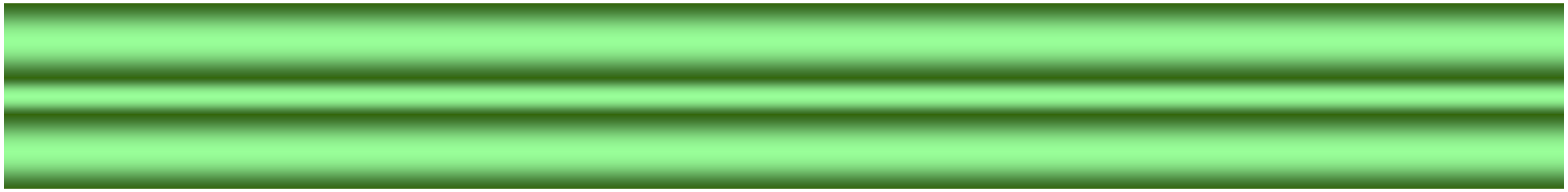
EE414 Embedded Systems

Ch 7. Network Interface

Part 2/2



Byung Kook Kim
School of Electrical Engineering
Korea Advanced Institute of Science and Technology

Overview

- 7.1 Advanced Communication Principles
- 7.2 Parallel Protocols
- 7.3 Wireless Protocols
- 7.4 Serial Protocols
- 7.5 Controller Area Network
- 7.6 CAN Protocol

- 7.7 Ethernet Hardware
- 7.8 Ethernet Protocol
- 7.9 Socket Programming

7.7 Ethernet Hardware

- **Ethernet**
  - Developed at Xerox PARC in 70s
  - Local area networking (LAN) standard
  - Wireless networks (802.11) to gigabit Ethernet

- Capabilities of Ethernet
  - Gain access to a network
  - Send data to a host computer
  - Access printers, file servers, databases, and Internet
  - Monitor and control embedded system
    - Weather station: Sensor, ADC, AT90S8515 AVR, and Ethernet interface
  - Gateway, firewall, bridge, switch, etc.

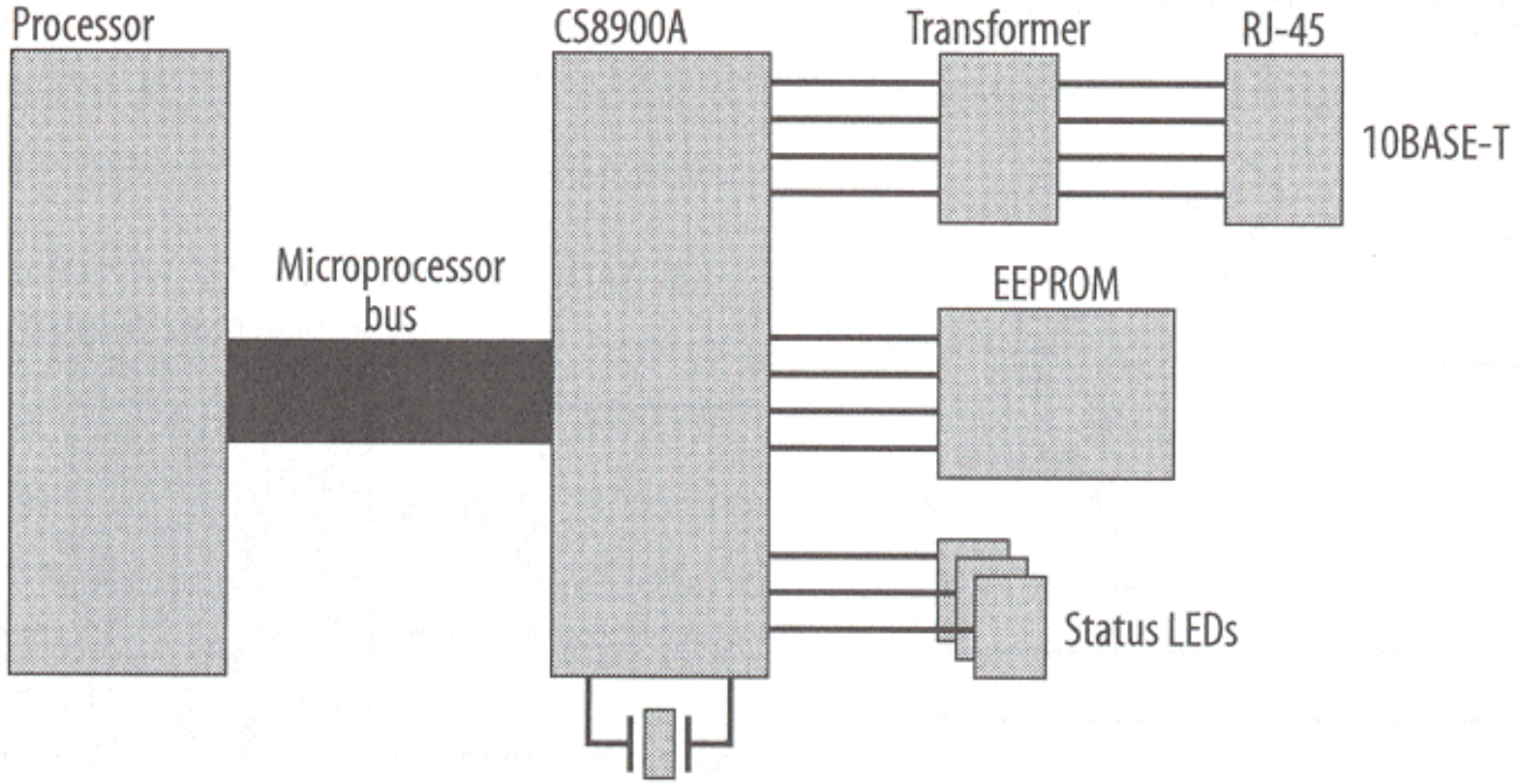
Ethernet (II)

- Adding an Ethernet interface
  - CS8900A
    - Single-chip 10 Mbps Ethernet controller by Cirrus Logic (*http://www.cirrus.com*), formerly Crystal Semiconductor.
    - Supports 10BASE-2, 10-BASET, and AUI (Attachment Unit Interface) Ethernet ports.
  - RJ-45 connector
    - Uses UTP (Unshielded Twisted Pair) category 5 cable (CAT5)
    - Four wires are used: Tx pair, Rx pair.
    - Pinouts →

Pin	Signal name	Purpose [Gbps extension]	Wire color
1	TD+	Transmitted data TX+-D1	White/orange
2	TD-	Transmitted data TX-_D1	Orange
3	RD+	Received data RX+_D2	White green
4	NC	BI+_D3	Blue
5	NC	BI-_D3	White/blue
6	RD-	Received data RX-_D2	Green
7	NC	BI+-D4	White/brown
8	NC	BI-_D4	Brown

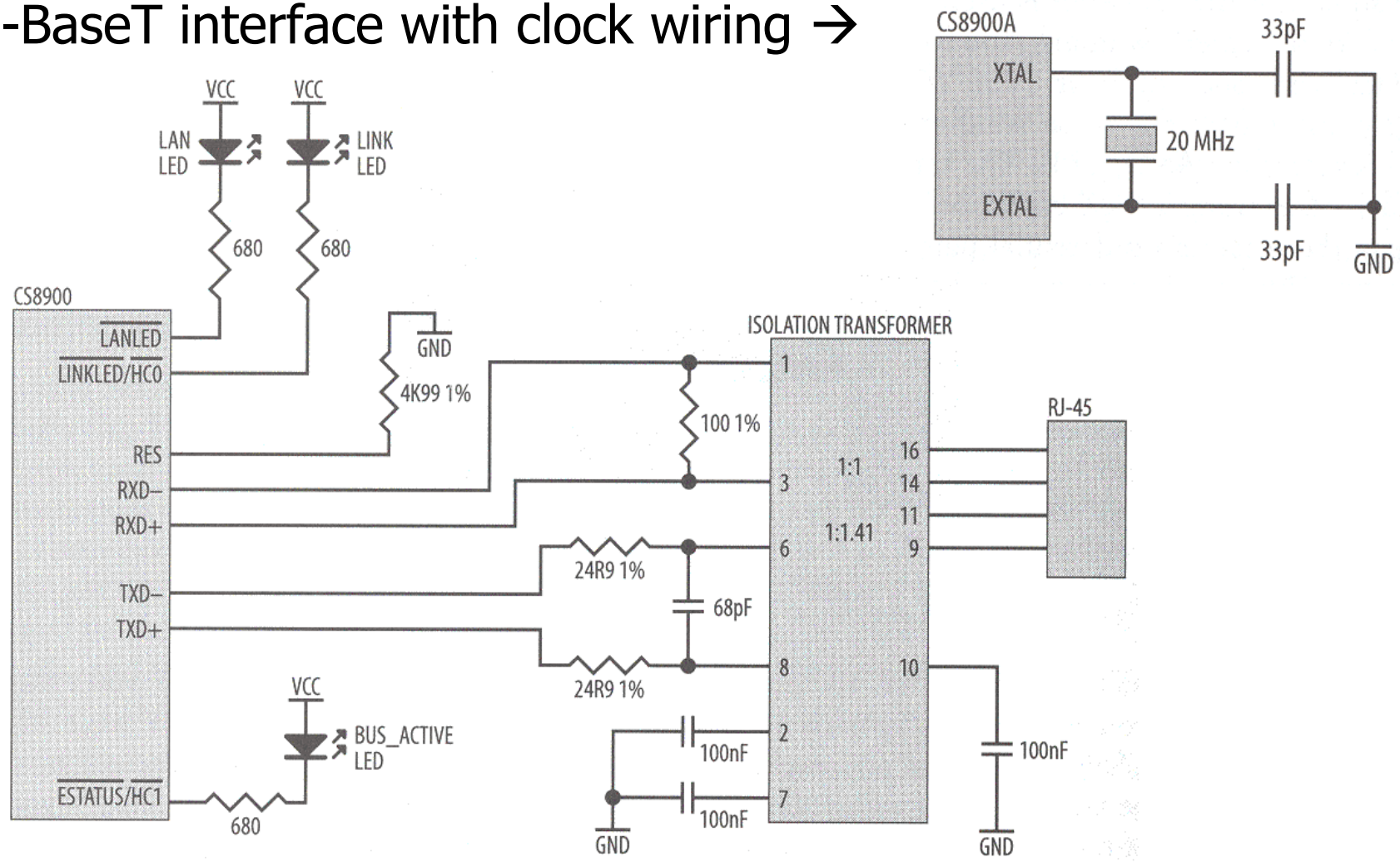
Ethernet (III)

- Block diagram of a CS9800A implementation →



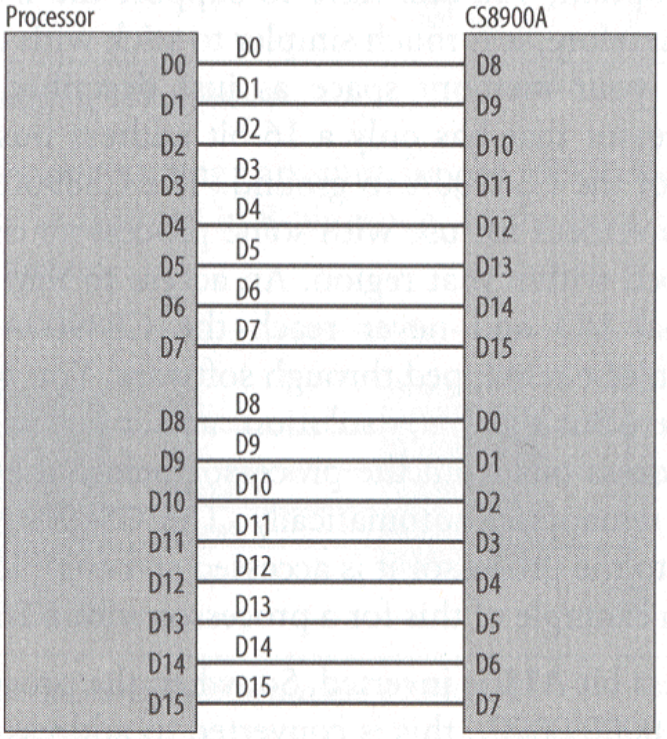
Ethernet (V)

- 10-BaseT interface with clock wiring →



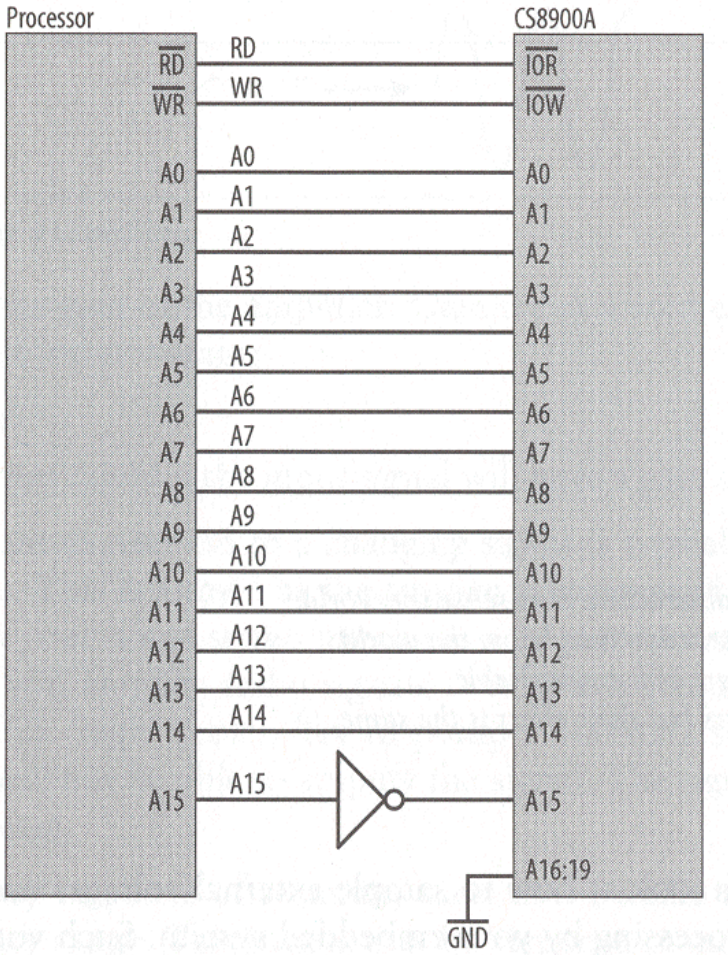
Ethernet (VI)

- Host interface of CS8900
  - Supports 16-bit ISA bus architecture
    - Easily adapted to work with non-ISA processors
    - Set SHBEb high
  - Also supports 8-bit data bus
    - Tie SHBEb to ground
  - Little-endian operation
    - Big-endian processors (Motorola, DSP56805)
      - Byte-swap in software
      - Byte-swap in hardware →



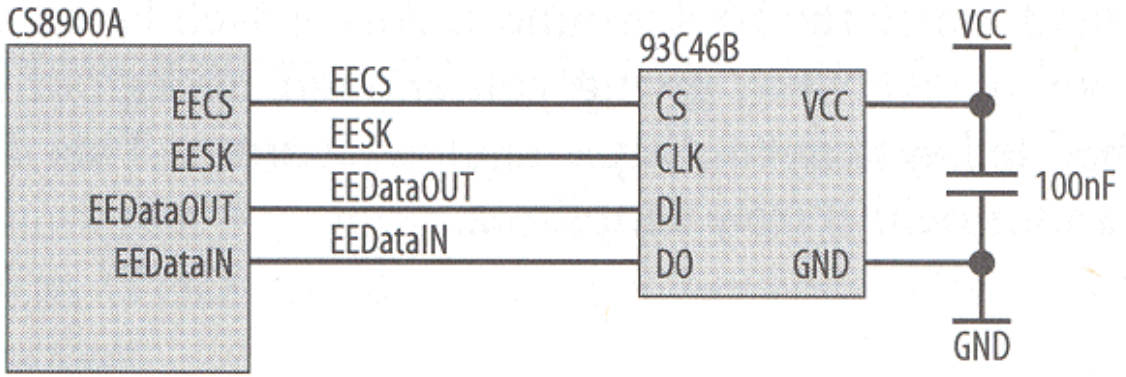
Ethernet (VII)

- Host interface of CS8900 (Cont'd)
  - 20 address inputs
    - ISA-bus device
      - Supports separate memory and I/O address spaces
    - CHIPSELb low: memory-mapped device
      - Controlled by MEMRb and MEMWb
    - CHIPSELb high: I/O space device
      - Expected to do their own address decoding
      - Default to I/O address 0x00300
      - Controlled by IORb and IOWb
  - Address remapping in hardware →
    - CPU address 0x8300 to I/O address 0x0300



Ethernet (VIII)

- Serial EEPROM
  - Used to store CS8900A configuration information and Physical Ethernet address
  - Optional: The host processor can store this data elsewhere in the system.
  - Standard SPI interface →



- Unused pins
  - Should be tied inactive (tied to Vcc/ground)

7.8 Ethernet Protocol

- Goal of Ethernet
  - Reliable communication over an unreliable medium.
- Ethernet technology
  - Wire: thick coaxial, thin coaxial, twisted pair, optical fiber.
  - Connector: BNC for thinnet, AUI connector for thicknet, RJ45 for 10Base-T. [Wikipedia]



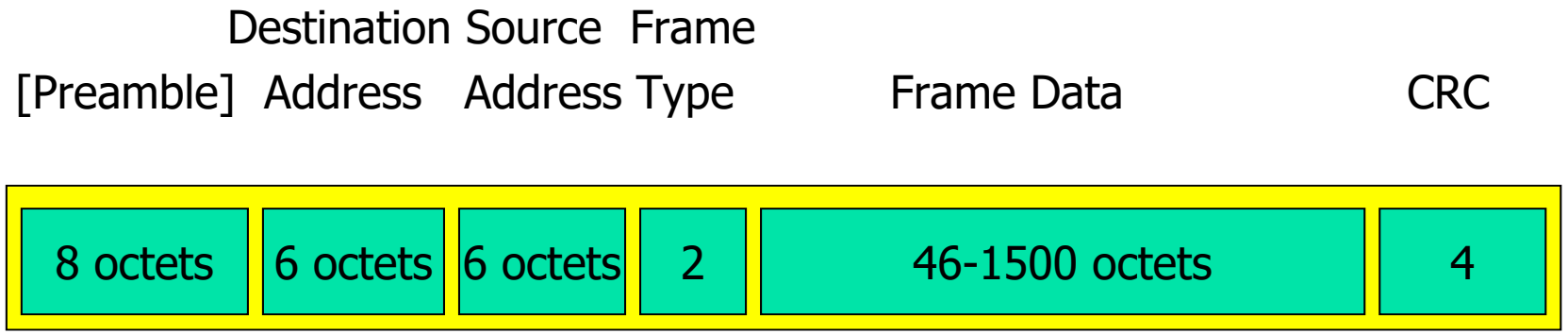
- Ethernet Properties:
  - Broadcast bus technology: all tranceivers receive every transmission.
  - Best effort delivery: message may be lost.
  - Distributed access control: Carrier Sense Multiple Access with Collision Detect (CSMA/CD).
  - Collision: binary exponential back-off policy.

A. Physical Layer of Ethernet

- CSMA/CD (Carrier Sense Multiple Access with Collision Detection):
  - sense collisions;
  - exponentially back off in time;
  - retransmit.
- Ethernet performance
  - Quality-of-service tends to non-linearly decrease at high load levels.
  - Can't guarantee real-time deadlines.
  - However, may provide very good service at proper load levels.

B. Data Link Layer of Ethernet

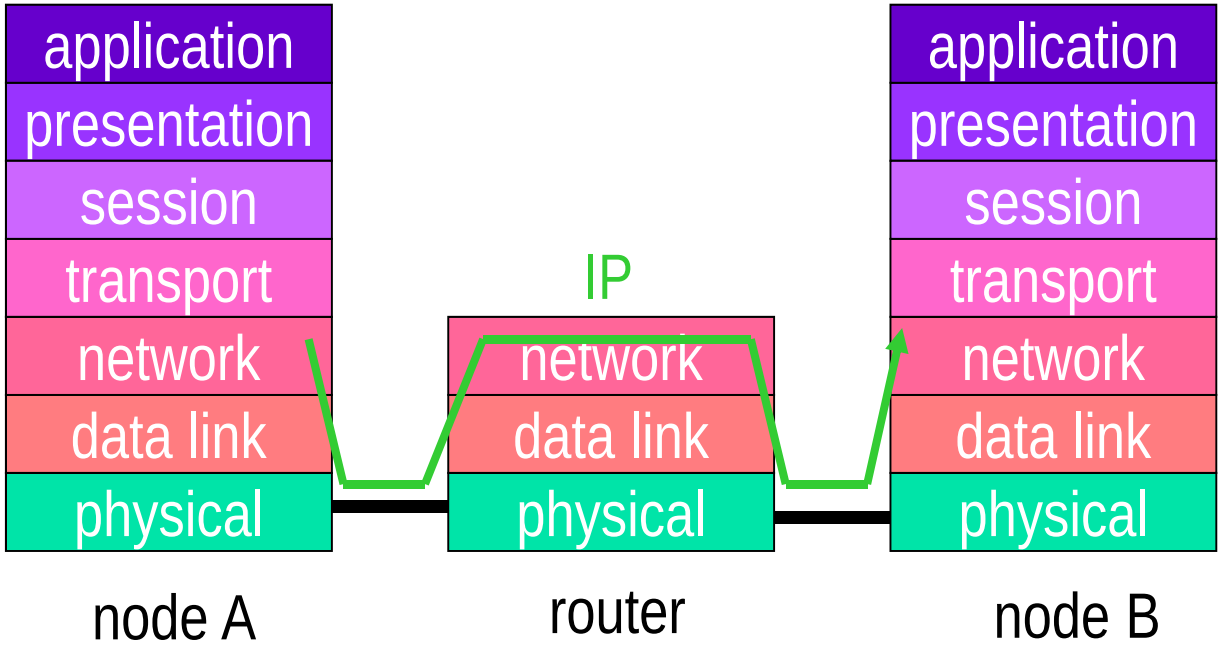
- **Ethernet Frames (Packets) Format**
  - Variable length: min 64 octets (bytes), max 1518 octets, excluding preamble.



  - Frame Type:
    - 0x806 ARP/RARP
    - 0x800 IP
    - Or Data length 0x0 – 0x600

C. Network Layer of Ethernet

- Internet Protocol
  - Internet Protocol (IP) is basis for Internet.
  - Provides an internetworking standard: between two Ethernets, Ethernet and token ring, etc.
  - Higher-level services are built on top of IP.



IP Packet

- IP packet includes:
  - version, service type, length
  - time to live, protocol
  - source and destination address
  - data payload (65,535 bytes max.)
- IP routing
  - Best effort routing:
    - doesn't guarantee data delivery at IP layer.
  - Routing can vary:
    - session to session;
    - packet to packet.

IP Address

- **IP (Internet Protocol) v4 address**
  - A connection to a network. Not a host.
  - 32 bit for IPv4. 128 bit for IPv6.
  - Dotted decimal notation: 143.248.150.1
  - Names (foo.baz.com) translated to IP address by domain name server (DNS).

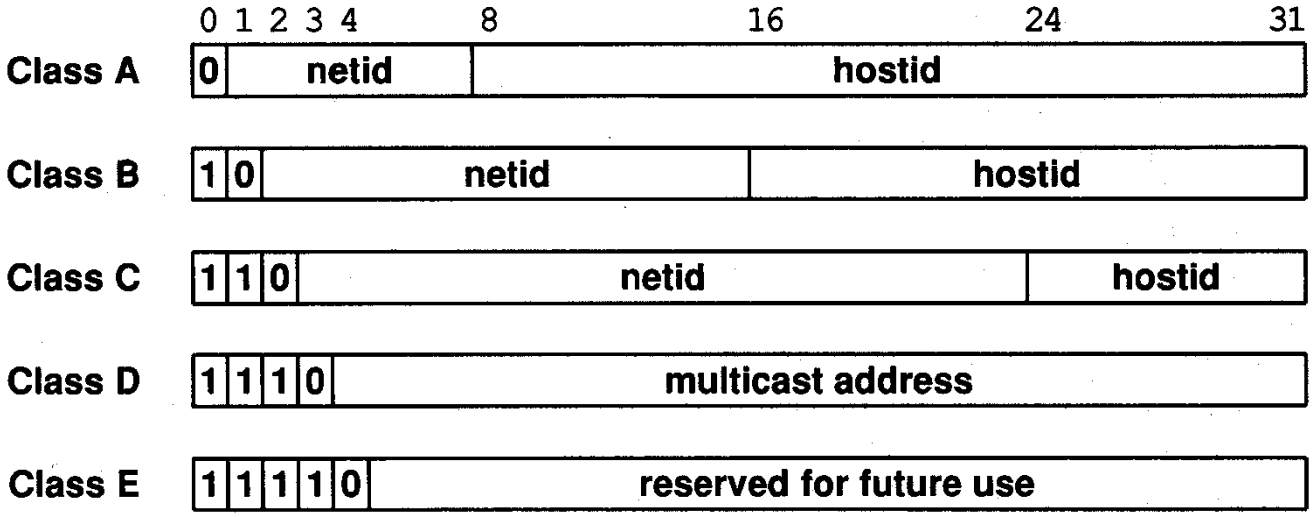


Figure 4.1 The five forms of Internet (IP) addresses. The three primary forms, classes *A*, *B* and *C*, can be distinguished by the first three bits.

Ethernet Address Resolution

- **Address Resolution Protocol (ARP)**
  - Allow a host to find the physical address of a target host, given only the target's IP address.
    - Hardware type: 1 for Ethernet. Protocol type: 0x0800
    - HLEN 6 (Hardware Address Length); PLEN 4 Protocol Address Length)
    - Operation: 1 ARP request, 2 ARP reply, 3 RARP request, 4 RARP reply.
    - Sender supplies Sender Hardware Address, Sender IP, and Target IP.
    - ARP/RARP message format:

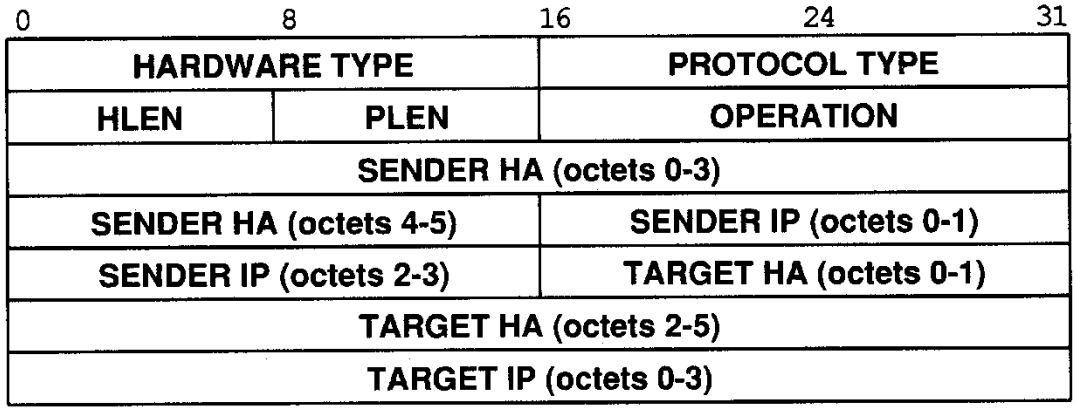


Figure 5.3 An example of the ARP/RARP message format when used for IP-to-Ethernet address resolution. The length of fields depends on the hardware and protocol address lengths, which are 6 octets for an Ethernet address and 4 octets for an IP address.

IP Datagram

- **IP (Internet Protocol) Datagram Format**
  - VERS: 4 (IPv4).
  - HLEN: Header length in 32-bit words (min 5).
  - Type of Service: Precedence (3 bits: 0 to 7), D (low delay), T (high throughput), R(high reliability), 2 unused bits.
  - TOTAL LENGTH: Length of IP datagram in bytes (65536 max).

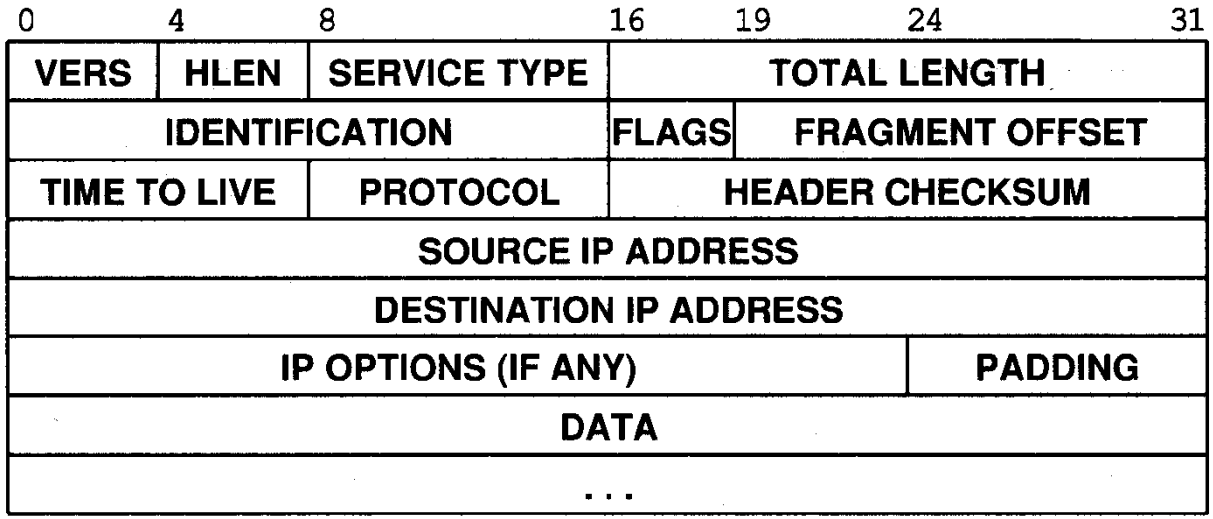
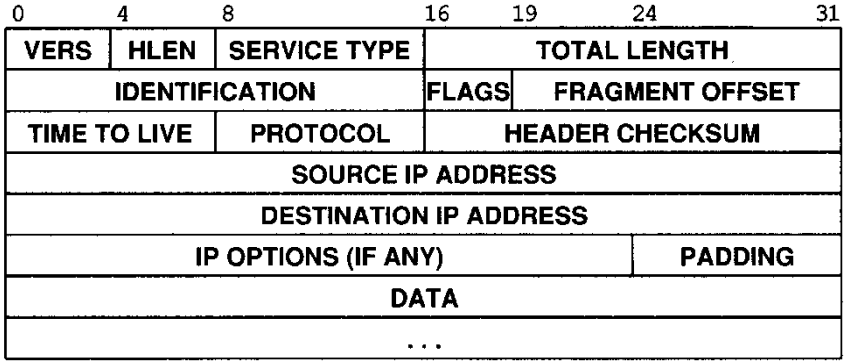


Figure 7.3 Format of an Internet datagram, the basic unit of transfer in a TCP/IP internet.

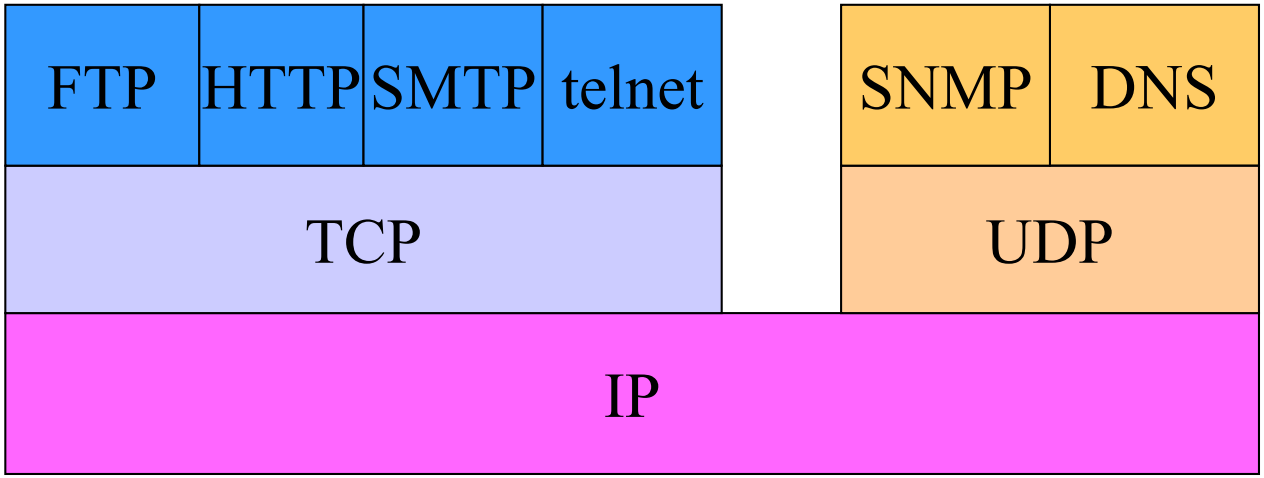
IP Datagram (II)



- *IP Datagram Format (Cont'd)*
  - Identification: Identification number
  - FLAGS:
    - Bit 1: Do not fragment
    - Bit3: More fragments
  - Fragment Offset: Offset for fragmented data.
  - Protocol: Specifies high-level protocol.
    - 0 RAW, 1 ICMP, 2 IGMP, 6 TCP, 8 EGP, 11 UDP, 59 OSPF.
  - Time-to-Live: How long the datagram is allowed to remain (in seconds).
    - Discard the datagram if the TTL field becomes zero.
  - Header Checksum: Checksum of header (not data) as 16-bit integers.
    - Adding using 1's complement arithmetic.
    - One's complement of the sum.

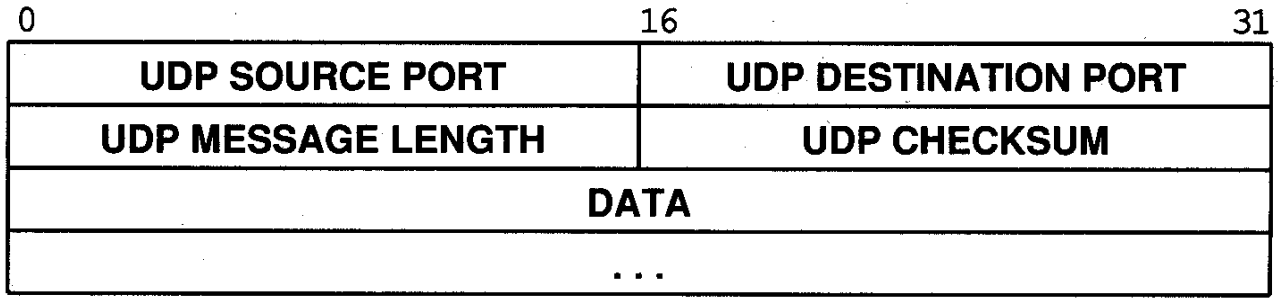
D. Transport Layer of Ethernet

- Higher-level Internet services
  - Transmission Control Protocol (TCP) provides connection-oriented service.
  - User Datagram Protocol (UDP): connectionless delivery
  - Quality-of-service (QoS) guaranteed services are under development.



Transport Layer (II)

- **UDP (User Datagram Protocol) Format**
  - Provide an unreliable connectionless delivery service.
    - Uses IP to carry messages, but adds the ability to distinguish among multiple destinations within a given computer.
  - 4 16-bit fields.
  - Source Port: 16-bit UDP protocol source port number.
  - Destination Port: 16-bit UDP protocol destination port number.
  - UDP Message Length: Count of octets in the UDP datagram including header and data.
  - UDP checksum: Optional (0: checksum not computed).
    - Only way to guarantee that data has arrived (No IP data checksum).



Transport Layer (III)

- **TCP (Transmission Control Protocol) Format**
  - Provide stream oriented, connection oriented, buffered data transfer with acknowledge, time-out, and retransmission.

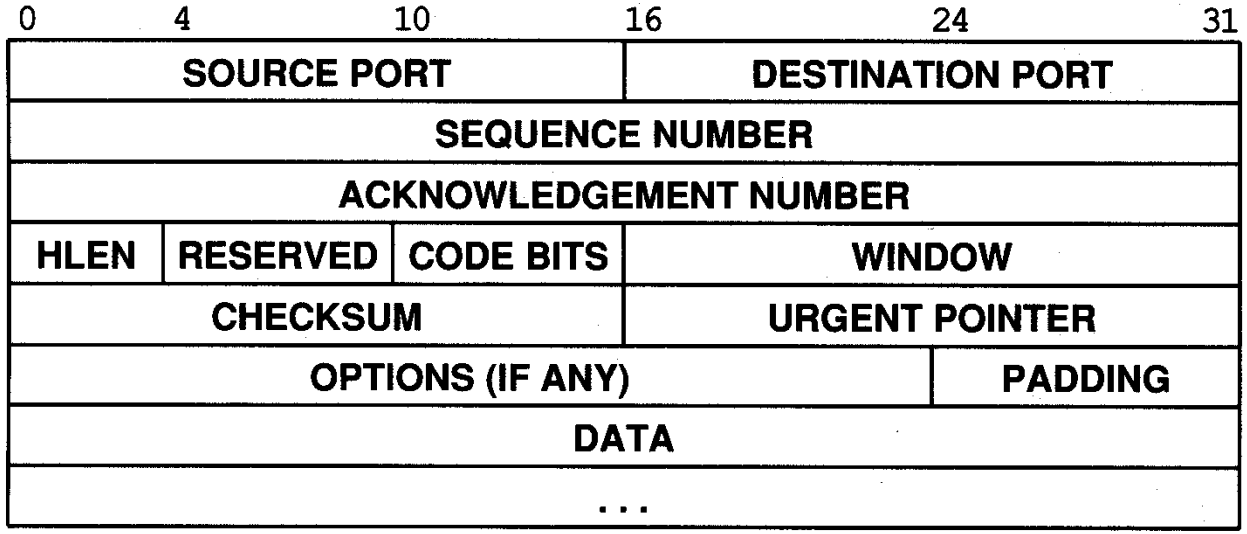
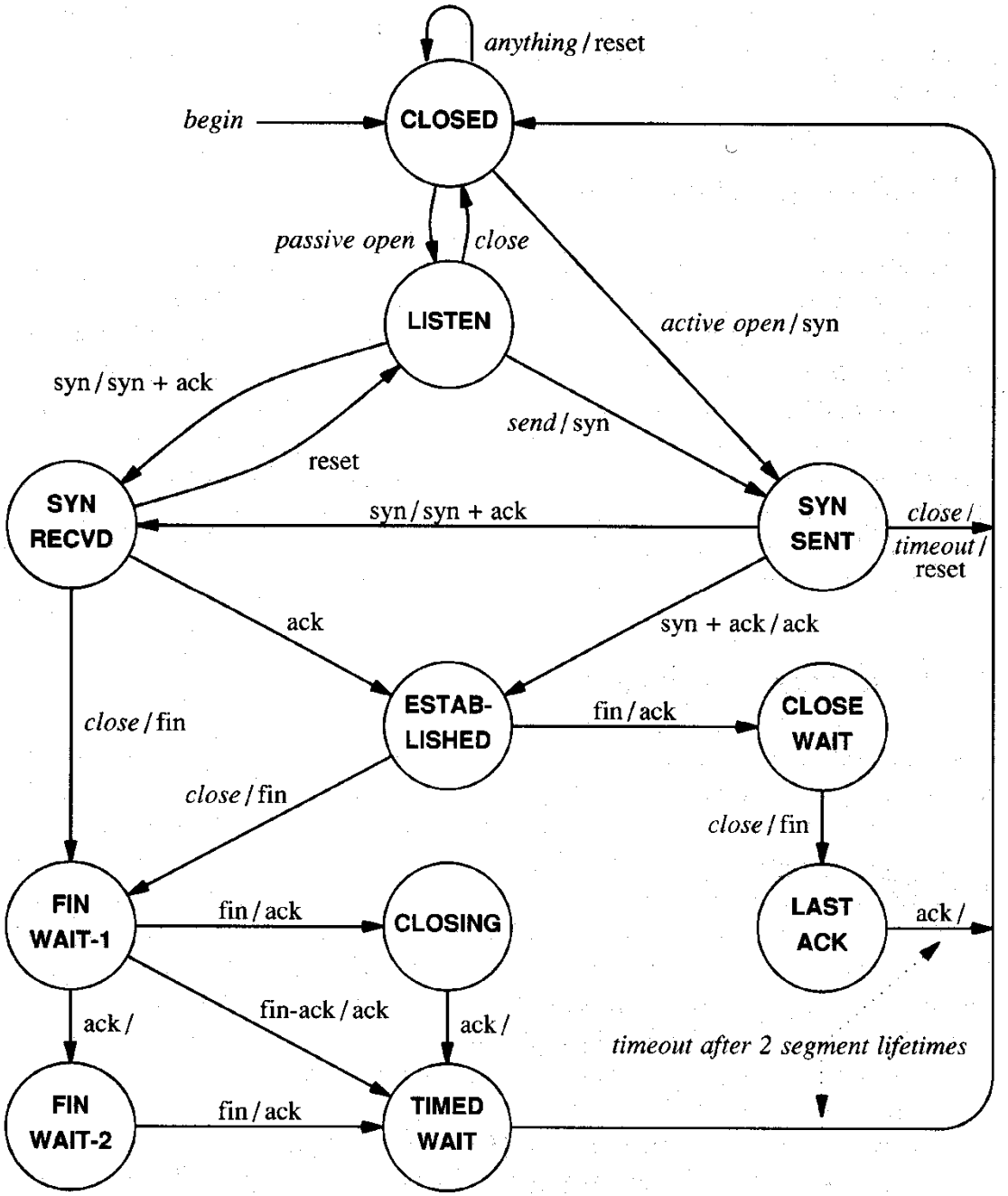


Figure 13.7 The format of a TCP segment with a TCP header followed by data. Segments are used to establish connections as well as to carry data and acknowledgements.

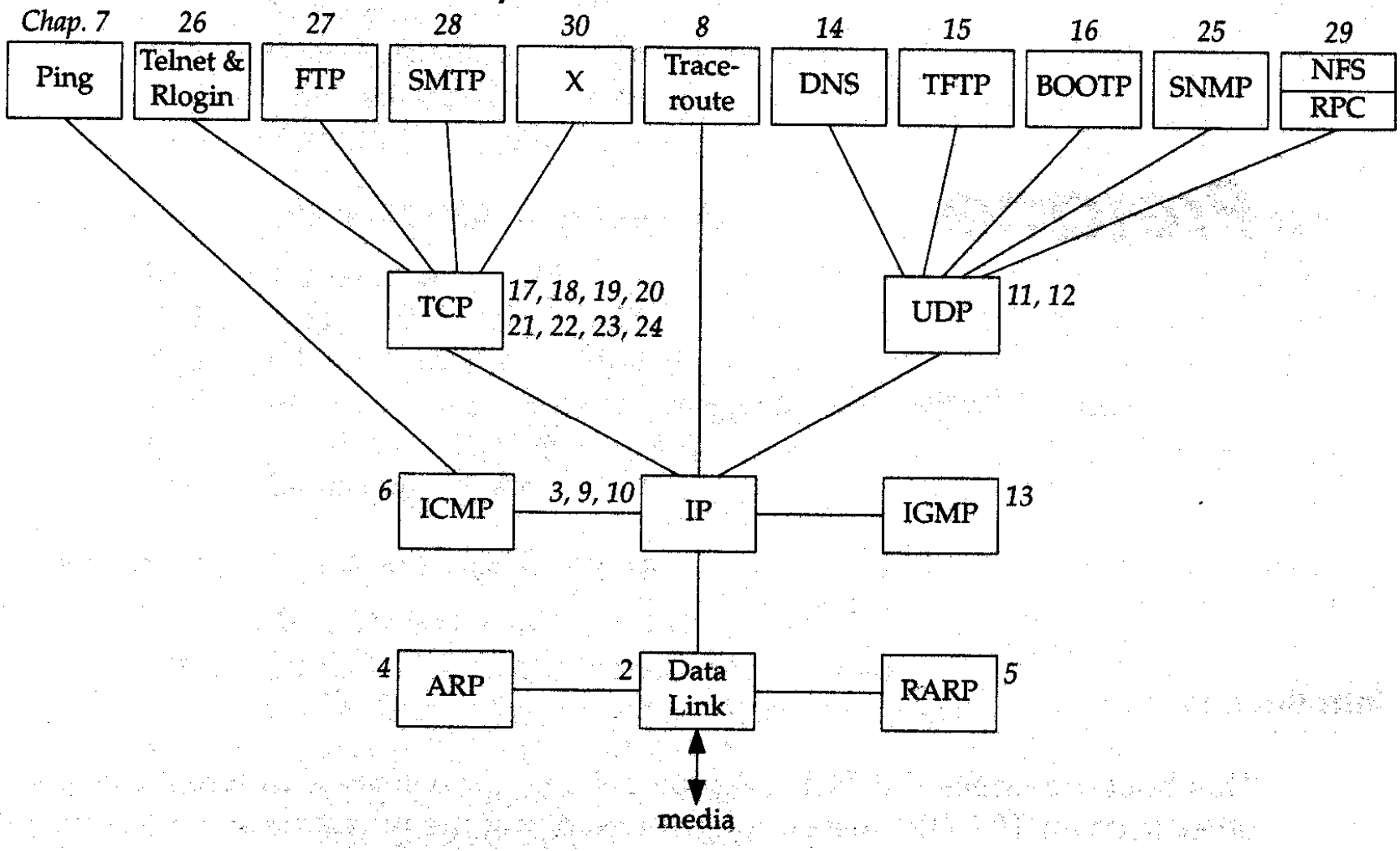
Transport Layer (IV)

- TCP Finite State Machine



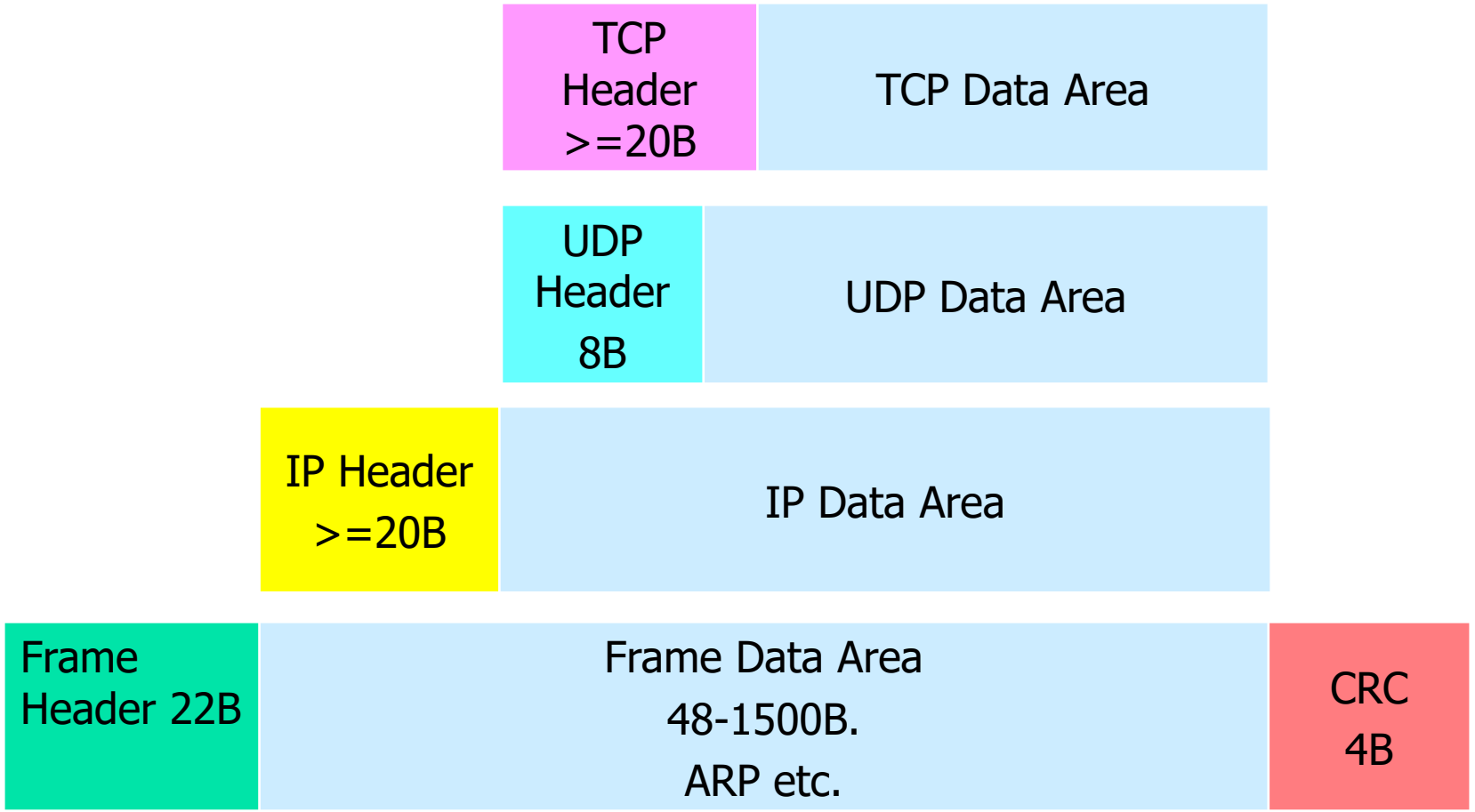
E. Ethernet

- Protocol Hierarchy



Ethernet (II)

- Data Encapsulation



Ethernet (III)

- ISO 7-Layer Reference Model

Layer	ISO-7	TCP/IP
7	Application	Application
6	Presentation	Transport
5	Session	
4	Transport	
3	Network	Internet
2	Data Link	Network Interface
1	Physical Hardware Connection	Hardware

Message or streams
Transport Protocol Packets
IP Datagrams
Network-Specific Frames

7.9 Socket Programming

- **What is a socket?**
  - A way to speak to other programs using standard Unix file descriptors.
    - A file descriptor is simply an integer associated with an open file.
    - But that file can be a network connection, a FIFO, a pipe, a terminal, a real on-the-disk file, or just about anything else.
  - Make a call to the socket() system routine.
    - Returns the socket descriptor, and you communicate through it using the specialized send() and recv() socket calls.
    - You can use normal read() and write(), but send() and recv() offer much greater control over your data transmission.

Socket Programming (II)

- **Two types of internet sockets**
  - **Stream sockets**
    - Reliable two-way connected communication streams.
      - Ordered
      - Error-free
    - **telnet** uses stream sockets.
      - All the characters you type need to arrive in the same order you type them,
    - Web browsers use the HTTP protocol which uses stream sockets to get pages.
    - Use a protocol called "TCP" (see RFC-793 for extremely detailed info on TCP.)
      - TCP makes sure your data arrives sequentially and error-free.

Socket Progrmming (III)

- *Two types of internet sockets (Cont'd)*
  - **Datagram sockets**
    - Connectionless
      - if you send a datagram, it may arrive. It may arrive out of order. If it arrives, the data within the packet will be error-free.
    - Use the "UDP" (see RFC-768.)
    - You don't have to maintain an open connection as you do with stream sockets.
      - You just build a packet, slap an IP header on it with destination information, and send it out.
      - No connection needed.
      - They are generally used for packet-by-packet transfers of information.
    - Sample applications: **tftp**, **bootp**, etc.

Socket Programming (IV)

- **Structs**
  - 1. a socket descriptor.
    - int (Just a regular int).
  - 2. Struct sockaddr.
    - Holds socket address information for many types of sockets:
    - struct sockaddr {

```
unsigned short      sa_family;     // address family, AF_xxx
char                sa_data[14]; // 14 bytes of protocol address
};
```

    - *sa_family* can be a variety of things, but it'll be AF_INET.
    - *sa_data* contains a destination address and port number for the socket.
  - 3. A parallel structure: struct sockaddr_in ("in" for "Internet".)
    - struct sockaddr_in {

```
short               int sin_family; // Address family
unsigned short int sin_port;              // Port number
struct in_addr      sin_addr;       // Internet address unsigned
char                sin_zero[8];    // Same size as struct sockaddr
};
```

    - This structure makes it easy to reference elements of the socket address.
  - 4. Internet address (a structure for historical reasons)
    - struct in_addr { unsigned long s_addr; // that's a 32-bit long, or 4 bytes };

Socket Programming (V)

- **System Calls**
  - socket()--Get the File Descriptor!
  - bind()--What port am I on?
  - connect()--Hey, you!
  - listen()--Will somebody please call me?
  - accept()--"Thank you for calling port 3490."
  - send() and recv()--Talk to me, baby!
  - sendto() and recvfrom()--Talk to me, DGRAM-style
  - close() and shutdown()--Get outta my face!
  - getpeername()--Who are you?
  - gethostname()--Who am I?

Socket Programming (VI)

- A simple stream server
  - Open stream socket
  - Set socket option
  - Bind socket_id to socket structure
  - Listen
  - Loop
    - Accept incoming connection request ←
    - Child process:
      - Send a message to the socket →

- A simple stream client
  - Get the host information
  - Open stream socket
  - ← Connect to the server
  - → Receive a message from the socket
  - Print it
  - Close the socket

References

- Ethernet Protocol & Socket programming
  - Douglas Comer, “Internetworking with TCP/IP. Volume 1: Principles, Protocols, and Architecture”, Third Ed., Prentice Hall, 1995.
  - Search Internet.