Data Link Layer

Presented by Hung Ba Ngo – email: nbhung@cit.ctu.edu.vn

Cantho – February, 2016

Objectives

- Services provided by Data link layer
- Framing
- Error control
- Flow control

Requisites

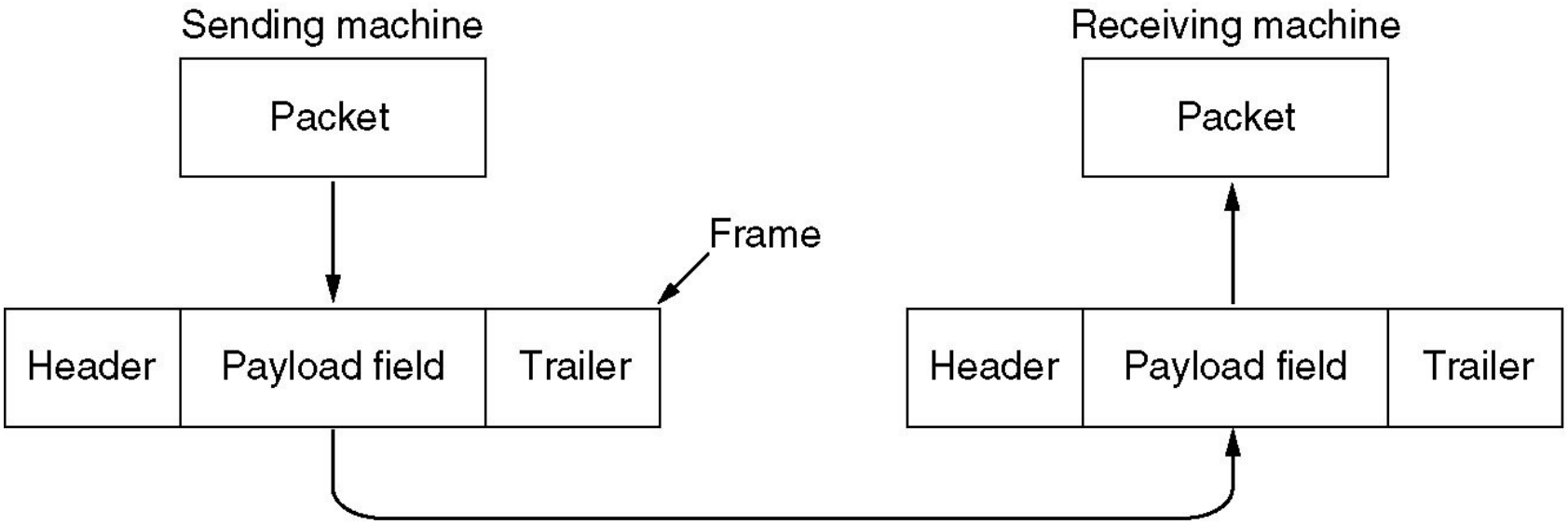
- Learners have to be able to
  - present functions of Data link layer in error control
  - introduce principal methods of framing such as Byte count, Flag bytes with byte stuffing and Flag bits with bit stuffing.
  - differentiate between error detection, error control and flow control in Data link layer
  - implement error detection methods such as Parity check, Checksum, Cyclic Redundancy Checks
  - implement error control protocols such as Stop and Wait, Sliding windows, Go-Back-N, Selective Repeat
  - present basic concept of HDLC protocol

Functions of Data link layer

- Providing well-defined service interfaces to the network layer
- Dealing with transmission errors
- Regulating the flow of data so that slow receivers are not swamped by fast senders

Functions of Data link layer (cont.)

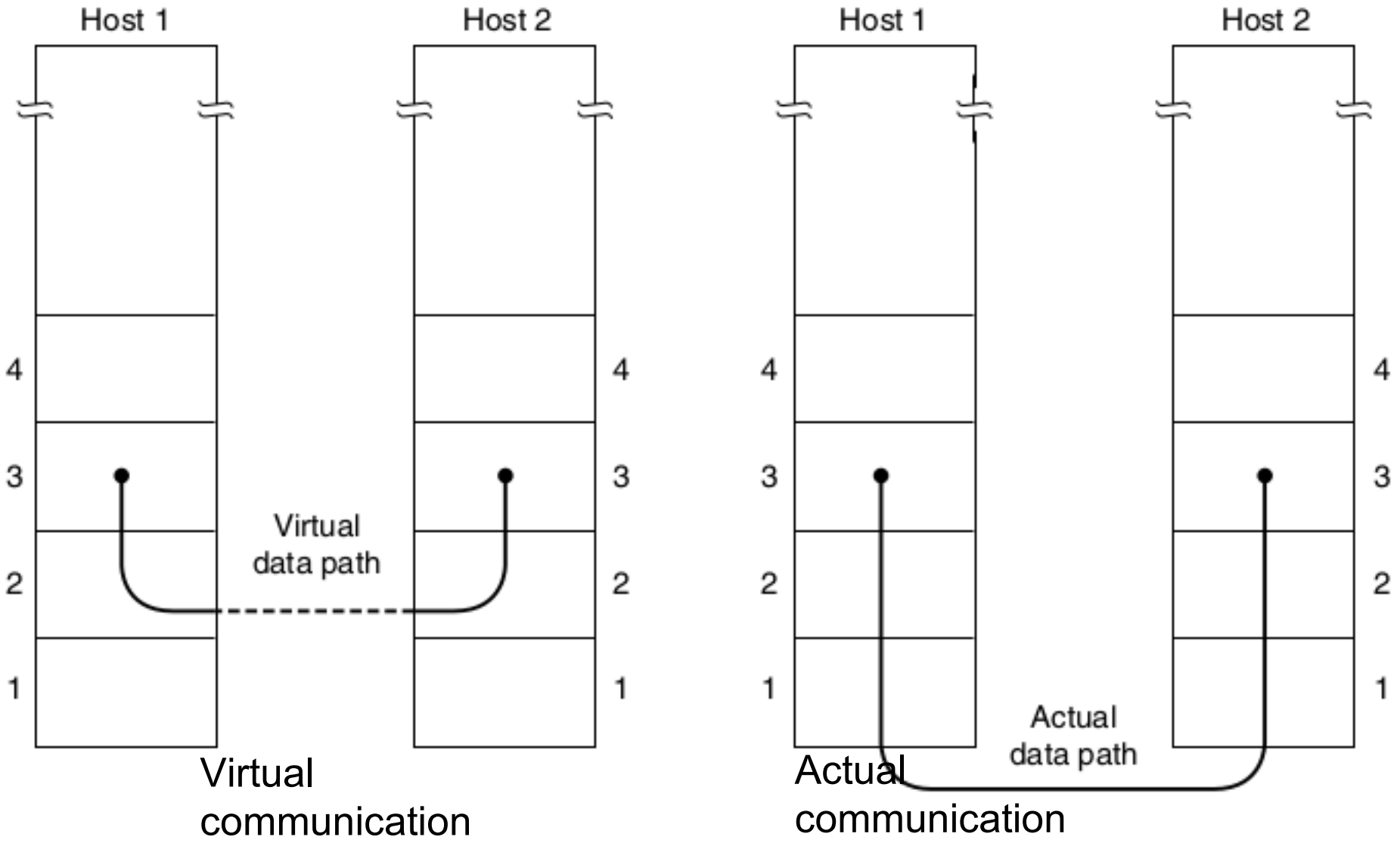
- Providing services the Network layer
- Takes the packets it gets from the network layer and encapsulates them into frames for transmission



Services provided to Network layer

- Unacknowledged connectionless service used in LANs
- Acknowledged connectionless service used in wireless LANs
- Acknowledged connection-oriented service used in WANs.

Services provided to Network layer



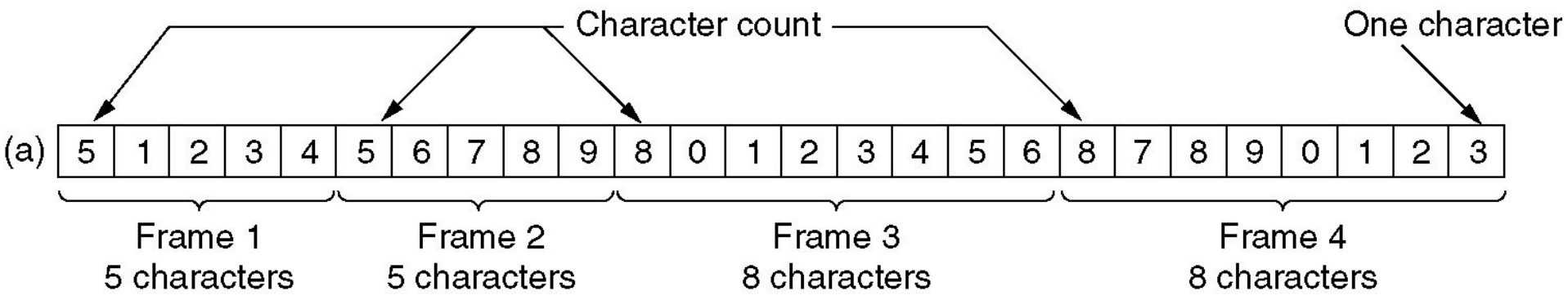
Framing

- Data link layer must use the service provided to it by Physical layer to provide services to the Network layer
- Bit stream received by the data link layer is not guaranteed to be error free:
  - bits may have different values and the number of bits received may be less than, equal to, or more than the number of bits transmitted
- Data link layer to break up the bit stream into discrete frames, compute a short token called a checksum for each frame, and include the checksum in the frame when it is transmitted
- When a frame arrives at the destination, the checksum is recomputed
- If the newly computed checksum is different from the one contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it

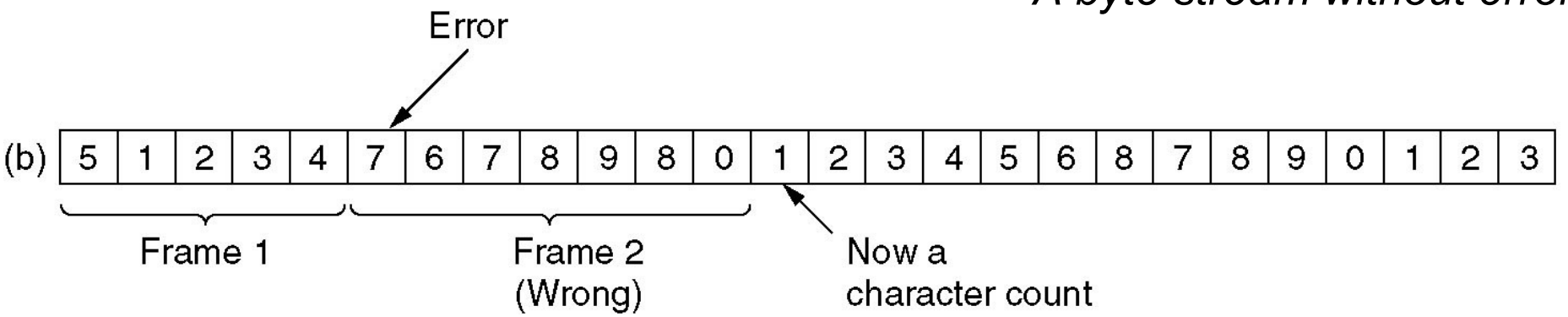
Framing methods

- Byte count
- Flag bytes with byte stuffing
- Flag bits with bit stuffing.

Byte Count



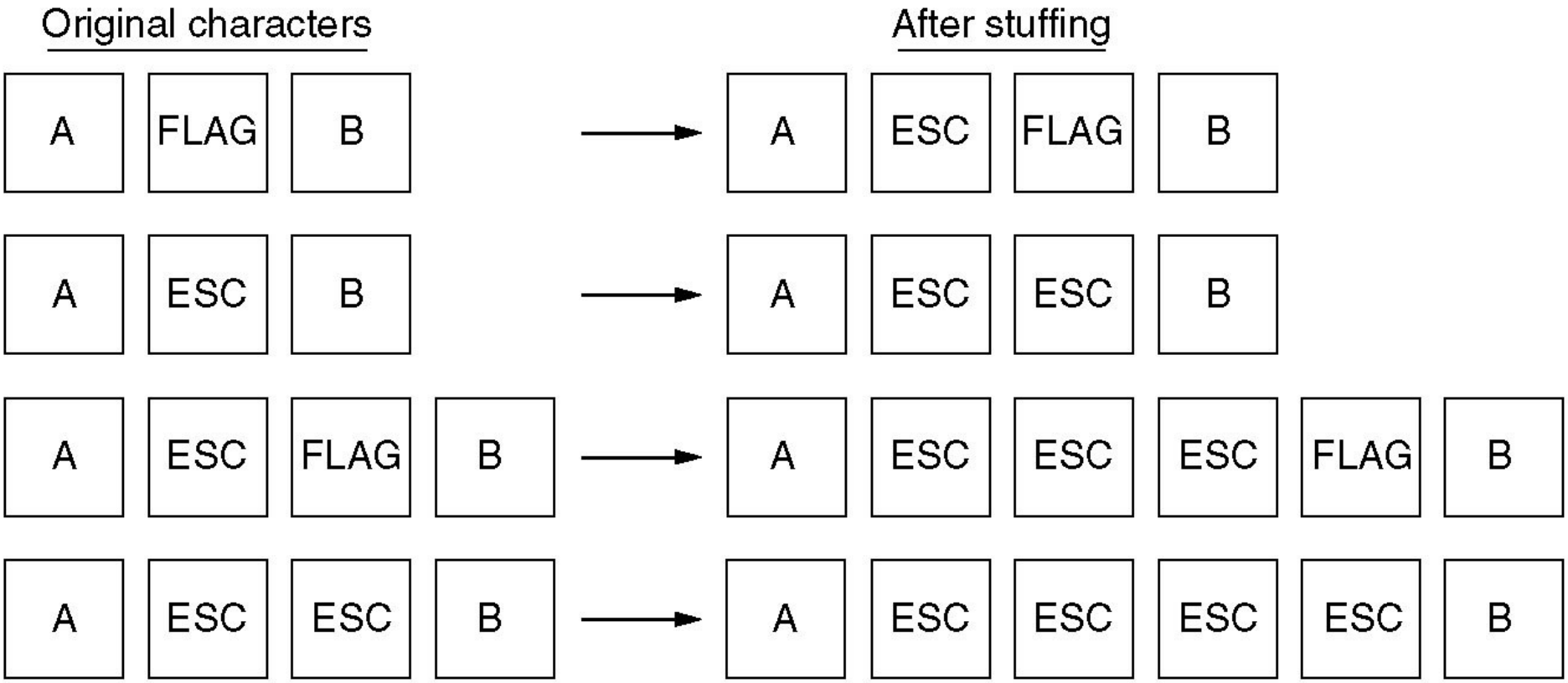
A byte stream without errors



A byte stream with errors

Flag byte with byte stuffing

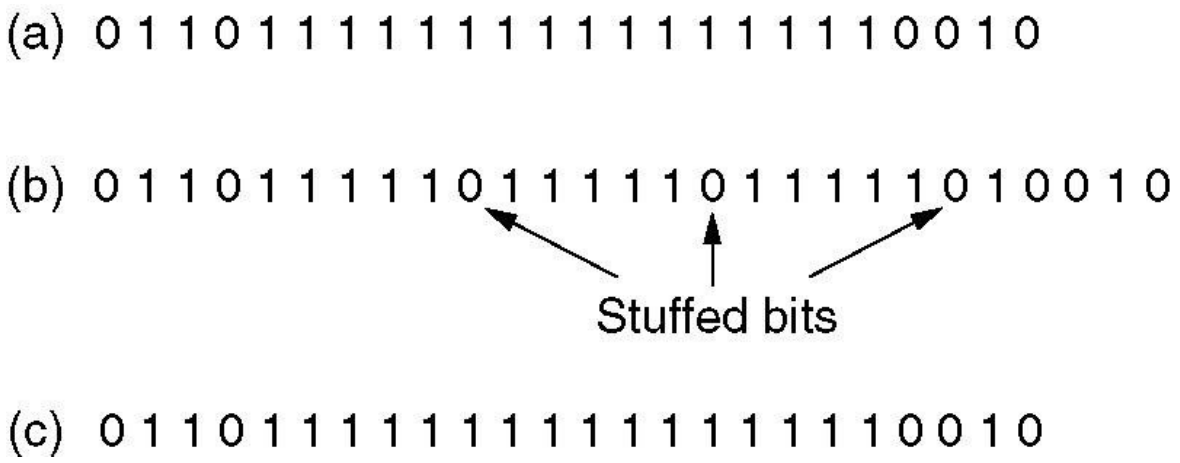
(a) A frame delimited by flag bytes



(b) Four examples of byte sequences before and after byte stuffing

Starting and ending flags with bit stuffing

- Using flag pattern 01111110 to delimit the start and the end of a frame



(a) Original data

(b) Data as they appear on the line

(c) Data as they are stored in the receiver's memory after destuffing

Bài tập 2.1

Hãy cài đặt bằng ngôn ngữ C/C++/Java các phương pháp định khung sau: Sử dụng các bytes làm cờ hiệu và Sử dụng cờ bắt đầu và kết thúc khung với các ràng buộc sau:

- Nhận đầu vào là 1 file, kích thước của khung, **bytes là cờ hiệu, bytes là ký tự ESC** (hoặc bytes chứa mẫu bits)

- int framing(char* filename, int len, byte flag, byte esc)

- int framing(char* filename, int len, byte flag)

- In ra màn hình các khung được tạo ra để chuyển tải dữ liệu cho file này

Error Control

- How to make sure all frames are eventually delivered to the network layer at the destination and in the proper order ?
  - Receiver sending back special control frames bearing positive or negative acknowledgements about the incoming frames:
    - Using acknowledgement frame
  - Avoiding the sender to be hung forever when acknowledge frame lost
    - Using a timer and time-out for each sent frame
  - Avoiding multiple times of receiving a frame
    - Assign each sent frame a sequence number

Flow Control

- Resolving problems occurring when send rate and receive rate are different
- Two approaches
  - Feedback-based flow control: the receiver sends back information to the sender giving it permission to send more data, or at least telling the sender what the receiver is doing
  - Rate-based flow control: the protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver

Error Detection

Error detection and correction

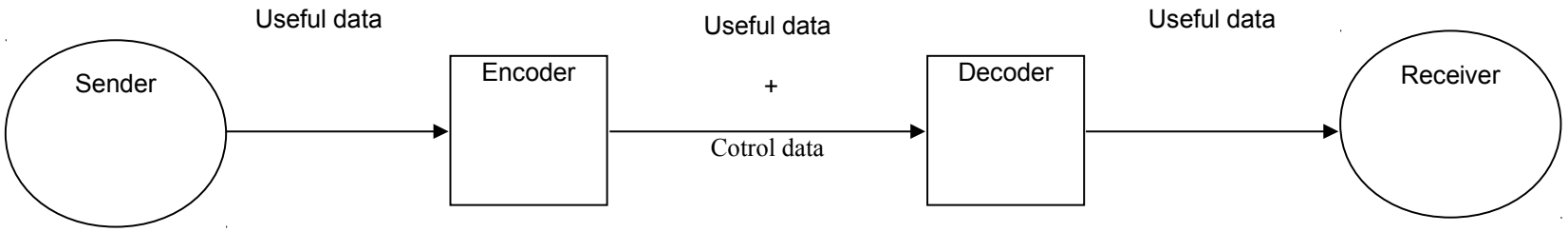
- What is error-detecting codes
- Popular error-detecting codes
  - Parity checks
  - Check sum
  - Cyclic redundancy check

Transmission errors

- Bit 1 becomes bit 0 and reversely
- Error rate
  - $\forall \tau$ = Error bits / Transmitted bits
  - $\forall \tau$: 10^{-5} to 10^{-8}
    - 88% : error of one bit
    - 10% : error of two adjacent bits

Error-detecting codes

- Sender: add control data into useful data for sending
- Receiver: decode control data to determine if the useful data are correct or not



Error-detecting codes

- Error-correcting codes
  - Allow receivers to identify error data and correct error data
- Error-detecting codes
  - Allow receivers to determine if received data are correct or not
  - If error is detected in received data, request for resending data
- Computer networks use error-detecting codes rather than error-correcting codes

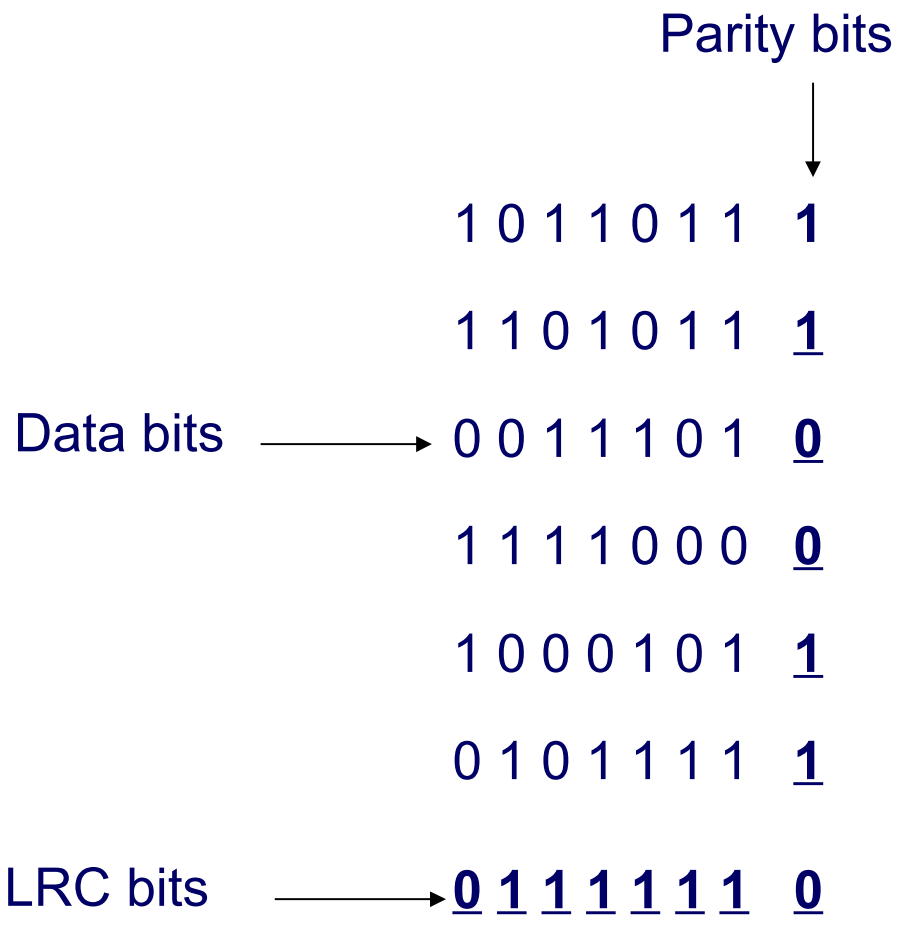
Parity Check

- xxxxxxx: Useful data need to be transmitted
- One parity bit is appended to useful data
- Real transmitted stream of bits: xxxxxxxp
- Calculation of p
  - Event parity: xxxxxxxp consists of an event number of bits 1
  - Odd parity: xxxxxxxp consists of an odd number of bits 1
- Detecting error in a stream of bits xxxxxxxp:
  - In event parity check:
    - If there is an even number of bits 1→ data xxxxxxx are error-free
    - Else data xxxxxxx are error
  - In odd parity check:
    - If there is an odd number of bits 1→ data xxxxxxx are error-free
    - Else data xxxxxxx are error

Parity Check

- Example: G = 1110001 are useful data
- Using even parity check:
  - p=0
  - Transmitted data: 1110001**0**
- Different cases of received data:
  - 11100010: 4 bits 1=> Error-free data
  - 11000010: 3 bits 1=> Error data
  - 11000110: 4 bits 1=> Error ???

Longitudinal Redundancy Check or Checksum



Bài tập 2.2

Bổ sung vào bài tập 2.1:

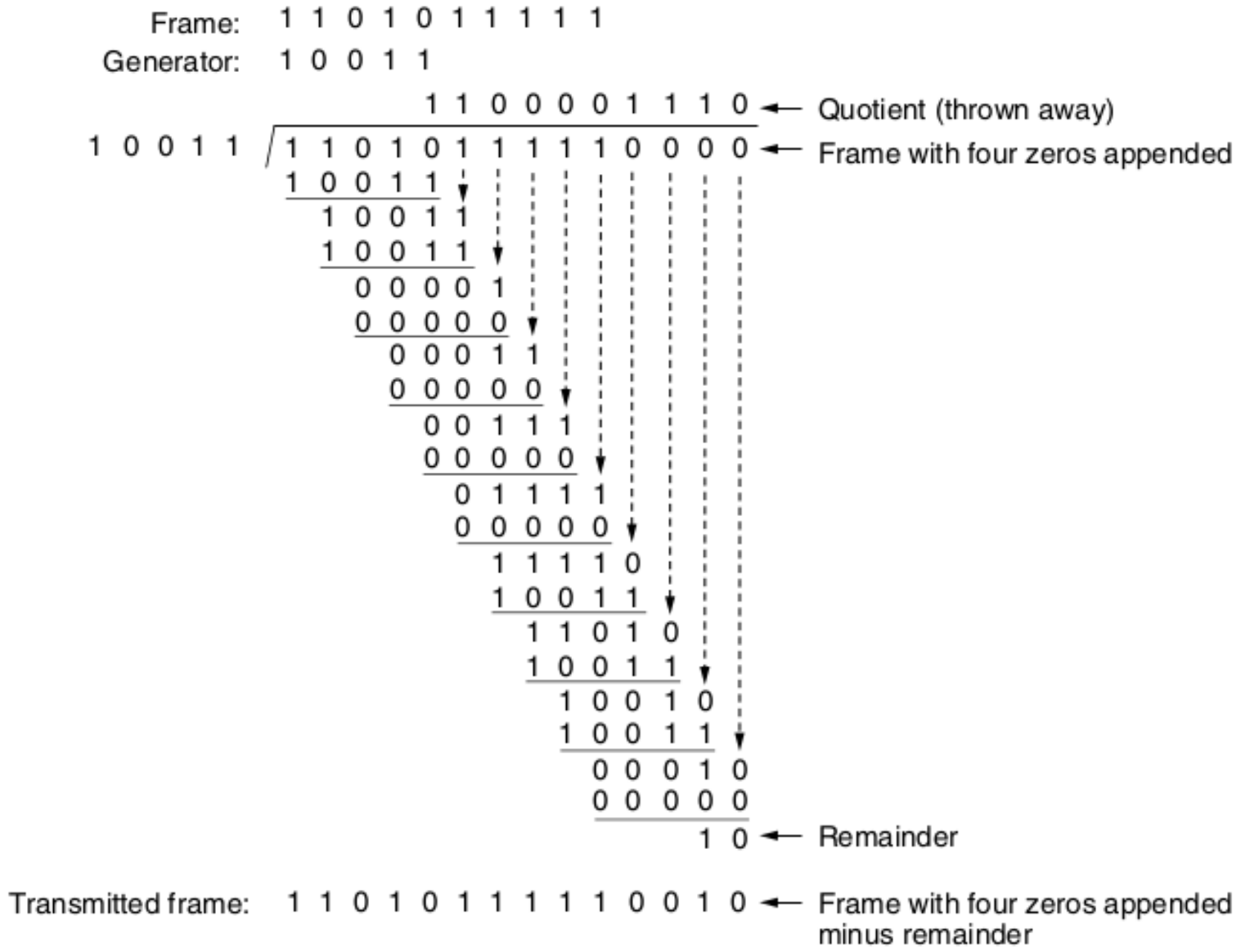
- Thêm vào mỗi byte dữ liệu 1 bit kiểm tra chẵn
- Thêm vào tổng kiểm tra LRC vào cuối mỗi khung
- Xây dựng hàm int error_detect(frame) nhận vào một khung, kiểm tra trả về có lỗi (-1) hay không có lỗi (0)
- Giới hạn: Mỗi byte dữ liệu là 7 bits
- Đề xuất giải pháp cho trường hợp dữ liệu 8 bits

Cyclic Redundancy Check

- Different methods of implementation:
  - Modulo 2,
  - Polynomial
  - Shift register
  - Exclusive-or gate

CRC-Modulo 2

- Given:
  - M: Message of k bits need to send
  - F : A frame check sequence of r bits is control data that is appended to M to help receiver to detect error on M
  - T =MF is the transmitted frame of (k + r) bits created by concatenating M and F, with r < k
- With M (k bits) , P (r+1 bits), F (r bits), T (k+r bits), the algorithm of calculating F and creating transmitted frame will be as following
  - Appending r bits 0 to the end of M, or multiply M with 2^r
  - Using binary division to divide $M*2^r$ with P.
  - Adding the remainder of above division to $M*2^r$ to create transmitted frame T
  - Note: P has to be longer than F one bit and the values of its most significant bit and the least significant must be 1
- Receiver applies binary division to divide T with P:
  - No remainder: Error-free, M extracted from (T – k) high order bits
  - Remainder existed: Transmission of T is error


Frame: 1 1 0 1 0 1 1 1 1 1
Generator: 1 0 0 1 1
1 1 0 0 0 0 1 1 1 0
Quotient (thrown away)
1 0 0 1 1
1 1 0 1 0 1 1 1 1 1 0 0 0 0
Frame with four zeros appended
1 0 0 1 1
1 0 0 1 1
1 0 0 1 1
0 0 0 0 1
0 0 0 0 0
0 0 0 1 1
0 0 0 0 0
0 0 1 1 1
0 0 0 0 0
0 1 1 1 1
0 0 0 0 0
1 1 1 1 0
1 0 0 1 1
1 1 0 1 0
1 0 0 1 1
1 0 0 1 0
1 0 0 1 1
0 0 0 1 0
0 0 0 0 0
1 0
Remainder
Transmitted frame: 1 1 0 1 0 1 1 1 1 1 0 0 1 0
Frame with four zeros appended minus remainder

CRC-Modulo 2

- Given:
  - M = 1010001101 (k=10 bits)
  - P = 110101 (r+1=6 bits)

Steps to calculate FCS (r=5bits):

- Calculate $M*2^5$ = 1010001101**00000**.
- Apply binary division to divide $M*2^5$ by P → Get remainder F = **01110**

Create transmitted frame

- T = $M*2^r$ + F = 1010001101**01110**

```
                                1 1 0 1 0 1 0 1 1 0     ( Q : Kết quả phép chia)
(P) 1 1 0 1 0 1 | 1 0 1 0 0 0 1 1 0 1 0 0 0 0 0        ( M*2^r)
                  1 1 0 1 0 1
                  -----------
                    1 1 1 0 1 1
                    1 1 0 1 0 1
                    -----------
                      0 1 1 1 0 1
                      0 0 0 0 0 0
                      -----------
                        1 1 1 0 1 0
                        1 1 0 1 0 1
                        -----------
                          0 1 1 1 1 1
                          0 0 0 0 0 0
                          -----------
                            1 1 1 1 1 0
                            1 1 0 1 0 1
                            -----------
                              0 1 0 1 1 0
                              0 0 0 0 0 0
                              -----------
                                1 0 1 1 0 0
                                1 1 0 1 0 1
                                -----------
                                  1 1 0 0 1 0
                                  1 1 0 1 0 1
                                  -----------
                                    0 0 1 1 1 0
                                    0 0 0 0 0 0
                                  +-----------+
                                  | 0 1 1 1 0 = F |
                                  +-----------+
```

Bài tập 2.3.1

Bên gởi và nhận đã thống nhất chọn P=110101

Cho biết các khung T nhận được sau có lỗi hay không

- **T=101.0001.1010.1110**
- **T=101.0101.1010.1110**

Bài tập 2.3.2

Viết hàm char * fcs(char* T, char* P) chia chuỗi bits T cho chuỗi bits P và trả về chuỗi bits số dư F

Bài tập 2.3.1

Cho dữ liệu đầu vào là:

- 1Chuỗi b**ị chia** gồm các ký tự 0 và 1
- 1 Chuỗi chia gồm các ký tự 0 và 1

Hãy tính phần dư theo phép chia modulo 2 của Chuỗi bị chia cho Chuỗi chia

char * modulo2(char * cbc, char *cc)

CRC - Polynomial

- Supposing that M=110011 and P = 11001, then M and P are represented by two following polynomials :
  - $M(x) = x^5 + x^4 + x + 1$
  - $P(x) = x^4 + x^3 + 1$
- Algorithm for computing the CRC is represented as follows:

$$\frac{X^r M(X)}{P(X)} = Q(X) + \frac{R(X)}{P(X)}$$

$$T(X) = X^n M(X) + R(X)$$

CRC - Polynomial

Popular versions of P

$$\text{CRC-12} = X^{12} + X^{11} + X^3 + X^2 + X + 1$$

$$\text{CRC-16} = X^{16} + X^{15} + X^2 + 1$$

$$\text{CRC-CCITT} = X^{16} + X^{12} + X^5 + 1$$

$$\text{CRC-32} = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

CRC - Polynomial

Example

- Given $M=1010001101, P=110101$
- Then $r=5$

$$M(x)= x^9+x^7+x^3+x^2+1$$

$$x^5M(x)= x^{14}+x^{12}+x^8+x^7+x^5$$

$$P(x) = x^5+x^4+x^2+1$$

- Compute

$$\frac{x^rM(x)}{P(x)} = Q(x) + \frac{F(x)}{P(x)}$$

$$\Rightarrow \quad Q(x)= x^9+ x^8+ x^6+x^4+x^2 +x^1$$

$$F(x)= x^3+ x^2+ x^1 <> \mathbf{01110}$$

⇨ Final transmission frame, $T= 1010001101\mathbf{01110}$

Error Control

CRC - Polynomial

Example

- Given $M=1010001101, P=110101$
- Then $r=5$

$$M(x)= x^9+x^7+x^3+x^2+1$$

$$x^5M(x)= x^{14}+x^{12}+x^8+x^7+x^5$$

$$P(x) = x^5+x^4+x^2+1$$

- Compute

$$\frac{x^rM(x)}{P(x)} = Q(x) + \frac{F(x)}{P(x)}$$

$$\Rightarrow \quad Q(x)= x^9+ x^8+ x^6+x^4+x^2 +x^1$$

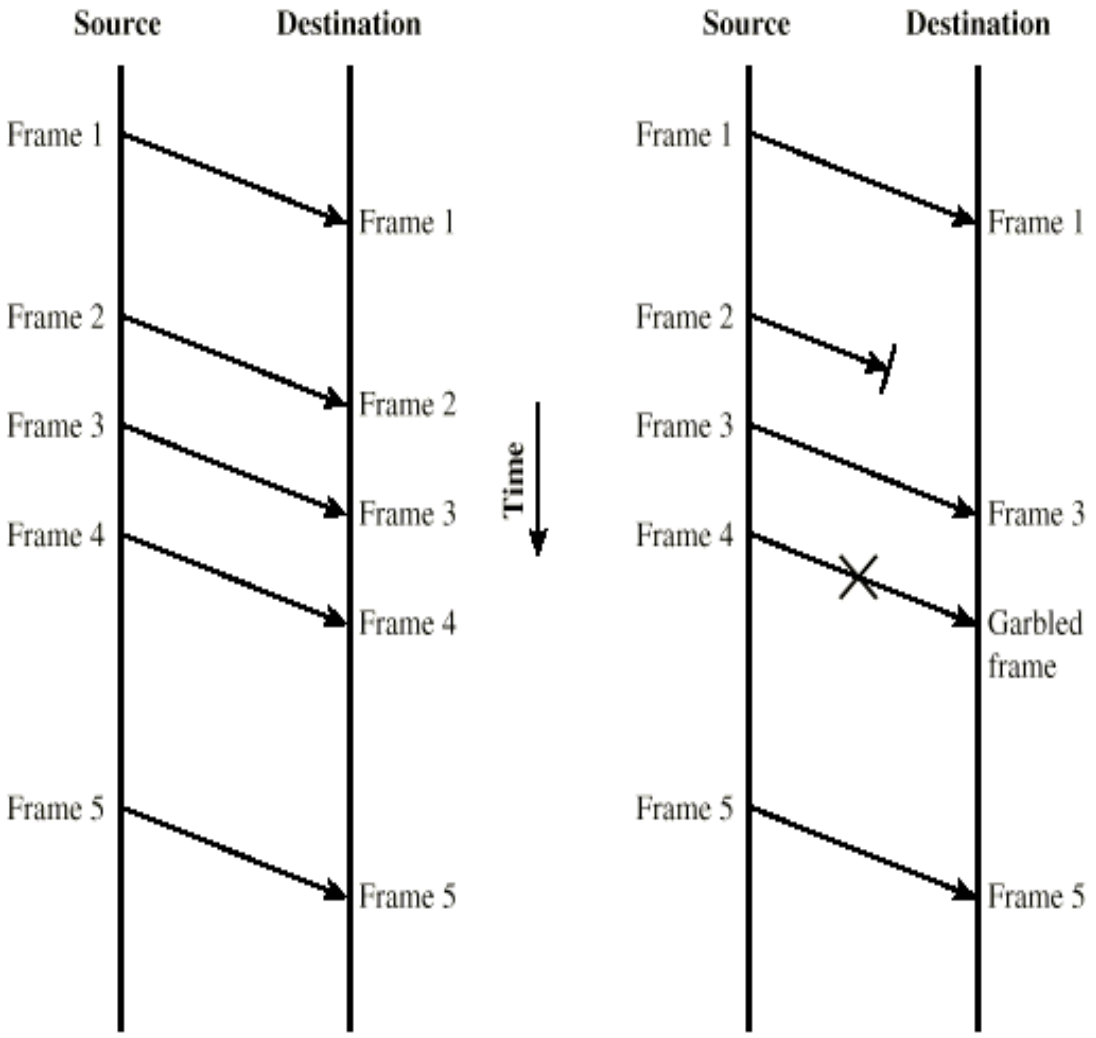
$$F(x)= x^3+ x^2+ x^1 <> \mathbf{01110}$$

⇨ Final transmission frame, $T= 1010001101\mathbf{01110}$

Error Control

- How to make sure all frames are eventually delivered to the network layer at the destination and in the proper order ?
  - Receiver sending back special control frames bearing positive or negative acknowledgements about the incoming frames:
    - Using acknowledgement frame
  - Avoiding the sender to be hung forever when acknowledge frame lost
    - Using a timer and time-out for each sent frame
  - Avoiding multiple times of receiving a frame
    - Assigning each sent frame a sequence number

Error control

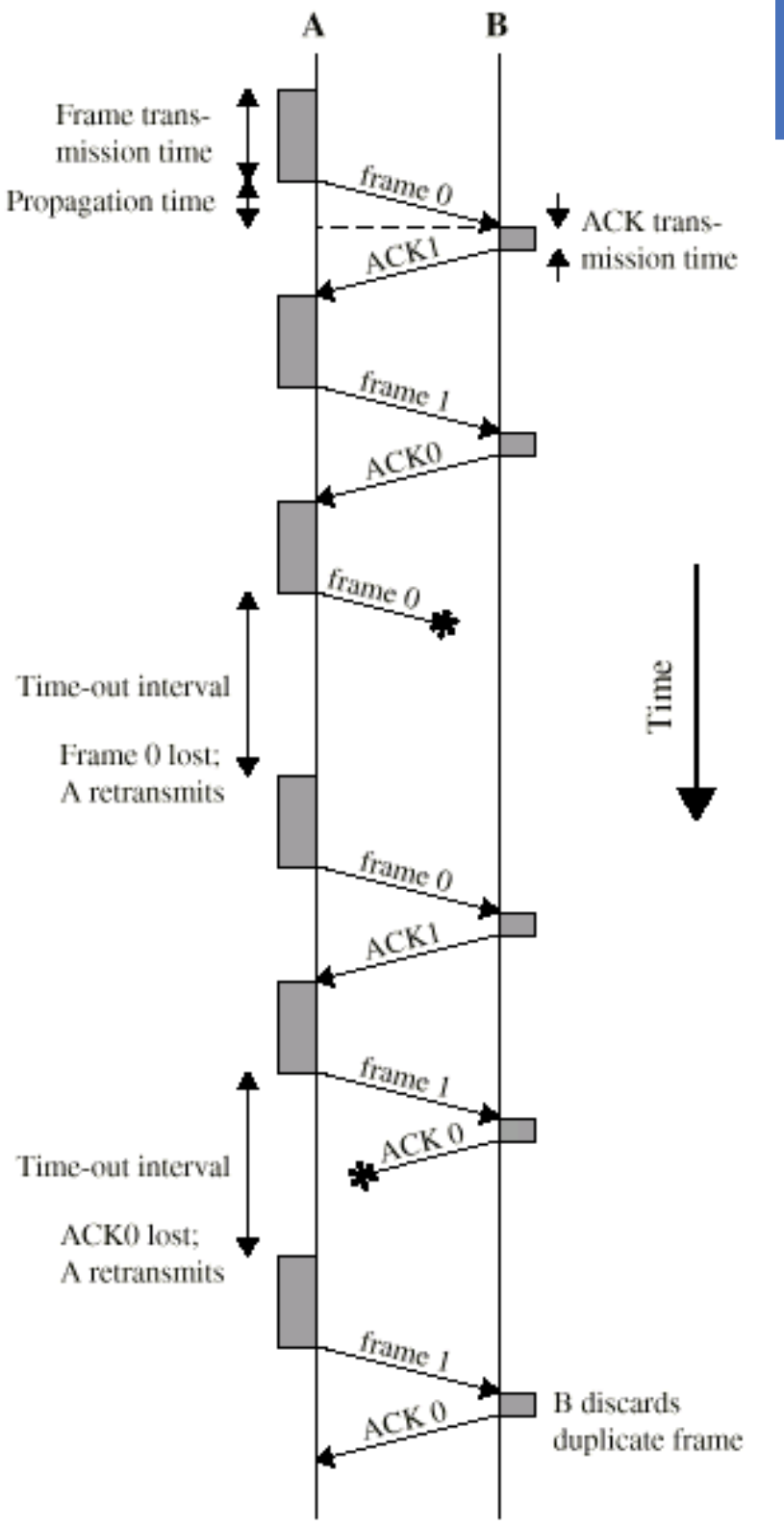


(a) Error-free transmission

(b) Transmission with losses and errors

Stop and Wait Diagram

- Sender doesn't know if the frame was transmitted correctly.
  - Solution: Use acknowledgement frame.
- Acknowledgement frame could be lost.
  - Solutions:
    - Timer.
    - Time-out
    - Resend
- Receiver can't recognize duplicate frames that are resent from sender
  - Solution: Assign a sequence number for each frame



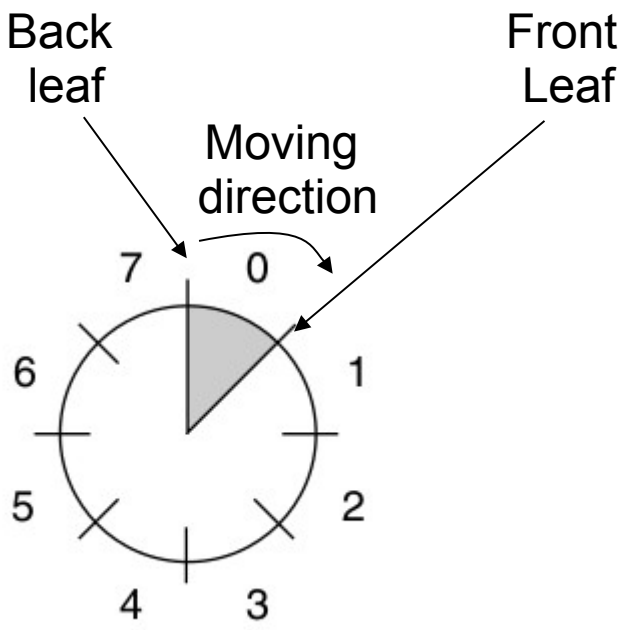
Duplex data transmission

- Stop and Wait: Simplex data transmission
- Need to achieve duplex data transmission to explore maximum channel capacity
- Principle for archiving duplex data transmission:
  - Maintain frame transmission
  - Define frame types: DATA, ACK, NACK
  - Using **piggyback** technology

Sliding windows

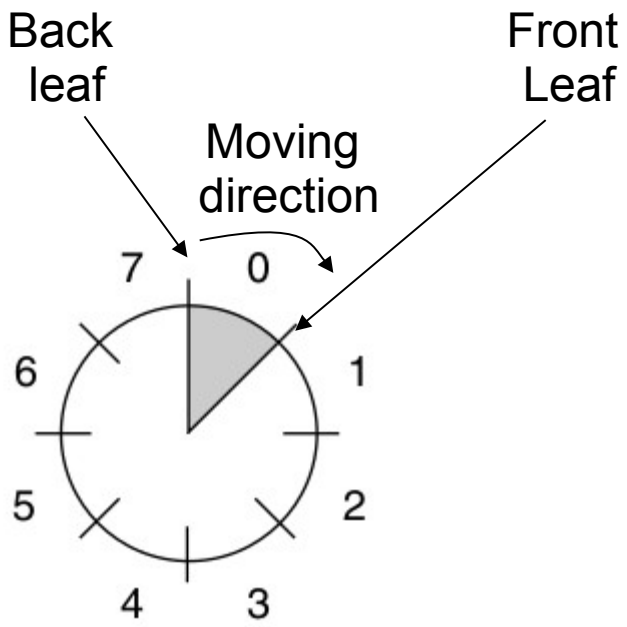
- Instead of sending a frame and then waiting for an acknowledgement frame from the receiver, Sliding window protocol allows a sender to send more frames before returning of acknowledgement frame
- Sending Windows: used by sender to monitor sent frames that acknowledgement frames are being waited for
- Receiving Windows: used by receiver to monitor the frames that are permitted to receive

Sliding Windows



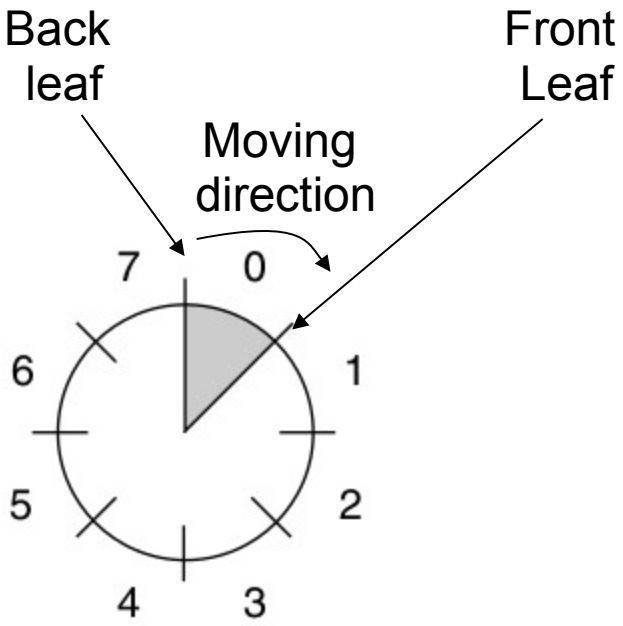
- A sliding window has two leaves: Front leaf and Back leaf. Both leaves have the same moving direction.
- The size of a sliding window is the length from back leaf to front leaf
- Size of sliding window is changed when leafs move:
  - When the front leaf moves, the sliding window becomes larger
  - When the back leaf moves, the sliding windows becomes narrower and it makes the windows turn around a center

Sliding Windows



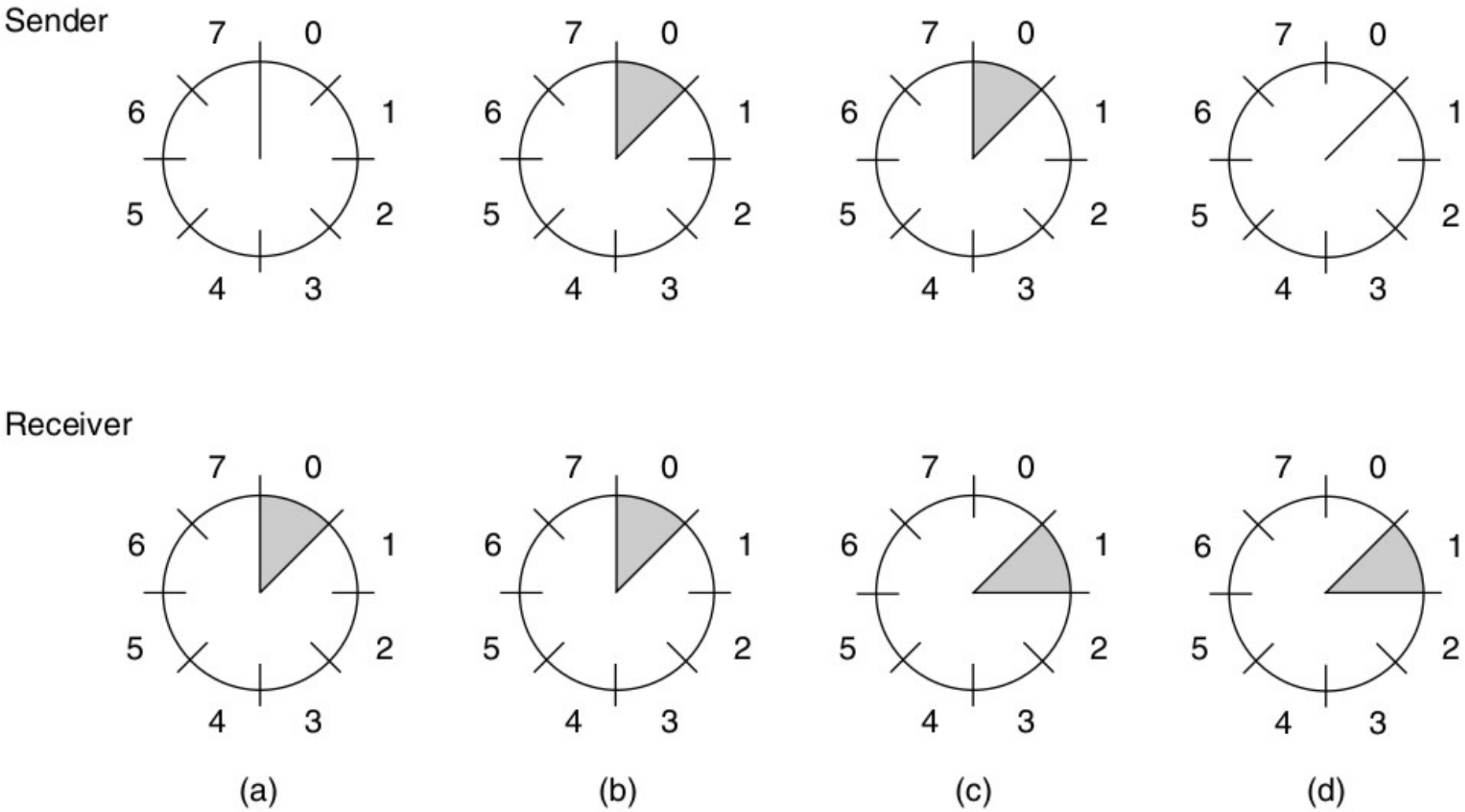
- The smallest size of a sliding window is 0
- The largest size of a sliding window is n-1
- If k bits is used to represent the sequence number of frames [0- (2^k-1)], then a sliding window will be divided in to 2^k positions that are corresponding to 2^k frame
- For sending window, potions are currently inside the window represent the sequence numbers of frames that sender has sent and is waiting for acknowledgements from receiver. Potions out-site the window represent the sequence numbers of frames that the sender can send more. However, the size of the window mustn't be larger than the maximum size of the window

Sliding Windows



- For a receiver, the positions inside its sliding window represent the sequence numbers of frames that the receiver is permitted to receive
- Maximum size of a sliding window represents the size of buffer for storing temporarily received frames before processing them
- Supposing that the receiver has a buffer of 4 frame sizes, then the maximum size of the sliding window will be 4.

Example with maximum window size of 1

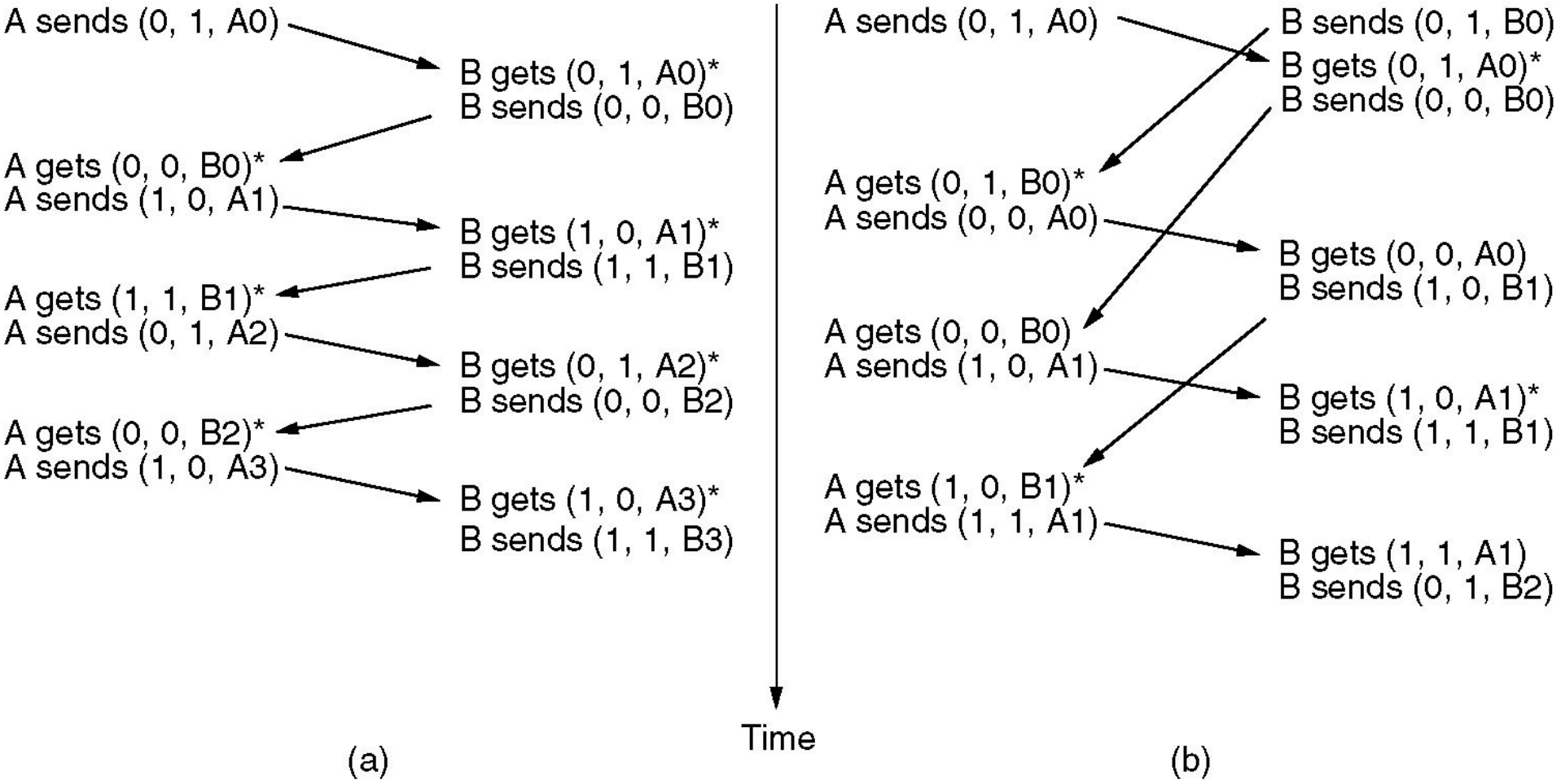


Bài tập

Vẽ hình minh họa hoạt động của cửa sổ trượt có 8 vị trí (0-7) với kích thước cửa sổ nhận tối đa là 2

Viết chương trình minh họa

Example with maximum window size of 1



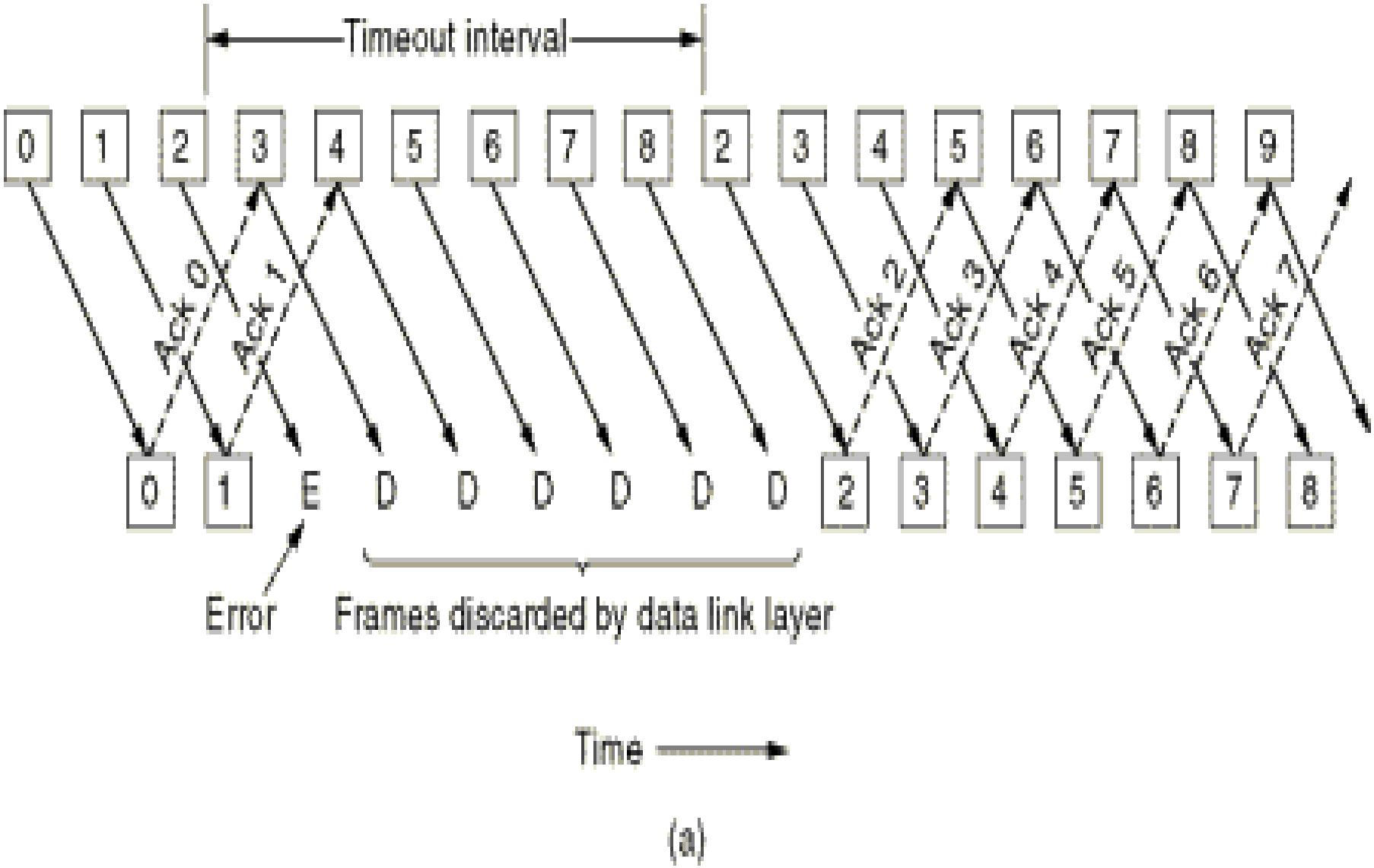
(a): Sending and receiving frame with normal order

(b): Sending and receiving frame with random order

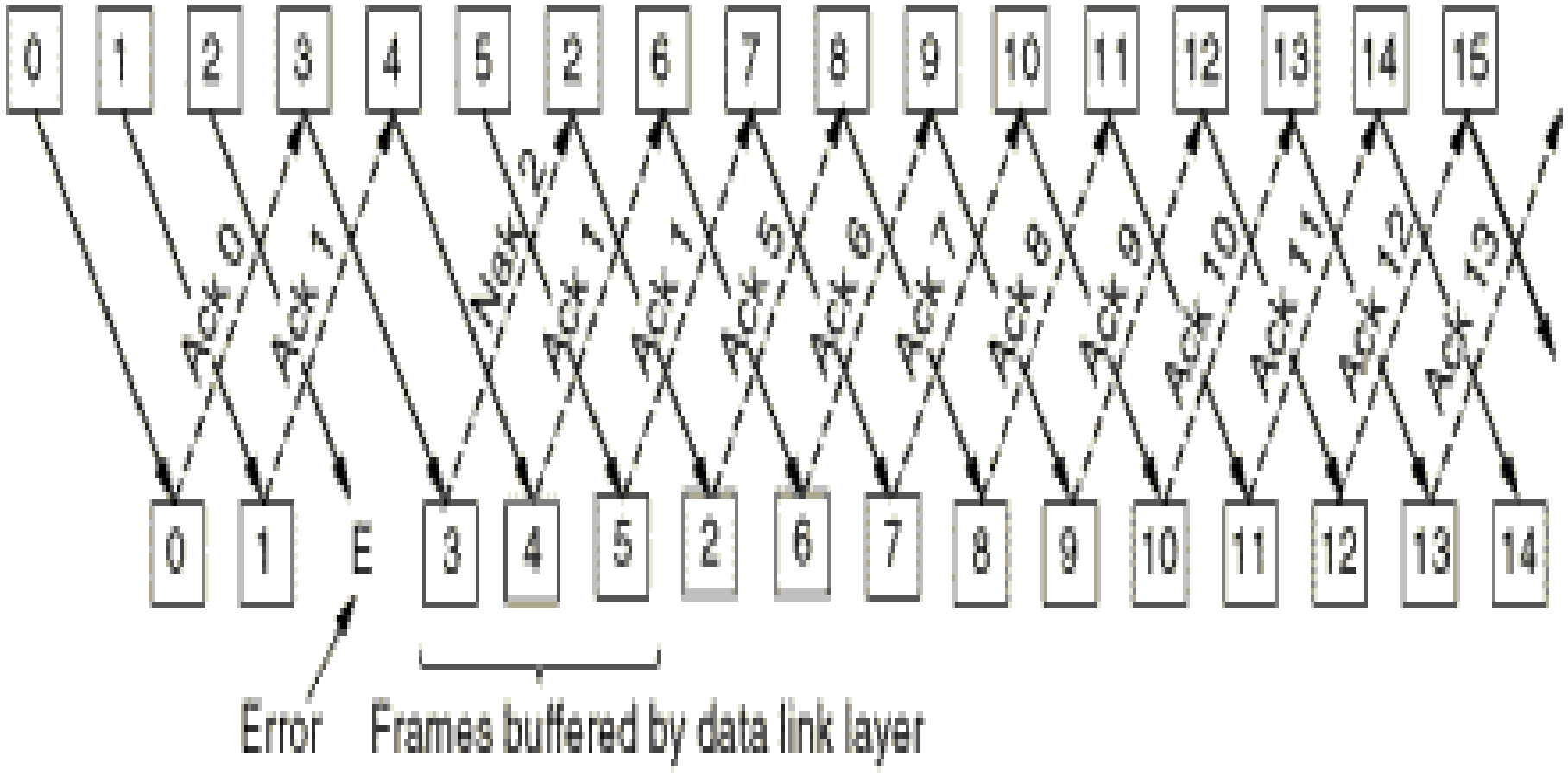
Go-Back-N protocol

- When a receiver receives an error frame it discards the frame.
- Because there is no acknowledgement frame for the error frame, a time-out event for this error frame will be sent to the sender. The sender then resends the error frame and all the frames that were sent after the error frame.

Go-Back-N protocol

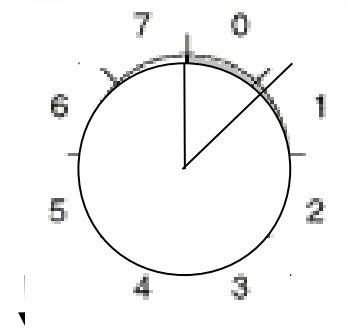


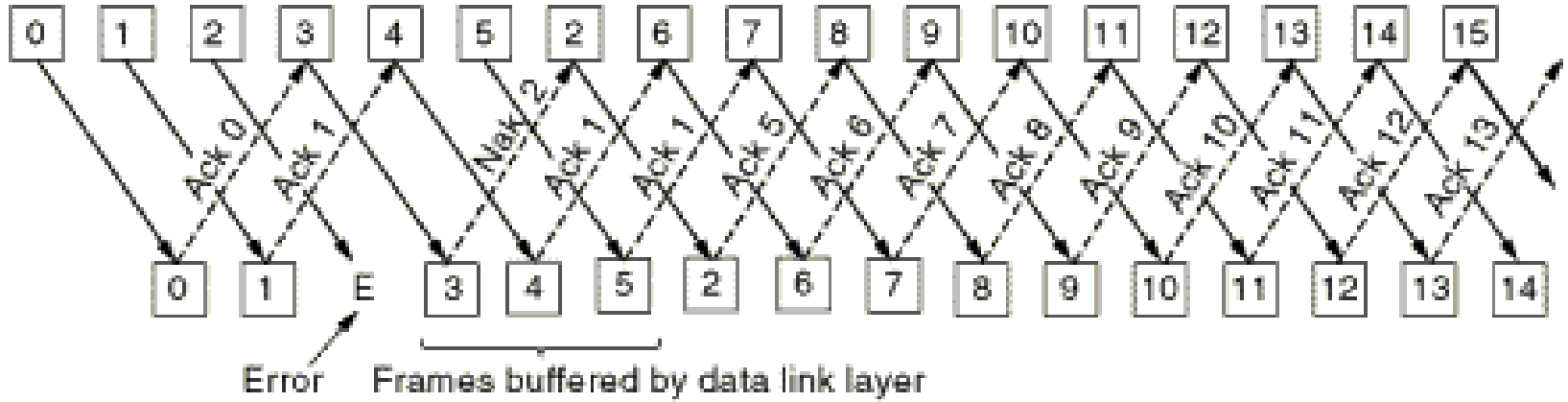
Selective Repeat protocol



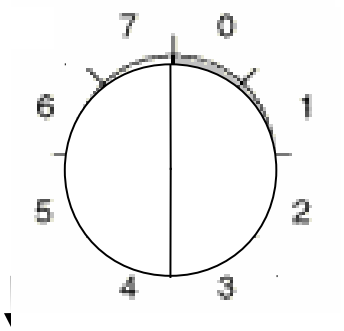
(b)

Giao thức Selective Repeat

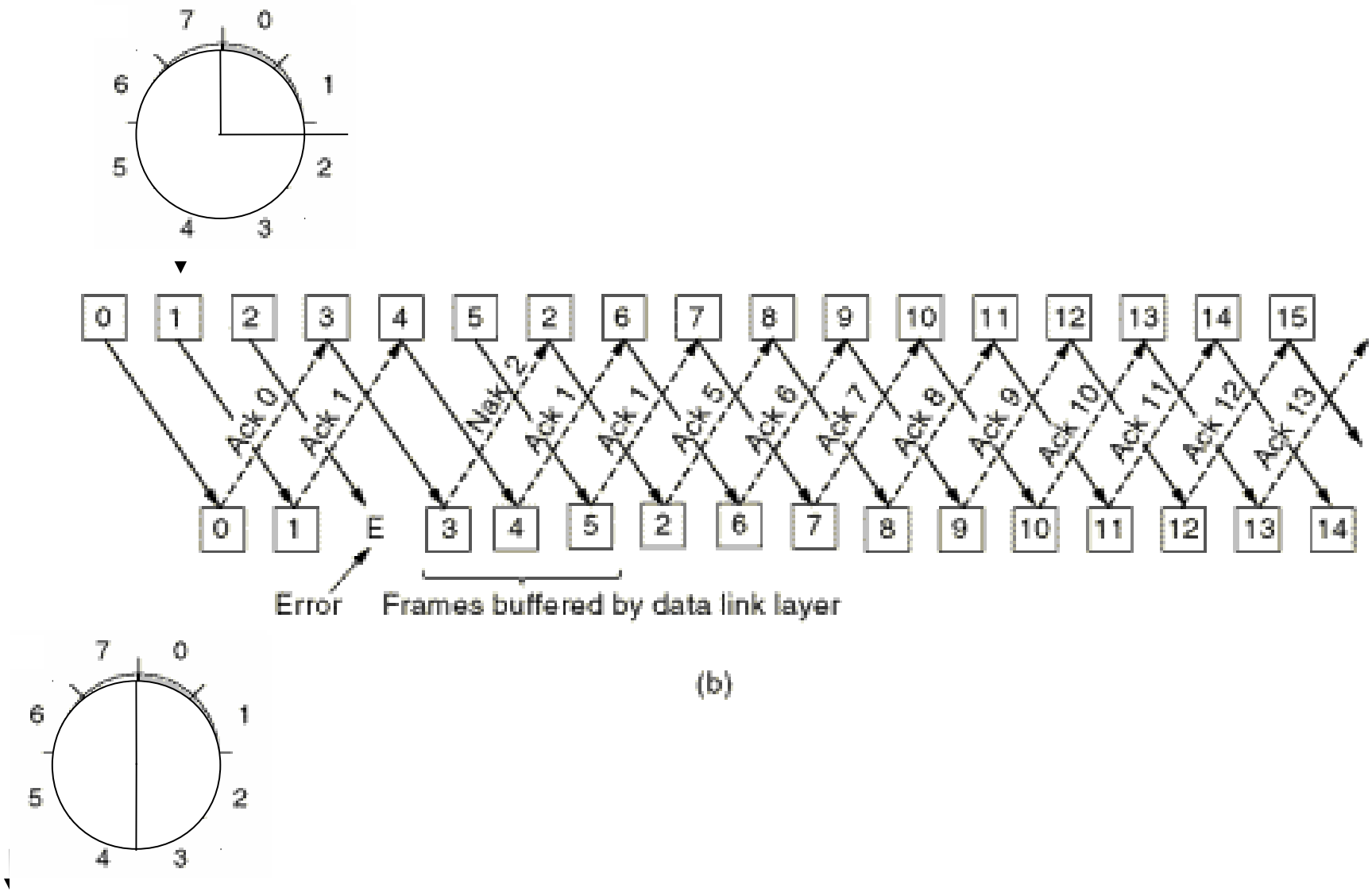




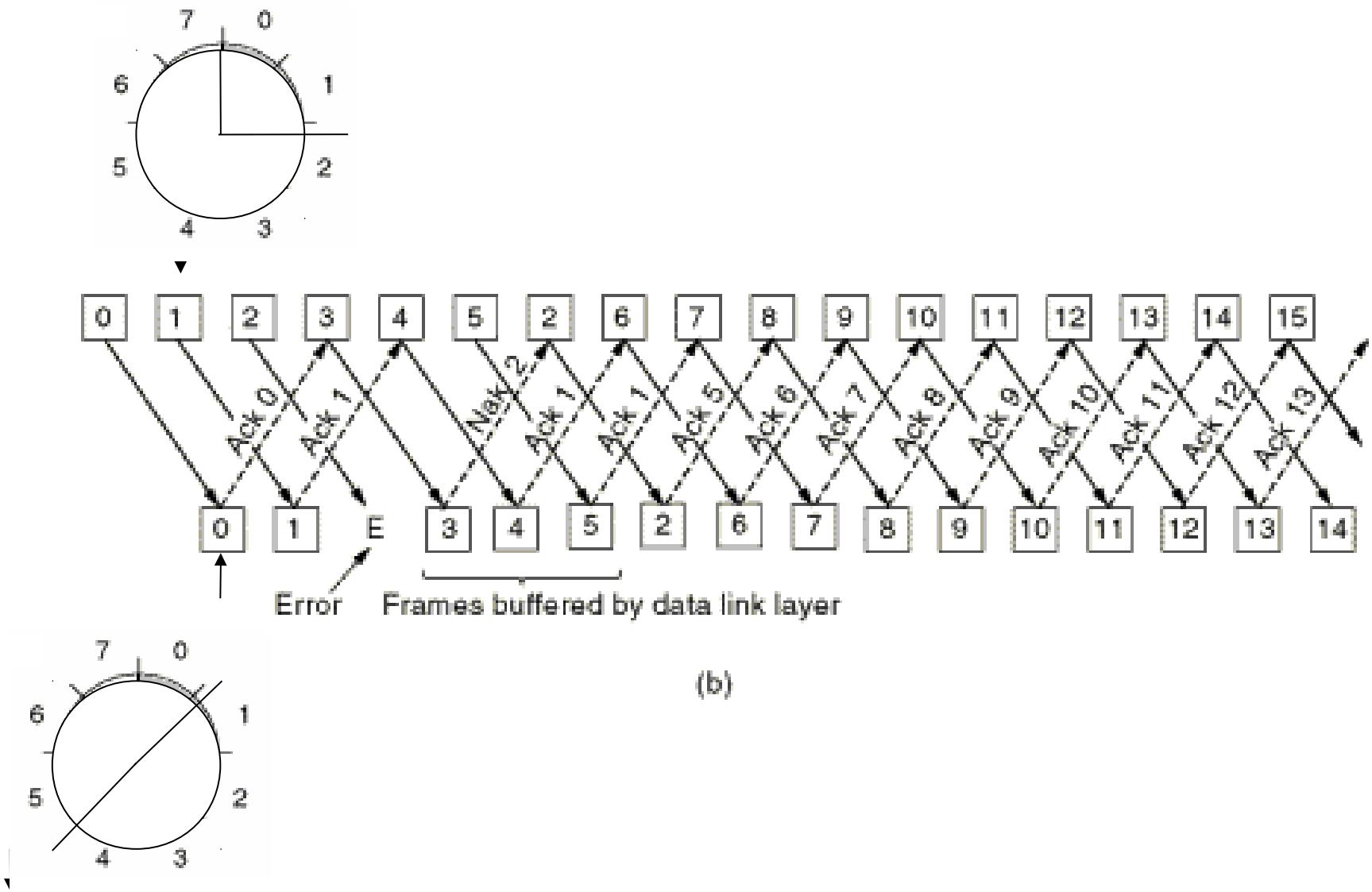
(b)



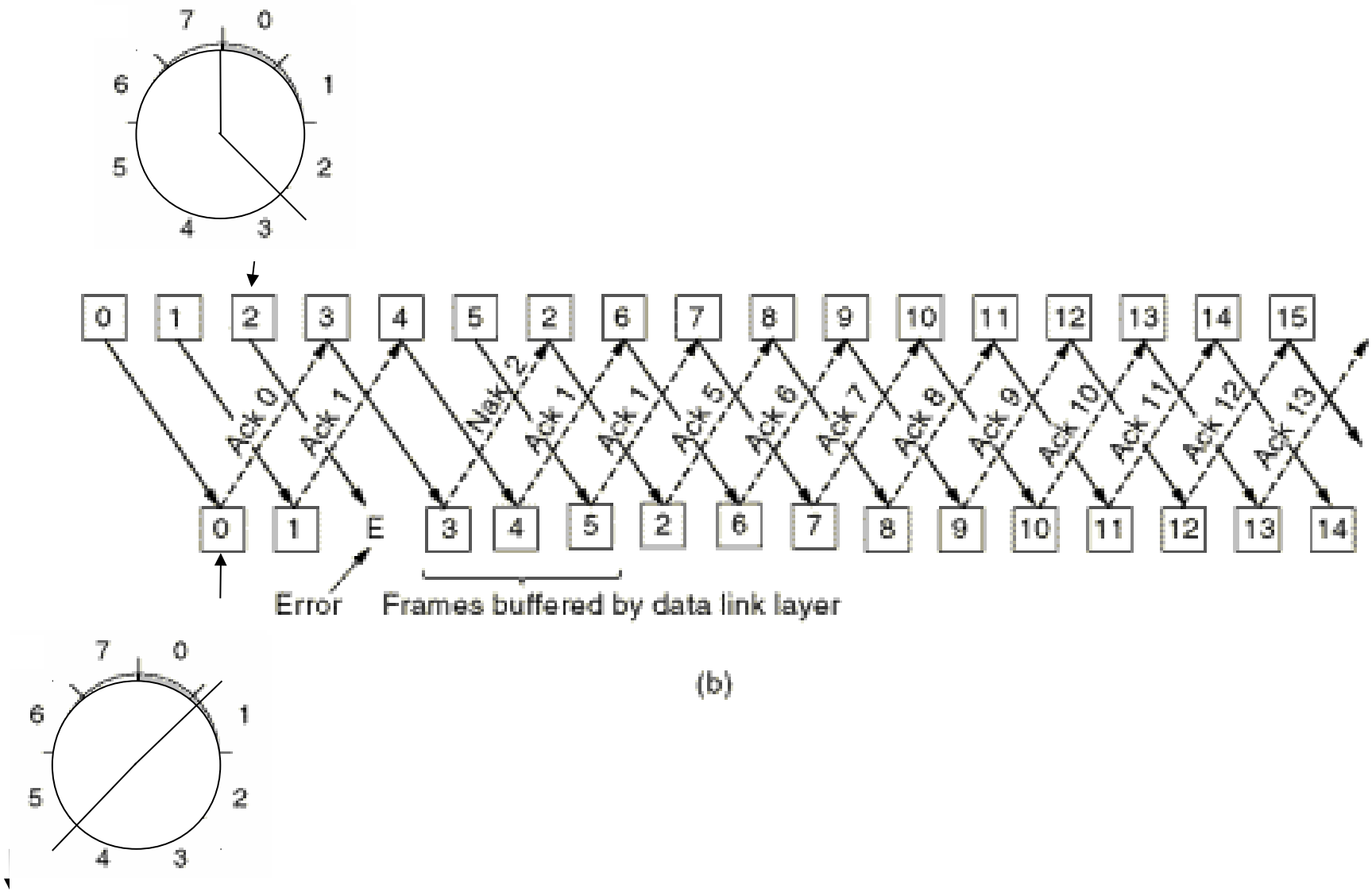
Giao thức Selective Repeat



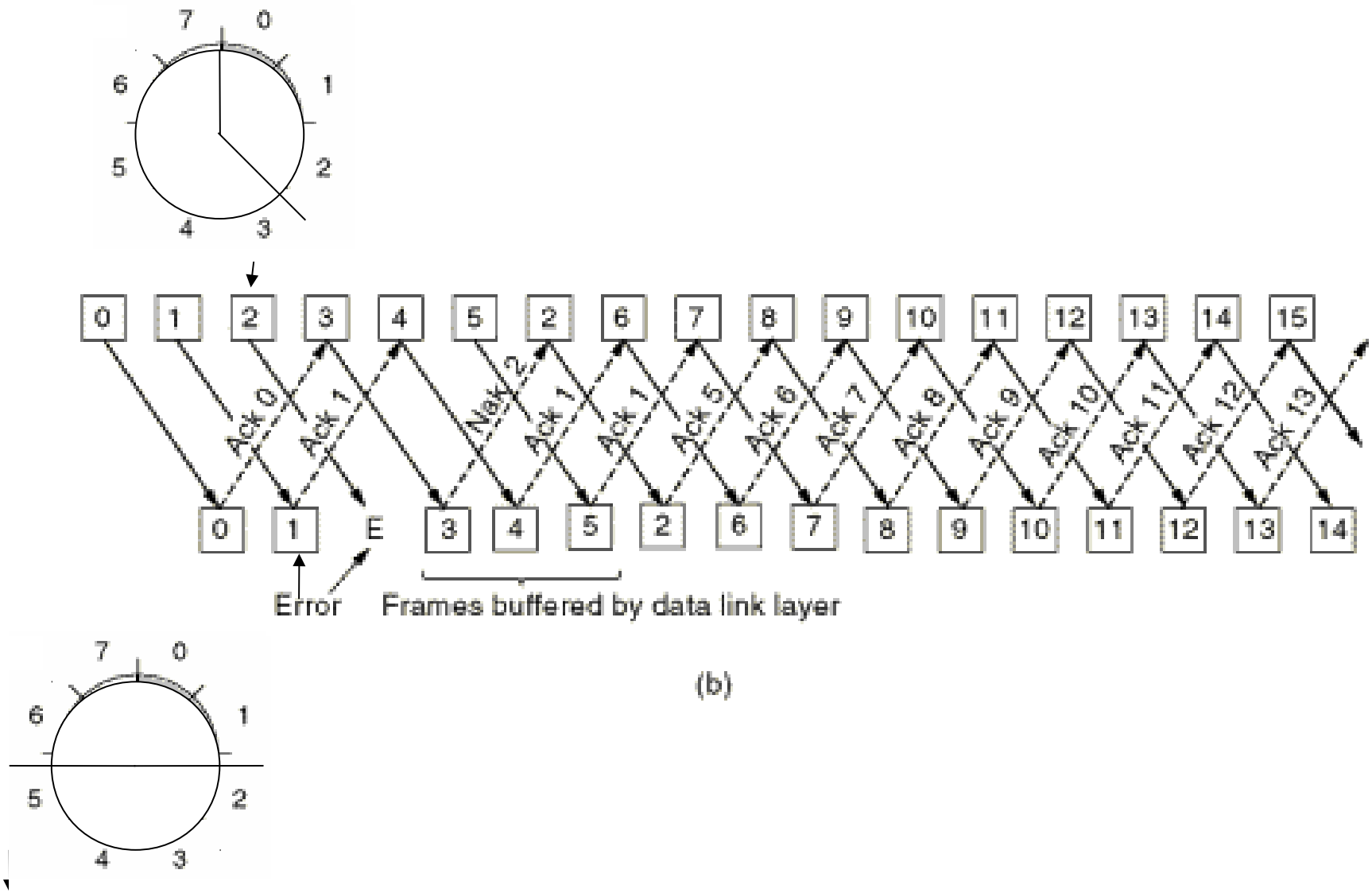
Giao thức Selective Repeat



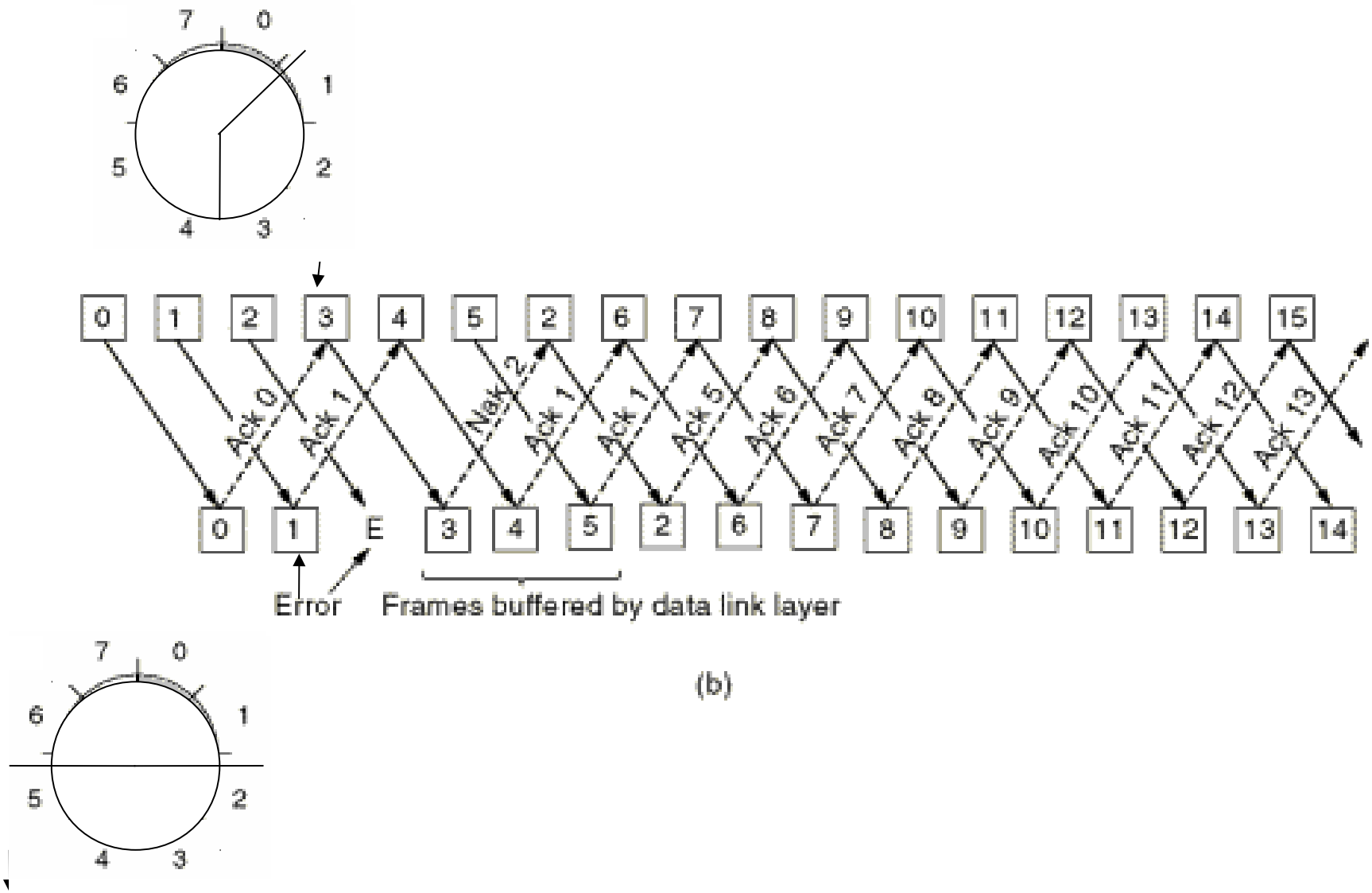
Giao thức Selective Repeat



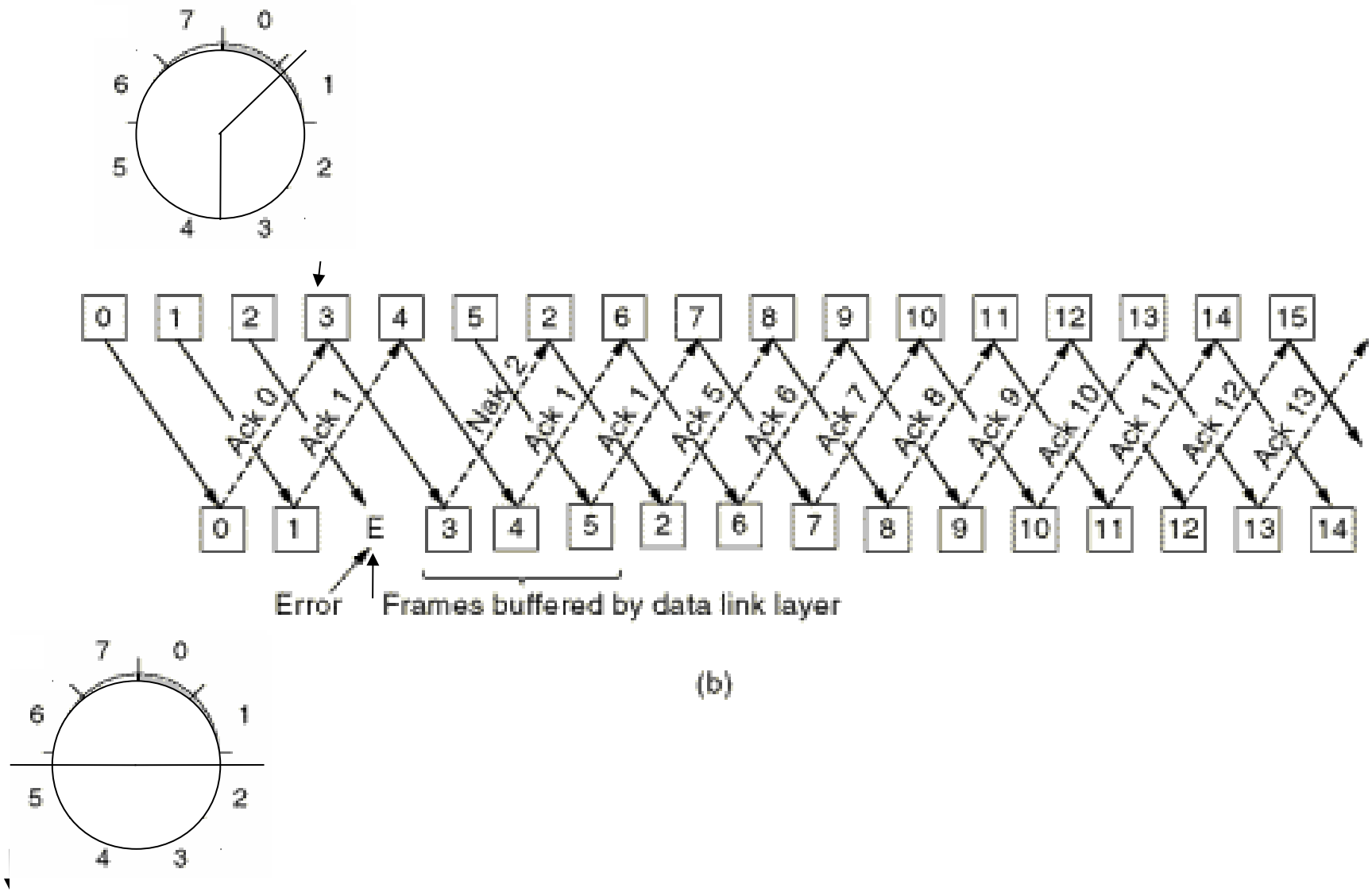
Giao thức Selective Repeat



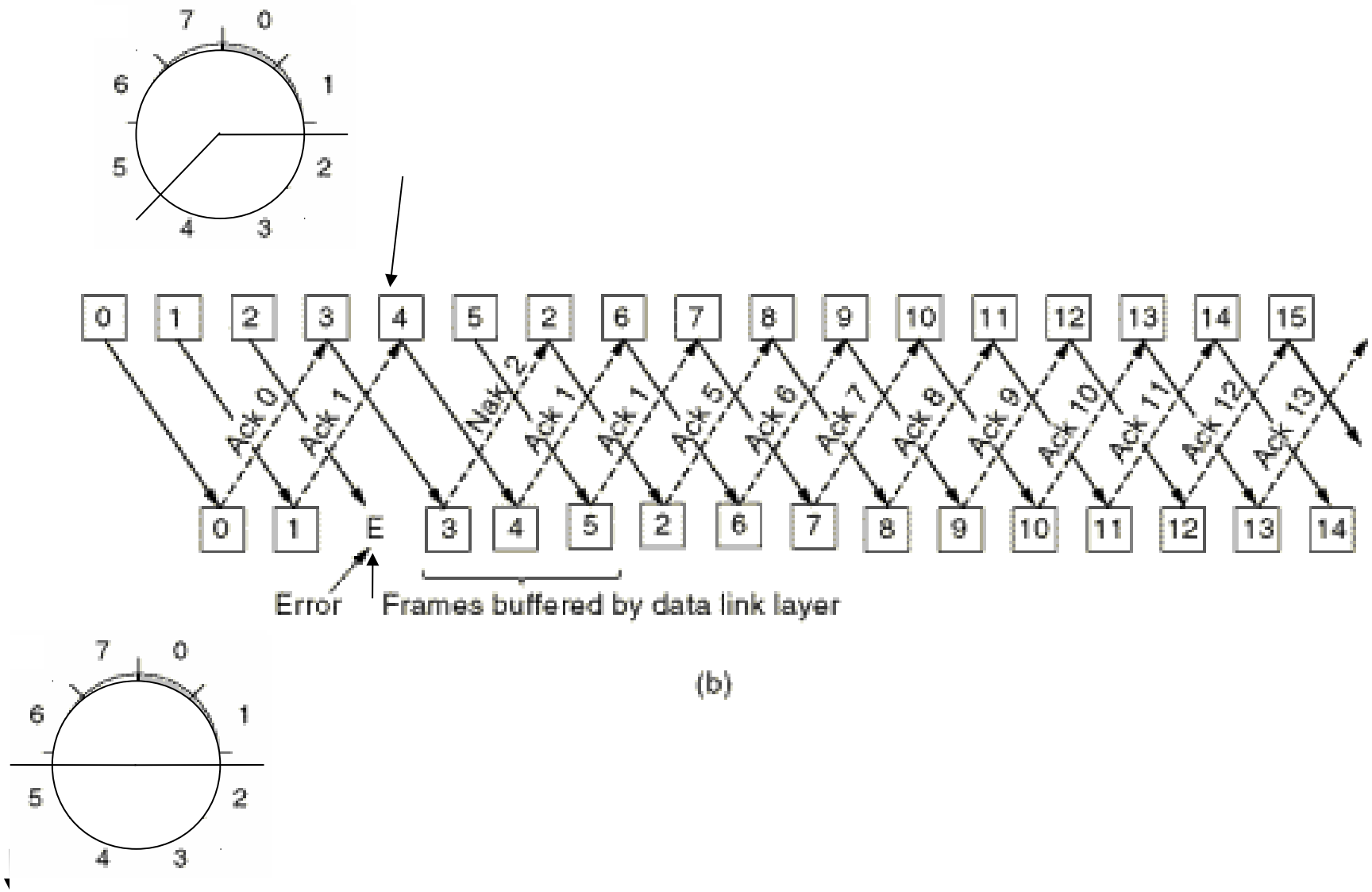
Giao thức Selective Repeat



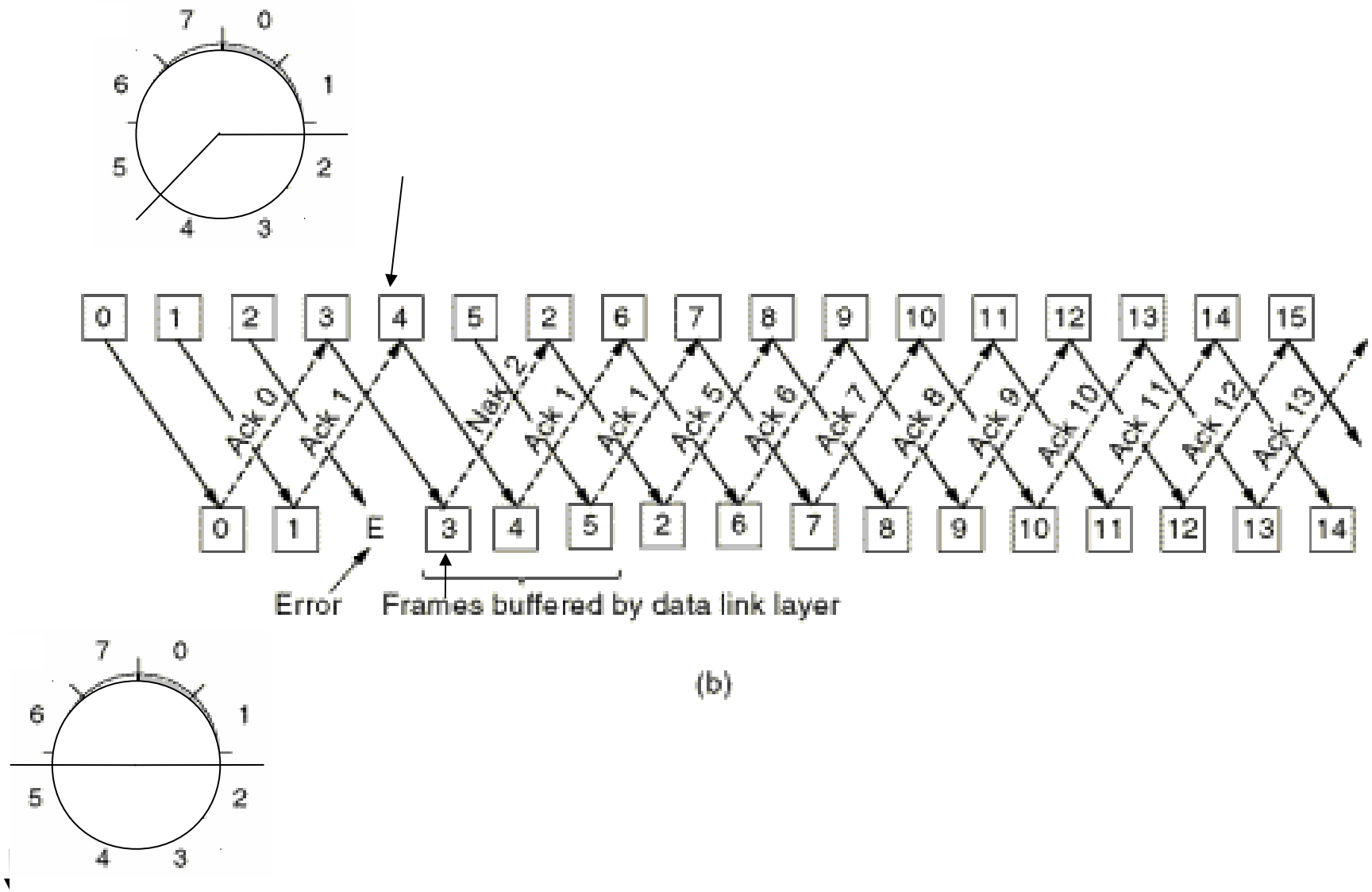
Giao thức Selective Repeat



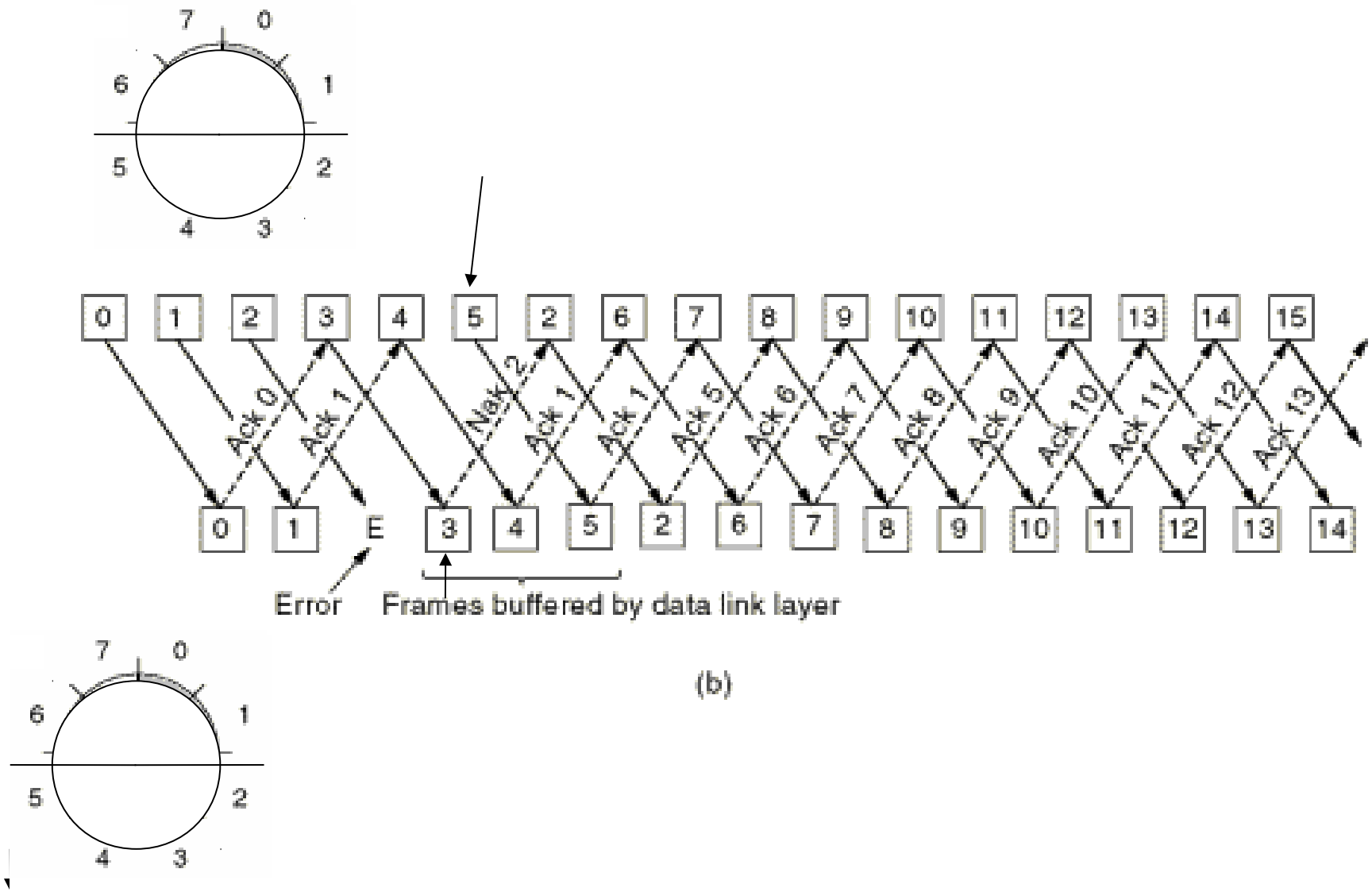
Giao thức Selective Repeat



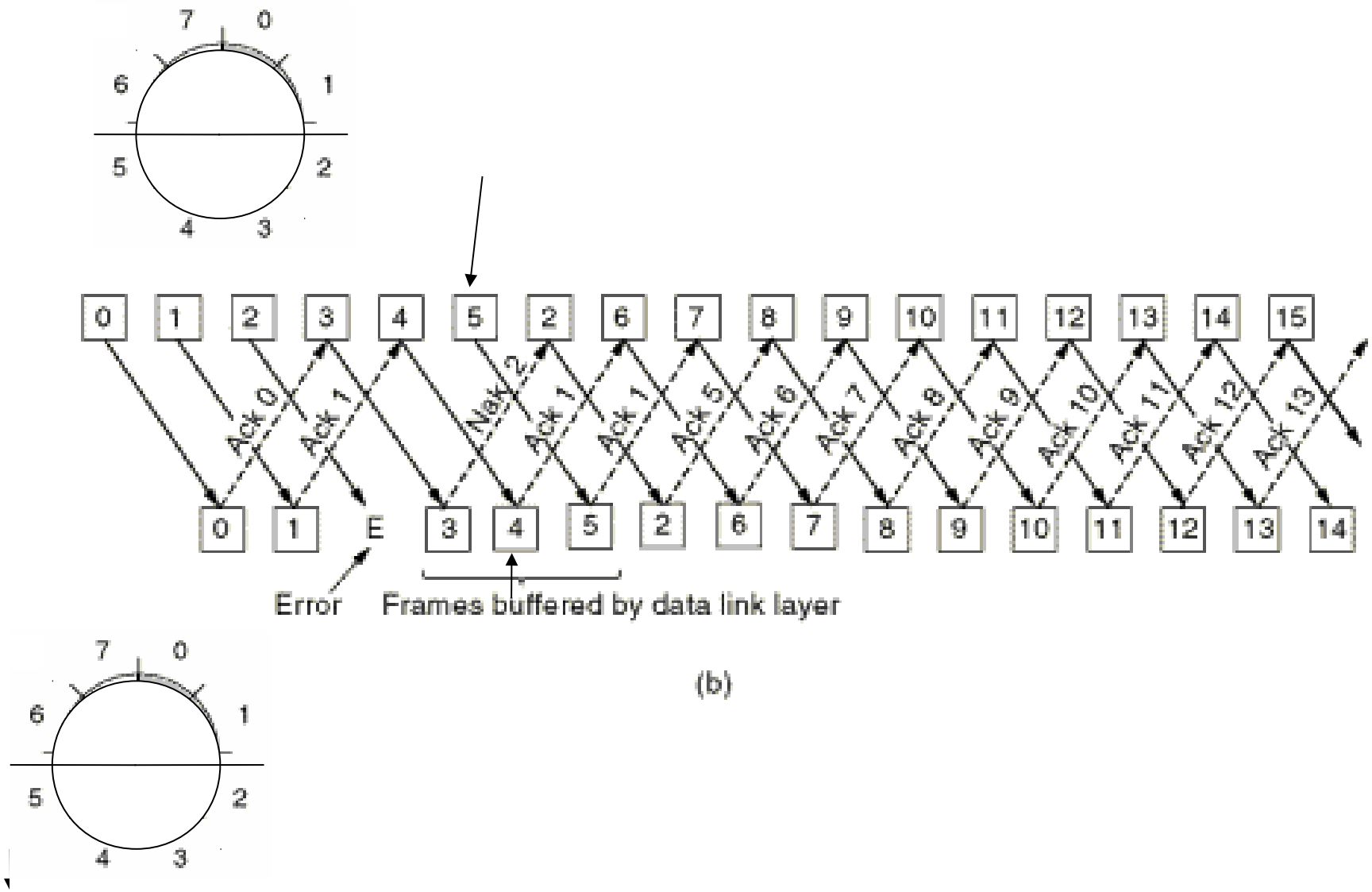
Giao thức Selective Repeat



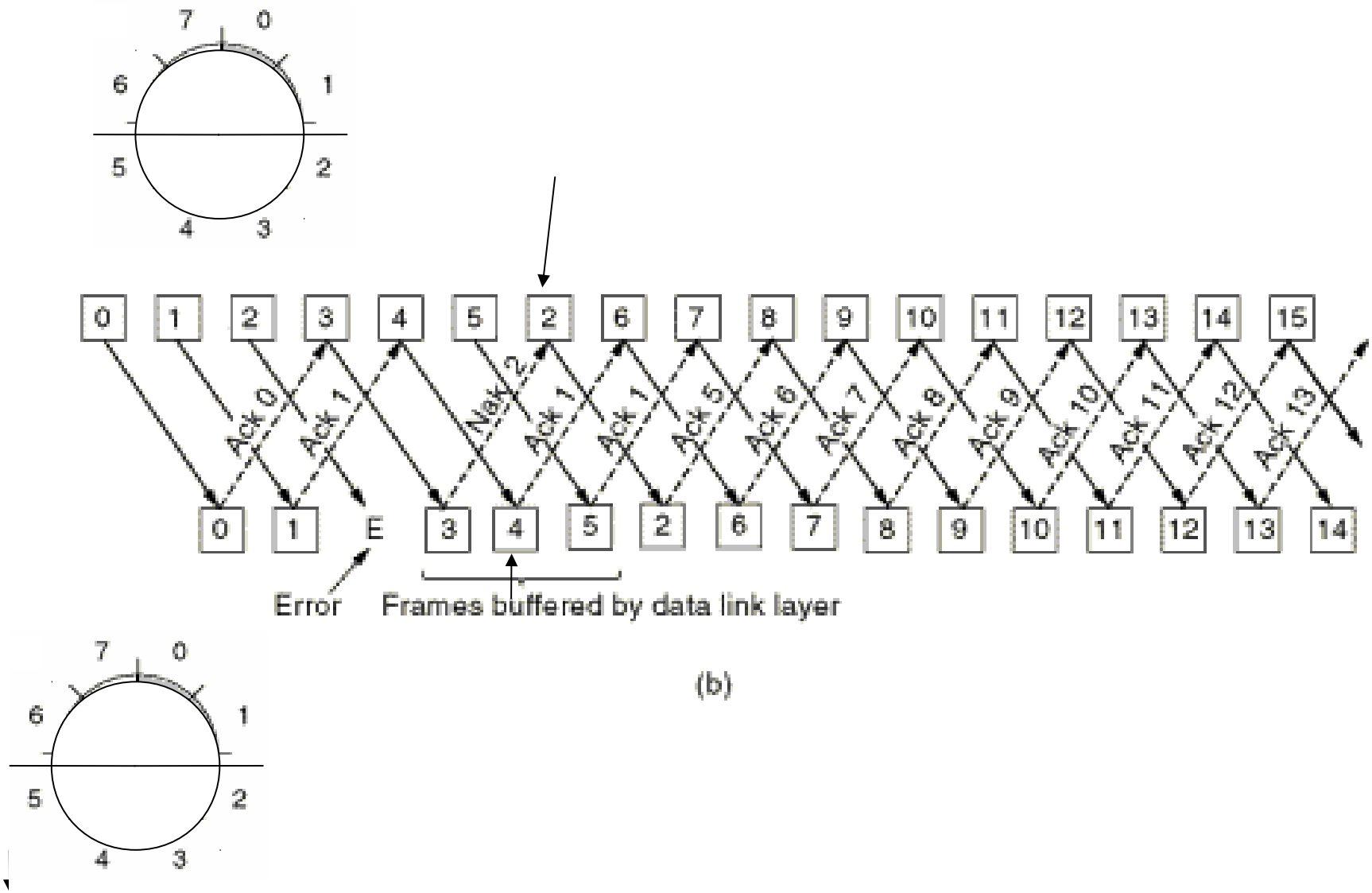
Giao thức Selective Repeat



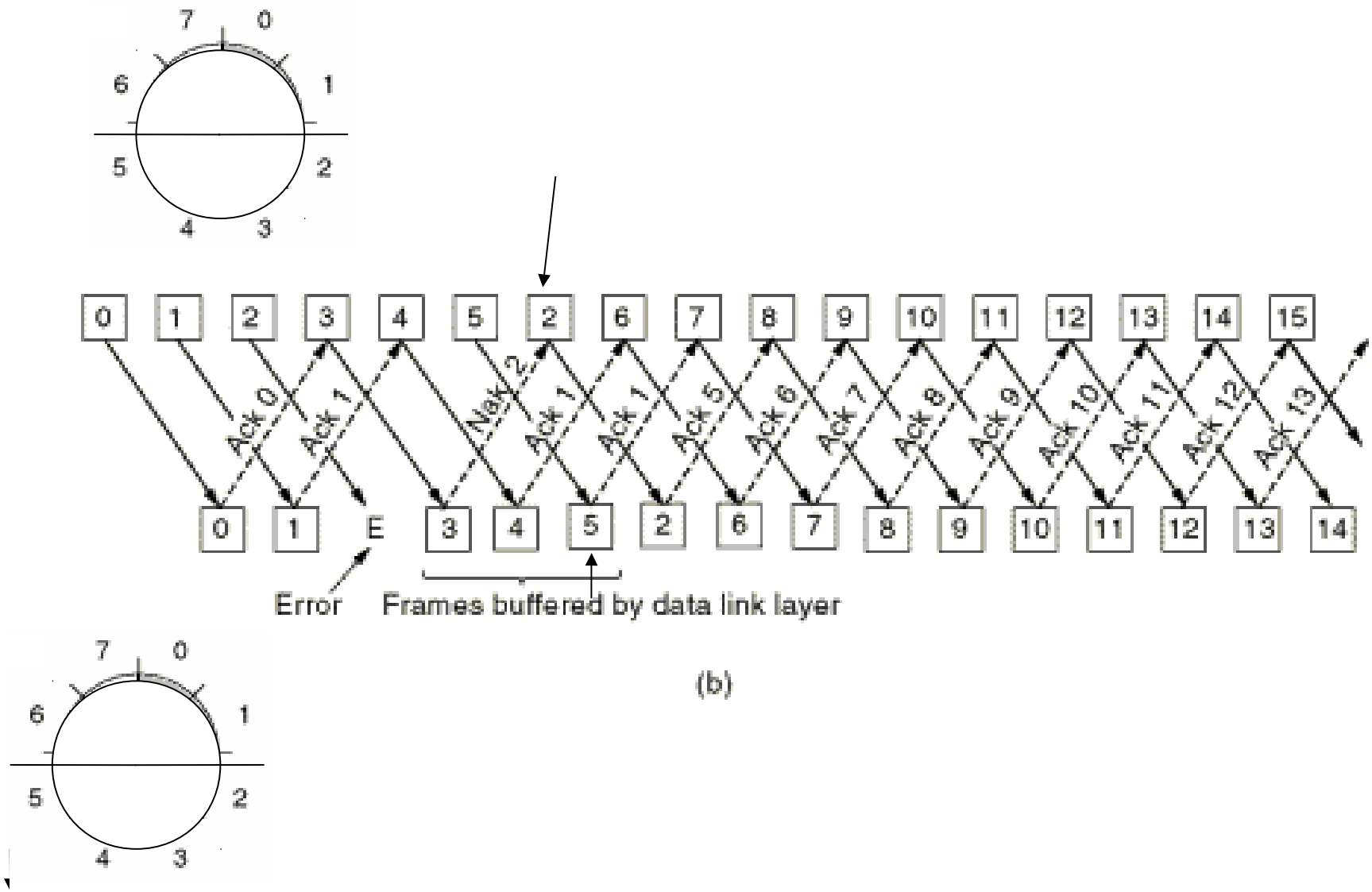
Giao thức Selective Repeat



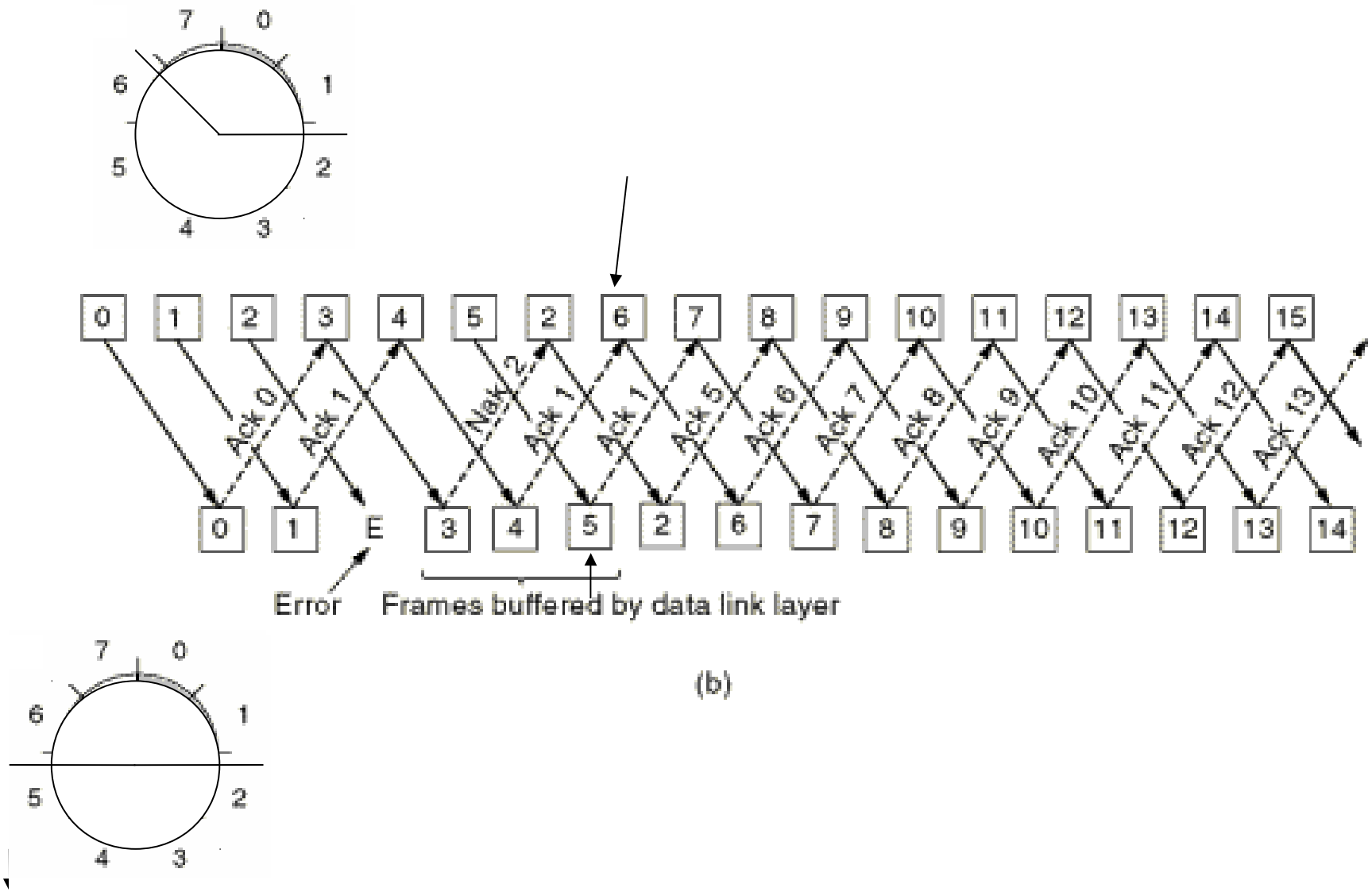
Giao thức Selective Repeat



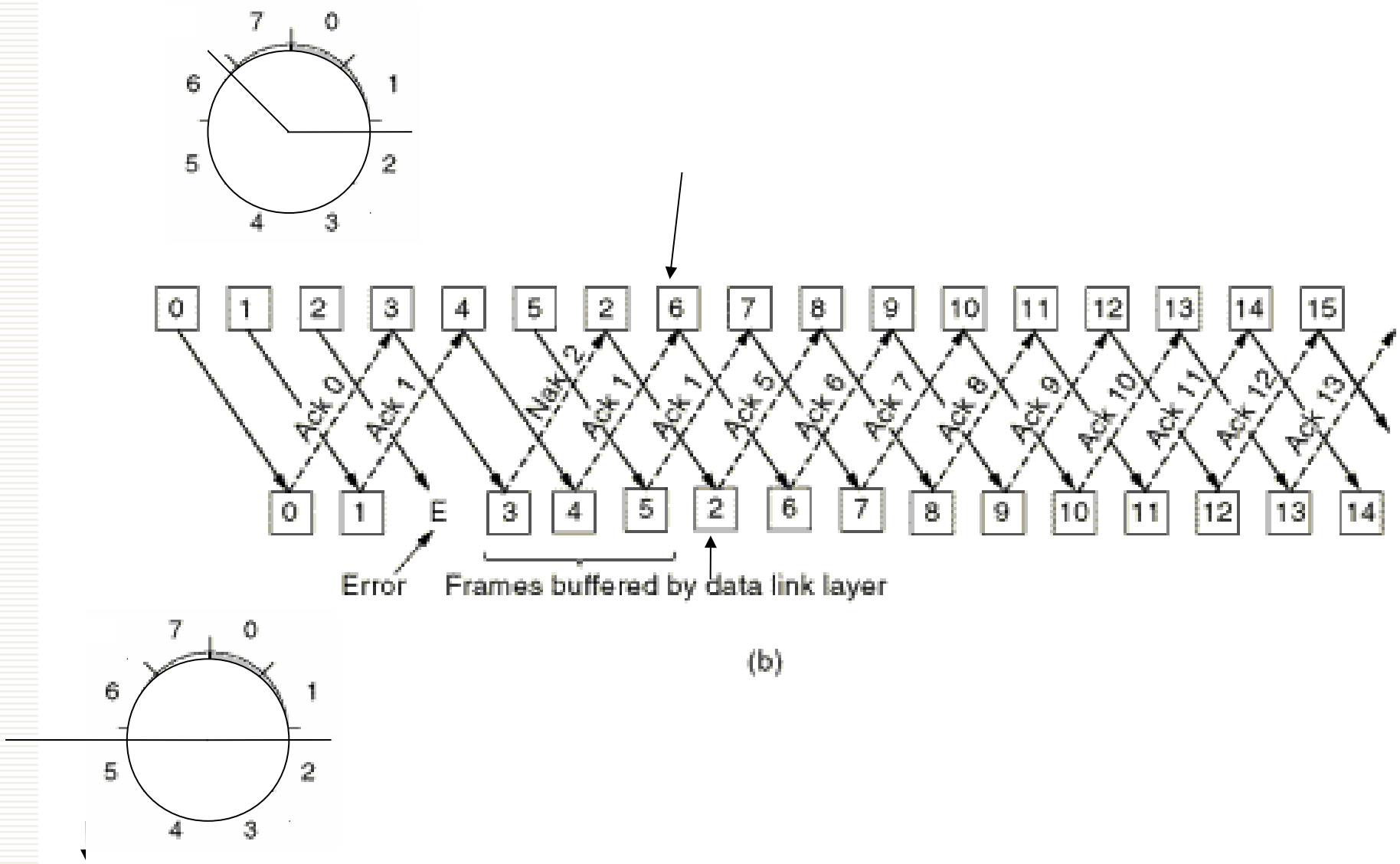
Giao thức Selective Repeat



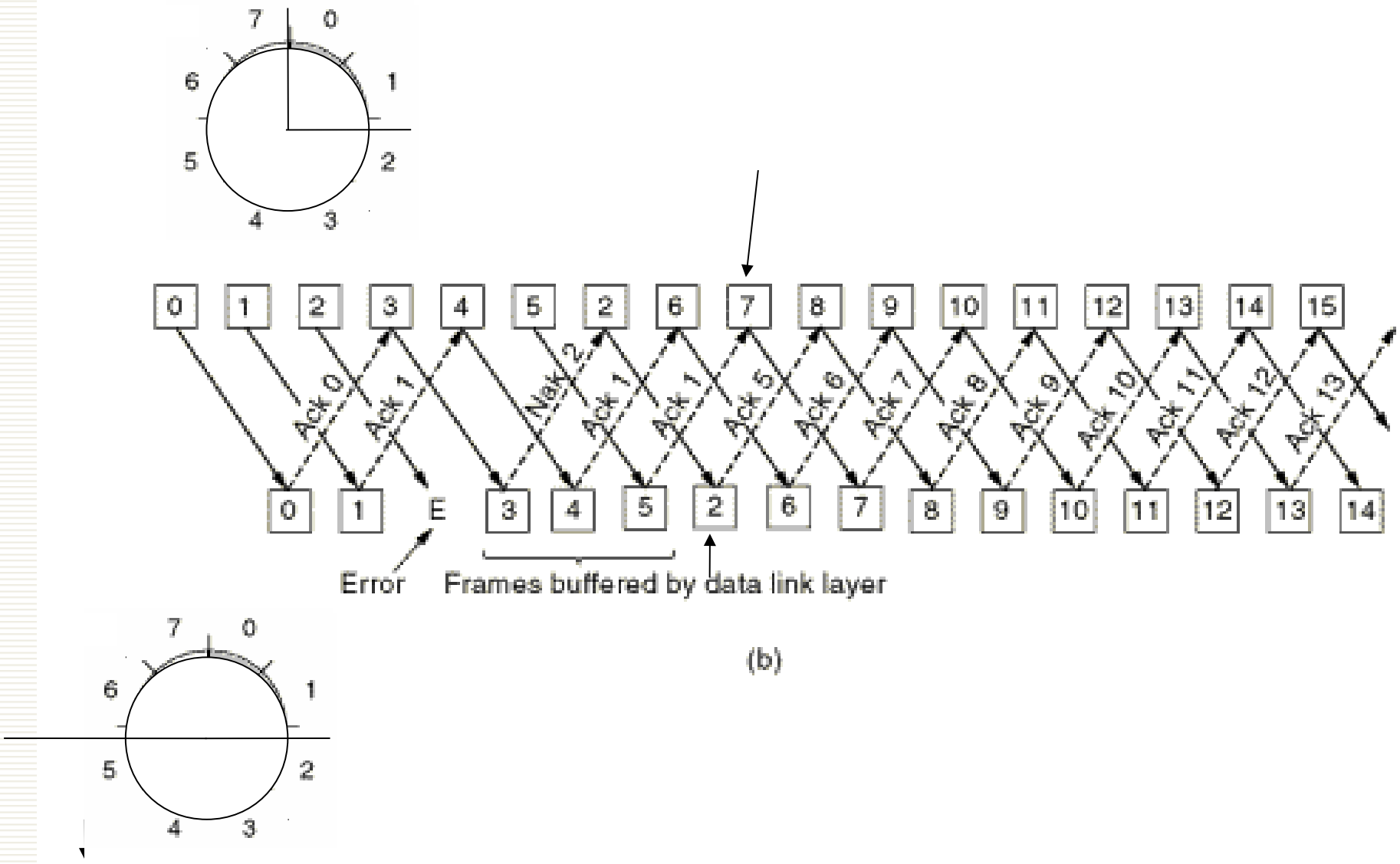
Giao thức Selective Repeat



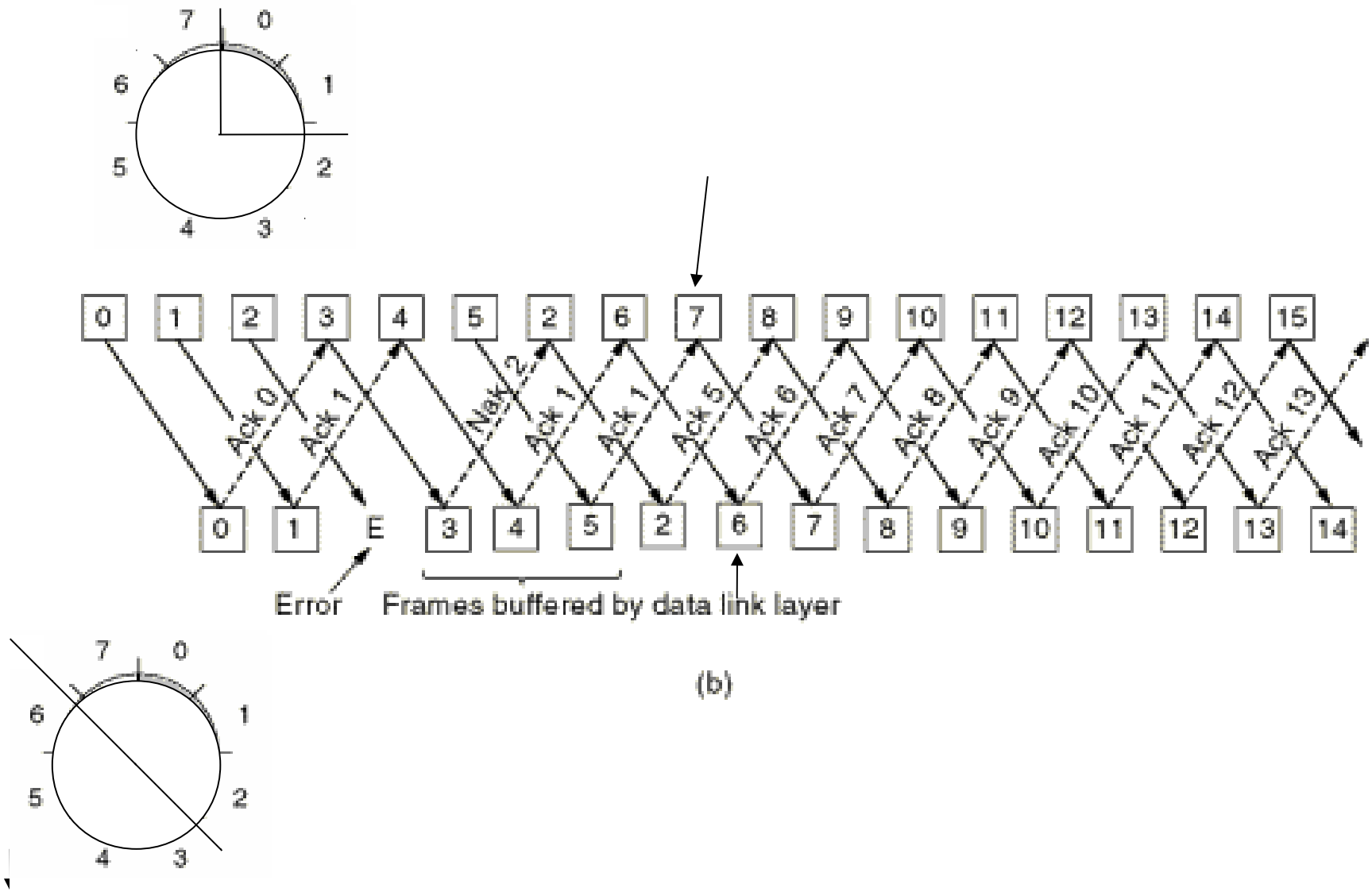
Giao thức Selective Repeat



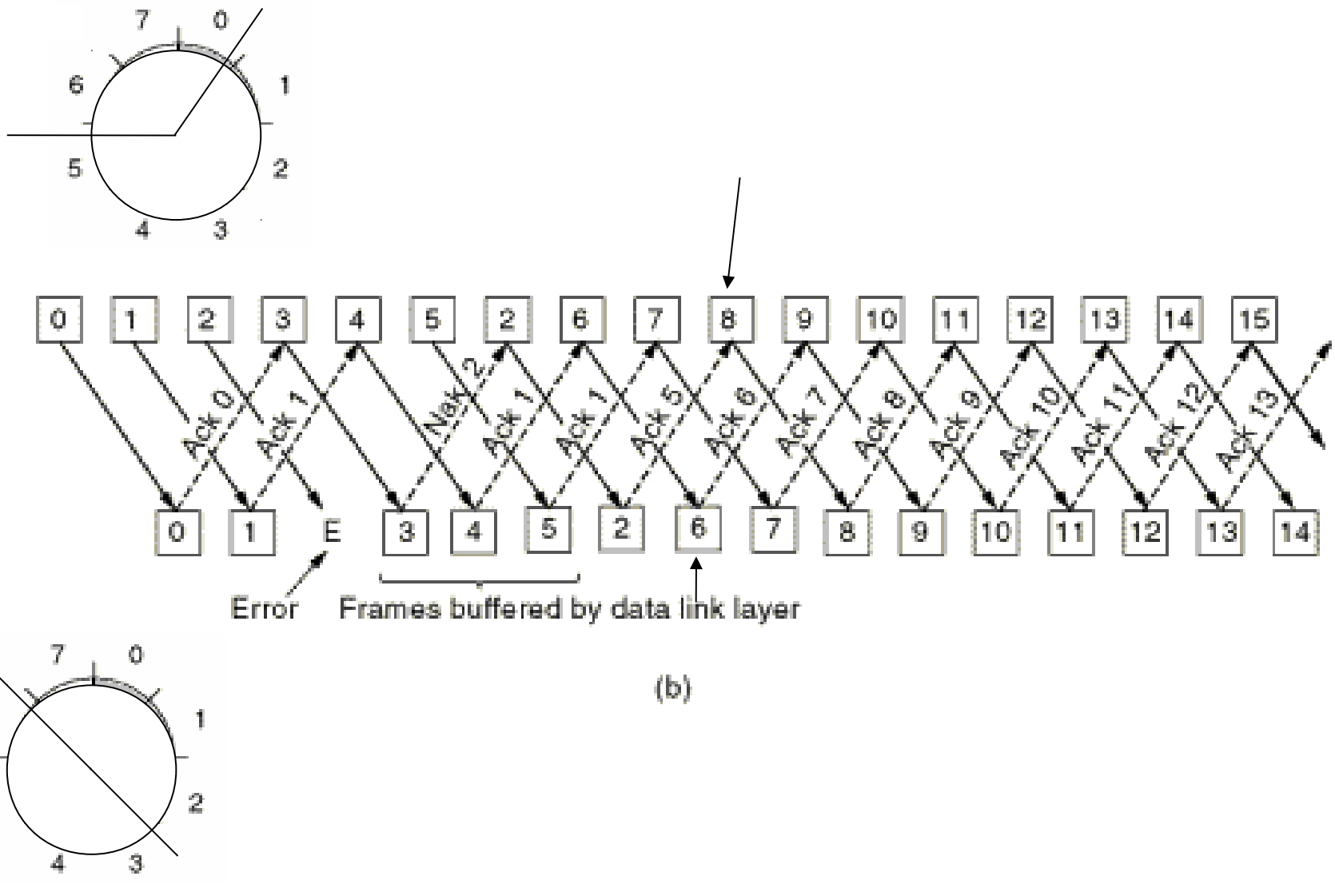
Giao thức Selective Repeat



Giao thức Selective Repeat



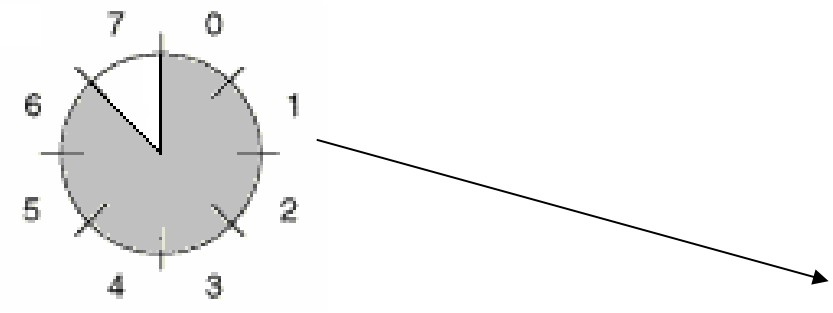
Giao thức Selective Repeat



Calculate the size of a sliding window

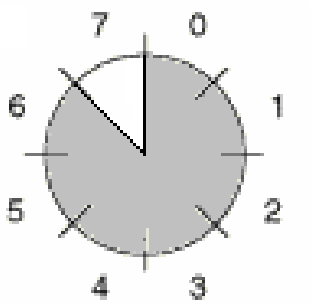
Given a sliding window using 3 bits for numbering frame → the size of window will be 7

Sender



Sent and waiting for acknowledgment for frames 0,1,2,3,4,5,6

Receiver

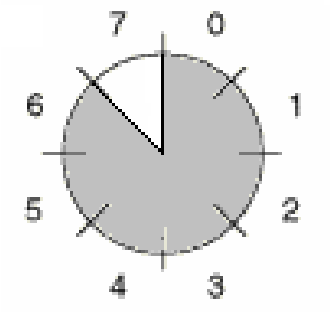


Receiving ready for frames 0,1,2,3,4,5,6

Calculate the size of a sliding window

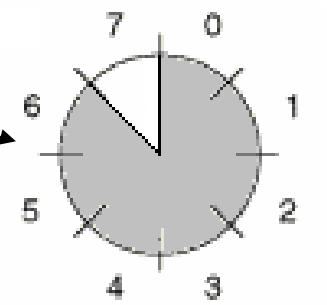
Given a sliding window using 3 bits for numbering frame → the size of window will be 7

Sender



Sent and waiting for acknowledgment for frames 0,1,2,3,4,5,6

Receiver

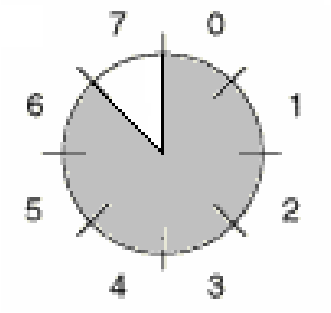


Received frames 0,1,2,3,4,5,6,
Checking for error

Calculate the size of a sliding window

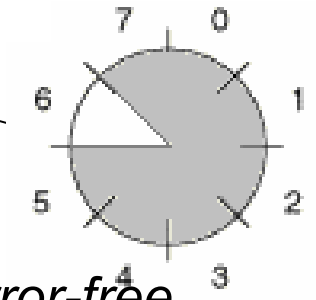
Given a sliding window using 3 bits for numbering frame → the size of window will be 7

Sender



Sent and waiting for acknowledgment for frames 0,1,2,3,4,5,6

Receiver

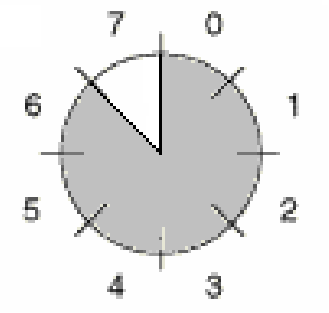


Frame 0,1,2,3,4,5,6 error-free
Sent acknowledgement for frames 0,1,2,3,4,5,6
Receiving ready for frames 7,0,1,2,3,4,5

Calculate the size of a sliding window

Given a sliding window using 3 bits for numbering frame → the size of window will be 7

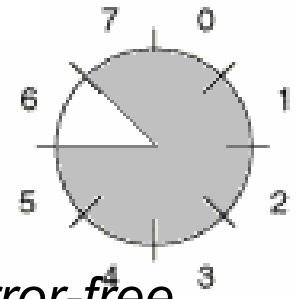
Sender



Transmission error
Acknowledgement frame doesn't reach sender

X

Receiver



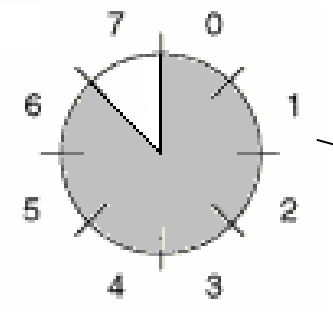
Sent and waiting for acknowledgment for frames 0,1,2,3,4,5,6

Frame 0,1,2,3,4,5,6 error-free
Sent acknowledgement for frames 0,1,2,3,4,5,6
Receiving ready for frames 7,0,1,2,3,4,5

Calculate the size of a sliding window

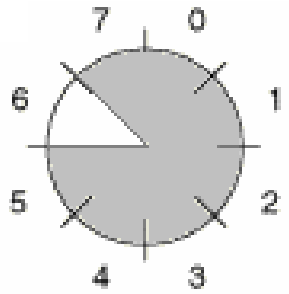
Given a sliding window using 3 bits for numbering frame → the size of window will be 7

Sender



Time-out for frame 0
Resend frame 0
Waiting acknowledgement for frames 0,1,2,3,4,5,6

Receiver

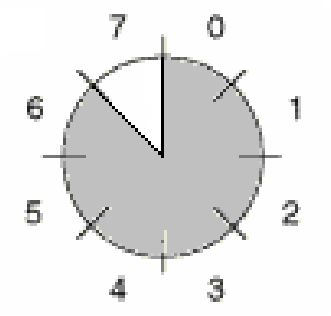


Receiving ready for frames 7,0,1,2,3,4,5

Calculate the size of a sliding window

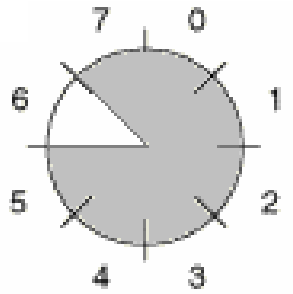
Given a sliding window using 3 bits for numbering frame → the size of windows will be 7

Sender



Time-out for frame 0
Resend frame 0
Waiting acknowledgement for frames 0,1,2,3,4,5,6

Receiver



frame 0 arrives, it is a waiting frame → receive frame 0 and send it to network layer → network layer receives duplicate frame 0

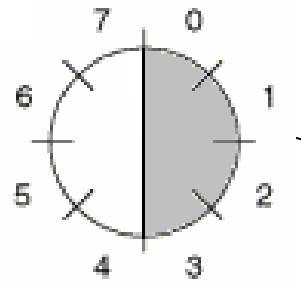
Calculate the size of a sliding window

- It's required that the new receiving window doesn't overlap the old one.
- Maximum size of sliding window is just a half of the number used for frame sequence number
- Examples:
  - If 3 bits are used for frame sequence number (from 0 to 7) then the maximum size of the sliding window is (7-0+1)/2 =4.
  - If 4 bits are used for frame sequence number (from 0 to 15) then the maximum size of the sliding window is (15-0+1)/2 =8

Calculate the size of a sliding window

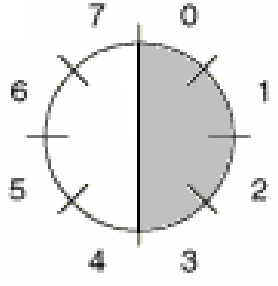
Given a sliding window using 3 bits for numbering frame → the size of windows will be 4

Sender



Sent and waiting acknowledgement for frames 0,1,2,3,

Receiver

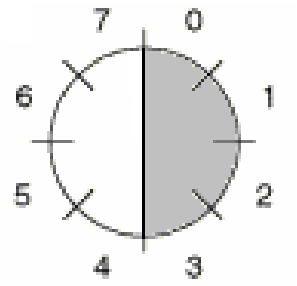


Receive ready for frames 0,1,2,3

Calculate the size of a sliding window

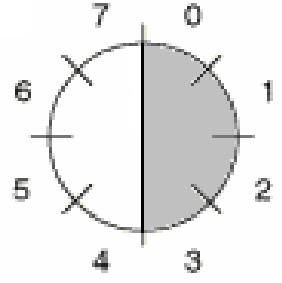
Given a sliding window using 3 bits for numbering frame → the size of windows will be 4

Sender



Sent and waiting acknowledgement for frames 0,1,2,3,

Receiver

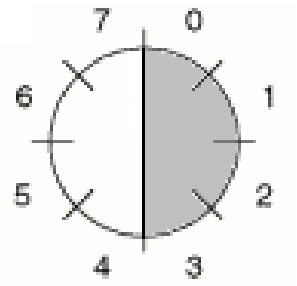


Received frames 0,1,2,3
Checking for error

Calculate the size of a sliding window

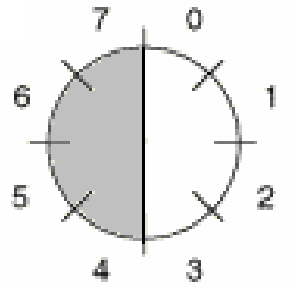
Given a sliding window using 3 bits for numbering frame → the size of windows will be 4

Sender



Sent and waiting acknowledgement for frames 0,1,2,3,

Receiver

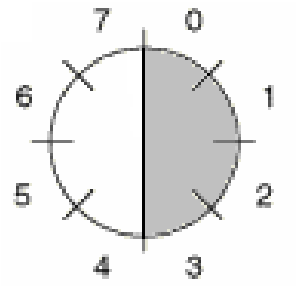


Frame 0,1,2,3 error-free
Sent acknowledgement for frames 0,1,2,3
Receive ready for frames 4,5,6,7

Calculate the size of a sliding window

Given a sliding window using 3 bits for numbering frame → the size of windows will be 4

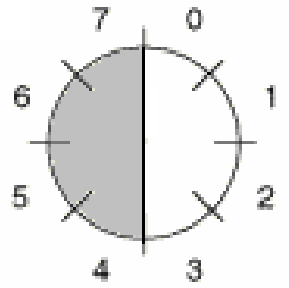
Sender



Transmission error
Acknowledgement frame doesn't reach sender

X

Receiver



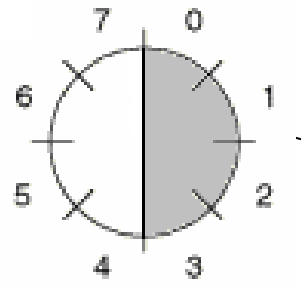
Sent and waiting acknowledgement for frames 0,1,2,3,

Frame 0,1,2,3 error-free
Sent acknowledgement for frames 0,1,2,3
Receive ready for frames 4,5,6,7

Calculate the size of a sliding window

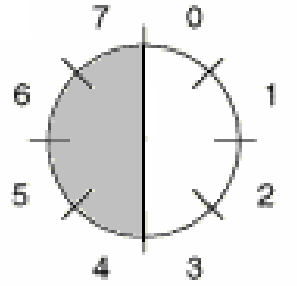
Given a sliding window using 3 bits for numbering frame → the size of windows will be 4

Sender



Time-out
Resend frame 0
Waiting acknowledgement for frames 0,1,2,3

Receiver

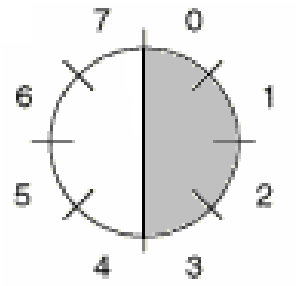


Receive ready for frames 4,5,6,7

Calculate the size of a sliding window

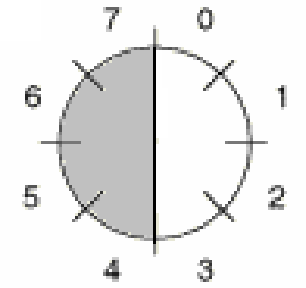
Given a sliding window using 3 bits for numbering frame → the size of windows will be 4

Sender



Time-out
Resend frame 0
Waiting acknowledgement for frames 0,1,2,3

Receiver



Frame 0 arrives, It isn't a waiting frame → discard it

Calculate the size of the buffer

- Size of buffer is just equal to the maximum size of receiving window.
- Example: If 3 bits are used to represent the sequence number of frames (from 0 to 7) then the maximum size of receiving window is (7-0+1)/2 =4 and the size of the buffer is also 4.

Time to send acknowledgement frames

- Piggy-back: Attach acknowledgement into data frame sent back by receiver
- In case the receiver has no data to send back to the sender
  - Create a timer for each received frame
  - If time-out event occurs and the receiver has no data to send back to the sender then the receiver will send an acknowledgement frame for the correspondent received frame

HDLC Protocol
(High Level Data Link Control)

HDLC Station Types

- Primary station
  - Control connection links
  - Send frames as commands
  - Maintain many logical links to secondary station
- Secondary station
  - Controlled by primary station
  - Send frames as responses to correspondent commands
- Combined station
  - Play both roles of Primary station and Secondary station
  - Can send commands or responses

HDLC Link Configurations

- Unbalanced configuration
  - One Primary station and one or many secondary stations
  - Support full duplex and half duplex
- Balanced configuration
  - Two combined stations
  - Support full duplex and half duplex

HDLC Transfer Modes

- Normal Response Mode (NRM)
- Asynchronous Balanced Mode (ABM)
- Asynchronous Response Mode (ARM)

HDLC Transfer Modes (1)

- Normal Response Mode (NRM)
  - Use unbalanced configuration
  - Primary station initializes a data transmission to a secondary station
  - A secondary station can only transfer data in form of the responses to reply the requests of primary station
  - Used with cables having many wires
  - Host plays role of a primary stations
  - Terminals play role of secondaries

HDLC Transfer Modes (2)

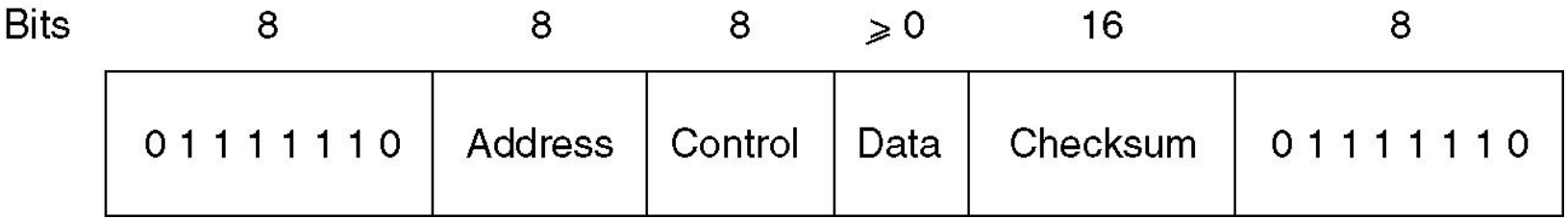
- Asynchronous Balanced Mode (ABM)
  - Use balanced configuration
  - All stations can initialize a data transmission without a permission from a primary station
  - Used popularly

HDLC Transfer Modes (3)

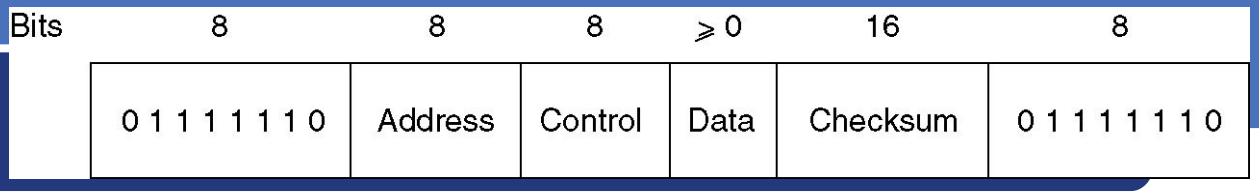
- Asynchronous Response Mode (ARM)
  - Use unbalanced configuration
  - Secondary can initialize a data transmission without a permission from a primary
  - Primary maintains connections
  - Used rarely

Frame structure

- Synchronous transmission
- All transmissions use frames
- One frame structure for both data and control



Frame structure



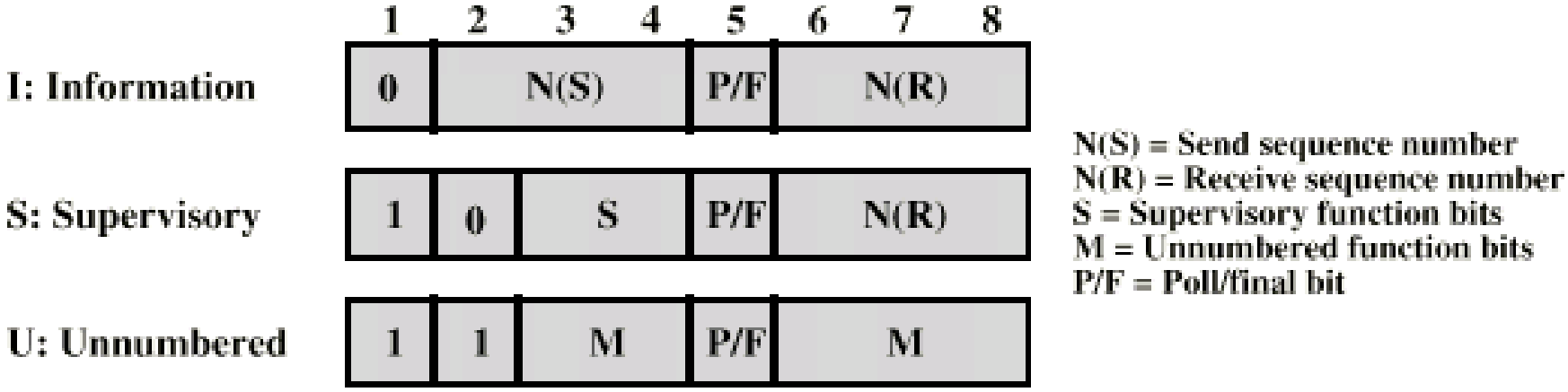
Flag (8 bit): 01111110 , uses bit stuffing technique

- **Address (8 bit):** Address of secondary station that is permitted to transfer or receive frames.
- **Control (8bit):** Type of frames:
  - Information frame
  - Supervisory frame
  - Unnumbered frame
- **Information(128-1024 bytes):** Data to transfer
- **FCS (Frame Check Sequence- 8 bit)**
  - CRC-CCITT = X16 + X12 + X5 +1

Control Field

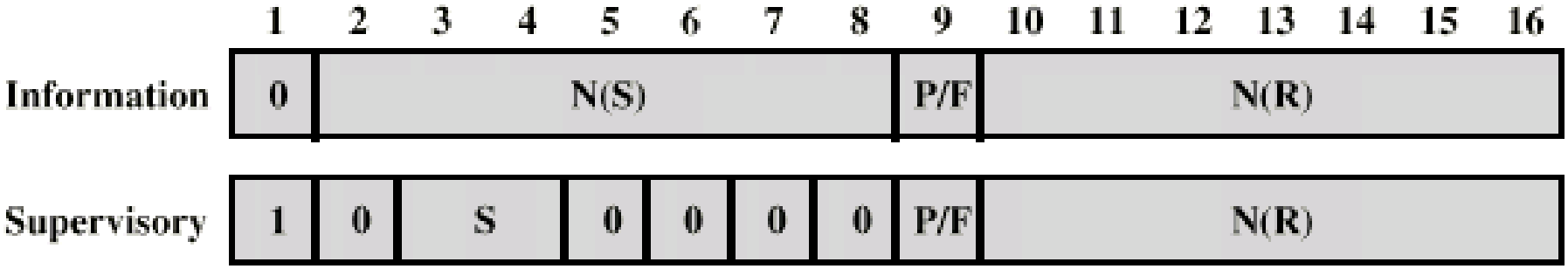
- Depending on frame type
  - Frame of Information :
    - contains data need to transfer
    - contains also acknowledgement (piggy-back)
  - Supervisory: Used as acknowledge frame when there is no data to send back to the sender
  - Unnumbered: Used to establish connections

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

Poll/Final Bit

- Used with different meaning depending on the context
- If the frame is a command
  - It means Poll
  - Value is set to 1 to demand the other side to send
- If the frame is response
  - Its means Final
  - Value is set to 1 to tell that it has no data to send any more

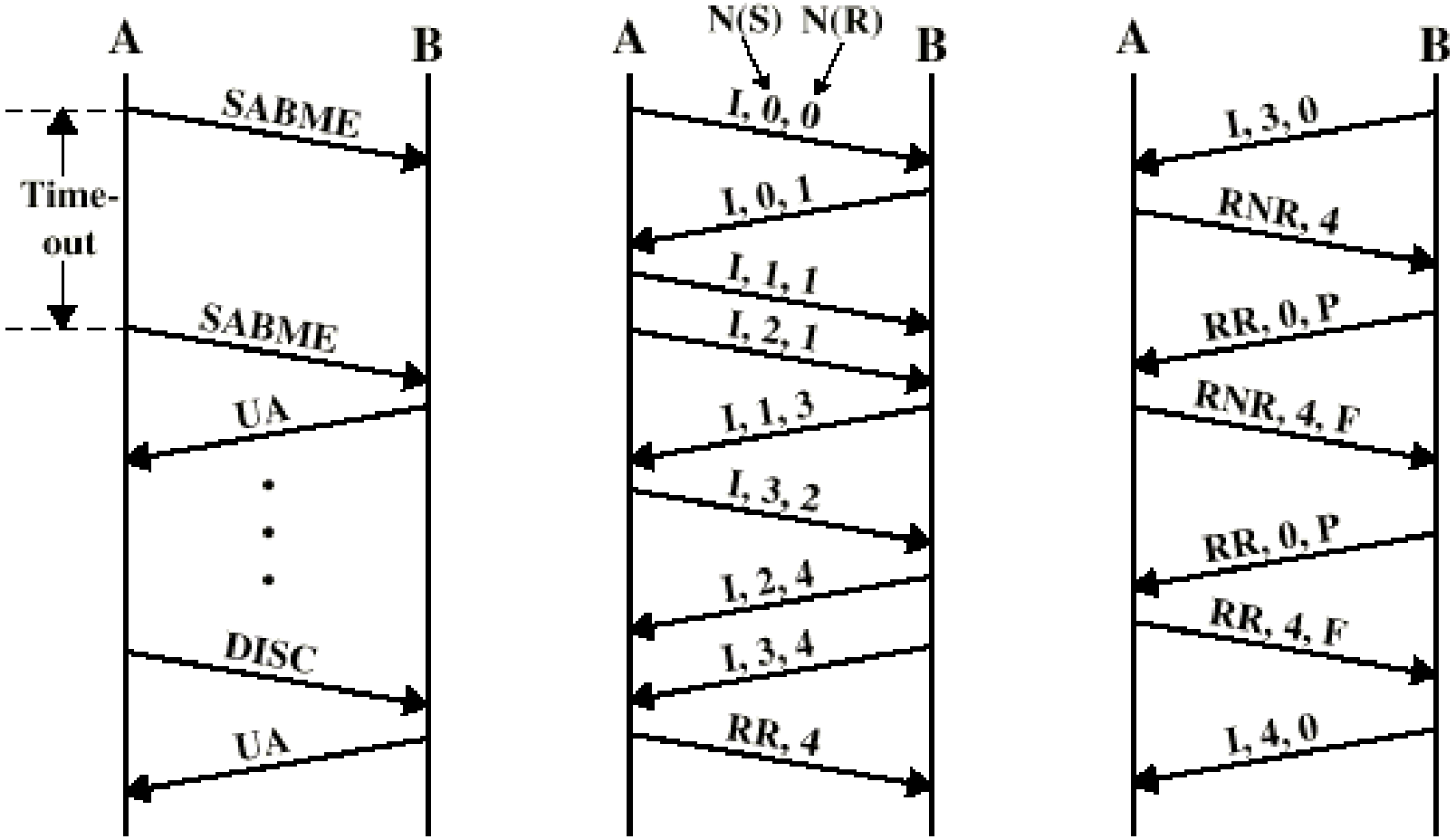
Supervisory function bits

SS=00	RR (Receive Ready), being an acknowledge frame, telling that it's ready to receive data, that it has received well until the frame Next-1 and are waiting for the frame numbered Next; used when there is no data to send back to the sender (figgyback)
SS=01	REJ (Reject): being a non acknowledgement frame, demanding the sender to resend the errors from Next frame
SS=10	RNR (Receive Not Ready): telling that it is not ready to receive frames, it has received frame Next-1, however, it hasn't been ready to receive frame Next
SS=11	SREJ (Selective Reject): demanding to resend the frame with the sequence number being Next

Unumbered Function Bits

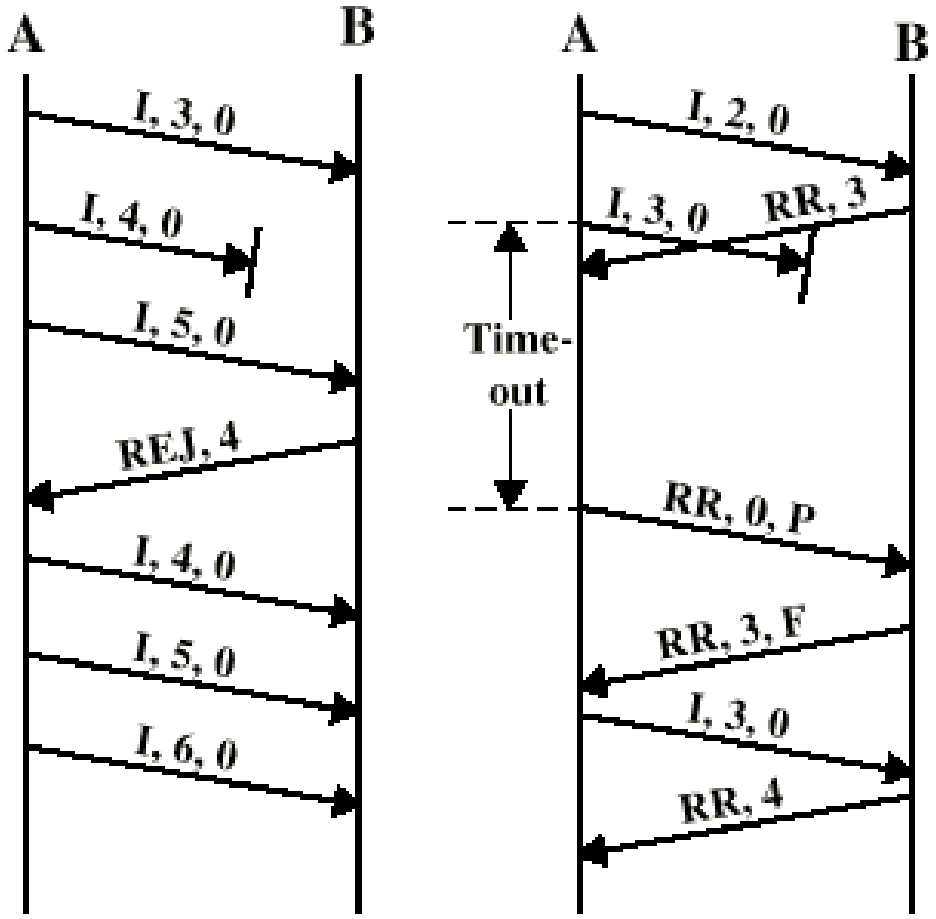
1111P100	This command requests to setup connection in Balanced mode (Set Asynchronous Balanced Mode).
1100P001	This command requests to setup connection in Normal response mode(Set Normal Response Mode).
1111P000	This command requests to setup connection in Asynchronous response mode (Set Asynchronous Response Mode).
1100P010	This command requests to release a connection DISC (Disconnect).
1100F110	UA (Unumbered Acknowledgment). This response is used by secondary stations to tell the primary station that they have already received and accepted all above U commands
1100F001	CMDR/FRMR (Command Reject/Frame Reject). This response is used by a secondary station to tell the primary station that it doesn't accept a well-received command

Scenarios



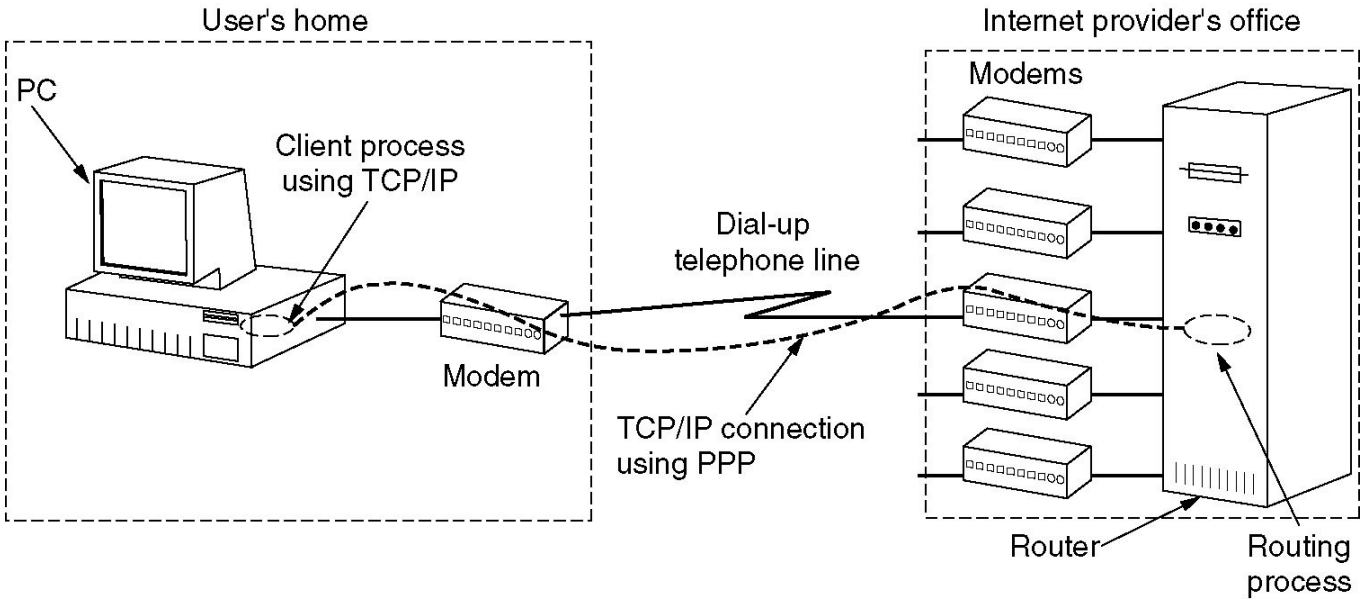
(a) Link setup and disconnect (b) Two-way data exchange (c) Busy condition

Scenarios



(d) Reject recovery (e) Timeout recovery

PPP- Point-to-Point Protocol)



- Used to transfer information between routers in a network or used to connect user's home machine to a network of a Internet Service Provider (ISP)
- LCP (Link Control Protocol).
- NCP (Network Control Protocol): choose network protocol used by network layer

Question and Answer please !