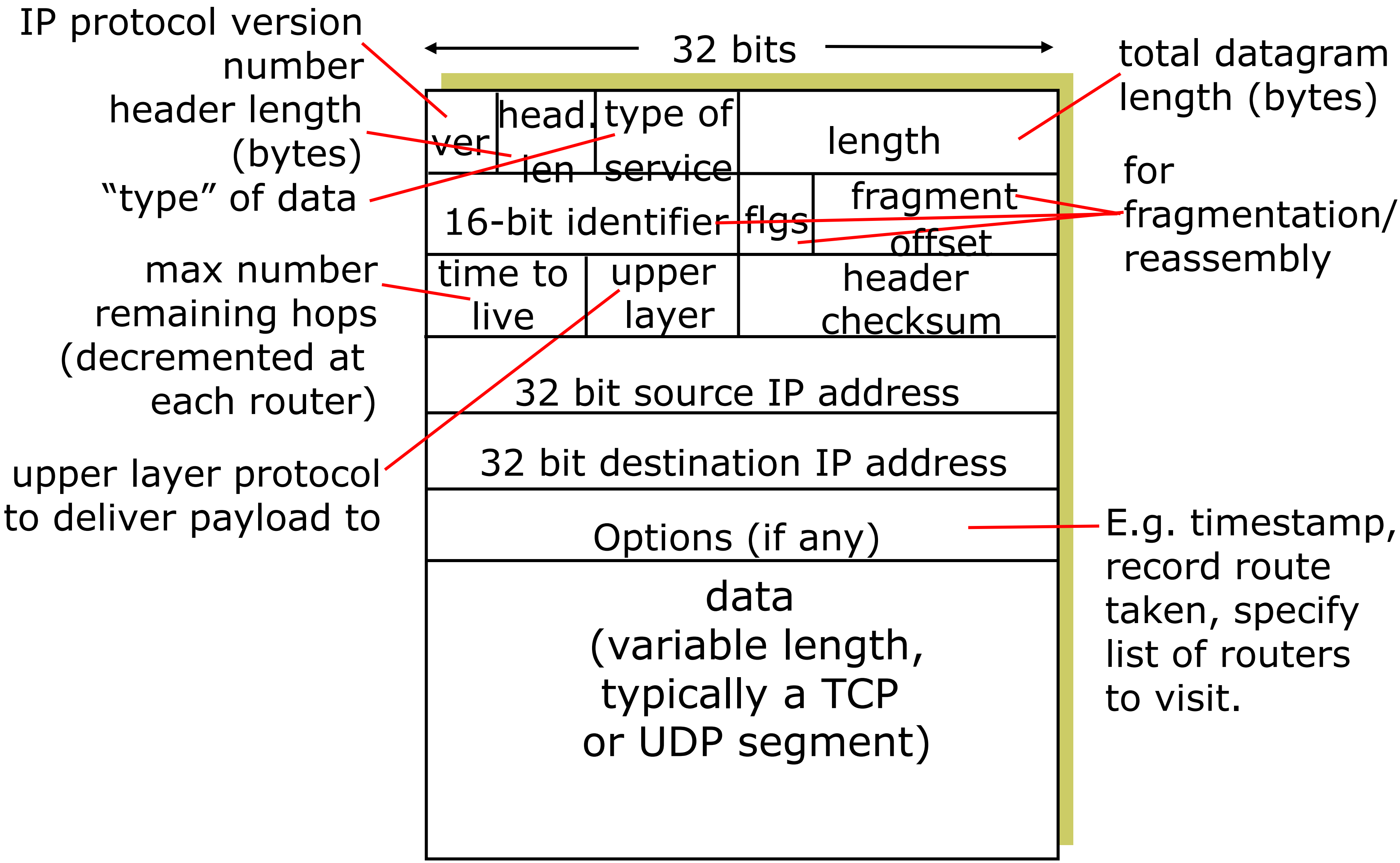
Network layer

Host, router network layer functions:



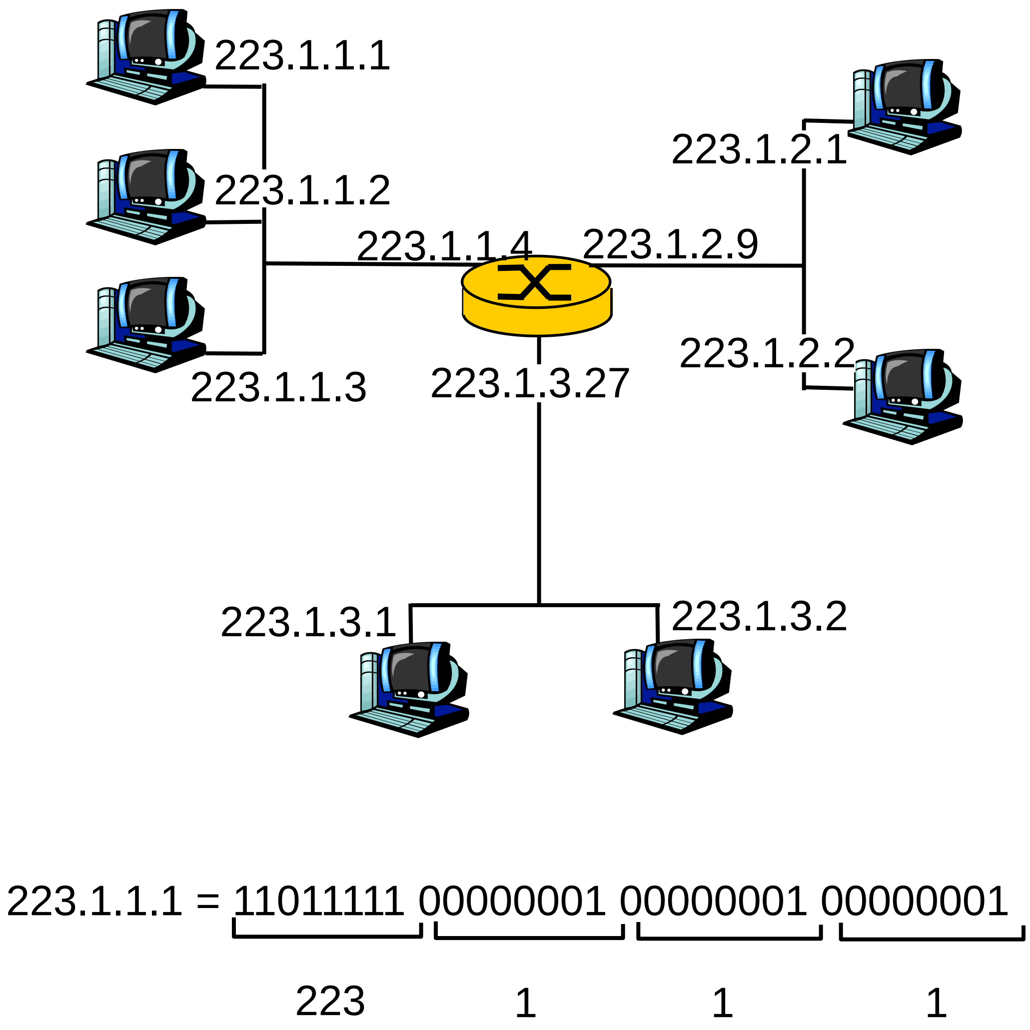
From Computer Networking by Kurose and Ross.

IP datagram format



IP Addressing: introduction

- IP address: 32-bit unique identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one interface
  - IP addresses associated with each interface



From Computer Networking by Kurose and Ross.

IP addressing
CIDR vs. Class-based addressing

CIDR: Classless InterDomain Routing

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

subnet part | host part

11001000 00010111 00010000 00000000

200.23.16.0/23

From Computer Networking by Kurose and Ross.

IP addresses: how to get one?

Q: How to get the (sub)network portion of the address?

A: ICANN: Internet Corporation for Assigned Names and Numbers

- allocates addresses,
- manages DNS
- assigns domain names, resolves disputes

ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010	00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100	00000000	200.23.20.0/23
...	.....	....	....
Organization 7	11001000 00010111 00011110	00000000	200.23.30.0/23

Q: Given the (sub)network portion, how to get *host* portion?

A: Local network owner

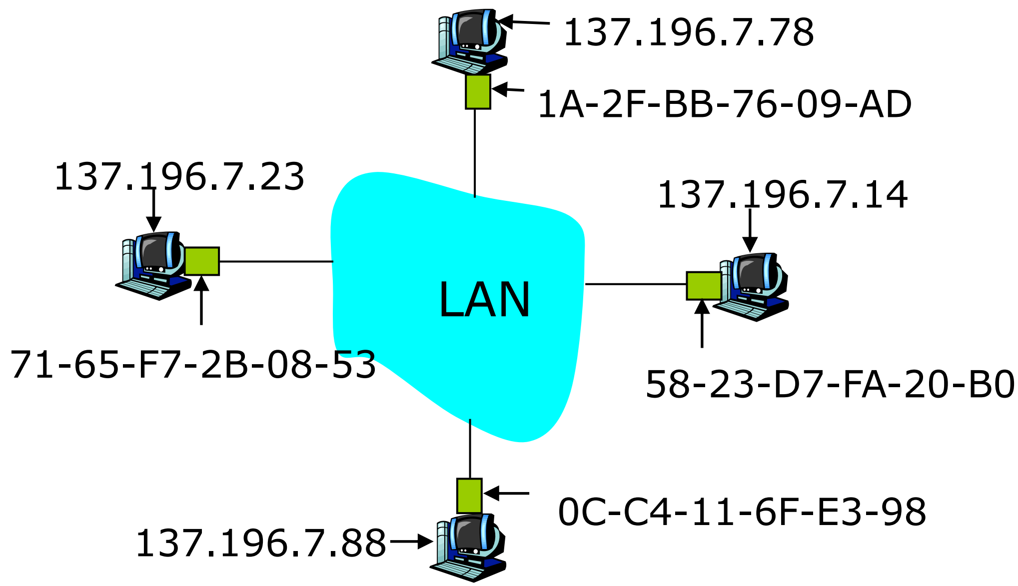
- hard-coded by system admin in a file
  - Windows: control-panel->network->configuration->tcp/ip->properties
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server "plug-and-play"

From Computer Networking by Kurose and Ross.

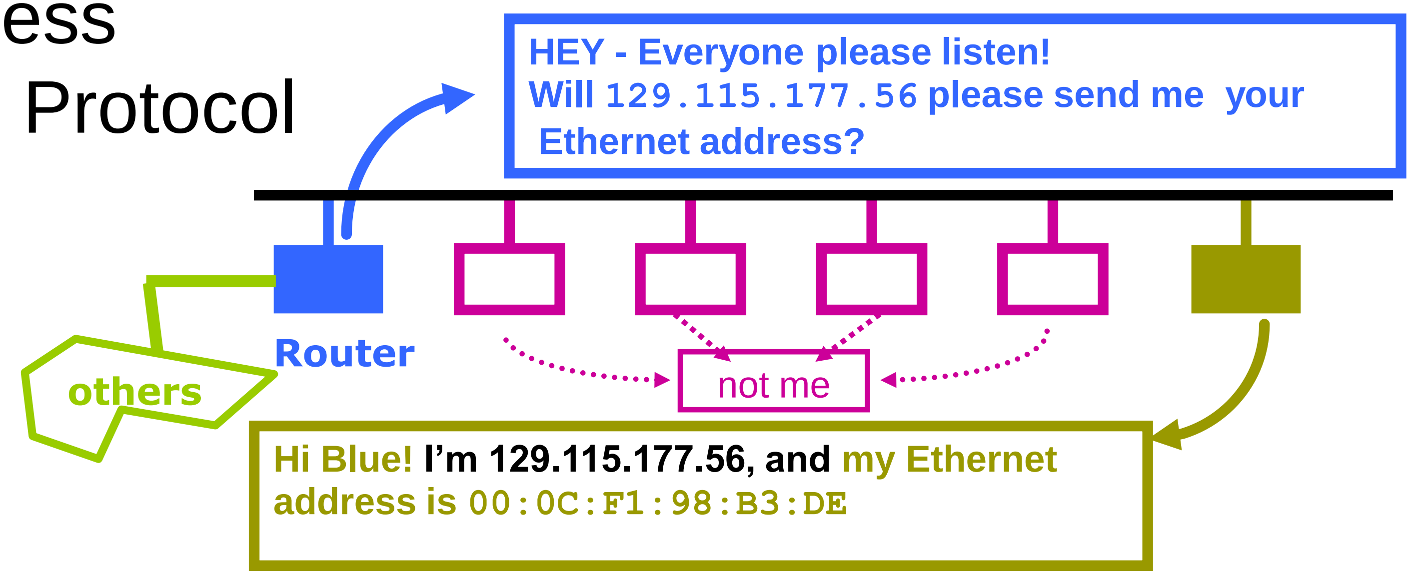
Interaction with IP and MAC addresses

32-bit IP address vs. 48-bit MAC address

- Why do we have both IP and MAC addresses?
- How to determine MAC address for a given IP address?
- ARP: Address Resolution Protocol



(A distributed system using broadcast)

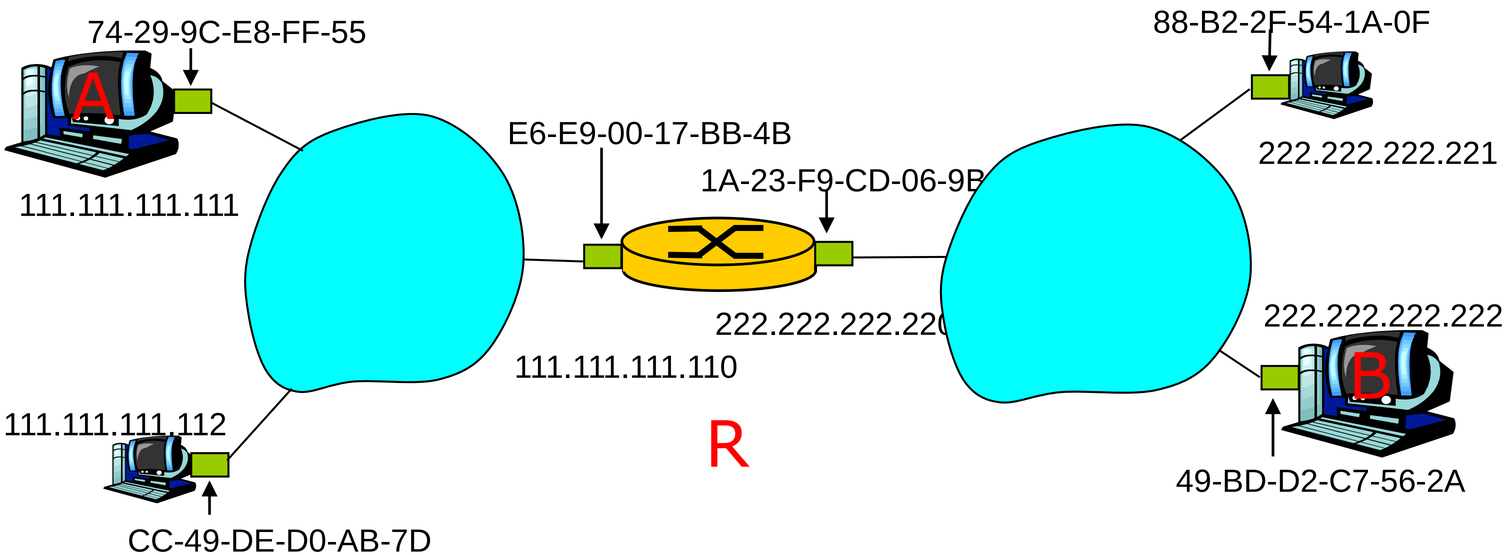


From Computer Networking by Kurose and Ross.

Addressing: routing to another LAN

walkthrough: send datagram from A to B via R

assume A knows B's IP address

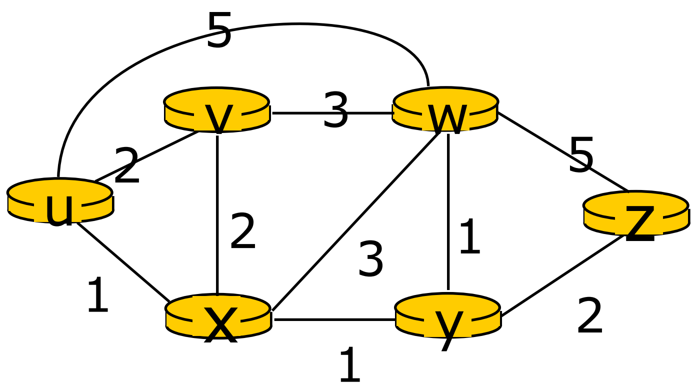


- two ARP tables in router R, one for each IP network (LAN)

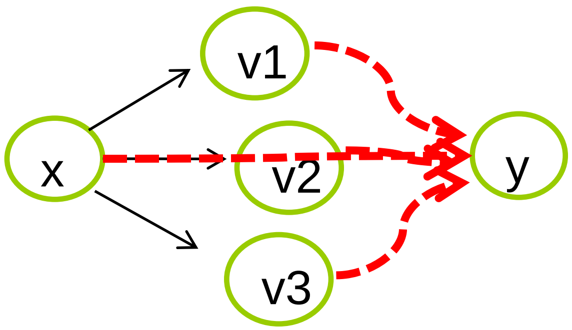
From Computer Networking by Kurose and Ross.

Routing Problem: Find the best path

- Link State algorithm (OSPF)
  - Dissemination link state to have the topology map at each node
  - Use Dijkstra's algorithm to compute the shortest route



- Distance Vector Algorithm (RIP)
  - $d_x(y) = \min \{c(x,v) + d_v(y)\}$



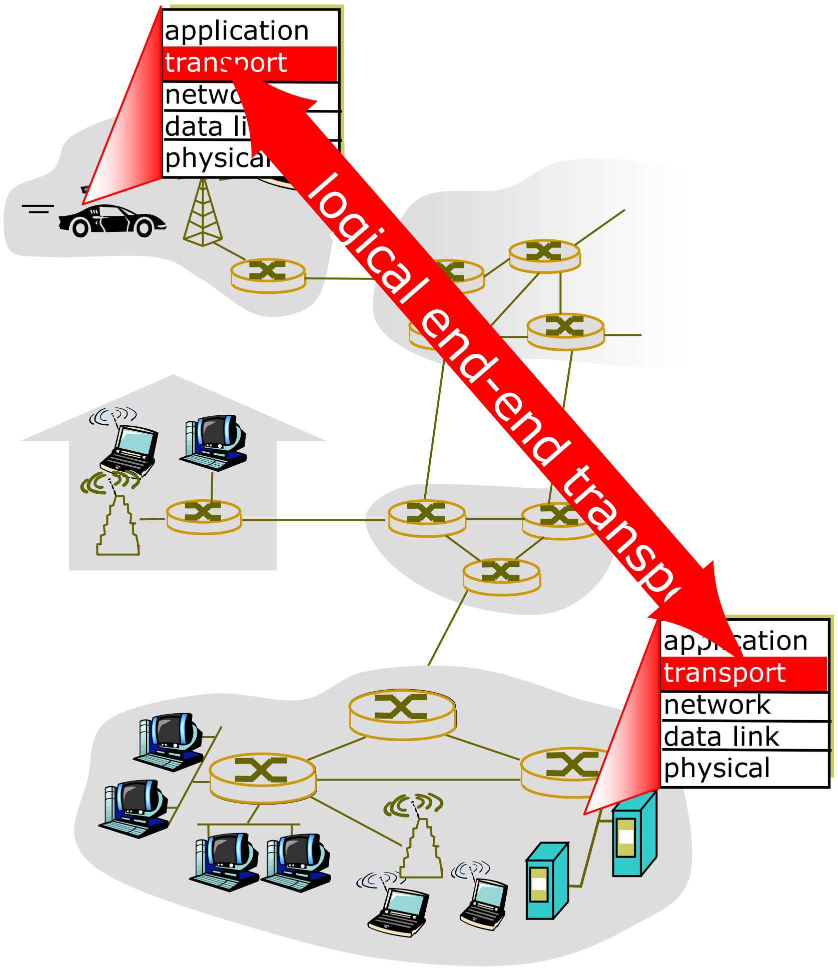
- Hierarchical routing
  - scale: with 200 million destinations
  - each network admin may want to control routing in its own network
- Inter-domain routing vs Intra-domain

A lot of distributed system problems

From Computer Networking by Kurose and Ross.

Transport Layer

- provide *logical communication* between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into segments, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP



From Computer Networking by Kurose and Ross.

Internet transport protocols services

TCP service:

- *connection-oriented:* setup required between client and server processes
- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum throughput guarantees, security

UDP service:

- unreliable data transfer between sending and receiving process
- does not provide: connection setup, reliability, flow control, congestion control, timing, throughput guarantee, or security

Q: why bother? Why is there a UDP?

From Computer Networking by Kurose and Ross.

UDP: User Datagram Protocol [RFC 768]

- “no frills,” “bare bones” Internet transport protocol
- “best effort” service, UDP segments may be:
  - lost
  - delivered out of order to app
- *connectionless:*
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others

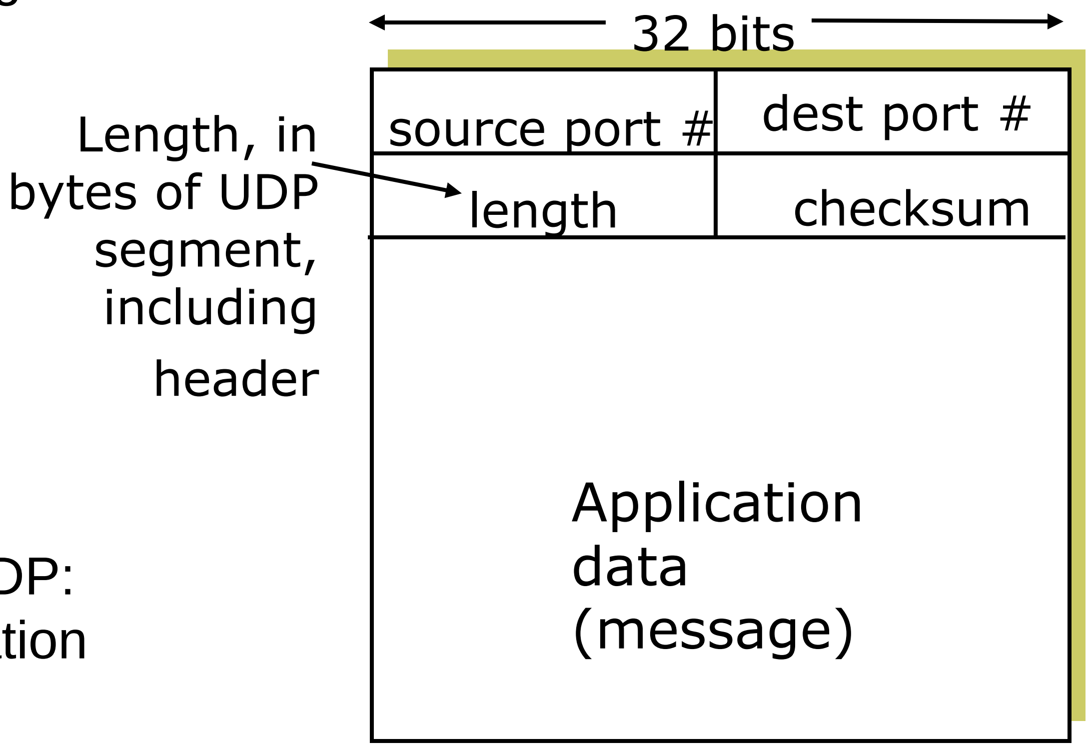
Why is there a UDP?

- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small segment header
- no congestion control: UDP can blast away as fast as desired

From Computer Networking by Kurose and Ross.

UDP: more

- often used for streaming multimedia apps
  - loss tolerant
  - rate sensitive
- other UDP uses
  - DNS
  - SNMP
- reliable transfer over UDP: add reliability at application layer
  - application-specific error recovery!

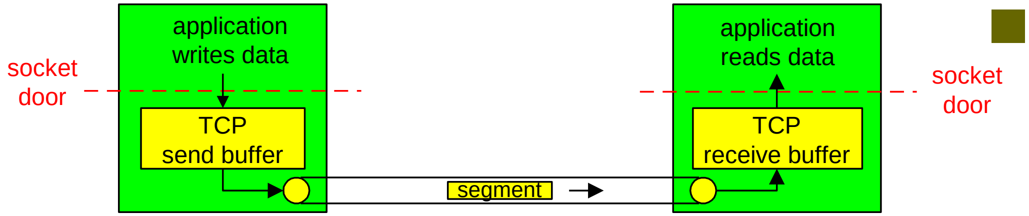


UDP segment format

From Computer Networking by Kurose and Ross.

TCP: Overview RFCs: 793, 1122, 1323, 2018, 2581

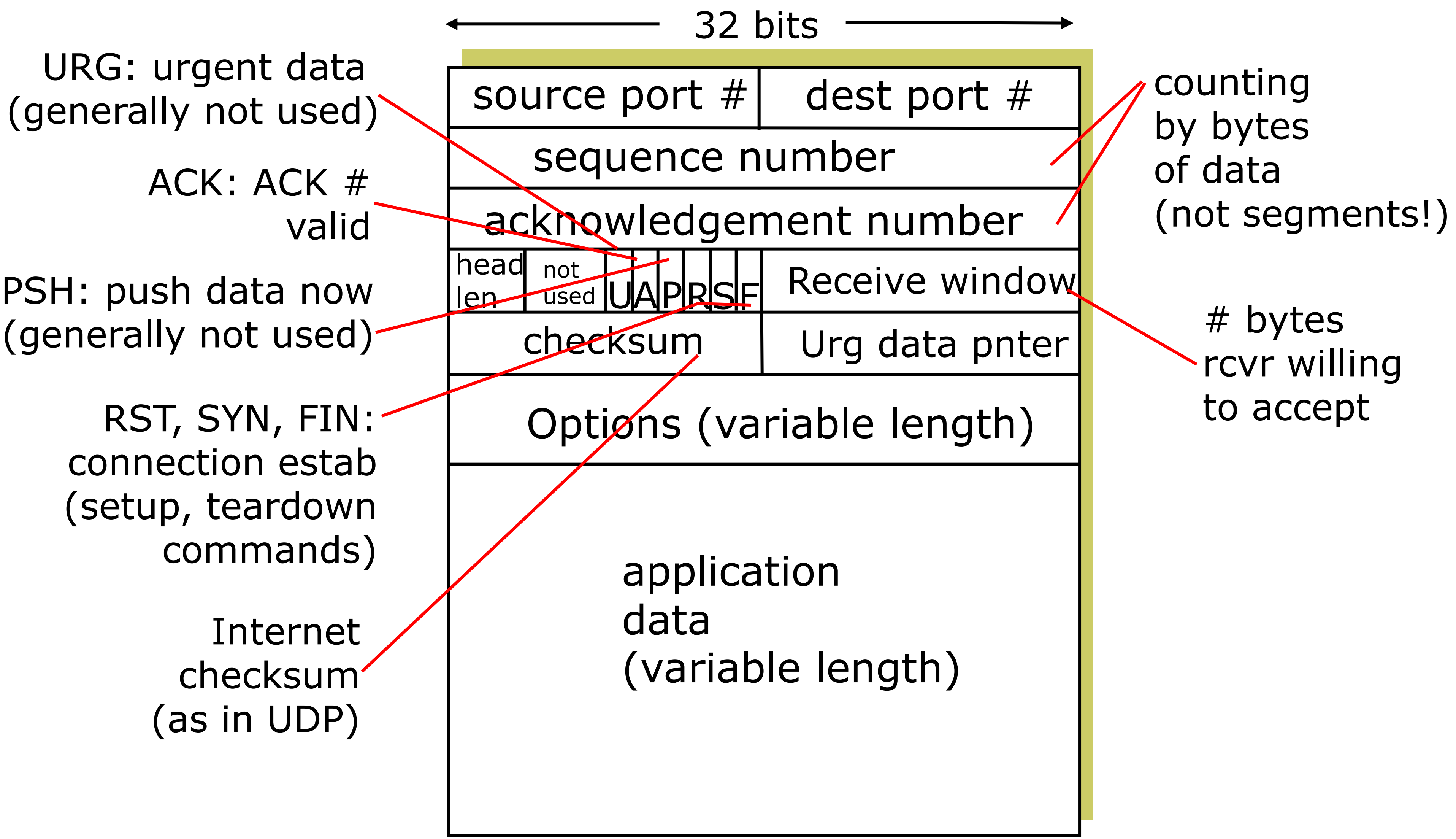
- point-to-point:
  - one sender, one receiver
- reliable, in-order *byte steam:*
  - no "message boundaries"
- pipelined:
  - TCP congestion and flow control set window size
- *send & receive buffers*



- full duplex data:
  - bi-directional data flow in same connection
  - MSS: maximum segment size
- connection-oriented:
  - handshaking (exchange of control msgs) init's sender, receiver state before data exchange
- flow controlled:
  - sender will not overwhelm receiver

From Computer Networking by Kurose and Ross.

TCP segment structure

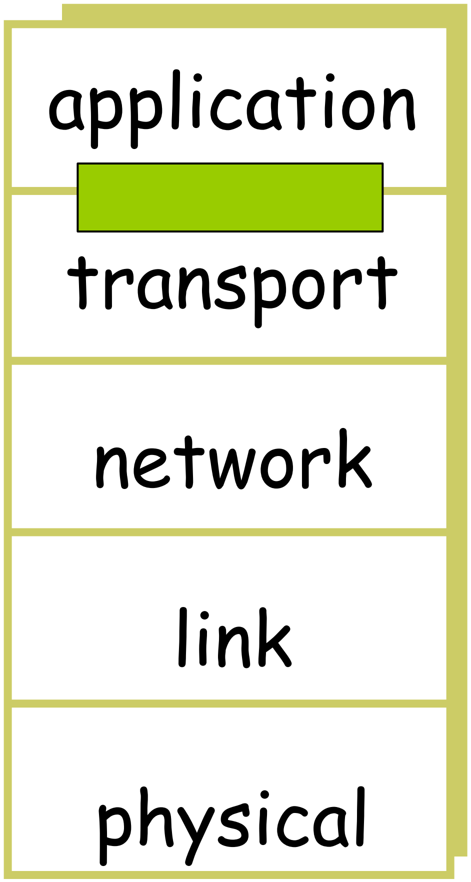


From Computer Networking by Kurose and Ross.

Application and middleware layers use the services provided by the network and transport layers through socket API.

SOCKETS

(MORE IN PART 2)



From Computer Networking by Kurose and Ross.

Processes-to-process communication

Process: program running within a host.

- within same host, two processes communicate using inter-process communication (shared memory defined by OS).
- processes in different hosts communicate by exchanging messages using transport layer

Client process: process that initiates communication

Server process: process that waits to be contacted

- Note: applications with P2P architectures have client processes & server processes

From Computer Networking by Kurose and Ross.

Addressing processes

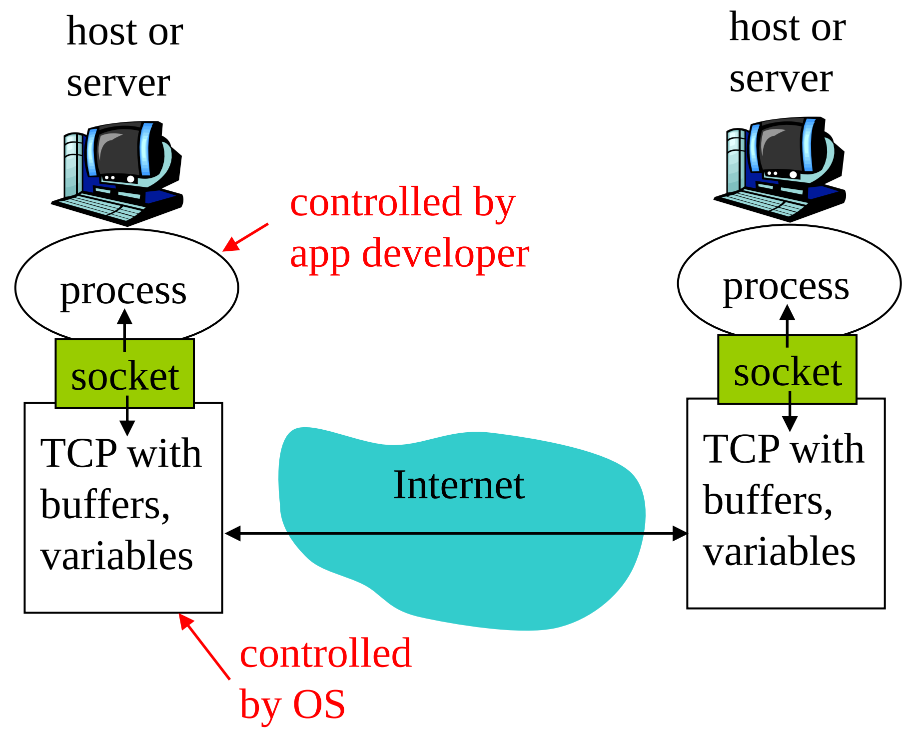
- to receive messages, process must have *identifier*
- host device has unique 32-bit IP address
- *Q:* does IP address of host on which process runs suffice for identifying the process?
  - *A:* No, *many* processes can be running on the same host
- *identifier* includes both IP address and port number associated with the process
- What is a port number?
  - 16 bits integer used by transport layer to identify end points (processes) on a host
  - well-known ports: 1 – 1023
    Telnet 23; FTP 21; HTTP 80
  - registered ports: 1024 – 49151
  - dynamic or private ports: 49152 - 65535

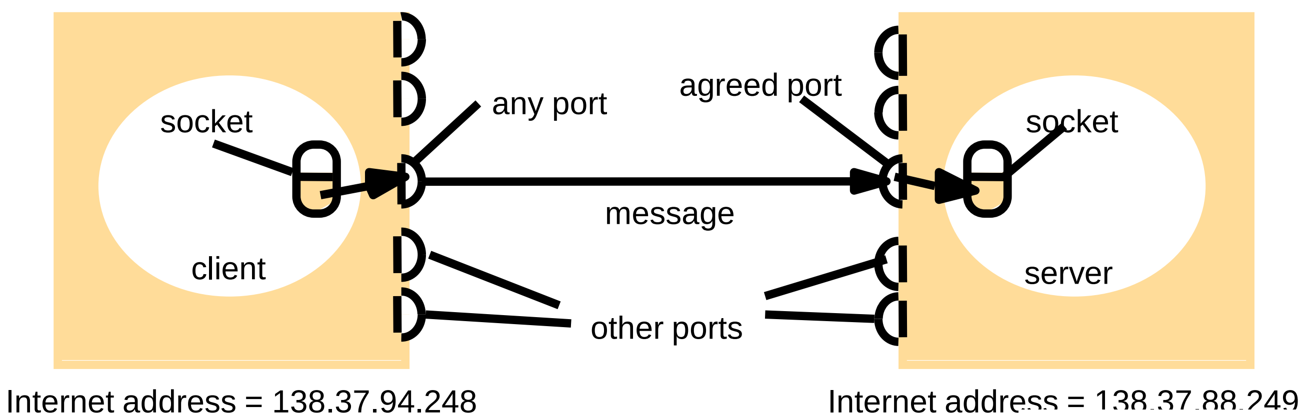
To communicate, client must know the server's IP address, and port number.
How will the server know the client's IP address and port number?

From Computer Networking by Kurose and Ross.

Sockets

- API, an interface, gate, door between a process and transport layer
- A socket must be bound to a local port
- Is (IP addr, port) enough to identify a socket?





From Computer Networking by Kurose and Ross.

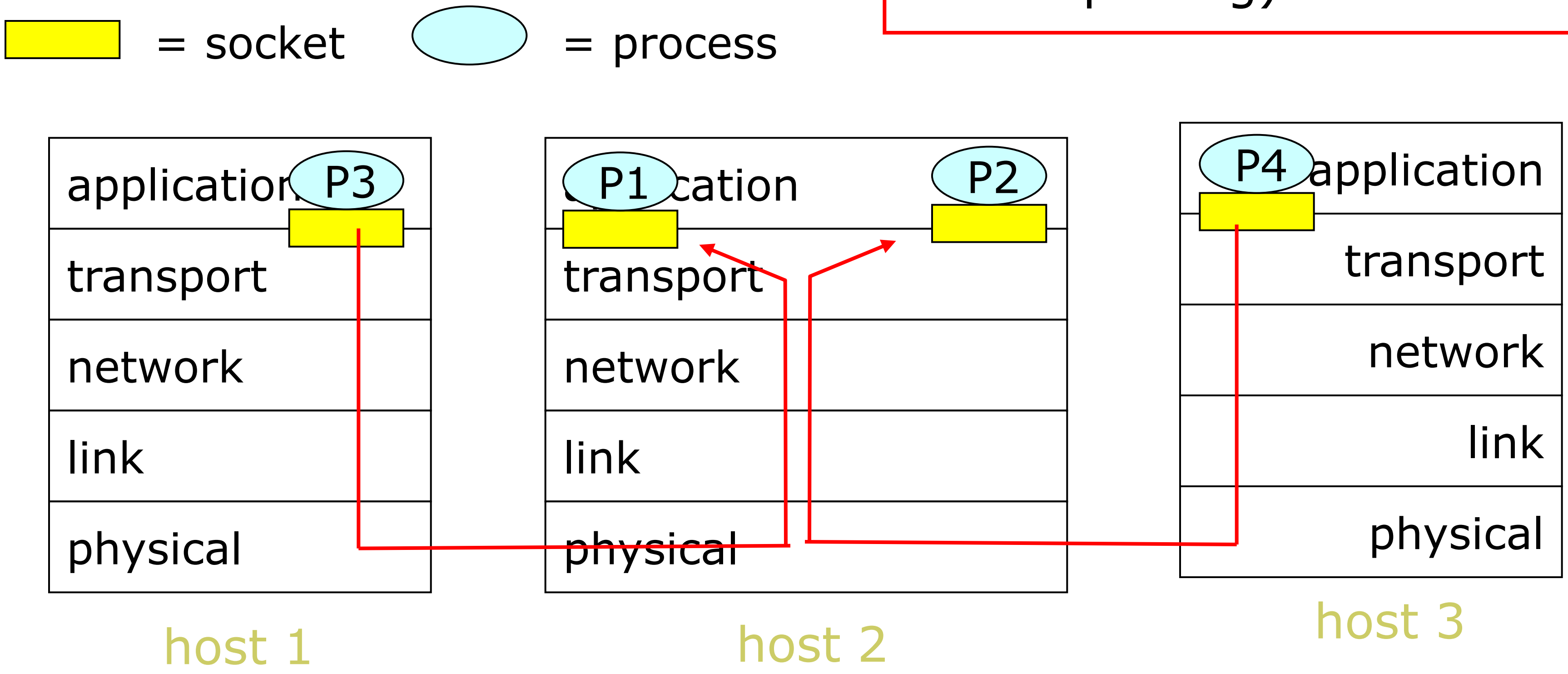
Multiplexing/demultiplexing

Demultiplexing at rcv host:

delivering received segments to correct socket

Multiplexing at send host:

gathering data from multiple sockets, enveloping data with header (later used for demultiplexing)



From Computer Networking by Kurose and Ross.

UDP: Connectionless demultiplexing

- Create sockets with port numbers:

```
DatagramSocket mySocket1 = new
   DatagramSocket(12534);
```

```
DatagramSocket mySocket2 = new
   DatagramSocket(12535);
```

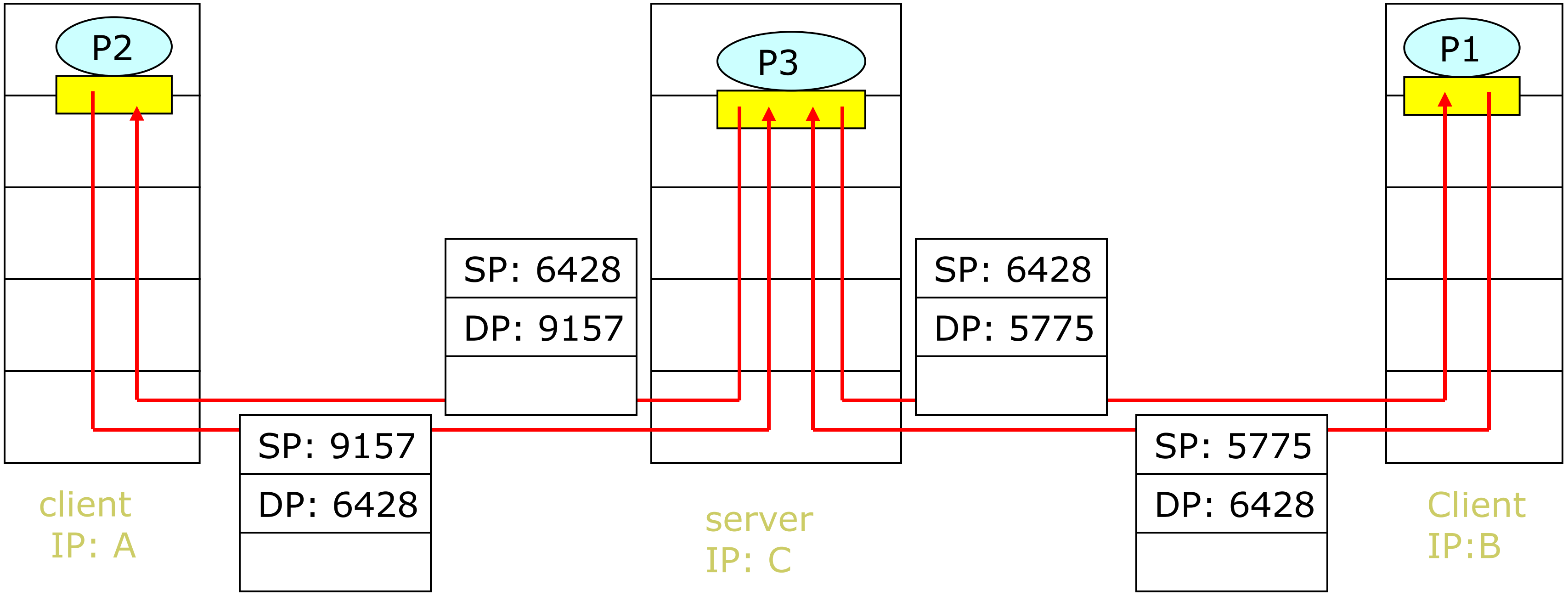
- UDP socket identified by two-tuple:

(dest IP address, dest port number)

- When host receives UDP segment:
  - checks destination port number in segment
  - directs UDP segment to socket with that port number
- IP datagrams with different source IP addresses and/or source port numbers directed to same socket

Connectionless demux (cont)

```
DatagramSocket serverSocket = new DatagramSocket(6428);
```



SP provides "return address"

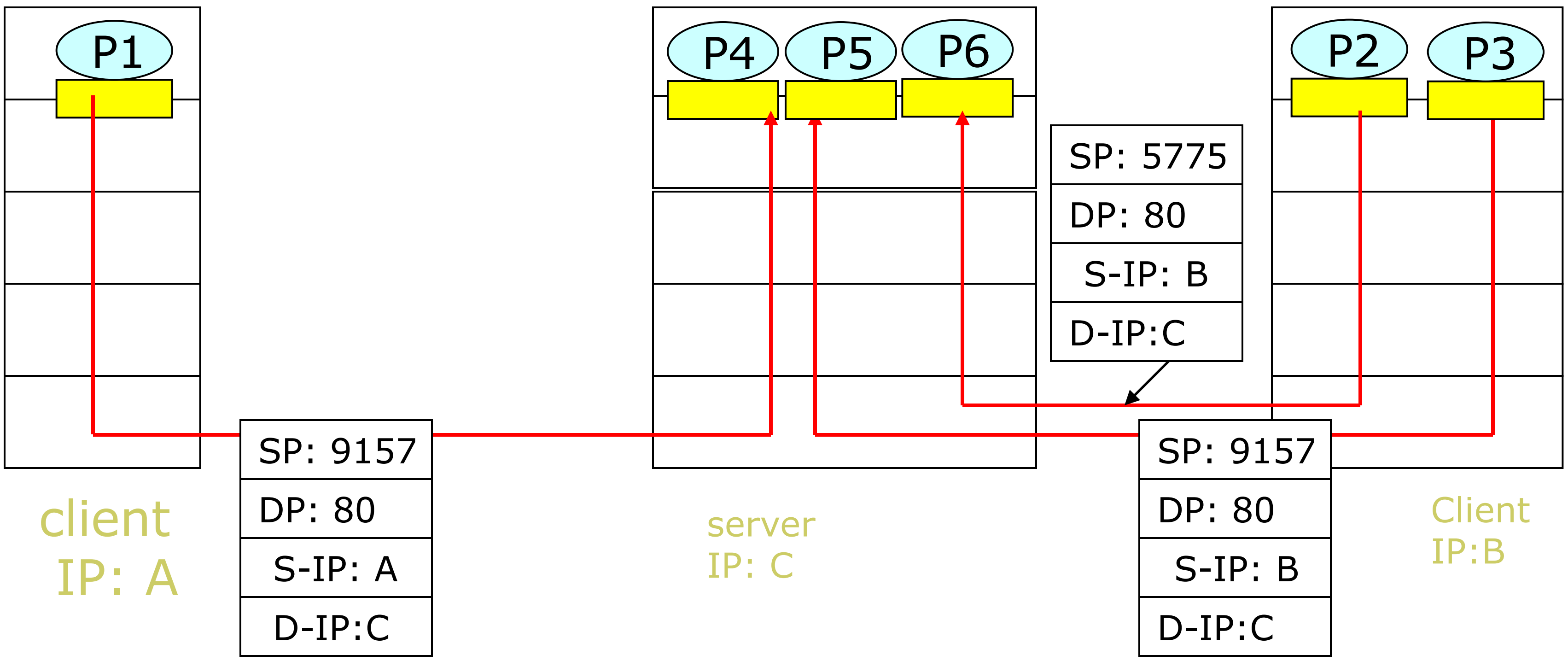
From Computer Networking by Kurose and Ross.

TCP: Connection-oriented demux

- TCP socket identified by 4-tuple:
  - source IP address
  - source port number
  - dest IP address
  - dest port number
- recv host uses all four values to direct segment to appropriate socket

- Server host may support many simultaneous TCP sockets:
  - each socket identified by its own 4-tuple
- Web servers have different sockets for each connecting client
  - non-persistent HTTP will have different socket for each request

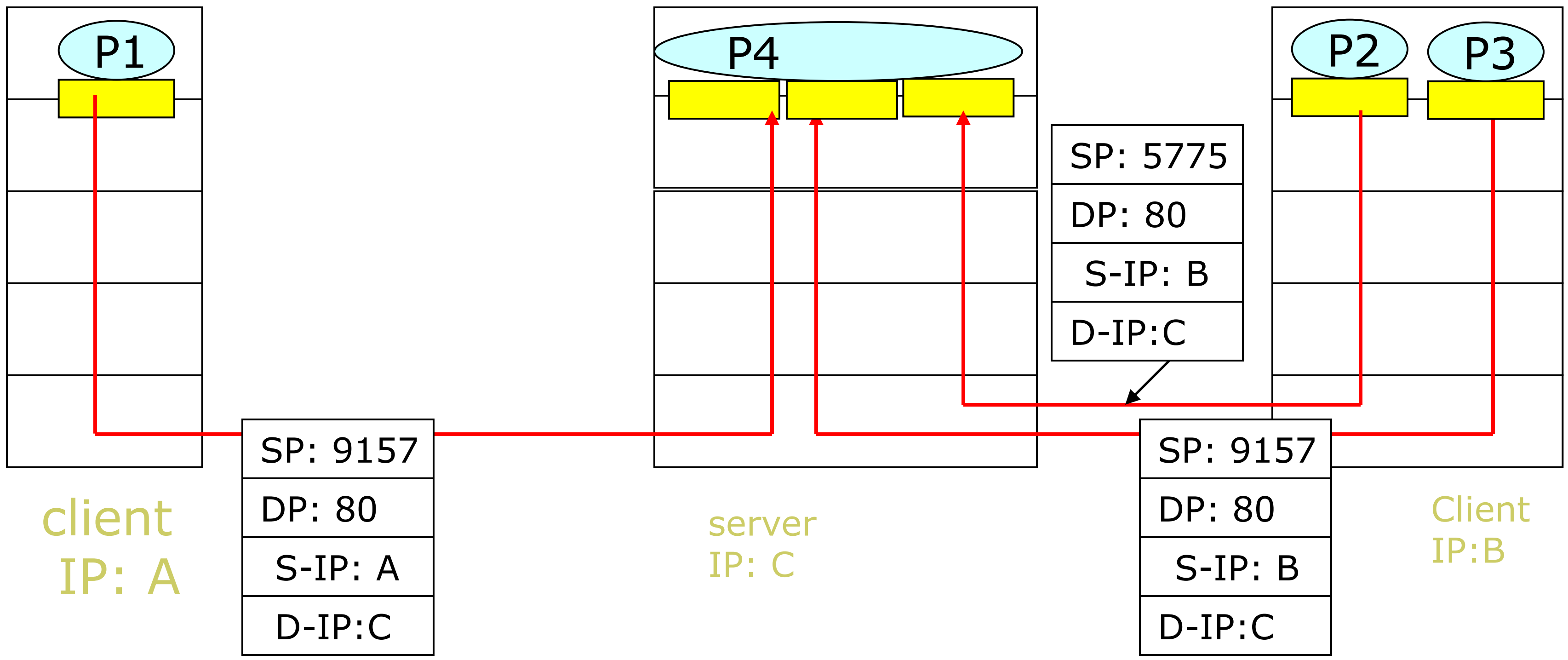
Connection-oriented demux (cont)



From Computer Networking by Kurose and Ross.

Connection-oriented demux:

Threaded Web Server



From Computer Networking by Kurose and Ross.

TCP Connection Management

Recall: TCP sender, receiver establish "connection" before exchanging data segments

- initialize TCP variables:
  - seq. #s
  - buffers, flow control info (e.g. **RcvWindow**)
- *client:* connection initiator

```
Socket clientSocket = new
 Socket("hostname","port

 number");
```

- *server:* contacted by client

```
Socket connectionSocket =
 welcomeSocket.accept();
```

Three way handshake:

Step 1: client host sends TCP SYN segment to server

- specifies initial seq #
- no data

Step 2: server host receives SYN, replies with SYNACK segment

- server allocates buffers
- specifies server initial seq. #

Step 3: client receives SYNACK, replies with ACK segment, which may contain data

From Computer Networking by Kurose and Ross.

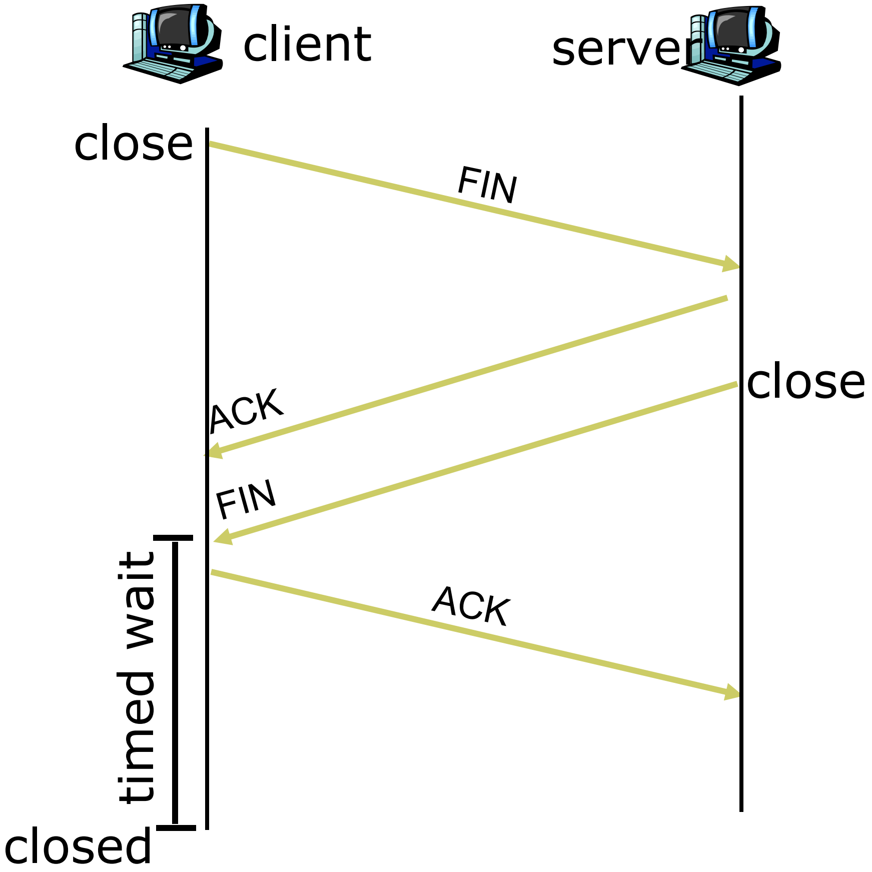
TCP Connection Management (cont.)

<u>Closing a connection:</u>

client closes socket:
`clientSocket.close();`

<u>Step 1:</u> client end system sends TCP FIN control segment to server

<u>Step 2:</u> server receives FIN, replies with ACK. Closes connection, sends FIN.



From Computer Networking by Kurose and Ross.

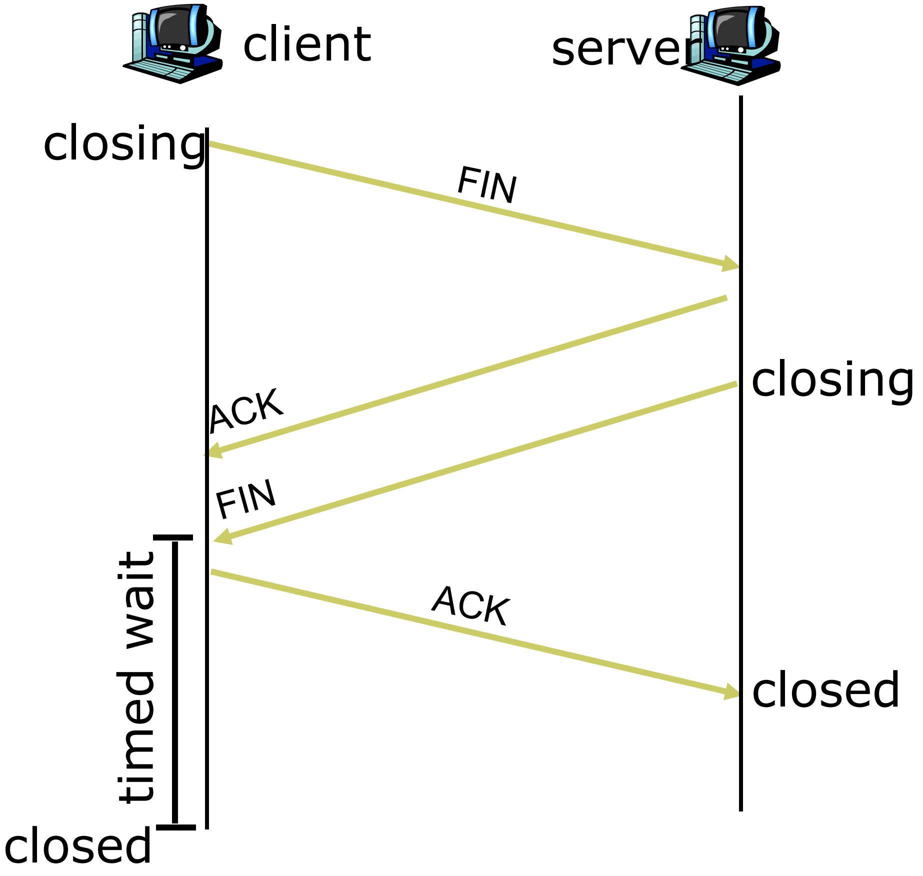
TCP Connection Management (cont.)

Step 3: client receives FIN, replies with ACK.

- Enters "timed wait" - will respond with ACK to received FINs

Step 4: server, receives ACK. Connection closed.

Note: with small modification, can handle simultaneous FINs.



From Computer Networking by Kurose and Ross.

If time permits

More Receivers

MULTICAST COMMUNICATION AT NETWORK LAYER

Multicast Communication

- **Broadcast** – sends a single message from one process to **all** processes (hosts)
  - Used for ARP in a LAN
  - Hard and expensive in WAN
- **Multicast** – sends a single message from one process to members of a group of processes (hosts)
- Who needs multicast?
- Who should provide it?
  - Application, transport, network layer?

Who needs it?

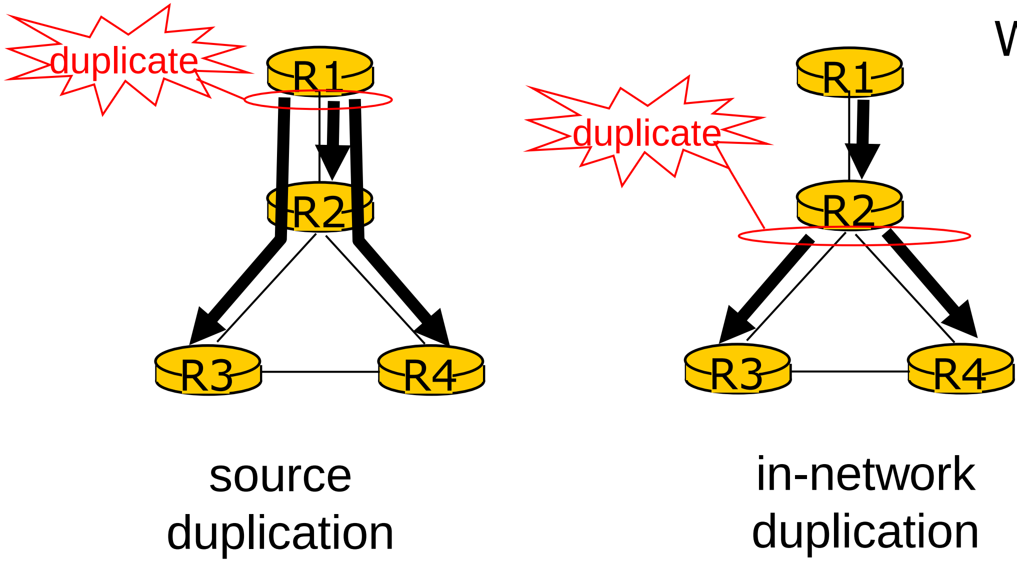
Uses of Multicast and Its Effects

- Fault tolerance based on replicated services
  - Requests multicast to group of servers
- Discovery in spontaneous networking
  - Locate available discovery services
- Performance from replicated data
  - Multicast changes to all replicas
- Propagation of event notifications in a distributed environment
  - News group: news → group of interested users

Who provides it?

Source vs. In-network Duplication

- Deliver packets from source to all other nodes
- Source duplication is inefficient:



What are needed?

Address to identify all members in the group

Multicast routers to forward multicast packet

IP multicasting is often considered a standard available service (which may be dangerous to assume). Actually, it is often disabled!

Application-Level Multicast (more later)

From Computer Networking by Kurose and Ross.

Multicast IP address

	octet 1	octet 2	octet 3		Range of add
	Network ID		Host ID		
Class A:	1 to 127	0 to 255	0 to 255	0 to 255	1.0.0.0 to 127.255.255.
	Network ID		Host ID		
Class B:	128 to 191	0 to 255	0 to 255	0 to 255	128.0.0.0 t 191.255.255.
	Network ID			Host ID	
Class C:	192 to 223	0 to 255	0 to 255	1 to 254	192.0.0.0 t 223.255.255.
	Multicast address				
Class D (multicast):	224 to 239	0 to 255	0 to 255	1 to 254	224.0.0.0 t 239.255.255.
Class E (reserved):	240 to 255	0 to 255	0 to 255	1 to 254	128.0.0.0 t 247.255.255.

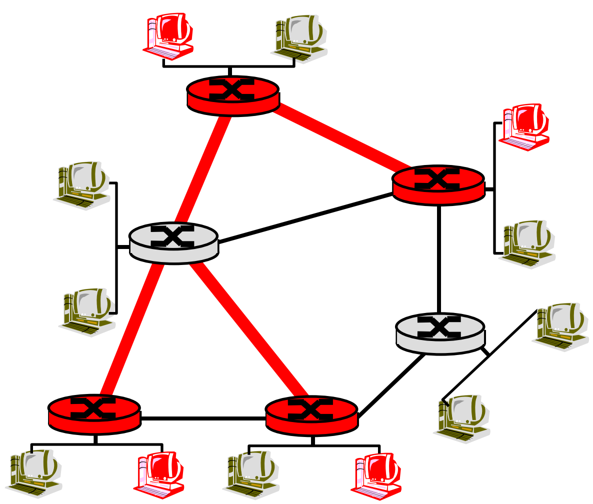
- 224.0.0.0 to 224.0.0.255 (224.0.0.0/24) → **local** subnet multicast traffic
- 224.0.1.0 to 238.255.255.255 → **globally** scoped addresses
- 239.0.0.0 to 239.255.255.255 (239.0.0.0/8) → **administratively** scoped addresses, boundary

IP Multicast Process

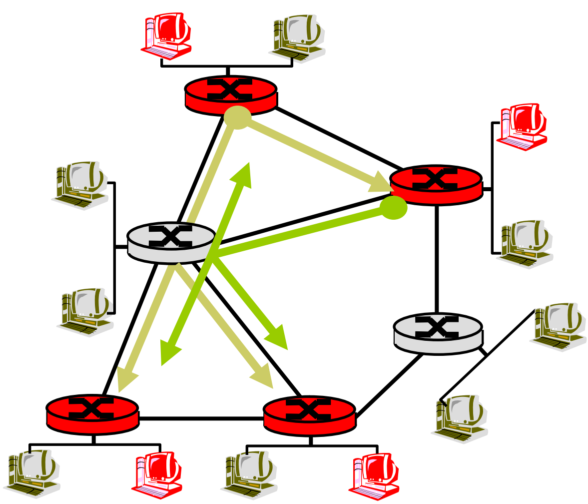
- Each multicast address → identify a group
- Internet Group Membership Protocol (IGMP)
  - Processes **register** a group with **local router** using **IGMP**
- Router update its multicast routing table
- Processes send message to a group
  - Do not need to be a member
- Router forward multicast messages

Multicast Routing Problem

- ***Goal:*** find a tree (or trees) connecting routers having local mcast group members
  - *tree:* not all paths between routers used
  - *source-based:* different tree from each sender to rcvrs
  - *shared-tree:* same tree used by all group members



Shared tree



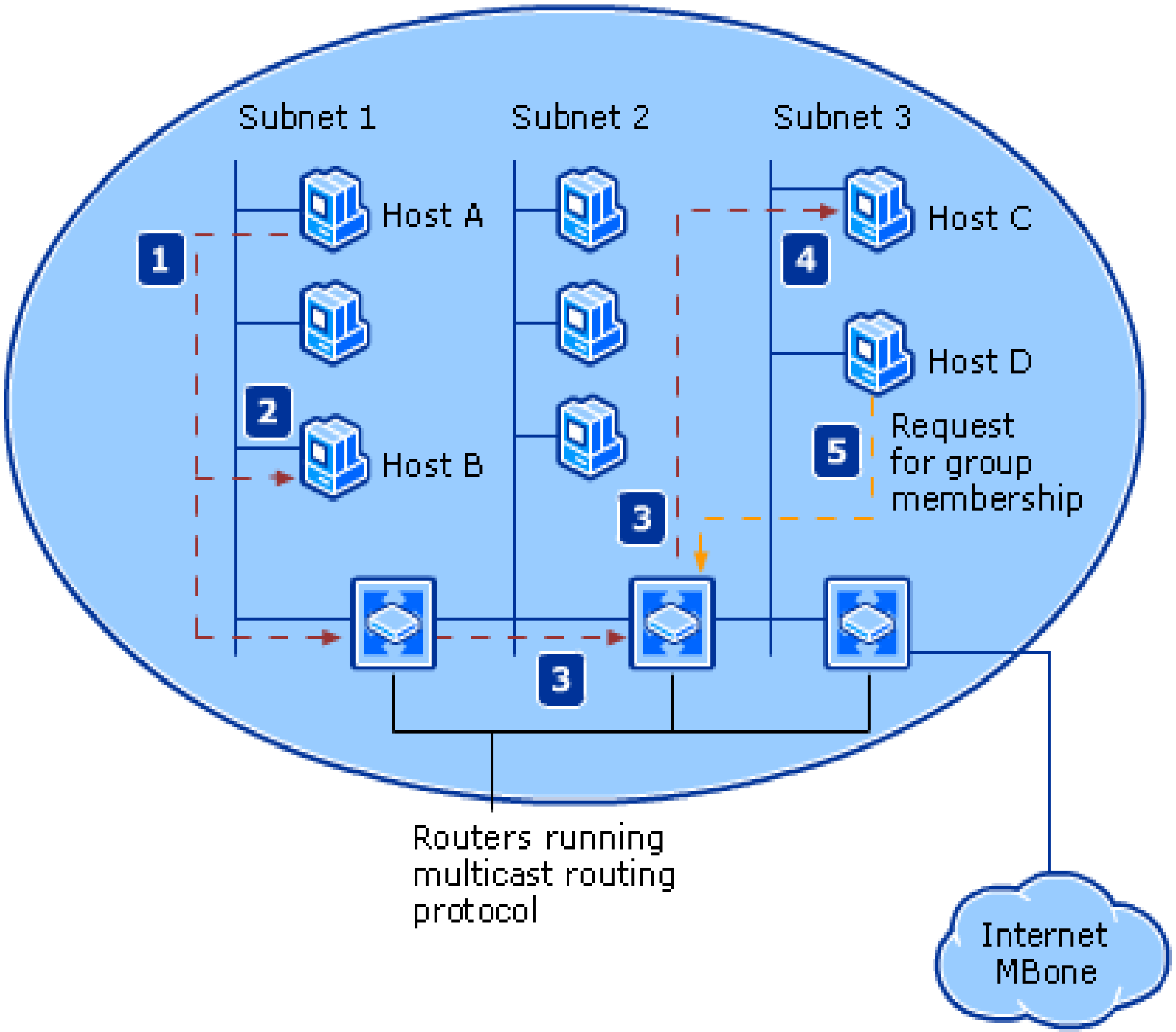
Source-based trees

DVMRP: distance vector multicast routing protocol, source-based trees, *flood and prune* reverse path forwarding (RPF)

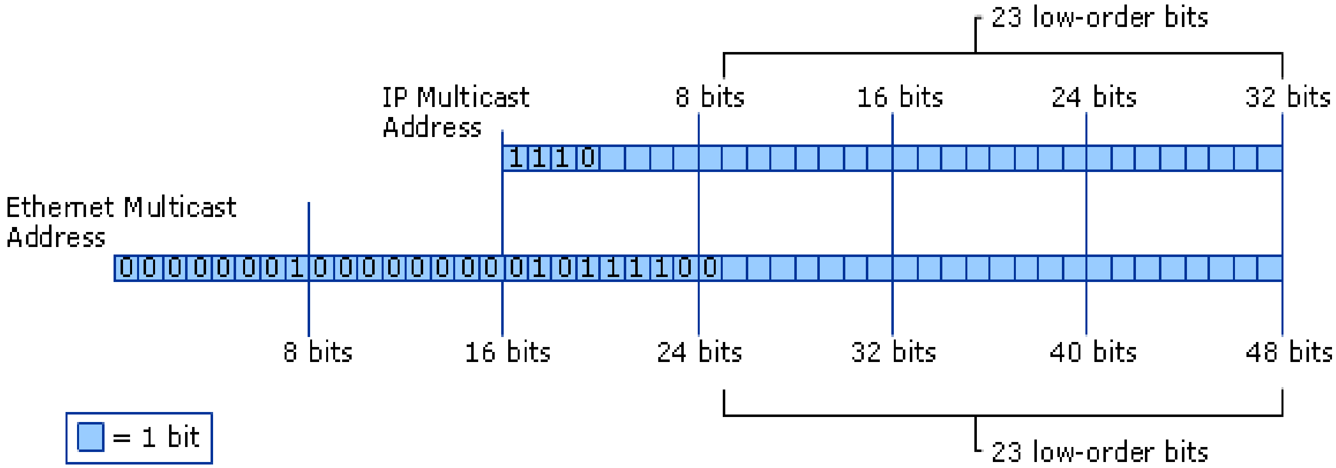
PIM: Protocol Independent Multicast, has two modes:
Dense mode: similar to DVMRP
Sparse mode: center-based approach

From Computer Networking by Kurose and Ross.

Multicast Architecture



What happens under the ground?



- MAC address (Ethernet: 0x01-00-5E-00-00-00 to 0x01-00-5E-7F-FF-FF)
- Map IP multicast address to Ethernet multicast address
- **Network adapter: maintains a table of interested MAC addresses**
  - Normally only has its own MAC address and broadcast address (0xFF-FF-FF-FF-FF-FF)
  - When processes register a group with IP multicast address, corresponding MAC address will be added to the table → forward packets to OS

Range of Multicast Message

- **TTL-based boundaries**
  - Time-To-Live (TTL): number of links/hops before dropped at a router
  - Use TTL to control how far a message can reach
  - Different groups use same multicast address and port number at different regions
- **Scope-based boundaries**
  - administrative scope address: 239.0.0.0 to 239.255.255.255
  - boundary router

TTL Value	Definition
0	Restricted to the same host
1	Restricted to the local subnet, no router hops
32	Restricted to the site
64	Restricted to the region
128	Restricted to the continent
255	Worldwide (unrestricted)

Summary

- Layered network models
  - OSI vs.TCP/IP
- Ethernet and local area network
- Inter-network Protocols (IP)
  - Addressing and routing etc.
- TCP/UDP protocols
  - Communication ports and sockets
  - **Socket Programming (later)**
- Multicast (network layer)