

Data Link Protocols

- ➡ Is a set of specifications used to implement the data link layer
- ➡ Data link protocols differ by message delineation, frame length, and frame field structure.
- ➡ Another fundamental difference is between asynchronous and synchronous transmission data link protocols.

Asynchronous Protocols

- ➡ In **asynchronous transmission** (sometimes called start-stop transmission), each character is sent independently.
- ➡ The transmission sequence begins with
 - ↪ a start bit
 - ↪ next the character is sent
 - ↪ then the parity bit
 - ↪ and finally a stop bit are sent.
 - The **start** bit is usually a 0 and the **stop** bit a 1.
- ➡ Between transmissions (called “idle time”), a series of stop bits are sent.
- ➡ When a new character is sent, the start bit is used by the receiver for **synchronization**.

Asynchronous Protocols

- ➔ Protocols that belong to asynchronous protocols
 - XMODEM
 - YMODEM
 - ZMODEM
 - BLAST
 - Kermit

Synchronous Protocols

- ➔ In synchronous transmission
 - ↳ data is sent in a large block called a frame
- ➔ Synchronous transmission is used on both
 - ↳ point-to-point
 - ↳ multipoint circuits
 - In multipoint circuits, addressing information needs to be included in the frame.
- ➔ Synchronous packets sometimes begin and end with a series of synchronization (SYN) characters that are used to help the receiver recognize incoming data.

Synchronous Protocols

⇒ Synchronous transmission protocols can be:

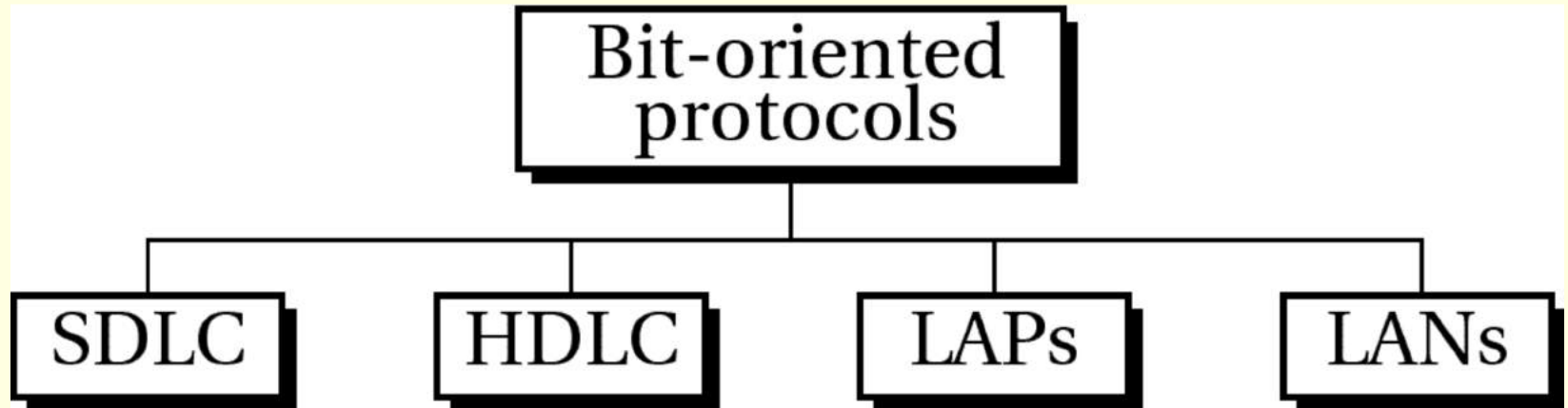
⇒ **character-oriented:**

- Also known as **byte-oriented protocols**
- Interpret a transmission frame as a succession of characters

⇒ **bit-oriented:**

- Interpret a transmission frame as a succession of individual bits
- Control information in a bit-oriented protocol can be one or multiple bits depending on the information embodied in the pattern

Bit-oriented protocols



HDLC : *High-level Data Link Control*

- ❑ It is a bit-oriented data link protocol
- ❑ Designed to support both half duplex and full duplex communication over point-to-point and multipoint links.
- ❑ It implements the ARQ mechanisms.
- ❑ The HDLC protocol embeds information in a data frame that allows devices to control data flow and correct errors

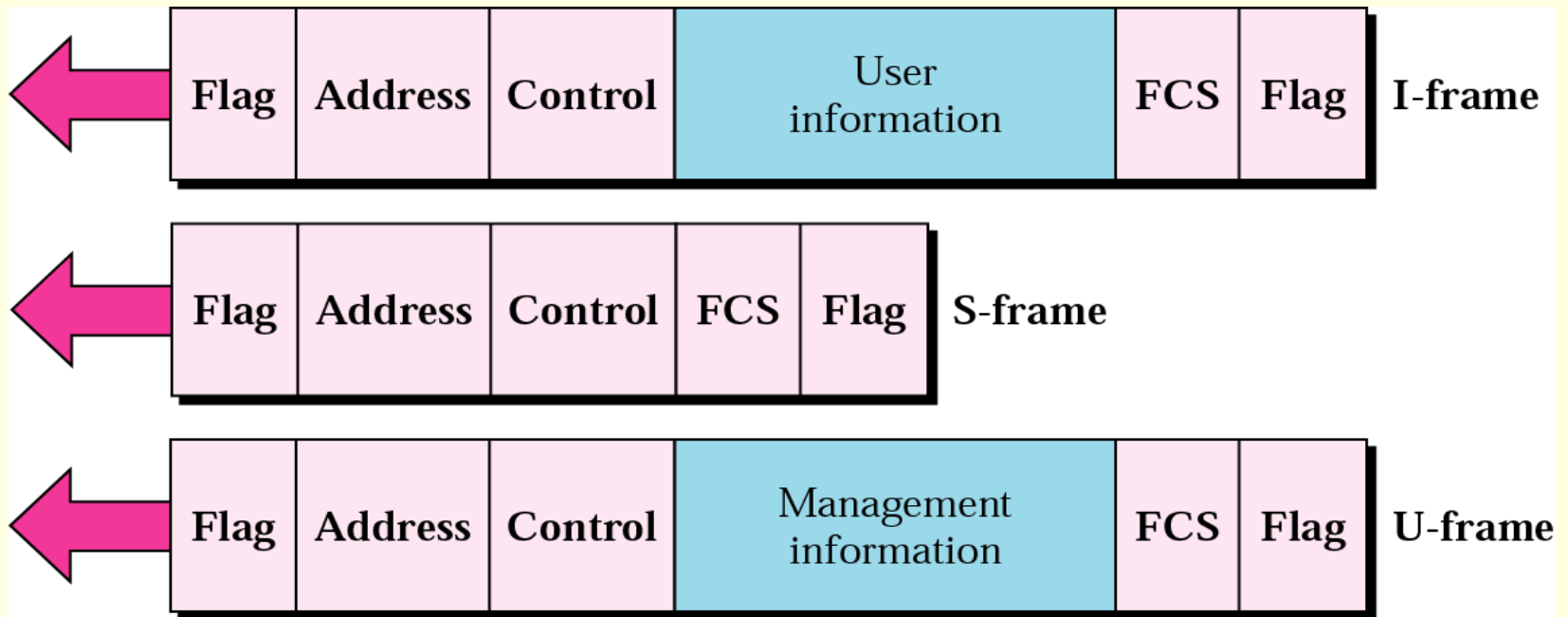
HDLC : *High-level Data Link Control*

- ➡ In 1979, the ISO made HDLC the standard as a Bit-oriented control protocol
- ➡ The HDLC provides a transparent transmission service at the data link layer
- ➡ The users of the HDLC service provides **PDU**s which are encapsulated to form data link layer frames. These frames are separated by HDLC "flags" and are modified by "zero bit insertion" to guarantee transparency

HDLC : *High-level Data Link Control*

- ➔ Each piece of data is encapsulated in an HDLC frame by adding a trailer and a header.
- ➔ **The header** contains an HDLC address and an HDLC control field.
- ➔ **The trailer** is found at the end of the frame, and contains a (CRC) which detects any errors which may occur during transmission.
- ➔ The frames are separated by HDLC flag sequences which are transmitted between each frame and whenever there is no data to be transmitted.

HDLC frame types



HDLC *Frame Fields*

Flag field

- ↗ is 8 bits of a fixed **pattern** (0111 1110).
- ↗ There is one flag at the beginning and one at the end frame.
- ↗ The ending flag of one Frame can be used as the beginning flag of the next frame.
- ↗ To guarantee that the flag does not appear anywhere else in the frame
- ↗ HDLC uses a process called **Bit Stuffing**.
- ↗ Every time a sender wants to transmit a bit sequence having more than 6 consecutive 1's, it inserts 1 redundant 0 after the 5th 1

Exceptions:

- When the **bit sequence** is really a **flag**.
- when **transmission** is being **aborted**.
- when the **channel** is being put into **idle**.

Bit Stuffing

- ➡ the process of adding one extra zero whenever there are 5 consecutive 1's in the data, so that the receiver doesn't mistake the data for a flag.

A frame before bit stuffing:

01111110 01111100 101101111 110010

After

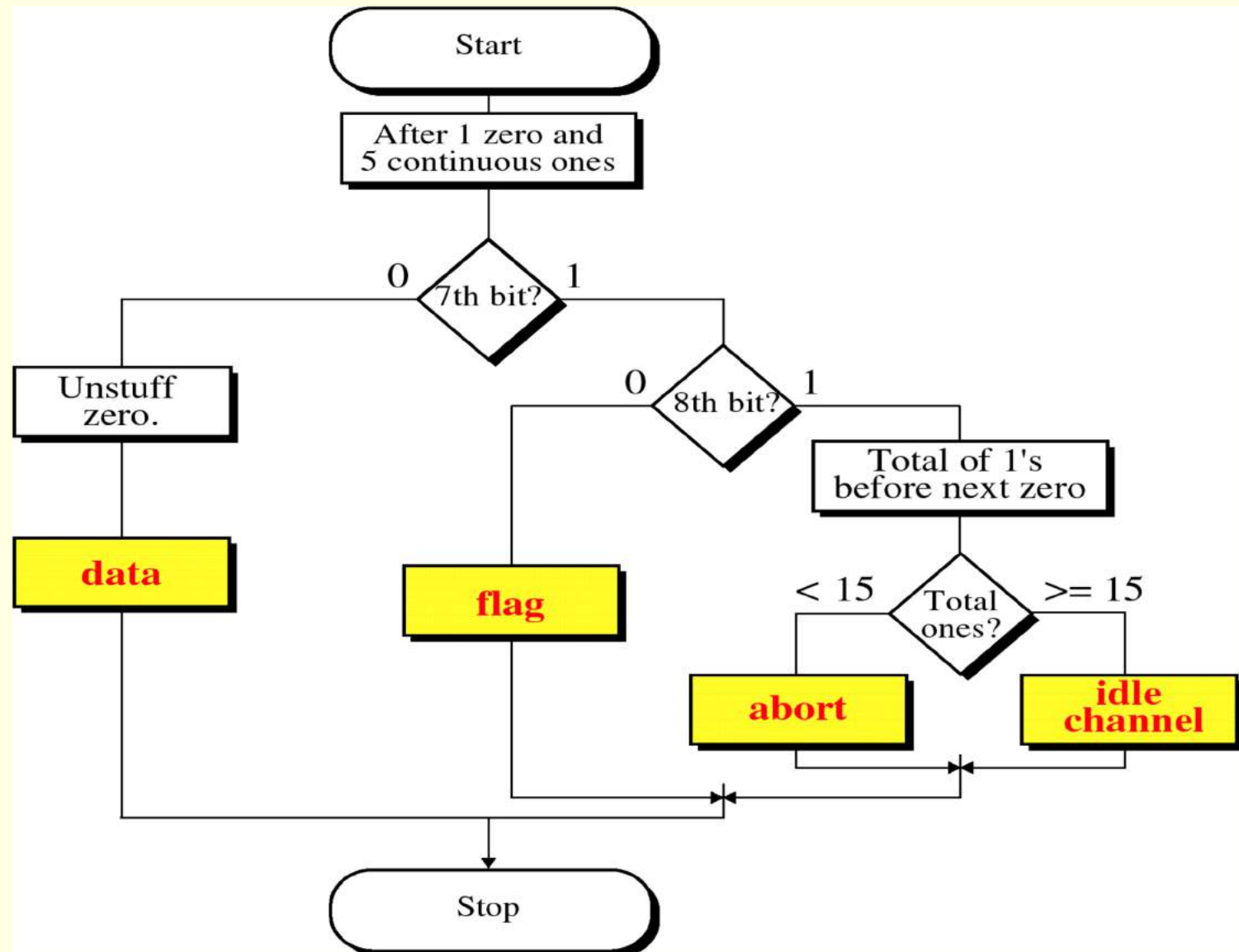
011111**0**10 011111**0**00 101101111 1**0**10010

How does the receiver identify a stuffed bit?

- ⇒ Receiver reads incoming bits and counts 1's.
- ⇒ When number of consecutive 1s after a zero is 5, it checks the next bit (7th bit).
- ⇒ If 7th bit = zero → receiver recognizes it as a **stuffed bit**, discard it and resets the counter.
- ⇒ If the 7th bit = 1 → then the receiver checks the 8th bit; If the 8th bit = 0, the sequence is recognized as a **flag**.

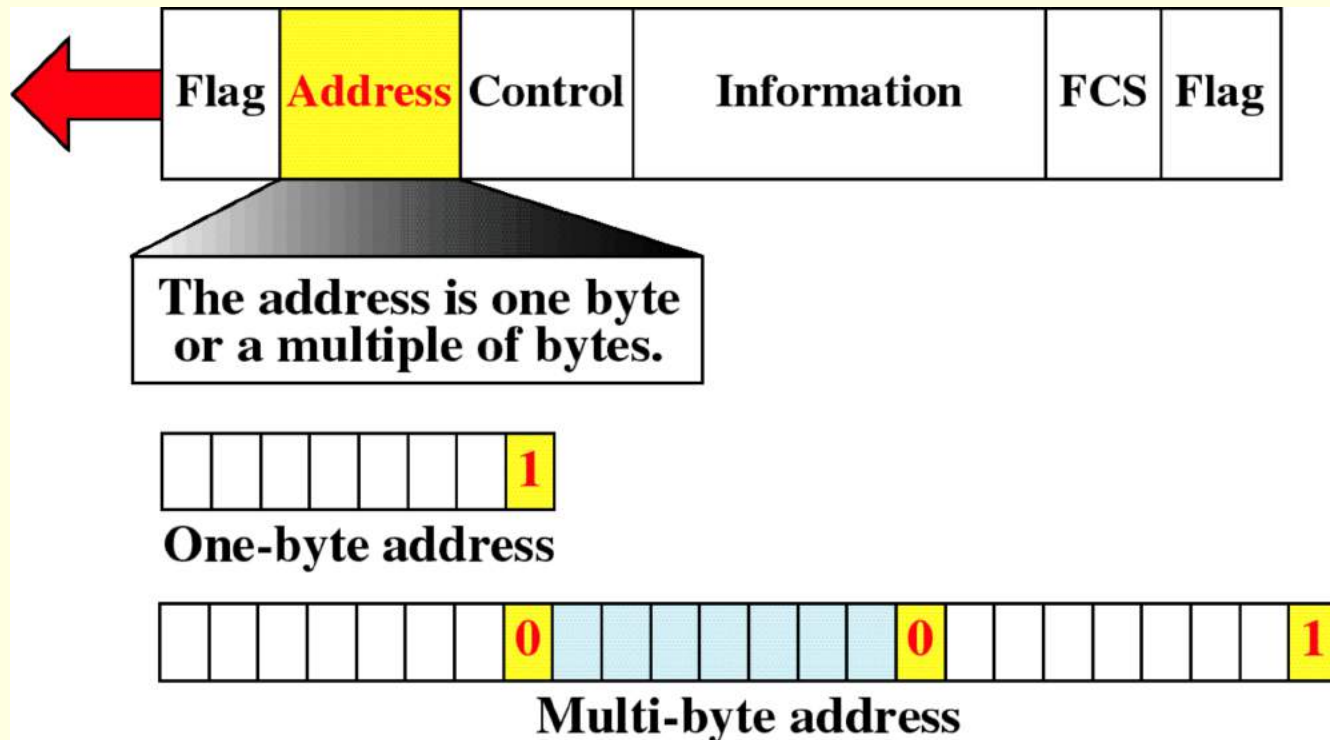
01111**0**10 011111**0**00 101101111 1**0**10010

How does the receiver identify a stuffed bit?



Address field

- Address field is one byte or more
- If the address is more than one byte, all bytes will end with 0, except the last one



HDLC Control Field



I-Frame



N(S)

N(R)

P/F Poll/final bit

N(S) Sequence number of frame sent

S-Frame



Code

N(R)

N(R) Sequence number of next frame expected

U-Frame

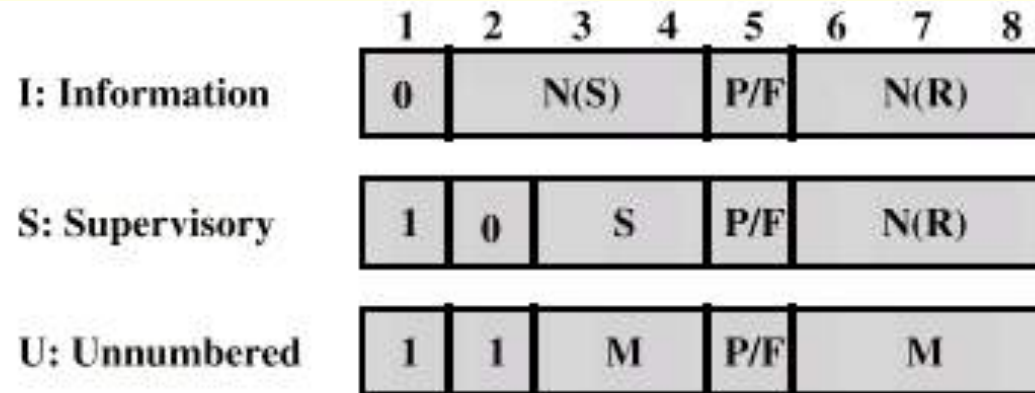


Code

Code

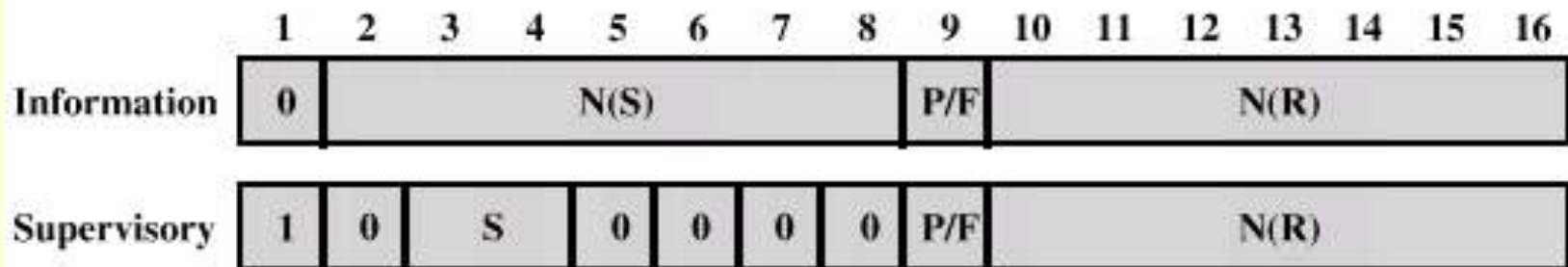
Code Code for supervisory or unnumbered frame

Control Field Diagram



N(S) = Send sequence number
 N(R) = Receive sequence number
 S = Supervisory function bits
 M = Unnumbered function bits
 P/F = Poll/final bit

(c) 8-bit control field format



(d) 16-bit control field format

Control Field

⇒ all three types contain a bit called (Poll/Final) P/F bit

I-Frame

⇒ **N(S)** : sequence # of the **sent frame**

⇒ **N(R)** : sequence # of frame **expected in return**

■ → **N(R)** is ACK field

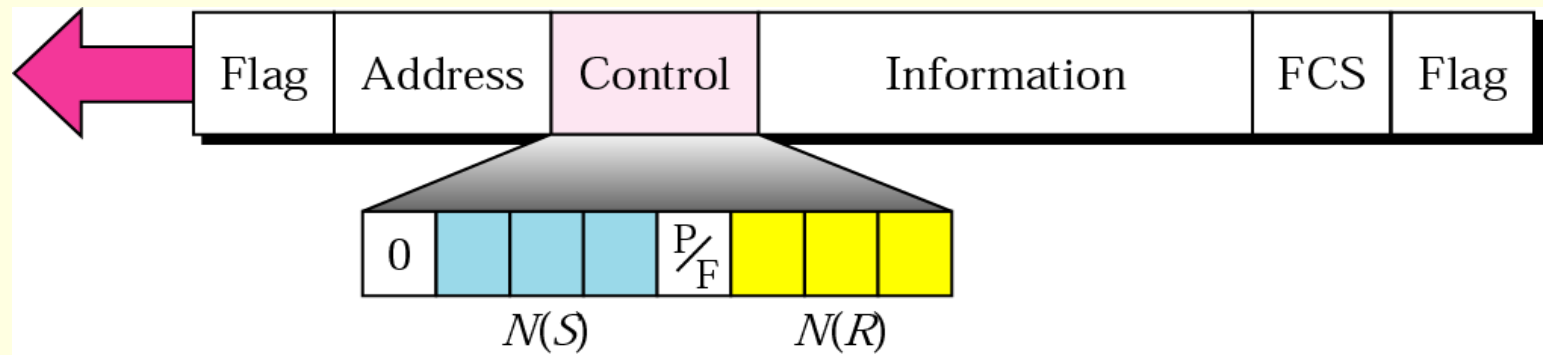
⇒ If last frame received is error free

→ N(R) number will be the next frame in sequence

⇒ If the frame was not received correctly

→ N(R) number will be the number of damaged frame indicating the need for retransmission

I frame

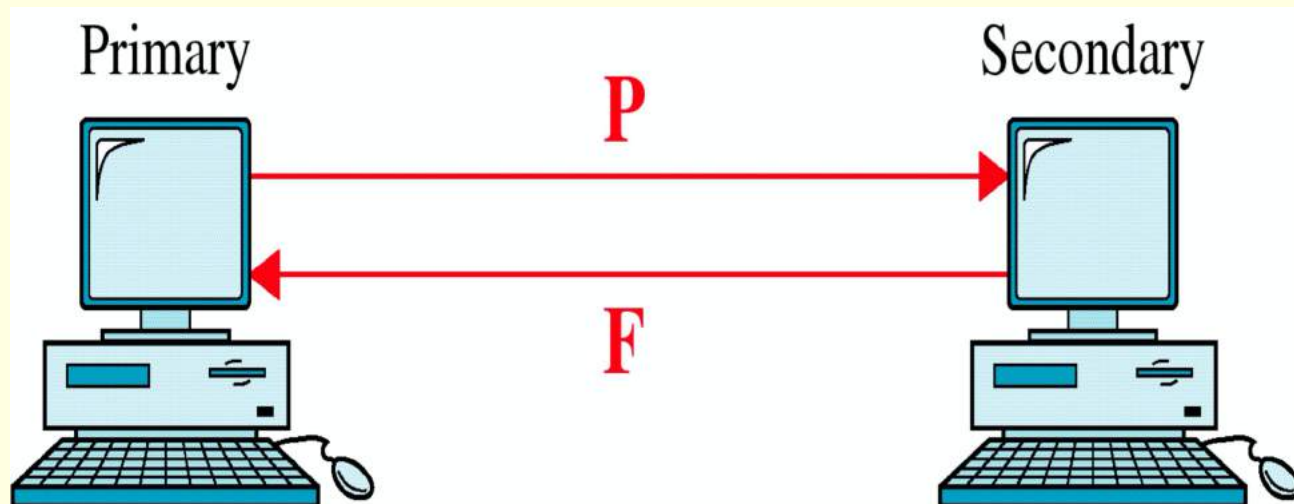


Poll/Final

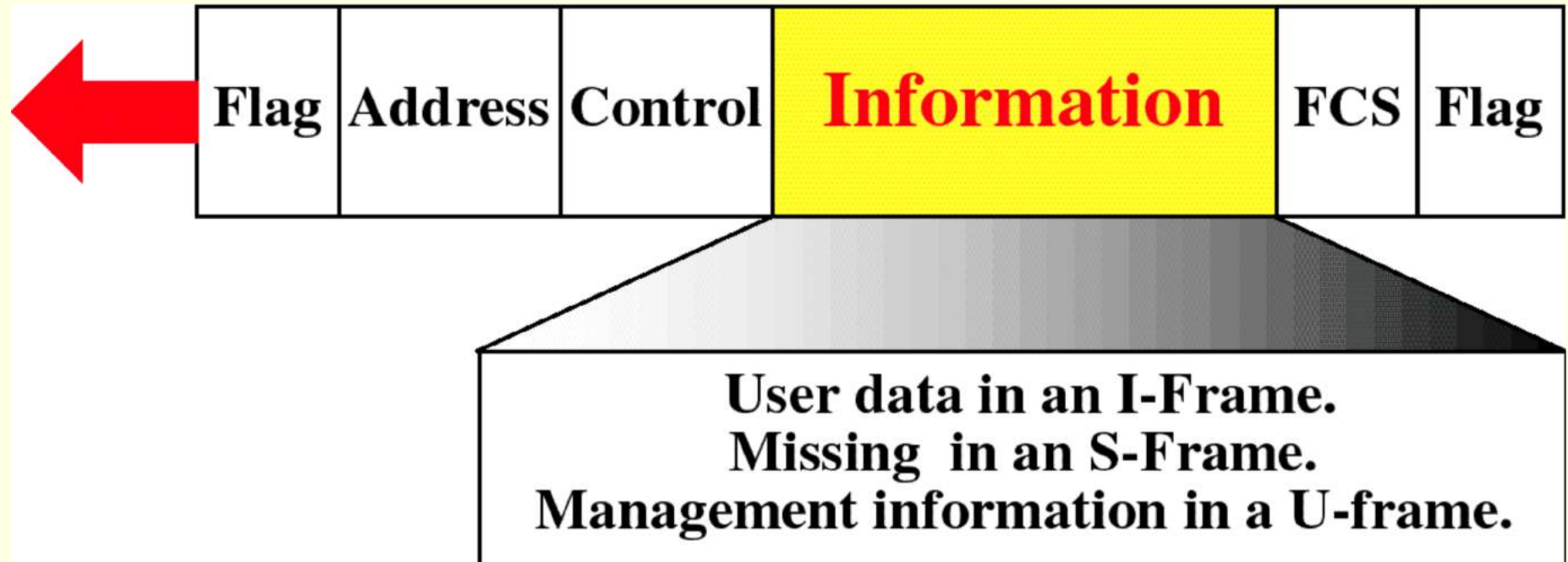
➡ $P/F = 1 \rightarrow$ POLL or Final

↪ **Poll** if frame is sent by the primary

↪ **Final** if frame is sent by the secondary



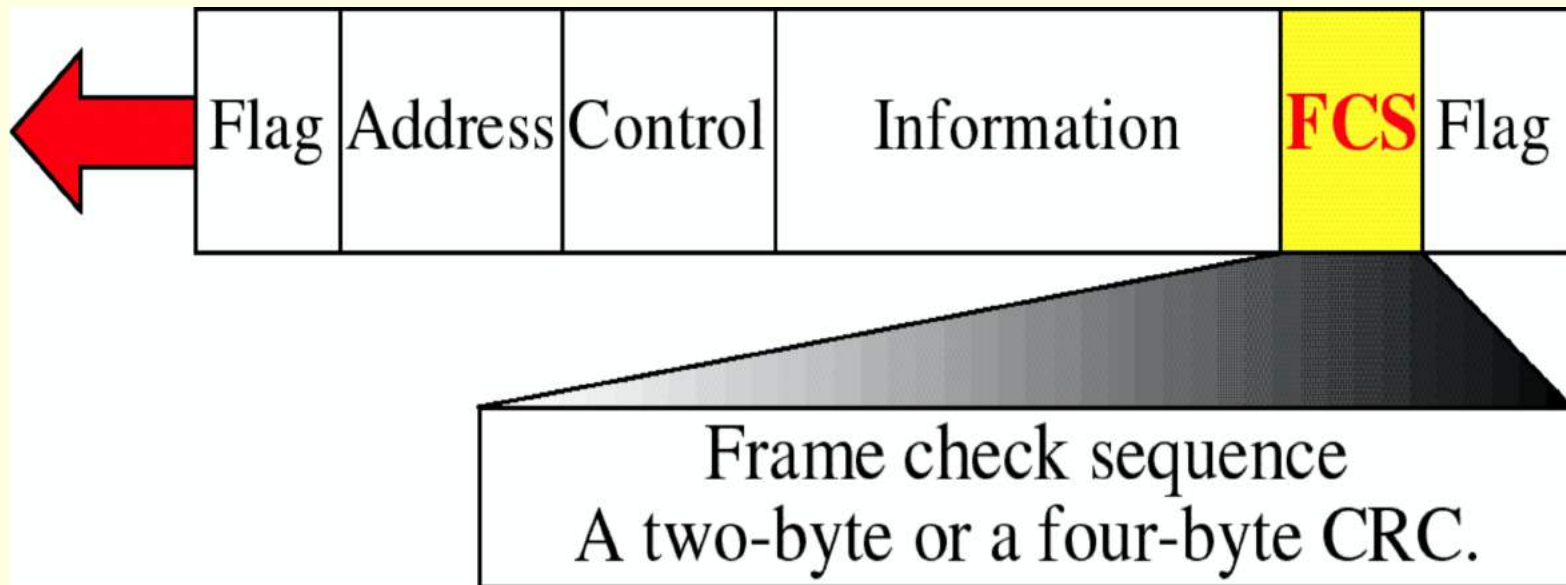
Information Field



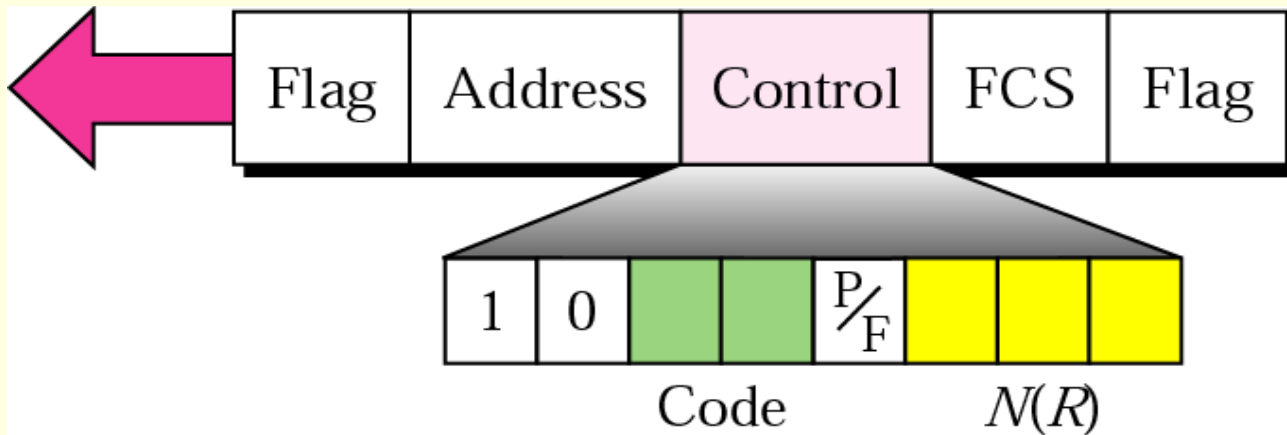
Information Field

- ◆ Contains **user data** in I-frame and **network management information** in a U-frame.
- ◆ It is possible to include flow and error control information in an I-frame that also contains data.
- ◆ In 2-way exchange of data (1/2 or full-duplex), the 2nd station can ACK receipt of data from the 1st station in the control field of its own data frame rather than sending a separate frame just for ACK.
- ◆ Combining data to be sent & ACK of the frame received in one single frame is called **PIGGYBACKING**.

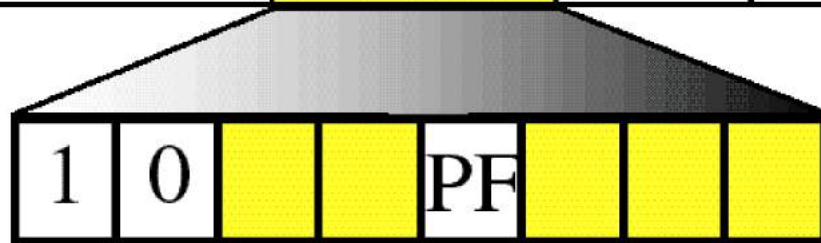
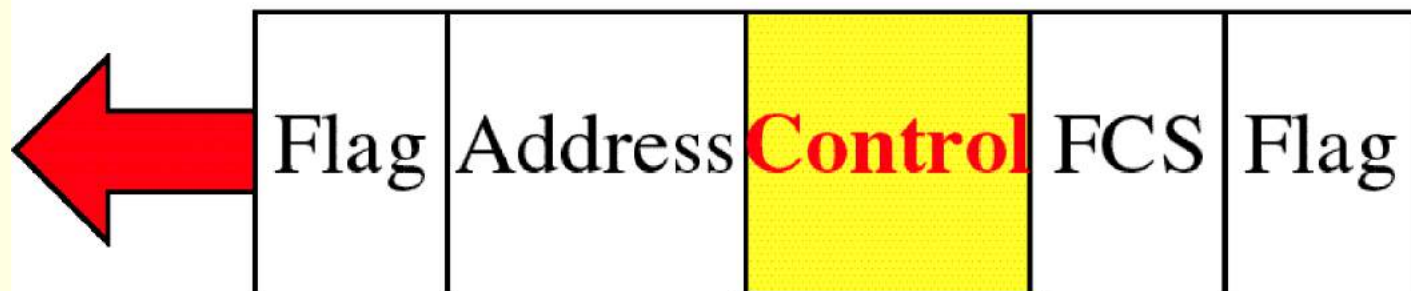
HDLC FCS Field



S-frame control field in HDLC



S-Frame



Code **N(R)**

| <u>Code</u> | <u>Command</u> |
|-------------|------------------------------|
| 00 | RR Receive ready |
| 01 | REJ Reject |
| 10 | RNR Receive not ready |
| 11 | SREJ Selective-reject |

⇒ Receive Ready (RR)

- ↘ Positive ACK of a received I- frame

⇒ Receive Not Ready (RNR)

- ↘ Is RR frame with additional duties

- ↘ It ACK the receipt of a frame and announces that the receiver is busy

⇒ Reject (REJ)

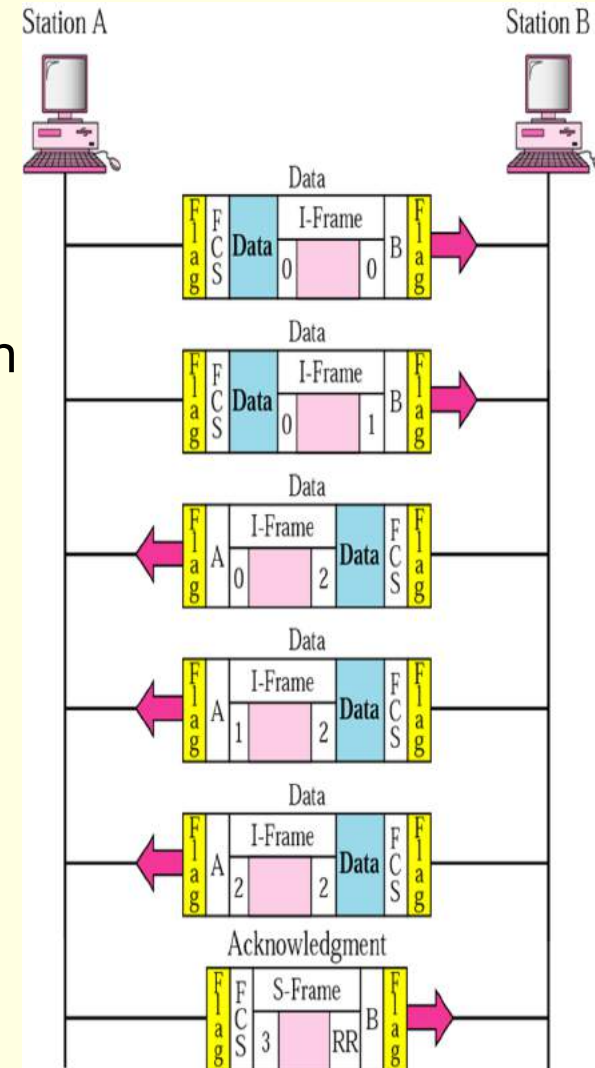
- ↘ This is a **NAK** frame that can be used in Go-back-n

⇒ Selective reject (SREJ)

- ↘ This is a NAK frame used in Selective Repeat ARQ

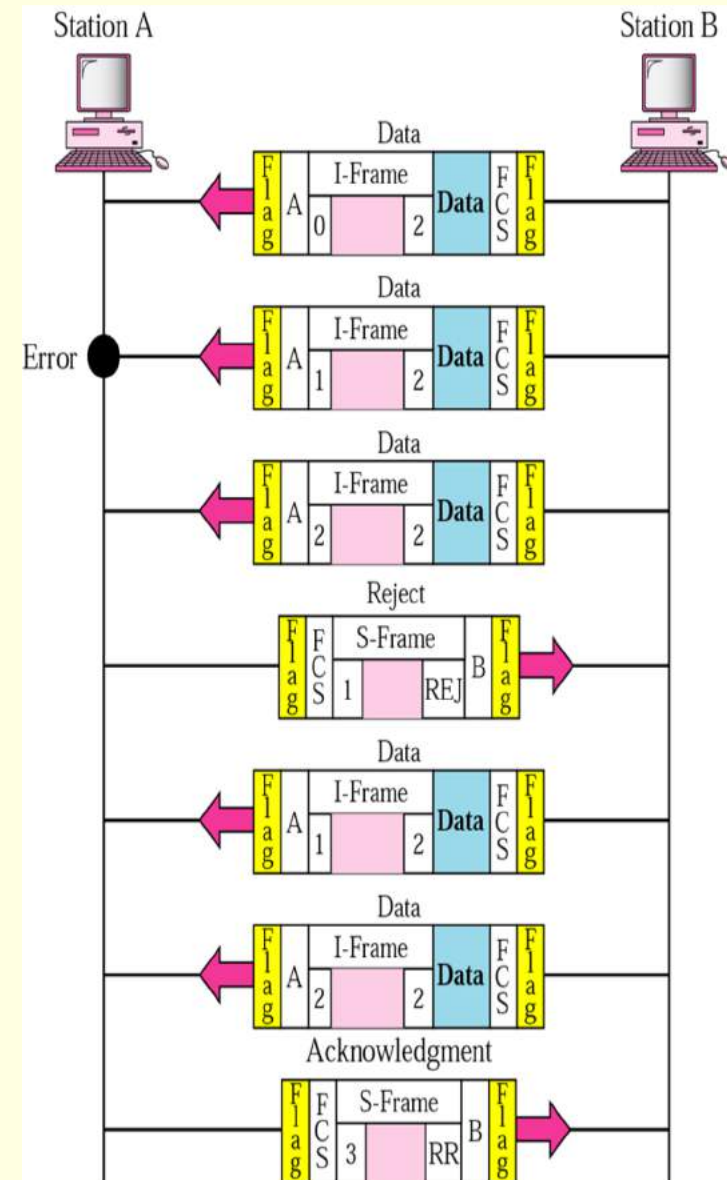
Example

- ➔ The figure shows an exchange using piggybacking where is no error
 - Station A begins the exchange of information with an I-frame numbered 0 followed by another I-frame numbered 1.
 - Station B piggybacks its acknowledgment of both frames onto an I-frame of its own.
 - Station B's first I-frame is also numbered 0 [N(S) field] and contains a 2 in its N(R) field, acknowledging the receipt of A's frames 1 and 0 and indicating that it expects frame 2 to arrive next.
 - Station B transmits its second and third I-frames (numbered 1 and 2) before accepting further frames from station A.
 - Its N(R) information, therefore, has not changed: B frames 1 and 2 indicate that station B is still expecting A frame 2 to arrive next.



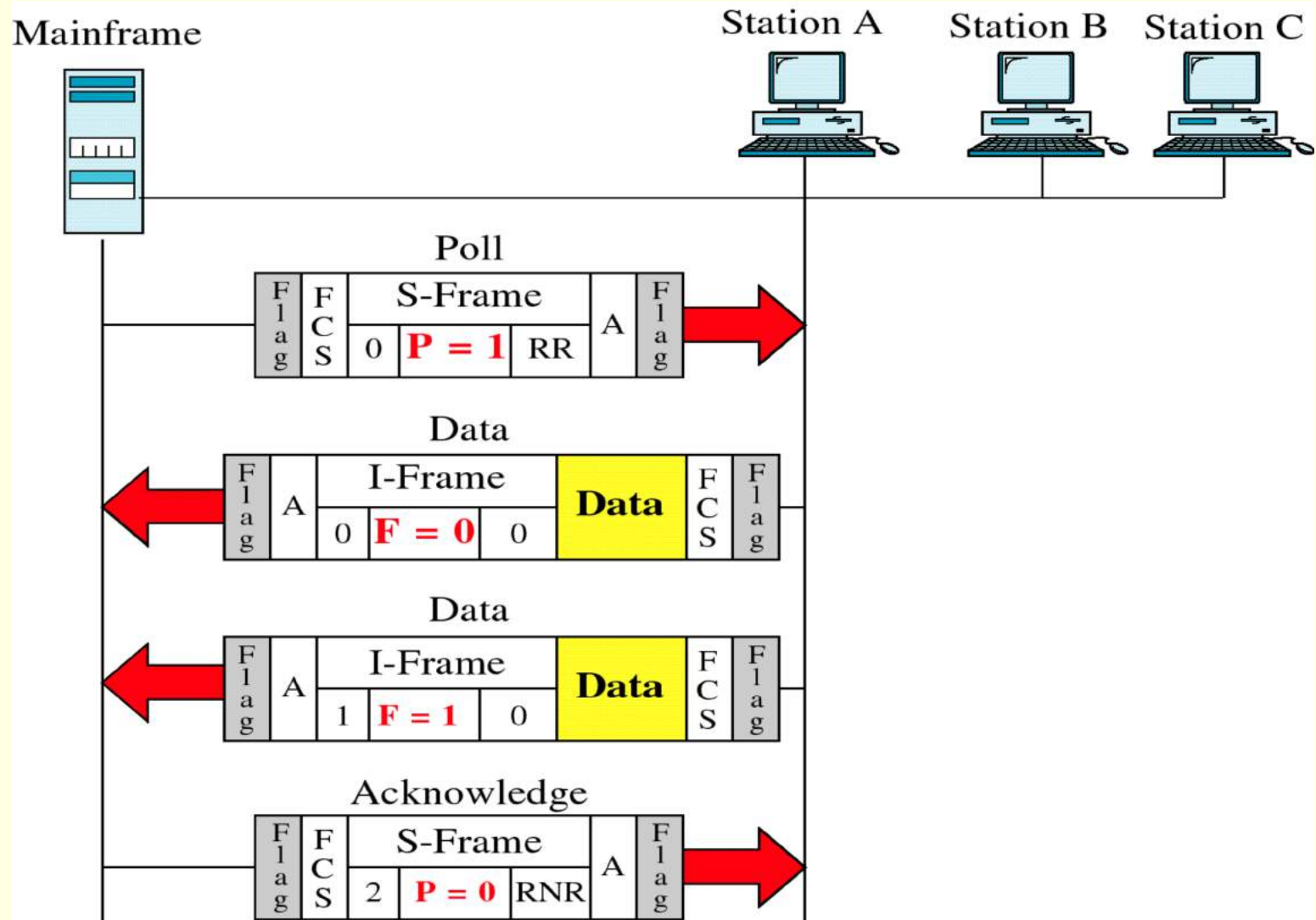
Example

- ➡ In the previous Example, suppose frame 1 sent from station B to station A has an error.
- ➡ Station A informs station B to resend frames 1 and 2 (the system is using the Go-Back-N mechanism)
- ➡ Station A sends a reject supervisory frame to announce the error in frame 1



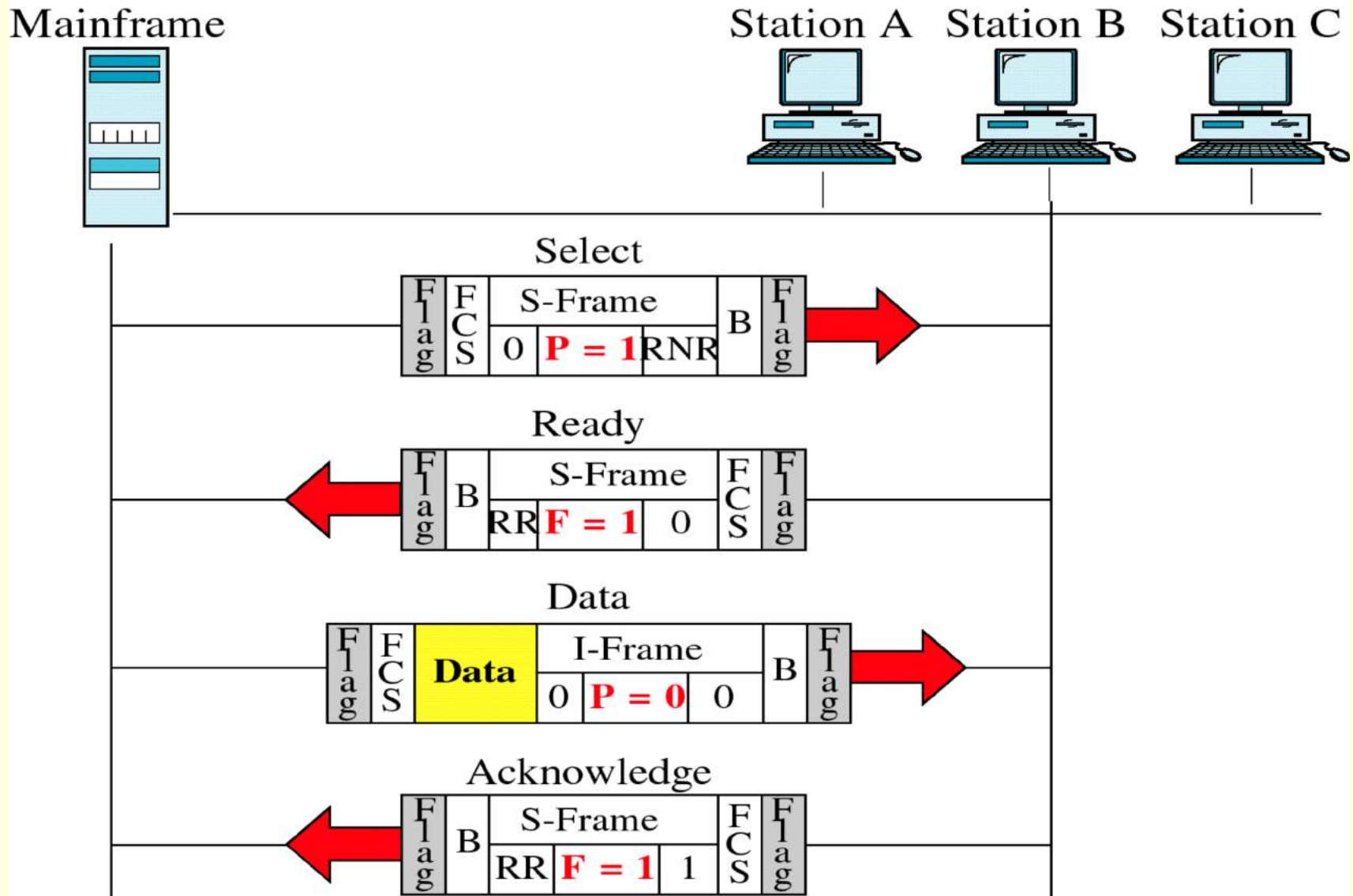
Polling Example

asking the secondary if it has anything to send



Selecting Example

A primary wishes to send data to secondary



HDLC Station Types

➡ Primary station

- ✚ Controls operation of link
- ✚ Frames issued are called commands
- ✚ Maintains separate logical link to each secondary station

➡ Secondary station

- ✚ Under control of primary station
- ✚ Frames issued called responses

➡ Combined station

- ✚ May issue commands and responses

HDLC Link Configurations

➡ Unbalanced

- ↪ One primary and one or more secondary stations
- ↪ Supports full duplex and half duplex

➡ Balanced

- ↪ Two combined stations
- ↪ Supports full duplex and half duplex

HDLC Transfer Modes (1)

➡ Normal Response Mode (NRM)

- ↪ Unbalanced configuration
- ↪ Primary initiates transfer to secondary
- ↪ Secondary may only transmit data in response to command from primary
- ↪ Used on multi-drop lines
- ↪ Host computer as primary
- ↪ Terminals as secondary

HDLC Transfer Modes (2)

➡ Asynchronous Balanced Mode (ABM)

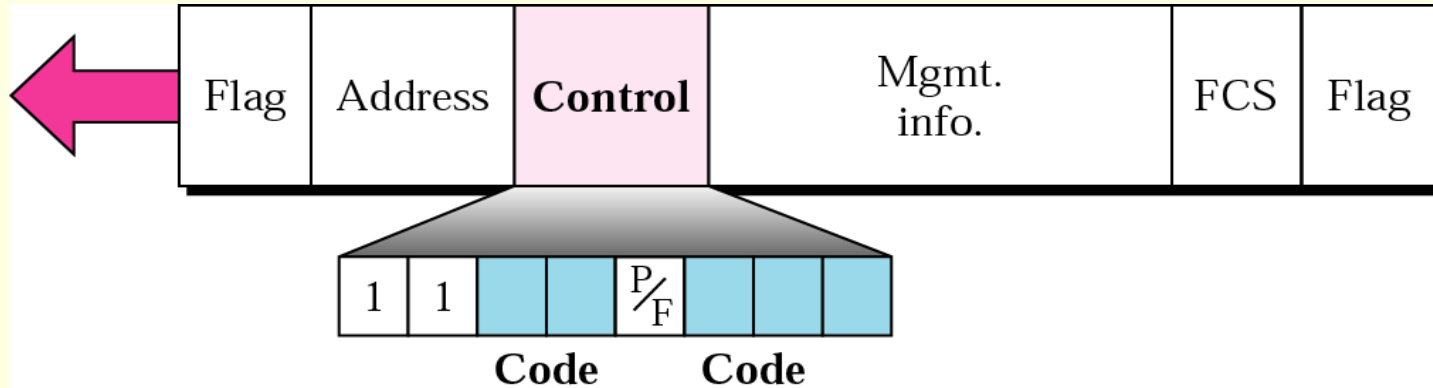
- ↪ Balanced configuration
- ↪ Either station may initiate transmission without receiving permission
- ↪ Most widely used
- ↪ No polling overhead

HDLC Transfer Modes (3)

➡ Asynchronous Response Mode (ARM)

- ↪ Unbalanced configuration
- ↪ Secondary may initiate transmission without permission from primary
- ↪ Primary responsible for line
- ↪ Rarely used

U-frame **control field** in HDLC



| Code | | Command | Response |
|------|-----|---------|----------|
| 00 | 001 | SNRM | |
| 11 | 011 | SNRME | |
| 11 | 100 | SABM | DM |
| 11 | 110 | SABME | |
| 00 | 000 | UI | UI |
| 00 | 110 | | UA |
| 00 | 010 | DISC | RD |
| 10 | 000 | SIM | RIM |
| 00 | 100 | UP | |
| 11 | 001 | RSET | |
| 11 | 101 | XID | XID |
| 10 | 001 | | FRMR |

U-frame control command and response

| Command/response | Meaning |
|------------------|---|
| SNRM | Set normal response mode |
| SNRME | Set normal response mode (extended)– control field 2 bytes |
| SABM | Set asynchronous balanced mode |
| SABME | Set asynchronous balanced mode (extended) |
| UP | Unnumbered poll |
| UI | Unnumbered information |
| UA | Unnumbered acknowledgment |
| RD | Request disconnect |
| DISC | Disconnect |
| DM | Disconnect mode |
| RIM | Request information mode |
| SIM | Set initialization mode |
| RSET | Reset |
| XID | Exchange ID |
| FRMR | Frame reject |

HDLC Operation

- ➡ Exchange of information, supervisory and unnumbered frames
- ➡ Three phases
 - ↪ Initialization
 - ↪ Data transfer
 - ↪ Disconnect

U-frame **Mode setting**

- ➡ Mode-setting commands sent by the primary or combined station wishing to control an exchange
- ➡ If a combined station wishes to establish a temporary primary-to-secondary relationship with another station it sends a U-frame containing code 00-001 (Normal Response Mode)

U-frame **Disconnection**

➔ There three disconnection codes

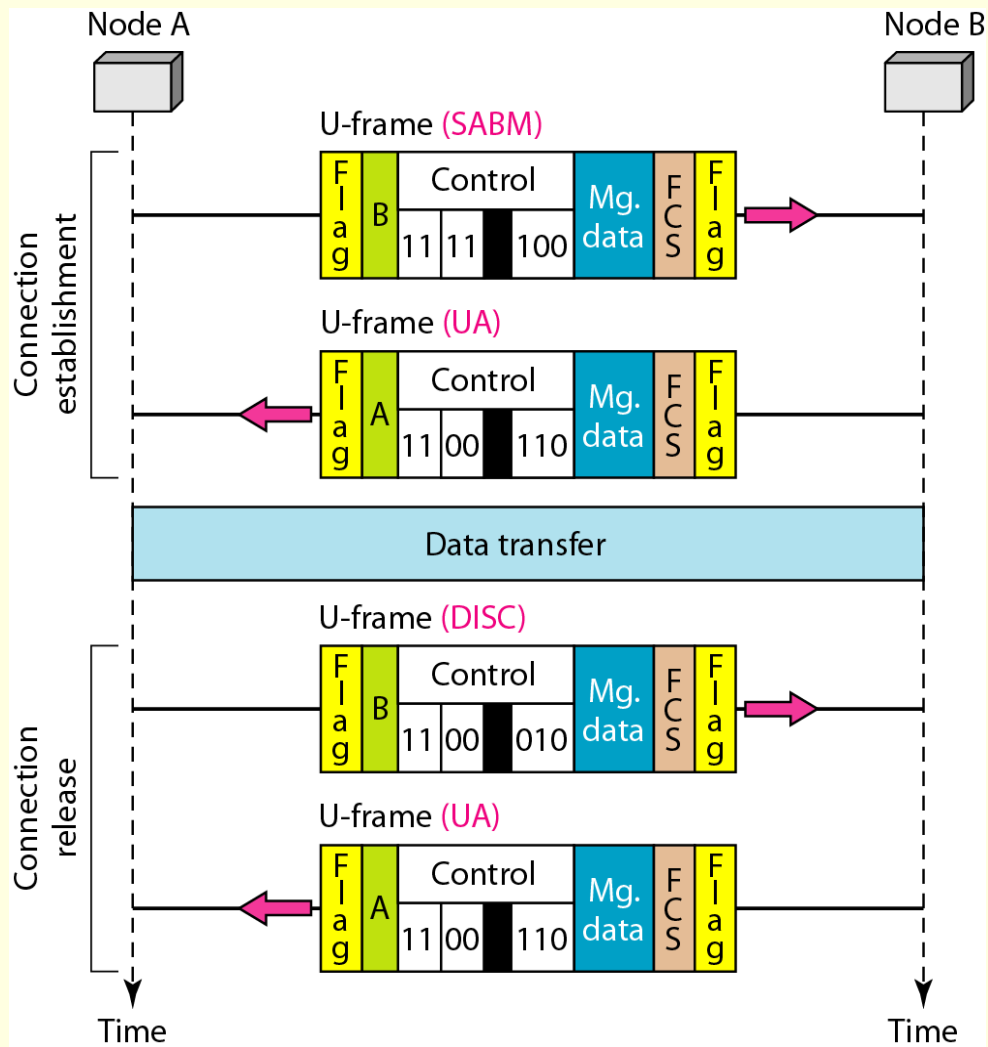
➤ One command from acting primary or combined station

- disconnection (DISC 00 010) is sent by the 1st station to the 2nd station to terminate the connection

➤ Two responses from the receiving station

- request disconnect (RD 00 010) is a request by the 2nd station to the 1st that a DISC be issued.
- disconnect mode (DM 11 000) is transmitted by the addressed station as a negative response to mode-setting command

Example of connection and disconnection



Other DLC Protocols (LAPB,LAPD)

➡ Link Access Procedure, Balanced (LAPB)

- ↪ Part of X.25 (ITU-T)
- ↪ Subset of HDLC - ABM
- ↪ Point to point link between system and packet switching network node

➡ Link Access Procedure, D-Channel

- ↪ ISDN (ITU-D)
- ↪ ABM
- ↪ Always 7-bit sequence numbers (no 3-bit)
- ↪ 16 bit address field contains two sub-addresses
 - One for device and one for user (next layer up)

Other DLC Protocols (LLC)

➡ Logical Link Control (LLC)

- ↪ IEEE 802
- ↪ Different frame format
- ↪ Link control split between medium access layer (MAC) and LLC (on top of MAC)
- ↪ No primary and secondary - all stations are peers
- ↪ Two addresses needed
 - Sender and receiver
- ↪ Error detection at MAC layer
 - 32 bit CRC
- ↪ Destination and source access points (DSAP, SSAP)

Other DLC Protocols

(Frame Relay) (1)

- ➡ Streamlined capability over high speed packet switched networks
- ➡ Used in place of X.25
- ➡ Uses Link Access Procedure for Frame-Mode Bearer Services (LAPF)
- ➡ Two protocols
 - ↳ Control - similar to HDLC
 - ↳ Core - subset of control

Other DLC Protocols

(Frame Relay) (2)

- ⇒ ABM
- ⇒ 7-bit sequence numbers
- ⇒ 16 bit CRC
- ⇒ 2, 3 or 4 octet address field
 - ↳ Data link connection identifier (DLCI)
 - ↳ Identifies logical connection

Other DLC Protocols (ATM)

- ➡ Asynchronous Transfer Mode
- ➡ Streamlined capability across high speed networks
- ➡ Not HDLC based
- ➡ Frame format called “cell”
- ➡ Fixed 53 octet (424 bit)

Point-to-Point Protocol (PPP)

Basics of PPP

- ⇒ Used on point-to-point links such as modem dialup, DSL, and cable modem
- ⇒ SLIP (serial line Internet protocol) was first but could only support IP and only static IP address assignment
- ⇒ PPP solves both above problems

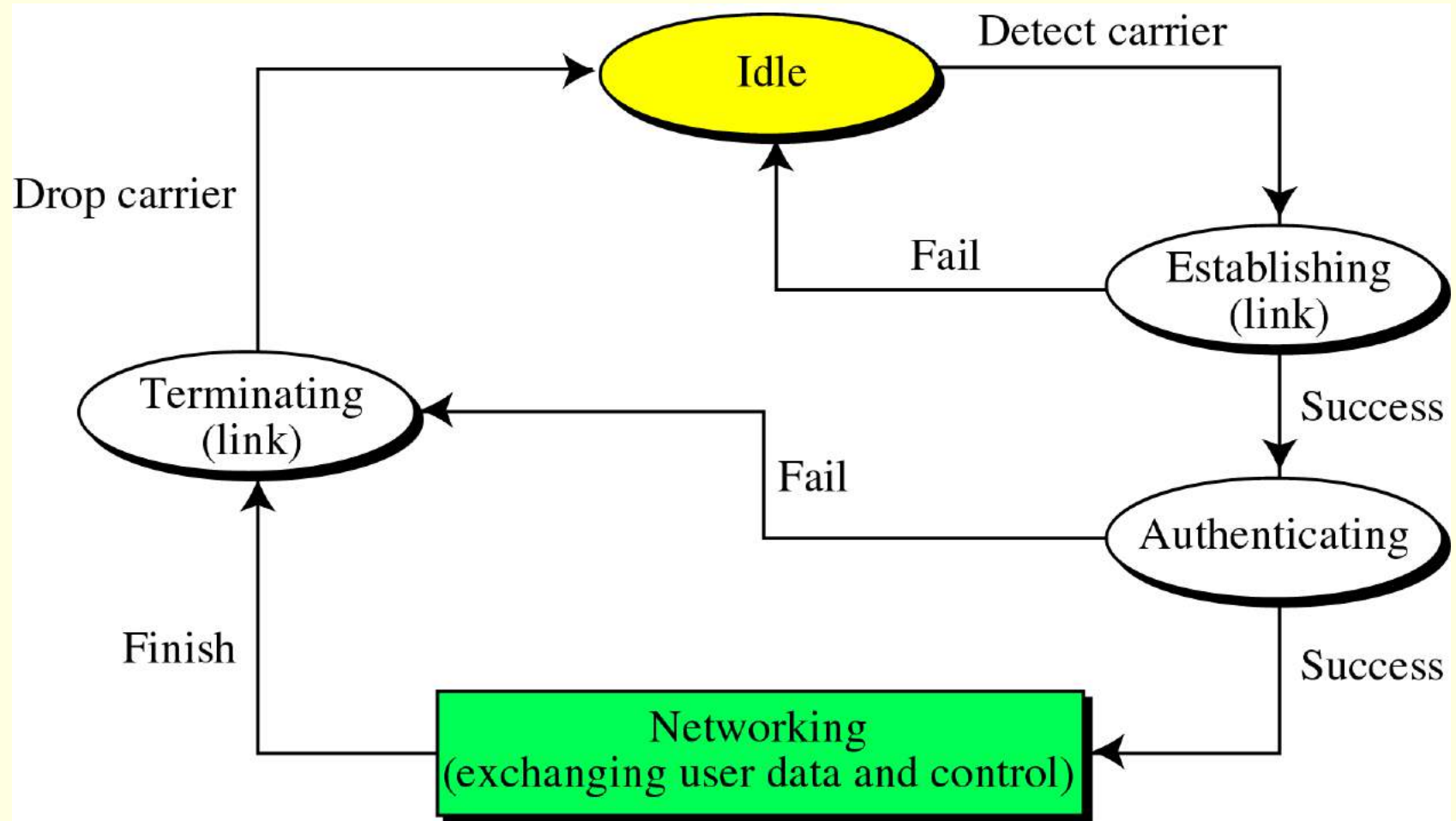
PPP States

- ➡ Idle state – link is not being used
- ➡ Establishing state – one endpoint starts a dialog; options are exchanged between endpoints; several packets may be exchanged
- ➡ Authenticating state – optional, two sides agree to authenticate (described later)

PPP States

- ➡ Networking state – primary state, exchange of user control and data packets can now be performed
- ➡ Terminate state – one side wishes to tear down connection; several packets exchanged (housekeeping)
- ➡ (See figure next slide)

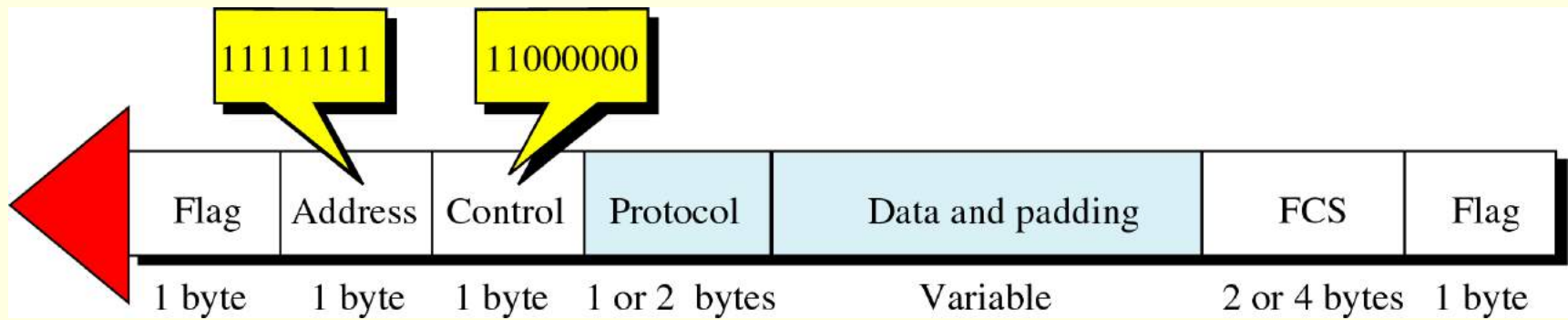
Transition States



PPP Layers

- ➔ PPP has only two layers – physical and data link
- ➔ Physical layer is not defined – it is whatever the user uses
- ➔ Data link layer looks like HDLC, except
address field = 11111111 (broadcast), control field = 11000000 (a HDLC U-frame)

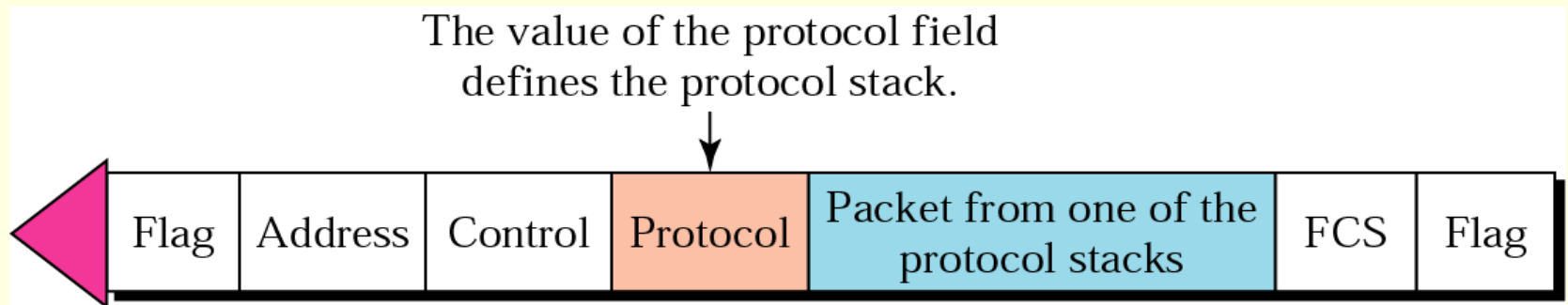
PPP Frames



PPP Layers

- ➡ The Data field carries the packets from one of three other protocols - Link Control Protocol, authentication protocols, and Network Control Protocol, all described shortly

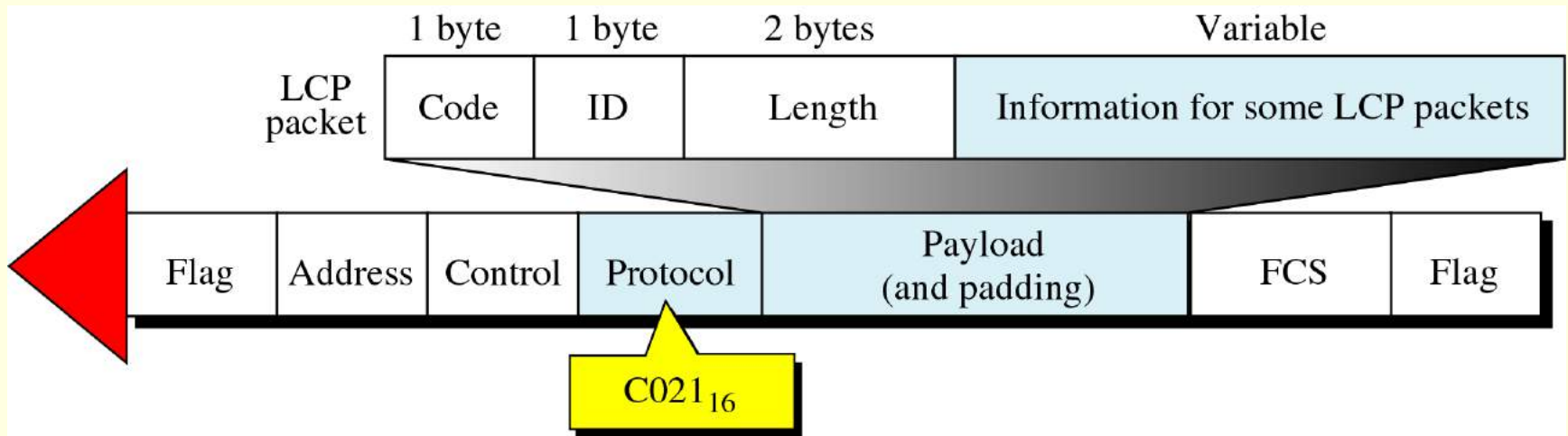
Protocol stack



Link Control Protocol (LCP)

- ➡ Responsible for establishing, maintaining, configuring, terminating link, and negotiation
- ➡ All LCP packets are carried in payload field of PPP frame – PPP field Protocol = hex C021

LCP Packet Encapsulated in a Frame



LCP packets and their codes

| Code | Packet Type | Description |
|-------------------------|-------------------|--|
| 01 ₁₆ | Configure-request | Contains the list of proposed options and their values |
| 02 ₁₆ | Configure-ack | Accepts all options proposed |
| 03 ₁₆ | Configure-nak | Announces that some options are not acceptable |
| 04 ₁₆ | Configure-reject | Announces that some options are not recognized |
| 05 ₁₆ | Terminate-request | Requests to shut down the line |
| 06 ₁₆ | Terminate-ack | Accepts the shut down request |
| 07 ₁₆ | Code-reject | Announces an unknown code |
| 08 ₁₆ | Protocol-reject | Announces an unknown protocol |
| 09 ₁₆ | Echo-request | A type of hello message to check if the other end is alive |
| 0A ₁₆ | Echo-reply | The response to the echo-request message |
| 0B ₁₆ | Discard-request | A request to discard the packet |

Common options

| Option | Default |
|---------------------------------------|---------|
| Maximum receive unit | 1500 |
| Authentication protocol | None |
| Protocol field compression | Off |
| Address and control field compression | Off |

Authentication

- ⇒ Potentially important since this is dial-up communication
- ⇒ Two possible protocols for authentication:
 - ↳ Password Authentication Protocol (PAP)
 - ↳ Challenge Handshake Authentication Protocol (CHAP)

Authentication – PAP

⇒ Two-step process

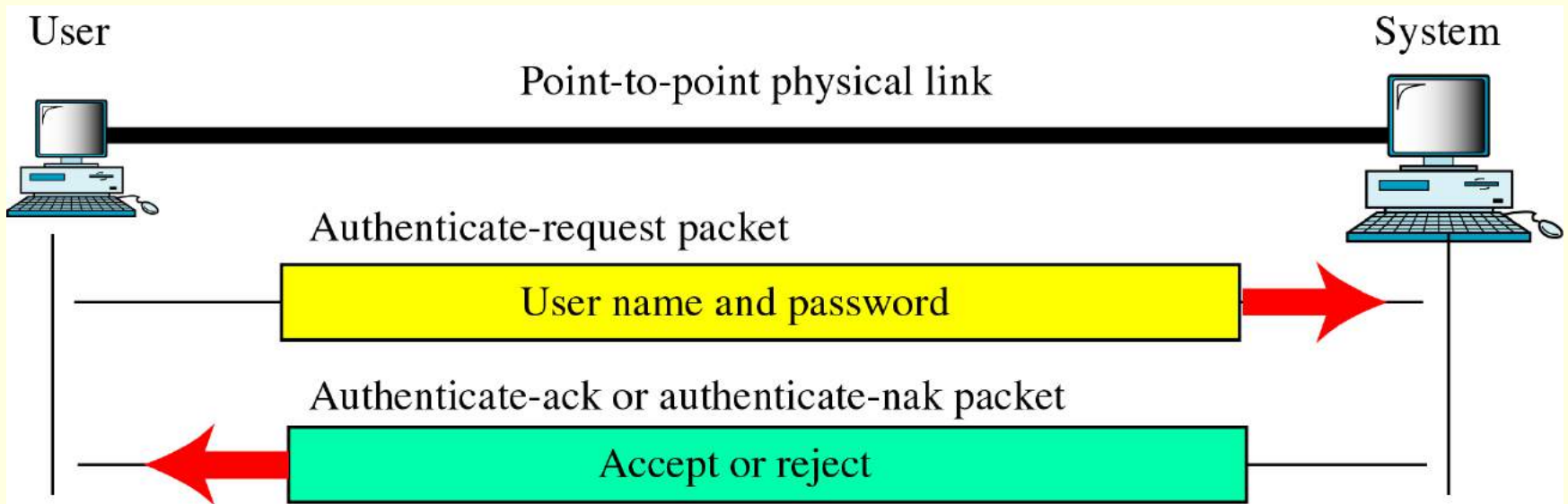
- ⇒ User sends ID and password

- ⇒ System verifies

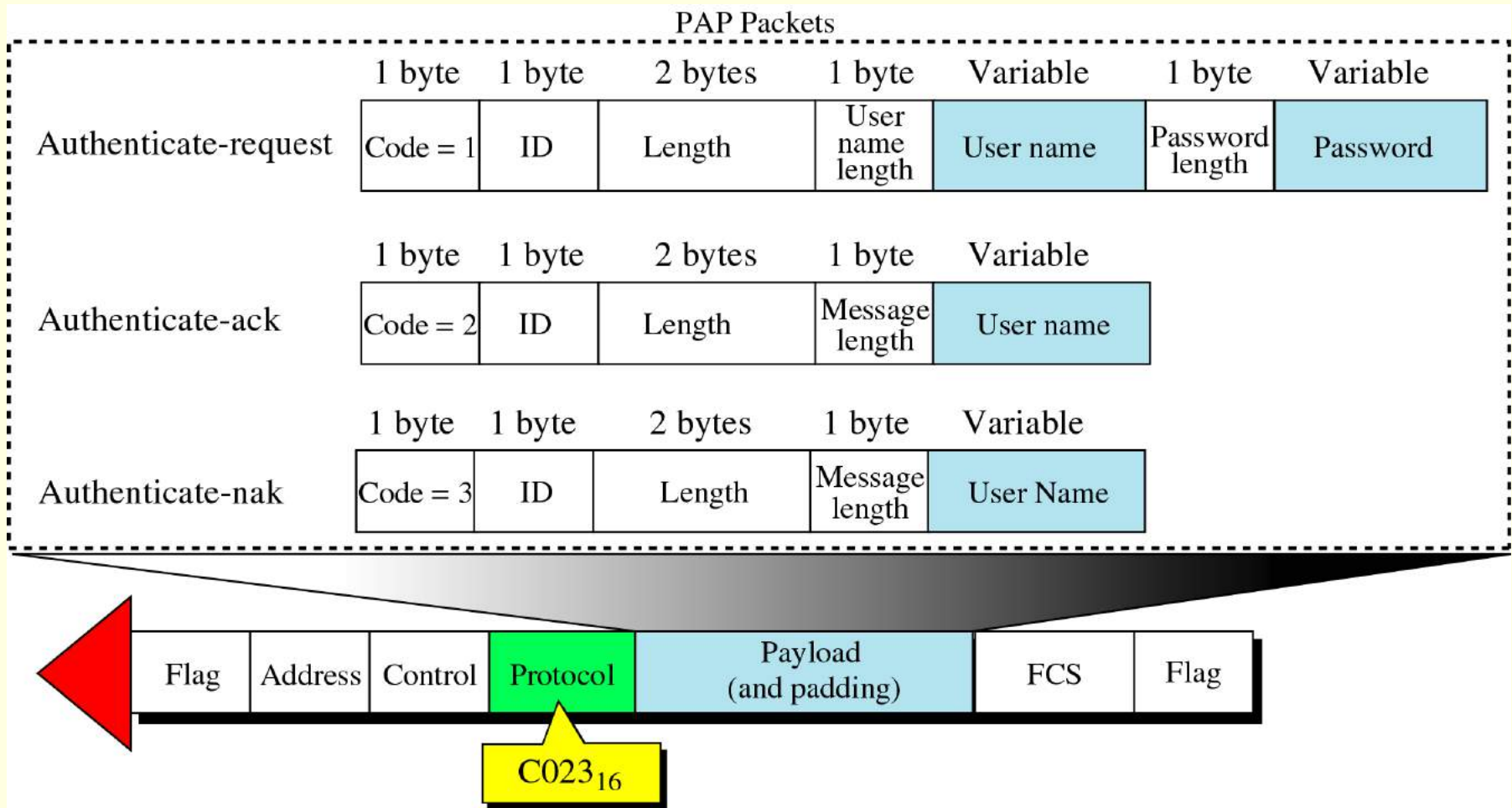
⇒ PAP packets are encapsulate in a PPP frame

⇒ There are 3 types of PAP packets (see the next two slides)

PAP



PAP Packets

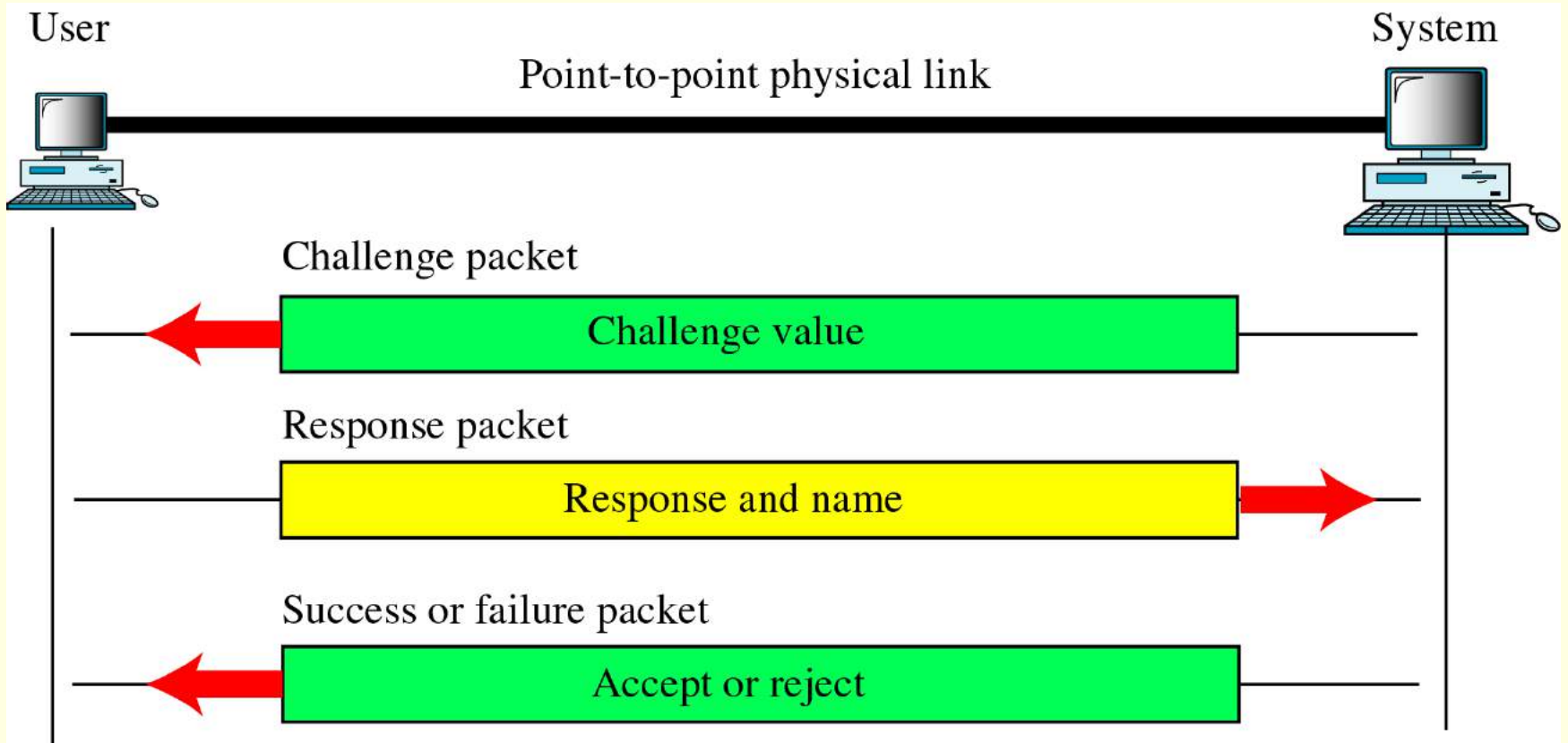


Authentication – CHAP

⇒ Three-way handshake

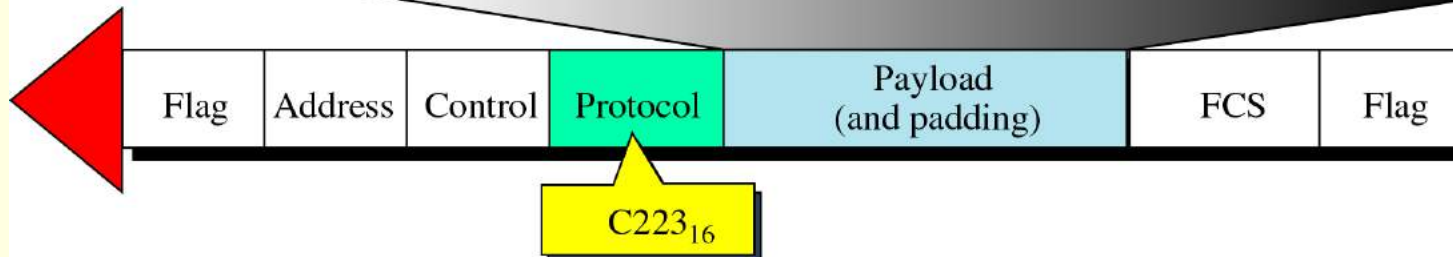
- ✚ System sends a challenge packet
- ✚ User applies a predefined function that takes the challenge value and the user's own password and creates a result
- ✚ System does the same; then compares its result to user's result

CHAP



CHAP Packets

| | | | | | | |
|-----------|----------|--------|---------|------------------|-----------------|----------|
| Challenge | 1 byte | 1 byte | 2 bytes | 1 byte | Variable | Variable |
| | Code = 1 | ID | Length | Challenge length | Challenge value | Name |
| | 1 byte | 1 byte | 2 bytes | 1 byte | Variable | Variable |
| | Code = 2 | ID | Length | Response length | Response value | Name |
| Success | 1 byte | 1 byte | 2 bytes | Variable | | |
| | Code = 3 | ID | Length | Message | | |
| Failure | 1 byte | 1 byte | 2 bytes | Variable | | |
| | Code = 4 | ID | Length | Message | | |



IPCP (An NCP Protocol)

- ⇒ Now that a link has been established and optional security has been established, we need to establish a *network layer connection*
- ⇒ IPCP, or Internetwork Protocol Control Protocol, is an NCP (Network Control Protocol)

IPCP

⇒ Seven packet types:

↳ Configure-request (01)

↳ Configure-ACK (02)

↳ Configure-NAK (03)

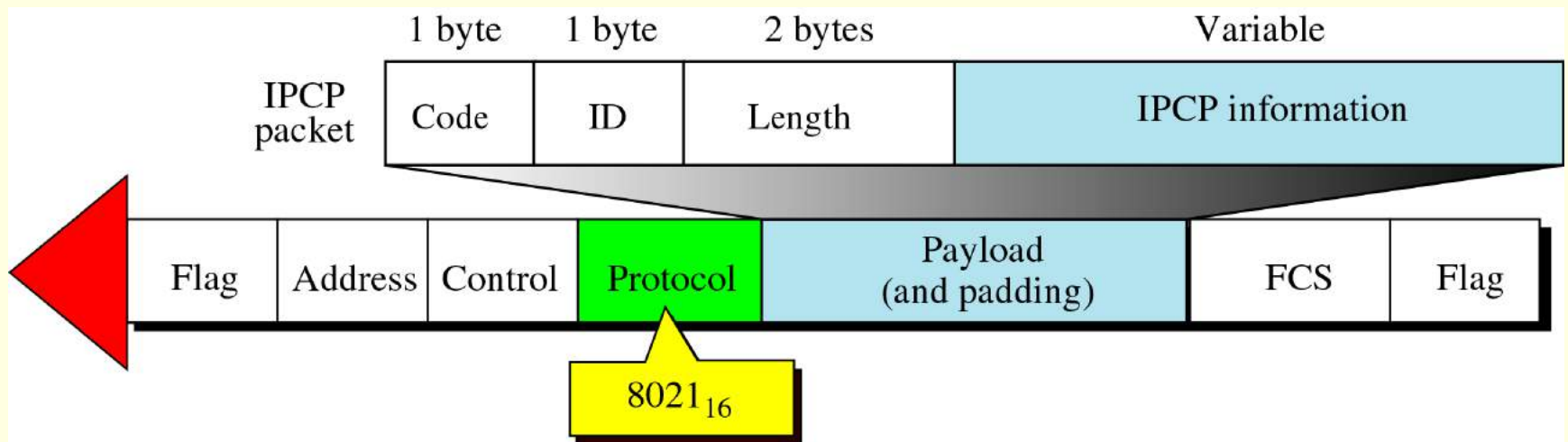
↳ Configure-reject (04)

↳ Terminate-request (05)

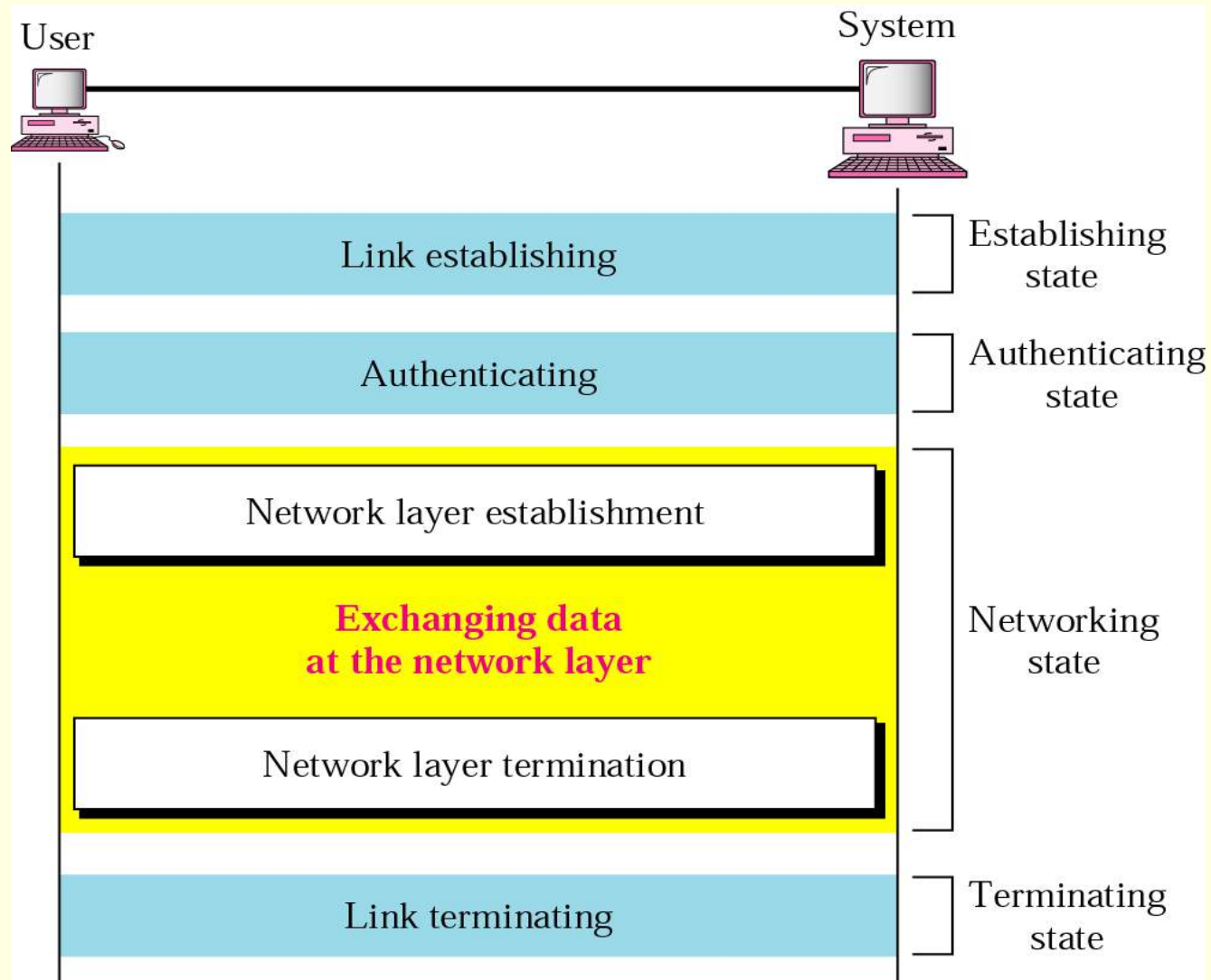
↳ Terminate-ACK (06)

↳ Code-reject (07)

IPCP Packet Encapsulated in PPP Frame



An example



An Example

