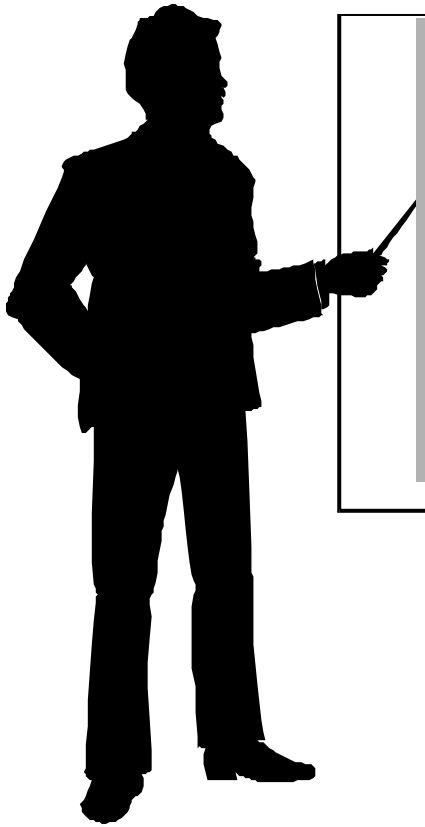
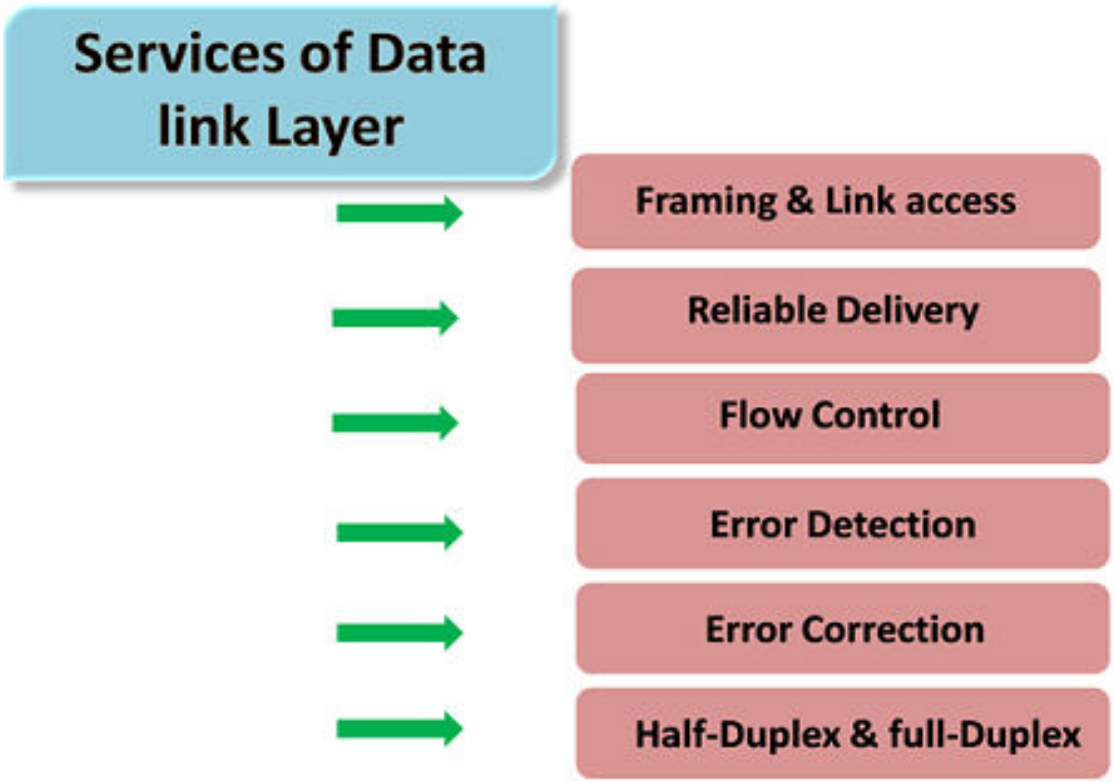
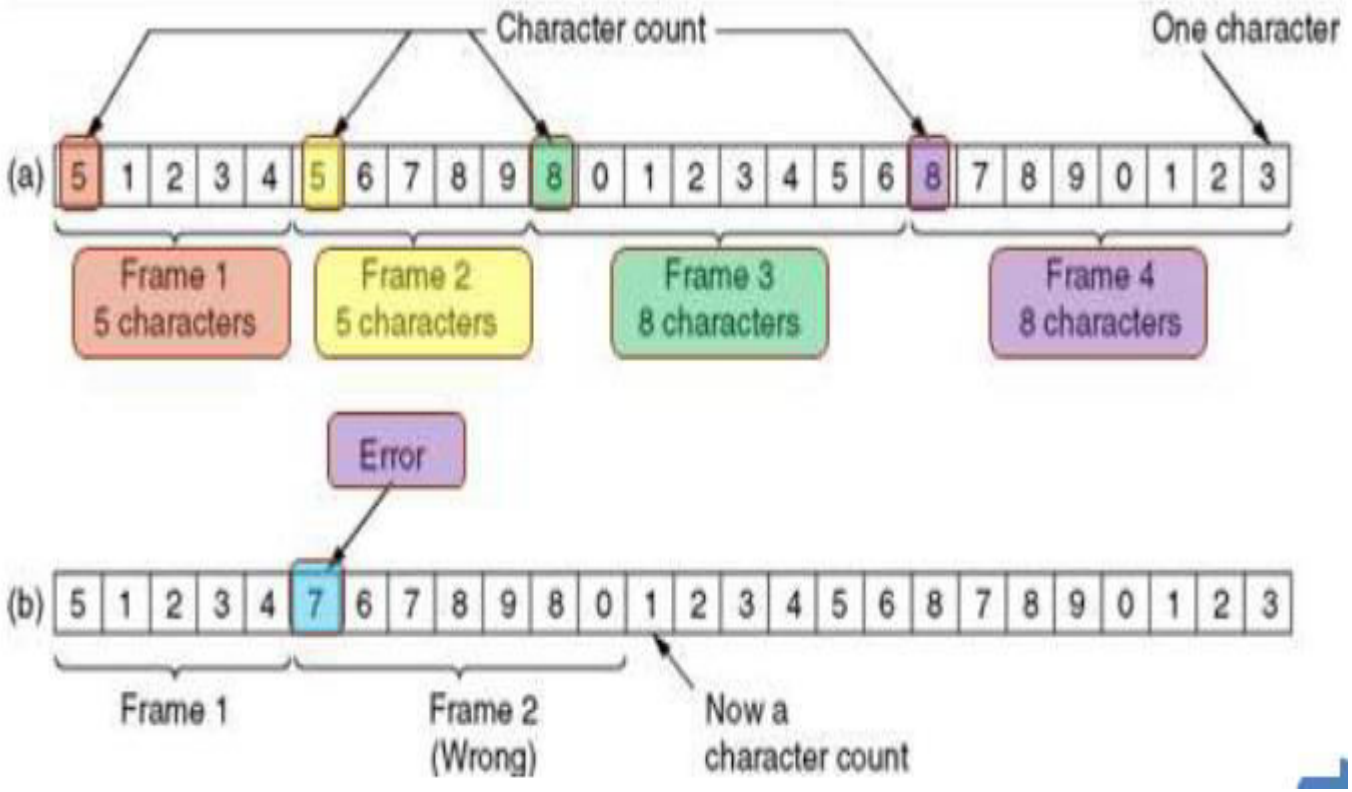


Data Link Protocols

In networking and communications, the transmission of a unit of data (frame, packet) from one node to another. Known as a "layer 2 protocol," the data link protocol is responsible for ensuring that the bits and bytes received are identical to the bits and bytes sent. For example, in a local network, if a message is split into 100 packets and a station receives 97, the data link protocol ensures those 97 are error free, but it is not aware of the three missing ones. TCP and other higher-layer protocols make sure all 100 are delivered (see TCP/IP). Following are major types:


Services of Data link Layer
Framing & Link access
Reliable Delivery
Flow Control
Error Detection
Error Correction
Half-Duplex & full-Duplex


Character count
One character
(a)
5 1 2 3 4 5 6 7 8 9 8 0 1 2 3 4 5 6 8 7 8 9 0 1 2 3
Frame 1
5 characters
Frame 2
5 characters
Frame 3
8 characters
Frame 4
8 characters
Error
(b)
5 1 2 3 4 7 6 7 8 9 8 0 1 2 3 4 5 6 8 7 8 9 0 1 2 3
Frame 1
Frame 2
(Wrong)
Now a
character count

11장 Data Link Protocols

11.1 Asynchronous protocol

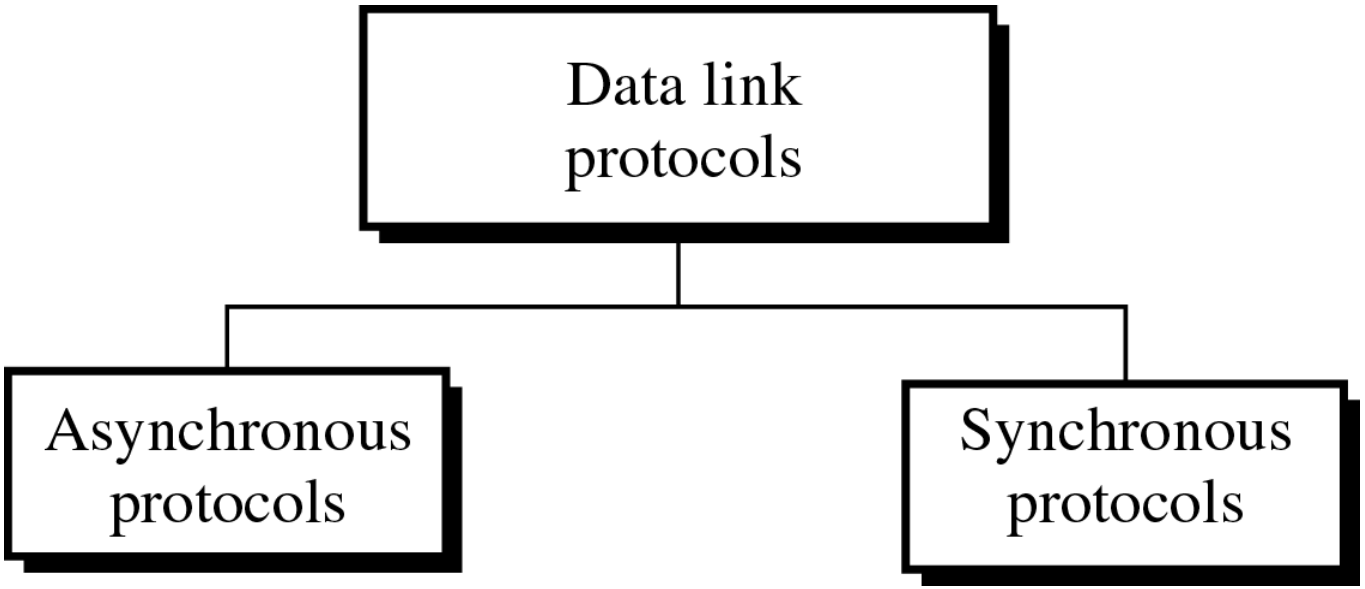
11.2 Synchronous protocol

11.3 Character-Oriented protocol

11.4 Bit-Oriented protocol

Introduction

- **Data Link Protocol : a set of specifications used to implementation of the data link layer**
  - **Protocol : referring to a set of rules or conventions for executing a particular task.**



Introduction (cont'd)

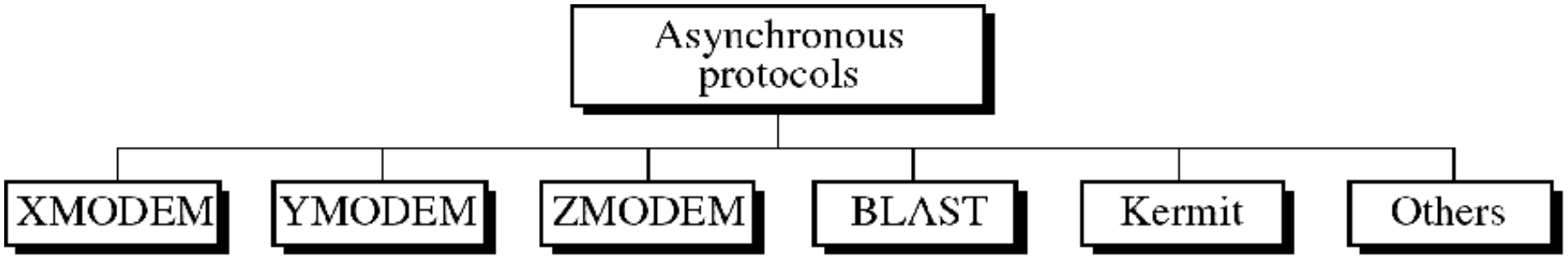
- Data Link Protocol
  - Asynchronous protocols

    ~ treat each character in a bit stream independently
  - Synchronous protocols

    ~ take the whole bit stream and chop it into characters of equal size

11.1 Asynchronous protocols

- Asynchronous protocols, used primarily in modems, feature start and stop bits and variable length gabs between characters
  - not complex
  - inexpensive to implement
  - data unit is transmitted with no timing coordination between sender and receiver



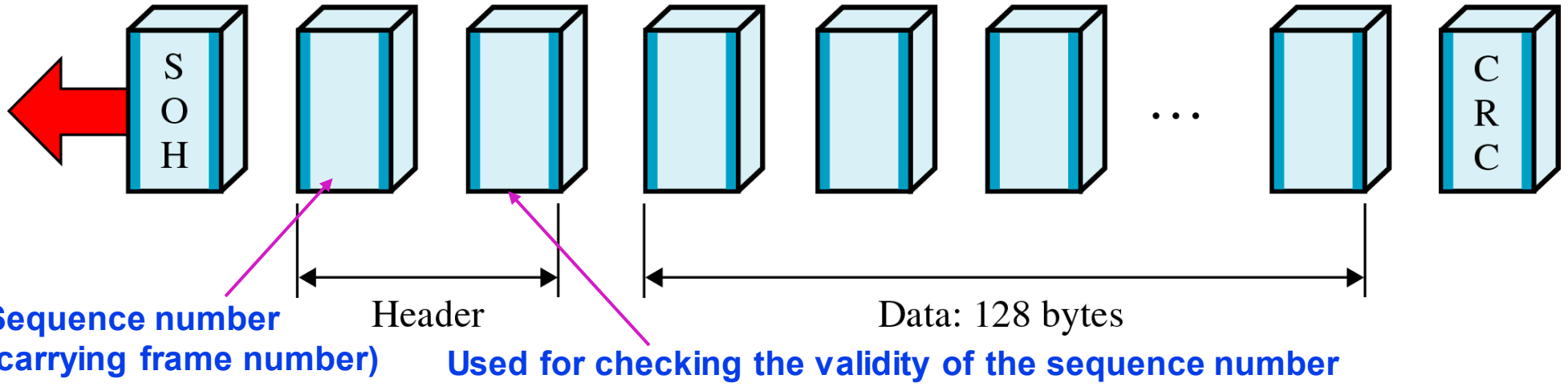
Asynchronous protocol(cont'd)

- Xmodem
  - Ward christiansen designed a file transfer protocol for telephone-line communication between PCs (1979)
  - half-duplex stop-and-wait ARQ protocol

Asynchronous protocol(cont'd)

XMODEM frame

Each character contains start and stop bits (dark portion of the box). Characters are separated from each other by gaps.
The header consists of two bytes: sequence number and its one's complement.



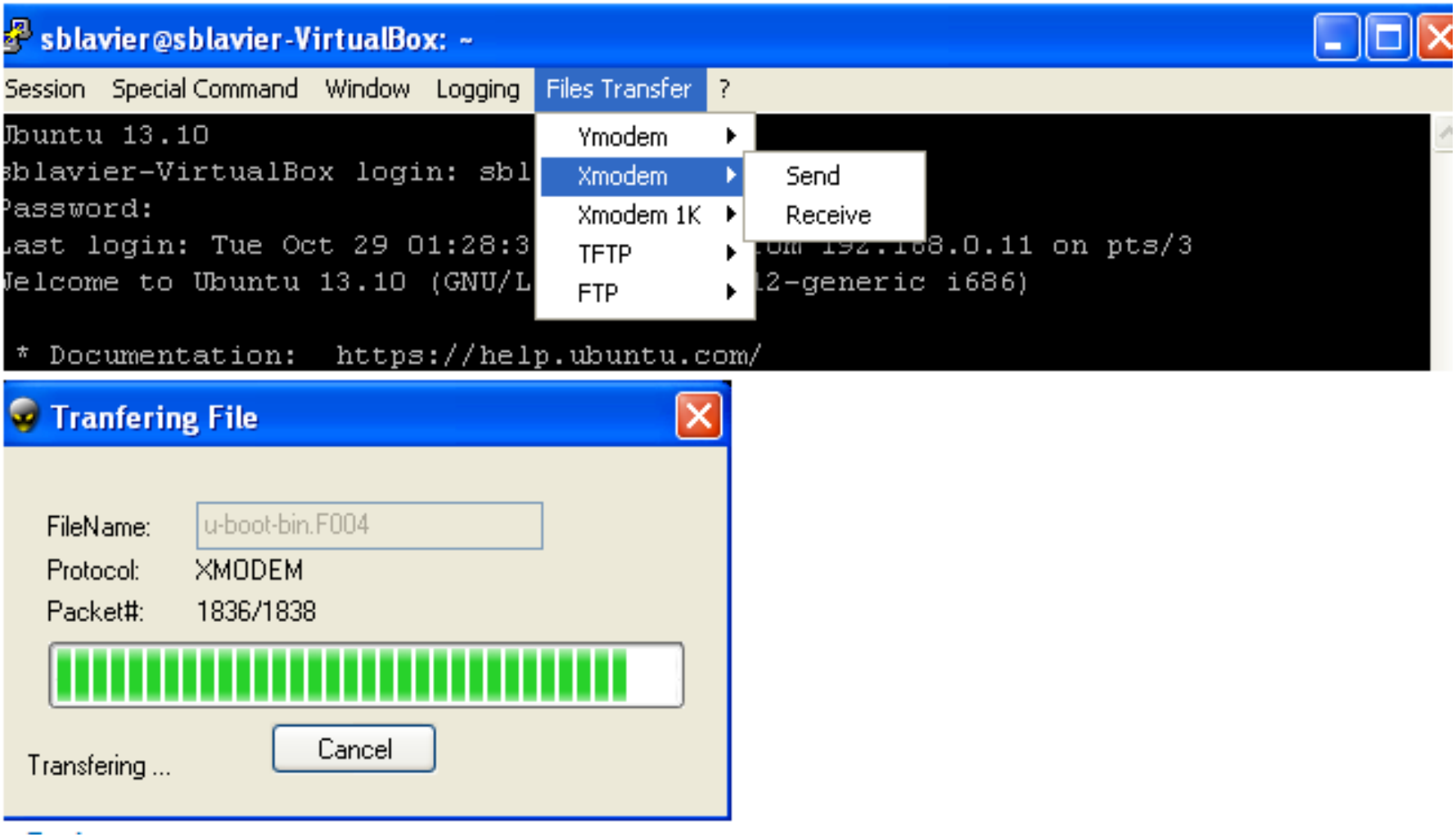
SOH : Start of Header

Asynchronous protocol(cont'd)

- XMODEM frame
  - SOH(start of Header) : 1 byte
  - Header : 2 bytes (as sequence number and for checking the validity of sequence number)
  - Data (Binary, ASCII, Boolean, Text, etc.) : 128 bytes
  - CRC : check for error in the data field

Asynchronous protocol(cont'd)

- XMODEM frame (cont'd)
  - Transmission begins with the sending of a NAK frame from the receiver to the sender.
    - Then, sender sends a frame
    - If seder receives ACK, sends next a frame
    - If it receives NAK, the previously sent frame is retransmitted
    - A frame can also be resent if a response is not received by the sender after a specified amount of time.
    - Besides a NAK or an ACK, the sender can receive a cancel signal (CAN), which aborts the transmission.

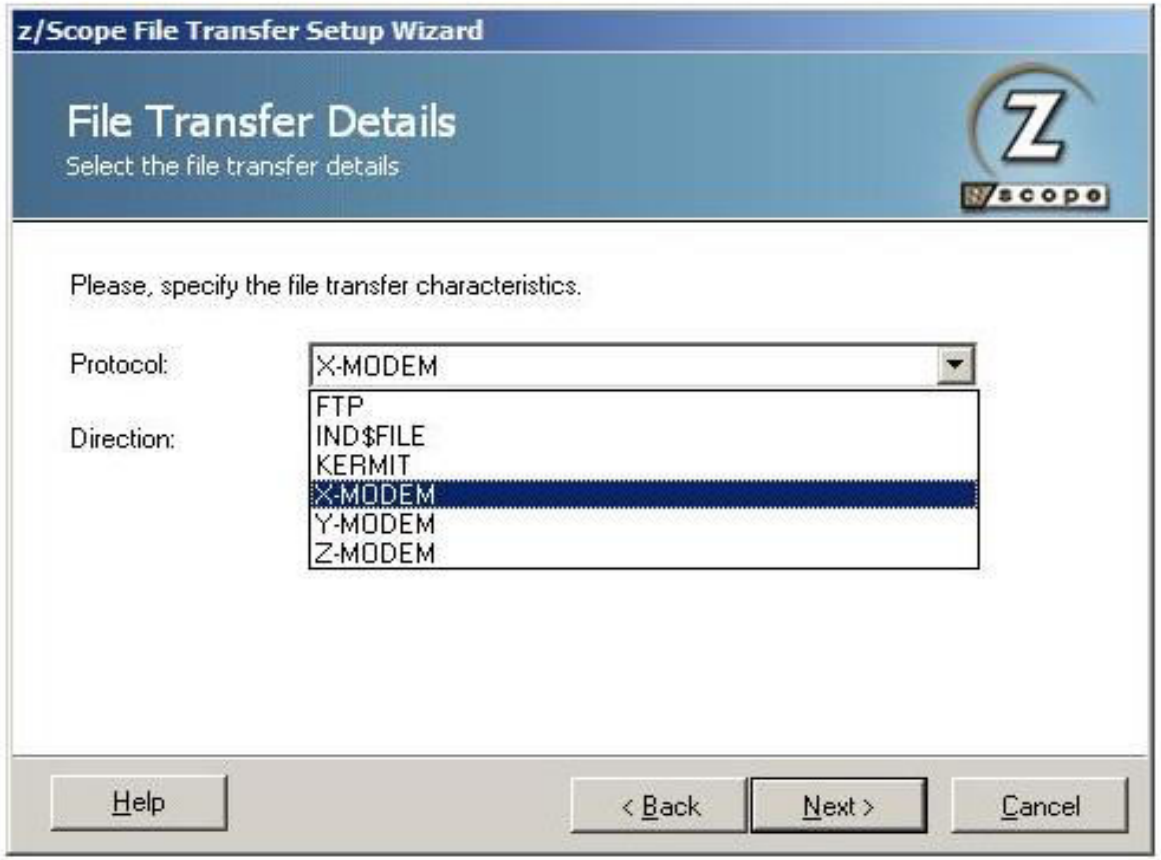

sblavier@sblavier-VirtualBox: ~
Session
Special Command
Window
Logging
Files Transfer
?
Ymodem
Xmodem
Xmodem 1K
TFTP
FTP
Send
Receive
Ubuntu 13.10
sblavier-VirtualBox login: sbl
Password:
Last login: Tue Oct 29 01:28:3
Welcome to Ubuntu 13.10 (GNU/L
* Documentation: https://help.ubuntu.com/
Tranfering File
FileName:
u-boot-bin.F004
Protocol:
XMODEM
Packet#:
1836/1838
Cancel
Transfering ...

https://www.cybelesoft.com/manuals/zScope/x-modem_filetrans_telnet.html

To add a new X-Modem File Transfer Job to the Queue, follow these steps:

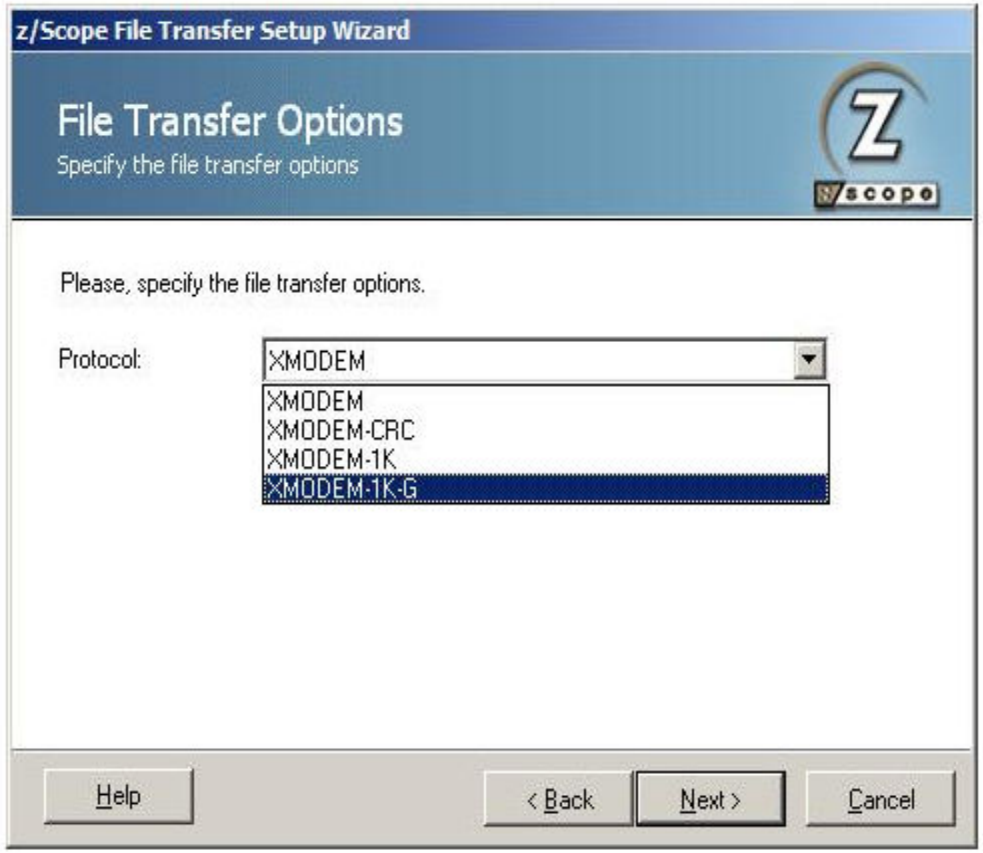
1. Click on the 'Add a New Transfer' button to launch the File Transfer Setup Wizard. Then click 'Next'.

2. Specify the file transfer characteristics. Set the protocol to X-MODEM. Then click 'Next'.



3. The Direction indicates if you are setting a download (RECEIVE) from the host to the PC, or an upload (SEND) from the PC to the host. You have the option to associate the file transfer with an existing connection. You can either select it or leave it blank according to your preferences. Then click 'Next'.

4. Choose from the combobox the specifical X-MODEM protocol you need to use: XMODEM, XMODEM-CRC, XMODEM-1K, XMODEM-1K-G. Then click 'Next'.



5. On the following screen you just need to specify the local file and click 'Finish'.

Once you finish the Setup Wizard the new file transfer job you specified will appear in the Static Queue.

In the future, whenever you need to change the File Transfer settings you can either click the 'Edit' button on the main menu or make a right click on the File to select the 'Edit' option.

Asynchronous protocol(cont'd)

- YMODEM

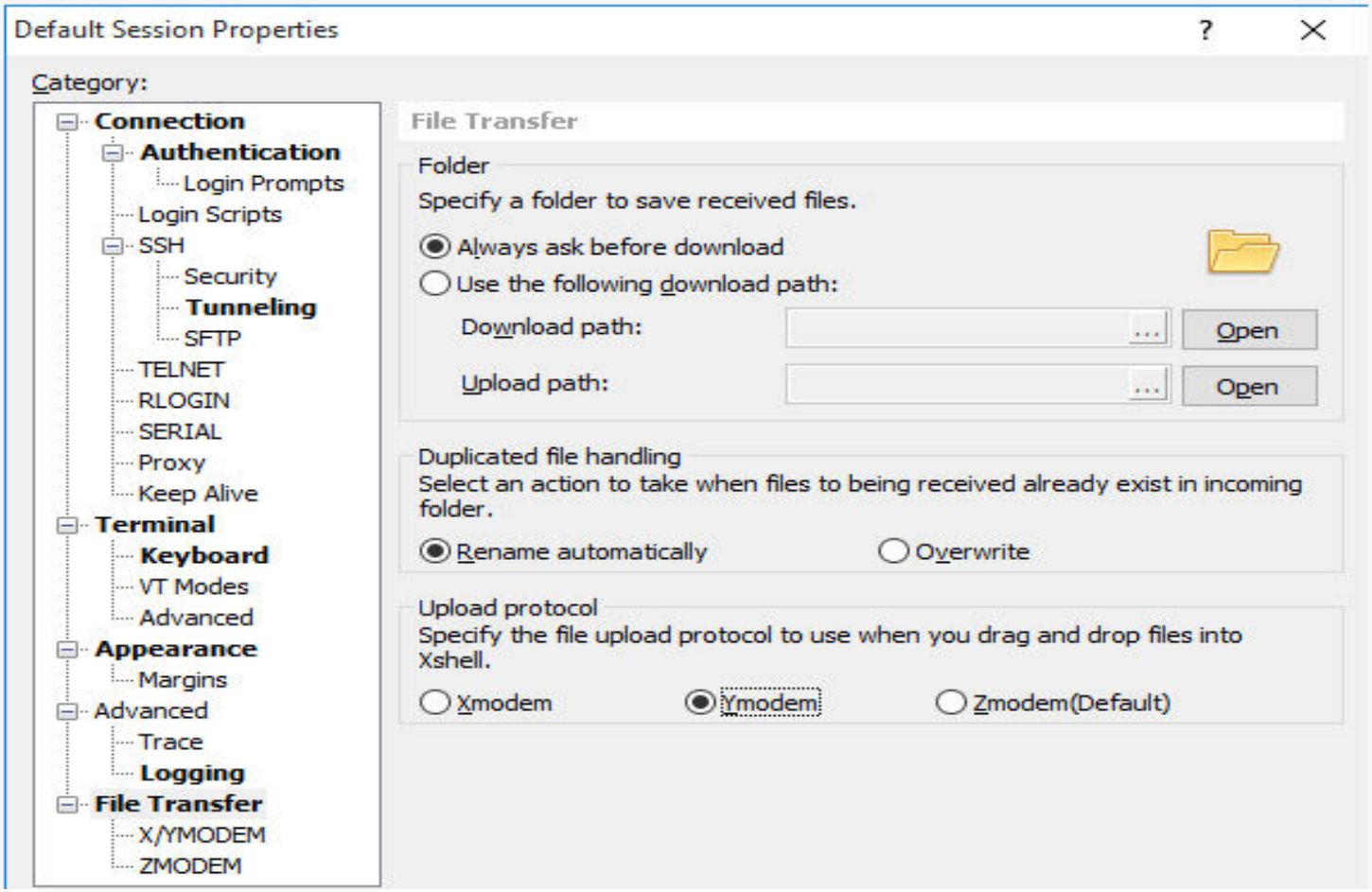
~ is a protocol similar to XMODEM

- major differences

  - data unit is 1024 bytes
  - Two CANs are sent to abort a transmission
  - ITU-T CRC-16 is used for error checking
  - Multiple files can be sent simultaneously

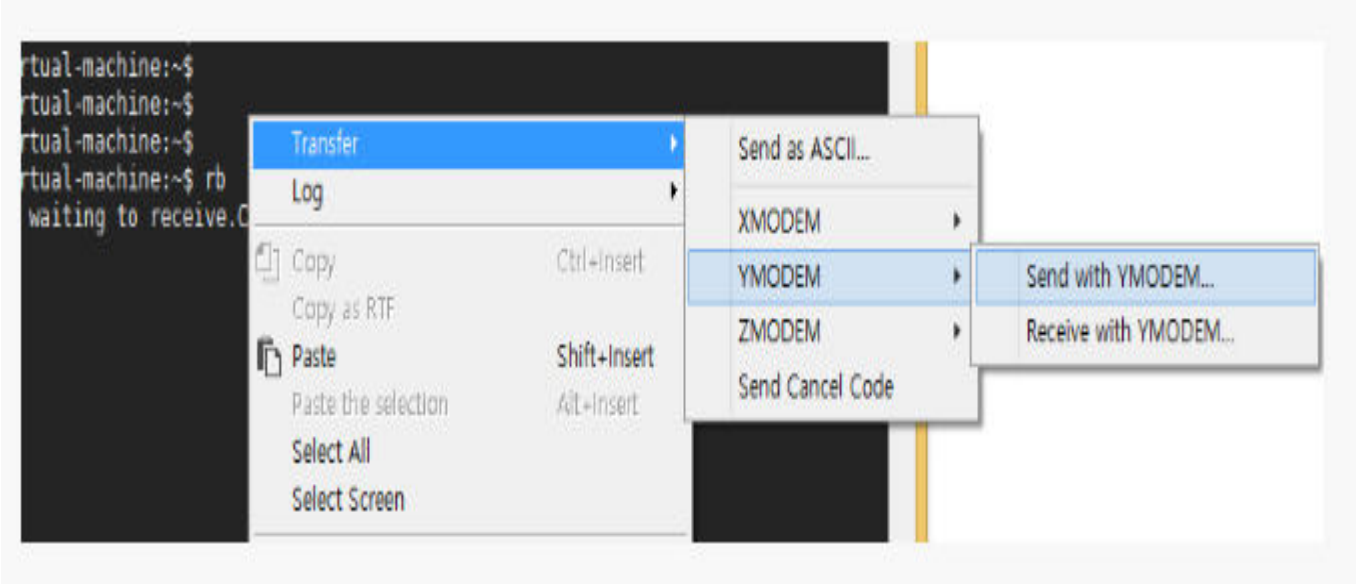
YMODEM is an asynchronous communication **protocol** for **modems** developed by Chuck Forsberg as a successor to Xmodem and Modem7. It supports batch file transfers and increases transfer block size, enabling the transmission of a whole list or batch of files at one time.

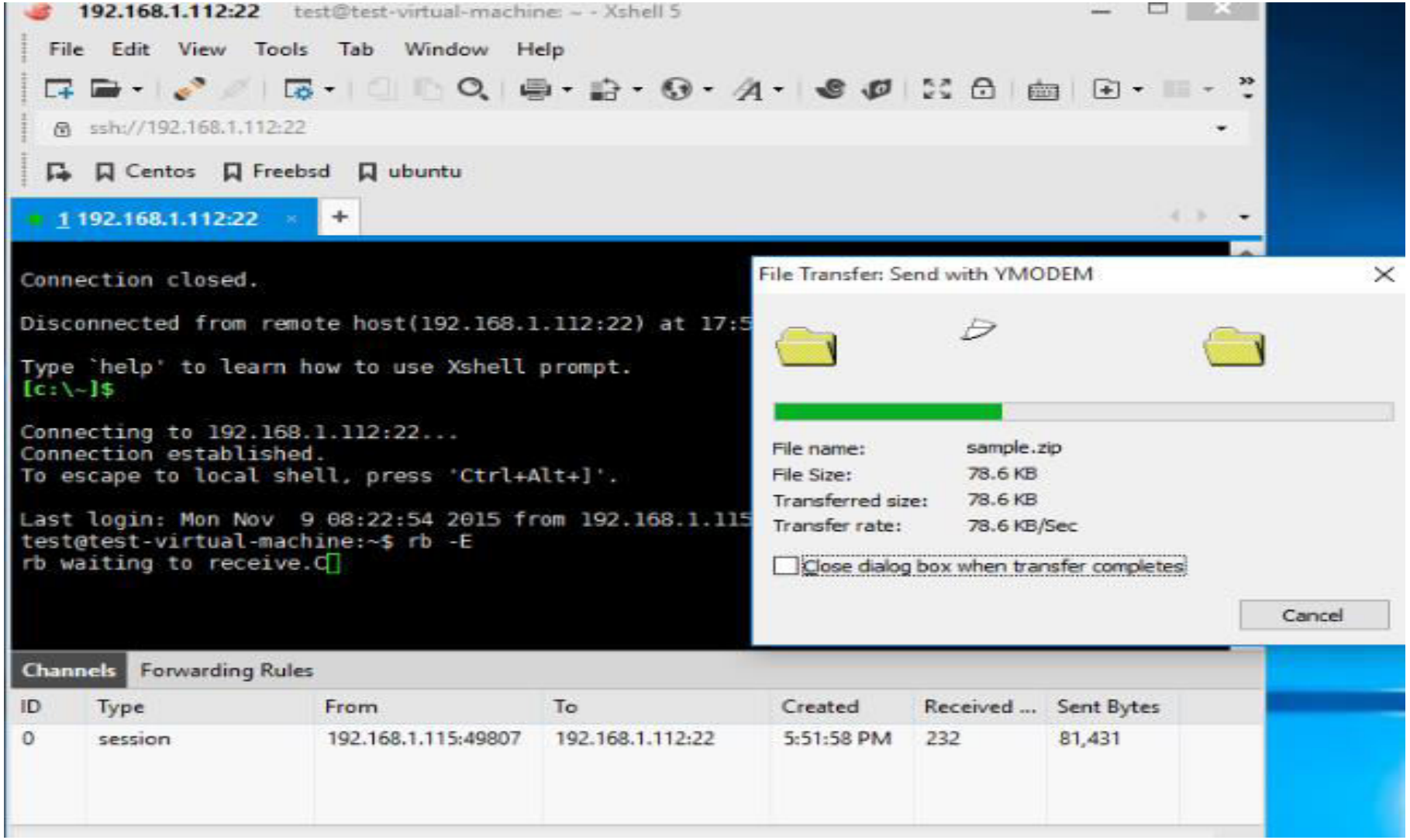
We've made some configurations to our File Transfer property sheet. Go to your session properties and select File Transfer to select your upload protocol:



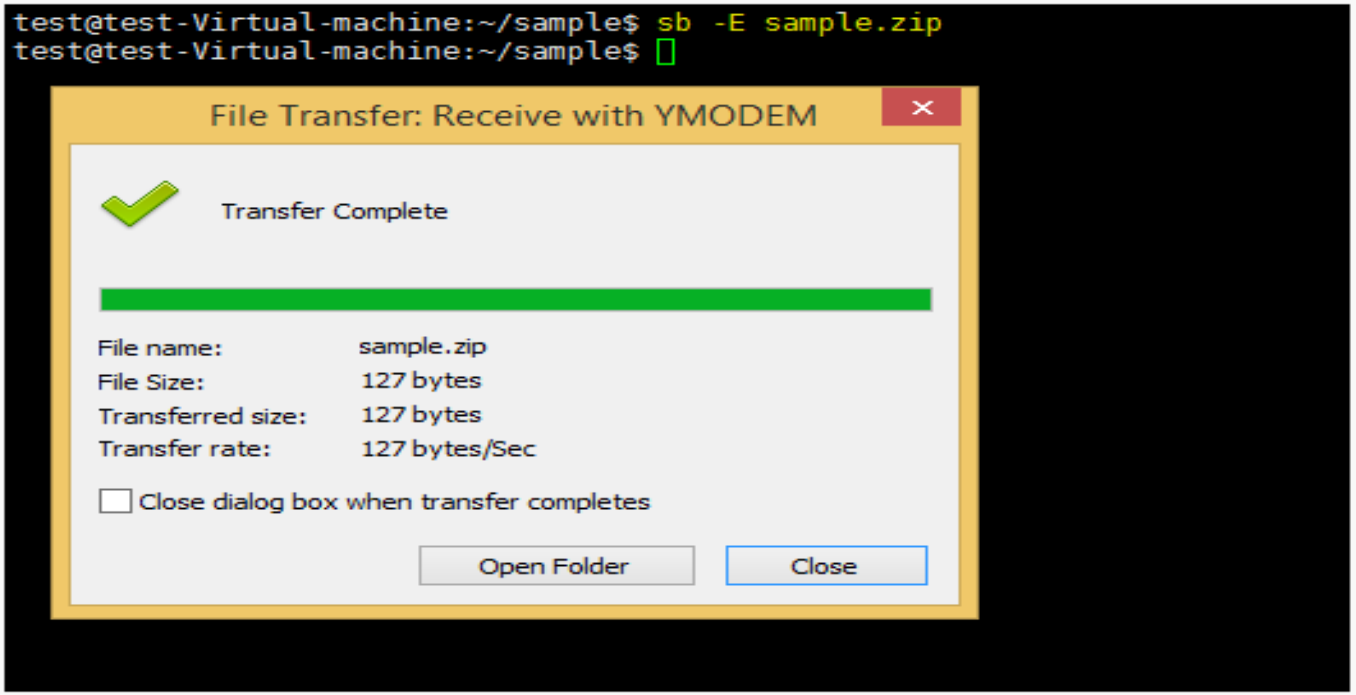
Go to X/YMODEM to select your packet size (1K or 128 Bytes). Here you can also set the upload commands. You'll also notice we've made changes to the context menu when right clicking within your Xshell session for user convenience.

Now let's try utilizing a Ymodem file upload. Make sure you've selected Ymodem as your upload protocol when dragging and dropping in session properties and connect to a host using SSH (Telnet is possible as well). Then navigate to your desired target directory. From here you'll need to type out the upload command as Ymodem (and Xmodem) doesn't have a trigger code. Type "rb -E" and select while file you would like to upload by navigating to "Send with YMODEM" in the context menu:




192.168.1.112:22 test@test-virtual-machine: ~ - Xshell 5
File Edit View Tools Tab Window Help
ssh://192.168.1.112:22
Centos Freebsd ubuntu
1 192.168.1.112:22
Connection closed.
Disconnected from remote host(192.168.1.112:22) at 17:5
Type `help' to learn how to use Xshell prompt.
[c:\~]$
Connecting to 192.168.1.112:22...
Connection established.
To escape to local shell, press 'Ctrl+Alt+]'.
Last login: Mon Nov 9 08:22:54 2015 from 192.168.1.115
test@test-virtual-machine:~$ rb -E
rb waiting to receive.C
File Transfer: Send with YMODEM
File name: sample.zip
File Size: 78.6 KB
Transferred size: 78.6 KB
Transfer rate: 78.6 KB/Sec
Close dialog box when transfer completes
Cancel
Channels Forwarding Rules
ID Type From To Created Received ... Sent Bytes
0 session 192.168.1.115:49807 192.168.1.112:22 5:51:58 PM 232 81,431

Now let's receive a file using Ymodem. Once again, you'll need to type the download command "sb -E 'filename'". Right click within your session and navigate to Transfer -> YMODEM -> Receive with YMODEM. Then browse to the folder you would like to save and hit OK. The file will begin downloading to your PC:



We've now done both a file send and receive utilizing the Ymodem protocol. Depending on your server requirements, you may have to use Xmodem or Ymodem downloads, but we recommend using the default Zmodem protocol.

Asynchronous protocol (cont'd)

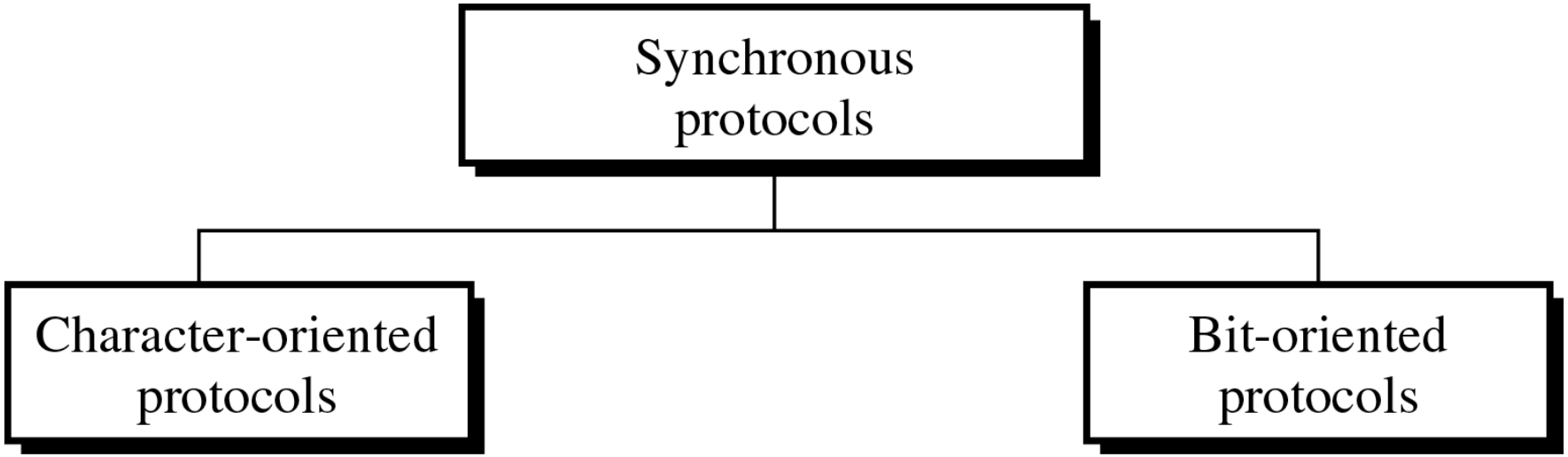
- ZMODEM
  - is a newer protocol combining features of both XMODEM and YMODEM
- BLAST(Blocked Asynchronous Transmission)
  - is full-duplex with sliding window flow control
  - is more powerful than XMODEM

Asynchronous protocol (cont'd)

- Kermit
  - file transfer protocol : similar in operation to XMODEM
  - is designed at Columbia University
  - is the most widely used asynchronous protocol

11.2 Synchronous protocol

- To be used for LAN, MAN, WAN



Synchronous protocol(cont'd)

- Character-oriented protocol (also called byte-oriented protocols)
  - frame or packet is interpreted as a series of characters

- Bit-oriented protocol
  - frame or packet is interpreted as a series of bits

11.3 Character-Oriented protocols

- are not as efficient as bit-oriented protocols and therefore are now seldom used

- a popular protocol : BSC(Binary synchronous communication) by IBM

Character-Oriented protocol(cont'd)

- BSC(Binary Synchronous Communication)
  - is developed by IBM in 1964
  - is usable in both point-to-point and multipoint configuration
  - supports half-duplex transmission using stop-and-wait ARQ flow control and error correction
  - does not support full-duplex transmission or sliding window protocol

Character-Oriented protocol(cont'd)

Control character for BSC

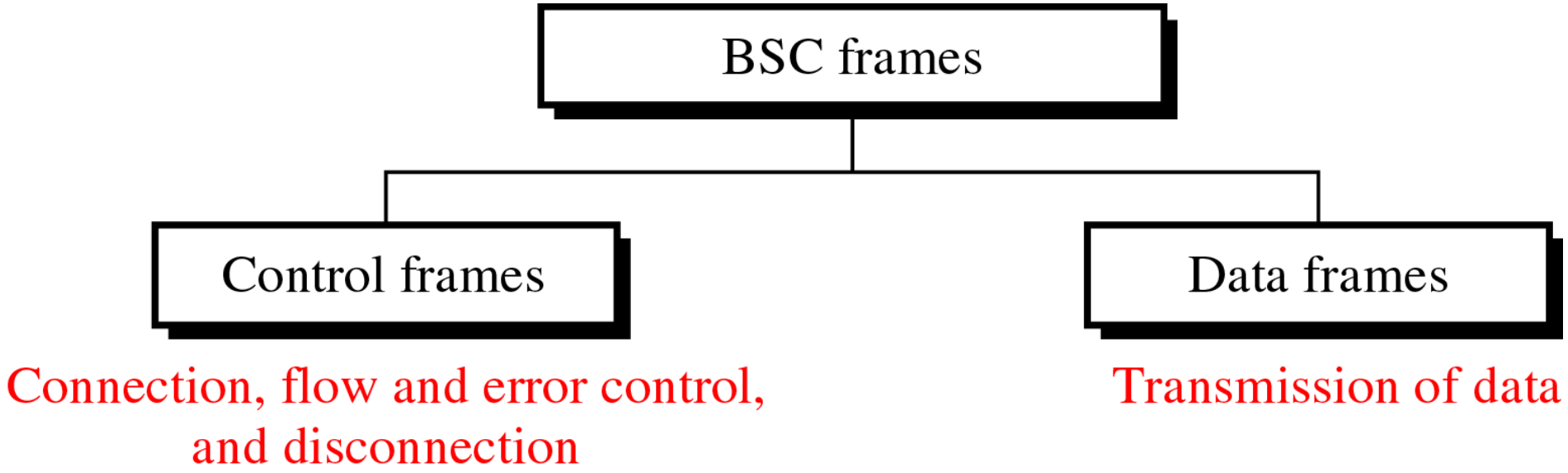
Character	ASCII Code	Function
ACK 0	DLE and 0	Good even frame received or ready to receive
ACK 1	DLE and 1	Good odd frame received
DLE	DLE	Data transparency maker
ENQ	ENQ	Request for a response
EOT	EOT	Sender terminating
ETB	ETB	End of transmission block; ACK required
ETX	ETX	End of text in a message
ITB	US	End of intermediate block in a multiblock transmission
NAK	NAK	Bad frame received nothing to send
NUL	NULL	Filler character
RVI	DLE and <	Urgent message from receiver
SOH	SOH	Header information begins
STX	STX	Text begins
SYN	SYN	Alerts receiver to incoming frame
TTD	STX and ENQ	Sender is pausing but not relinquishing the line
WACK	DLE and ;	Good frame received but not ready to receive more

Character-Oriented protocol(cont'd)

- ASCII codes
  - whatever the system, not all control characters can be represented by a single character. Often they must be represented by two or three characters

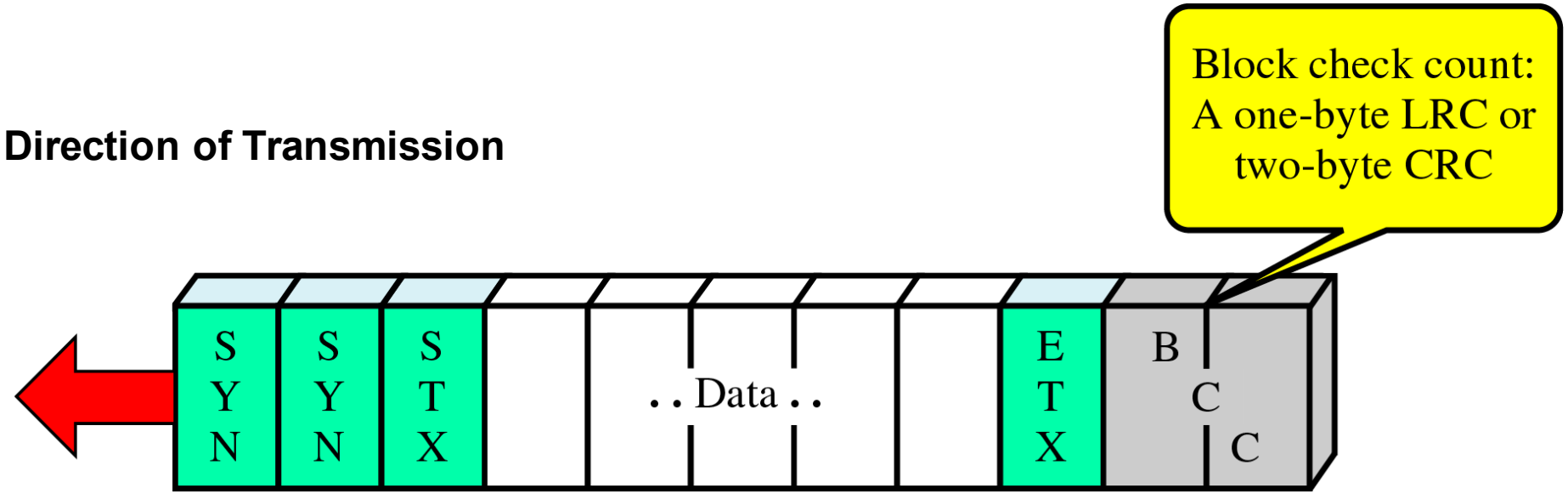
Character-Oriented protocol(cont'd)

- **BSC frame**



Character-Oriented protocol(cont'd)

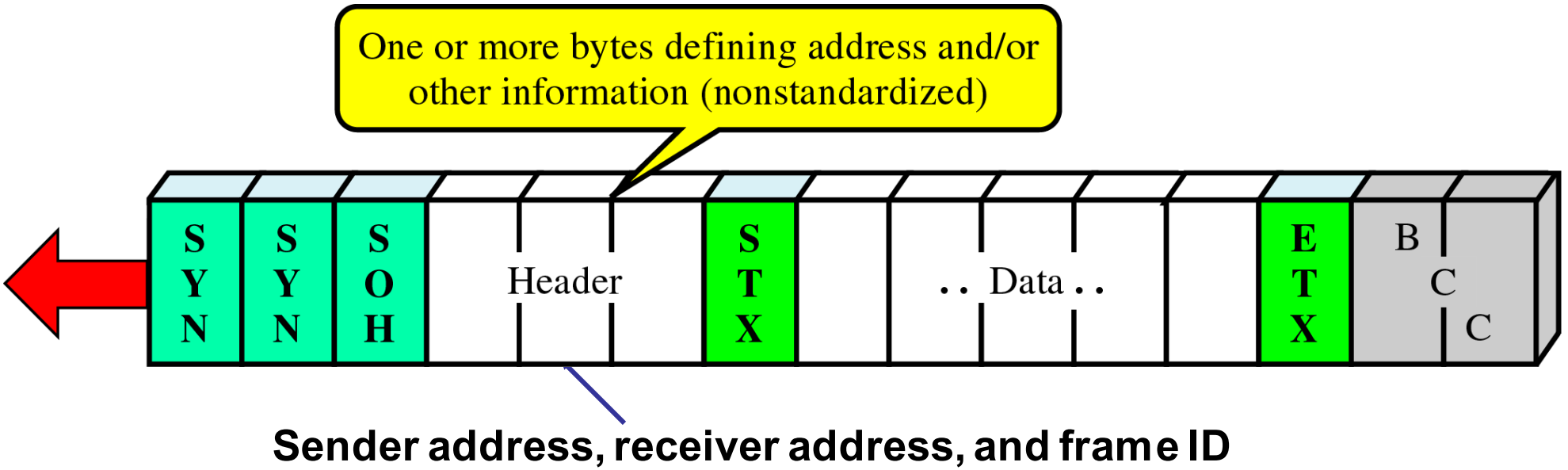
- Data frame



SYN : 0010110 as ASCII
Usually, 00010110 (adding 0 at eighth bit)

Character-Oriented protocol(cont'd)

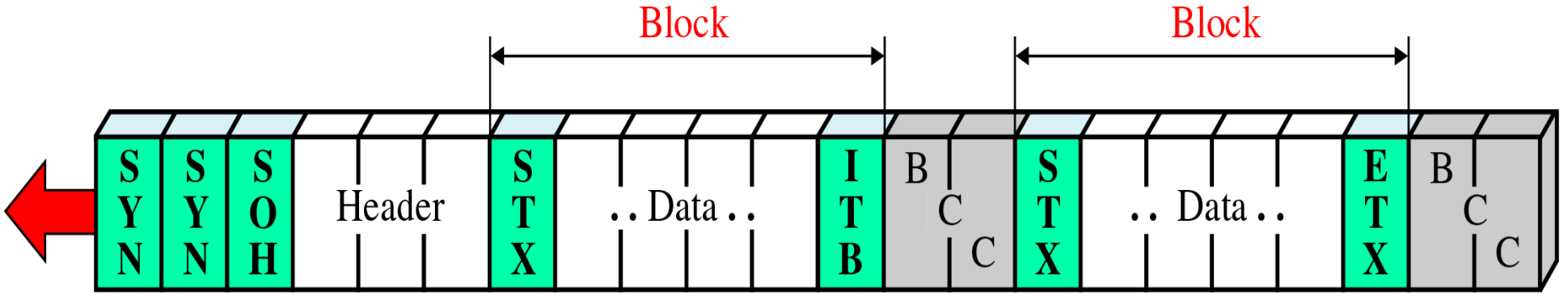
- Header field



Character-Oriented protocol(cont'd)

Multiblock Frame

- text in a message is often divided between several blocks

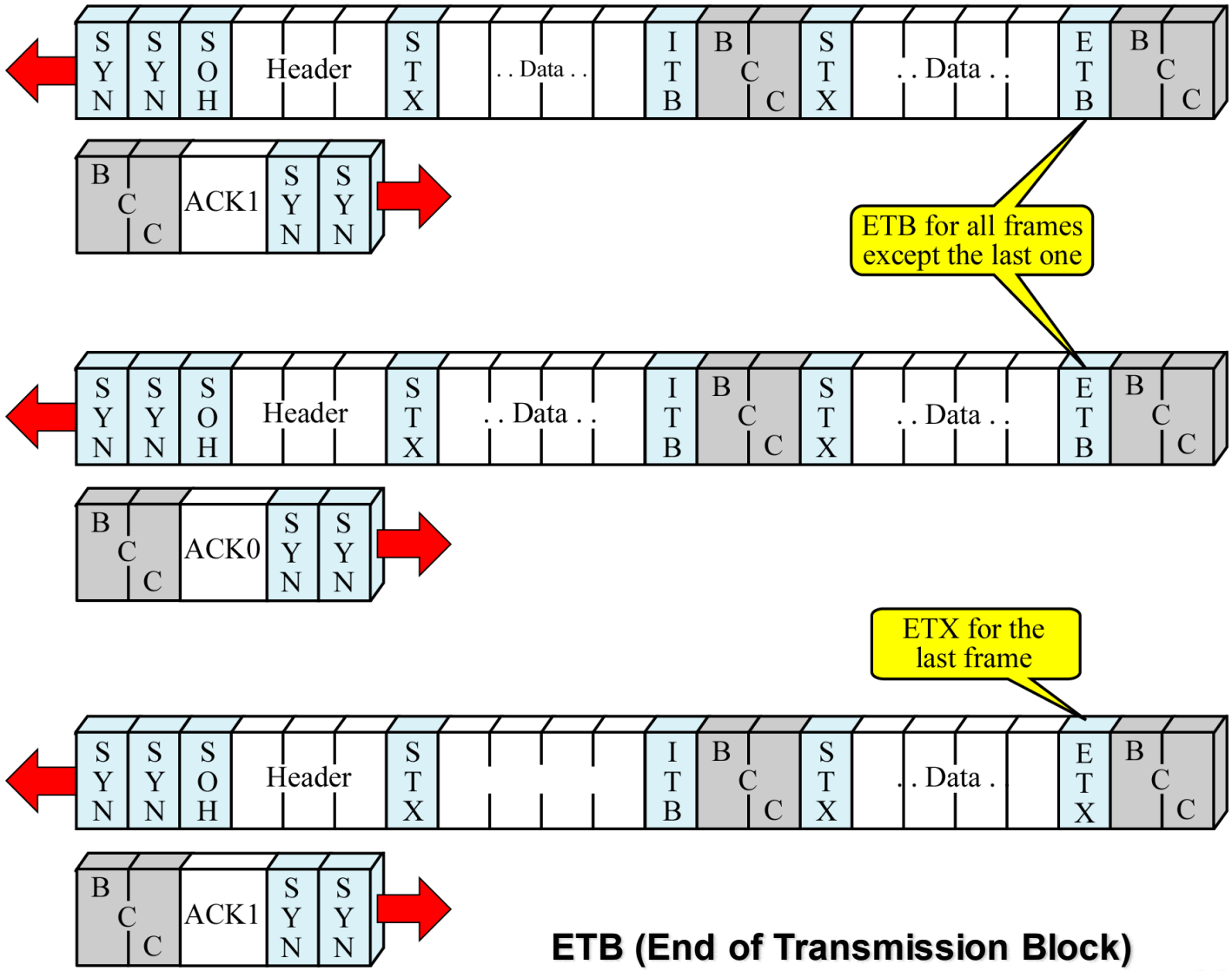


- The receiver sends a single acknowledgment for the entire frame

ITB (Intermediate Text Block)

Character-Oriented protocol(cont'd)

Multiframe Transmission

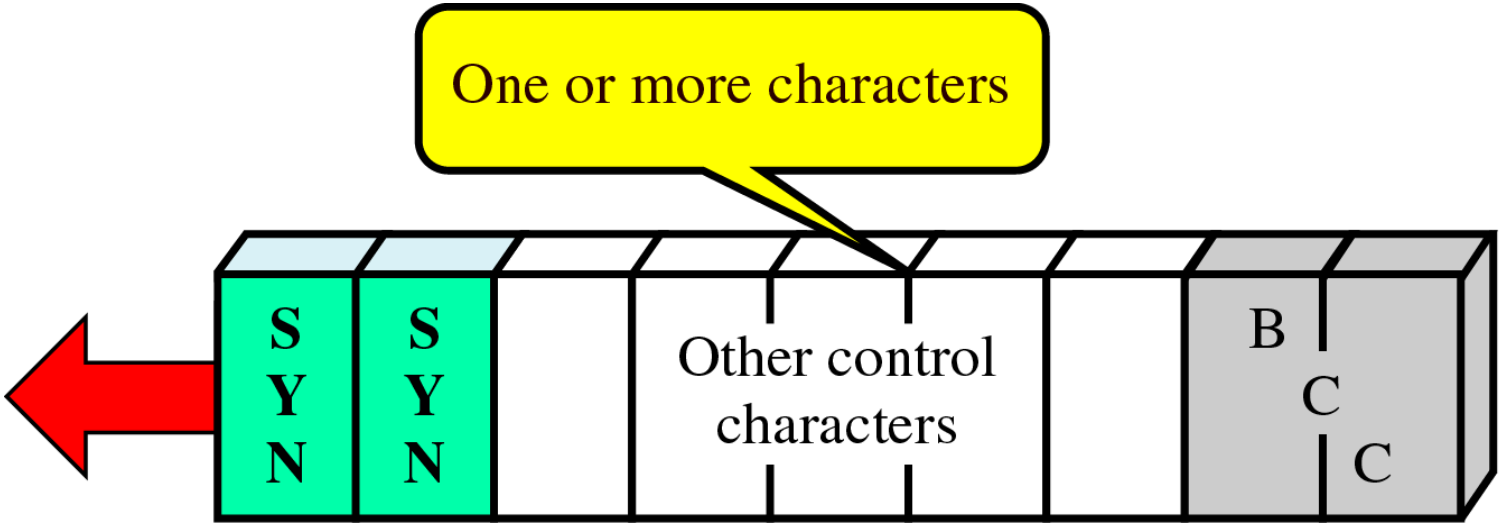


ETB (End of Transmission Block)

Character-Oriented protocol(cont'd)

- **Control Frame**

  ~ is used by one device to send commands to, or solicit information from, another device

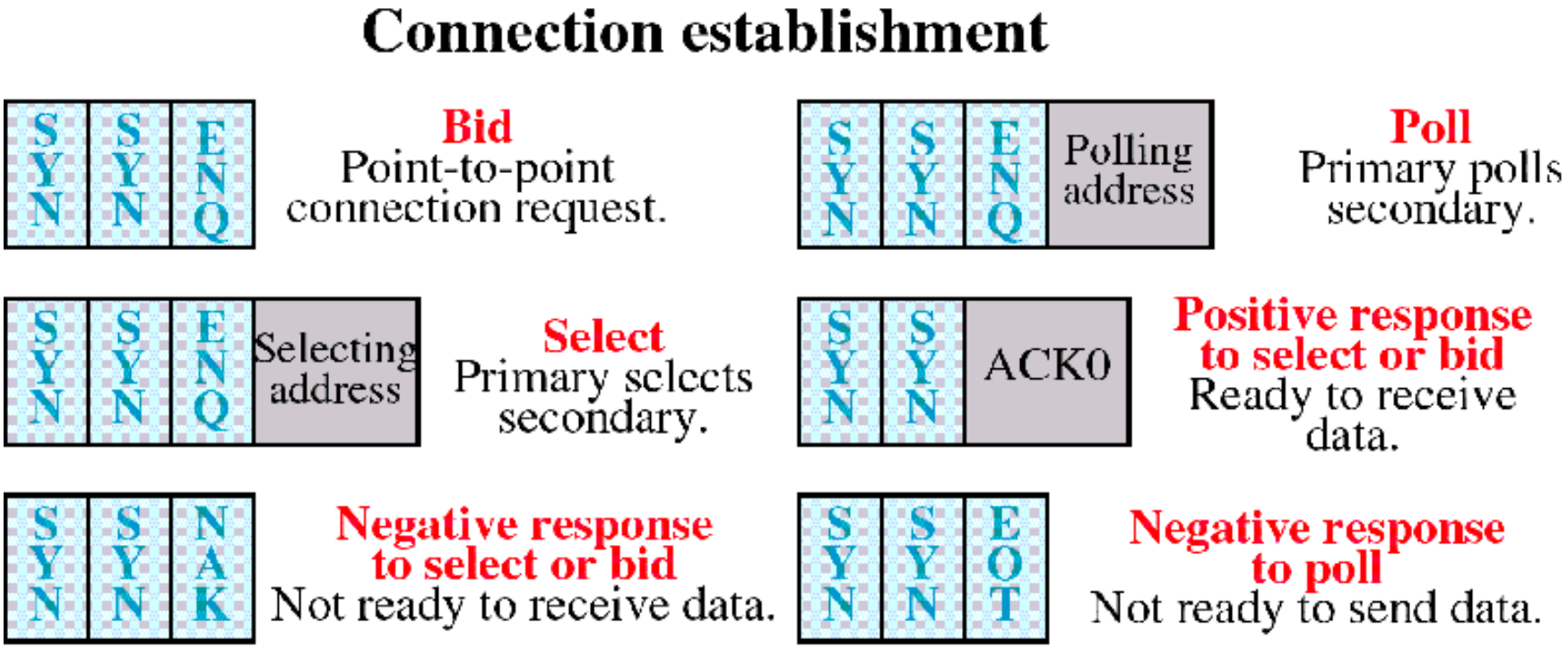


Character-Oriented protocol(cont'd)

- Control Frames serve three purpose
  - establishing connections
  - maintaining flow and error control during data transmission
  - terminating connection

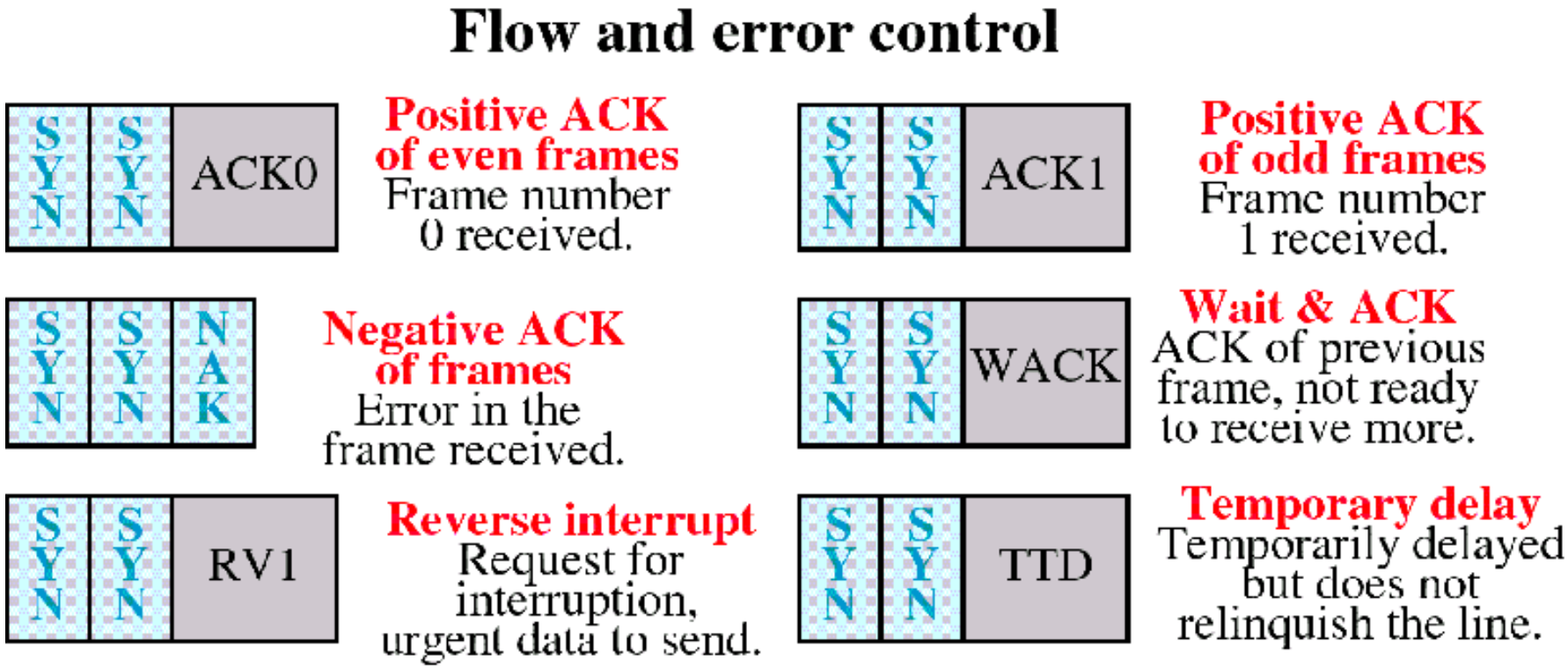
Character-Oriented protocol(cont'd)

- Control Frame(1)



Character-Oriented protocol(cont'd)

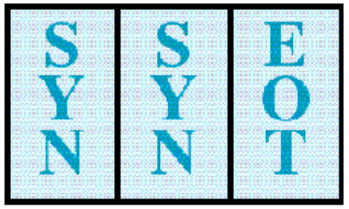
- Control Frame(2)



Character-Oriented protocol(cont'd)

- Control Frame(3)

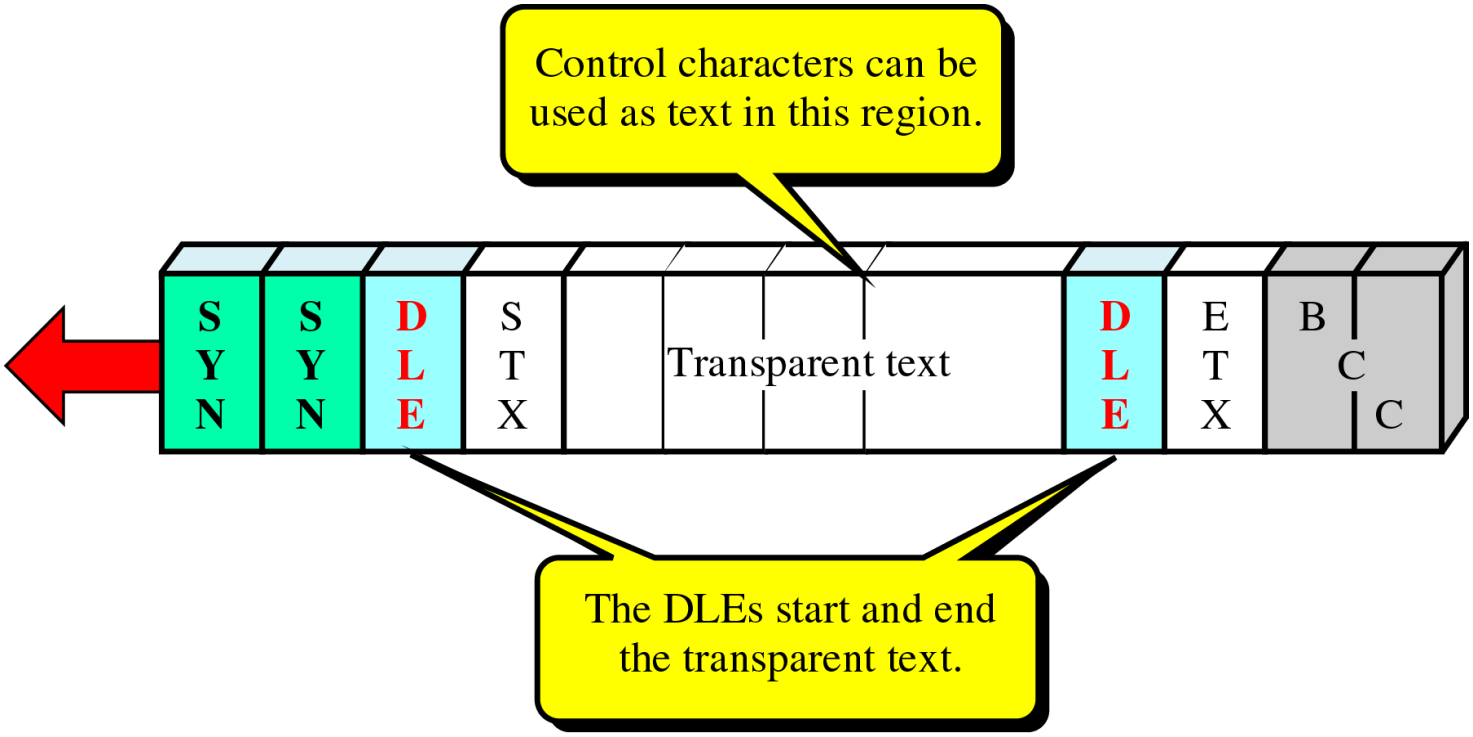
Connection termination



End of transmission
Station finished sending data.

Character-Oriented protocol(cont'd)

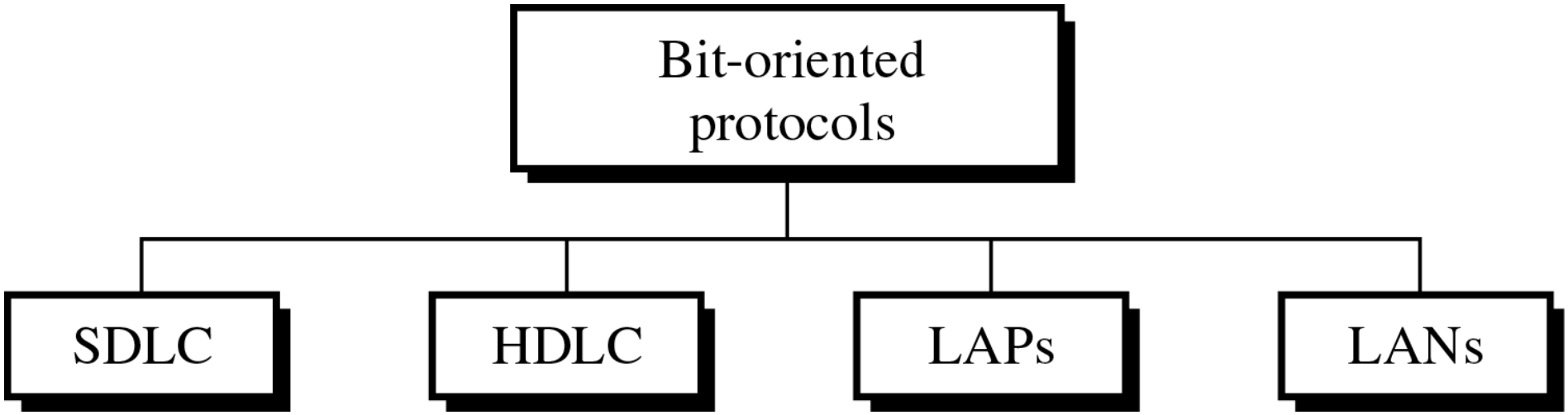
- Data Transparency
  - Confusion between control information and data is called a lack of data transparency
  - means we should be able to send any combination of bits as data (byte stuffing)



For example,
ETX : 0000011 as DATA
DLE : Data Link Escape

11.4 Bit-Oriented protocol

- can pack more information into shorter frames and avoid the transparency problem of character-oriented protocol



Bit-Oriented protocol(cont'd)

- SDLC(Synchronous Data Link Control)
  - developed by IBM in 1975
- HDLC(High-Level Data Link Control)
  - developed by ISO in 1979
- LAPs (LAPB, LAPD, LAPM, LAPX, etc)
  - developed by ITU-T since 1981
  - based on HDLC
- PPP, frame relay
  - developed by ITU-T and ANSI
  - based on HDLC

Bit-Oriented protocol - HDLC (cont'd)

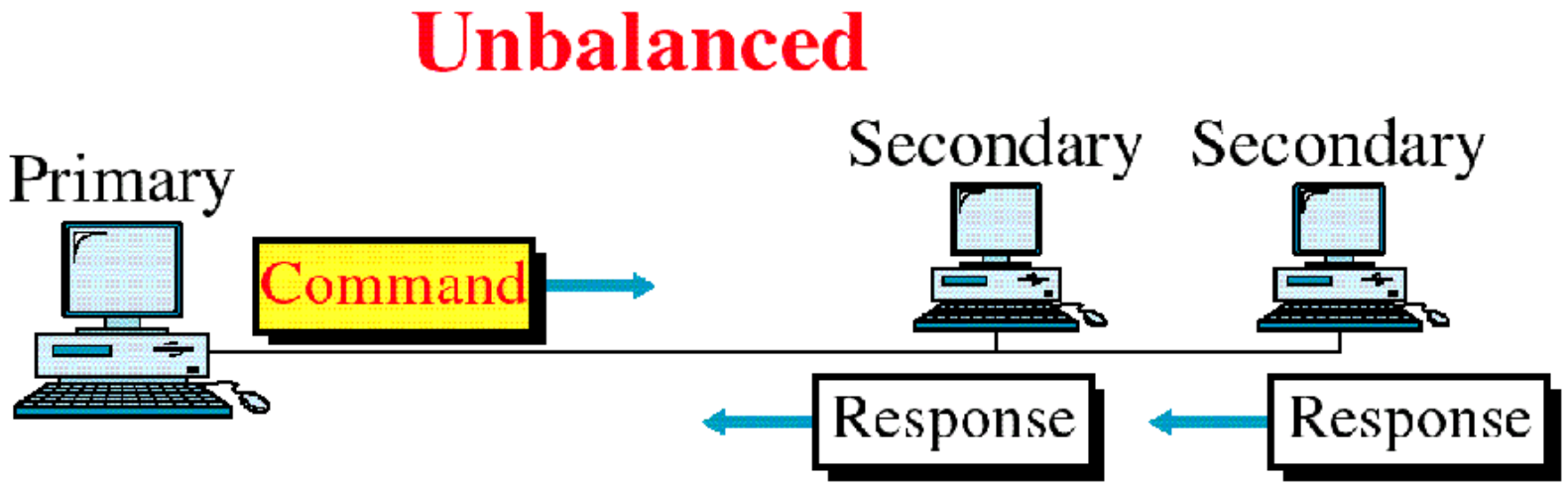
- HDLC
  - All bit-oriented protocols are related to high-level data link control(HDLC), which published by ISO.
  - HDLC supports both half-duplex and full-duplex modes in point-to-point and multipoint configurations
  - HDLC can be characterized by their station types, their configurations, and their response modes

Bit-Oriented protocol - HDLC (cont'd)

- Station Types
  - primary : send commands
  - secondary : send response
  - combined : send command and response

Bit-Oriented protocol - HDLC (cont'd)

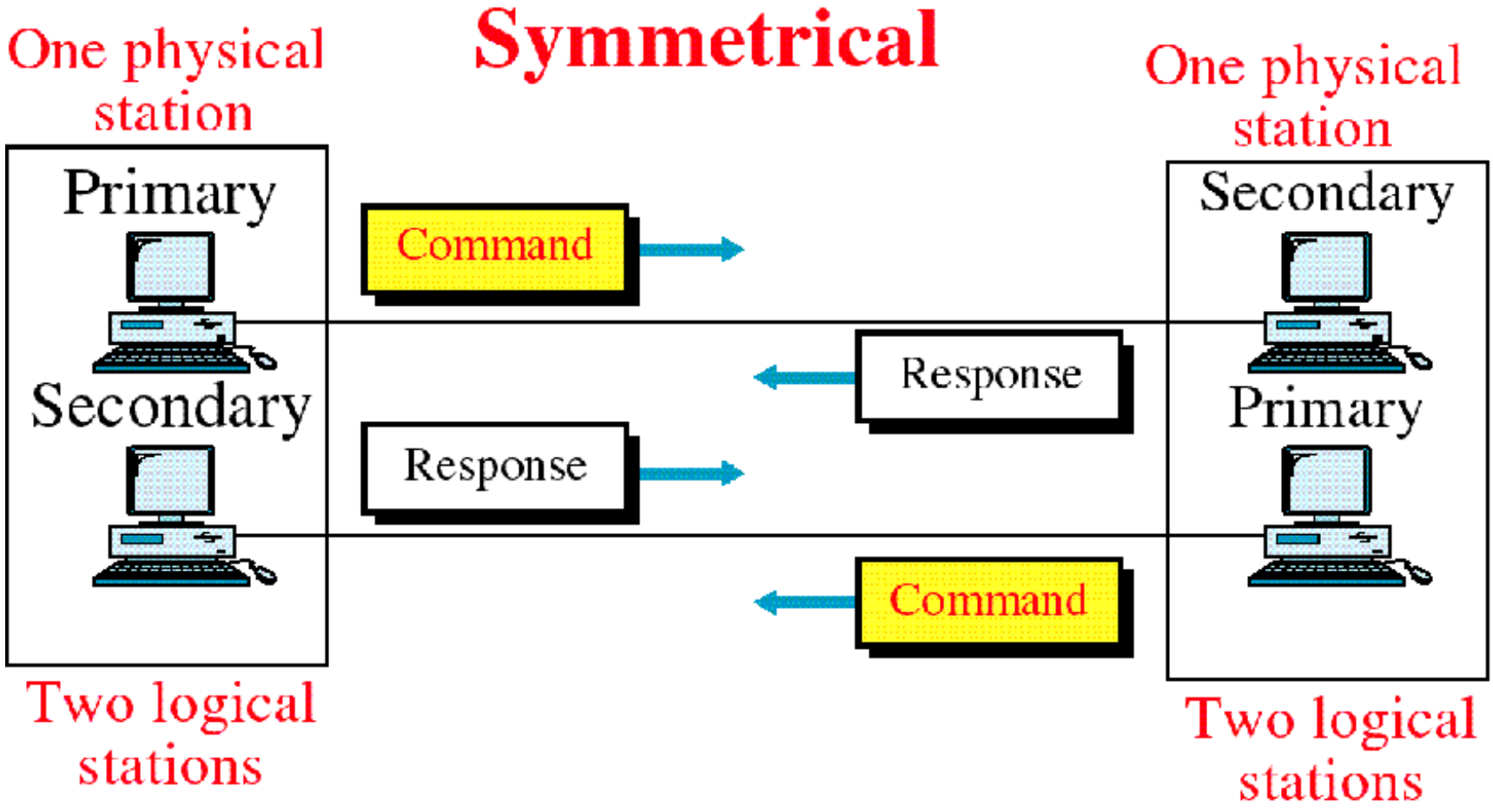
- Configurations
  - refer to the relationship of hardware devices on a link
  - Point-to-point or point-to-multipoint



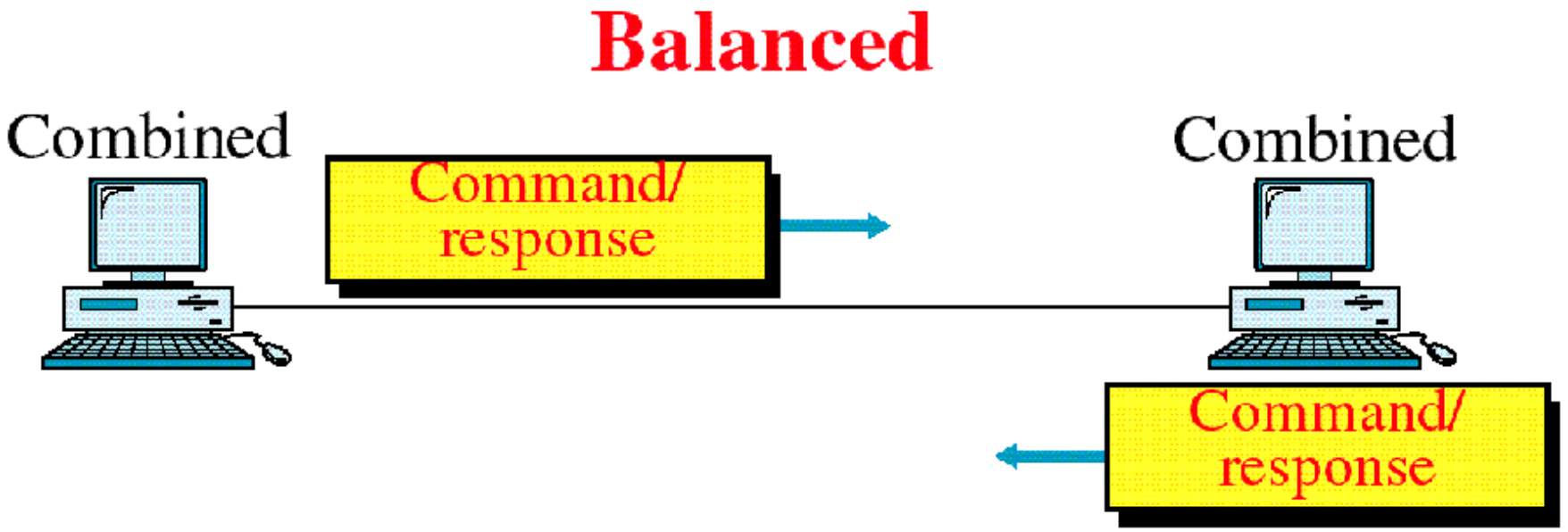
Master/slave configuration

Bit-Oriented protocol - HDLC (cont'd)

- Like an unbalanced mode except that control of the link can shift between the two stations



Bit-Oriented protocol - HDLC (cont'd)



- HDLC does not support balanced multipoint. This necessitated the invention of media access protocols for LANs

Bit-Oriented protocol - HDLC (cont'd)

- A mode in HDLC is the relationship between two devices involved in an exchange; The mode of communication describes who controls the link
- HDLC supports three modes of communication between stations
  - NRM(Normal Response Mode)
  - ARM(Asynchronous Response Mode)
  - ABM(Asynchronous Balanced Mode)

Bit-Oriented protocol - HDLC (cont'd)

- NRM(Normal Response Mode)
  - refers to the standard primary-secondary relationship
  - secondary device must have permission from the primary device before transmitting

Bit-Oriented protocol - HDLC (cont'd)

- ARM(Asynchronous Response Mode)
  - secondary may initiate a transmission without permission from the primary whenever the channel is idle
  - does not alter the primary-secondary relationship in any other way
  - All transmission from a secondary (even to another secondary on the same link) must still be made to the primary for relay to a final destination.

Bit-Oriented protocol - HDLC (cont'd)

- ABM(Asynchronous Balanced Mode)
  - all stations are equal and therefore only combined stations connected in point-to-point are used
  - Either combined station may initiate transmission with the other combined station without permission

Bit-Oriented protocol - HDLC (cont'd)

- HDLC modes

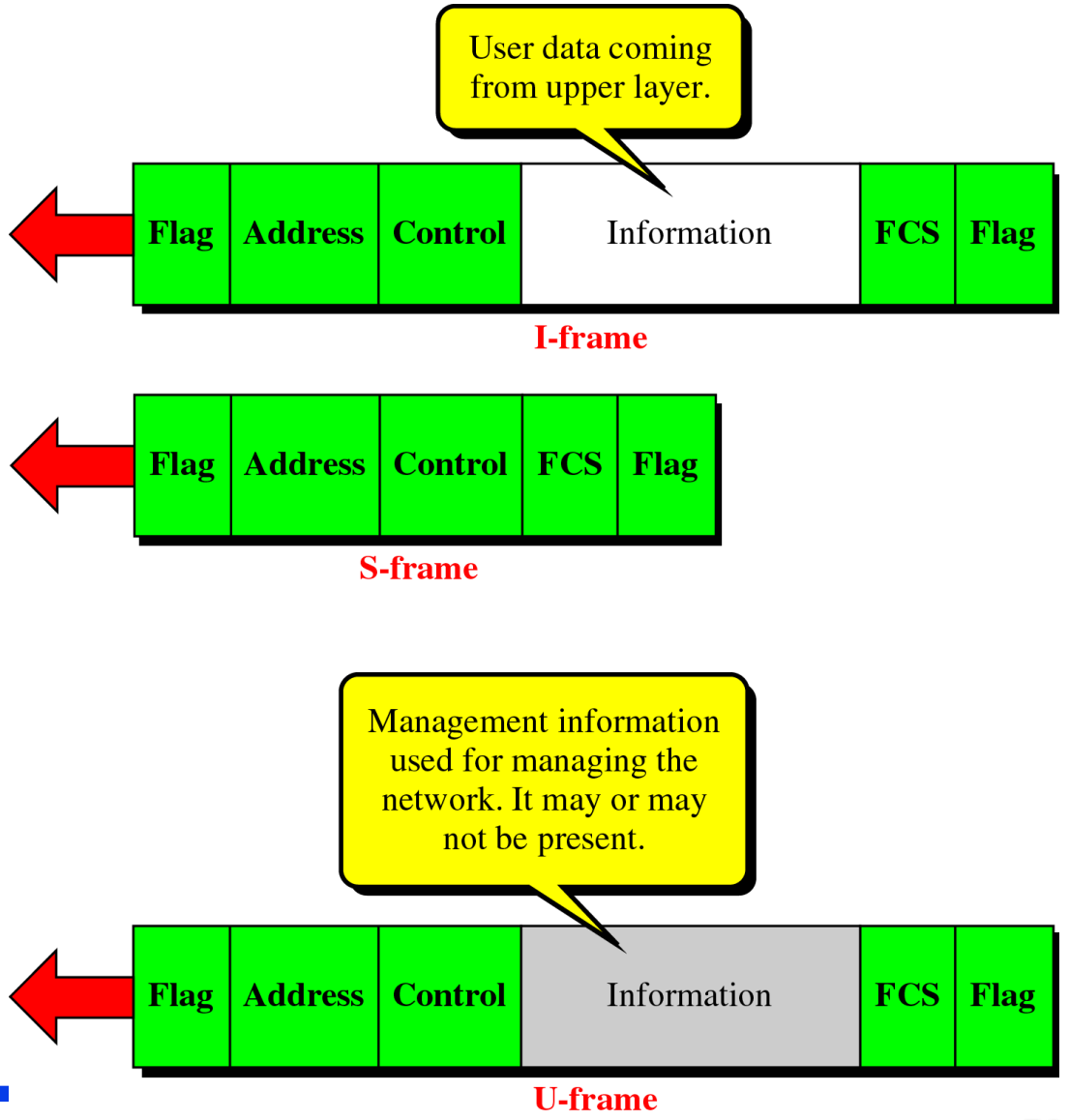
	NRM	ARM	ABM
Station type	Primary & secondary	Primary & secondary	Combined
Initiator	Primary	Either	Any

Bit-Oriented protocol - HDLC (cont'd)

- Frame
  - I (Information) Frame
    - used to transport user data and control information relating to user data
  - S (Supervisory) Frame
    - used to only to transport control information, primarily data link layer flow and error controls
  - U (Unnumbered) Frame
    - is reserved for system management
    - Information carried by U-frame is intended for managing the link itself

Bit-Oriented protocol - HDLC (cont'd)

- HDLC Frame types



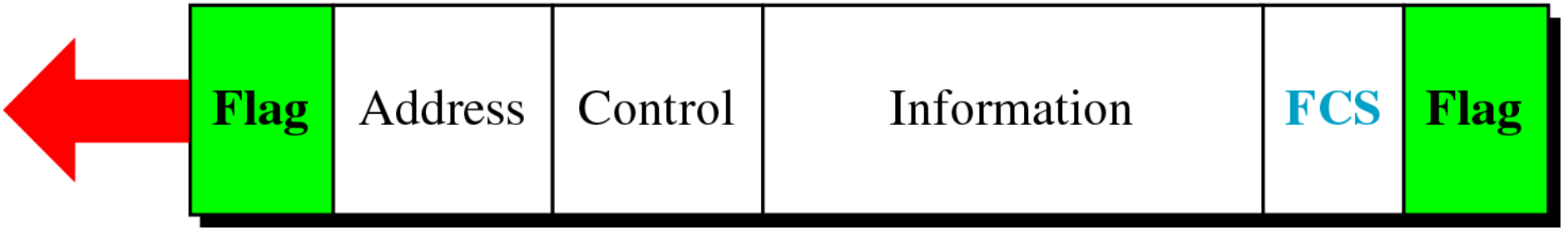
Bit-Oriented protocol - HDLC (cont'd)

- Frame may contain up to six fields
  - beginning flag
  - address
  - control
  - information
  - FCS(Frame Check Sequence)
  - Ending flag

Bit-Oriented protocol - HDLC (cont'd)

- Flag Field serves as a synchronization pattern for the receiver

The flag is 8 bits of a fixed pattern.
It is made of 6 ones enclosed in 2 zeros.
There is 1 flag at the beginning and 1 at the end of the frame. The ending flag of 1 frame can be used as the beginning flag of the next frame.
01111110



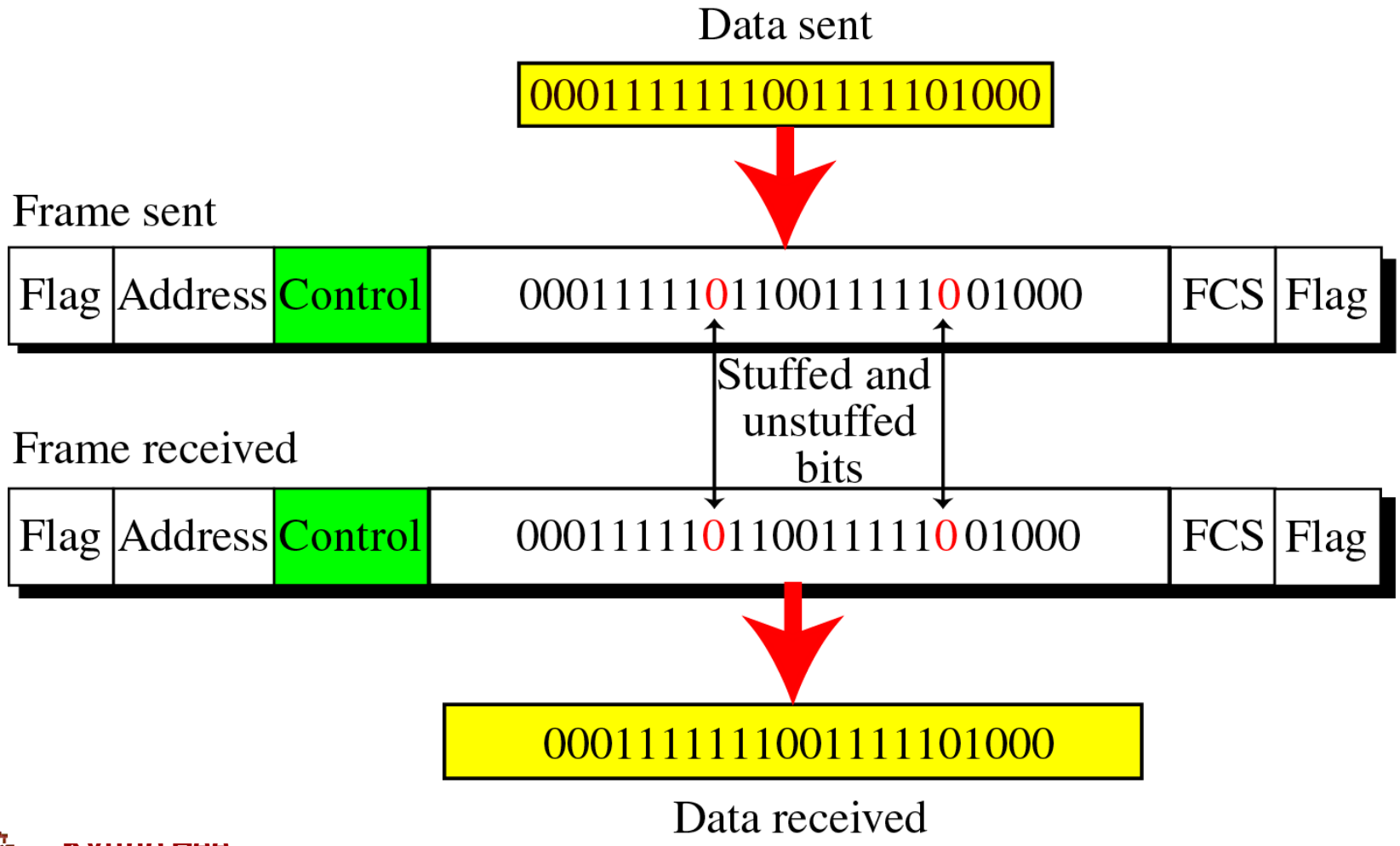
Bit-Oriented protocol - HDLC (cont'd)

- **Bit stuffing**
  - the process of adding one extra 0 whenever there are five consecutive 1s in the data so that the receiver does not mistake the data for flag

ex) 011111111000 --> 0111110111000

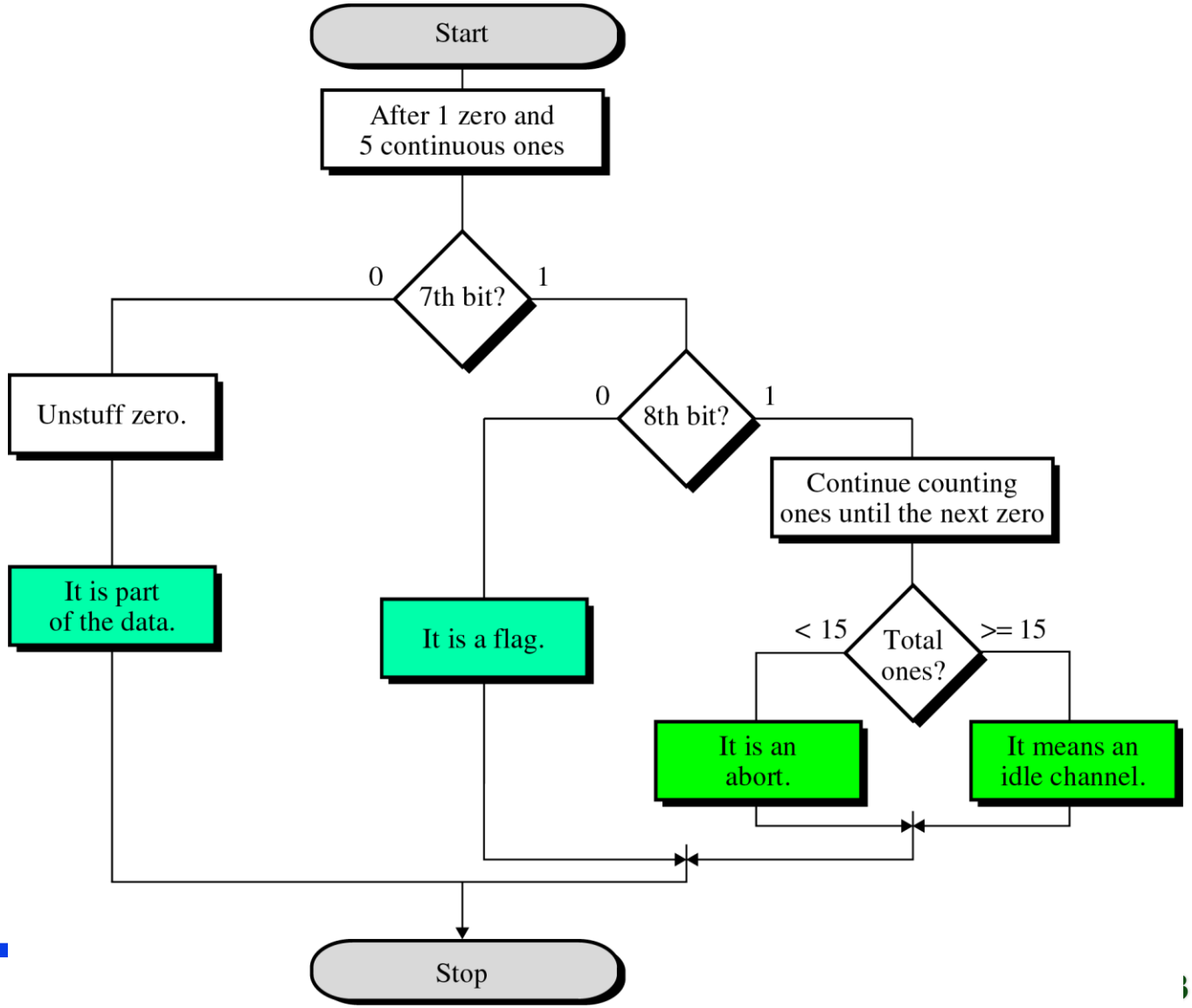
Bit-Oriented protocol - HDLC (cont'd)

- **Bit stuffing (cont'd)**



Bit-Oriented protocol - HDLC (cont'd)

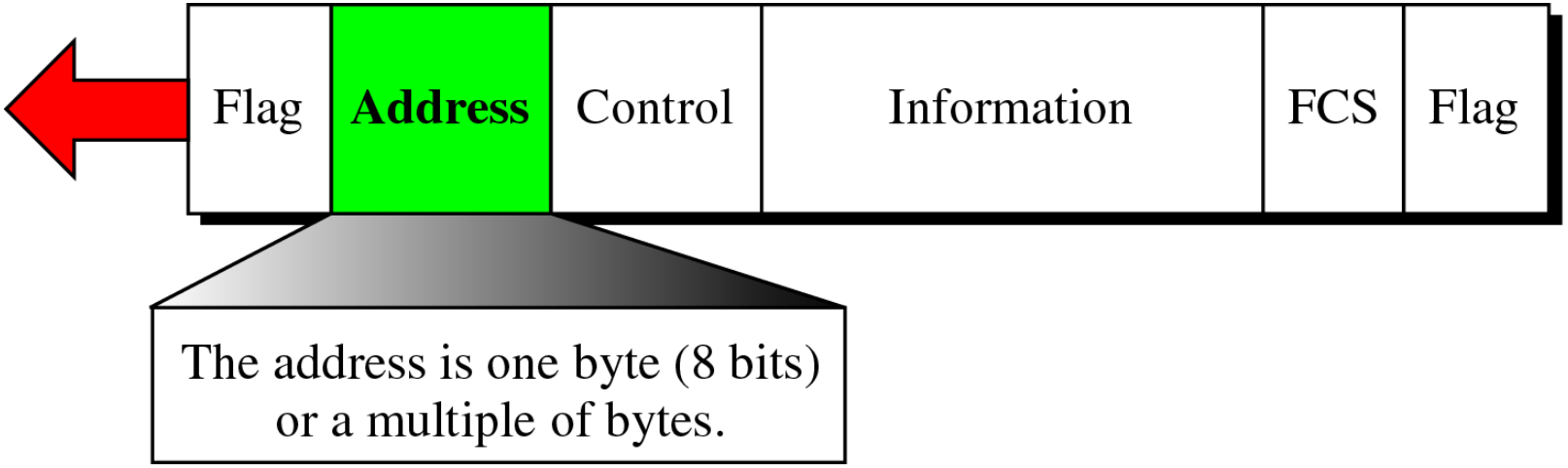
- Bit Stuffing in HDLC (at the receiver side)



Bit-Oriented protocol - HDLC (cont'd)

Address Field

~ contains the address of the secondary station that is either the originator or destination of the frame

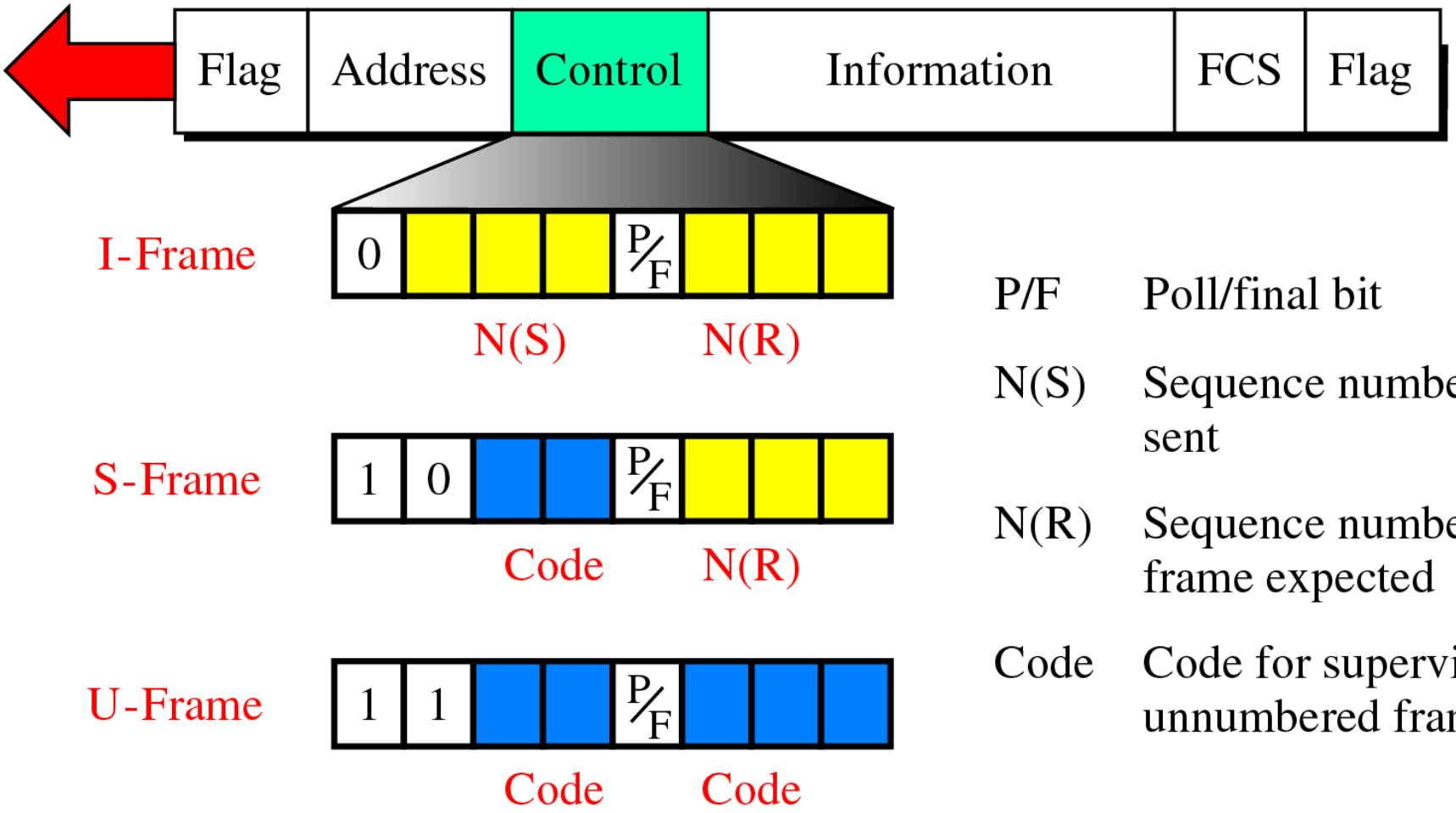


One-byte address

Multibyte address

Bit-Oriented protocol(cont'd)

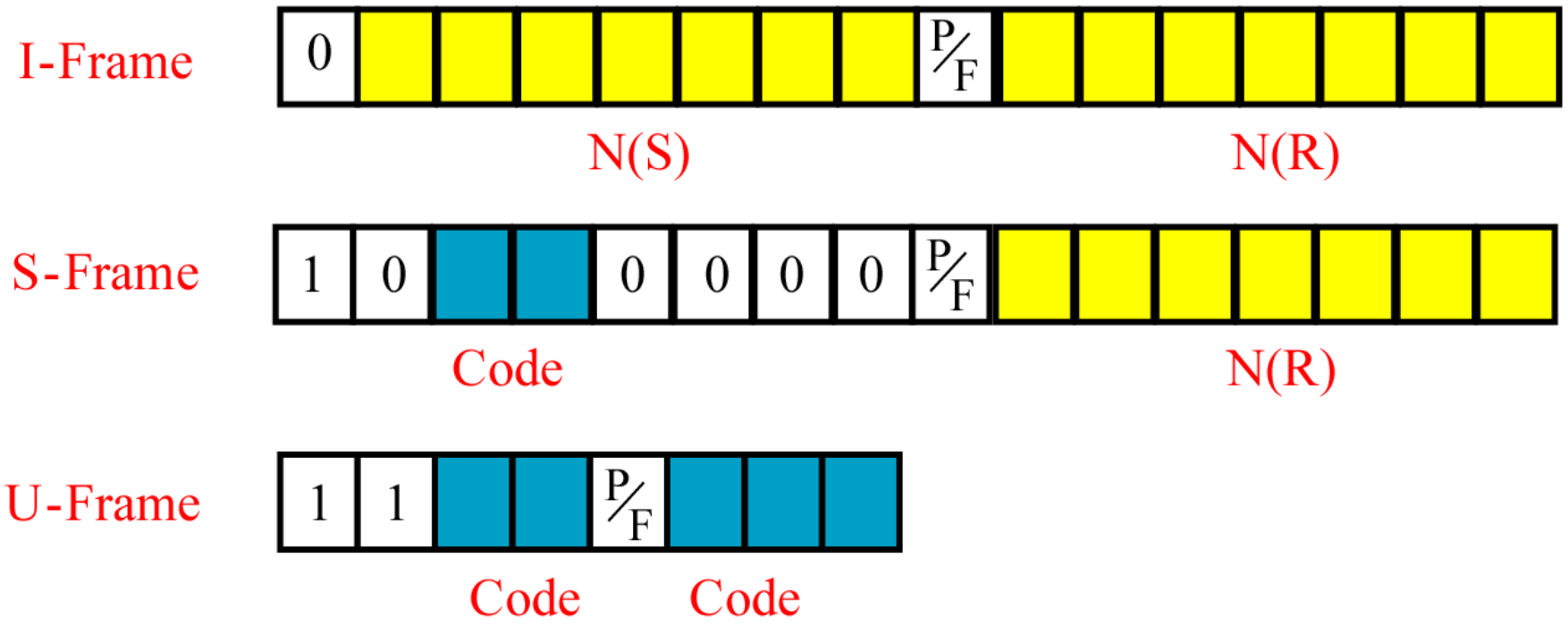
Control field



P/F	Poll/final bit
N(S)	Sequence number of frame sent
N(R)	Sequence number of next frame expected
Code	Code for supervisory or unnumbered frame

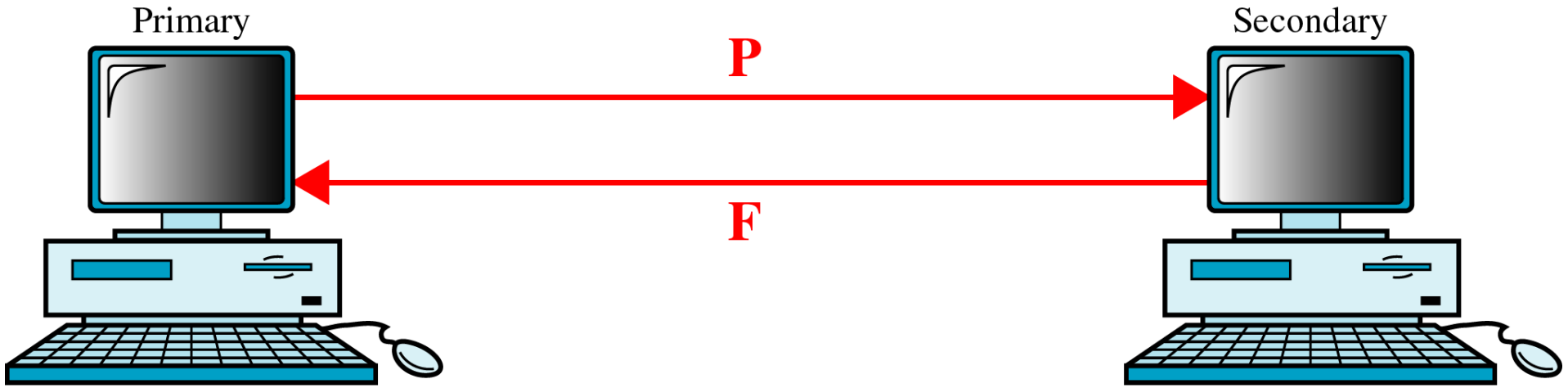
Bit-Oriented protocol(cont'd)

- Control field (extended mode)



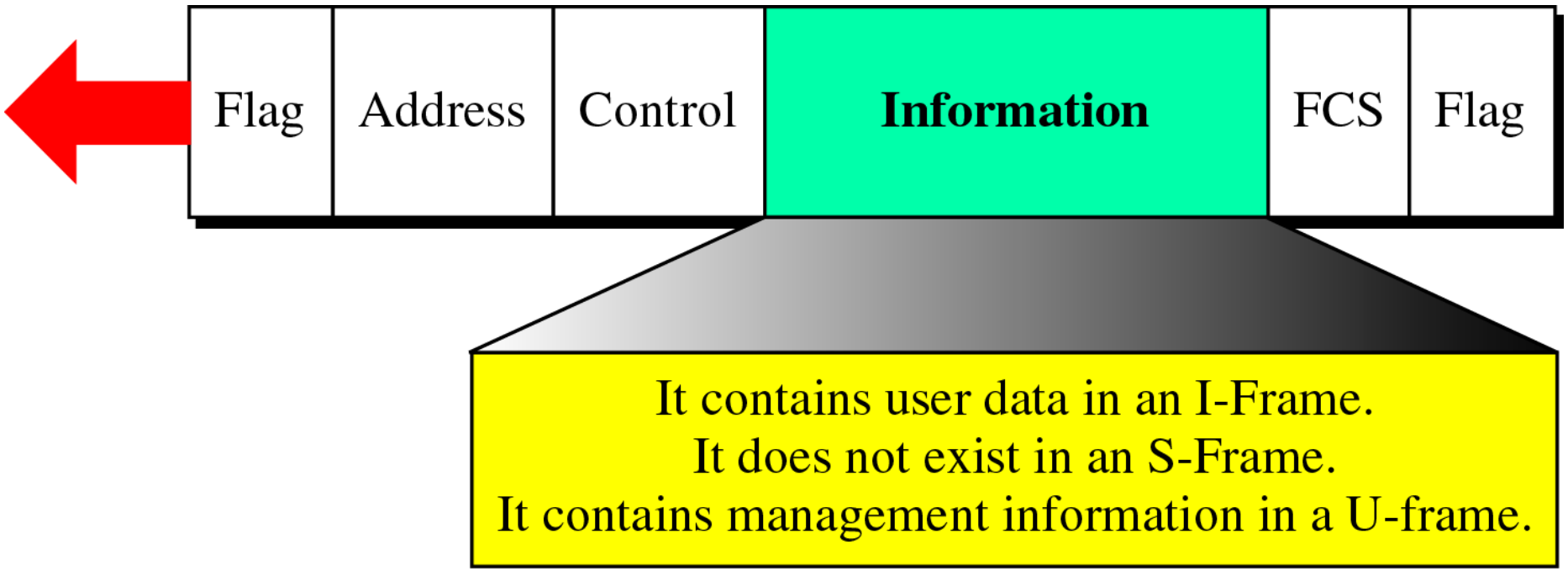
Bit-Oriented protocol(cont'd)

- Poll/Final field in HDLC



Bit-Oriented protocol(cont'd)

- Information field



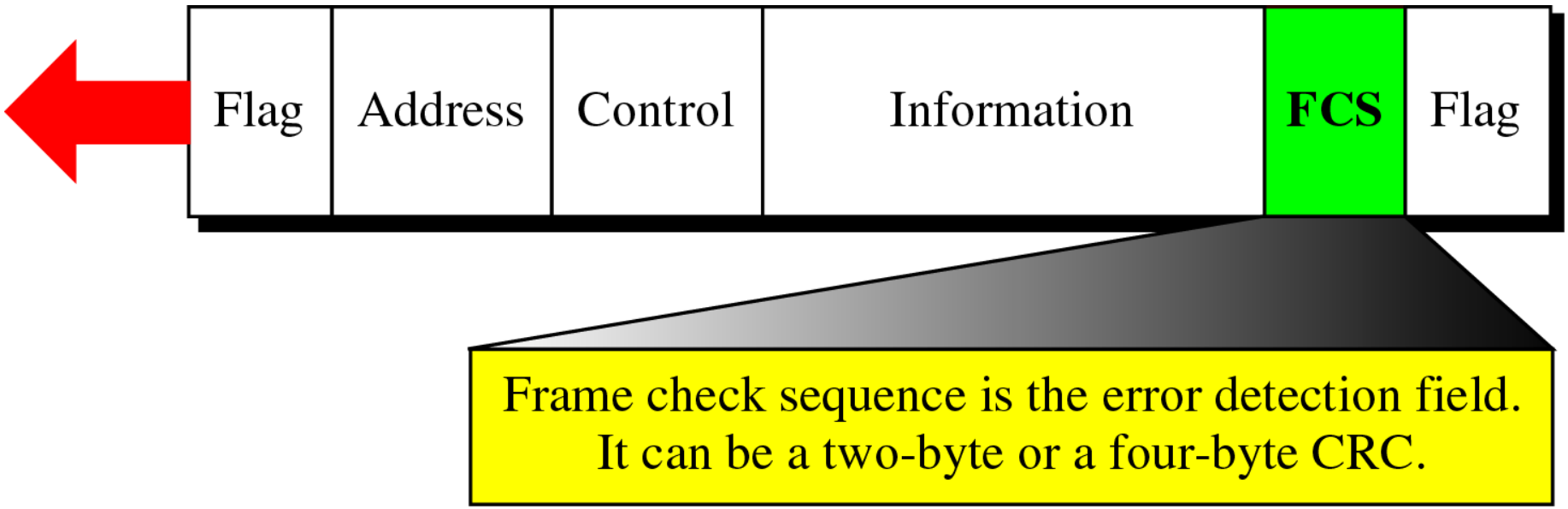
Bit-Oriented protocol(cont'd)

- Piggybacking

  ~ means combining data to be sent and acknowledgment of the frame received in one single frame

Bit-Oriented protocol(cont'd)

- FCS field

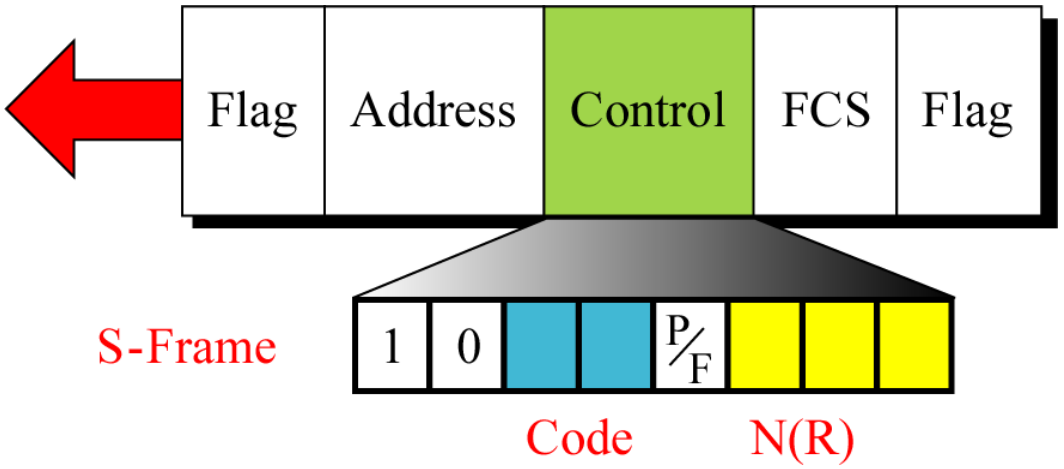


Bit-Oriented protocol(cont'd)

More about Frames

s-frame

~ is used for acknowledgment, flow control, and error control



Code	Command	
00	RR	Receive ready
01	REJ	Reject
10	RNR	Receive not ready
11	SREJ	Selective-reject

Bit-Oriented protocol(cont'd)

- RR(Receive Ready)
  - ACK
    - Used by a receiving station to return a positive acknowledgment
    - N (R) field having 3 bits (up to 8 frames)
  - Poll
    - When transmitted by primary with P/F bit set, RR asks if it has anything to send.
  - Negative response to poll
    - RR tells primary that secondary has nothing to send. If the secondary does have data to transmit, it responds to poll with an I-frame, not an S-frame
  - Positive response to select
    - When a secondary is able to receive a transmission from the primary

Bit-Oriented protocol(cont'd)

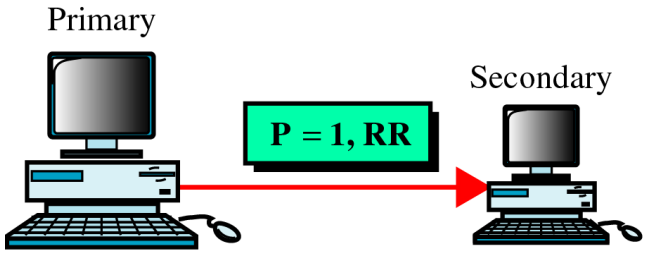
- RNR(Receive Net Ready)
  - ACK
    - RNR returned by a receiver to a sending station acknowledges receipt of all frames up to, but no including , the fame indicated in the N(R) field
  - Select
    - When a primary wishes to transmit data to a specific secondary, it alerts the secondary by sending an RNR frame with the P/F (used as P) set.
  - Negative response to select
    - When a selected secondary is unable to receive data, it returns an RNR.

Bit-Oriented protocol(cont'd)

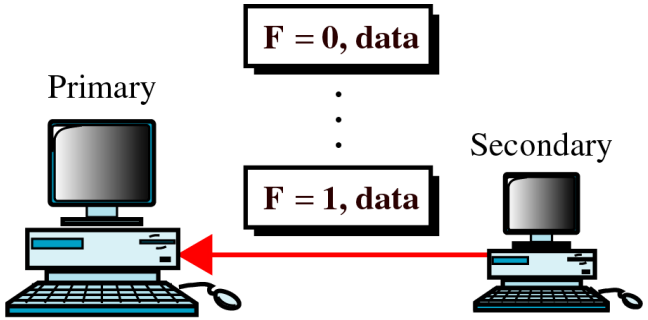
- REJ(Reject)
  - the negative acknowledgment returned by a receiver in a go-back-n ARQ error correction system
  - In an REJ frame, the N(R) field contains the number of the damaged frame
- SREJ(Selective-reject)
  - the negative acknowledgment returned by a receiver in a selective-reject ARQ error correction system

Bit-Oriented protocol(cont'd)

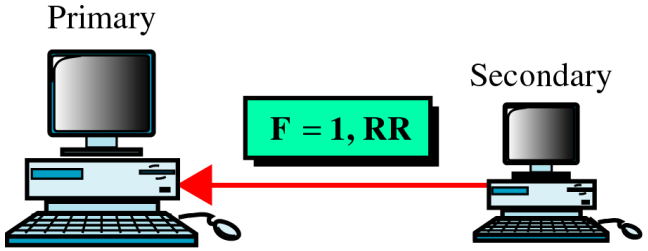
- Use of P/F bit in polling and selection



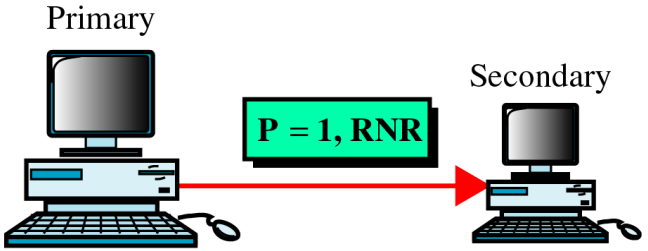
a. Poll



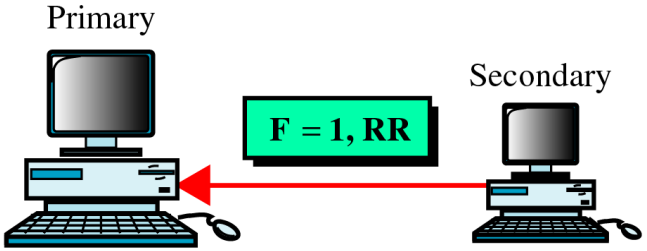
b. Positive response to poll



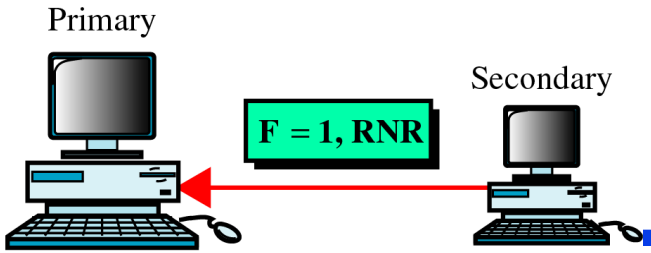
c. Negative response to poll



d. Select



e. Positive response to select

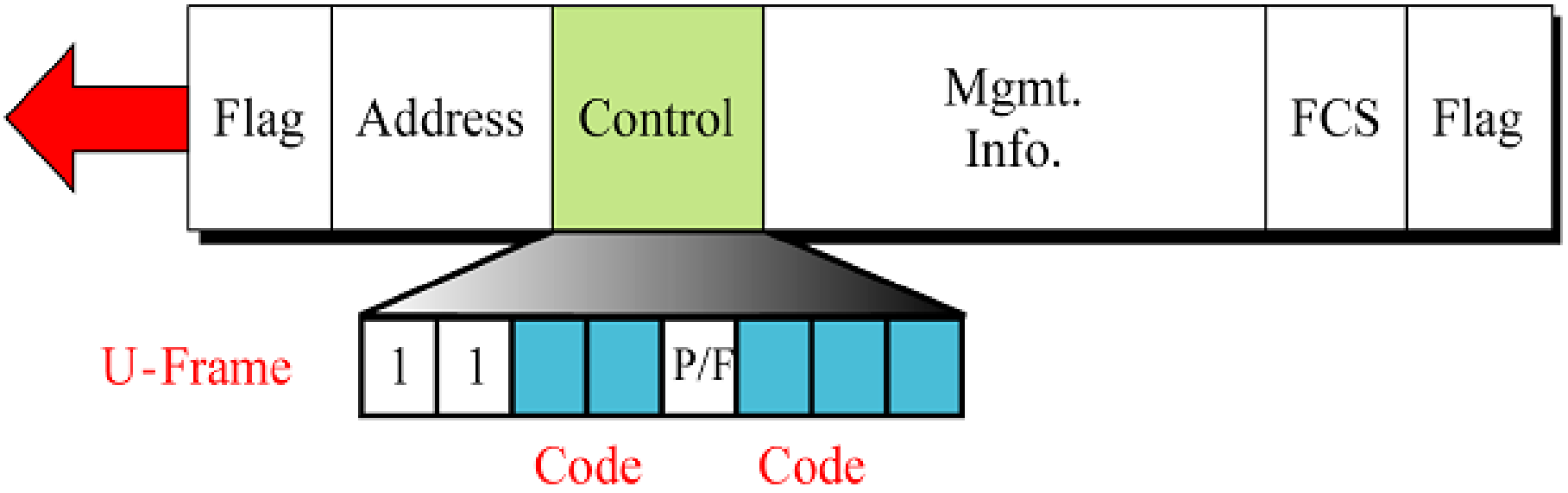


f. Negative response to select

Bit-Oriented protocol(cont'd)

- U-Frame

  ~ is used to exchange session management and control information between connected devices



Bit-Oriented protocol(cont'd)

Code	Command	Response
00 001	SNRM	
11 011	SNRME	
11 000	SARM	DM
11 010	SARME	
11 100	SABM	
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

Bit-Oriented protocol(cont'd)

U-Frame control command and response

Command/ response	Meaning
SNRM	Set normal response mode
SNRME	Set normal response mode(extended)
SARM	Set asynchronous response mode
SARME	Set asynchronous response mode(extended)
SABM	Set asynchronous balanced mode
SABME	Set asynchronous balanced mode(extended)
UP	Unnumbered poll
UI	Unnumbered information
UA	Unnumbered acknowledgement
RD	Request disconnect
DISC	Disconnect
DM	Disconnect mode
RIM	Request information mode
SIM	Set initialization mode
RSET	Reset
XID	Exchange ID
FRMR	Frame reject

Bit-Oriented protocol(cont'd)

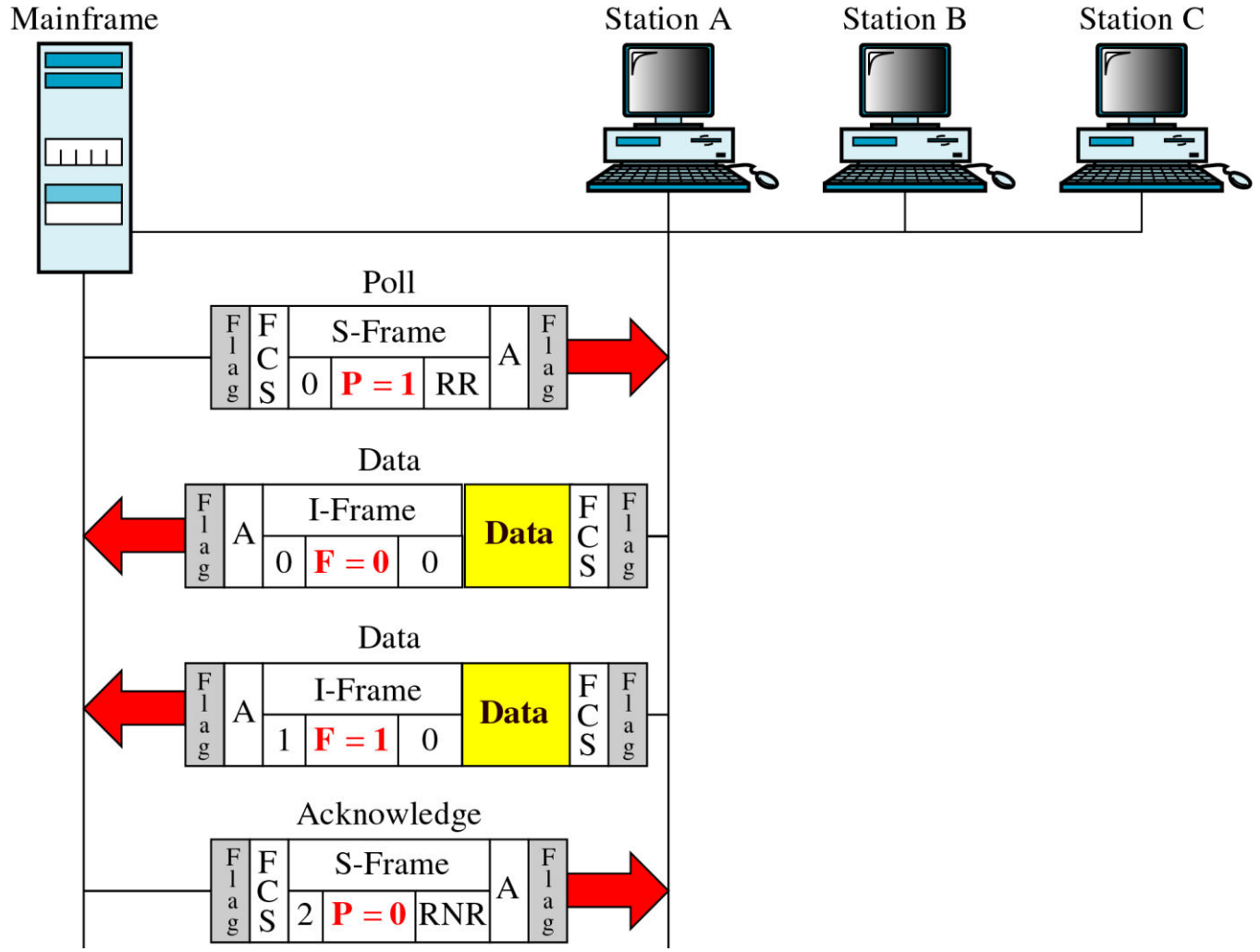
- U-Frame

  ~ can be divided into five basic functional category

  - Mode setting commands
    - are sent by the primary station, or by a combined station wishing to control an exchange, to establish the mode of the session(table 11.2)
    - SNRM, SNRME, SARM, SARME, SABM, SABME
  - Unnumbered-Exchange
    - are used to send or solicit specific pieces of data link information between device (table 11.2)
    - UP, UI, UA
  - Disconnection : RD, DISC, DM
  - Initialization Mode : RIM, SIM
  - Miscellaneous : RESET, XID, FRMR

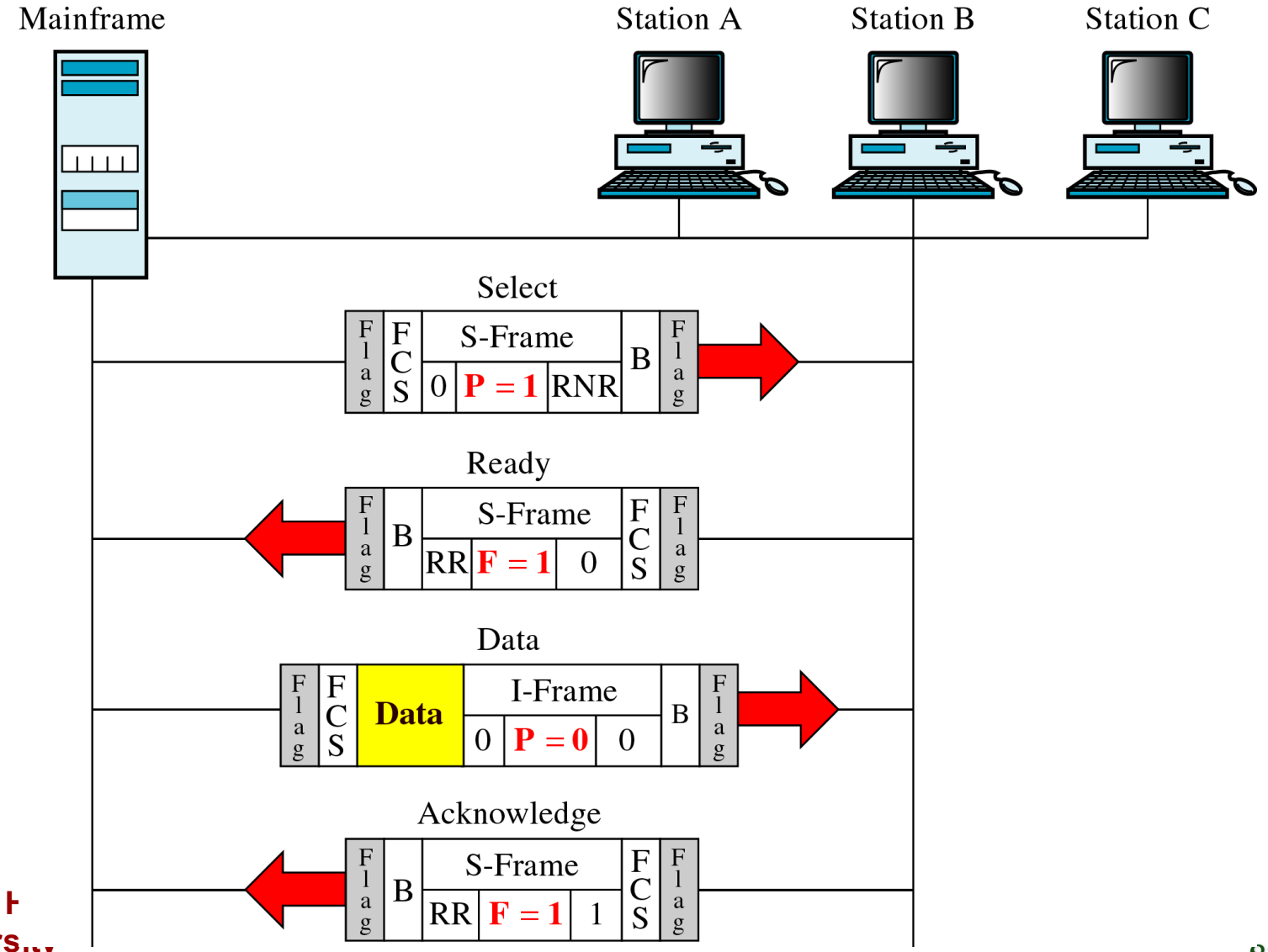
Bit-Oriented protocol(cont'd)

- **Example 1 : Poll/Response**



Bit-Oriented protocol(cont'd)

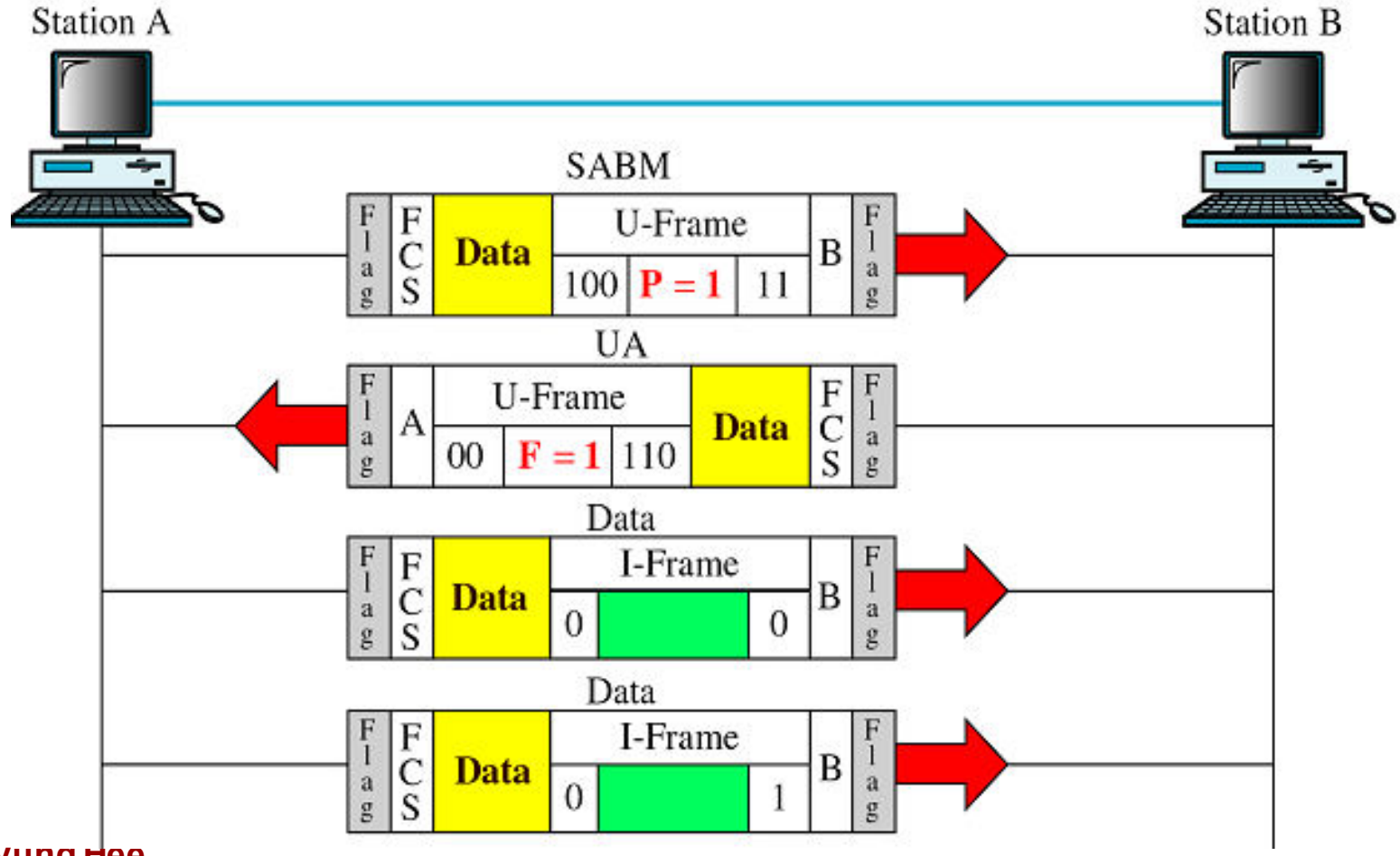
Example 2 : Select/Response



Bit-Oriented protocol(cont'd)

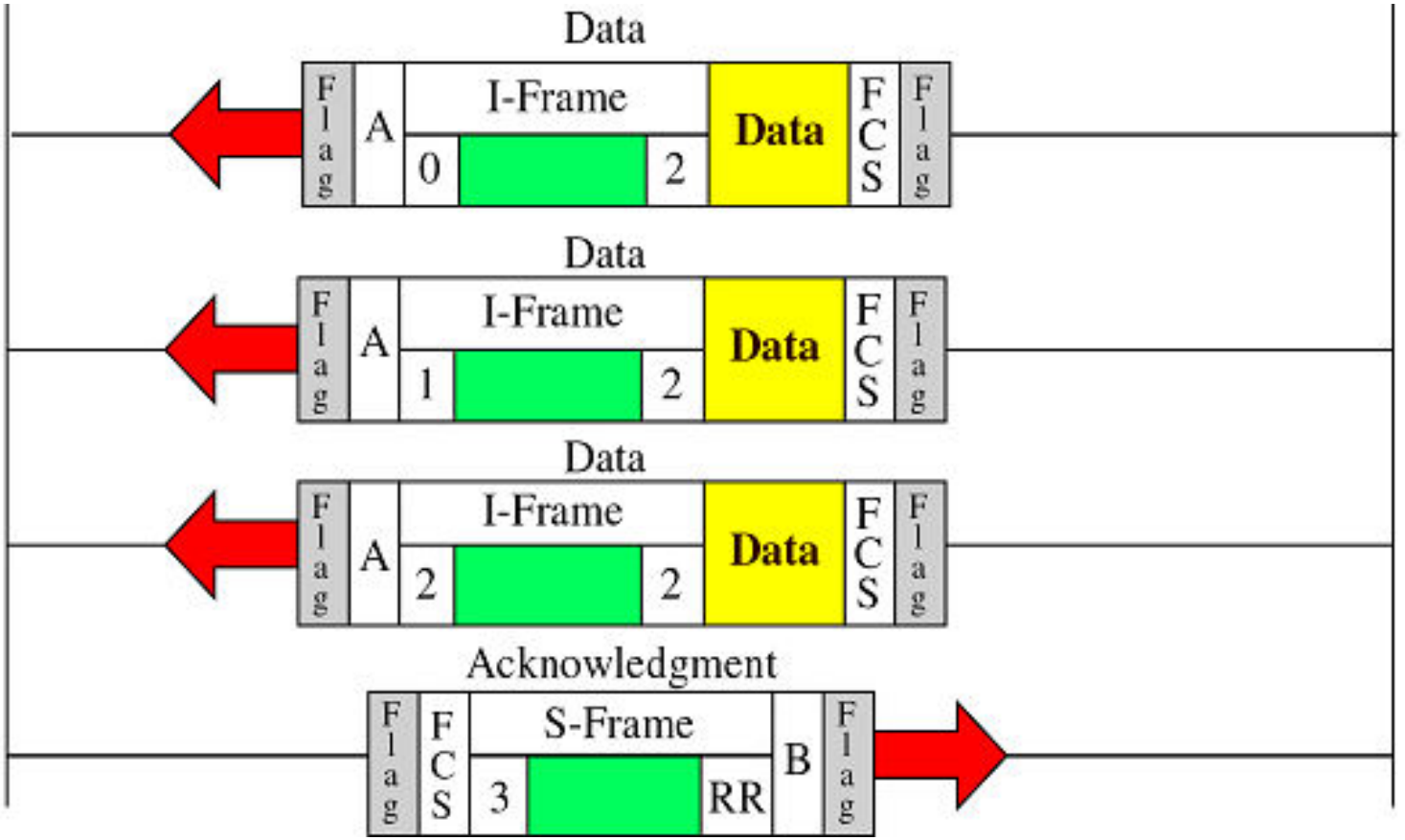
Example 3 : Peer Devices(1)

- Showing asynchronous balanced mode (ABM) using piggybacked acknowledgments



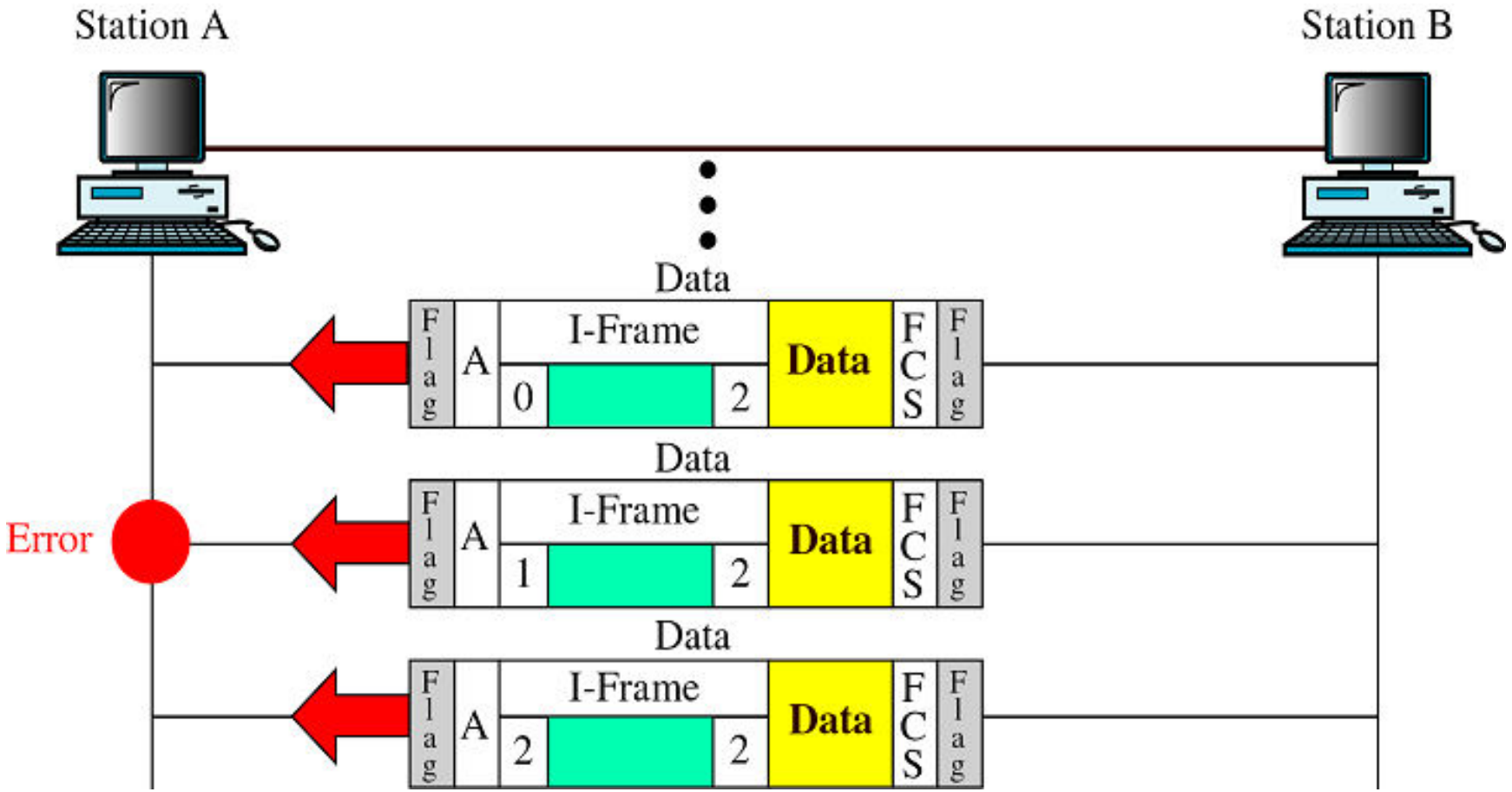
Bit-Oriented protocol(cont'd)

Example 3 : Peer Devices(2)



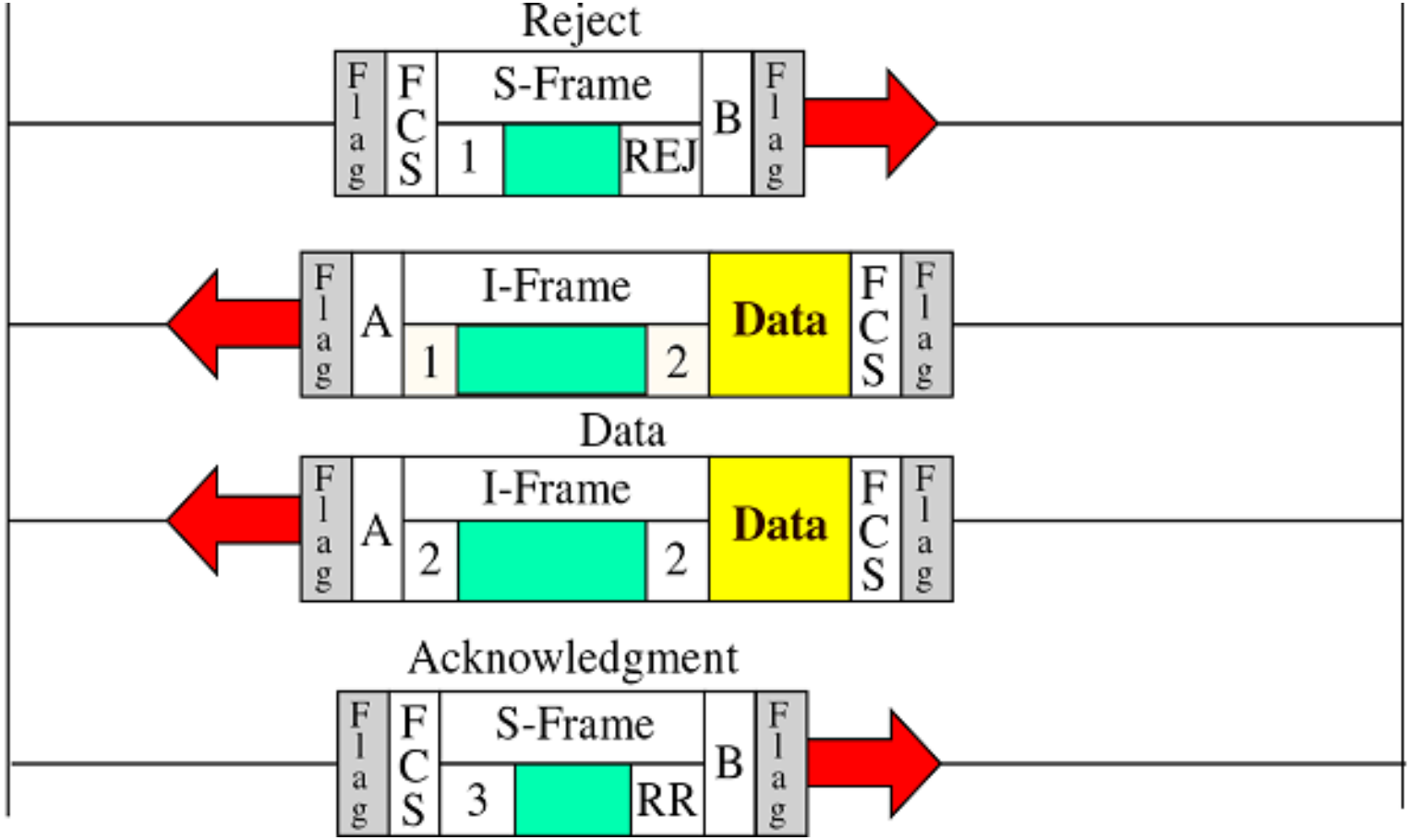
Bit-Oriented protocol(cont'd)

Example 4 : Peer Devices(1)



Bit-Oriented protocol(cont'd)

- **Example 4 : Peer Devices(2)**



Bit-Oriented protocol(cont'd)

- LAP(Link Access Procedure)
  - LAPB(Link Access Procedure Balanced)
    - ~ provides those basic control function required for communication between a DTE and a DCE
    - ~ is used only in balanced configuration of two devices
    - ~ is used in ISDN on B channels
  - LAPD(Link Access Procedure for D channel)
    - ~ used in ISDN
    - ~ use ABM(Asynchronous Balanced Mode)
  - LAPM(Link Access Procedure for Modem)
    - ~ is designed to do asynchronous-synchronous conversation, error detection, and retransmission
    - ~ has become developed to apply HDLC feature to modem