

# 8. Networks



- ⌘ Used to build distributed embedded systems
- ⌘ Processing elements (PEs) are connected by a network
- ⌘ The application is distributed over the PEs

# Network-based embedded systems



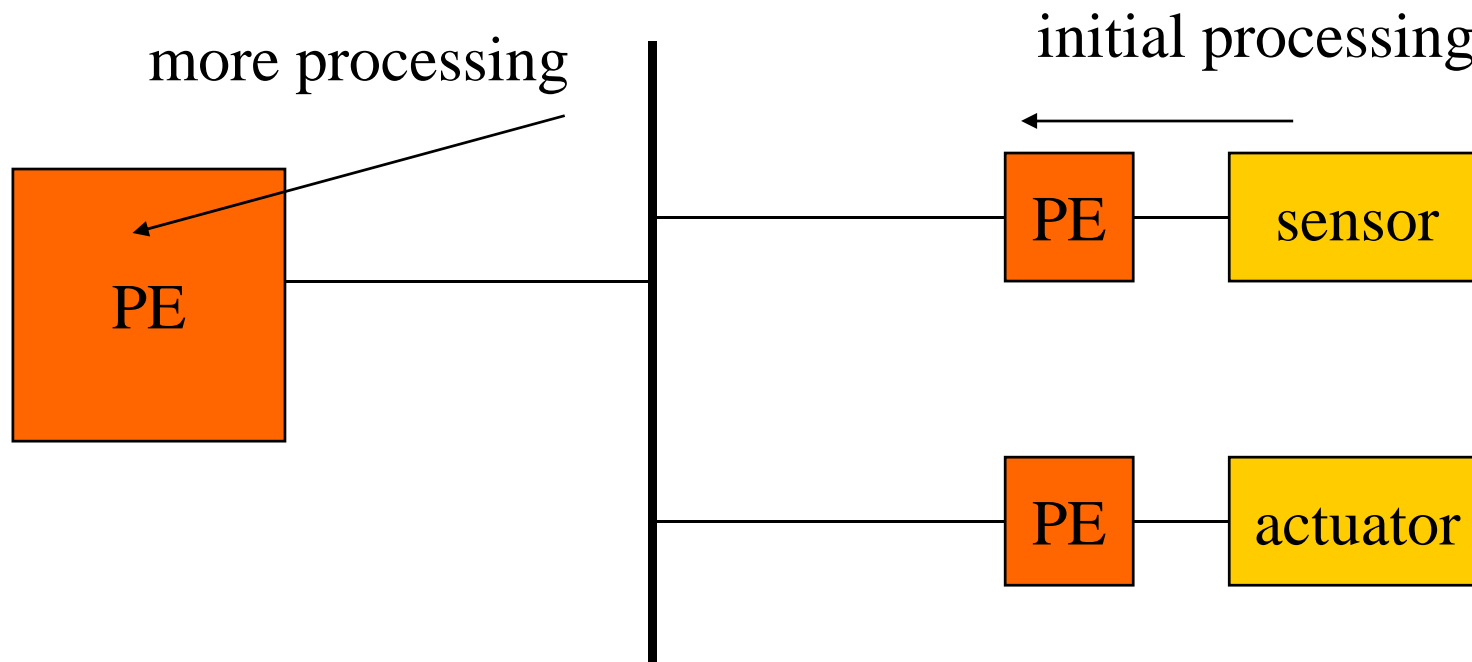
- ⌘ Physically distributed activities---time constants may not allow transmission to central site.
  - ☒ Engine control: short time delays required for the task
- ⌘ Data reduction is another important reason for distributed processing.
- ⌘ Modularity is another motivation for network-based design.
  - ☒ A large system is assembled out of existing components.

# Network-based embedded systems



- ⌘ Improved debugging---use one CPU in network to debug others.
- ⌘ In some cases, networks are used to build fault tolerance into systems.
- ⌘ Distributed embedded system design is a good example of hardware/software co-design since we must design the network topology as well as the software running on the network node.

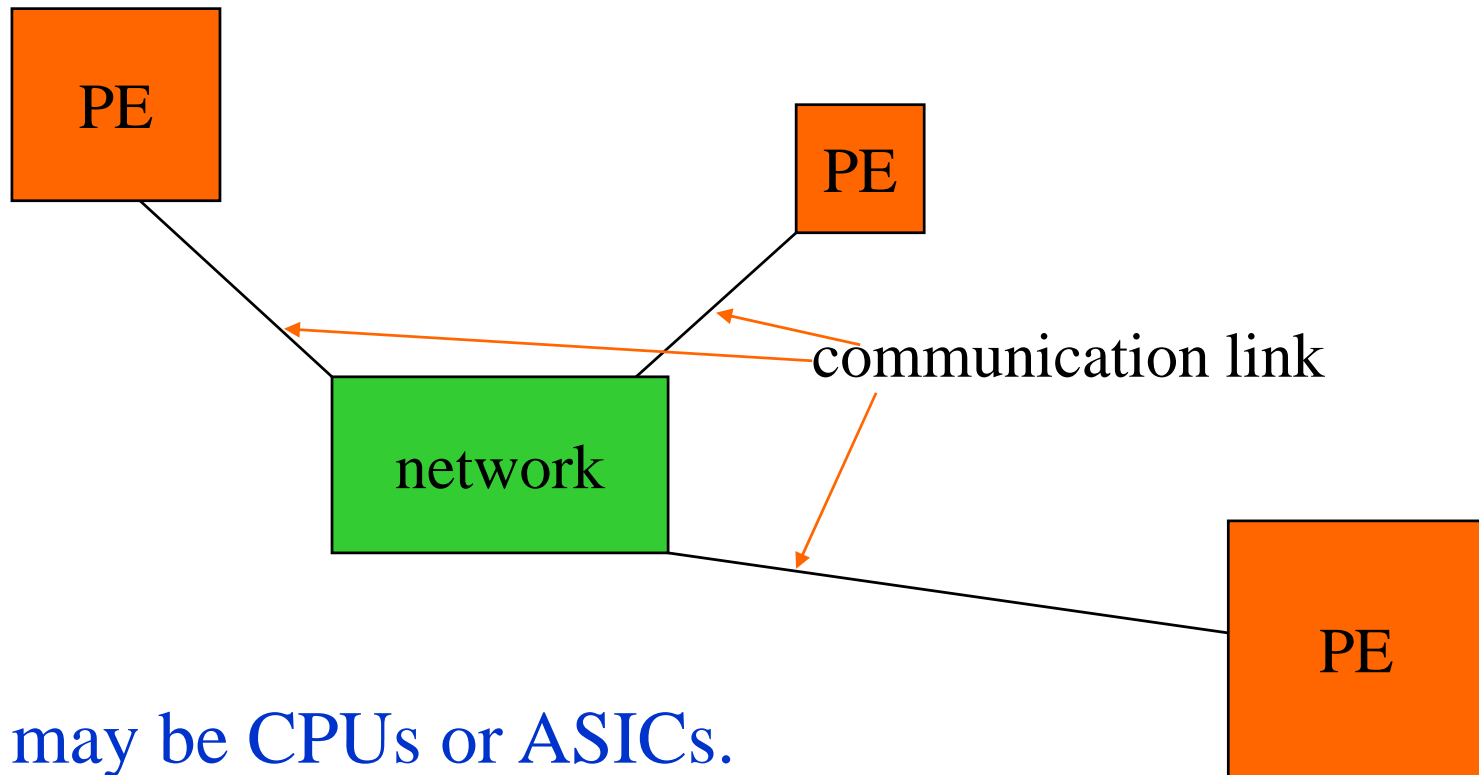
# Networks in embedded systems



It is necessary to put some PEs near where the events occur.

# Network elements

distributed computing platform:



PEs may be CPUs or ASICs.

# Hardware platform



- ⌘ Network of the PEs
- ⌘ Unlike the system bus, the distributed embedded system does not have memory on the bus
- ⌘ PEs do not fetch instruction over the network as they do on the microprocessor bus.

# Why distributed?



## ⌘ Network-based embedded system

- ☑ More complicated

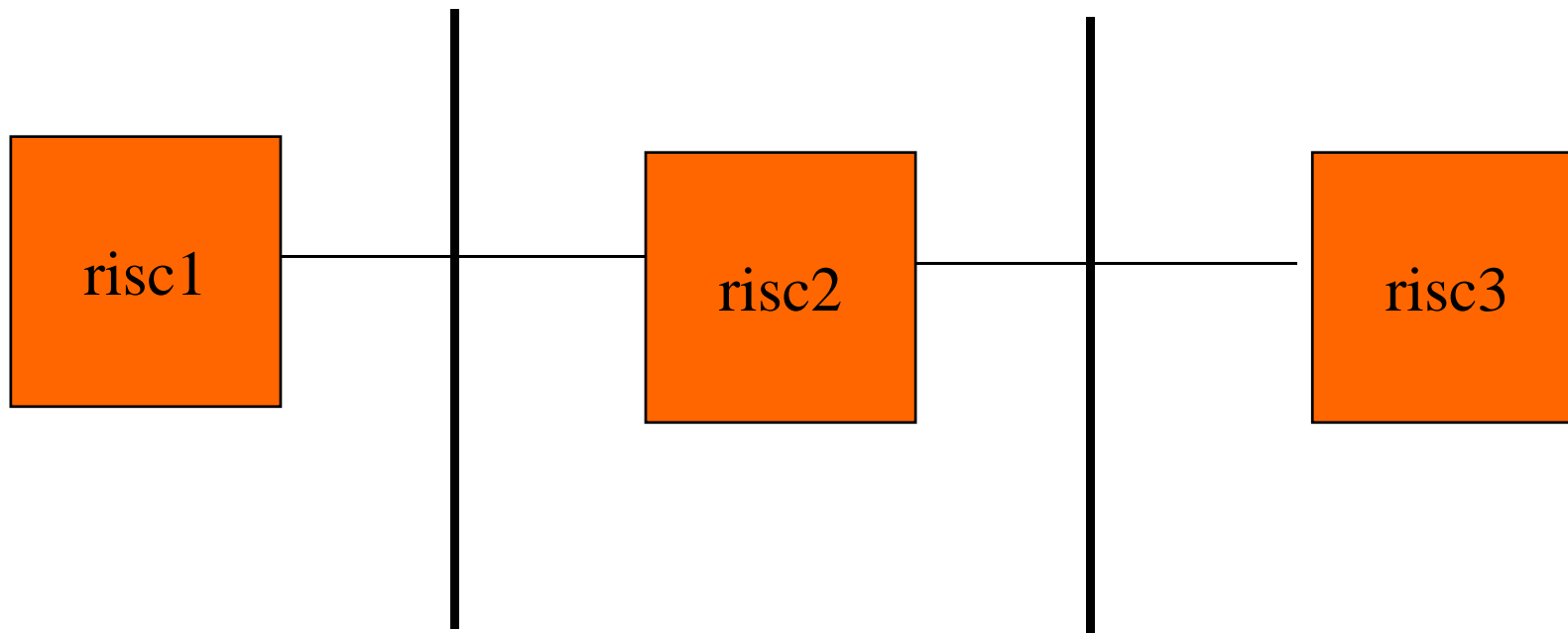
## ⌘ Physically separated & deadline is short

- ☑ Rather than building high-speed network to carry the data to a distant, fast PE.

## ⌘ Advantage of a distributed system with several CPU

- ☑ A part of the system can be used to debug another part.

# Debugging a multi-core system



To diagnose risc 2, we can use risc1 to generate inputs and risc3 to watch output.



# Network abstractions



- ⌘ Networks are complex
- ⌘ Ideally, they provide high-level services while hiding details of data transmission
- ⌘ International Standards Organization (ISO) developed the **Open Systems Interconnection (OSI)** model to help us to understand networks:
  - ☑ 7-layer model.
- ⌘ Provides a standard way to classify network components and operations.

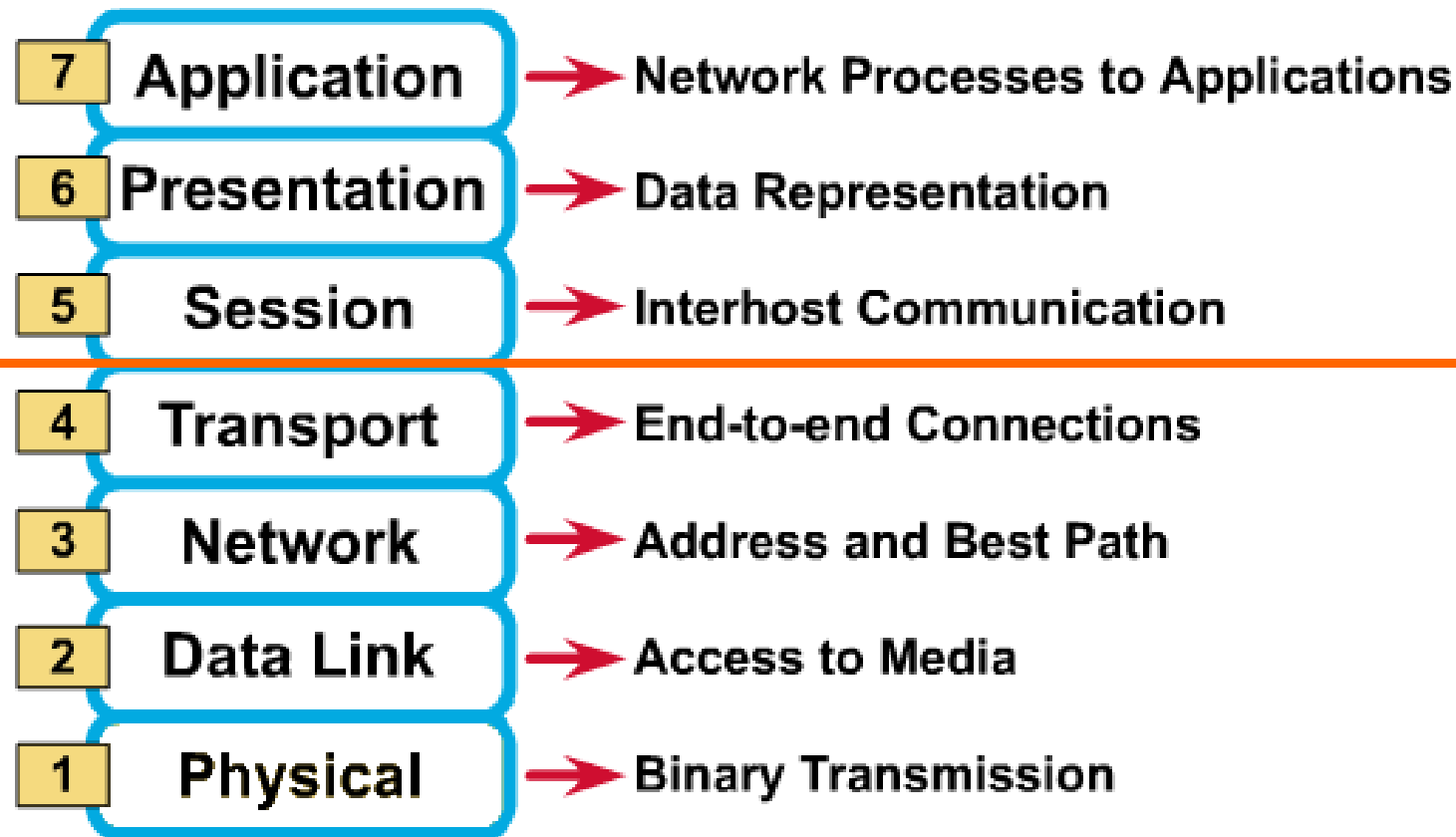
# OSI reference model



7. application	end-use interface
6. presentation	data format
5. session	application dialog control
4. transport	connections
3. network	end-to-end service
2. data link	reliable data transport
1. physical	mechanical, electrical

---

# OSI reference model



# OSI layers



- ⌘ **Physical**: connectors, bit formats, etc.
- ⌘ **Data link**: error detection and control across a single link (single hop).
- ⌘ **Network**: end-to-end multi-hop data communication.
- ⌘ **Transport**: provides connections; may optimize network resources.
- ⌘ **Session**: services for end-user applications: data grouping, checkpointing, etc.
- ⌘ **Presentation**: data formats, transformation services.
- ⌘ **Application**: interface between network and end-user programs.

# LAN - Local Area Network

⌘ connects computers that are physically close together ( < 1 mile).

☑ high speed

☑ multi-access

⌘ Technologies:

☑ Ethernet      10 Mbps, 100Mbps

☑ Token Ring    16 Mbps

☑ FDDI            100 Mbps

# WAN - Wide Area Network



⌘ connects computers that are physically far apart. “long-haul network”.

☑ typically slower than a LAN.

☑ typically less reliable than a LAN.

☑ point-to-point

⌘ Technologies:

☑ telephone lines

☑ Satellite communications

# MAN - Metropolitan Area Network



⌘ Larger than a LAN and smaller than a WAN

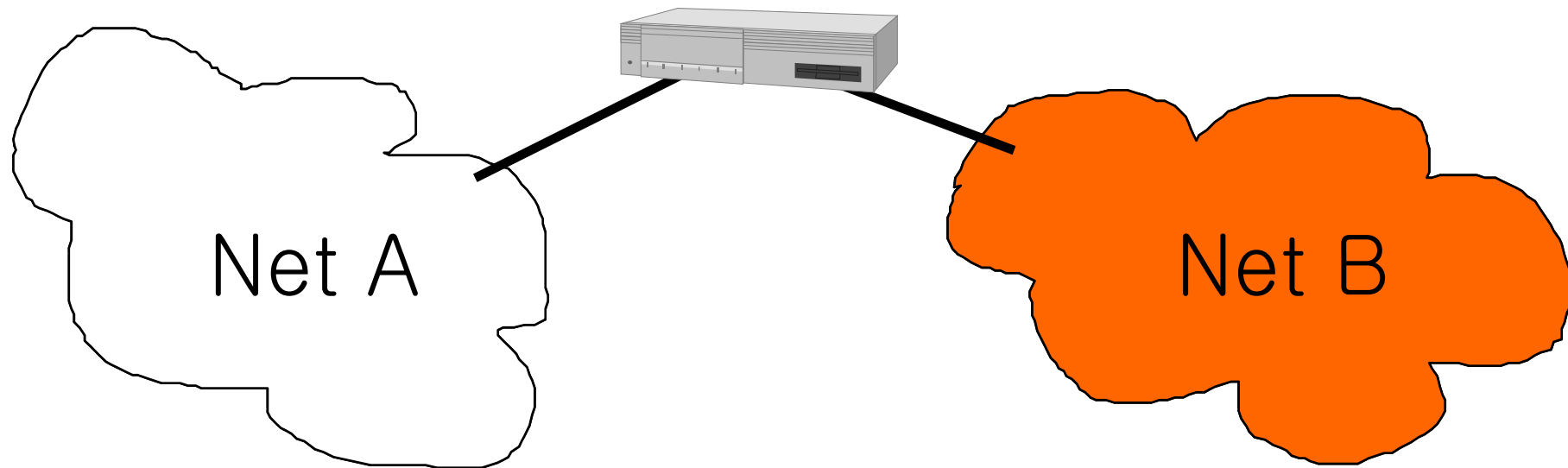
- example: campus-wide network
- multi-access network

⌘ Technologies:

- ☑ coaxial cable
- ☑ microwave

# Internetwork

- ⌘ Connection of 2 or more distinct (possibly dissimilar) networks.
- ⌘ Requires some kind of network device to facilitate the connection.





# Physical Layer

- ⌘ deals with the physical characteristics of the transmission medium.
- ⌘ defines the electrical, mechanical, procedural, and functional specifications for activating, maintaining, and deactivating the physical link between end systems.
- ⌘ defines such characteristics as voltage levels, timing of voltage changes, physical data rates, maximum transmission distances, physical connectors, and other similar attributes
- ⌘ Examples : EIA/TIA-232, RJ45, NRZ.

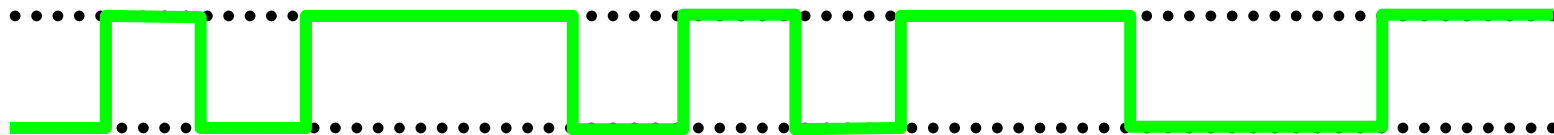
# Physical Layer

## ⌘ Responsibility:

- ☑ transmission of raw bits over a communication channel.

## ⌘ Issues:

- ☑ mechanical and electrical interfaces
- ☑ time per bit
- ☑ distances



# Data Link Layer



- ⌘ provides access to the media and physical transmission across the media, which enables the data to locate its intended destination
- ⌘ provides reliable transit of data across a physical link by using the MAC addresses, which define a hardware or data link address in order for multiple stations to share the same medium and still uniquely identify each other.
- ⌘ Concerned with network topology, network access, error notification, ordered delivery of frames, and flow control.
- ⌘ Examples: Ethernet, Frame Relay, FDDI.

# Data Link Layer



## ⌘ Two sublayers

☒ Data link control, which provide an error-free communication link

☒ *framing* (dividing data into chunks)

- header & trailer bits

☒ addressing

☒ MAC, which is needed by multiaccess networks.

☒ MAC provides DLC with “virtual wires” on multiaccess networks.

# Network Layer



⌘ defines

- ☑ end-to-end delivery of packets.
- ☑ logical addressing so that any endpoint can be identified.
- ☑ how routing works and how routes are learned so that the packets can be delivered.
- ☑ how to fragment a packet into smaller packets to accommodate different media.

⌘ Routers operate at Layer 3.

# Network Layer



⌘ Examples: IP, IPX, AppleTalk.

⌘ Responsibilities:

☑ path selection between end-systems (routing).

☑ subnet flow control.

☑ fragmentation & reassembly

☑ translation between different network types.

⌘ Issues:

☑ *packet* headers

☑ virtual circuits

# Transport Layer



- ⌘ regulates information flow to ensure end-to-end connectivity between host applications reliably and accurately.
- ⌘ segments data from the sender and reassembles the data into a data stream on the receiver.
- ⌘ the lower four layers are concerned with data transport issues.

# Transport Layer

## ⌘ Layer 4 protocols include

- ☑ TCP (Transmission Control Protocol) and
- ☑ UDP (User Datagram Protocol).

## ⌘ Responsibilities:

- ☑ provides virtual end-to-end links between peer processes.
- ☑ end-to-end flow control

## ⌘ Issues:

- ☑ headers
- ☑ error detection
- ☑ reliable communication



# Session Layer



- ⌘ defines how to start, control and end sessions between applications.
- ⌘ includes the control and management of multiple bi-directional messages using dialogue control.
- ⌘ synchronizes dialogue between two hosts' presentation layers and manages their data exchange.
- ⌘ The upper three layers (the application, presentation, and session layers) are concerned with application issues.

# Session Layer

- ⌘ The session layer offers provisions for efficient data transfer.
- ⌘ Examples: SQL, ASP (AppleTalk Session Protocol).
- ⌘ Responsibilities:
  - ☑ establishes, manages, and terminates sessions between applications.
  - ☑ service location lookup
- ⌘ Many protocol suites do not include a session layer.

# Presentation Layer



- ⌘ The information that the application layer of one system sends out is ensured to be read by the application layer of another system.
- ⌘ If necessary, translates between multiple data formats by using a common format.
- ⌘ provides encryption and compression of data.

# Presentation Layer



⌘ Examples: JPEG, MPEG, ASCII, EBCDIC, HTML.

⌘ Responsibilities:

- ☑ data encryption

- ☑ data compression

- ☑ data conversion

⌘ Many protocol suites do not include a Presentation Layer.

# Application Layer

- ⌘ provides network services to the user's applications.
- ⌘ does not provide services to any other OSI layer, but rather, only to applications outside the OSI model.
- ⌘ Examples of such applications: spreadsheet programs, word processing programs, and bank terminal programs.

# Application Layer

⌘ establishes the availability of intended communication partners, synchronizes and establishes agreement on procedures for error recovery and control of data integrity.

⌘ Responsibilities:

☒ anything not provided by any of the other layers

⌘ Issues:

☒ application level protocols

☒ appropriate selection of “type of service”

# TCP/IP protocols

- ⌘ Application layer (telnet, ssh, http, ftp, etc)
  - ☒ Main functionality: run application protocols
- ⌘ Transport layer (TCP, UDP)
  - ☒ Main functionality: reliability
- ⌘ Network layer (IPv4, IPv6)
  - ☒ Main functionality:  
routing/fragmentation/internetworking
- ⌘ Host to Network layer (Ethernet)
  - ☒ Main functionality: medium access, encoding

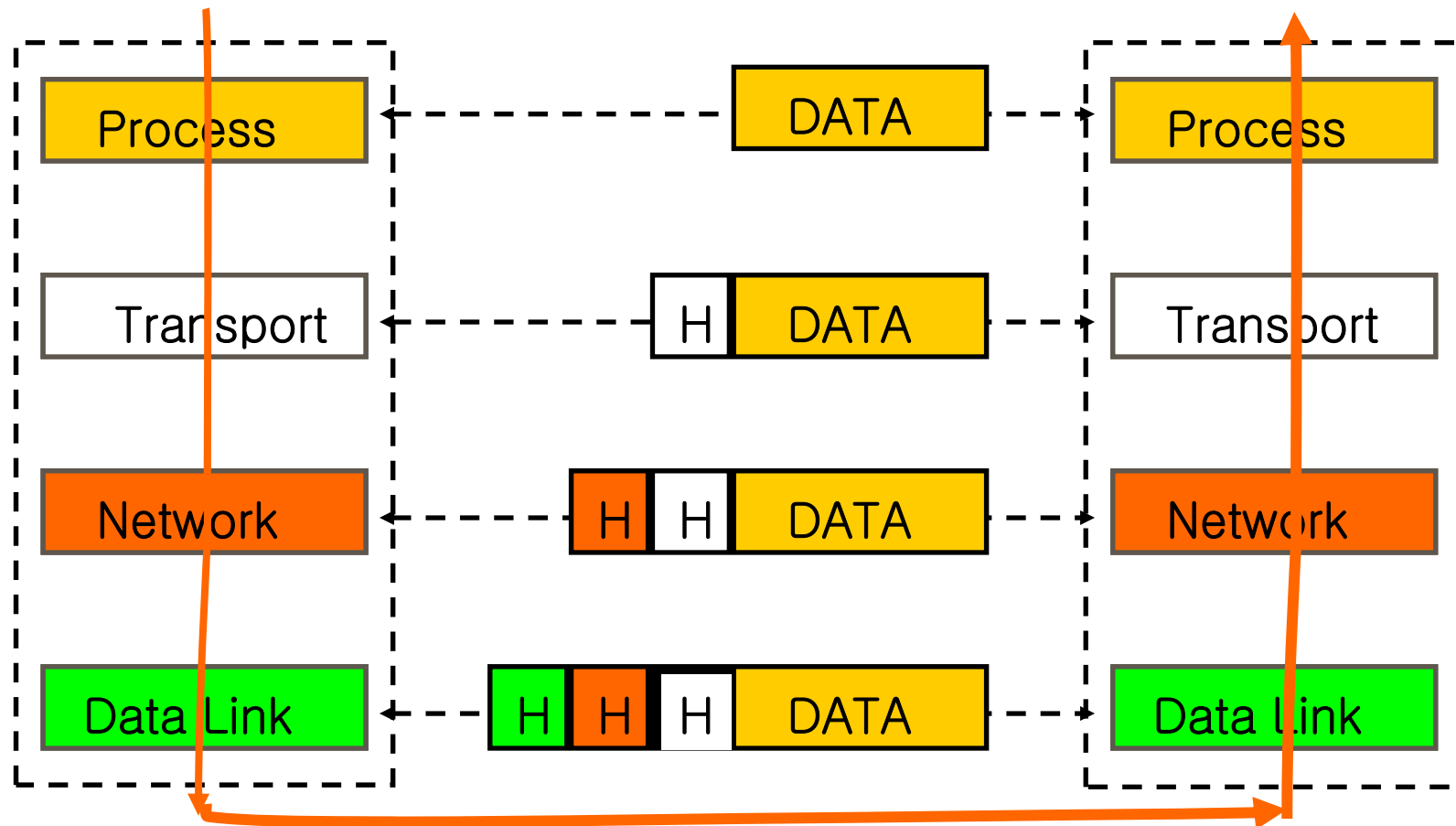
# Layering & Headers



- ⌘ Each layer needs to add some control information to the data in order to do its job.
- ⌘ This information is typically appended to the data before being given to the lower layer.
- ⌘ Once the lower layers deliver the the data and control information - the peer layer uses the control information.



# Headers



# What are the headers?



Physical: no header - just a bunch of bits.

Data Link:

- ☒ address of the receiving endpoints
- ☒ address of the sending endpoint
- ☒ length of the data
- ☒ checksum.

# Network layer header - examples

- ⌘ protocol suite version
  - protocol
- ⌘ type of service
  - header checksum
- ⌘ length of the data
  - source network address
- ⌘ packet identifier
  - destination network address
- ⌘ fragment number
- ⌘ time to live

# Important Summary



- ⌘ Data-Link: communication between machines on the same network.
- ⌘ Network: communication between machines on possibly different networks.
- ⌘ Transport: communication between processes (running on machines on possibly different networks).

# Connecting Networks



- ⌘ Repeater: physical layer
- ⌘ Bridge: data link layer
- ⌘ Router: network layer
- ⌘ Gateway: network layer and above.

# Repeater

- ⌘ Copies bits from one network to another
- ⌘ Does not look at any bits
- ⌘ Allows the extension of a network beyond physical length limitations



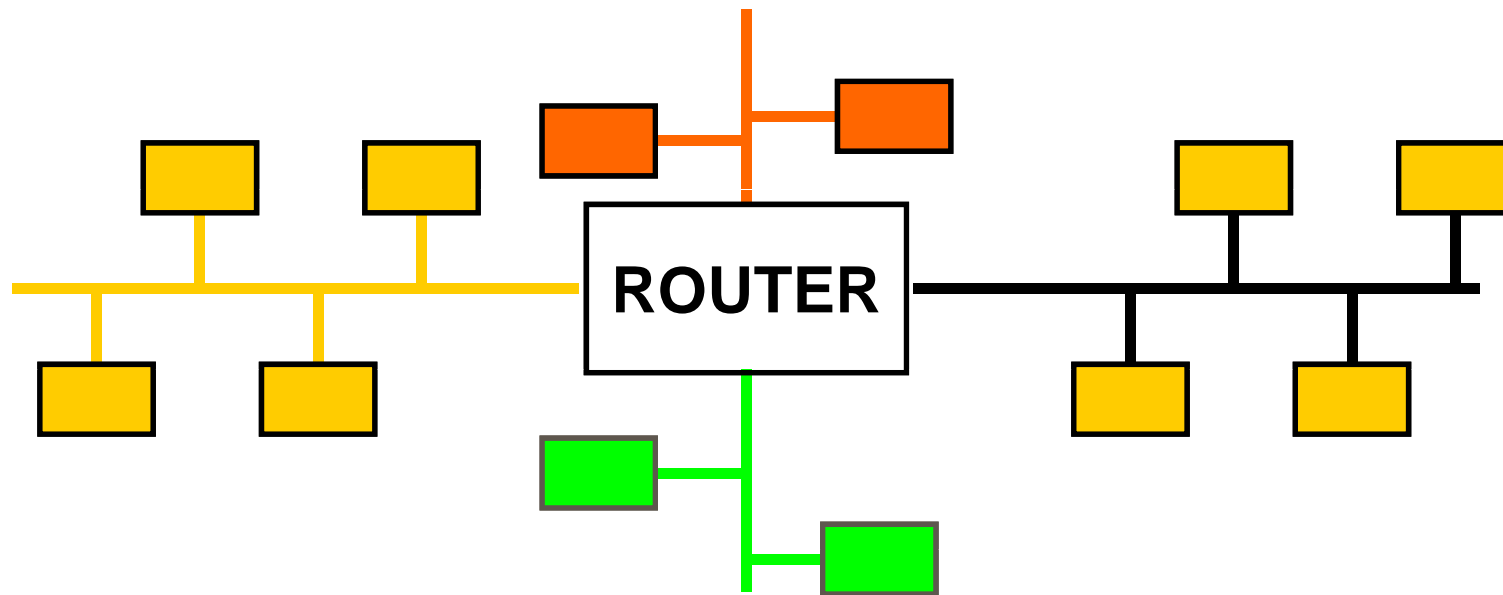
# Bridge

- ⌘ Copies frames from one network to another
- ⌘ Can operate selectively - does not copy all frames (must look at data-link headers).
- ⌘ Extends the network beyond physical length limitations.



# Router

- ⌘ Copies packets from one network to another.
- ⌘ Makes decisions about what *route* a packet should take (looks at network headers).



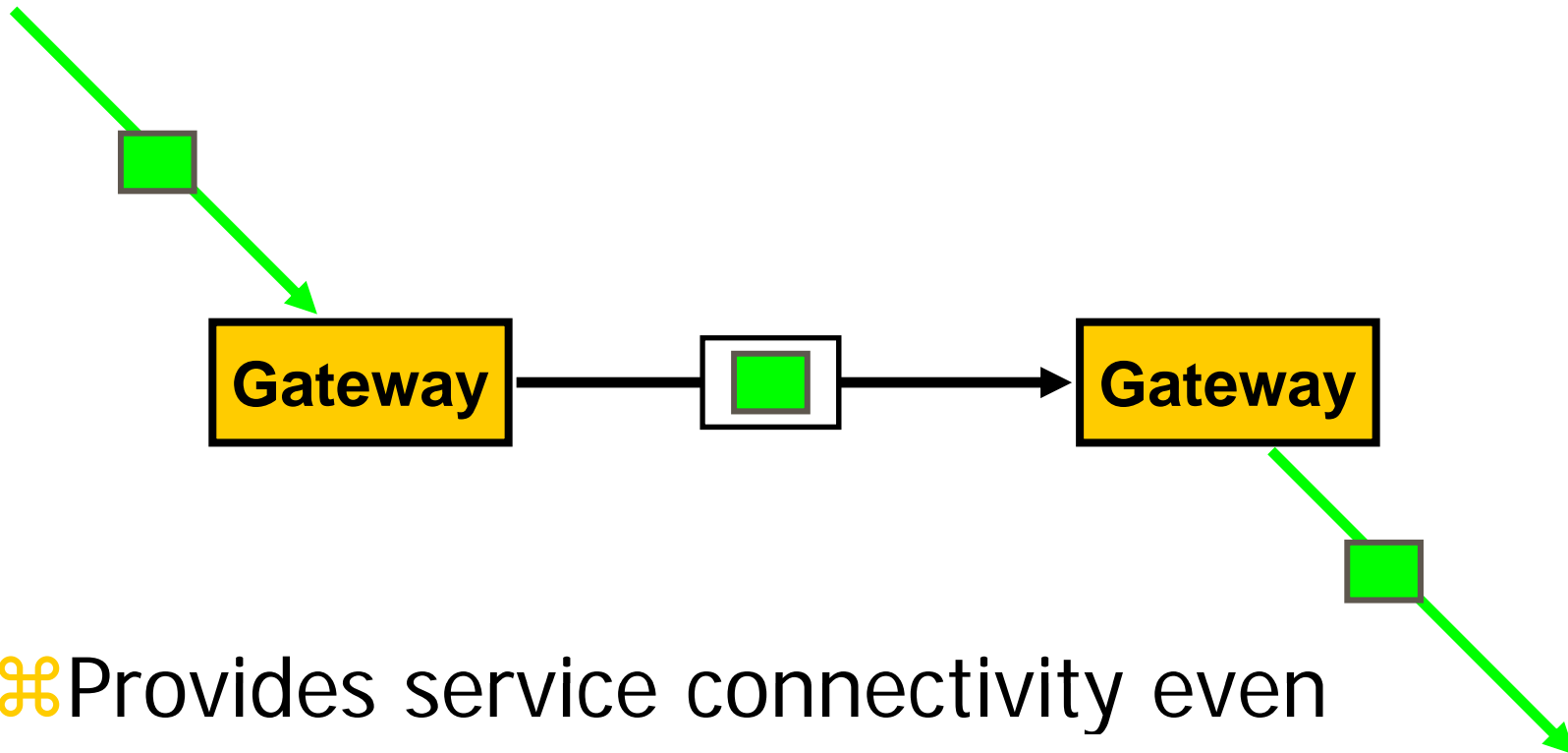


# Gateway



- ⌘ Operates as a router
- ⌘ Data conversions above the network layer.
- ⌘ Conversions:
  - encapsulation - use an intermediate network
  - translation - connect different application protocols
  - encrpytion - could be done by a gateway

# Encapsulation Example



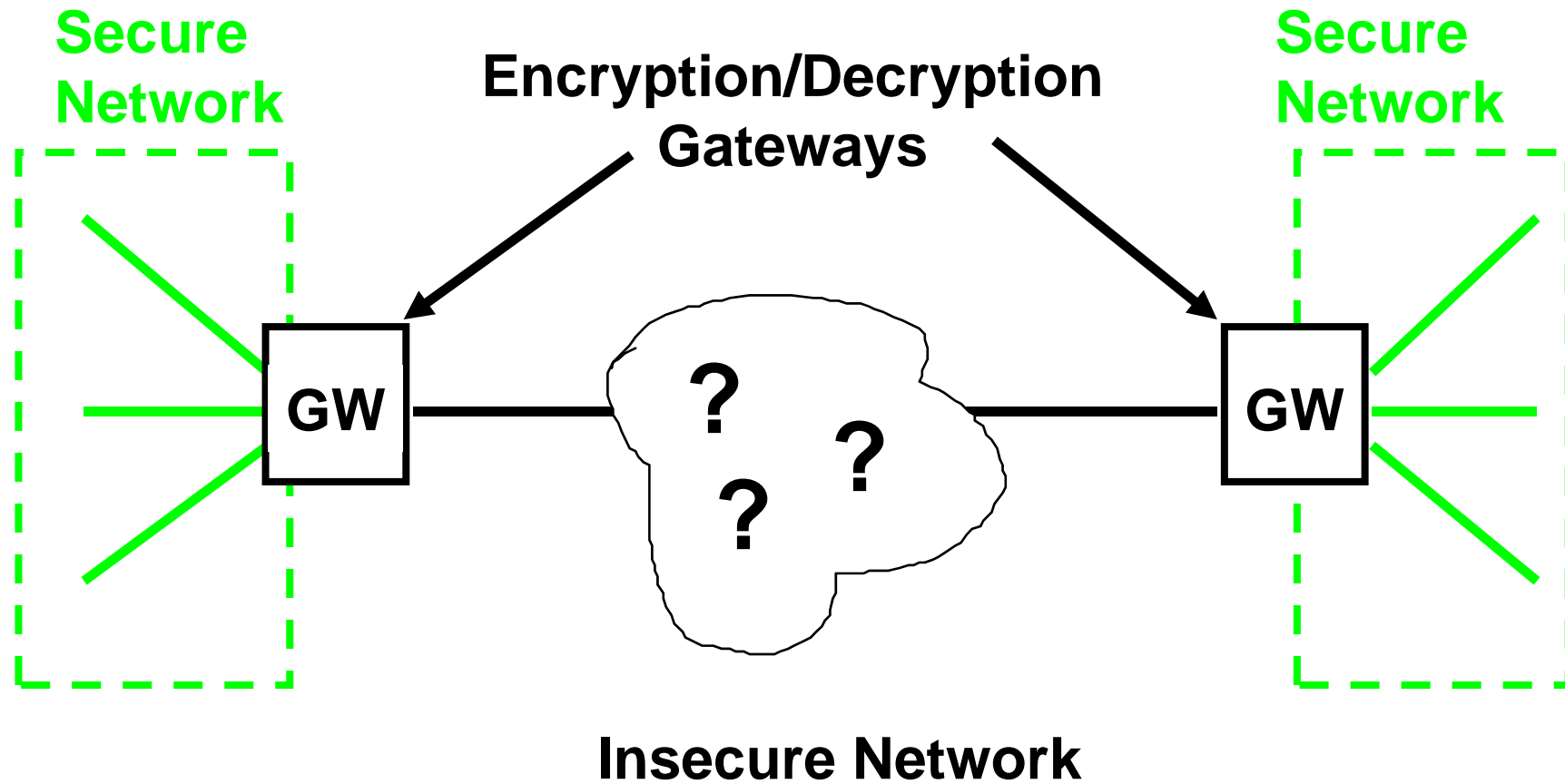
- ⌘ Provides service connectivity even though intermediate network does not support protocols.

# Translation



⌘ Translate from green protocol to brown protocol

# Encryption gateway



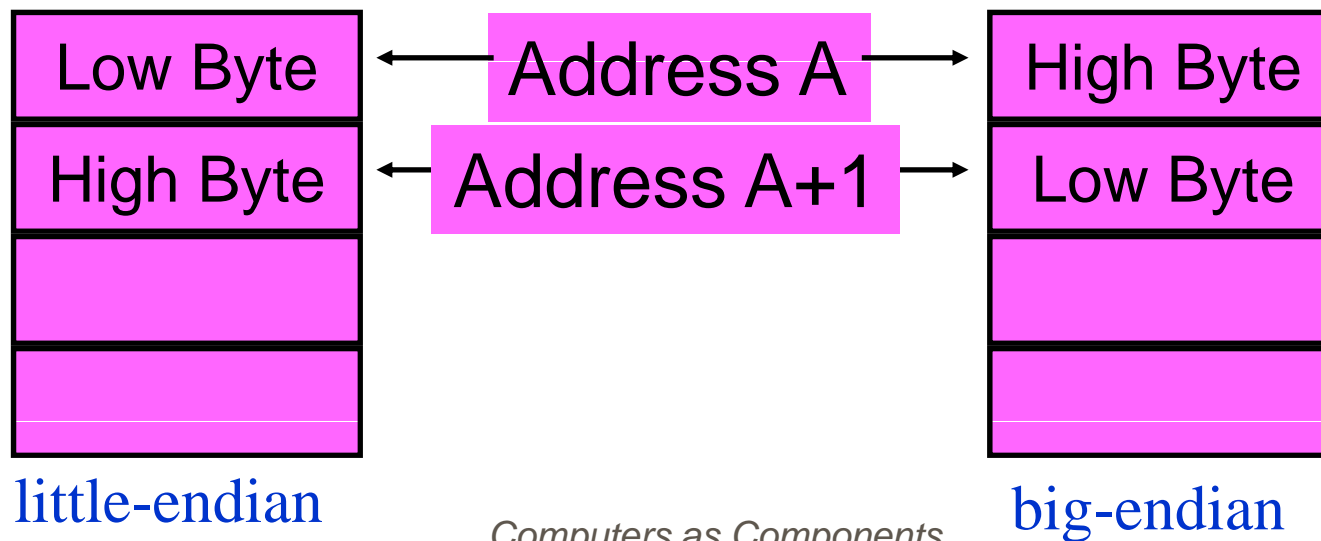
# Hardware vs. Software



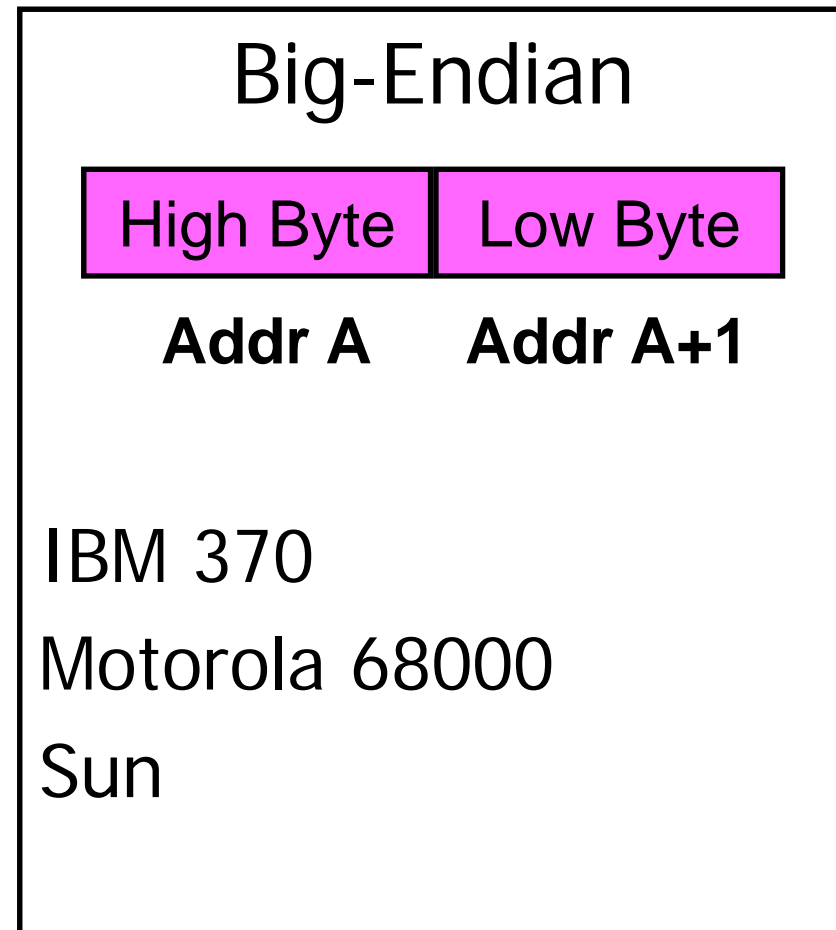
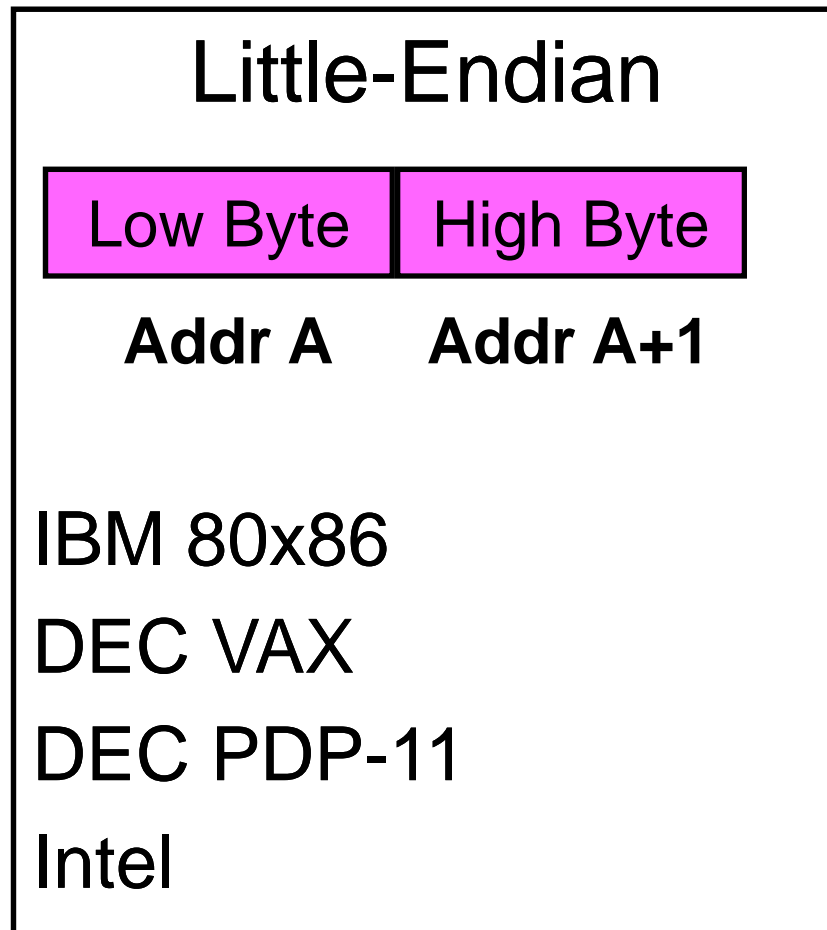
- ⌘ Repeaters are typically hardware devices.
- ⌘ Bridges can be implemented in hardware or software.
- ⌘ Routers & Gateways are typically implemented in software so that they can be extended to handle new protocols.
- ⌘ Many workstations can operate as routers or gateways.

# Byte Ordering

- ⌘ Different computer architectures use different byte ordering to represent multibyte values.
- ⌘ 16 bit integer:

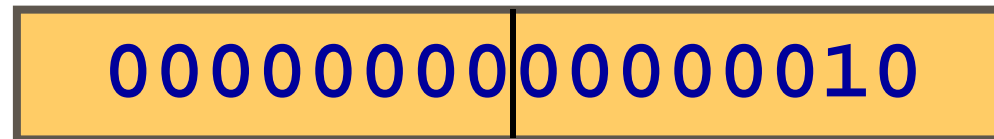


# Byte Ordering

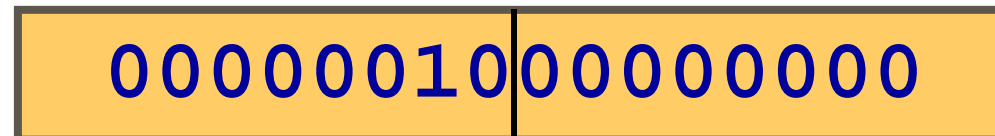


# Byte Order and Networking

⌘ Suppose a Big Endian machine sends a 16 bit integer with the value 2:



⌘ A Little Endian machine will think it got the number 512:





# Network Byte Order



- ⌘ Conversion of application-level data is left up to the presentation layer.
- ⌘ But , How do lower level layers communicate if they all represent values differently ? (data length fields in headers)
- ⌘ A fixed byte order is used (called *network byte order*) for all control data.

# Multiplexing



- ⌘ “.. to combine many into one”.
- ⌘ Many processes sharing a single network interface.
- ⌘ A single process could use multiple protocols.
- ⌘ More on this when we look at TCP/IP.

# Modes of Service



- ⌘ connection-oriented vs. connectionless
- ⌘ sequencing
- ⌘ error-control
- ⌘ flow-control
- ⌘ byte stream vs. message based
- ⌘ full-duplex vs. half-duplex.

# Connection-Oriented vs. Connectionless Service

- ⌘ A connection-oriented service includes the establishment of a logical connection between 2 processes.
  - ☑ establish logical connection
  - ☑ transfer data
  - ☑ terminate connection.
- ⌘ Connectionless services involve sending of independent messages.

# Sequencing



- ⌘ Sequencing provides support for an order to communications.
- ⌘ A service that includes sequencing requires that messages (or bytes) are received in the same order they are sent.

# Error Control



- ⌘ Some services require error detection (it is important to know when a transmission error has occurred).
- ⌘ Checksums provide a simple error detection mechanism.
- ⌘ Error control sometimes involves notification and retransmission.

# Flow Control



- ⌘ Flow control prevents the sending process from overwhelming the receiving process.
- ⌘ Flow control can be handled a variety of ways - this is one of the major research issues in the development of the next generation of networks (ATM).

# Byte Stream vs. Message



- ⌘ Byte stream implies an ordered sequence of bytes with no message boundaries.
- ⌘ Message oriented services provide communication service to chunks of data called datagrams.



# Full- vs. Half-Duplex



⌘ Full-Duplex services support the transfer of data in both directions.



⌘ Half-Duplex services support the transfer of data in a single direction.



# End-to-End vs. Hop-to-Hop



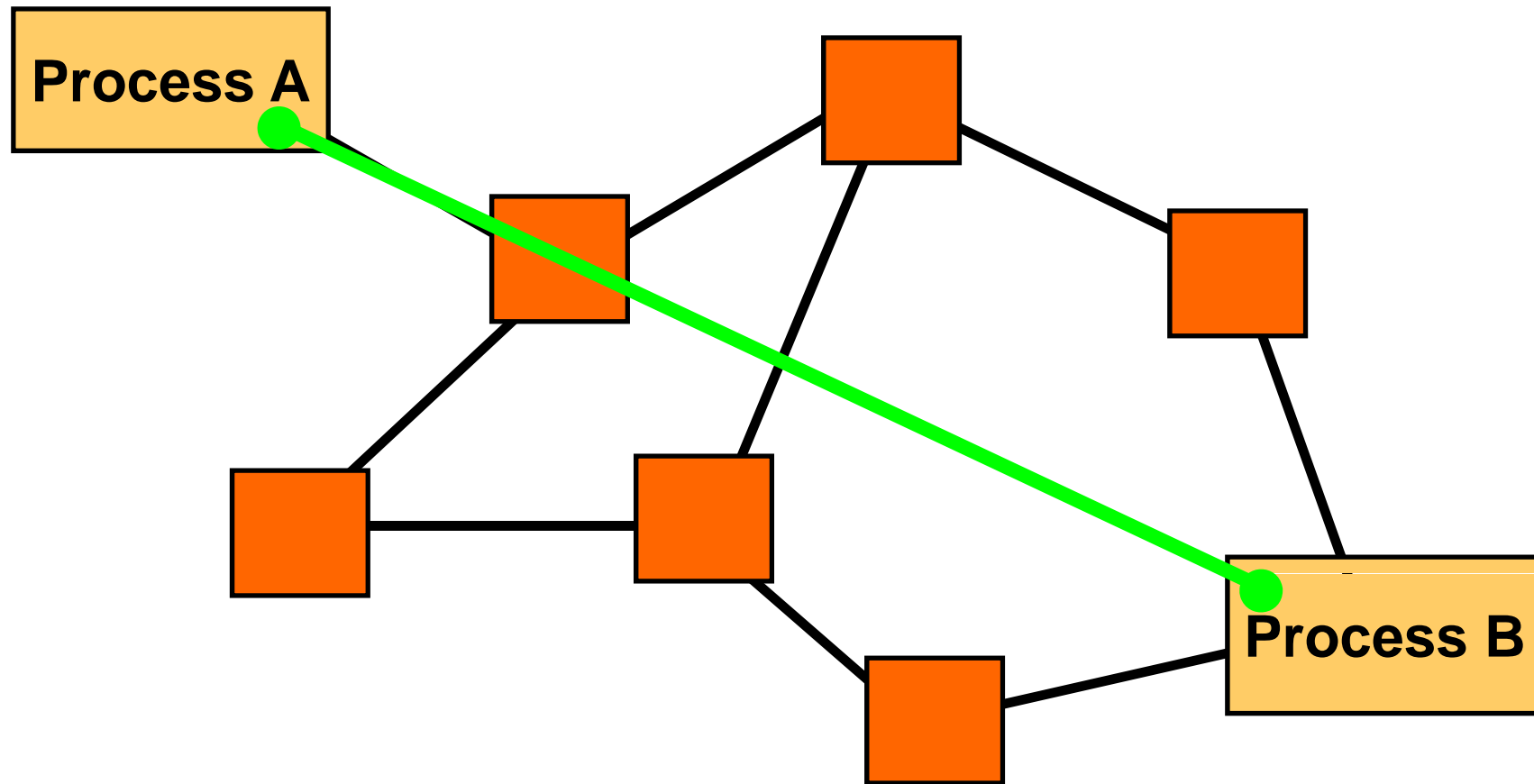
⌘ Many service modes/features such as flow control and error control can be done either:

between endpoints of the communication.

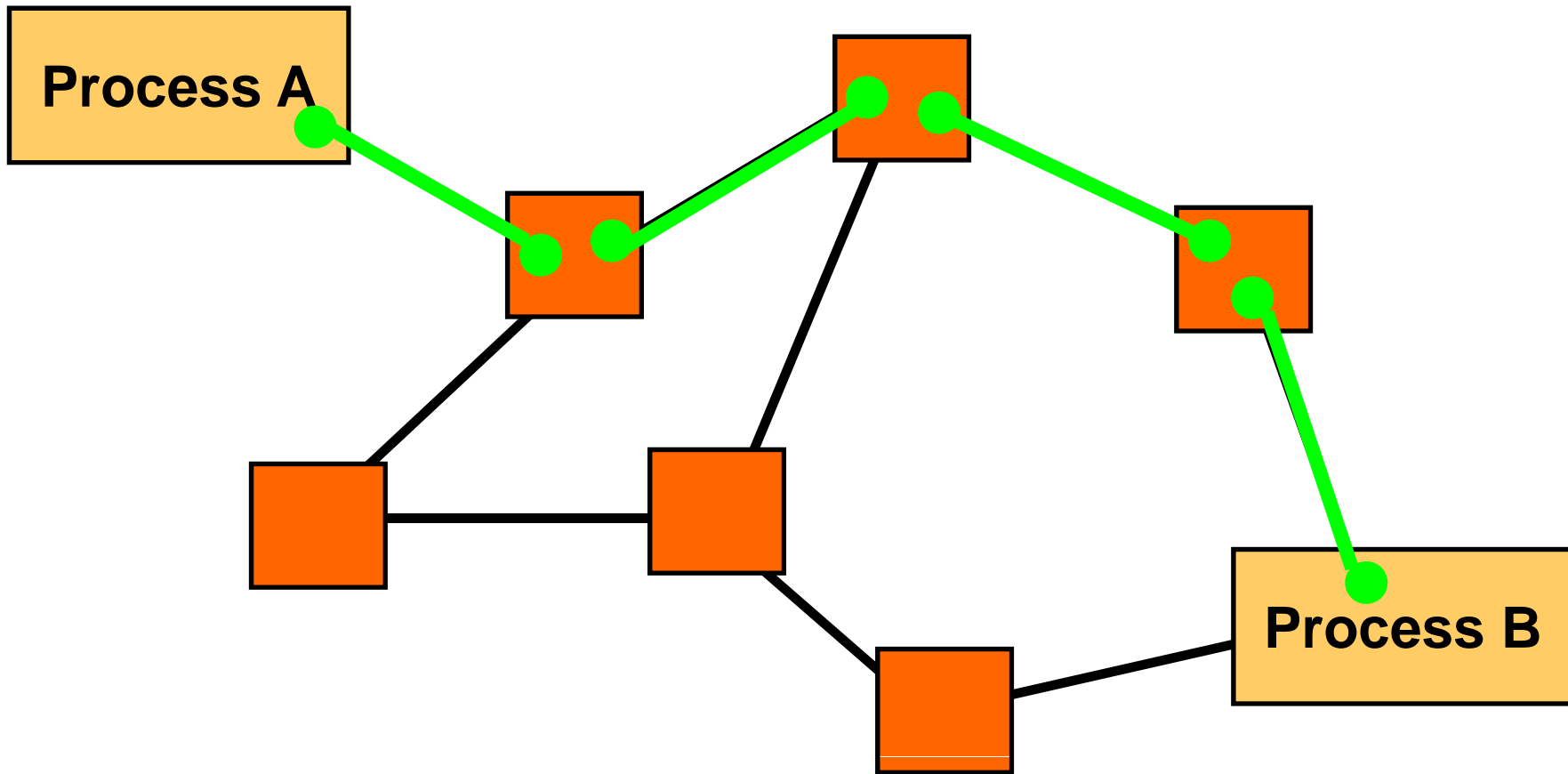
-or-

between every 2 nodes on the path between the endpoints.

# End-to-End



# Hop-by-Hop



# Buffering

- ⌘ Buffering can provide more efficient communications.
- ⌘ Buffering is most useful for byte stream services.



# Addresses



- ⌘ Each communication endpoint must have an address.
- ⌘ Consider 2 processes communicating over an internet:
  - ☑ the network must be specified
  - ☑ the host (end-system) must be specified
  - ☑ the process must be specified.

# Addresses at Layers



- ⌘ Physical Layer: no address necessary
- ⌘ Data Link Layer - address must be able to select any host on the network.
- ⌘ Network Layer - address must be able to provide information to enable routing.
- ⌘ Transport Layer - address must identify the destination process.

# Broadcasts



- ⌘ Many networks support the notion of sending a message from one host to all other hosts on the network.
- ⌘ A special address called the “broadcast address” is often used.
- ⌘ Some popular network services are based on broadcasting (YP/NIS, rcp, rusers)



# Hardware architectures



⌘ Many different types of networks:

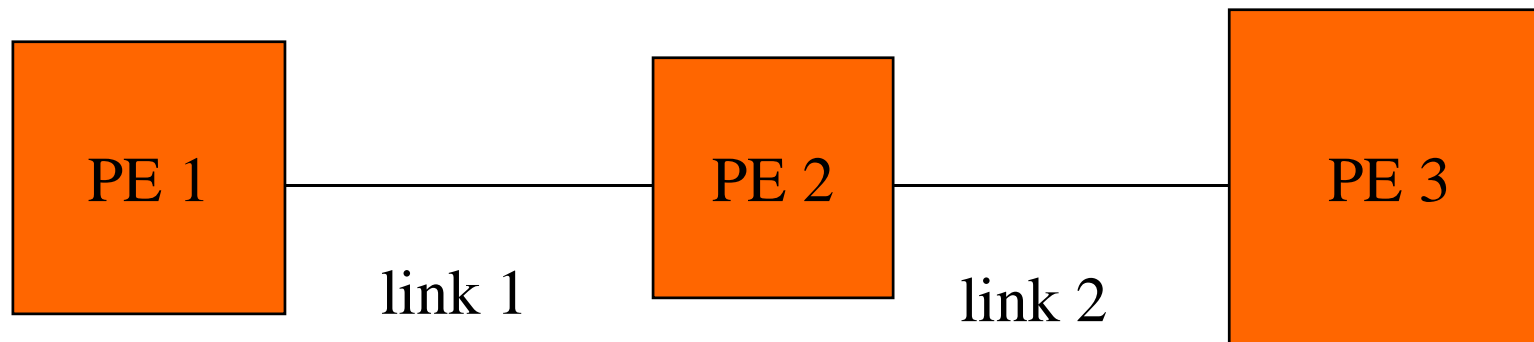
- ☑ topology;

- ☑ scheduling of communication;

- ☑ routing.

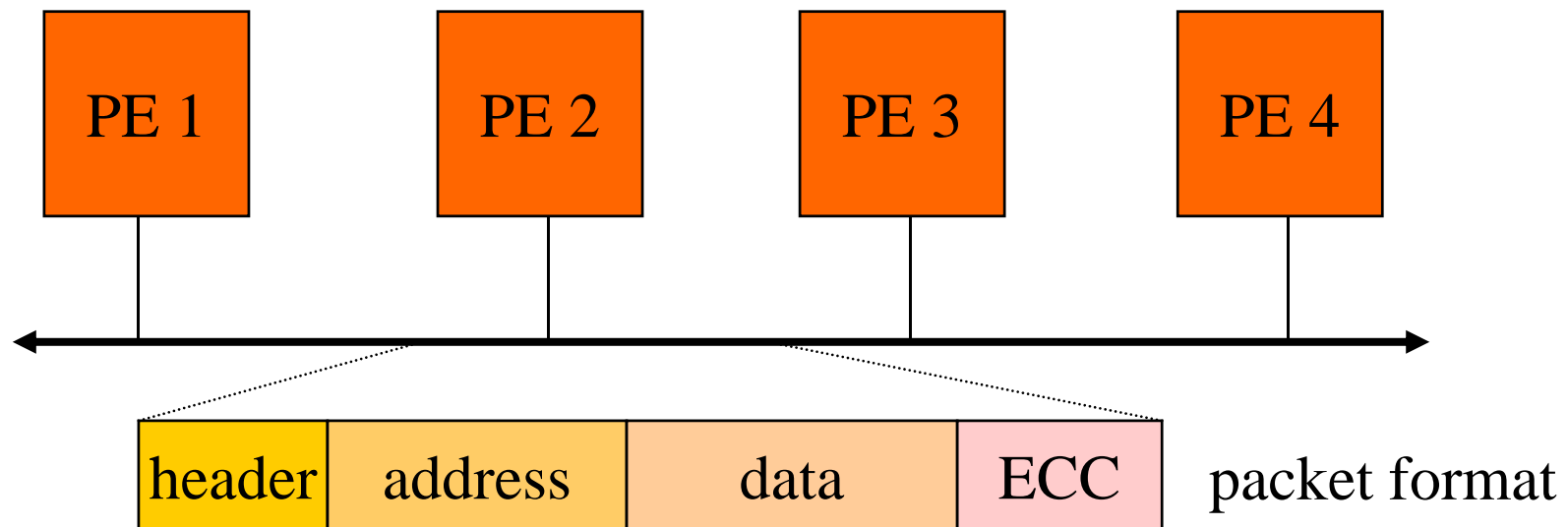
# Point-to-point networks

⌘ One source, one or more destinations, no data switching (serial port):



# Bus networks

⌘ Common physical connection:

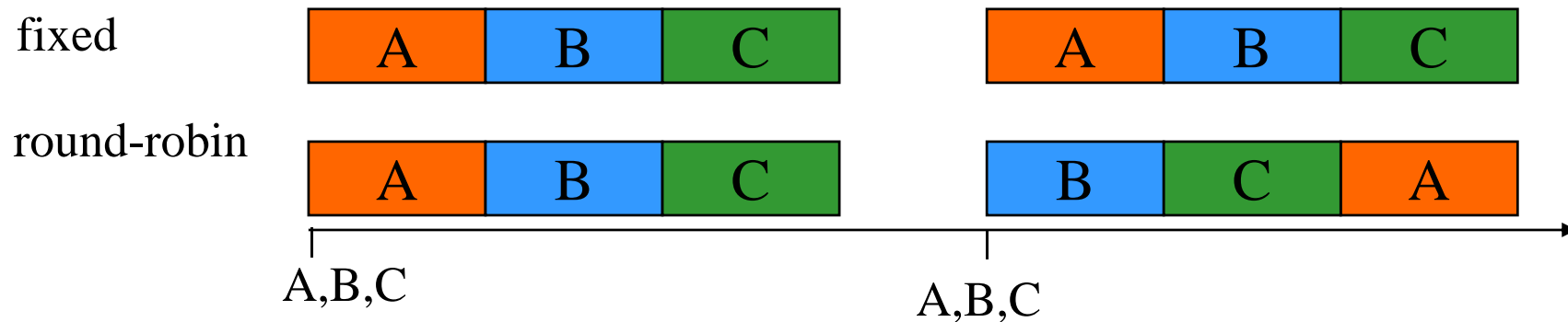


# Bus arbitration

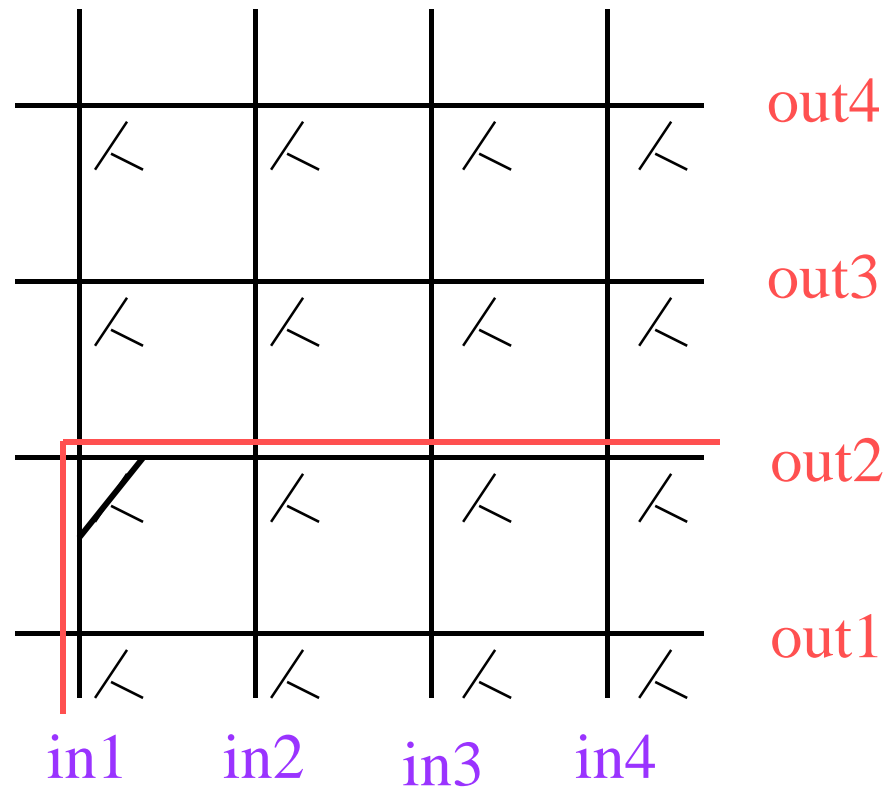
⌘ **Fixed**: Same order of resolution every time.

⌘ **Fair**: every PE has same access over long periods.

⏏ **round-robin**: rotate top priority among PEs.



# Crossbar



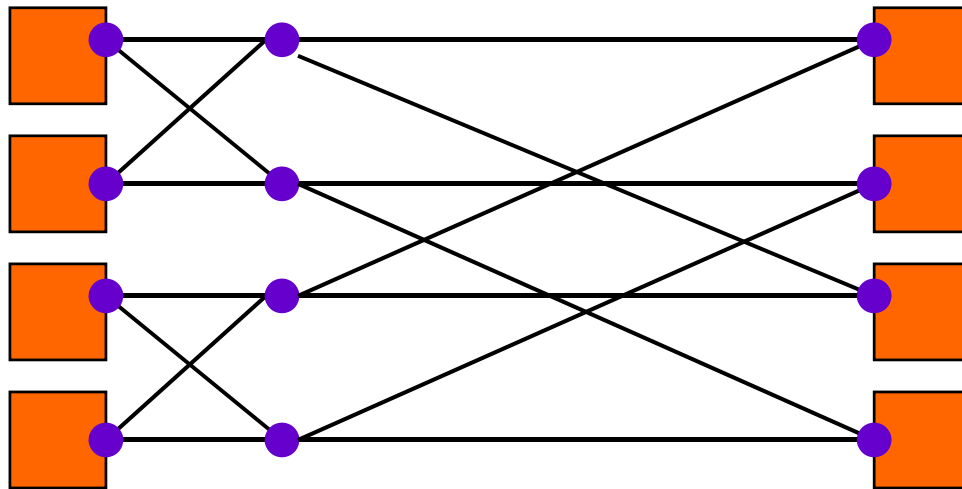
# Crossbar characteristics



- ⌘ Non-blocking (?)
- ⌘ Can handle multiple multi-cast combinations.
- ⌘ Size proportional to  $n^2$ .

# Multi-stage networks

- ⌘ Use several stages of switching elements.
- ⌘ Often blocking.
- ⌘ Often smaller than crossbar.



# Message-based programming



⌘ Transport layer provides message-based programming interface:

```
send_msg(adrs, data1);
```

⌘ Data must be broken into packets at source, reassembled at destination.

⌘ Message passing: blocking

⌘ Nonblocking network interface requires a queue of data to be sent

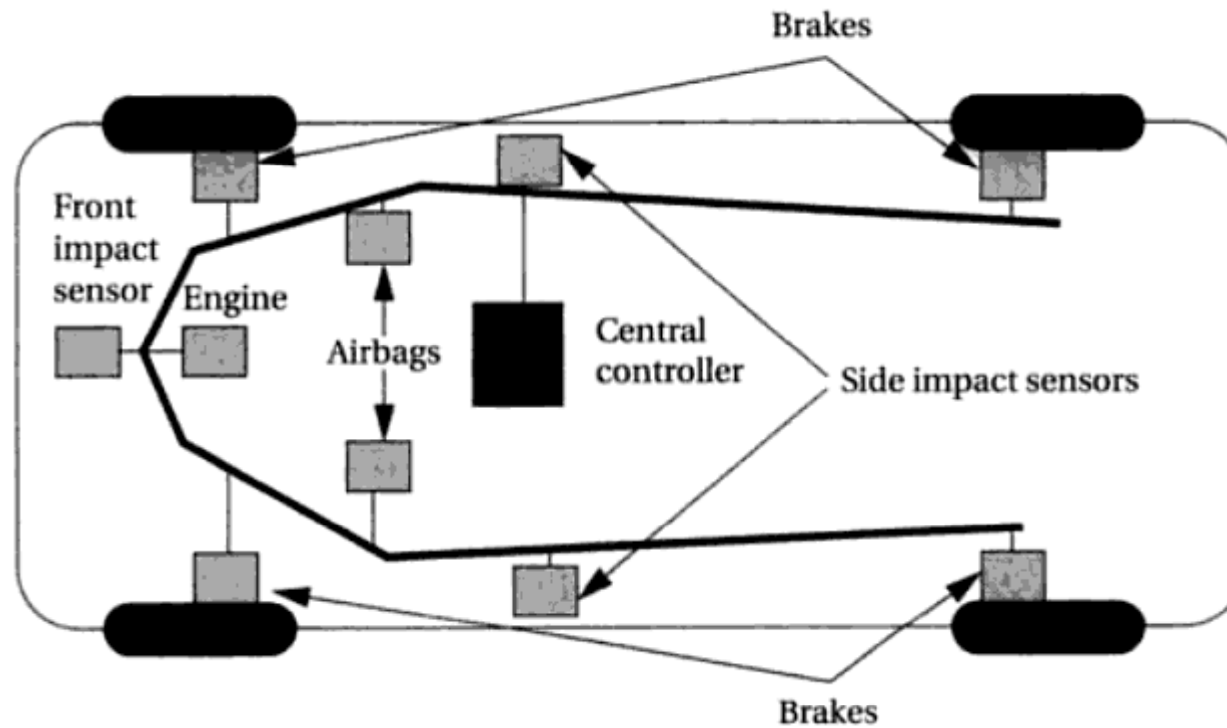


# Message-based programming



- ⌘ **Data-push programming**: make things happen in network based on data transfers.
- ⌘ Data-push programming makes sense for periodic data, which reduces data traffic on the network by automatically sending it when it is needed.

# Data-push network



The sensors generally need to be sampled periodically. In such a system, it makes sense for sensors to transmit their **data** automatically rather than waiting for the controller to request it.

# System buses



- ⌘ Multibus [Intel] & VME [Motorola]: multi-card computer system
- ⌘ ISA bus: I/O cards for PC-based systems
- ⌘ PCI bus: high-speed interfaces for PC-based applications; replace ISA

# Interconnection networks



- ⌘ For embedded systems
- ⌘ I<sup>2</sup>C bus: microcontroller-based systems
- ⌘ CAN bus: for automotive electronics
- ⌘ Echelon LON network: for home and industrial automation

# I<sup>2</sup>C bus



- ⌘ Designed for low-cost, medium data rate applications.
- ⌘ Command interface in a MPEG2 video chip
- ⌘ Characteristics:
  - ☑ serial;
  - ☑ multiple-master;
  - ☑ fixed-priority arbitration.
- ⌘ Several microcontrollers come with built-in I<sup>2</sup>C controllers.

# I<sup>2</sup>C bus



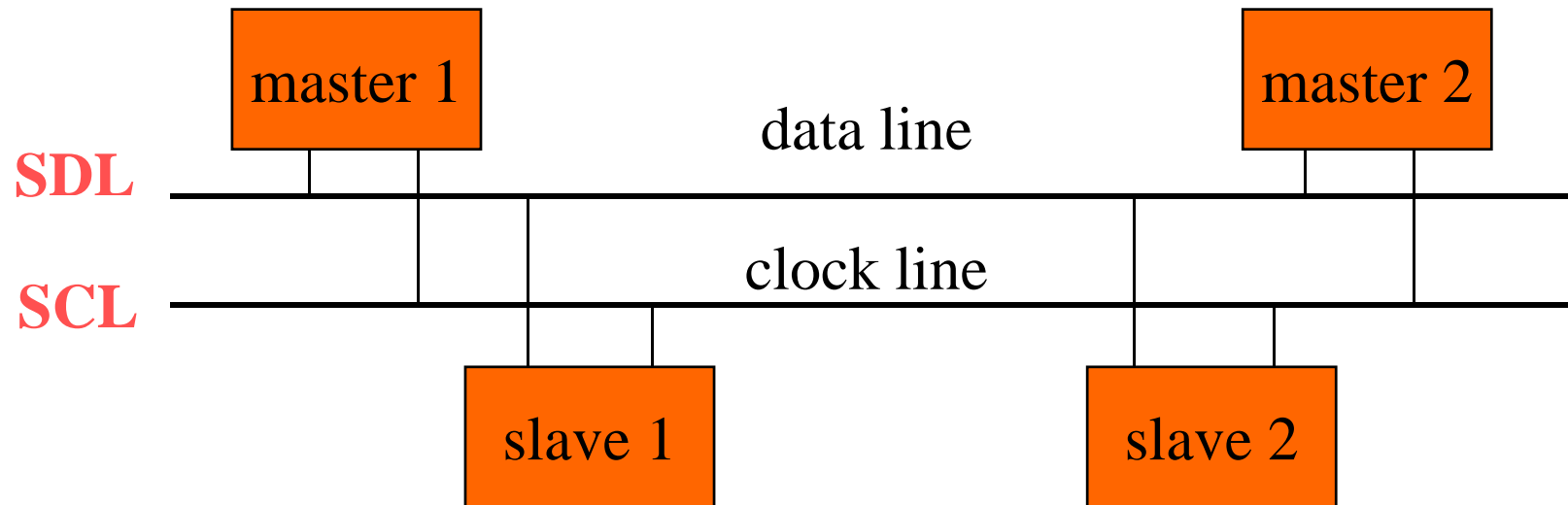
⌘ ~ up to 100 kbps (standard) ~up to 400 bps (extended)

⌘ Two lines

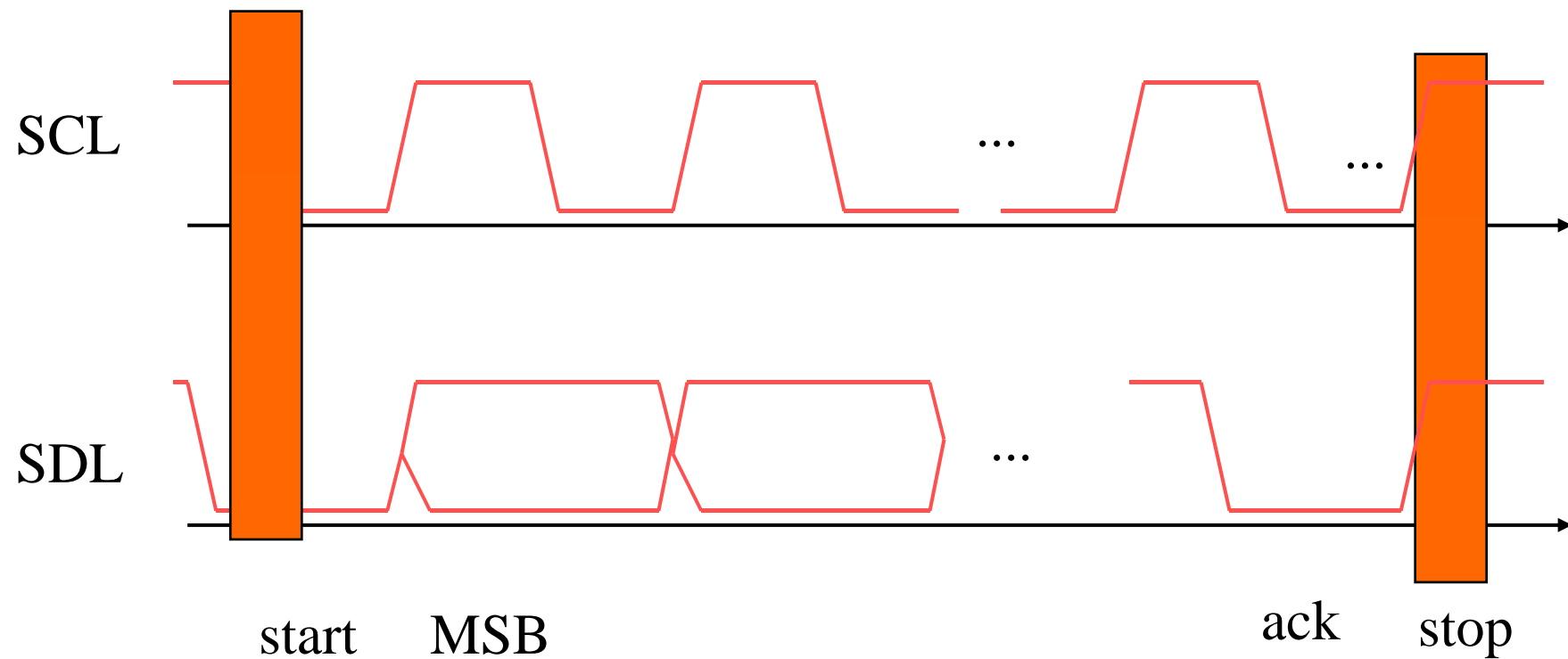
☒ SDL: serial data line

☒ SCL: serial clock line

# I<sup>2</sup>C physical layer



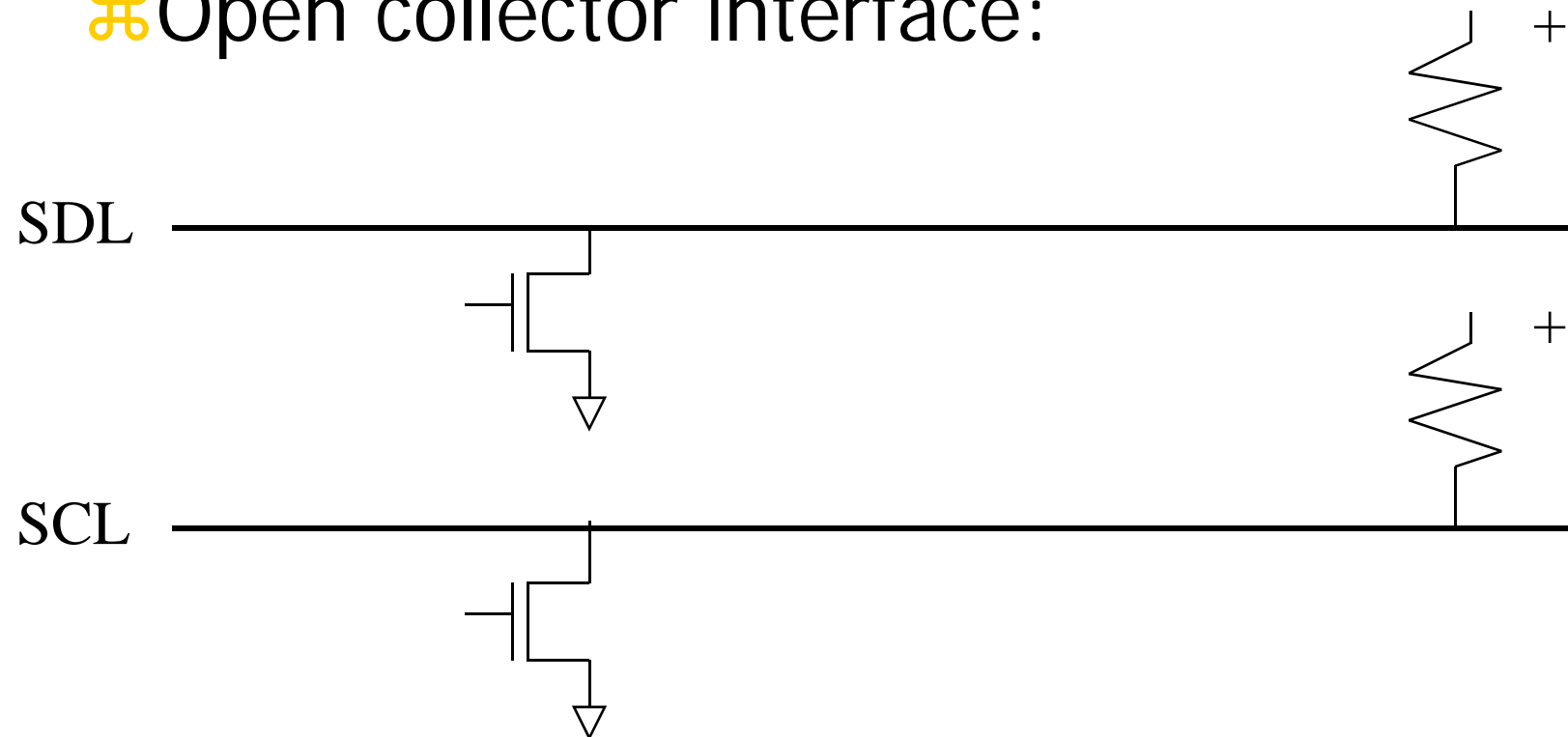
# I<sup>2</sup>C data format





# I<sup>2</sup>C electrical interface

⌘ Open collector interface:



# I<sup>2</sup>C signaling



- ⌘ Sender pulls down bus for 0.
- ⌘ Sender listens to bus---if it tried to send a 1 and heard a 0, someone else is simultaneously transmitting.
- ⌘ Transmissions occur in 8-bit bytes.

# I<sup>2</sup>C data link layer



- ⌘ Every device has an address (7 bits in standard, 10 bits in extension).
  - ☑ Bit 8 of address signals read or write.
- ⌘ General call address allows broadcast.

# I<sup>2</sup>C bus arbitration



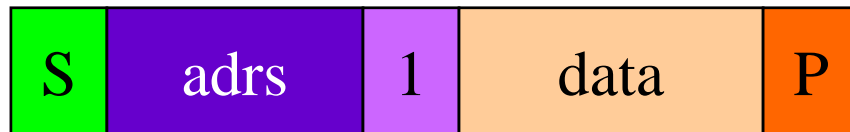
- ⌘ Sender listens while sending address.
- ⌘ When sender hears a conflict, if its address is higher, it stops signaling.
- ⌘ Low-priority senders relinquish control early enough in clock cycle to allow bit to be transmitted reliably.

# I<sup>2</sup>C transmissions

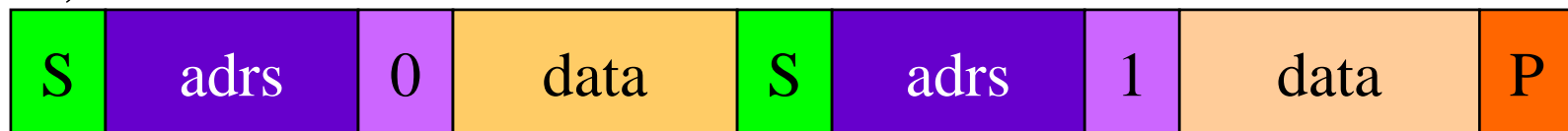
multi-byte write



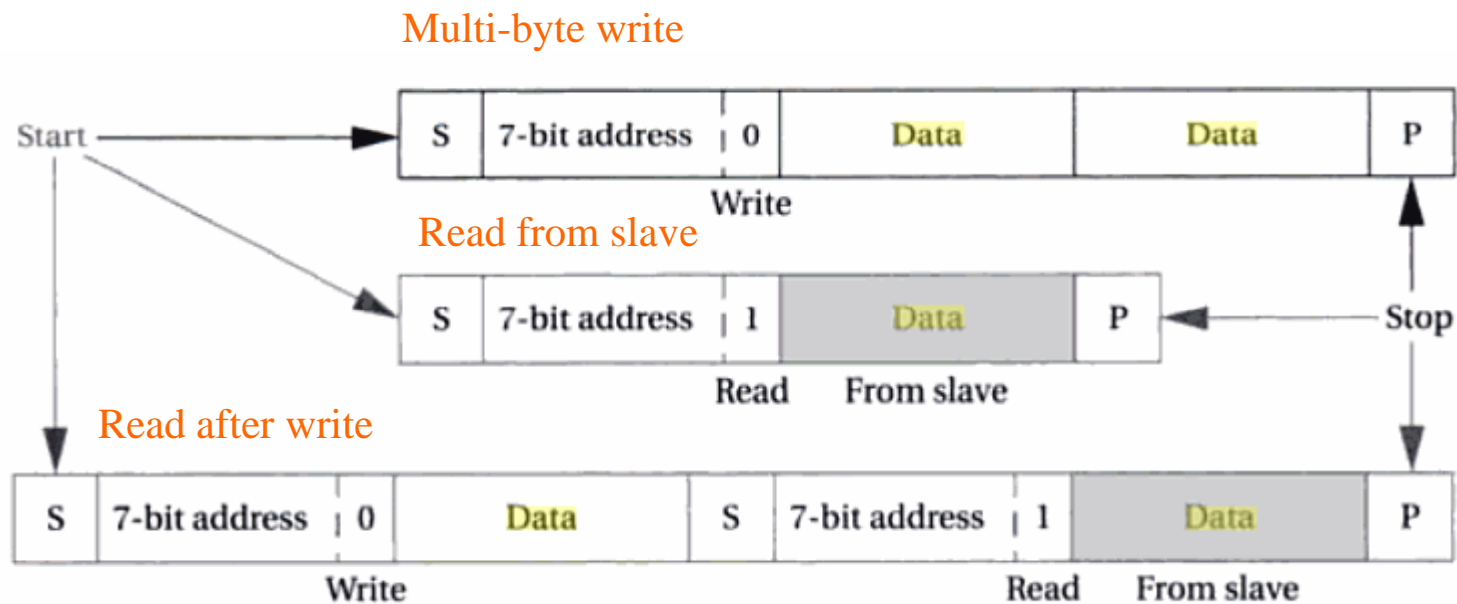
read from slave



write, then read



# Bus transactions on the I<sup>2</sup>C



# Ethernet



⌘ Dominant non-telephone LAN.

⌘ History

- ☑ Developed by Bob Metcalfe and others at Xerox PARC in mid-1970s
- ☑ Roots in Aloha packet-radio network
- ☑ Standardized by Xerox, DEC, and Intel in 1978
- ☑ LAN standards define MAC and physical layer connectivity
  - ☒ IEEE 802.3 (CSMA/CD - Ethernet) standard – originally 2Mbps
  - ☒ IEEE 802.3u standard for 100Mbps Ethernet
  - ☒ IEEE 802.3z standard for 1,000Mbps Ethernet

# Overview of Ethernet

⌘ **Ethernet** is the most popular network architecture

- ☑ Advantages: easy to install, scalable, broad media support, and low cost
- ☑ Supported transmission speeds: 10 Mbps to 10 Gbps
- ☑ Uses the NIC's MAC address to address frames
- ☑ Ethernet variations are compatible with one another
  - ☒ Basic operation and frame formatting is the same
  - ☒ Cabling, speed of transmission, and method by which bits are encoded on the medium differ





# Ethernet Operation



- ⌘ Ethernet is a best-effort delivery system
  - ☑ It works at the Data Link layer of the OSI model
    - ☑ Relies on the upper-layer protocols to ensure reliable delivery of data
- ⌘ Understanding the following concepts is important:
  - ☑ How Ethernet accesses network media
  - ☑ Collisions and collision domains
  - ☑ How Ethernet handles errors
  - ☑ Half-duplex and full-duplex communications

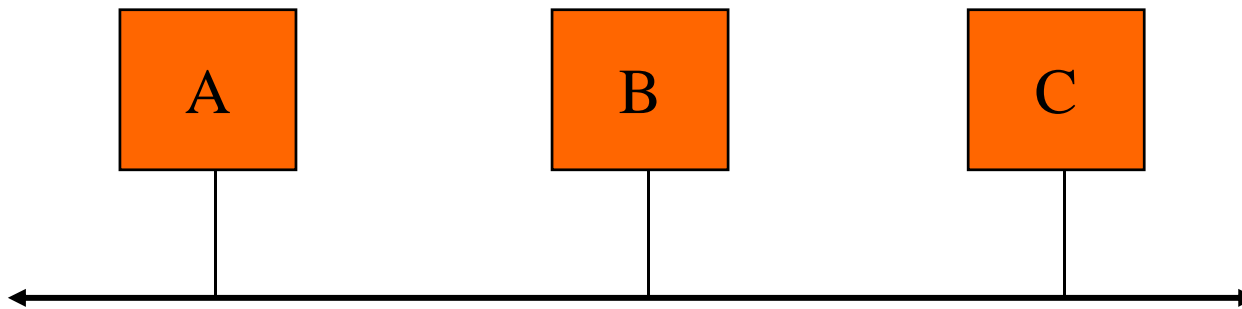
# Accessing Network Media



- ⌘ Ethernet uses CSMA/CD in a shared-media environment (a logical bus)
  - ☑ Ethernet device listens for a signal or carrier (carrier sense) on the medium first
  - ☑ If no signal is present, no other device is using the medium, so a frame can be sent
  - ☑ Ethernet devices have circuitry that detects collisions and automatically resends the frame that was involved in the collision

# Ethernet topology

⌘ Bus-based system, several possible physical layers:



# Collision Domains

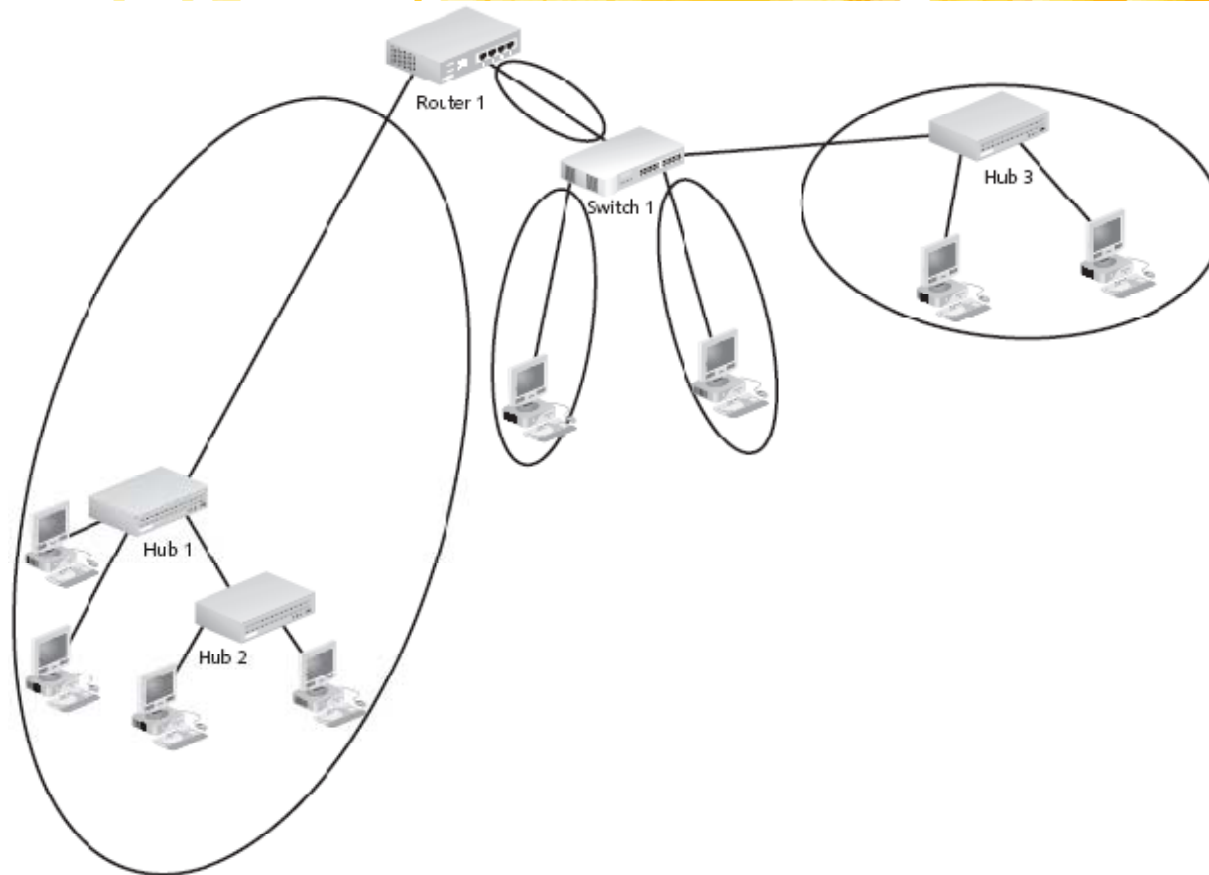


Figure 7-4 A network diagram showing five collision domains (circled)

# Ethernet Error Handling



- ⌘ Collisions are the only type of error for which Ethernet automatically attempts to resend the data
- ⌘ Errors can occur when data is altered in medium
  - ☑ Usually caused by noise or faulty media connections
  - ☑ When the destination computer receives a frame, the CRC is recalculated and compared against the CRC value in the FCS
  - ☑ If values match, the data is assumed to be okay
  - ☑ If values don't match, the data was corrupted
    - ☒ Destination computer discards the frame
    - ☒ No notice is given to the sender

# Half-Duplex Versus Full-Duplex Communications



- ⌘ When **half-duplex communication** is used with Ethernet, CSMA/CD must also be used
- ⌘ When using a switched topology, a computer can send and receive data simultaneously (**full-duplex communication**)
  - ☑ The collision detection circuitry is turned off because collisions aren't possible
  - ☑ Results in a considerable performance advantage

# Ethernet Standards



- ⌘ Each Ethernet variation is associated with an IEEE standard
- ⌘ The following sections discuss many of the standards, some of which are obsolete or had limited use
- ⌘ Keep in mind that Ethernet over UTP cabling has been the dominant technology since the early 1990s, and will likely to continue to be for the foreseeable future

# Physical Layer Configurations for 802.3

- ⌘ Physical layer configurations are specified in three parts
- ⌘ Data rate (10, 100, 1,000)
  - ⌘ 10, 100, 1,000Mbps
- ⌘ Signaling method (base, broad)
  - ⌘ Baseband
    - ⌘ Digital signaling
  - ⌘ Broadband (Obsolete?)
    - ⌘ Analog signaling
- ⌘ Cabling (2, 5, T, F, S, L)
  - ⌘ 5 - Thick coax (original Ethernet cabling)
  - ⌘ F – Optical fiber
  - ⌘ S – Short wave laser over multimode fiber
  - ⌘ L – Long wave laser over single mode fiber



# 100 Mbps IEEE Standards

⌘ The most widely accepted Ethernet standard today is 100BaseT, which is also called **fast Ethernet**

⌘ The current IEEE standard for 100BaseT is 802.3u

⌘ Subcategories:

- *100BaseTX*: Two-pair Category 5 or higher UTP
- *100BaseT4*: Four-pair Category 3 or higher UTP
- *100BaseFX*: Two-strand fiber-optic cable

⌘ Because of its widespread use, the cable and equipment in fast Ethernet are inexpensive

⌘ Architecture of choice for all but heavily used servers and multimedia applications

# 100BaseT Design Considerations

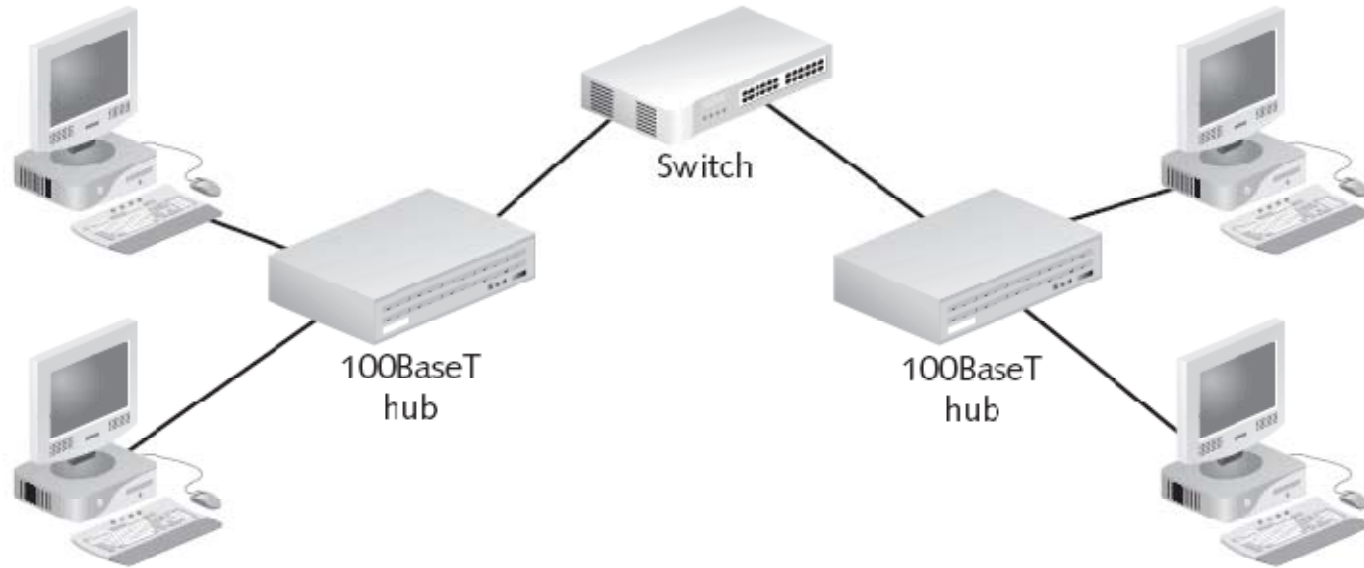


Figure 7-5 Using a switch to interconnect 100BaseT hubs

# 100BaseT Design Considerations (continued)

Table 7-6 100BaseT Ethernet summary

Category	Summary
IEEE specification	802.3u
Advantages	Fast; easy to configure and troubleshoot
Disadvantages	High cost; limited distance
Topology	Star
Cable type	Category 3 or higher UTP for 100BaseT4 Category 5 or higher UTP for 100BaseTX Fiber-optic for 100BaseFX
Channel access method	CSMA/CD
Transceiver location	On NIC
Maximum cable segment length	100 m (328 ft.) for 100BaseT4 and 100BaseTX; 2000 m (6561 ft.) for 100BaseFX
Maximum number of segments	1024
Maximum devices per segment	1
Maximum devices per network	1024
Transmission speed	100 Mbps

# 10 Mbps IEEE Standards



- ⌘ Four major implementations of 10 Mbps Ethernet
  - ☒ *10Base5*: Ethernet using thicknet coaxial cable
  - ☒ *10Base2*: Ethernet using thinnet coaxial cable
  - ☒ *10BaseT*: Ethernet over UTP cable
  - ☒ *10BaseF*: Ethernet over fiber-optic cable
- ⌘ Of these 10 Mbps standards, only 10BaseT and 10BaseF are seen today
- ⌘ 10Base2 and 10Base5 are essentially obsolete

# 10BaseT

Table 7-7 10BaseT Ethernet summary

Category	Summary
IEEE specification	802.3
Advantages	Very inexpensive; easy to install and troubleshoot
Disadvantages	Small maximum cable segment length
Topology	Star
Cable type	Category 3 or higher UTP, but typically Cat 5e or 6 today
Channel access method	CSMA/CD
Transceiver location	On NIC
Maximum cable segment length	100 m (328 ft.)
Minimum distance between devices	N/A
Maximum number of segments	1024
Maximum devices per segment	2
Maximum devices per network	1024
Transmission speed	10 Mbps

# Gigabit Ethernet

## ⌘ Gigabit Ethernet implementations

- ☒ 802.3z-1998 covers 1000BaseX specifications, including the L (long wavelength laser/fiber-optic), S (short wavelength laser/fiber-optic), and C (copper jumper cables)
- ☒ 802.3ab-1999 covers 1000BaseT specifications, which require four pairs of 100 ohm Category 5 or higher cable

# 1000BaseT

Table 7-9 1000BaseT Ethernet summary

Category	Summary
IEEE specification	802.3ab
Advantages	Fast; supports full-duplex communications
Disadvantages	High cost; short-haul cable segments only
Topology	Star
Cable type	Four-pair, balanced Category 5 cable; 100-ohm impedance
Channel access method	CSMA/CD or switching
Transceiver location	On NIC
Maximum cable segment length	Half-duplex: 100 m (328 ft.) Full-duplex: 100 m (328 ft.)
Maximum number of segments	1024
Maximum devices per segment	2
Maximum devices per network	1024
Transmission speed	1000 Mbps; 2000 Mbps in full-duplex mode

# 10 Gigabit Ethernet



- ⌘ Defined to run only on fiber-optic cabling, both SMF and MMF, on a maximum distance of 40 km
  - ☑ Provides bandwidth that can transform how WAN speeds are thought of
- ⌘ Runs in full-duplex mode only
  - ☑ CSMA/CD is not necessary
- ⌘ Primary use: as network backbone
  - ☑ It also has its place in storage area networks (SANs)
  - ☑ Will be the interface for enterprise-level servers



# What's Next for Ethernet?



- ⌘ Implementations of 40 Gbps Ethernet are underway
- ⌘ Ethernet could increase tenfold every 4-6 years
  - ☒ 100 Gbps Ethernet available by 2006 to 2008, terabit Ethernet by 2011, and 10 terabit Ethernet by 2015
- ⌘ In October 2005, Lucent Technologies demonstrated for the first time the transmission of Ethernet over fiber-optic cable at 100 Gbps
  - ☒ It will be able to transfer data across the city faster than today's CPUs can transfer data to memory
  - ☒ This level of speed has major implications for the entertainment industry and many other areas

# Ethernet Frame Types

- ⌘ Ethernet supports four non-compatible **frame types**
  - ☒ Ethernet 802.3: used by IPX/SPX on Novell NetWare 2.x and 3.x networks
  - ☒ Ethernet 802.2: used by IPX/SPX on Novell NetWare 3.12 and 4.x networks
    - ☒ Supported by default in Microsoft NWLink
  - ☒ Ethernet SNAP: used in EtherTalk and mainframes
  - ☒ Ethernet II is used by TCP/IP
- ⌘ All Ethernet frame types support a packet size between 64 and 1518 bytes, and can be used by all network architectures mentioned previously

# Ethernet 802.3

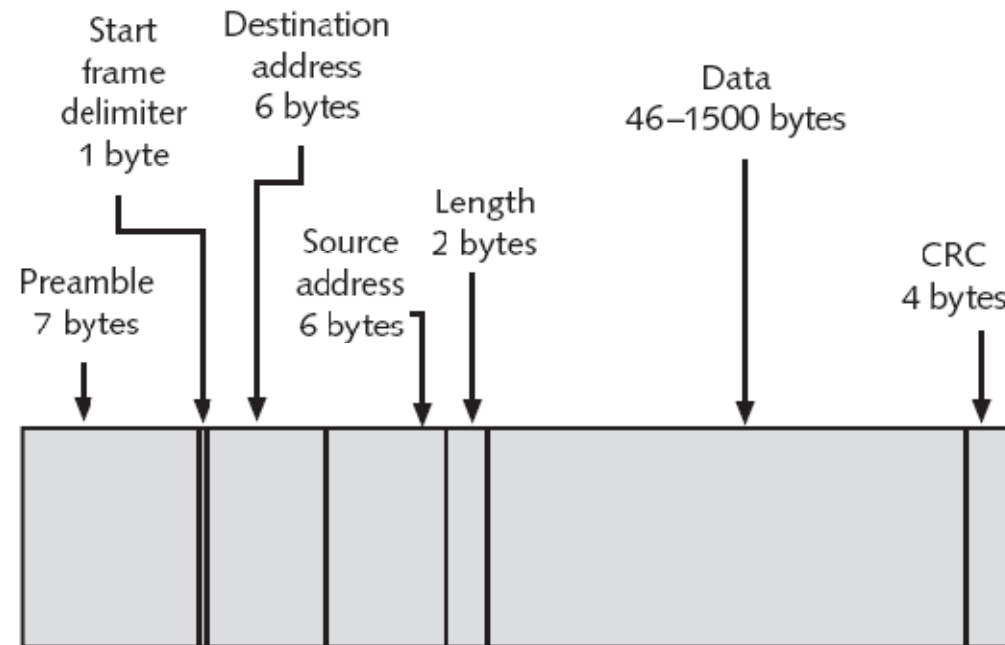


Figure 7-6 An Ethernet 802.3 frame

# Ethernet II

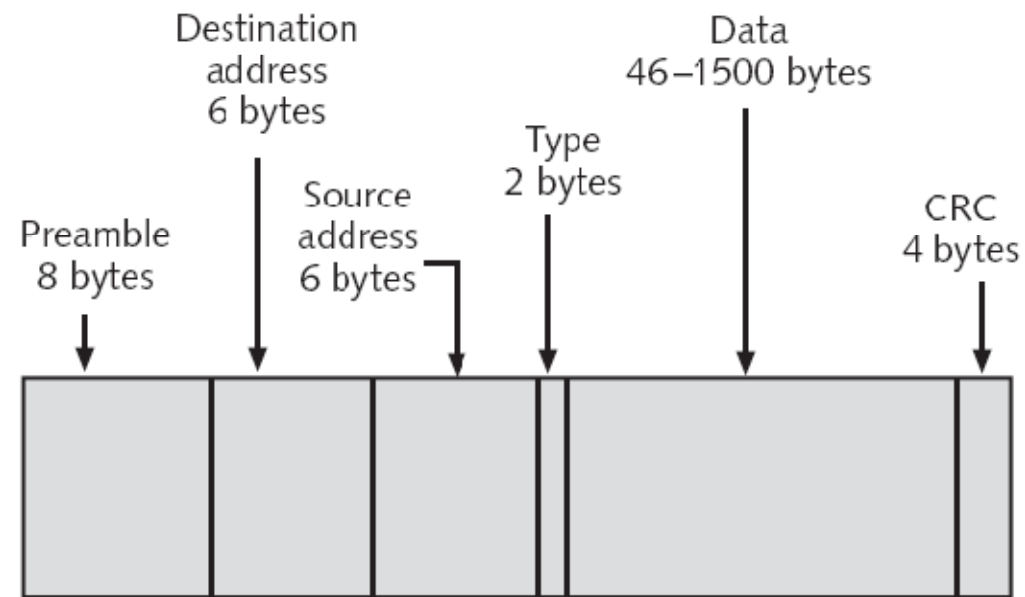


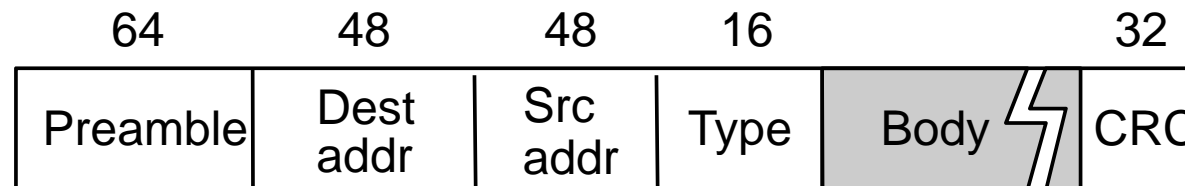
Figure 7-7 An Ethernet II frame uses a type field

# Wireless Ethernet: IEEE 802.11b, a, and g

- ⌘ AP serves as the center of a star topology network
- ⌘ Stations can't send and receive at the same time
  - ⊞ CSMA/CA is used instead of CSMA/CD
- ⌘ 802.11b/a/g use handshaking before transmission
  - ⊞ Station sends AP an RTS and it responds with CTS
- ⌘ Standards define a maximum transmission rate, but speeds might be dropped to increase reliability
- ⌘ No fixed segment length
  - ⊞ Maximum of 300 feet without obstructions
    - ⊞ Can be extended with large, high-quality antennas

# Ethernet Overview

- ⌘ Ethernet by definition is a broadcast protocol
  - ☑ Any signal can be received by all hosts
  - ☑ Switching enables individual hosts to communicate
- ⌘ Network layer packets are transmitted over an Ethernet by encapsulating
- ⌘ Frame Format



# Ethernet packet format



# Ethernet Frames



- ⌘ Preamble is a sequence of 7 bytes, each set to “10101010”
  - ⊞ Used to synchronize receiver before actual data is sent
- ⌘ Addresses
  - ⊞ unique, 48-bit unicast address assigned to each adapter
    - ⊞ example: **8:0:e4:b1:2**
    - ⊞ Each manufacturer gets their own address range
  - ⊞ broadcast: all **1s**
  - ⊞ multicast: first bit is **1**
- ⌘ Type field is a demultiplexing key used to determine which higher level protocol the frame should be delivered to
- ⌘ Body can contain up to 1500 bytes of data



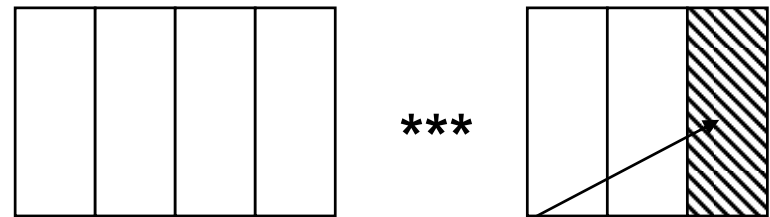
# Concept: Redundant Check

- ⌘ Send a message  $M$  and a “check” word  $C$
- ⌘ Simple function on  $\langle M, C \rangle$  to determine if both received correctly (with high probability)
- ⌘ Example: XOR all the bytes in  $M$  and append the “checksum” byte,  $C$ , at the end

☑ Receiver XORs  $\langle M, C \rangle$

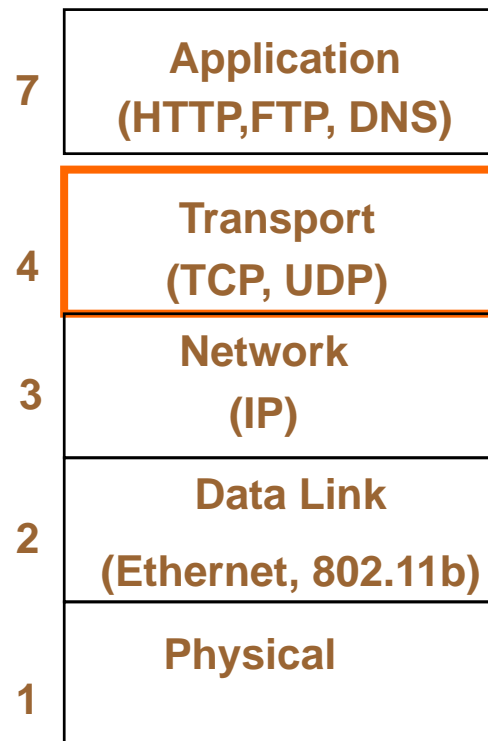
☑ What should result be?

☑ What errors are caught?

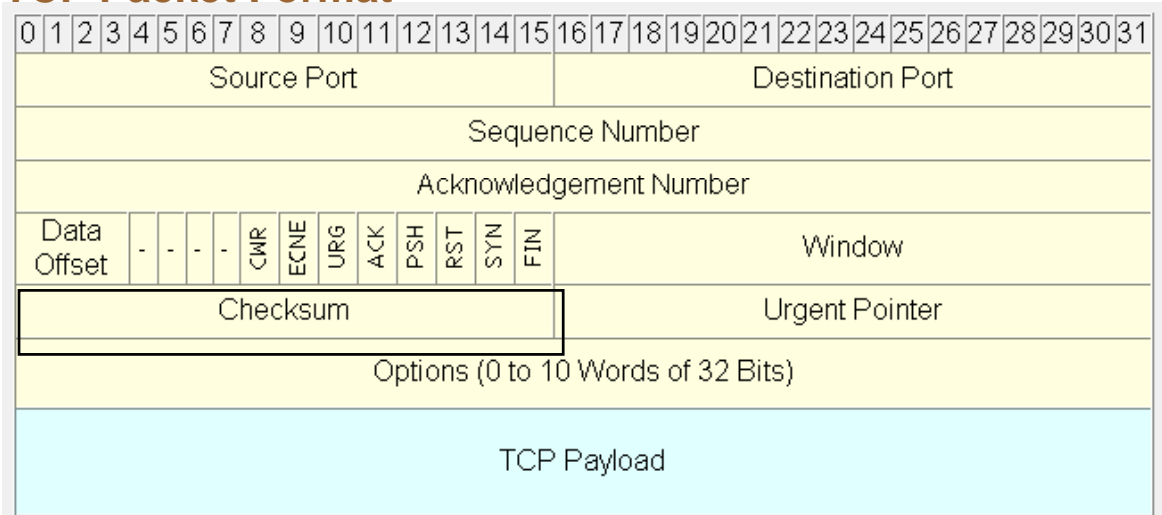


bit  $i$  is XOR of  $i$ th bit of each byte

# Example: TCP Checksum



## TCP Packet Format



- TCP Checksum a 16-bit **checksum**, consisting of the **one's complement** of the **one's complement sum** of the contents of the TCP segment header and data, is computed by a sender, and included in a segment transmission. (note end-around carry)
- Summing all the words, including the checksum word, should yield zero


# What is this ‘packet checksum’?



- ⌘ According to the Intel documentation, the packet checksum is “the unadjusted 16-bit ones complement of the packet”
- ⌘ Now what exactly does that mean?
- ⌘ And why did Intel’s hardware designers believe that device-driver software would need this value to be available for every ethernet packet that the NIC receives?

# 'end-around-carry'

- ⌘ When you want to add two binary numbers using the one's complement system, you can do it by first performing ordinary binary addition, and then adding in the carry-bit:

10101010	(-85)
+ 11110000	(-15)
-----	
1 10011010	(9-bits in the normal total)
+ 	(apply 'end-around carry')
-----	
10011011	(-100: 8-bits in the ones-complement total)

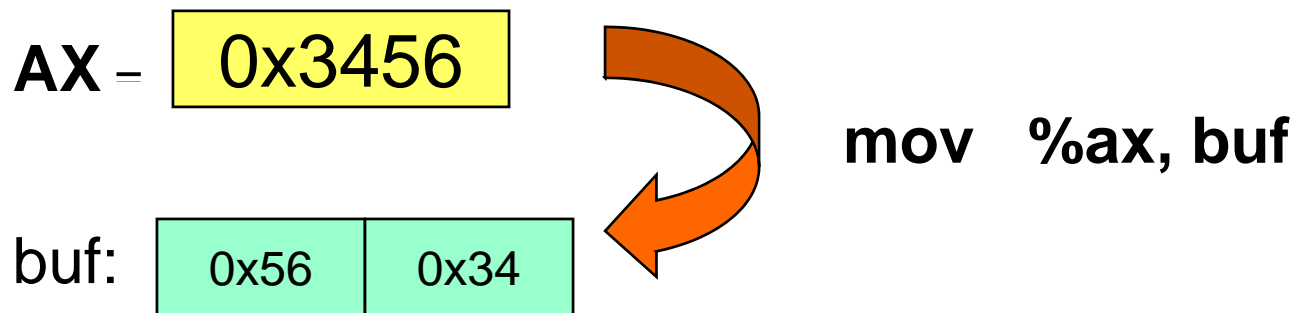
# NIC uses 'one's complement'



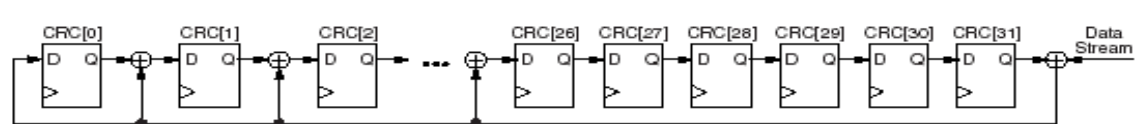
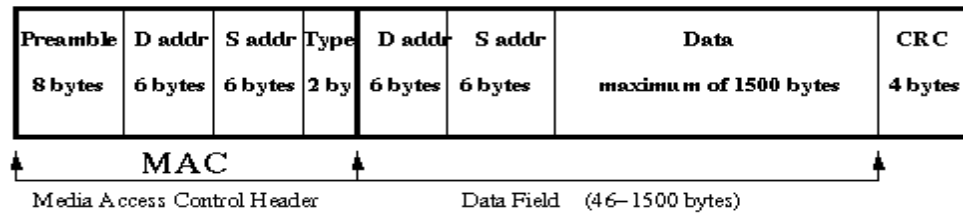
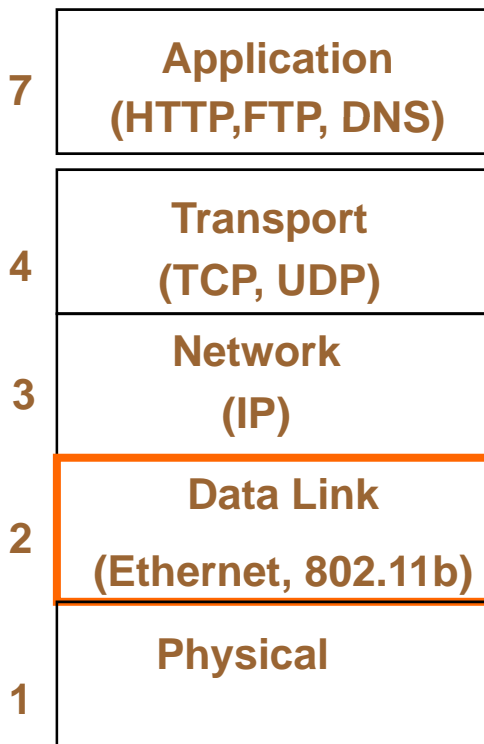
- ⌘ For network programming nowadays it is common practice for 'one's complement' to be used when computing checksums
- ⌘ It is also common practice for multi-byte integers to be 'sent out over the wire' in so called 'big-endian' byte-order (i.e., the most significant byte goes out ahead of the bytes having lesser significance)

# Intel uses 'little endian'

- ⌘ Whenever our x86 processor 'stores' a multi-byte value into a series of cells in memory, it puts the least significant byte first (i.e., at the earliest memory address), followed by the remaining bytes having greater significance)



# Example: Ethernet CRC-32



# CRC concept

- ⌘ A msg polynomial  $M(x)$  of degree  $m$
- ⌘ A generator poly  $G(x)$  of degree  $m$
- ⌘ Let  $r(x) = \text{remainder of } M(x) x^n / G(x)$ 
  - ⊞  $M(x) x^n = G(x)p(x) + r(x)$
  - ⊞  $r(x)$  is of degree  $n$
- ⌘ What is  $(M(x) x^n - r(x)) / G(x)$  ?
  
- ⌘ Send  $M(x) x^n - r(x)$ 
  - ⊞  $m+n$  degree polynomial
  - ⊞ Divide by  $G(x)$  to check
  - ⊞  $M(x)$  is just the  $m$  most significant coefficients,  $r(x)$  the lower  $m$
- ⌘  $n$ -bit Message is viewed as coefficients of  $n$ -degree polynomial over binary numbers

**n bits of zero at the end**



# CSMA/CD



## ⌘ CSMA/CD: Ethernet's Media Access Control (MAC) policy

- ⊞ CS = carrier sense

- ⊞ Send only if medium is idle

- ⊞ MA = multiple access

- ⊞ CD = collision detection

- ⊞ Stop sending immediately if collision is detected

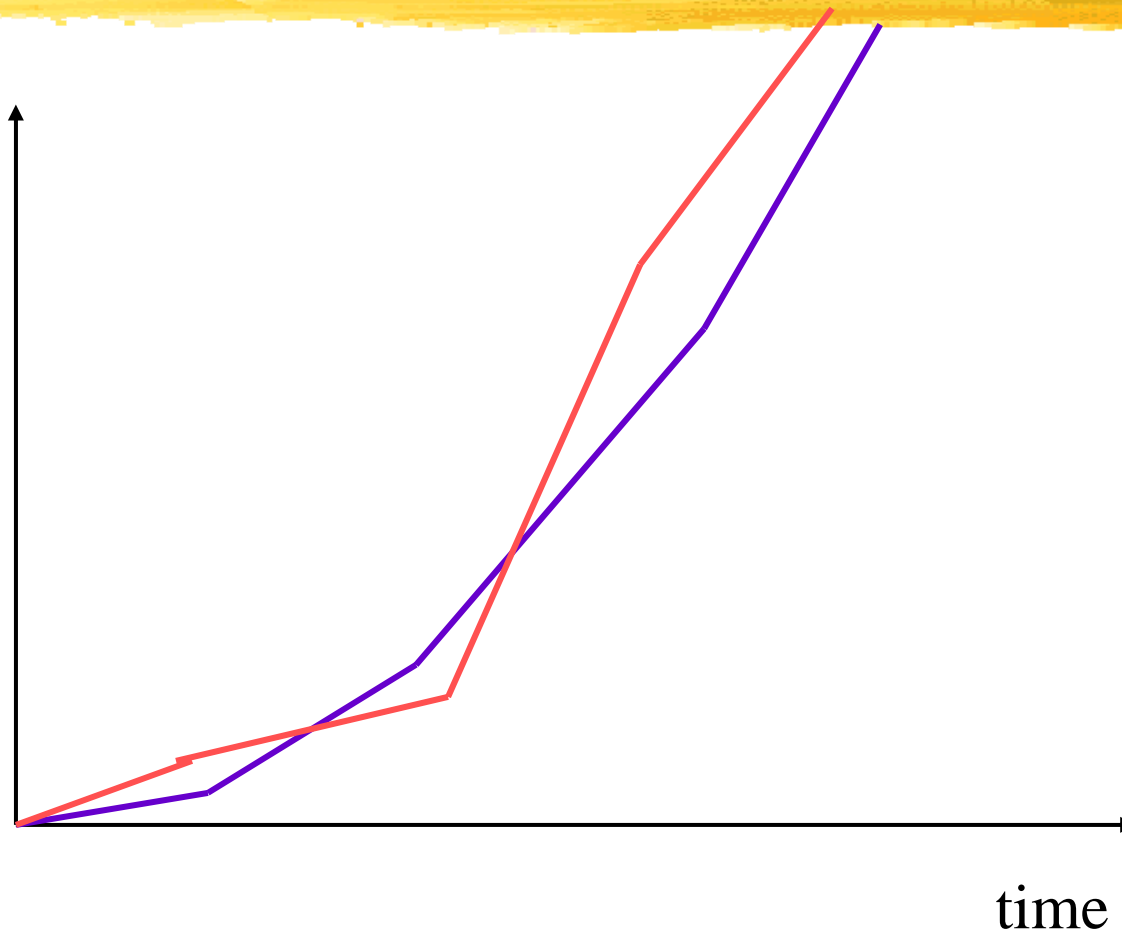
## ⌘ Carrier sense multiple access with collision detection:

- ⊞ sense collisions;

- ⊞ exponentially back off in time;

- ⊞ retransmit.

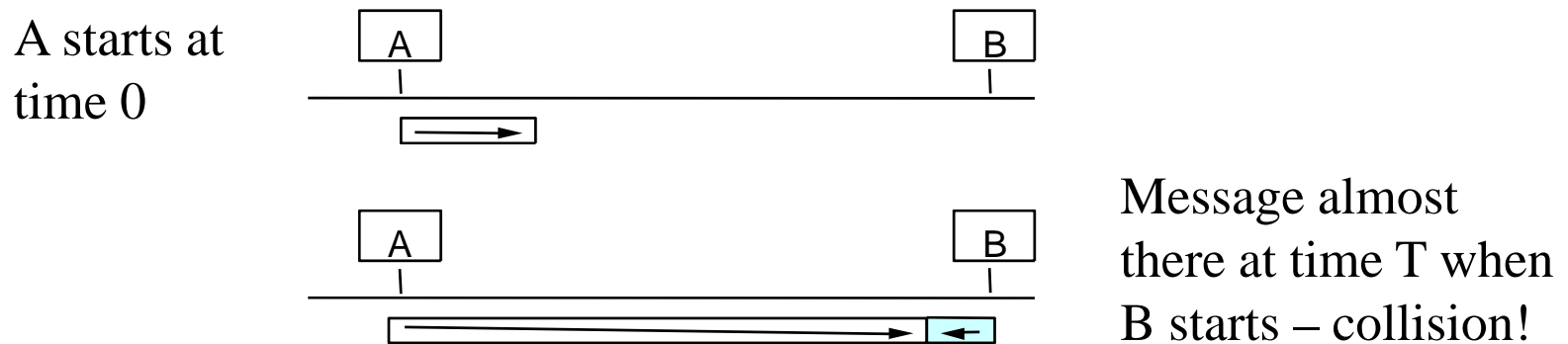
# Exponential back-off times



# Collisions

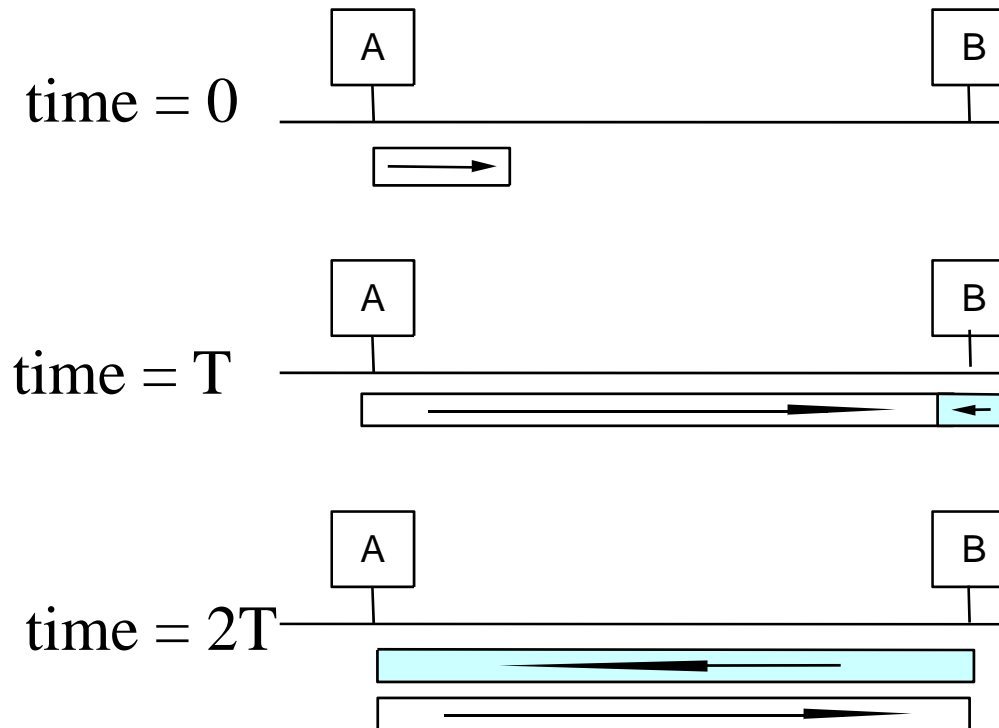
Collisions are caused when two adaptors transmit at the same time (adaptors sense collision based on voltage differences)

- Both found line to be idle
- Both had been waiting to for a busy line to become idle



How can we be sure A knows about the collision?

# Collision Detection contd.



# Collision Detection

- ⌘ How can A know that a collision has taken place?
  - ⊞ There must be a mechanism to insure retransmission on collision
  - ⊞ A's message reaches B at time  $T$
  - ⊞ B's message reaches A at time  $2T$
  - ⊞ So, A must still be transmitting at  $2T$
- ⌘ IEEE 802.3 specifies max value of  $2T$  to be 51.2us
  - ⊞ This relates to maximum distance of 2500m between hosts
  - ⊞ At 10Mbps it takes 0.1us to transmit one bit so 512 bits (64B) take 51.2us to send
  - ⊞ So, Ethernet frames must be at least 64B long
    - ⊞ 14B header, 46B data, 4B CRC
    - ⊞ Padding is used if data is less than 46B
- ⌘ Send jamming signal of 48 bits after collision is detected to insure all hosts see collision

# Exponential Backoff

- ⌘ If a collision is detected, delay and try again
- ⌘ Delay time is selected using binary exponential backoff
  - ⊞ 1st time: choose  $K$  from  $\{0,1\}$  then delay =  $K * 51.2\mu\text{s}$
  - ⊞ 2nd time: choose  $K$  from  $\{0,1,2,3\}$  then delay =  $K * 51.2\mu\text{s}$
  - ⊞  $n$ th time: delay =  $K \times 51.2\mu\text{s}$ , for  $K=0..2^n - 1$ 
    - ⊗ Note max value for  $k = 1023$
  - ⊞ give up after several tries (usually 16)
    - ⊗ Report transmit error to host
- ⌘ Why not just choose from small set for  $K$ 
  - ⊞ This works fine for a small number of hosts
  - ⊞ Large number of nodes would result in more collisions

# Fieldbus



- ⌘ Used for industrial control and instrumentation---factories, etc.
- ⌘ H1 standard based on 31.25 MB/s twisted pair medium.
- ⌘ High Speed Ethernet (HSE) standard based on 100 Mb/s Ethernet is used for backbone networks in industrial plants.

# Networks



- ⌘ Network-based design.
  - ☑ Communication analysis.
  - ☑ System performance analysis.
- ⌘ Internet.
- ⌘ Internet-enabled systems.
- ⌘ Vehicles as networks.
- ⌘ Sensor networks



# Communication analysis



- ⌘ First, understand delay for single message.
- ⌘ Delay for multiple messages depends on:
  - ☑ network protocol;
  - ☑ devices on network.

# Message delay

⌘ Assume:

☑ single message;

☑ no contention.

⌘ Delay:

$$\text{☑ } t_m = t_x + t_n + t_r$$

= xmtr overhead + network xmit time +  
rcvr overhead

# Example: I<sup>2</sup>C message delay

- ⌘ Network transmission time dominates.
- ⌘ Assume 100 kbits/sec, one 8-bit byte.
- ⌘ Number of bits in packet:
  - ⊠  $n_{\text{packet}} = \text{start} + \text{address} + \text{data} + \text{stop}$   
 $= 1 + 8 + 8 + 1 = 18 \text{ bits}$
- ⌘ Time required to transmit:  $1.8 \times 10^{-4} \text{ sec.}$
- ⌘ 20 instructions on 8 MHz controller adds  $2.5 \times 10^{-6}$  delay on xmtr, rcvr.
- ⌘  $T_m = 2.5 \text{ us} + 180 \text{ us} + 2.5 \text{ us} = 185 \text{ us}$

# Multiple messages

⌘ If messages can interfere with each other, analysis is more complex.

⌘ Model total message delay:

$$\boxed{\wedge} t_y = t_d + t_m$$

= wait time for network + message delay

# Arbitration and delay



- ⌘ Fixed-priority arbitration introduces unbounded delay for all but highest-priority device.
  - ☑ Unless higher-priority devices are known to have limited rates that allow lower devices to transmit.
- ⌘ Round-robin arbitration introduces bounded delay proportional to the number of devices ( $N$ ).

# Further complications



- ⌘ Acknowledgment time.
- ⌘ Transmission errors.

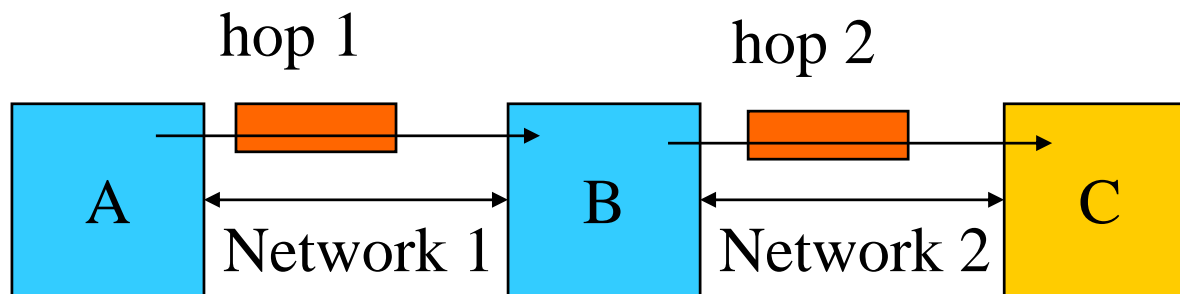
# Priority inversion in networks



- ⌘ In many networks, a packet cannot be interrupted.
- ⌘ Result is priority inversion:
  - ☑ low-priority message holds up higher-priority message.
- ⌘ Doesn't cause deadlock, but can slow down important communications.

# Multihop networks

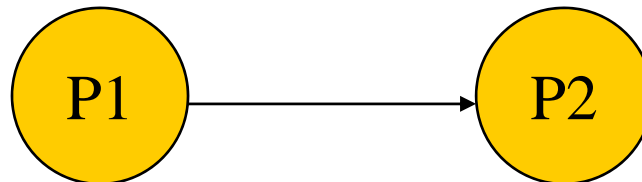
- ⌘ In multihop networks, one node receives message, then retransmits to destination (or intermediate).





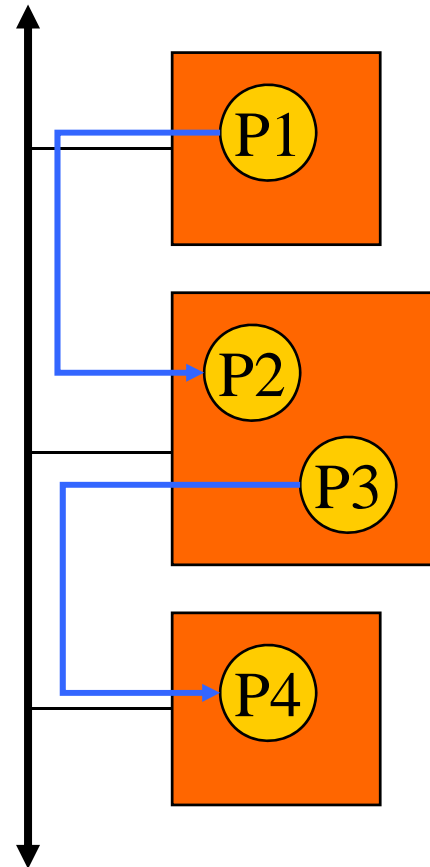
# System performance analysis

- ⌘ System analysis is difficult in general.
  - ☑ multiprocessor performance analysis is hard;
  - ☑ communication performance analysis is hard.
- ⌘ Simple example: uncertainty in P1 finish time -> uncertainty in P2 start time.



# Analysis challenges

- ⌘ P2 and P3 can delay each other, even though they are in separate tasks.
- ⌘ Delays in P1 propagate to P2, then P3, then to P4.



# Lower bounds on system



## ⌘ Computational requirements:

- ☑ sum up process requirements over least-common multiple of periods, average over one period.

## ⌘ Communication requirements:

- ☑ Count all transmissions in one period.

# Hardware platform design



⌘ Need to choose:

- ☑ number and types of PEs;

- ☑ number and types of networks.

⌘ Evaluate a platform by allocating processes, scheduling processes and communication.

# I/O-intensive systems



⌘ Start with I/O devices, then consider computation:

- ☑ catalog required devices;
- ☑ identify critical deadlines;
- ☑ chooses devices that can share PEs;
- ☑ analyze communication times;
- ☑ choose PEs to go with devices.

# Computation-intensive systems



- ⌘ Start with shortest-deadline tasks:
  - ☑ Put shortest-deadline tasks on separate PEs.
  - ☑ Check for interference on critical communications.
  - ☑ Allocate low-priority tasks to common PEs wherever possible.
- ⌘ Balance loads wherever possible.

# Internet-enabled embedded system



- ⌘ Internet-enabled embedded system: any embedded system that includes an Internet interface (e.g., refrigerator).
- ⌘ Internet appliance: embedded system designed for a particular Internet task (e.g. email).
- ⌘ Examples:
  - ☑ Cell phone.
  - ☑ Laser printer.

# TCP/IP & OSI



- ⌘ In OSI reference model terminology - the TCP/IP protocol suite covers the *transport* and *network* layers.
- ⌘ TCP/IP can be used on many data-link layers (can support many network hardware implementations).

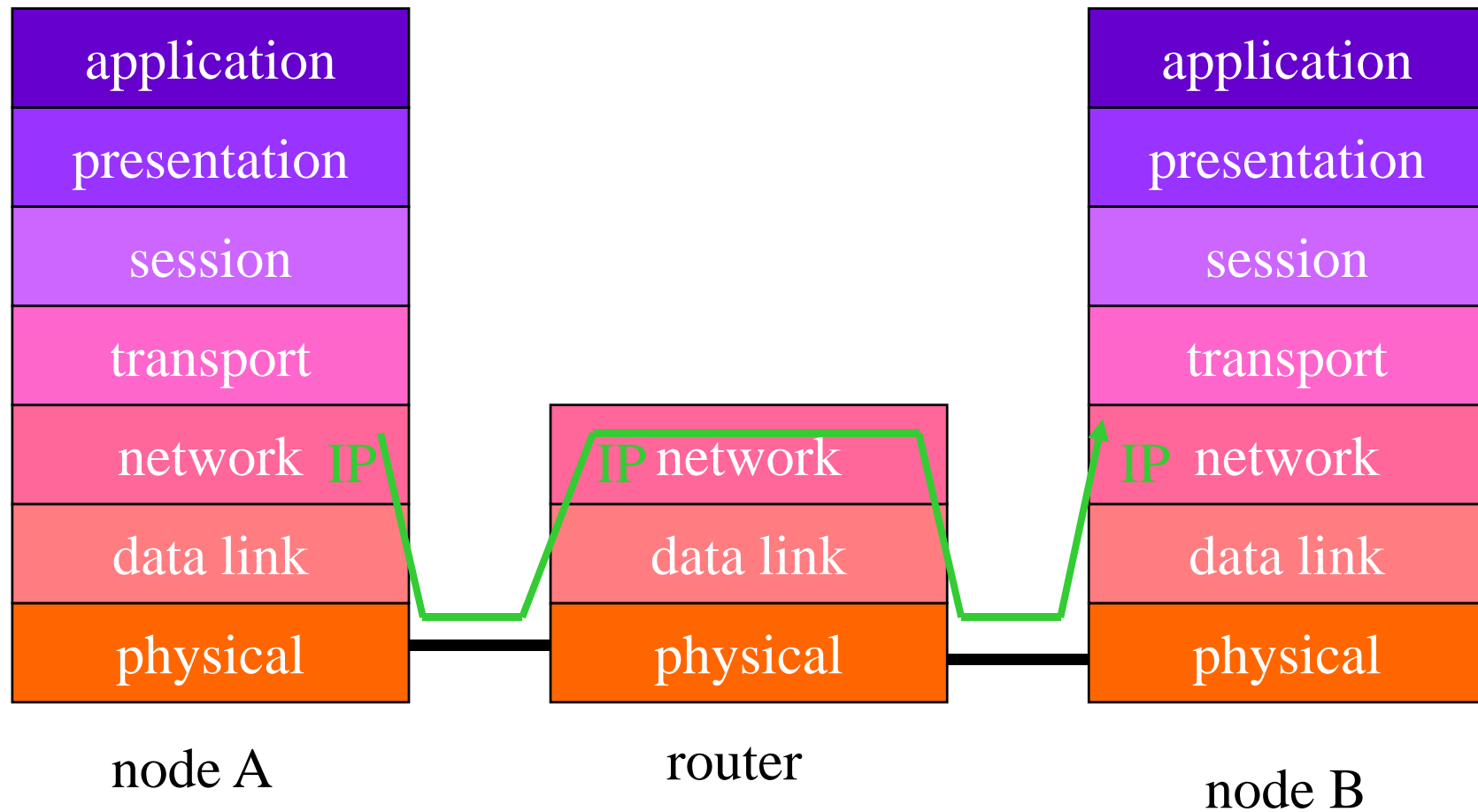


# Internet Protocol



- ⌘ **Internet Protocol (IP)** is basis for Internet.
- ⌘ IP is the network layer
  - ☑ packet delivery service (host-to-host).
  - ☑ translation between different data-link protocols.
- ⌘ Provides an **internetworking** standard: between two Ethernets, Ethernet and token ring, etc.
- ⌘ Higher-level services are built on top of IP.

# IP in communication



# Internet routing



## ⌘ Best effort routing:

- ☑ doesn't guarantee data delivery at IP layer.

## ⌘ Routing can vary:

- ☑ session to session;

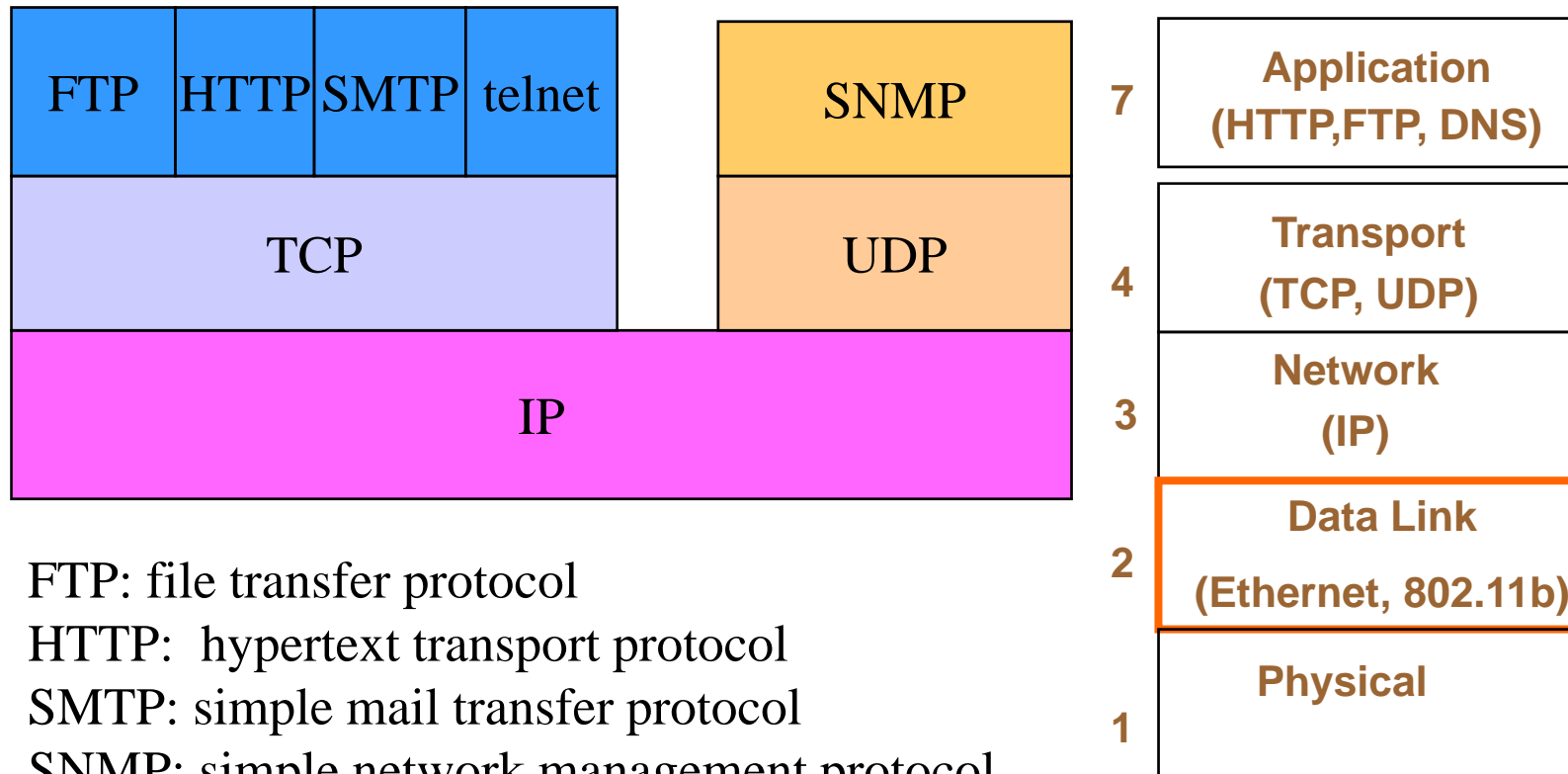
- ☑ packet to packet.

# Higher-level Internet services



- ⌘ **Transmission Control Protocol (TCP)** provides connection-oriented service.
- ⌘ **Quality-of-service (QoS) guaranteed services** are under development.

# The Internet service stack



FTP: file transfer protocol

HTTP: hypertext transport protocol

SMTP: simple mail transfer protocol

SNMP: simple network management protocol

UDP: user datagram protocol

# IP packet



## ⌘ Includes:

- ☑ version, service type, length
- ☑ time to live, protocol
- ☑ source and destination address
- ☑ data payload

⌘ Maximum data payload is 65,535 bytes.

# IP Datagrams

⌘ IP provides connectionless, unreliable delivery of *IP datagrams*.

☑ Connectionless: each datagram is independent of all others.

☑ Unreliable: there is no guarantee that datagrams are delivered correctly or even delivered at all.

An IP packet is called a *datagram*

# IP Datagram

1 byte

1 byte

1 byte

1 byte

<b>VERS</b>	<b>HL</b>	<b>Service</b>	<b>Fragment Length</b>	
<b>Datagram ID</b>			<b>FLAG</b>	<b>Fragment Offset</b>
<b>TTL (time to live)</b>		<b>Protocol</b>	<b>Header Checksum</b>	
<b>Source Address</b>				
<b>Destination Address</b>				
<b>Options (if any)</b>				
<b>Data</b>				



# IP addresses



- ⌘ 32 bits in early IP, 128 bits in IPv6.
- ⌘ Typically written in form **xxx.xx.xx.xx**.
- ⌘ Names (foo.baz.com) translated to IP address by **domain name server** (DNS).

# IP Addresses



⌘ IP addresses are not the same as the underlying data-link (MAC) addresses.


Why ?

# IP Addresses



- ⌘ IP is a network layer - it must be capable of providing communication between hosts on different kinds of networks (different data-link implementations).
- ⌘ The address must include information about what *network* the receiving host is on. This is what makes routing feasible.

# IP Addresses

- ⌘ IP addresses are *logical* addresses (not physical)
- ⌘ 32 bits.  IPv4 (version 4)
- ⌘ Includes a network ID and a host ID.
- ⌘ Every host must have a unique IP address.
- ⌘ IP addresses are assigned by a central authority (*American Registry for Internet Numbers* for North America).

# Formats of IP Addresses

## Class



8 bits

8 bits

8 bits

8 bits

# Formats of IP Addresses

## Class A

- 128 possible network IDs
- over 4 million host IDs per network ID

## Class B

- 16K possible network IDs
- 64K host IDs per network ID

## Class C

- over 2 million possible network IDs
- about 256 host IDs per network ID

# Network and Host IDs

- ⌘ A Network ID is assigned to an organization by a global authority.
- ⌘ Host IDs are assigned locally by a system administrator.
- ⌘ Both the Network ID and the Host ID are used for routing.

# IP Addresses

⌘ IP Addresses are usually shown in *dotted decimal* notation:

1.2.3.4            00000001 00000010 00000011 00000100

⌘ cs.rpi.edu is 128.213.1.1

10000000 11010101 00000001 00000001



CS has a class B network



# Host and Network Addresses

- ⌘ A single network interface is assigned a single IP address called the *host* address.
- ⌘ A host may have multiple interfaces, and therefore multiple *host* addresses.
- ⌘ Hosts that share a network all have the same IP *network* address (the network ID).

# IP Broadcast and Network Addresses

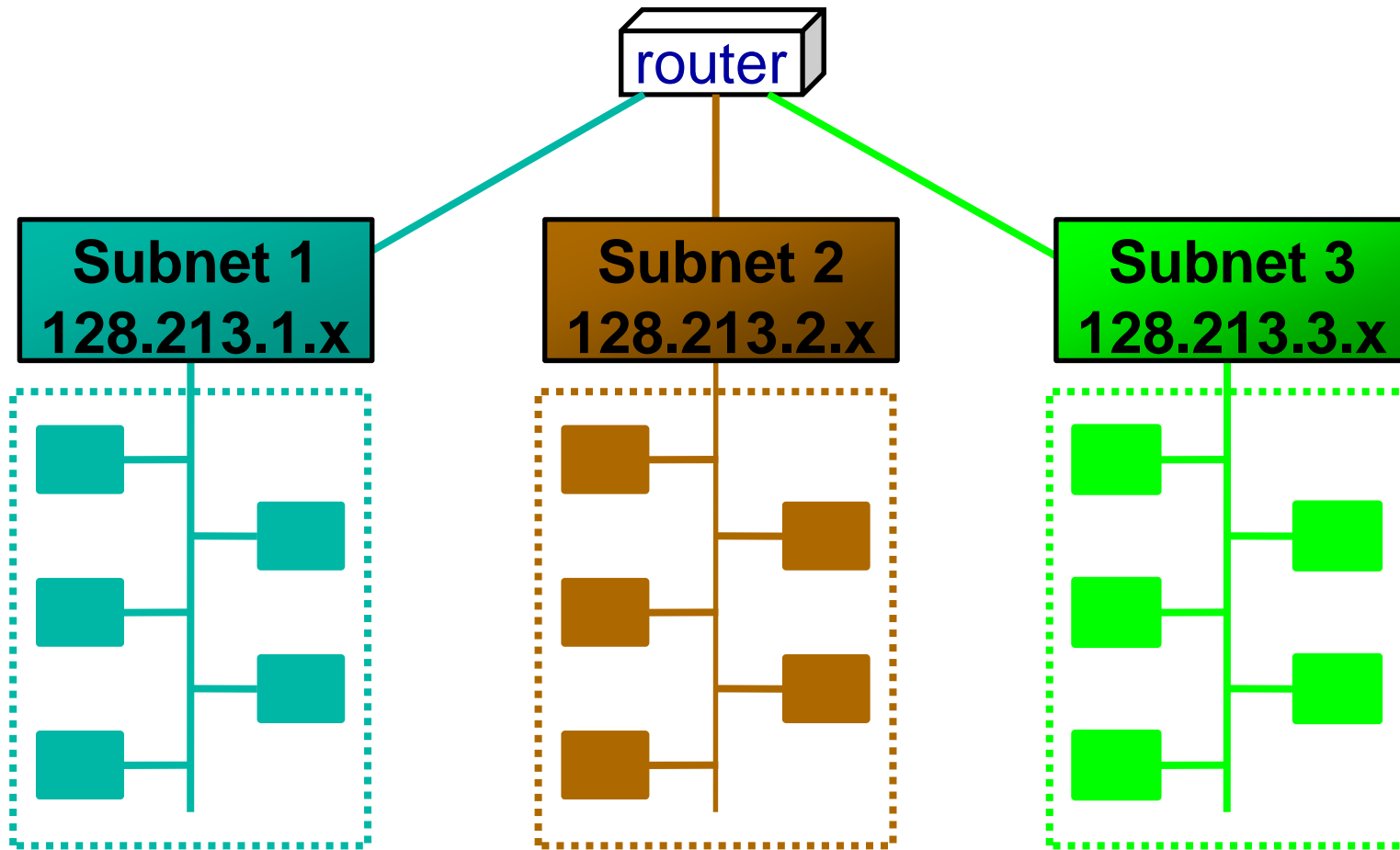
- ⌘ An IP broadcast address has a host ID of all 1s.
- ⌘ IP broadcasting is not necessarily a true broadcast, it relies on the underlying hardware technology.
- ⌘ An IP address that has a host ID of all 0s is called a *network address* and refers to an entire network.

# Subnet Addresses

- ⌘ An organization can subdivide its host address space into groups called subnets.
- ⌘ The subnet ID is generally used to group hosts based on the physical network topology.

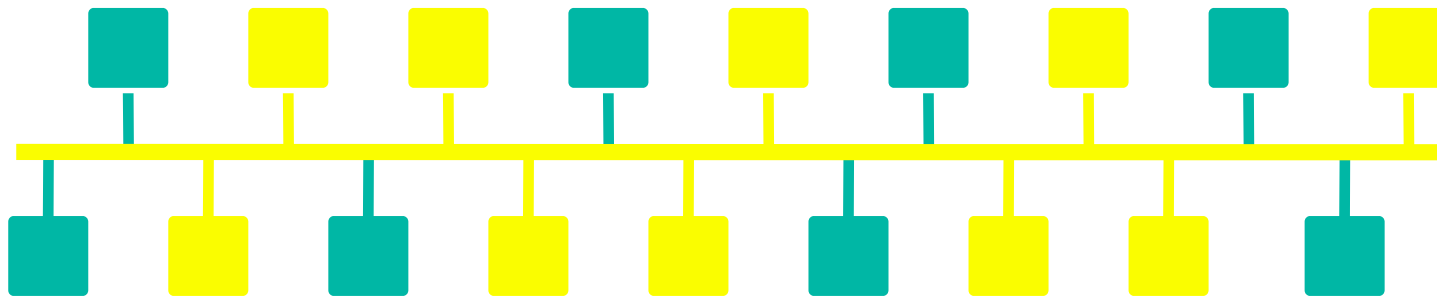


# Subnetting



# Subnetting

- ⌘ Subnets can simplify routing.
- ⌘ IP subnet broadcasts have a hostID of all 1s.
- ⌘ It is possible to have a single wire network with multiple subnets.



# Mapping IP Addresses to Hardware Addresses

- ⌘ IP Addresses are not recognized by hardware.
- ⌘ If we know the IP address of a host, how do we find out the hardware address ?
- ⌘ The process of finding the hardware address of a host given the IP address is called *Address Resolution*

# Reverse Address Resolution

- ⌘ The process of finding out the IP address of a host given a hardware address is called *Reverse Address Resolution*
- ⌘ Reverse address resolution is needed by diskless workstations when booting (which used to be quite common).

# ARP



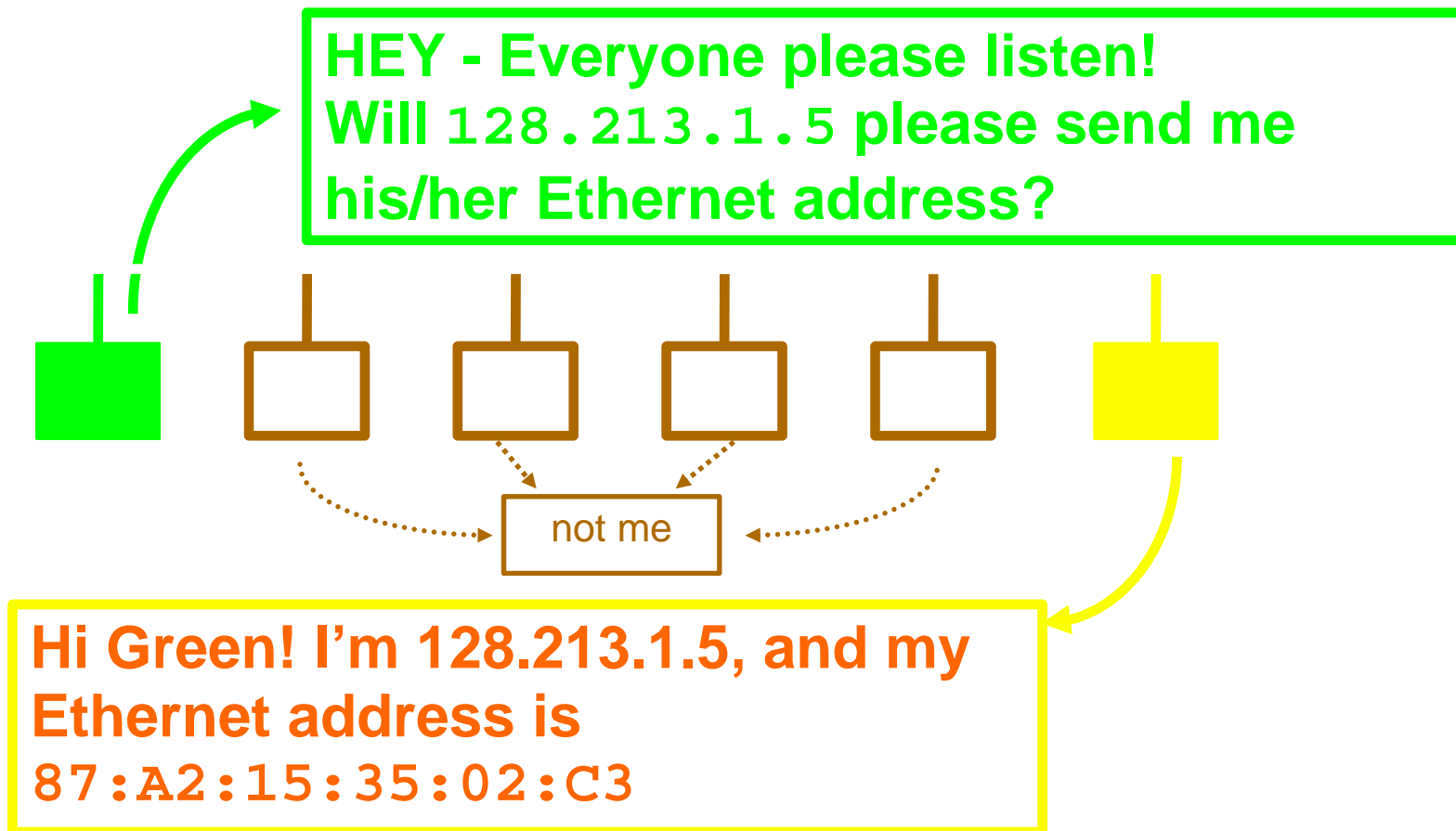
- ⌘ The *Address Resolution Protocol* is used by a sending host when it knows the IP address of the destination but needs the Ethernet (or whatever) address.
- ⌘ ARP is a broadcast protocol - every host on the network receives the request.
- ⌘ Each host checks the request against its IP address - the right one responds.



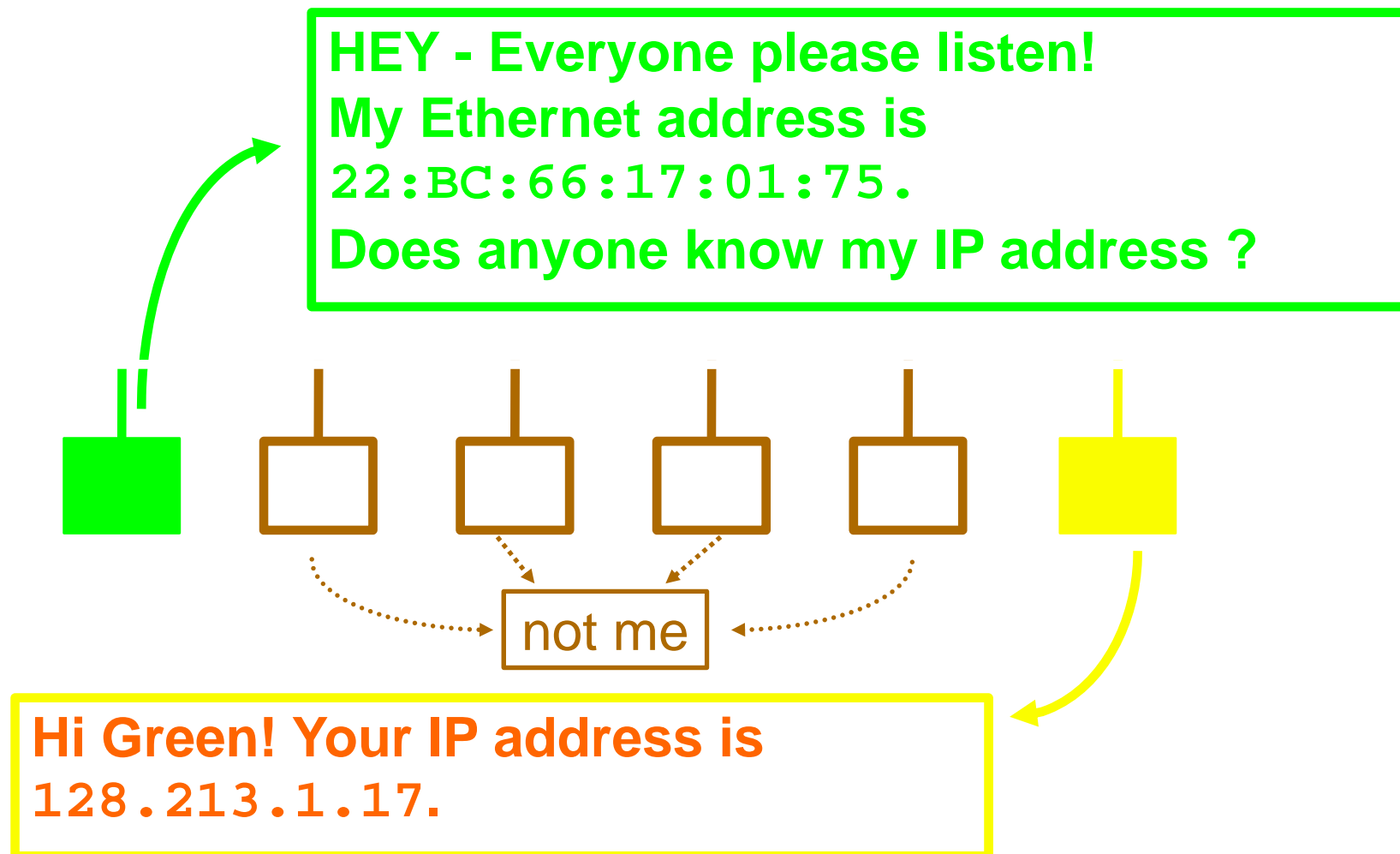
# ARP (cont.)

- ⌘ ARP does not need to be done every time an IP datagram is sent - hosts *remember* the hardware addresses of each other.
- ⌘ Part of the ARP protocol specifies that the receiving host should also remember the IP and hardware addresses of the sending host.

# ARP conversation



# RARP conversation



# Services provided by IP

- ⌘ Connectionless Delivery (each datagram is treated individually).
- ⌘ Unreliable (delivery is not guaranteed).
- ⌘ Fragmentation / Reassembly (based on hardware MTU).
- ⌘ Routing.
- ⌘ Error detection.

# IP Datagram Fragmentation



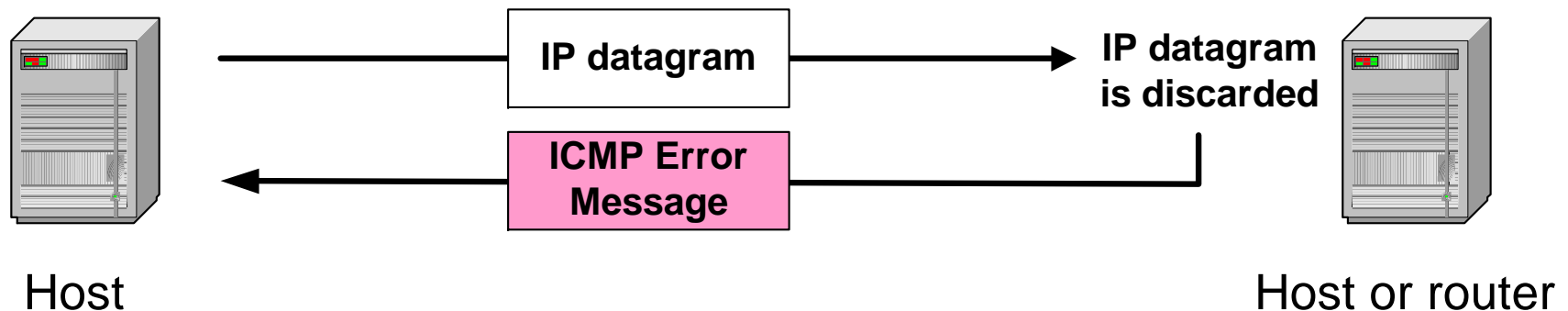
- ⌘ Each fragment (packet) has the same structure as the IP datagram.
- ⌘ IP specifies that datagram reassembly is done only at the destination (not on a hop-by-hop basis).
- ⌘ If any of the fragments are lost - the entire datagram is discarded (and an ICMP message is sent to the sender).

ICMP: internet control message protocol

# IP Flow Control & Error Detection

- ⌘ If packets arrive too fast - the receiver discards excessive packets and sends an ICMP message to the sender (SOURCE QUENCH).
- ⌘ If an error is found (header checksum problem) the packet is discarded and an ICMP message is sent to the sender.

# ICMP Error message



- ⌘ ICMP error messages report error conditions
- ⌘ Typically sent when a datagram is discarded
- ⌘ Error message is often passed from ICMP to the application program

# ICMP protocol



- ⌘ ICMP is a protocol used for exchanging control messages.
- ⌘ ICMP uses IP to deliver messages.
- ⌘ ICMP messages are usually generated and processed by the IP software, not the user process.

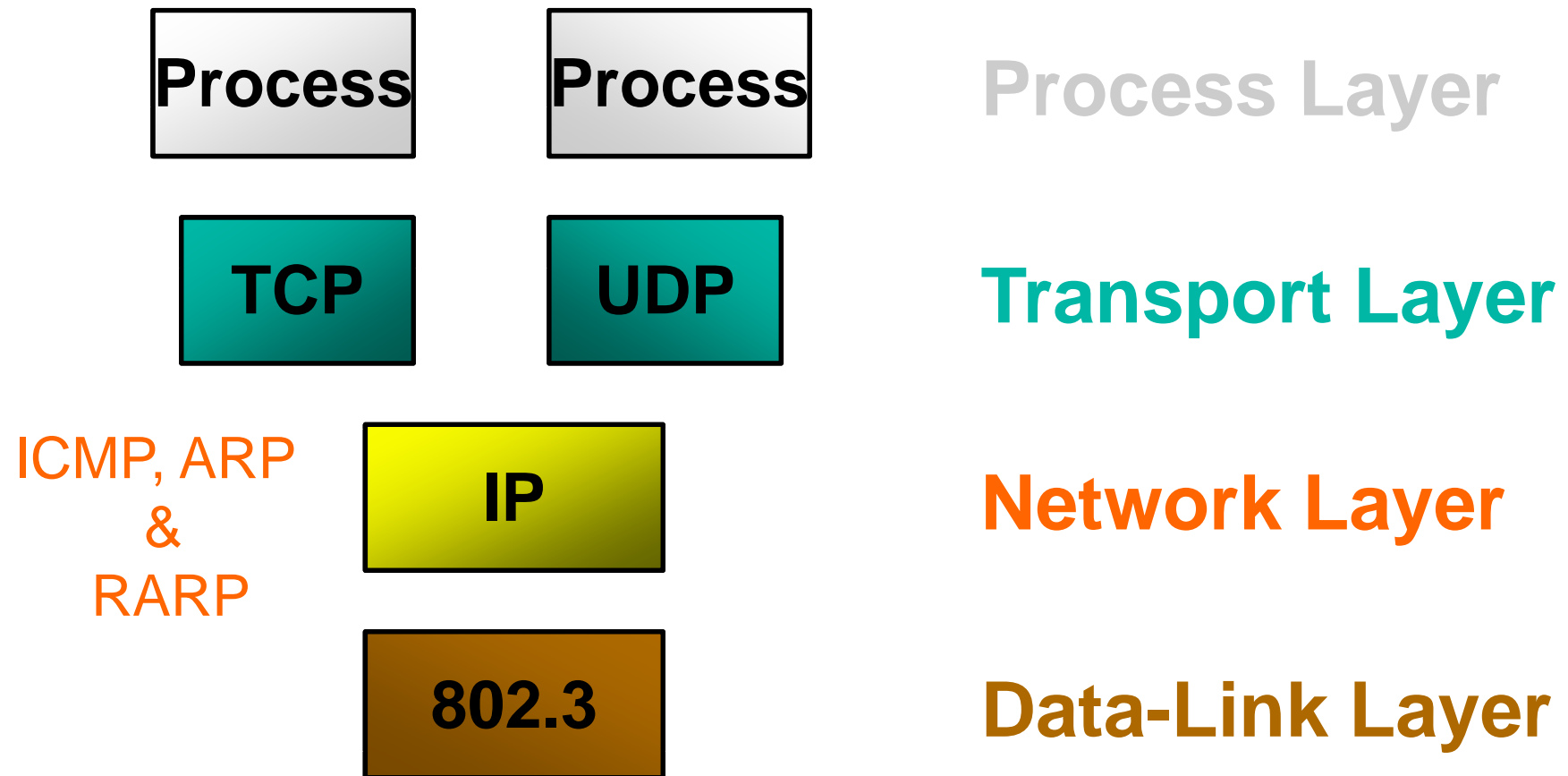


# ICMP Message Types



- ⌘ Echo Request
- ⌘ Echo Response
- ⌘ Destination Unreachable
- ⌘ Redirect
- ⌘ Time Exceeded
- ⌘ Redirect (route change)
- ⌘ there are more ...

# Transport Layer & TCP/IP



# UDP (User Datagram Protocol)

- ⌘ UDP is a transport protocol
  - ☑ communication between processes
- ⌘ UDP uses IP to deliver datagrams to the right host.
- ⌘ UDP uses *ports* to provide communication services to individual processes.

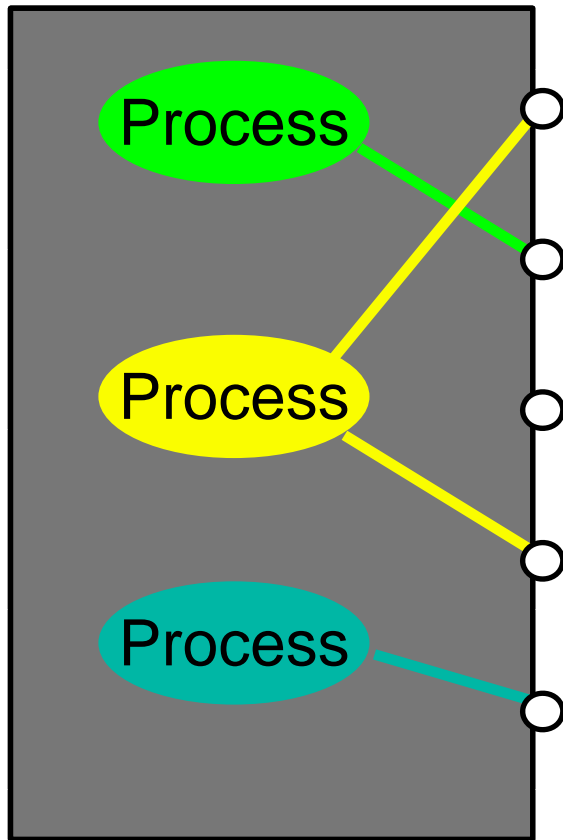
# Ports



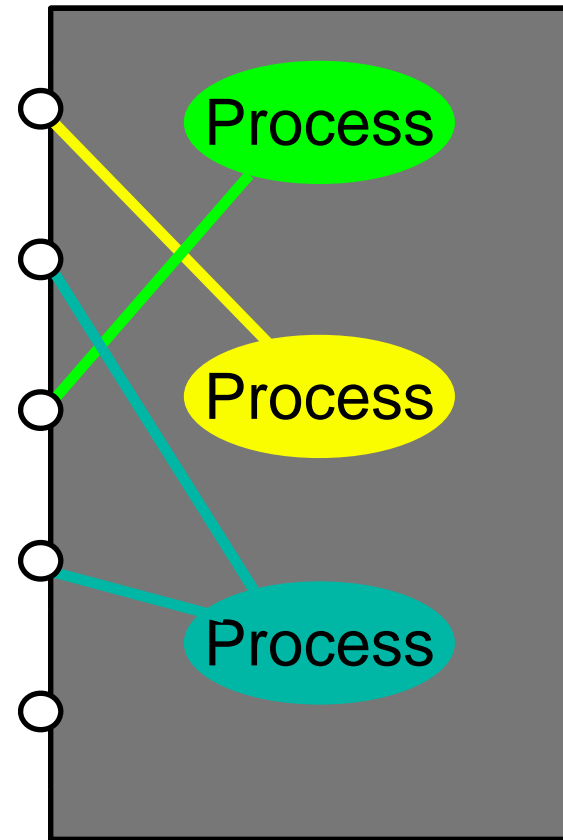
- ⌘ TCP/IP uses an abstract destination point called a protocol port.
- ⌘ Ports are identified by a positive integer.
- ⌘ Operating systems provide some mechanism that processes use to specify a port.

# Ports

Host A



Host B



# TCP



- ⌘ TCP is an alternative transport layer protocol supported by TCP/IP.
- ⌘ TCP provides:
  - ☑ Connection-oriented
  - ☑ Reliable
  - ☑ Full-duplex
  - ☑ Byte-Stream

# Buffering



- ⌘ TCP is responsible for buffering data and determining when it is time to send a datagram.
- ⌘ It is possible for an application to tell TCP to send the data it has buffered without waiting for a buffer to fill up.

# Full Duplex



- ⌘ TCP provides transfer in both directions (over a single virtual connection).
- ⌘ To the application program these appear as 2 unrelated data streams, although TCP can piggyback control and data communication by providing control information (such as an ACK) along with user data.



# TCP Ports



- ⌘ Interprocess communication via TCP is achieved with the use of ports (just like UDP).
- ⌘ UDP ports have no relation to TCP ports (different name spaces).

# TCP Segments

- ⌘ The chunk of data that TCP asks IP to deliver is called a *TCP segment*.
- ⌘ Each segment contains:
  - ☑ data bytes from the byte stream
  - ☑ control information that identifies the data bytes

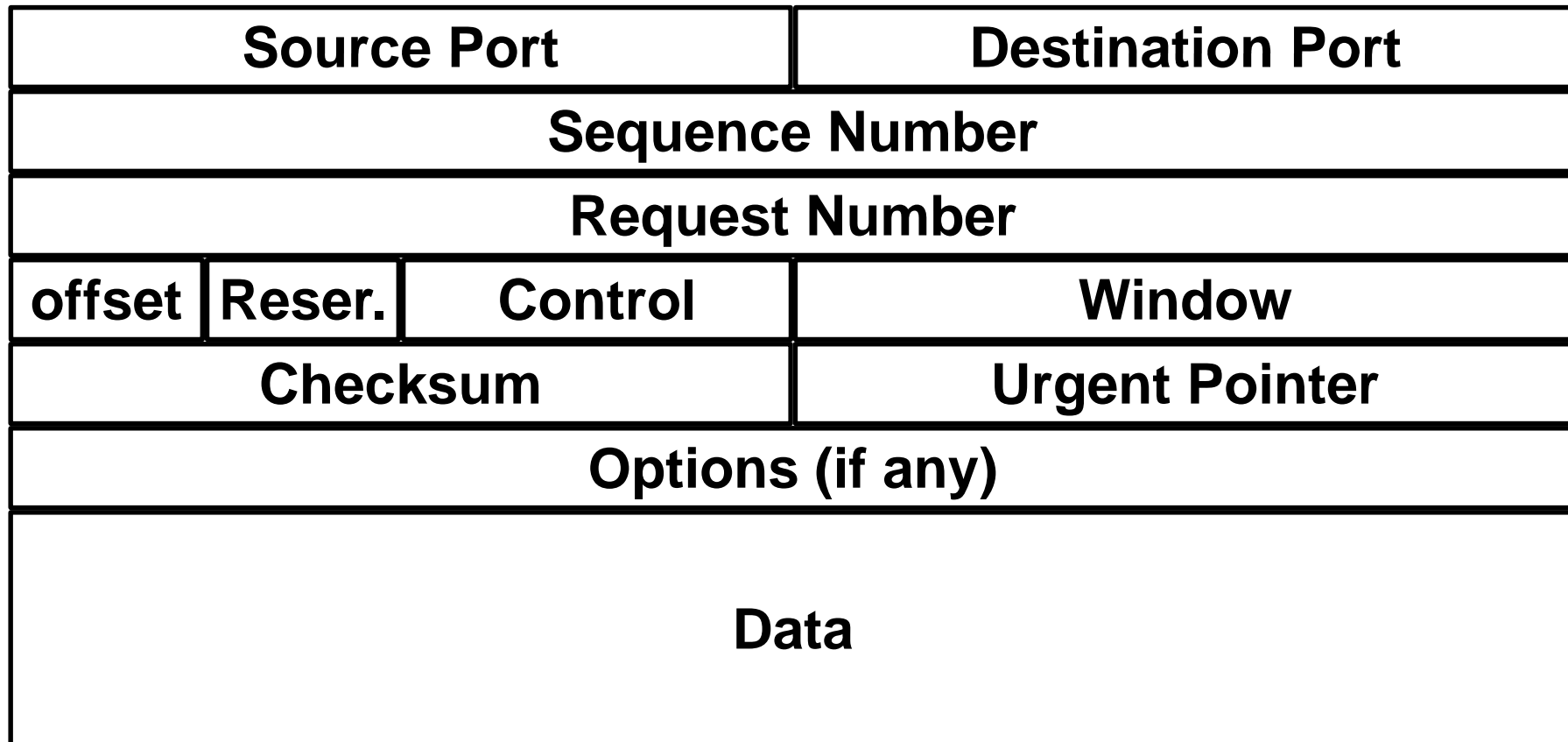
# TCP Segment Format

1 byte

1 byte

1 byte

1 byte



# Addressing in TCP/IP



⌘ Each TCP/IP address includes:

- ☑ Internet Address
- ☑ Protocol (UDP or TCP)
- ☑ Port Number

NOTE: TCP/IP is a *protocol suite* that includes IP, TCP and UDP.

# TCP vs. UDP



Q: Which protocol is better ?

A: It depends on the application.

TCP provides a connection-oriented, reliable, byte stream service (lots of overhead).

UDP offers minimal datagram delivery service (as little overhead as possible).

# TCP/IP Summary



⌘ IP: network layer protocol

☑ unreliable datagram delivery between hosts.

⌘ UDP: transport layer protocol

☑ unreliable datagram delivery between processes.

⌘ TCP: transport layer protocol

☑ reliable, byte-stream delivery between processes.

# Example: Javacam

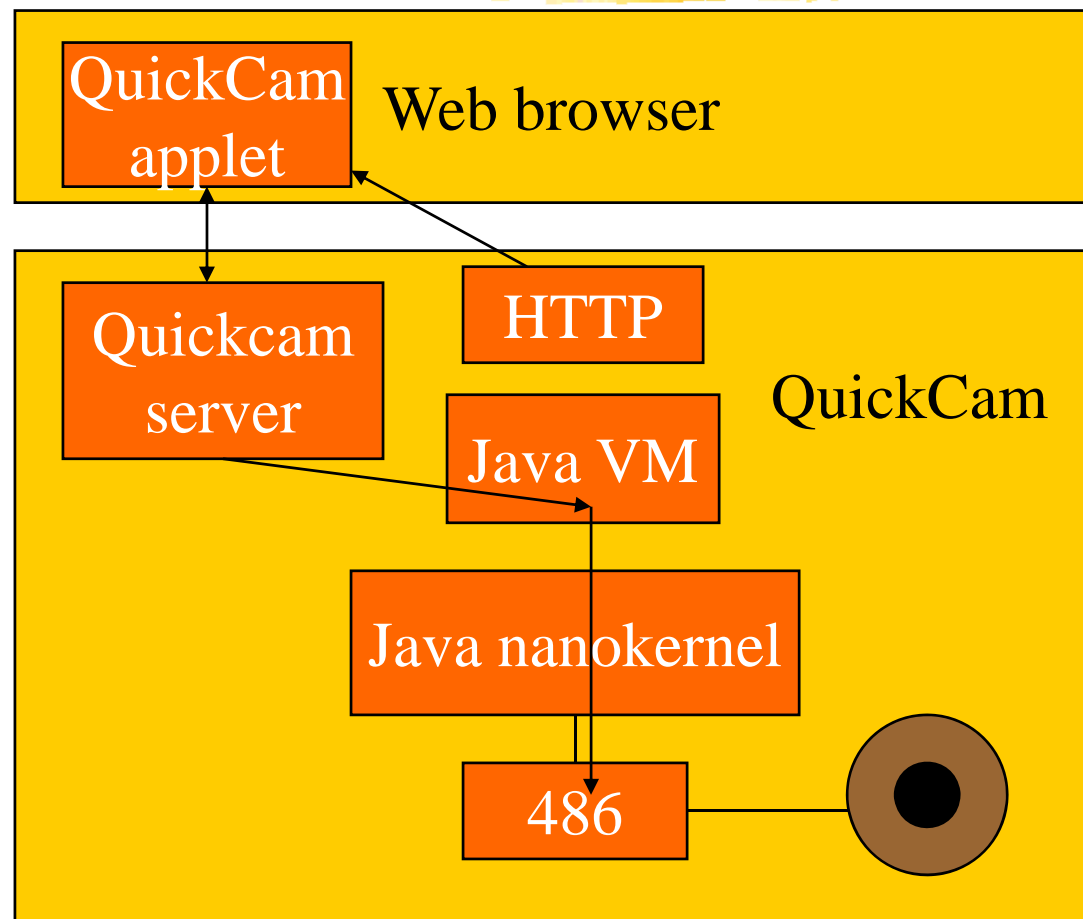


## ⌘ Hardware platform:

- ☒ parallel-port camera;
- ☒ National Semi NS486SXF;
- ☒ 1.5 Mbytes memory.

## ⌘ Uses memory-efficient Java Nanokernel.

# Javacam architecture





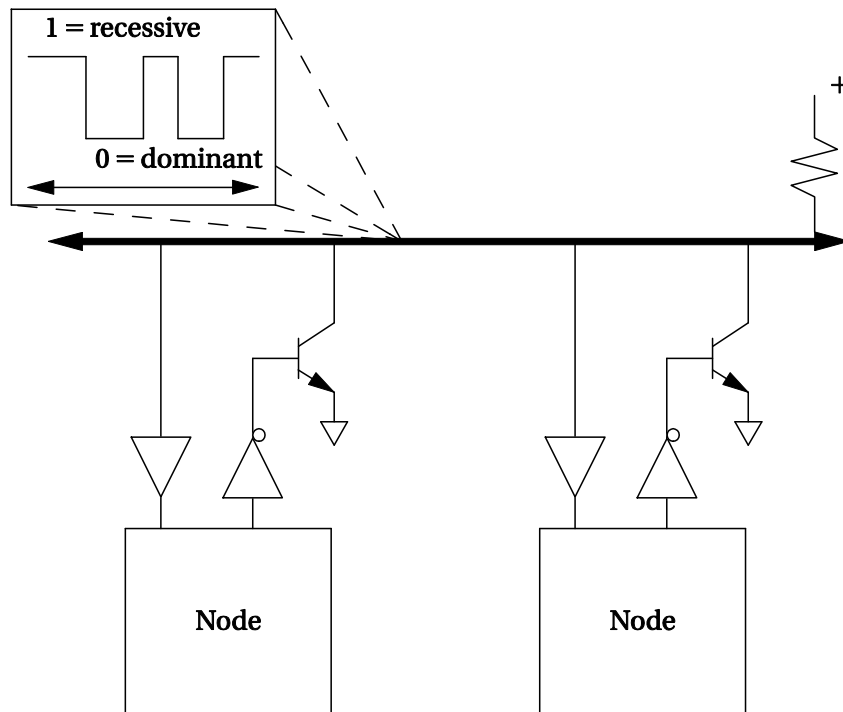
# Vehicles as networks



- ⌘ 1/3 of cost of car/airplane is electronics/avionics.
- ⌘ Dozens of microprocessors are used throughout the vehicle.
- ⌘ Network applications:
  - ☑ Vehicle control.
  - ☑ Instrumentation.
  - ☑ Communication.
  - ☑ Passenger entertainment systems.

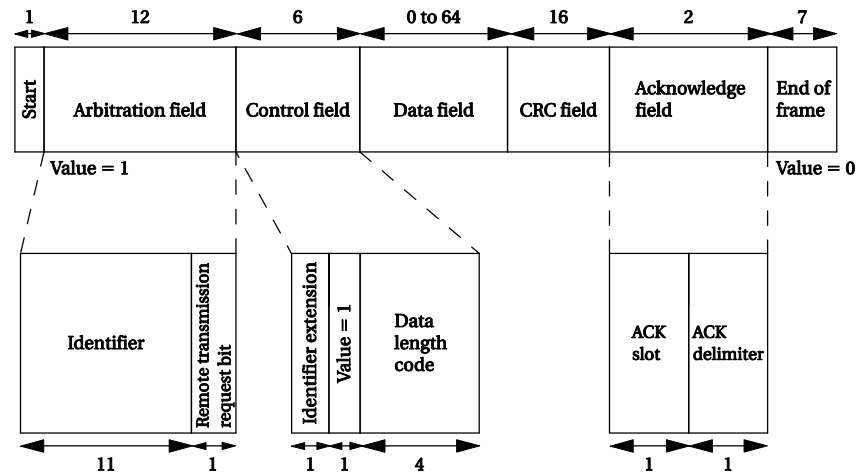
# CAN bus

- ⌘ First used in 1991.
- ⌘ Serial bus, 1 Mb/sec up to 40 m.
- ⌘ Synchronous bus.
- ⌘ Logic 0 dominates logic 1 on bus.
- ⌘ Arbitrated with CSMA/AMP:
  - ☑ Arbitration on message priority.



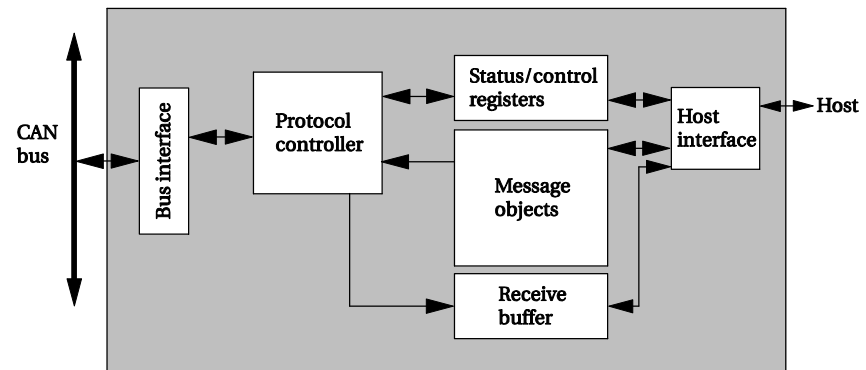
# CAN data frame

- ⌘ 11 bit destination address.
- ⌘ RTR bit determines read/write from/to destination.
- ⌘ Any node can detect bus error, interrupt packet for retransmission.



# CAN controller

- ⌘ Controller implements physical and data link layers.
- ⌘ No network layer needed---bus provides end-to-end connections.



# Other vehicle busses



⌘ FlexRay is next generation:

- ☑ Time triggered protocol.

- ☑ 10 Mb/s.

⌘ Local Interconnect Network (LIN) connects devices in a small area (e.g., door).

⌘ Passenger entertainment networks:

- ☑ Bluetooth.

- ☑ Media Oriented Systems Transport (MOST).

# Avionics



- ⌘ Anything permanently attached to the aircraft must be certified by FAA/national agency.
- ⌘ Traditional architecture uses separate electronics for each instrument/device.
  - ☒ Line replaceable unit (LRU) can be physically removed and replaced.
- ⌘ Federated architecture shares processors across a subsystem (nav/comm, etc.)

# Sensor networks



- ⌘ Wireless networks, small nodes.
- ⌘ Ad hoc networks---organizes itself without system administrator:
  - ☑ Must be able to declare membership in network, find other networks.
  - ☑ Must be able to determine routes for data.
  - ☑ Must update configuration as nodes enter/leave.

# Node capabilities



- ⌘ Must be able to turn radio on/off quickly with low power overhead.
  - ☑ Communication/computation power = 100x.
- ⌘ Radios should operate at several different power levels to avoid interference with other nodes.
- ⌘ Must buffer, route network traffic.



# Networks



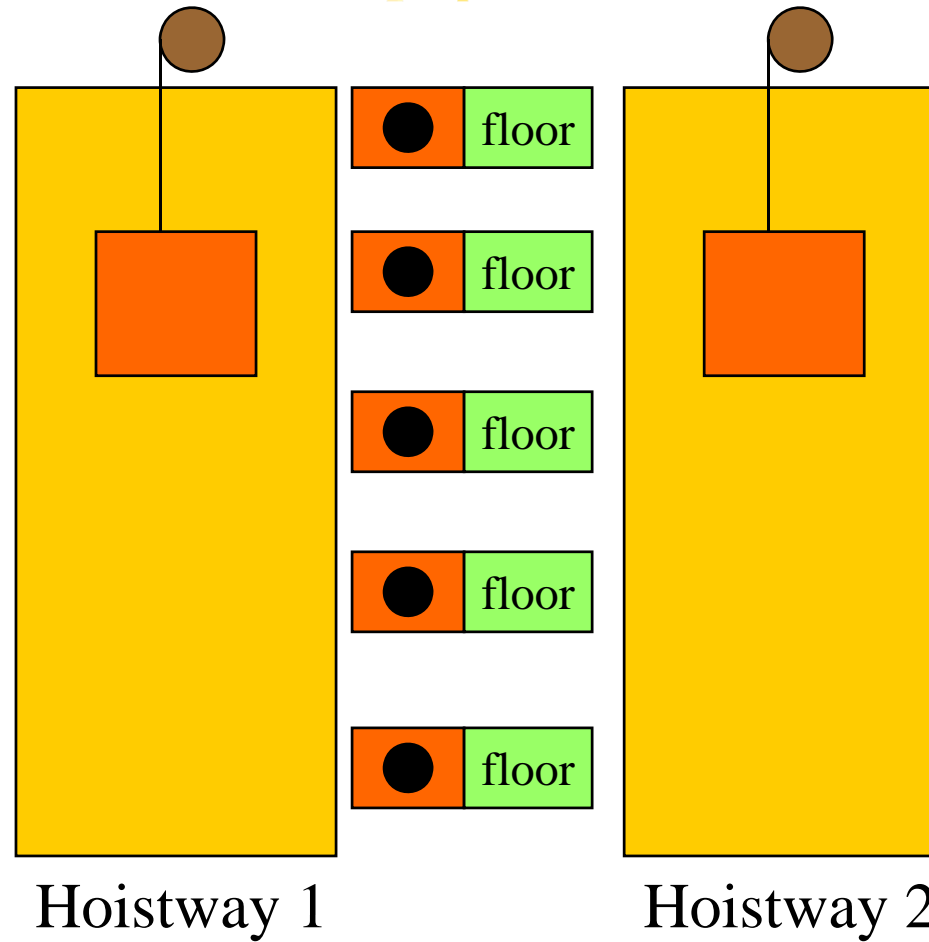
⌘ Example: elevator controller.

# Terminology



- ⌘ **Elevator car**: holds passengers.
- ⌘ **Hoistway**: elevator shaft.
- ⌘ **Car control panel**: buttons in each car.
- ⌘ **Floor control panel**: elevator request, etc. per floor.

# Elevator system

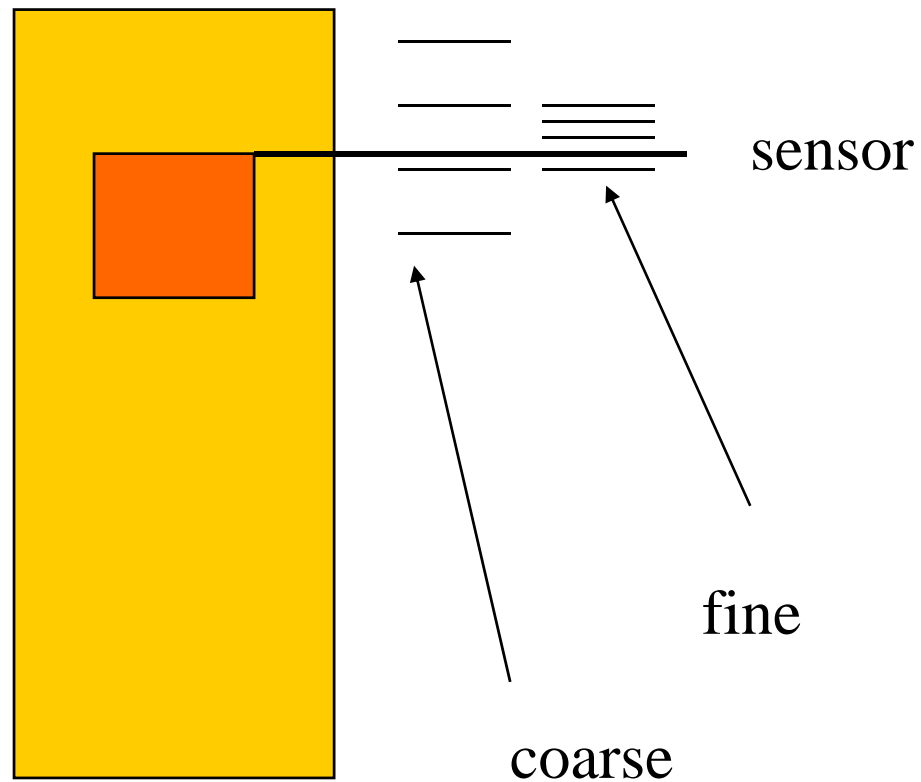


# Theory of operation



- ⌘ Each floor has control panel, display.
- ⌘ Each car has control panel:
  - ☑ one button per floor;
  - ☑ emergency stop.
- ⌘ Controlled by a single controller.

# Elevator position sensing



# Elevator control



⌘ Elevator control has up and down.

☑ To stop, disable both.

⌘ Master controller:

☑ reads elevator positions;

☑ reads requests;

☑ schedules elevators;

☑ controls movement;

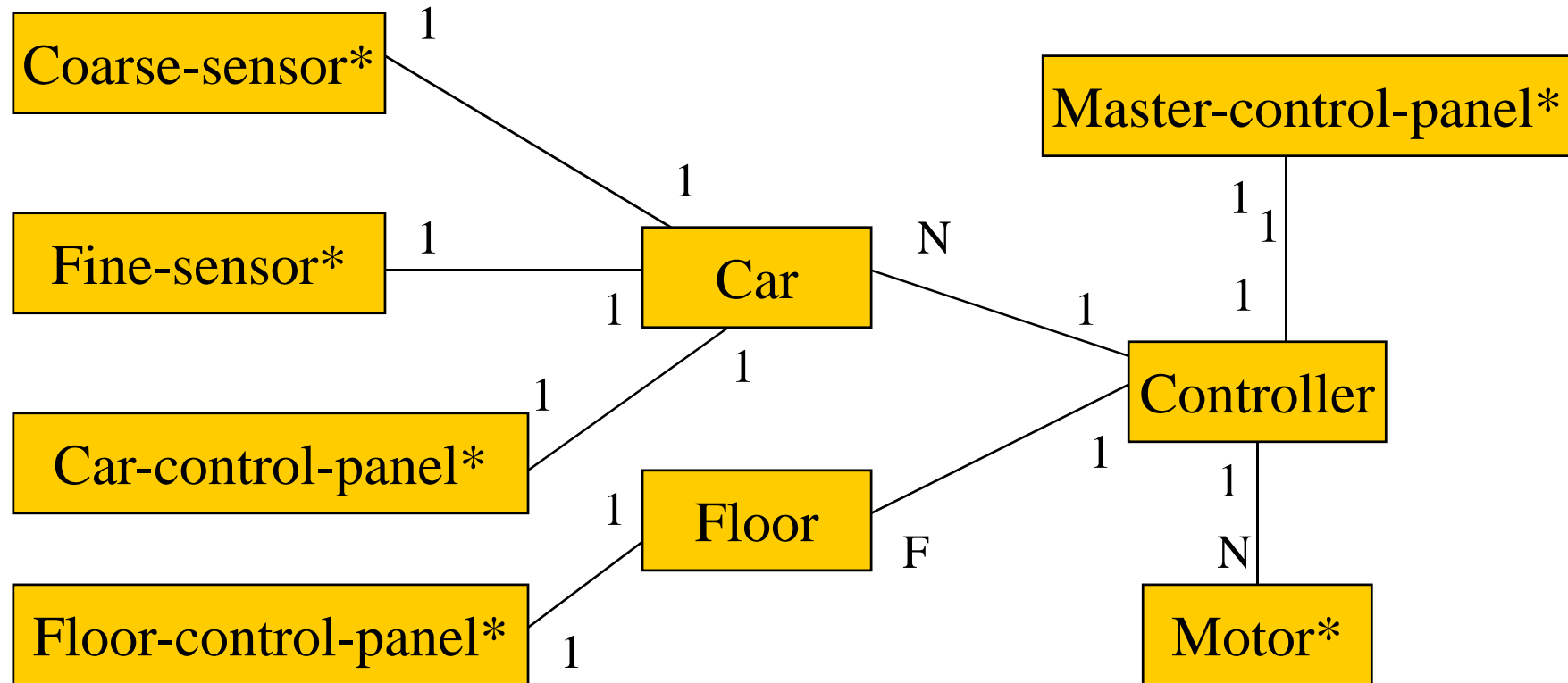
☑ controls doors.

# Elevator system requirements



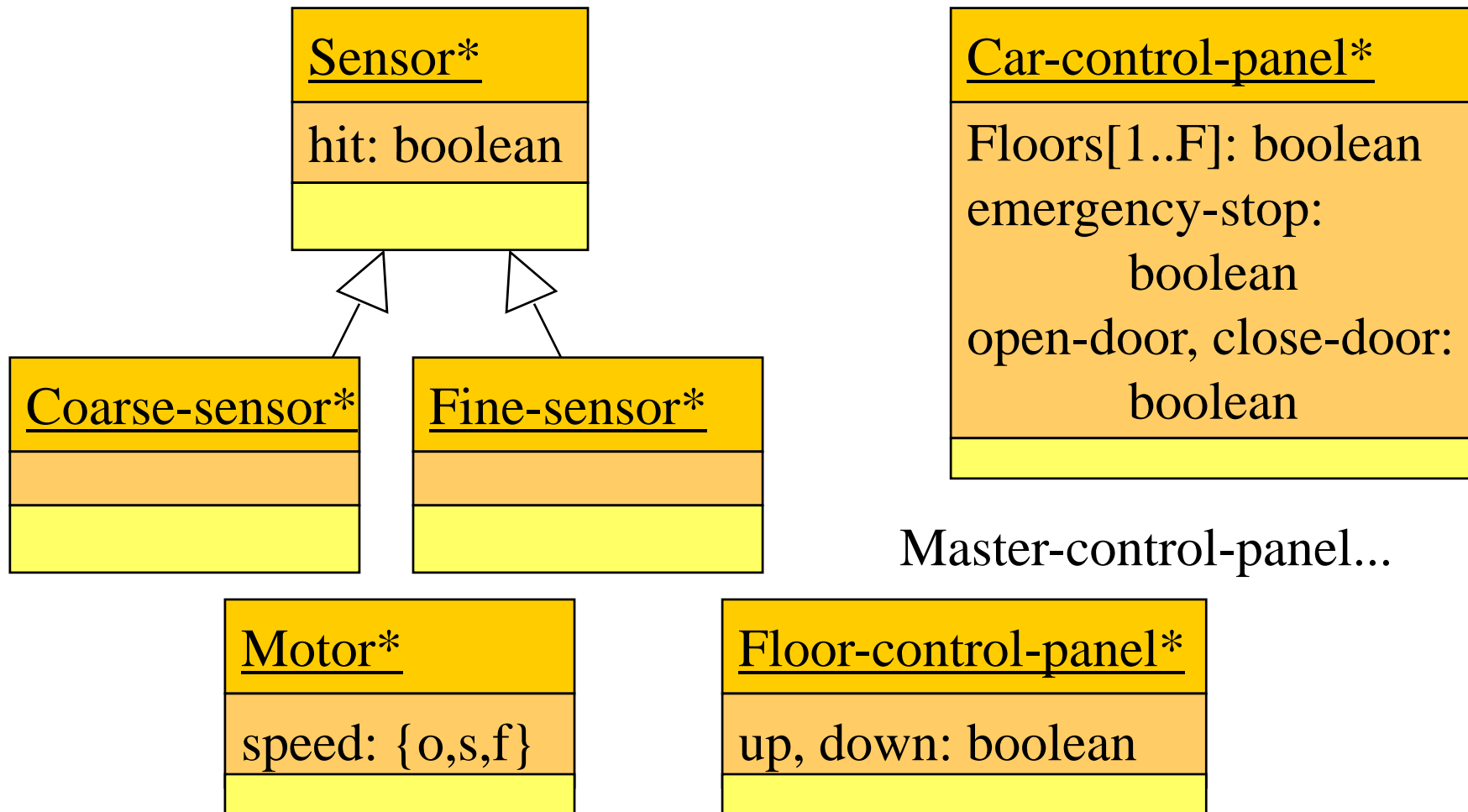
name	elevator system
inputs	F floor control, N position, N car control, 1 master
outputs	F displays, N motor controllers
functions	responds to requests, operates safely
performance	elevator control is time-critical
manufacturing cost	electronics is small part of total
power	electronics consumes small fraction of total
physical size/weight	cabling is important

# Elevator system class diagram

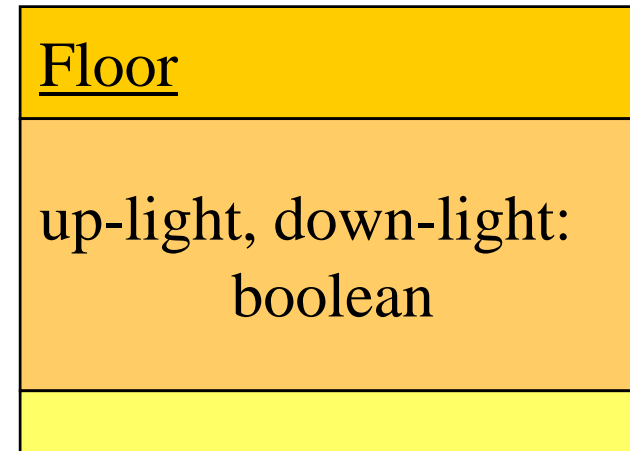
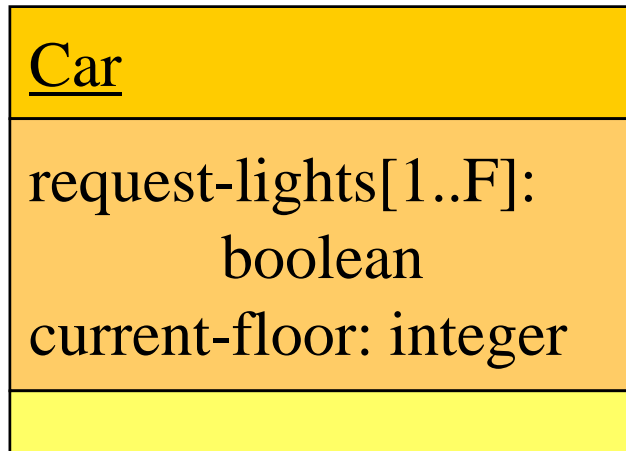




# Physical interfaces



# Car and Floor classes



# Controller class



## Controller

car-floor[1..H]: integer  
emergency-stop[1..H]:  
integer

scan-cars()  
scan-floors()  
scan-master-panel()  
operate()

# Architecture



⌘ Computation and I/O occur at:

☑ floor control panels/displays;

☑ elevator cars;

☑ system controller.

# Panels and cab controller



⌘ Panels are straightforward---no real-time requirements.

⌘ Cab controller:

- ☑ read buttons and send events to system controller;

- ☑ read sensor inputs and send to system controller.

# System controller



⌘ Must take inputs from many sources:

☑ car controllers;

☑ floors.

⌘ Must control cars to hard real-time deadlines.

⌘ User interface, scheduling are soft deadlines.

# Testing



⌘ Build an elevator simulator using an FPGA:

- ☑ simulate multiple elevators;
- ☑ simulate real-time control demands.