Embedded Systems

Ch 11B
Network Interface

Byung Kook Kim
Dept of EECS
Korea Advanced Institute of Science and Technology

Overview

- *1. Introduction*
- *2. RS-485*
- *3. Controller Area Network (CAN)*
- *4. Ethernet*
- *5. CAN Protocol*
- *6. Ethernet Protocol*
- *7. Socket Programming*

5. CAN Protocol

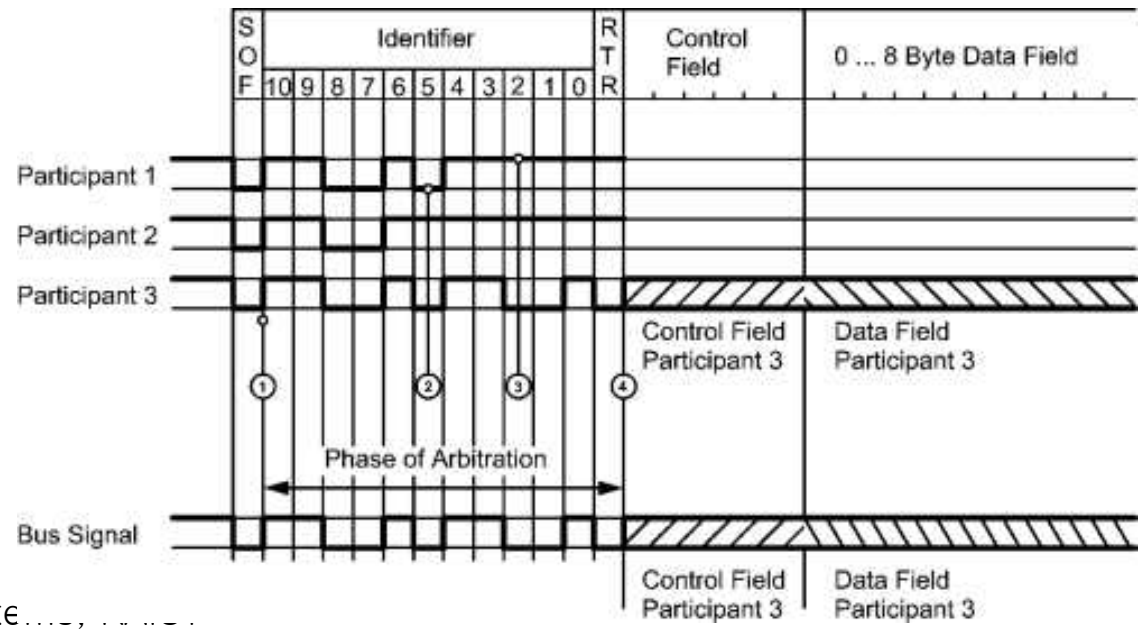
- **Controller Area Network (CAN)**
  - A small broadcast network
  - Local bus topology with outputs of all stations are wire-ANDed by the bus.
  - Message with 11 (or 29) bit identifier and 1 – 8 bytes of data.
  - Two-wire, half duplex, high-speed network system and is well suited for high speed applications using short messages.
  - Its robustness, reliability and the large following from the semiconductor industry are some of the benefits with CAN.
  - CAN can theoretically link up to 2032 devices (assuming one node with one identifier) on a single network.
    - Practical limitation of the hardware (transceivers), it can only link up to 110 nodes (with 82C250, Philips) on a single network.
  - High-speed communication rate up to 1 Mbits/sec thus allows real-time control.
  - Error confinement and the error detection feature: reliable in noise critical environment.

CAN Protocol (II)

- **CAN networking**
  - *CAN is a multimaster network using CSMA/CD+AMP (Carrier Sense Multiple Access/Collision Detection with Arbitration on Message Priority).*
    - Before sending a message the CAN node checks if the bus is busy.
    - It also uses collision detection.
    - Data messages transmitted from any node on a CAN bus do not contain addresses of either the transmitting node, or of any intended receiving node.
  - *The content of the message is labeled by an identifier that is unique throughout the network.*
    - All other nodes on the network receive the message and each performs an acceptance test on the identifier to determine if the message, and thus its content, is relevant to that particular node.
    - If the message is relevant, it will be processed; otherwise it is ignored.

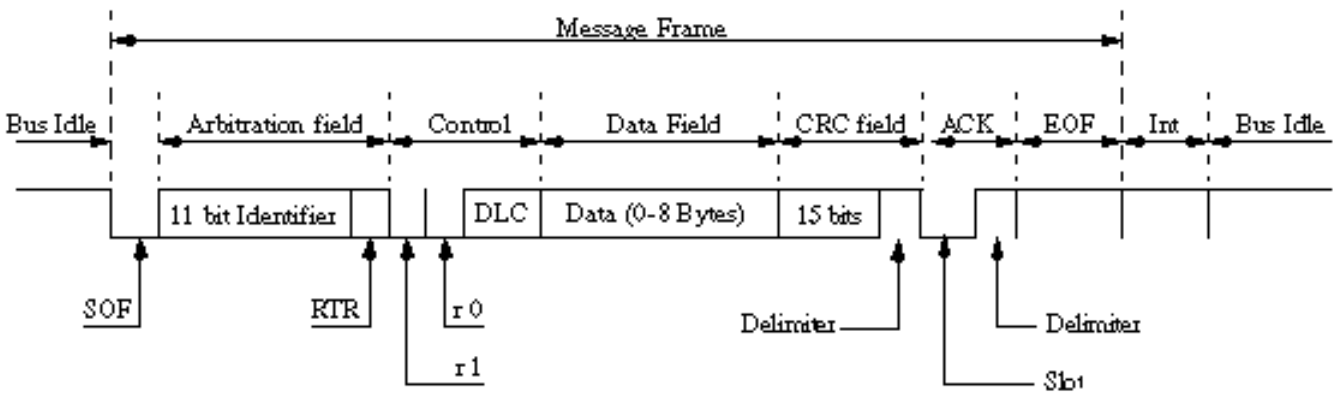
CAN Protocol (III)

- **Non-destructive contention-based bus arbitration *(CAN 1.0)***
  - *All nodes are allowed to start the transmission of a frame after the bus is idle.*
  - More than one node are starting transmission at the same time.
  - Each node monitors the bus during transmission of the identifier field and the RTR bit.
  - As soon as it detects a dominant bit while transmitting a recessive bit it releases the bus, immediately stops transmission and continues receiving the frame.

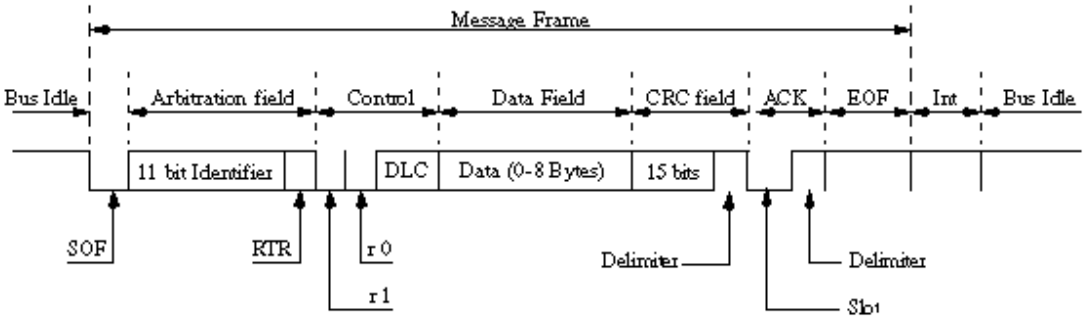


CAN Protocol (IV)

- **The format of a standard CAN 2.0A message: 7 bit fields**
  - A Start of Frame (SOF) field. This is a dominant (logic 0) bit that indicates the beginning of a message frame.
  - An Arbitration field:
    - An 11 bit message identifier
    - The Remote Transmission Request (RTR) bit.
      - A dominant (logic 0), RTR bit indicates that the message is a Data Frame.
      - A recessive (logic 1) value indicates that the message is a Remote Transmission Request (otherwise known as Remote Frame).
      - A Remote Frame is a request by one node for data from some other node on the bus. Remote Frames do not contain a Data Field. The Data Length Code specifies the number of bytes of data in the requested Message Frame.



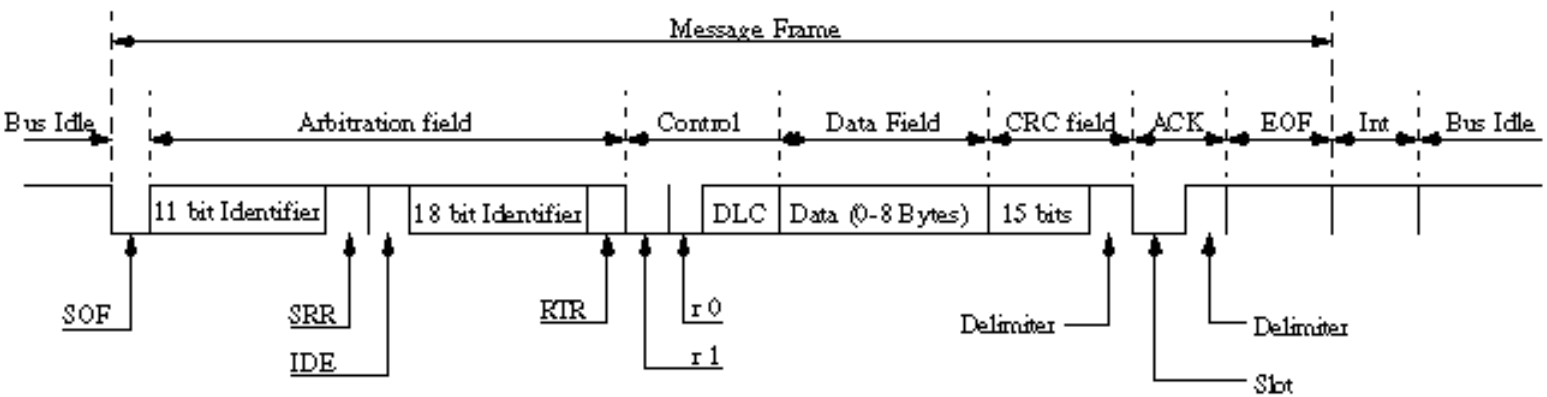
CAN Protocol (V)



- **The format of a standard CAN 2.0A message (II)**
  - A Control Field containing six bits:
    - Two dominant bits (r0 and r1) that are reserved for future use, and
    - A four bit Data Length Code (DLC). The DLC indicates the number of bytes in the Data Field that follows
  - A Data Field, containing from zero to eight bytes.
  - The CRC field, containing a fifteen bit cyclic redundancy check code and a recessive delimiter bit
  - The ACKnowledge field, consisting of two bits.
    - The first is the Slot bit which is transmitted as a recessive bit, but is subsequently over written by dominant bits transmitted from all other nodes that successfully receive the message.
    - The second bit is a recessive delimiter bit.
  - The End of Frame field, consisting of seven recessive bits.
  - The INTermission field consisting of three recessive bits.
    - After the three bit INTermission period the bus is recognized to be free.
    - Bus Idle time may be of any arbitrary length including zero.

CAN Protocol (VI)

- **The CAN 2.0B format**
  - In Version 2.0B the Arbitration field contains two identifier bit fields.
    - The first (the base ID) is eleven (11) bits long for compatibility with Version 2.0A.
    - The second field (the ID extension) is eighteen (18) bits long, to give a total length of twenty nine (29) bits.
    - The distinction between the two formats is made using an Identifier Extension (IDE) bit.
  - A Substitute Remote Request (SRR) bit is included in the Arbitration Field.
    - The SRR bit is always transmitted as a recessive bit: Ensure that, in the case of arbitration between a Standard Data Frame and an Extended Data Frame, the Standard Data Frame will always have priority if both messages have the same base (11 bit) identifier.



CAN Protocol (VII)

- **Sophisticated error detection and error confinement mechanisms**
  - Very sophisticated error detection and error confinement mechanisms: resulting in a low residual probability of not detected errors.
  - Monitoring of the transmitted bit level by the transmitting node.
    - If the monitored bit level is different from the bit level that is sent, a bit error is detected.
    - With this mechanism all globally effective bus errors are detected.
  - Checking of fixed format elements and by checking of the CRC segment through a receiver.
    - Providing a means for detecting of only locally effective errors with a very high probability.

CAN Protocol (VIII)

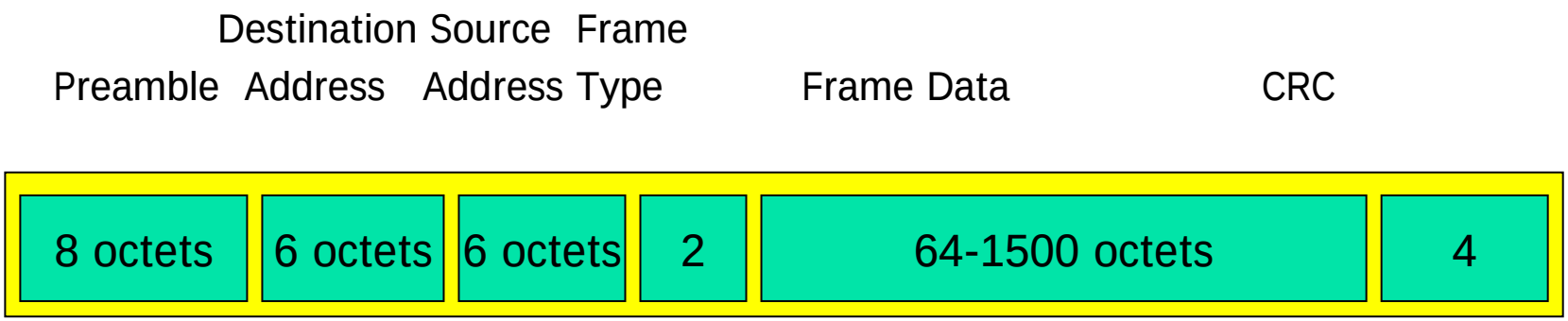
- **Error signaling instead of message confirmation provides system-wide data consistency and low error recovery times**
  - Only a corrupted message is signaled by means of an error frame.
  - An error frame is transmitted as soon as an error condition is detected by a transmitting or receiving node: The transmitted message is destroyed and so network-wide data consistency provided.
  - If a transmitting node sends or receives an error frame it automatically starts retransmission of the corrupted message: Provides a very short error recovery time which is much lower than that of competing protocols.
  - Counter for counting of transmission and receiving errors and for performing a sophisticated error management.

6. Ethernet Protocol

- **Ethernet technology**
  - Wire: thick coaxial, thin coaxial, twisted pair, optical fiber.
  - Connector: BNC for thinnet, AUI connector for thicknet, RJ45 for 10Base-T.
  - 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.

Ethernet Protocol (II)

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



  - Frame Type:
    - 0x806 ARP/RARP
    - 0x800 IP

Ethernet Protocol (III)

- **IP (Internet Protocol) address**
  - A connection to a network. Not a host.
  - 32 bit for IPv4. 128 bit for IPv6.
  - Dotted decimal notation: 143.248.150.1

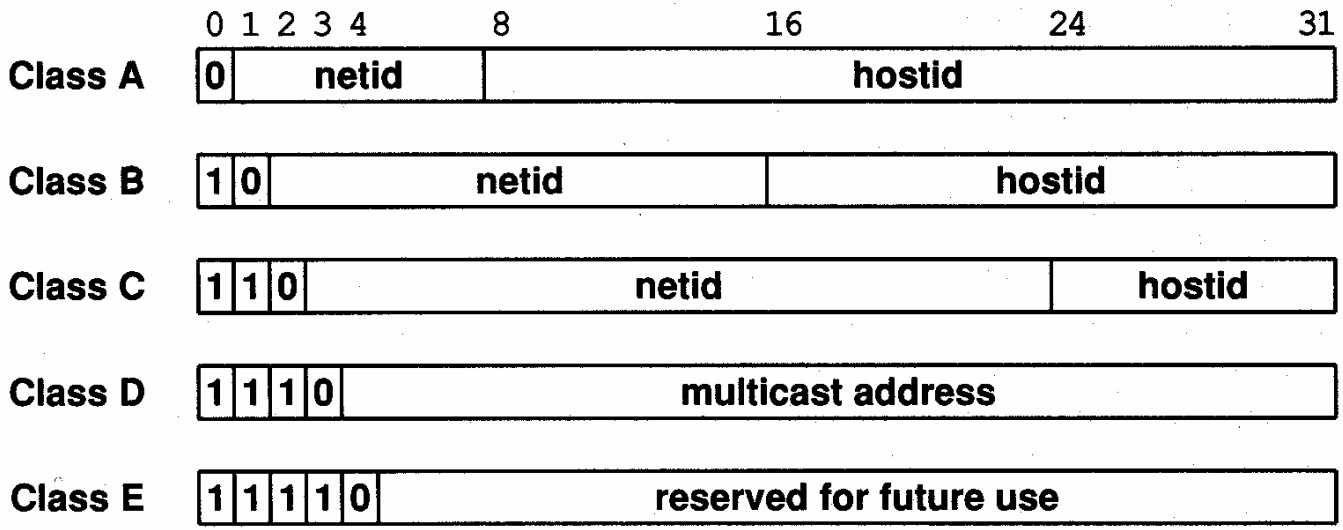


Figure 4.1 The five forms of Internet (IP) addresses. The three primary forms, classes *A*, *B* and *C*, can be distinguished by the first three bits.

Ethernet Protocol (IV)

- **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:

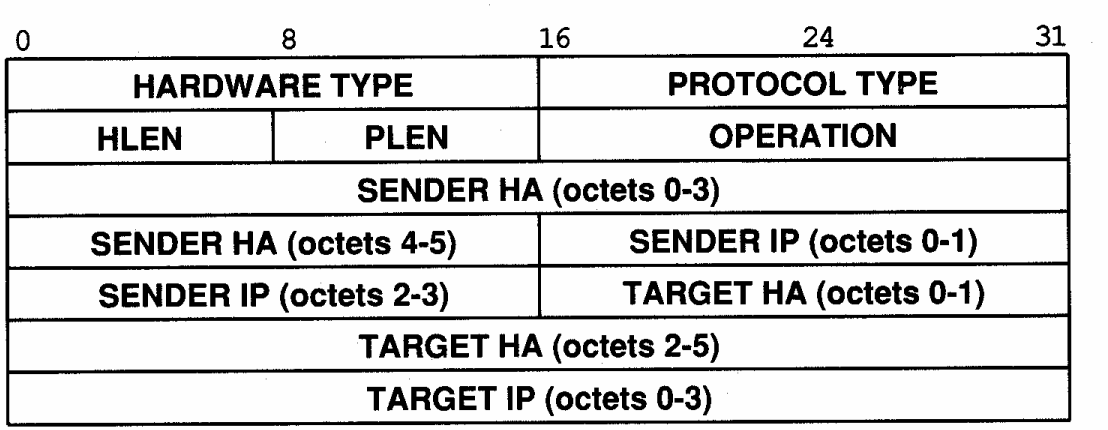


Figure 5.3 An example of the ARP/RARP message format when used for IP-to-Ethernet address resolution. The length of fields depends on the hardware and protocol address lengths, which are 6 octets for an Ethernet address and 4 octets for an IP address.

Ethernet Protocol (V)

- **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).

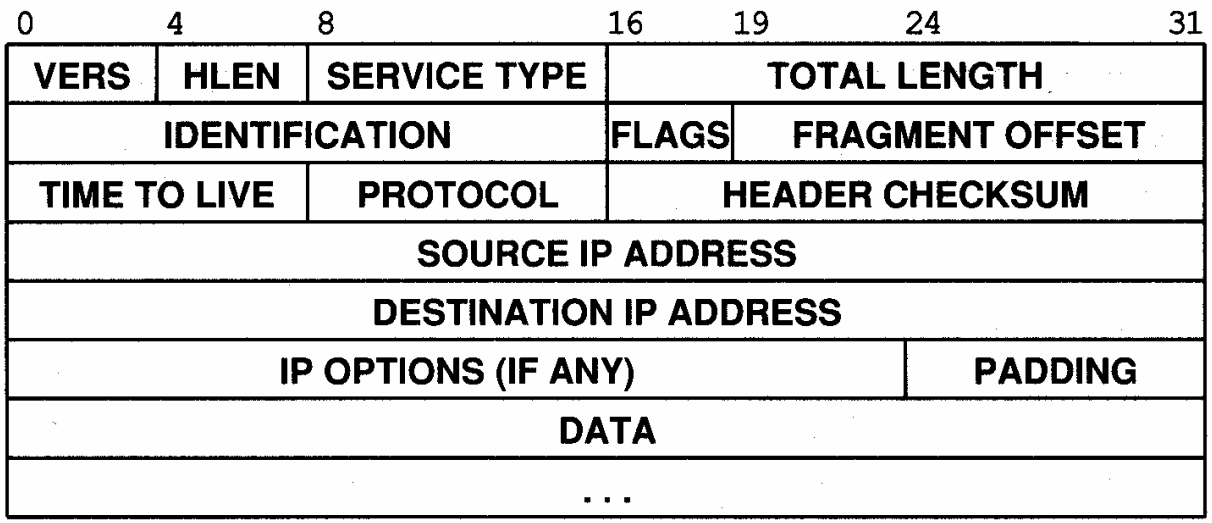
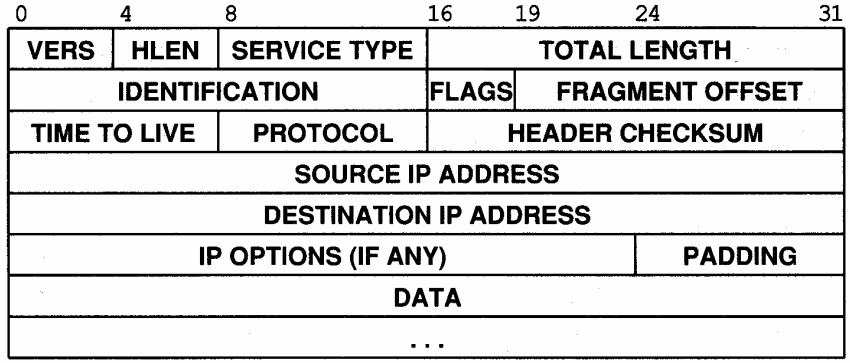


Figure 7.3 Format of an Internet datagram, the basic unit of transfer in a TCP/IP internet.

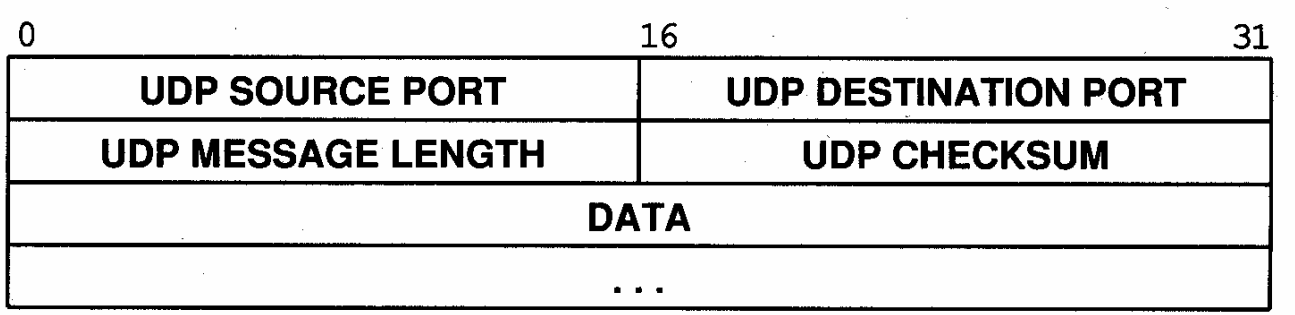
Ethernet Protocol (VI)



- *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.

Ethernet Protocol (VII)

- **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).



Ethernet Protocol (VIII)

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

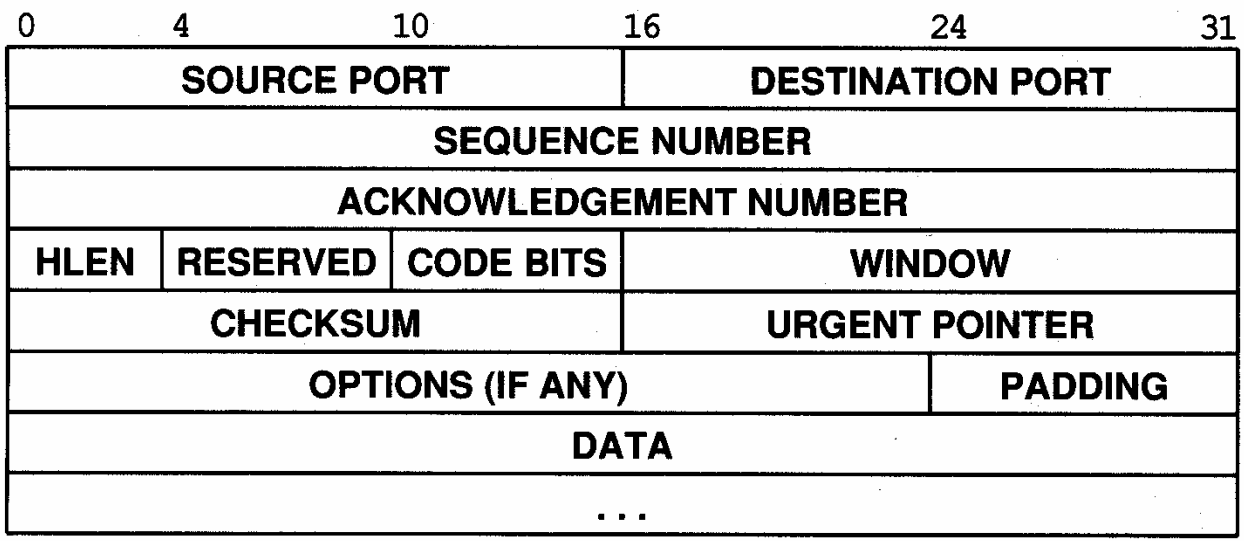
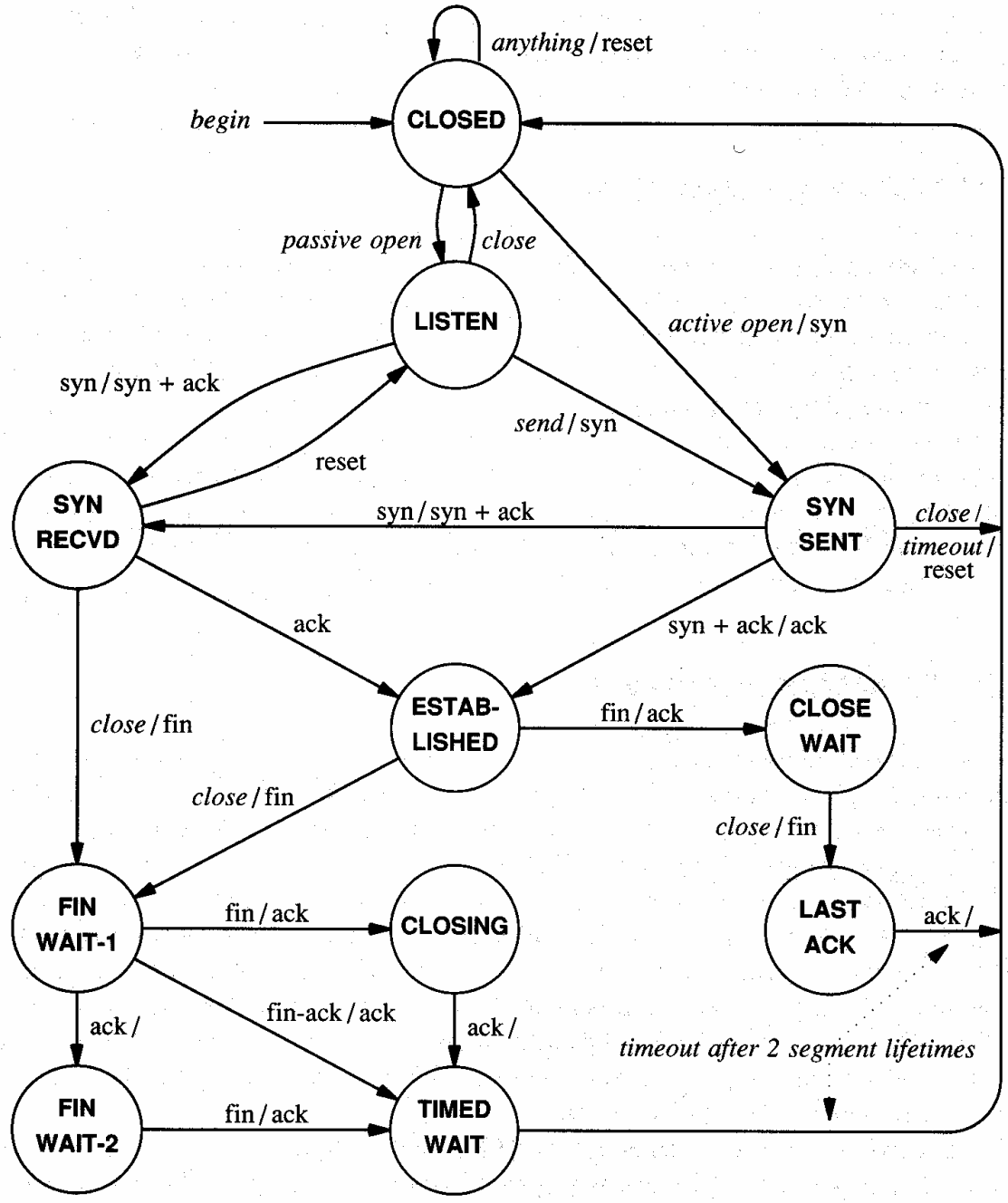


Figure 13.7 The format of a TCP segment with a TCP header followed by data. Segments are used to establish connections as well as to carry data and acknowledgements.

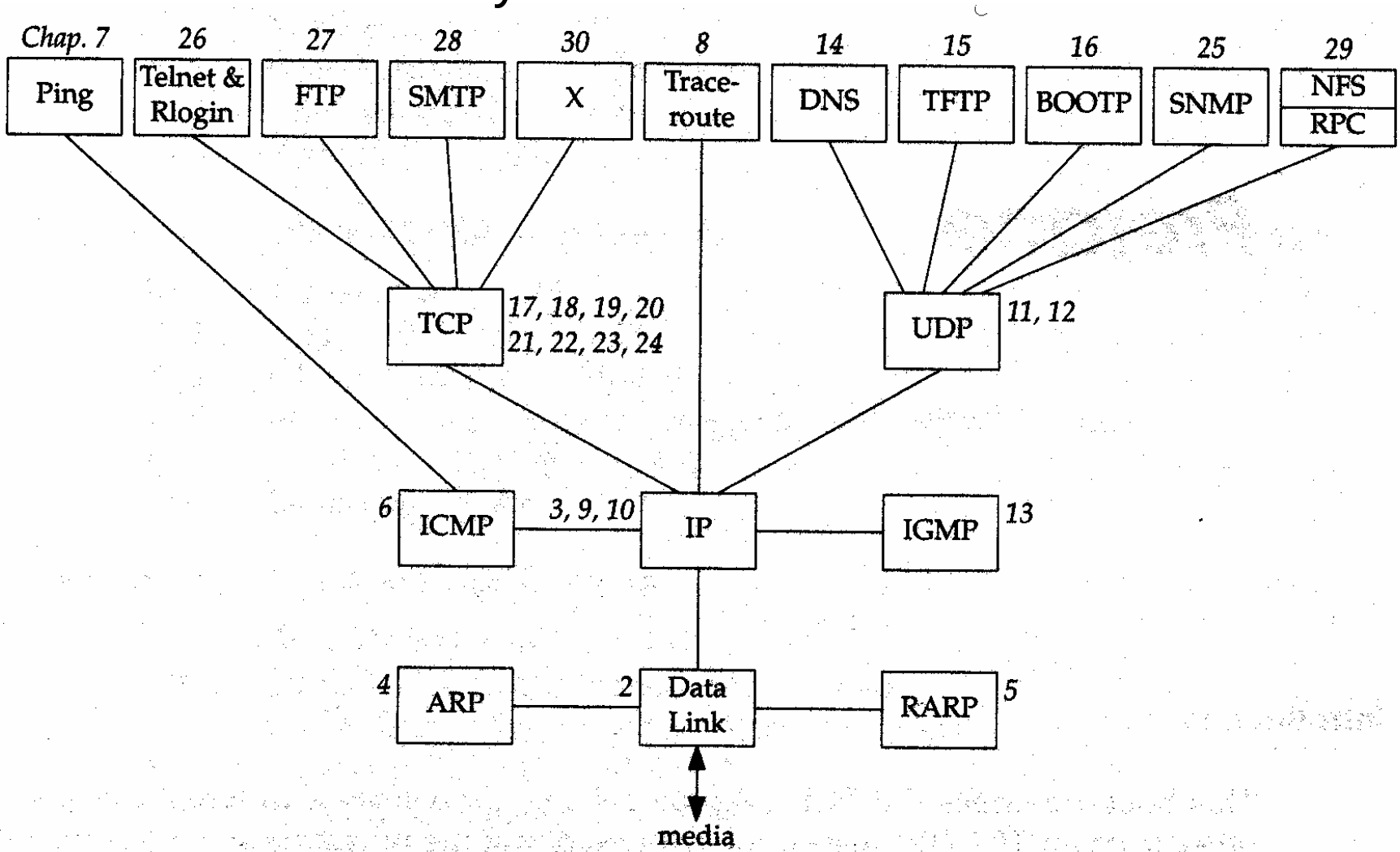
Ethernet Protocol (IX)

- TCP Finite State Machine



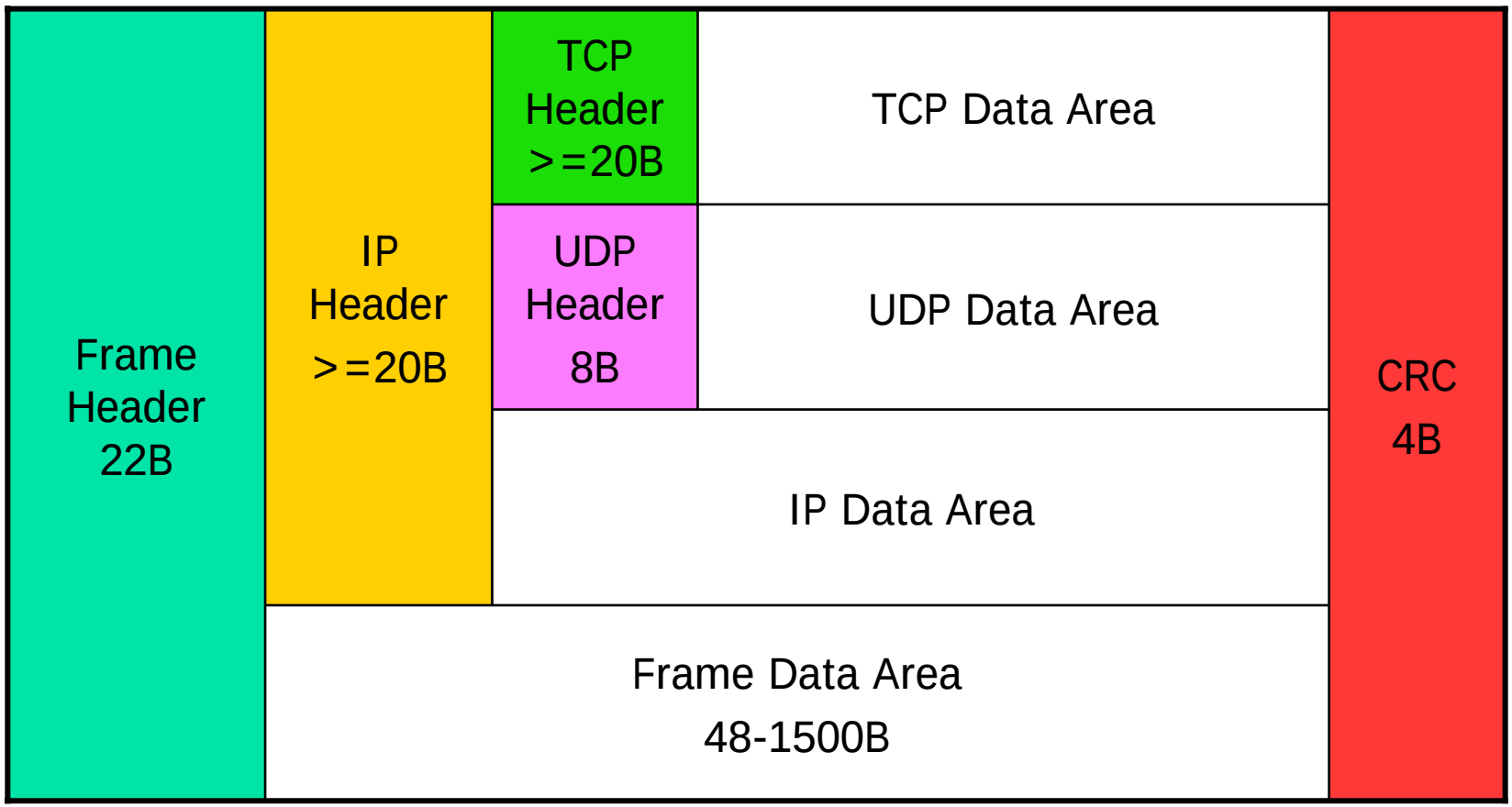
Ethernet Protocol (X)

- Protocol Hierarchy



Ethernet Protocol (XI)

- Data Encapsulation



Ethernet Protocol (XII)

- 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. 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 "The Transmission Control Protocol", otherwise known as "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 "User Datagram Protocol", or "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

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