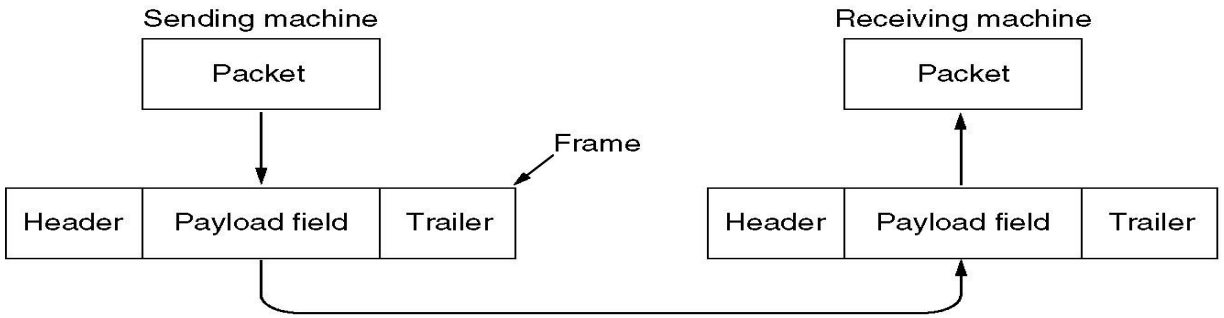
TẦNG LIÊN KẾT DỮ LIỆU (DATA LINK LAYER)

Trình bày: Bùi Minh Quân
bmquan@ctu.edu.vn
Khoa MMT&TT – Trường CNTT&TT - ĐHCT

Nội dung

- Các chức năng cơ bản của tầng Data Link
- Các phương pháp phát hiện lỗi của tầng Data Link.
- Các giao thức điều khiển lỗi trong tầng Data Link
- Giao thức HDLC (High-Level Data Link Control)
- Giao thức PPP (Point to Point Protocol).

Chức năng của tầng liên kết dữ liệu



- Định khung **(frame)** dữ liệu: đóng gói các gói tin **(Packet)** nhận được từ tầng mạng vào các khung để truyền đi.
- Truyền tải dữ liệu nhận được từ tầng mạng trên máy gởi đến tầng mạng trên máy nhận
- Cung cấp các dịch vụ cho tầng mạng
- Xử lý lỗi trong quá trình truyền dữ liệu

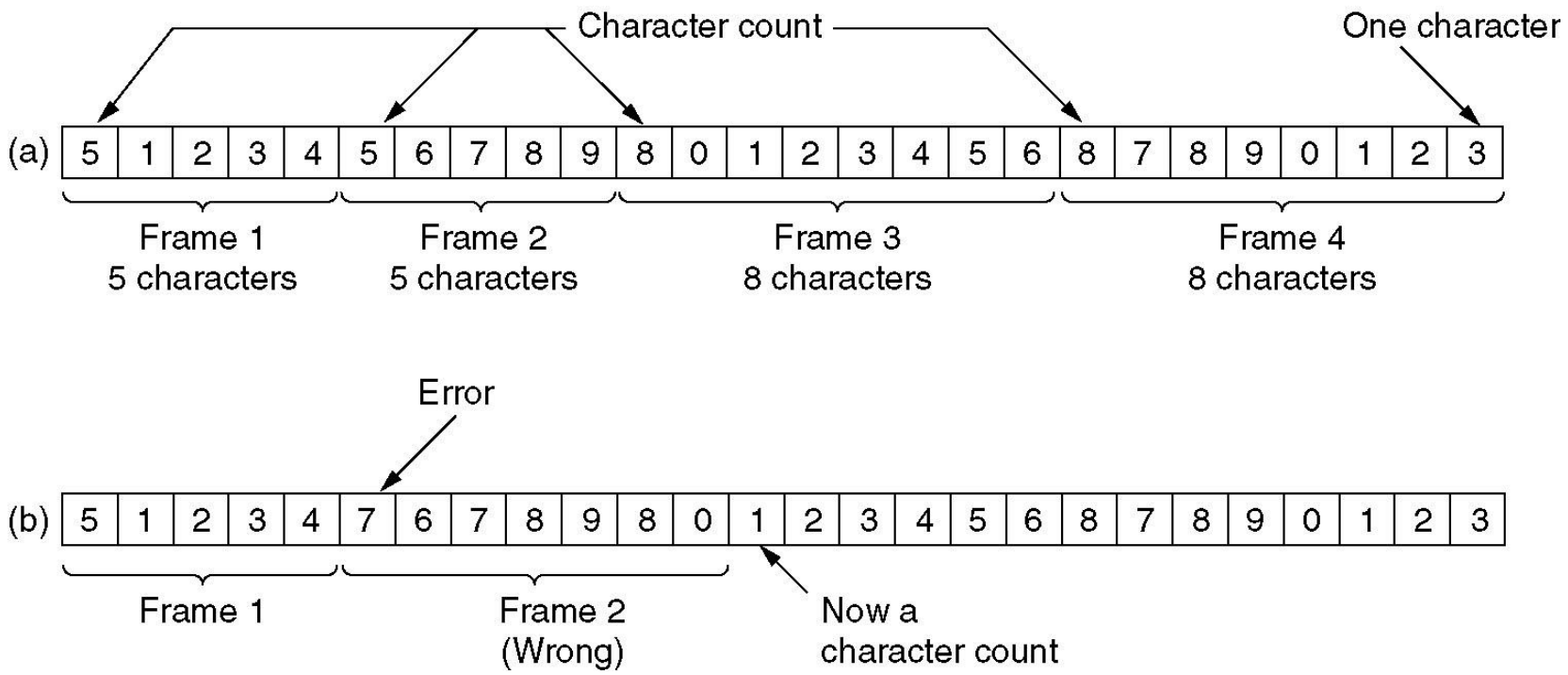
Chức năng của tầng liên kết dữ liệu
Các dịch vụ cơ bản

- Dịch vụ không nối kết không báo nhận (unacknowledged connectionless service), thường được sử dụng trong mạng LAN.
- Dịch vụ không nối kết có báo nhận (acknowledged connectionless service), thường dùng cho mạng không dây.
- Dịch vụ nối kết định hướng có báo nhận (acknowledged connection-oriented service), thường dùng trong mạng WANs.

Chức năng của tầng liên kết dữ liệu - định khung

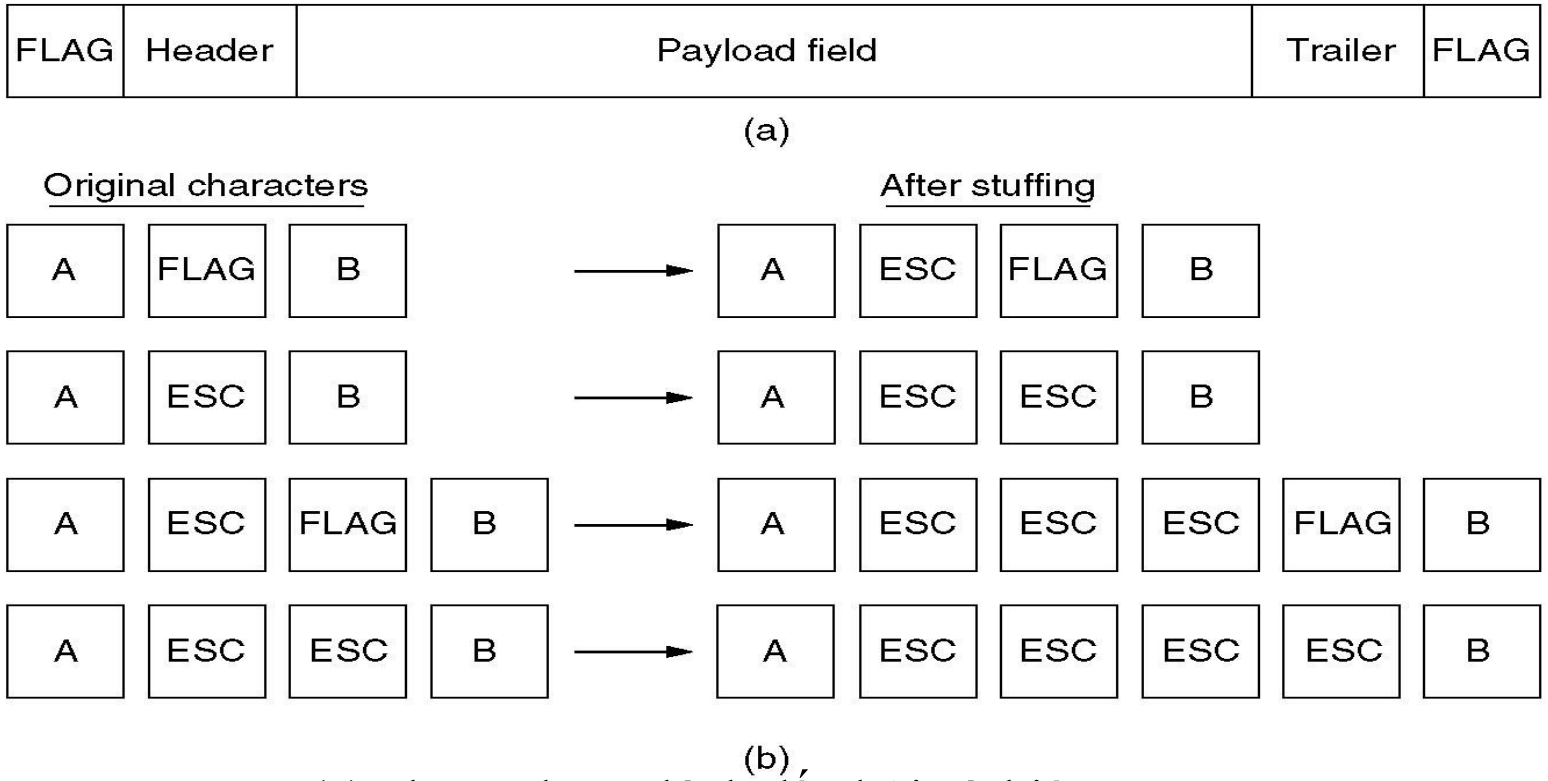
- Qui định khuôn dạng của khung được sử dụng ở tầng liên kết dữ liệu
- **3 phương pháp định khung phổ biến:**
  - Đếm ký tự (Charater count)
  - Sử dụng các **bytes** làm cờ hiệu và các bytes độn (Flag byte with byte stuffing)
  - Sử dụng **cờ** bắt đầu và kết thúc khung cùng với các **bit độn** (Starting and ending flags with bit stuffing)

Phương pháp đếm ký tự (Character Count)



Nếu một khung nào đó bị sai -> không xác định được các khung tiếp theo sau

Phương pháp sử dụng byte làm cờ và các byte độn (Flag byte with byte stuffing)

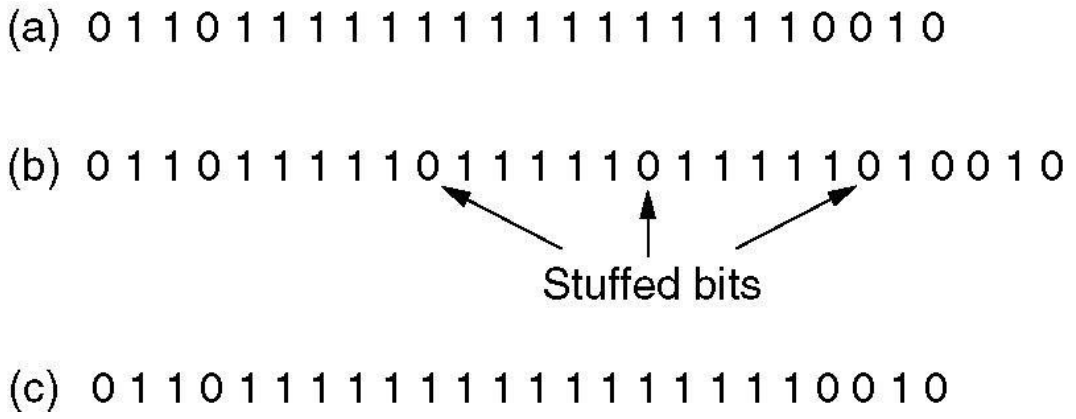


(a)Khung được đánh dấu bởi cờ hiệu,

(b) Dữ liệu có chứa cờ hiệu và byte ESC.

Phương pháp sử dụng cờ bắt đầu và kết thúc khung cùng với các bit độn (Starting and ending flags with bit stuffing)

- Sử dụng mẫu bit đặc biệt, 01111110, để làm cờ đánh dấu điểm bắt đầu và kết thúc khung



(a) Dữ liệu gốc,

(b) Dữ liệu chuyển lên đường truyền,

(c) Dữ liệu nhận sau khi loại bỏ các bit độn.

Chức năng của tầng liên kết dữ liệu
Điều khiển lỗi (Error Control)

- Cách nào để đảm bảo rằng toàn bộ các khung đã được phân phát **đến** tầng mạng và được phân phát theo **đúng** trình tự chúng đã được gởi ?
  - Người nhân báo về tình trạng nhận khung:
    - Sử dụng Khung báo nhận (acknowledgement)
  - Tránh chờ vĩnh viễn:
    - Sử dụng bộ đếm thời gian (timer) + time-out
  - Trùng lắp gói tin nhận:
    - Gán số thứ tự cho khung

Chức năng của tầng liên kết dữ liệu
Điều khiển luồng (Flow Control)

- Giải quyết sự khác biệt về tốc độ truyền / nhận dữ liệu của bên truyền và bên nhận
- Hai tiếp cận:
  - Tiếp cận điều khiển luồng dựa trên phản hồi (feedback based flow control): người nhận gởi thông tin về cho người gởi cho phép người gởi gởi thêm dữ liệu, cũng như báo với người gởi những gì mà người nhận đang làm.
  - Tiếp cận điều khiển luồng dựa trên tần số (rate based flow control): trong giao thức truyền tin cài sẵn cơ chế giới hạn tần suất mà người gởi có thể truyền tin

VẤN ĐỀ XỬ LÝ LỖI

Vấn đề xử lý lỗi

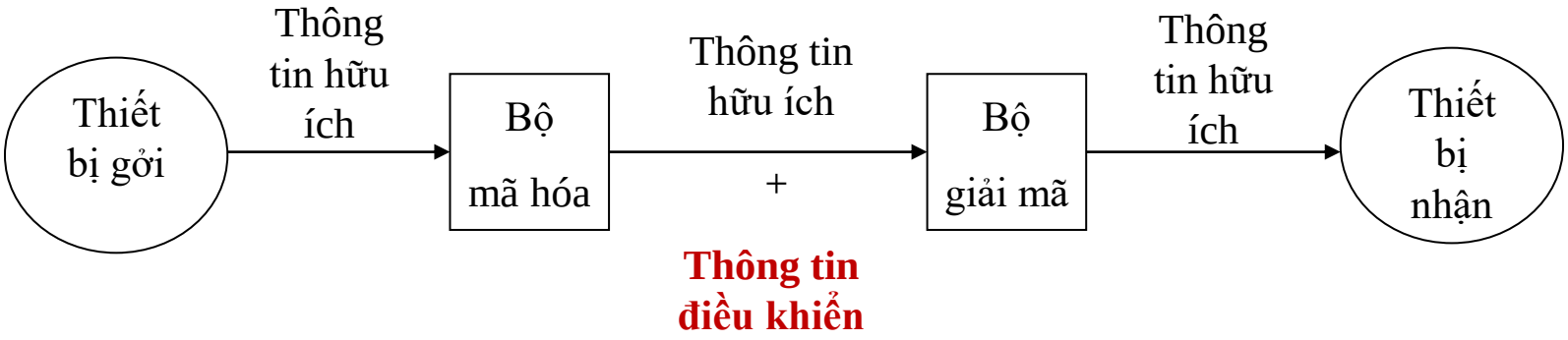
- Bộ mã phát hiện lỗi là gì ?
- Những bộ mã phát hiện lỗi
  - Kiểm tra chẵn lẻ (Parity checks)
  - Kiểm tra thêm theo chiều dọc (Longitudinal reduncy check - LRC)
  - Kiểm tra phần dư tuần hoàn (Cyclic redundancy check - CRC)

Lỗi trên đường truyền

- Bit 1 thành bit 0 và ngược lại
- Tỷ lệ lỗi
  - τ = Số bít bị lỗi / Tổng số bít được truyền
  - τ : 10^{-5} đến 10^{-8}
  - 88% : sai lệch một bit
  - 10% : sai lệch 2 bit kề nhau

Bộ mã phát hiện lỗi

- Bên cạnh các thông tin hữu ích cần truyền đi, ta thêm vào các thông tin điều khiển. Bên nhận thực hiện việc giải mã các thông tin điều khiển này để phân tích xem thông tin nhận được là chính xác hay có lỗi.



Bộ mã phát hiện lỗi

- Bộ mã sửa lỗi (Error-correcting code):
  - Cho phép bên nhận có thể tính toán và suy ra được các thông tin bị lỗi (sửa dữ liệu bị lỗi)
- Bộ mã phát hiện lỗi (Error-detecting code):
  - Cho phép bên nhận phát hiện ra dữ liệu có lỗi hay không
  - Nếu có lỗi bên nhận sẽ yêu cầu bên gởi gởi lại thông tin
- Các hệ thống mạng ngày nay có xu hướng chọn bộ mã phát hiện lỗi.

Phương pháp Kiểm tra chẵn lẽ (Parity Check)

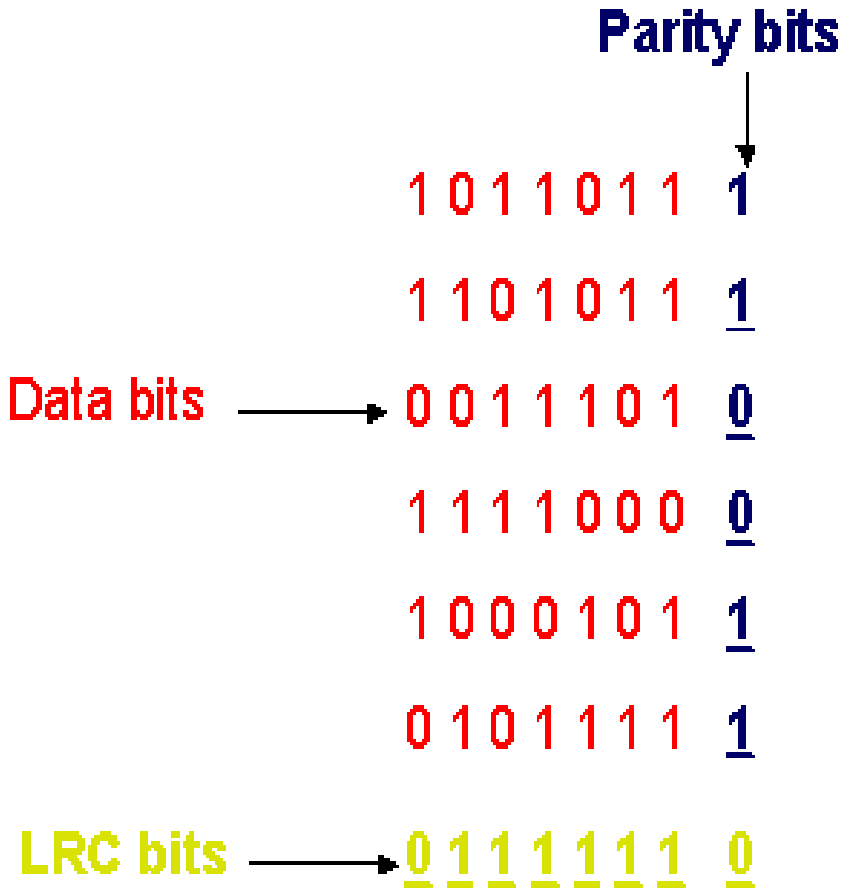
- xxxxxxx: chuỗi bits dữ liệu cần truyền
- Thêm vào **1 bit** chẵn-lẽ p
- Chuỗi bit truyền là: xxxxxxxp
- **p** được tính để đảm bảo:
  - Phương pháp kiểm tra chẵn: xxxxxxx**p** có một số chẵn các bit 1
  - Phương pháp kiểm tra lẽ: xxxxxxx**p** có một số lẽ các bit 1
- Bên nhận nhận được chuỗi xxxxxxx**p**:
  - Phương pháp kiểm tra chẵn:
    - Nếu có 1 số chẵn các bit 1: Dữ liệu xxxxxxx không có lỗi
    - Ngược lại là có lỗi
  - Phương pháp kiểm tra lẽ:
    - Nếu có 1 số lẽ các bit 1: Dữ liệu xxxxxxx không có lỗi
    - Ngược lại là có lỗi

Phương pháp Kiểm tra chẵn lẽ (Parity Check)

- Ví dụ: cần truyền ký tự G = 1110001
- Sử dụng phương pháp kiểm tra chẵn:
  - p=0
  - Chuỗi truyền đi là: 1110001**0**
- Bên nhận nhận được chuỗi:
  - 11100010: 4 bit 1=> không có lỗi
  - 11000010: 3 bit 1=> dữ liệu có lỗi
  - 11000110: 4 bit 1=> không có lỗi ???

Kiểm tra thêm theo chiều dọc (Longitudinal Redundancy Check or Checksum)

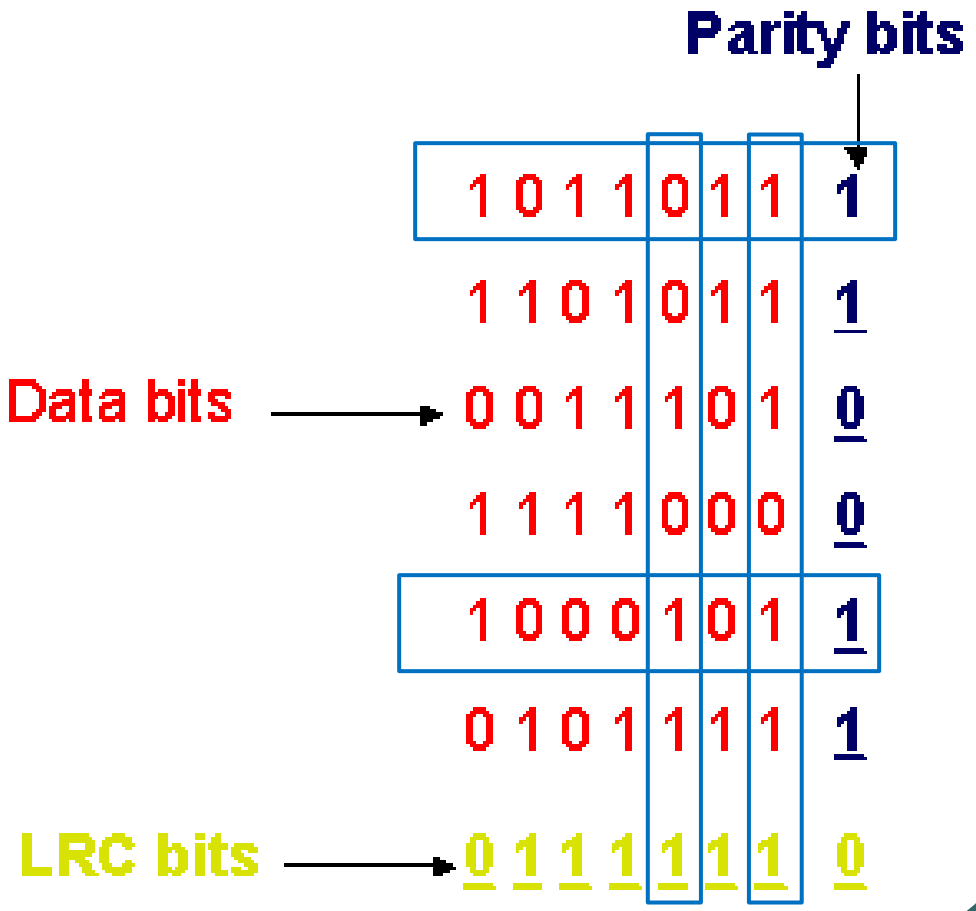
- Để cải tiến phương pháp kiểm tra chẵn lẻ, phương pháp LRC xem các khung như một khối nhiều ký tự được xắp xếp theo dạng 02 chiều
- Việc kiểm tra chẵn lẻ sẽ được thực hiện cả theo chiều ngang và chiều dọc



Kiểm tra thêm theo chiều dọc (Longitudinal Redundancy Check or Checksum)

- Phương pháp này giảm tỷ lệ lỗi từ 2 đến 4 lần so với phương pháp chẵn lẻ.

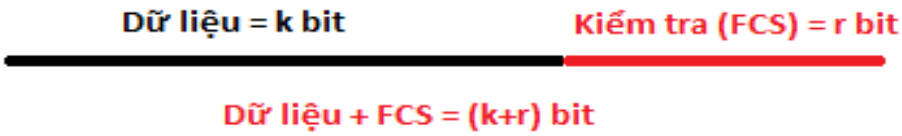
- **Ví dụ:** bit 1 và 3 của **ký tự 1** bị lỗi hệ thống phát hiện được, tuy nhiên nếu có thêm bit 1 và 3 của **ký tự 5** bị lỗi thì hệ thống không phát hiện được



Kiểm tra phần dư tuần hoàn (Cyclic Redundancy Check)

- Một số phương pháp cài đặt khác nhau như:
  - Modulo 2,
  - Đa thức,
  - Thanh ghi dịch
  - Các cổng Exclusive-or

Kiểm tra phần dư tuần hoàn Modulo 2



- Giả sử ta có:
  - M: Thông điệp k bit cần gởi sang bên nhận.
  - F : Chuỗi kiểm tra khung FCS gồm **r** bit là thông tin điều khiển được gởi theo M để giúp bên nhận có thể phát hiện được lỗi.
  - T =MF là khung **(k + r)** bit, được hình thành bằng cách nối M và F lại với nhau. T sẽ được truyền sang bên nhận, với r < k
- Với M (k bit) , **P** (r+1 bit), F (r bit), T (k+r bit), thủ tục tiến hành để xác định checksum F và tạo khung truyền như sau:
  - Nối **r** bit 0 vào cuối M, hay thực hiện phép nhân M với 2^r
  - Dùng phép chia modulo 2 chia chuỗi bit **M*2^r** cho **P**.
  - **Phần dư** của phép chia sẽ được **cộng** với M*2^r tạo thành khung **T** truyền đi.
  - Trong đó **P** được chọn dài hơn **F** một bit, và cả hai bit cao nhất và thấp nhất phải là 1
- Bên nhận thực hiện phép chia T cho P:
  - Chia hết: T không có lỗi, dữ liệu M từ T – **k bits trọng số cao**
  - Chia không hết: T có lỗi

Một số tính chất của phép toán Mod-2

- Phép cộng Mod-2 là phép cộng nhị phân không nhớ, dưới đây là thí dụ về phép cộng và phép nhân:

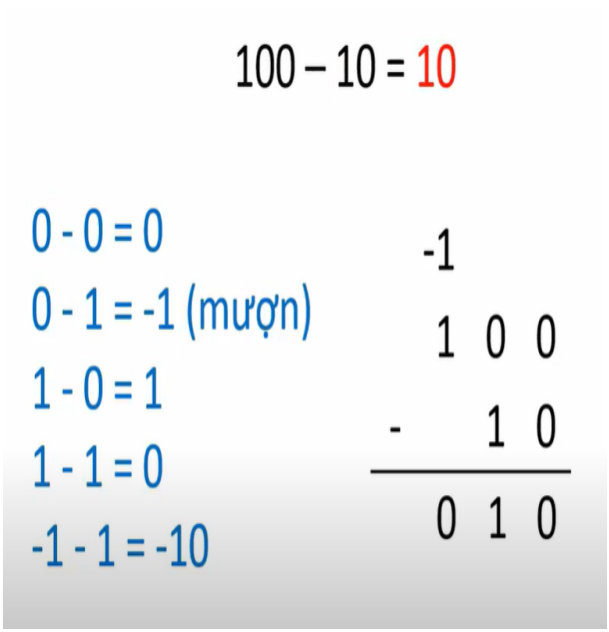
```
  1 1 1 1
+ 1 0 1 0
  0 1 0 1
```

```
  11001
x 11
  11001
 11001
 101011
```

Một số tính chất của phép toán Mod-2

- Phép trừ Mod-2 giống như phép cộng (xem lại)
- Nhân Mod-2 một số với 2^n tương ứng với dời số đó n bit về bên trái và thêm n bit 0 vào bên phải số đó,

  Thí dụ: $11001 * 2^3 = 11001000$
- Phép chia Mod-2 được thực hiện giống như phép chia thường nhưng **nhớ là phép trừ trong khi chia được thực hiện như phép cộng**.



Kiểm tra phần dư tuần hoàn Modulo 2

- Giả sử ta có:
  - M = 1010001101 (10 bit)
  - P = 110101 (6 bit)
  - FCS cần phải tính toán (5 bit)
- Lần lượt thực hiện các bước sau:
  - Tính $M*2^5$ = 1010001101**00000**.
  - Thực hiện phép chia modulo $M*2^5$ cho P ta được phần dư F = **01110**
  - Tạo khung gởi đi là $T = M*2^r + F$ = 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*2r)
                  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

Bài 1: Xác định T= ???

P: 10011

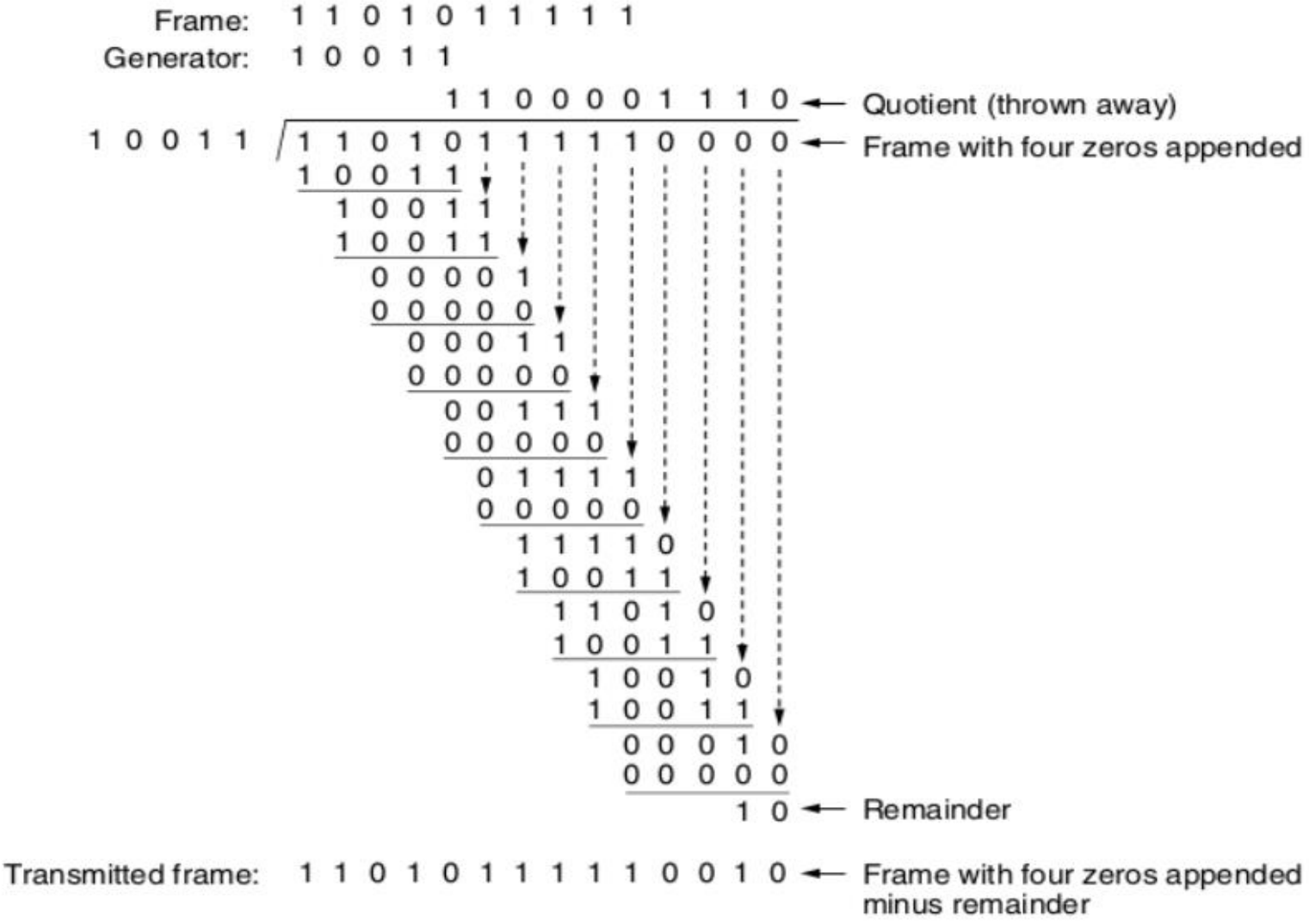
M: 110 101 1111

Bài 2: kiểm tra T nhận có lỗi không?

Với P: 10011

T1: 110 101 1111 0010

T2: 110 101 1111 0110


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

Kiểm tra phần dư tuần hoàn Phương pháp đa thức

- Giả sử ta có M=110011và P = **11001**, khi đó M và P được biểu diễn lại bằng 2 đa thức sau:
  - M(x) = x5 + x4 + x + 1
  - P(x) = **x4 + x3 + 1**
- Quá trình tính CRC được mô tả dưới dạng các biểu thức sau:

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

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

Kiểm tra phần dư tuần hoàn
Phương pháp đa thức

Các version thường được sử dụng của P là :

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

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

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

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

Kiểm tra phần dư tuần hoàn
Phương pháp đa thức

Ví dụ:

- Cho: M=1010001101, P=110101
- Ta có: r=5

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

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

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

- Thực hiện phép toán:

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

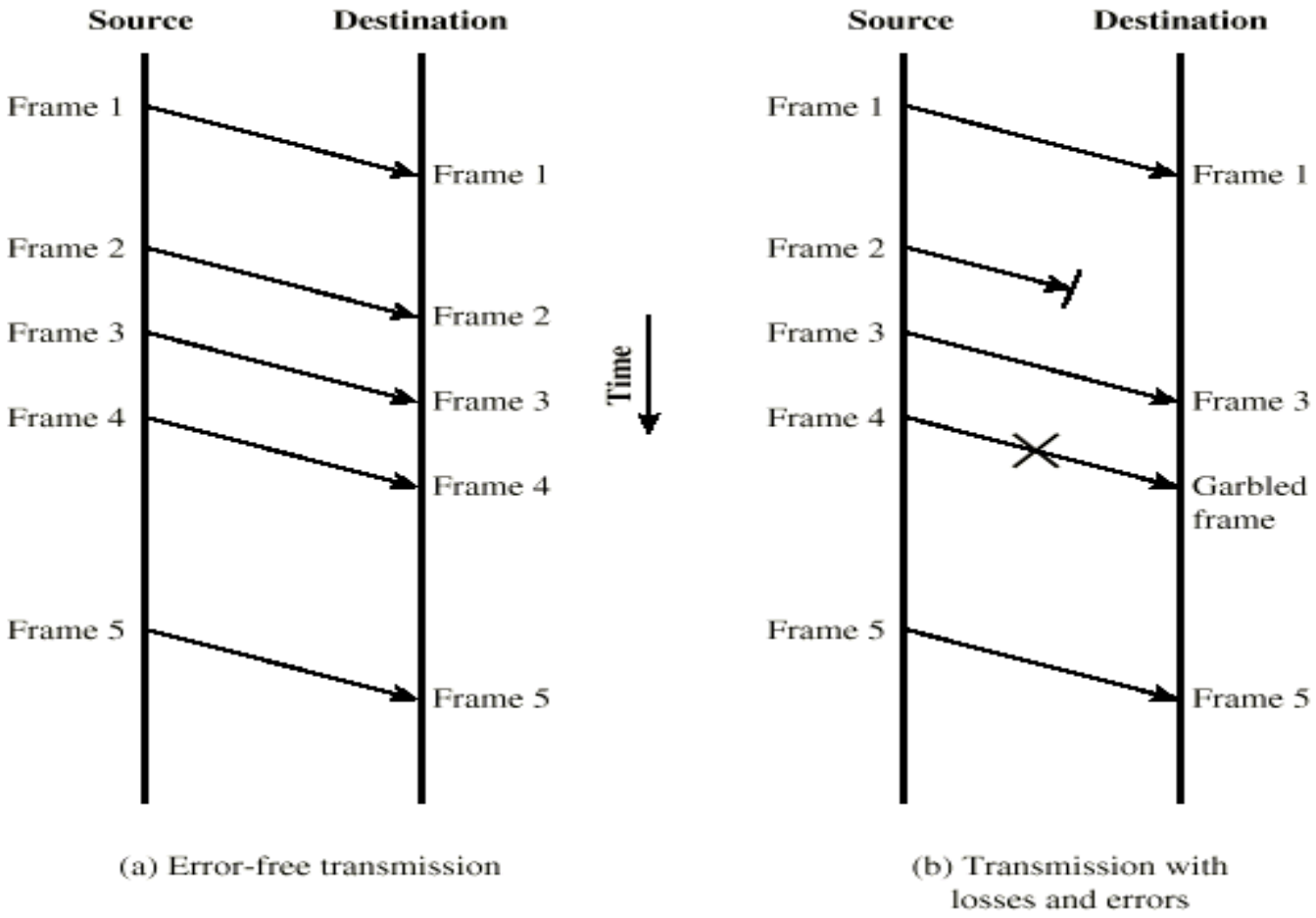
$$\Rightarrow \quad Q(x)= x^9+ x^8+ x^6+x^4+x^2 +x^1$$

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

⇨ Khung cần truyền đi là T= 1010001101**01110**

ĐIỀU KHIỂN LỖI
(Error Control)

Điều khiển lỗi



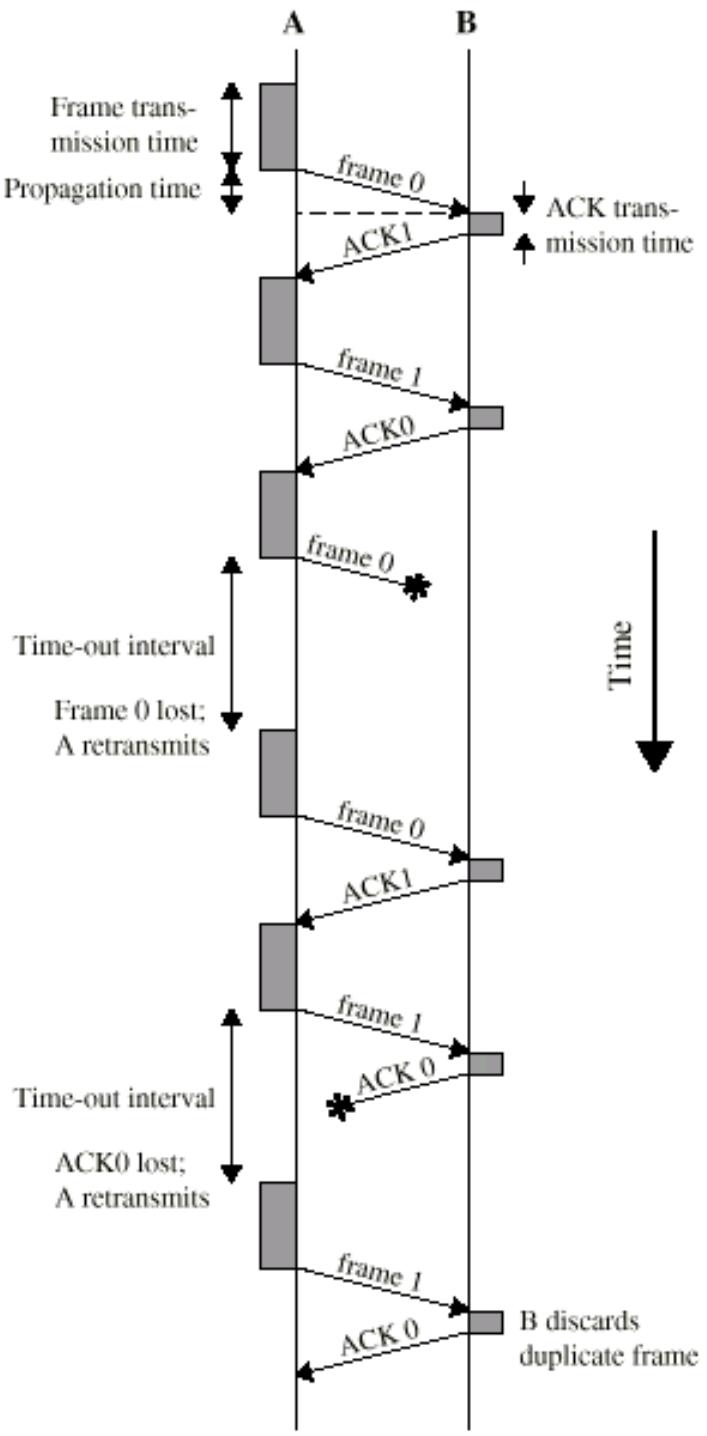
(a) Error-free transmission

(b) Transmission with losses and errors

Stop and Wait - Diagram

- Người gởi không biết được khung có đến nơi nhận tốt hay không.
  - Giải pháp: Khung báo nhận.
- Các khung báo nhận có thể bị mất.
  - Giải pháp:
    - Timer.
    - Time-out
    - Gởi lại
- Bên nhận không phân biệt được các khung trùng lắp do bên gởi gởi lại.
  - Giải pháp: Mỗi khung sẽ có một số thứ tự

Stop and Wait: truyền đơn công (Simplex)



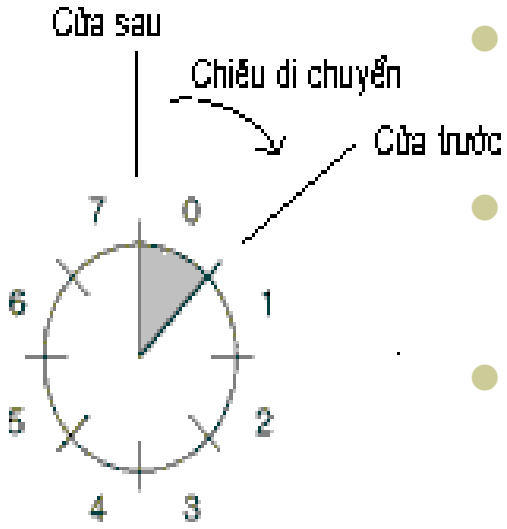
Vấn đề truyền tải thông tin theo hai chiều (Duplex)

- Mong muốn việc truyền tải thông tin theo chế độ song công (Duplex) để khai thác tối đa năng lực kênh truyền. Nguyên tắc thực hiện như sau:
  - Vẫn thực hiện việc truyền tải khung,
  - Phân loại khung: DATA, ACK, NACK
  - Sử dụng kỹ thuật **piggyback**.

Giao thức của sổ trượt (Sliding windows)

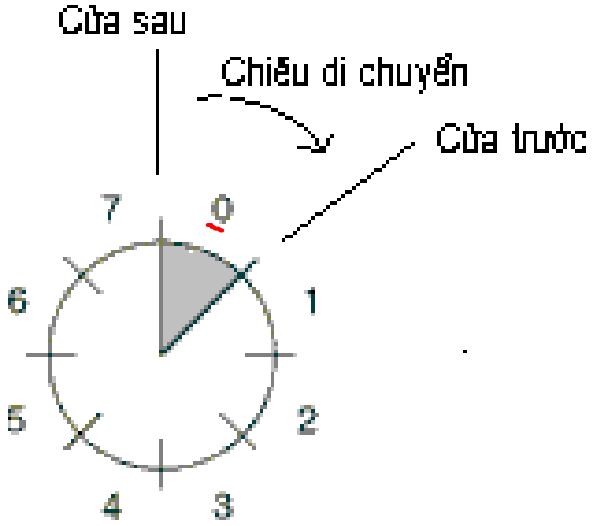
- Thay vì chỉ truyền đi một khung tại một thời điểm (simplex), giao thức cửa sổ trượt cho phép bên gởi có thể gởi đi nhiều khung.
- Cửa sổ gởi (Sending Windows): Bên gởi theo dõi các khung mà nó được phép gởi đi và các khung mà nó đang chờ báo nhận
- Cửa sổ nhận (Receiving Windows): Bên nhận theo dõi các khung mà nó được phép nhận

Cửa sổ trượt (Sliding Windows)



- Cửa sổ gồm có **cửa trước** và **cửa sau** cùng di chuyển theo một chiều.
- Kích thước của cửa sổ là chiều của cung giới hạn từ cửa sau đến cửa trước (***phần tô đen***).
- Kích thước của cửa sổ có thể thay đổi:
  - Khi cửa trước di chuyển, cửa sổ được mở rộng ra.
  - Khi cửa sau di chuyển, kích thước của cửa sổ bị thu hẹp lại và nó làm cho cửa sổ thay đổi vị trí, trượt / quay quanh một tâm.

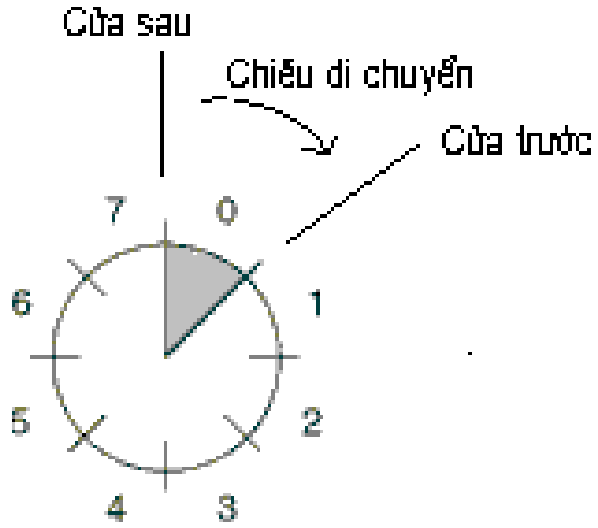
Cửa sổ trượt (Sliding Windows)



- Kích thước nhỏ nhất của cửa số là 0
- Kích thước tối đa của cửa sổ là n-1
- k bit để đánh số thứ tự khung [0- (2^k-1)] => Khi đó cửa sổ trượt sẽ được chia thành 2^k vị trí tương ứng với 2^k khung.

- **Đối với cửa sổ gởi**, các vị trí nằm trong cửa sổ trượt biểu hiện số thứ tự của các khung mà bên gởi đang chờ bên nhận báo nhận. Phần bên ngoài cửa sổ là các khung có thể gởi tiếp. Tuy nhiên phải đảm bảo rằng, cửa sổ gởi không được vượt quá kích thước tối đa của cửa sổ.

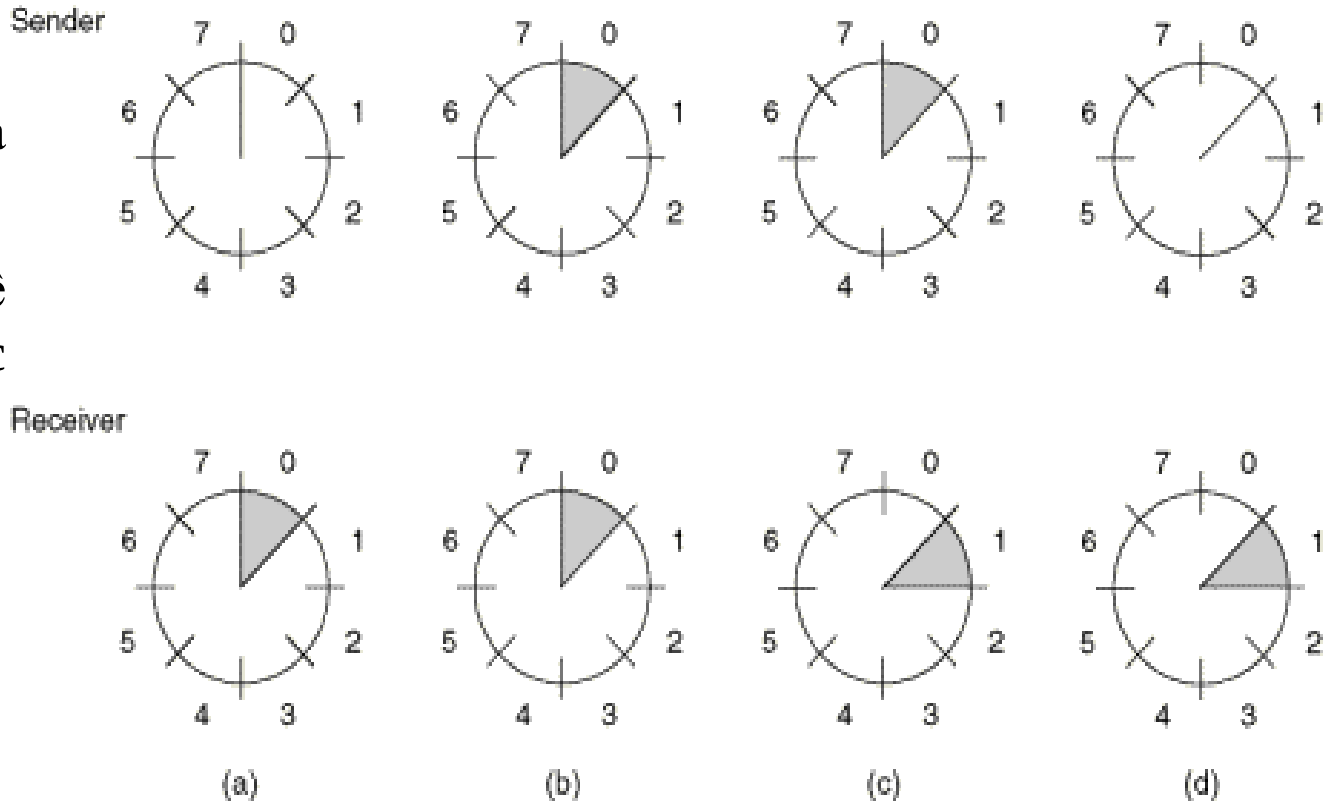
Cửa sổ trượt (Sliding Windows)



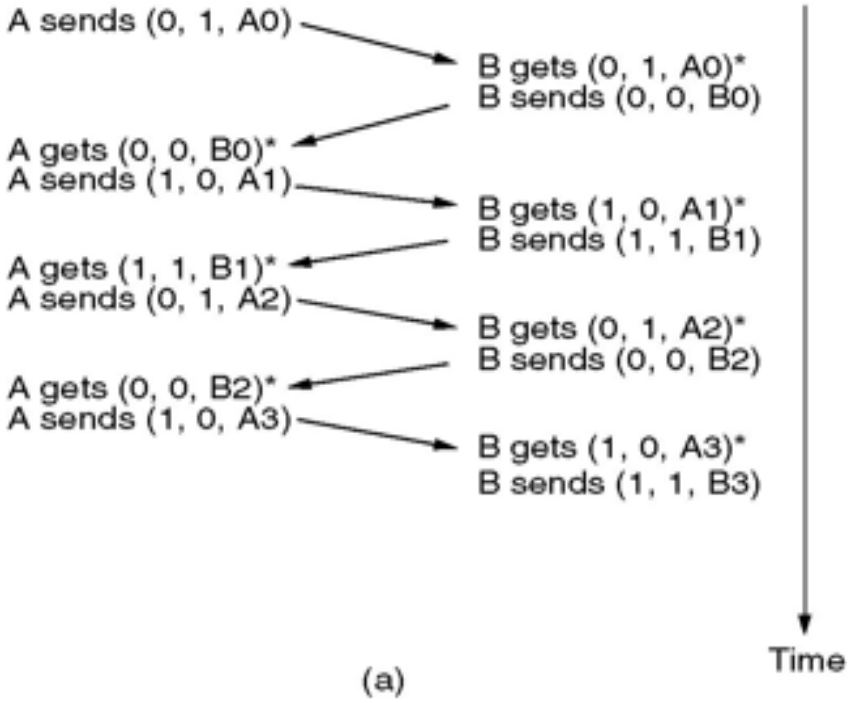
- **Đối với bên nhận**, các vị trí nằm trong cửa sổ biểu hiện số thứ tự các khung mà nó đang sẵn sàng chờ nhận.
- Kích thước tối đa của cửa sổ biểu thị dung lượng bộ nhớ đệm của bên nhận có thể lưu tạm thời các gói tin nhận được trước khi xử lý chúng.
- **Giả sử bên nhận** có một vùng bộ nhớ đệm có khả năng lưu trữ 4 khung nhận được. Khi đó, kích thước tối đa của cửa sổ sẽ là 4

Hoạt động cửa sổ trượt

- Kích thước của của số là 1
- Sử dụng 3 bit để đánh số thứ tự các khung



Ví dụ giao thức cửa sổ trượt với kích thước là 1



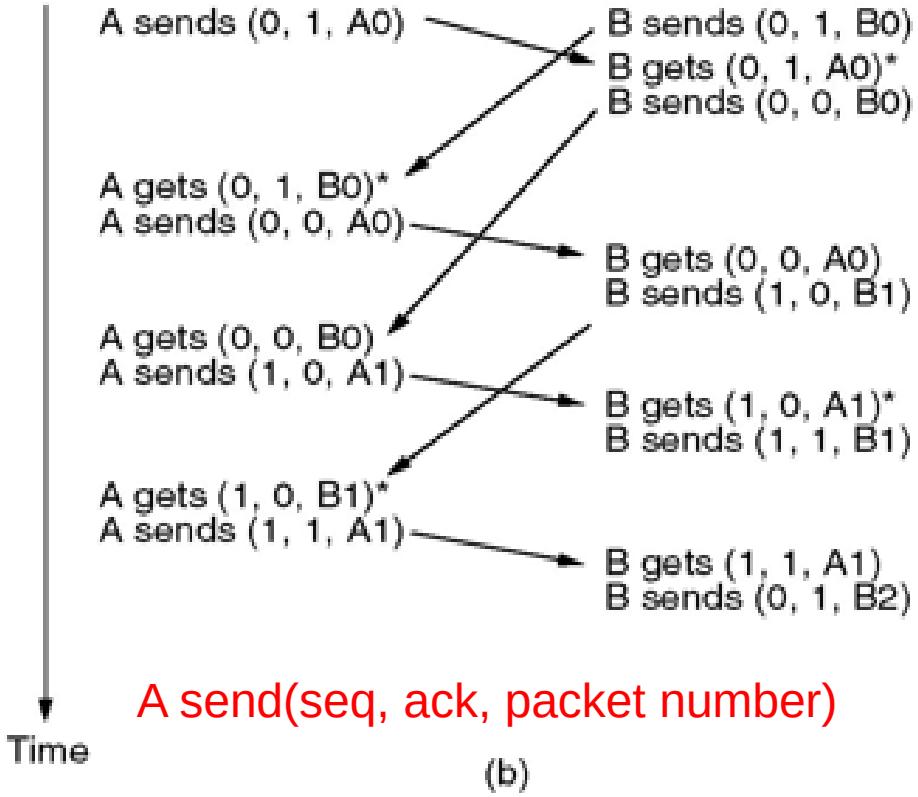
A send(seq, ack, packet number)

(a): Việc gởi nhận diễn ra bình thường theo đúng tuần tự

https://humphryscomputing.com/Notes/Networks/data.sliding.html

Ví dụ giao thức cửa sổ trượt với kích thước là 1

In (b) half the frames contain duplicates (nothing to send to Network layer), even though there are no transmission errors.



A send(seq, ack, packet number)

Figure (b) above:

Problem if both send initial packet at same time.

A sends 0,1

B sends 0,1
B gets 0,1 (got 0, but no ack of my 0)
B **unnecessarily re-sends** 0,0

A gets 0,1 (got 0, but no ack of my 0)
A **unnecessarily re-sends** 0,0

B gets 0,0 (already have 0, **nothing to pass to NB**, but this is good ack of my 0)
B sends 1,0

A gets 0,0 (already have 0, **nothing to pass to NA**, but this is good ack of my 0)
A sends 1,0

B gets 1,0 (got 1, but no ack of my 1)
B **unnecessarily re-sends** 1,1

A gets 1,0 (got 1, but no ack of my 1)
A **unnecessarily re-sends** 1,1

B gets 1,1 (already have 1, **nothing to pass to NB**, but this is good ack of my 1)
B sends 0,1

....

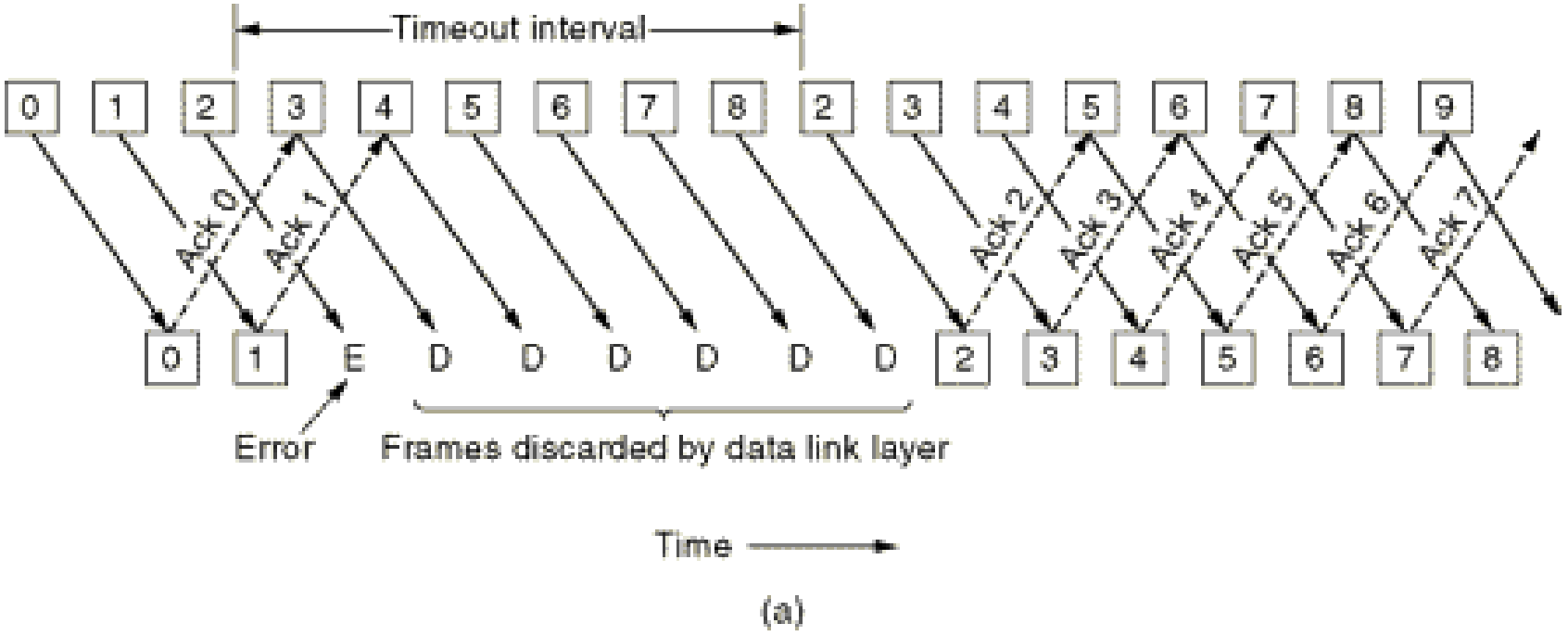
(b): Việc gởi nhận diễn ra theo một trình tự bất kỳ

https://humphryscomputing.com/Notes/Networks/data.sliding.html

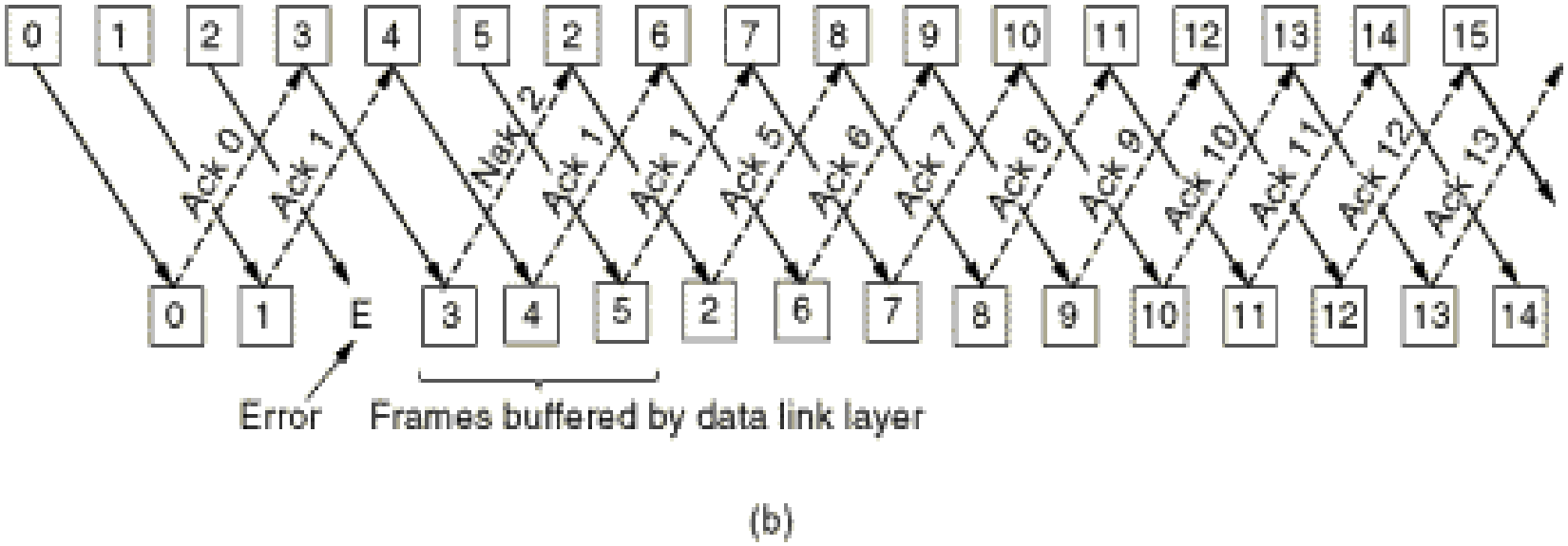
Giao thức Go-Back-N

- Khi một khung bị lỗi. Bên nhận bỏ qua khung.
- Vì không một báo nhận cho khung lỗi, khi hết thời gian chờ, bên gởi phải gởi lại khung bị lỗi và toàn bộ các khung phía sau nó.

Giao thức Go-Back-N



Giao thức Selective Repeat

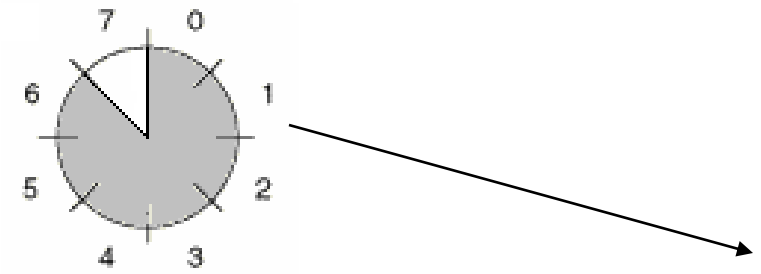


(b)

Xác định kích thước cửa sổ trượt

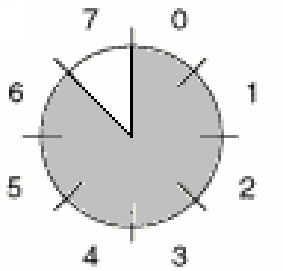
- Xét cửa sổ trượt sử dụng 3 bits để đánh chỉ số khung => Kích thước cửa sổ là 7

Sender



Đã gởi và chờ bào nhận các khung 0,1,2,3,4,5,6

Receiver

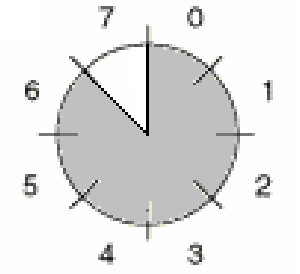


Đang sẵn sàng chờ nhận các khung 0,1,2,3,4,5,6

Xác định kích thước cửa sổ trượt

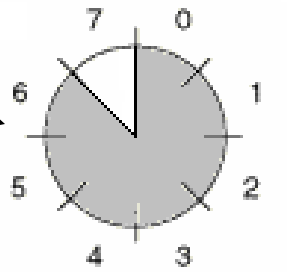
- Xét cửa sổ trượt sử dụng 3 bits để đánh chỉ số khung => Kích thước cửa sổ là 7

Sender



Đã gởi và chờ báo nhận các khung 0,1,2,3,4,5,6

Receiver

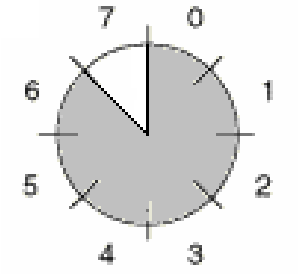


Nhận các 0,1,2,3,4,5,6, Kiểm tra lỗi

Xác định kích thước cửa sổ trượt

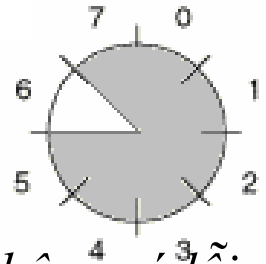
- Xét cửa sổ trượt sử dụng 3 bits để đánh chỉ số khung => Kích thước cửa sổ là 7

Sender



Đã gởi và chờ bào nhận các khung 0,1,2,3,4,5,6

Receiver

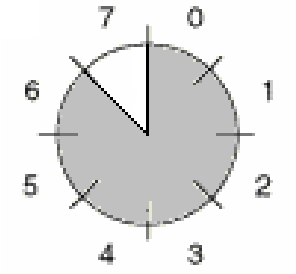


Khung 0,1,2,3,4,5,6 không có lỗi,
Gởi báo nhận cho các khung 0,1,2,3,4,5,6
Sẵn sàng chờ nhận các khung 7,0,1,2,3,4,5

Xác định kích thước cửa sổ trượt

- Xét cửa sổ trượt sử dụng 3 bits để đánh chỉ số khung => Kích thước cửa sổ là 7

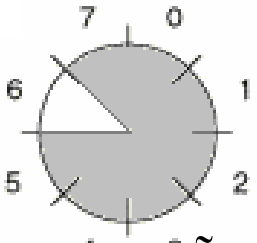
Sender



Đường truyên bị lỗi
Khung báo nhận không đến nơi

X

Receiver



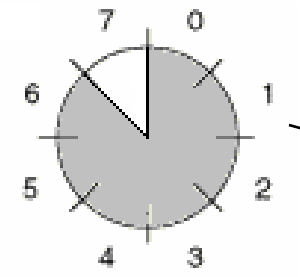
Đã gởi và chờ bào nhận các khung 0,1,2,3,4,5,6

Khung 0,1,2,3,4,5,6 không có lỗi,
Gởi báo nhận cho các khung 0,1,2,3,4,5,6
Sẵn sàng chờ nhận các khung 7,0,1,2,3,4,5

Xác định kích thước cửa sổ trượt

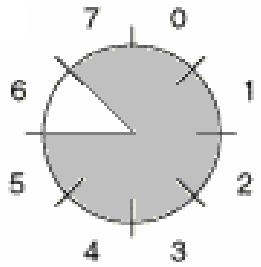
- Xét cửa sổ trượt sử dụng 3 bits để đánh chỉ số khung => Kích thước cửa sổ là 7

Sender



Quá thời hạn
Gởi lại khung 0
Chờ bào nhận
các khung 0,1,2,3,4,5,6

Receiver

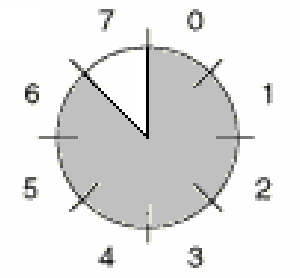


Đang sẵn sàng chờ nhận
các khung 7,0,1,2,3,4,5

Xác định kích thước cửa sổ trượt

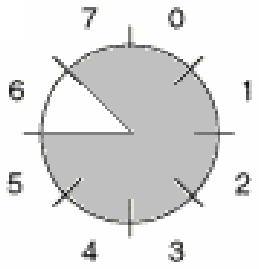
- Xét cửa sổ trượt sử dụng 3 bits để đánh chỉ số khung => Kích thước cửa sổ là 7

Sender



Quá thời hạn
Gởi lại khung 0
Chờ bào nhận
các khung 0,1,2,3,4,5,6

Receiver



Khung 0 đến nơi,
Là khung đang chờ nhận, đưa lên tầng mạng => ***tầng mạng nhận 2 lần khung 0***

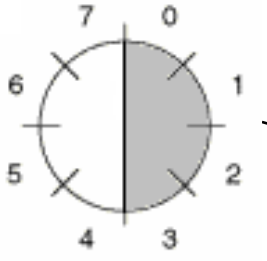
Xác định kích thước cửa sổ trượt

- Phải đảm bảo rằng cửa sổ nhận mới không đè chồng lên cửa sổ trước đó.
- Kích thước tối đa của của sổ nhận bằng một nửa khoảng đánh số thứ tự của khung.
  - Ví dụ:
    - Nếu dùng 3 bit để đánh số thứ tự khung từ 0 đến 7 thì kích thước tối đa cửa sổ nhận là (7-0+1)/2 = **4**.
    - Nếu dùng 4 bit để đánh số thứ tự khung từ 0 đến 15 thì kích thước tối đa cửa sổ nhậnt là (15-0+1)/2 = **8**.

Xác định kích thước cửa sổ trượt

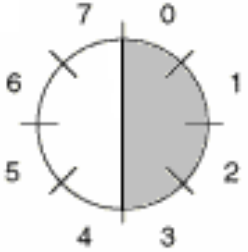
- Xét cửa sổ trượt sử dụng 3 bits để đánh chỉ số khung => Kích thước cửa sổ là 4

Sender



Đã gởi và chờ báo nhận các khung 0,1,2,3,

Receiver

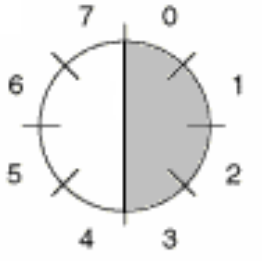


Đang sẵn sàng chờ nhận các khung 0,1,2,3

Xác định kích thước cửa sổ trượt

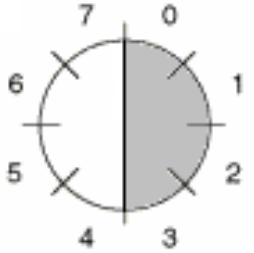
- Xét cửa sổ trượt sử dụng 3 bits để đánh chỉ số khung => Kích thước cửa sổ là 7

Sender



Đã gởi và chờ báo nhận các khung 0,1,2,3

Receiver

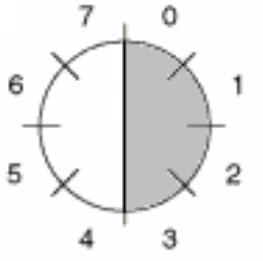


Nhận các 0,1,2,3
Kiểm tra lỗi

Xác định kích thước cửa sổ trượt

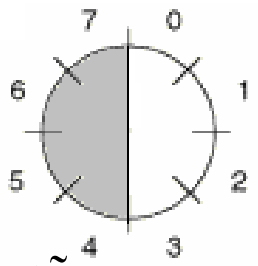
- Xét cửa sổ trượt sử dụng 3 bits để đánh chỉ số khung => Kích thước cửa sổ là 7

Sender



Đã gởi và chờ báo nhận các khung 0,1,2,3

Receiver

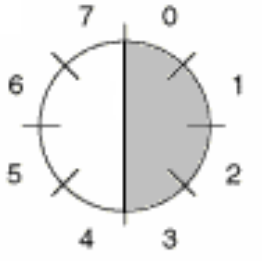


Khung 0,1,2,3 không có lỗi,
Gởi báo nhận cho các khung 0,1,2,3
Sẵn sàng chờ nhận các khung 4,5,6,7

Xác định kích thước cửa sổ trượt

- Xét cửa sổ trượt sử dụng 3 bits để đánh chỉ số khung => Kích thước cửa sổ là 7

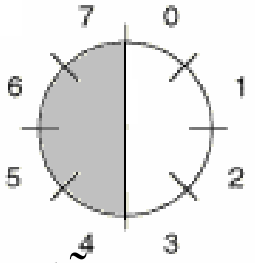
Sender



Đã gởi và chờ báo nhận các khung 0,1,2,3

Đường truyền bị lỗi
Khung báo nhận không đến nơi

Receiver

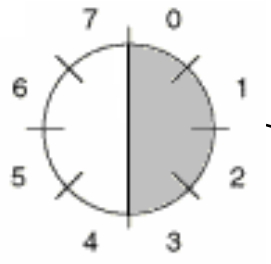


Khung 0,1,2,3 không có lỗi,
Gởi báo nhận cho các khung 0,1,2,3
Sẵn sàng chờ nhận các khung 4,5,6,7

Xác định kích thước cửa sổ trượt

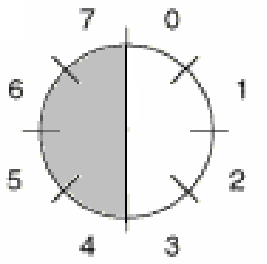
- Xét cửa sổ trượt sử dụng 3 bits để đánh chỉ số khung => Kích thước cửa sổ là 7

Sender



Quá thời hạn
Gởi lại khung 0
Chờ bào nhận
các khung 0,1,2,3

Receiver

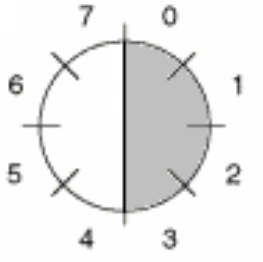


Đang sẵn sàng chờ nhận
các khung 4,5,6,7

Xác định kích thước cửa sổ trượt

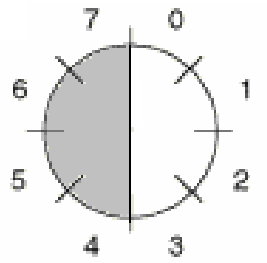
- Xét cửa sổ trượt sử dụng 3 bits để đánh chỉ số khung => Kích thước cửa sổ là 7

Sender



Quá thời hạn
Gởi lại khung 0
Chờ bào nhận
các khung 0,1,2,3

Receiver



Khung 0 đến nơi, Là khung đã nhận
*=> **không đưa lên tầng mạng***

Kích thước vùng đệm dữ liệu (buffer)

- Số lượng buffer chỉ cần bằng kích thước tối đa của cửa sổ nhận, không cần thiết phải bằng số lượng khung.
- Ví dụ: Nếu dùng 3 bit để đánh số thứ tự khung từ 0 đến 7 thì kích thước tối đa cửa sổ nhận là (7-0+1)/2 =4 và số lượng buffer cần thiết cũng là 4.

Thời điểm gởi báo nhận

- Piggy-back: gói báo nhận vào khung dữ liệu của bên nhận
- Bên nhận không còn dữ liệu để gởi đi ?
  - Mỗi lần khung đến khởi động một timer (start_ack_timer < thời gian chờ báo nhận khung dữ liệu).
  - Khi Timer đến giá trị ack_timeout mà bên nhận không có dữ liệu để gởi => Gởi một khung báo nhận riêng (ack_timeout < thời gian chờ báo nhận).

GIAO THỨC HDLC
(High Level Data Link Control)

Giao thức HDLC

- Được sử dụng rất rộng rãi.
- Là cơ sở cho nhiều giao thức điều khiển liên kết dữ liệu.
- HDLC định nghĩa:
  - 03 loại máy trạm.
  - 02 cấu hình đường kết nối.
  - 03 chế độ điều khiển truyền tải.

Các loại trạm (HDLC Station Types)

- **Primary station**
  - Điều khiển đường nối kết
  - Khung gởi đi là các lệnh
  - Duy trì nhiều nối kết luận lý đến các secondary station
- **Secondary station**
  - Chịu sự điều khiển của primary station
  - Các khung gởi đi là các trả lời
- **Combined station**
  - Có đặc tính của cả Primary station và Secondary station
  - Có thể gởi lệnh và trả lời

Các cấu hình đường nối kết (HDLC Link Configurations)

- **Không cân bằng (Unbalanced)**
  - **Một** Primary station và **một hoặc nhiều** secondary stations
  - Hỗ trợ full duplex và half duplex
- **Cân bằng (Balanced)**
  - Gồm hai combined stations
  - Hỗ trợ full duplex và half duplex

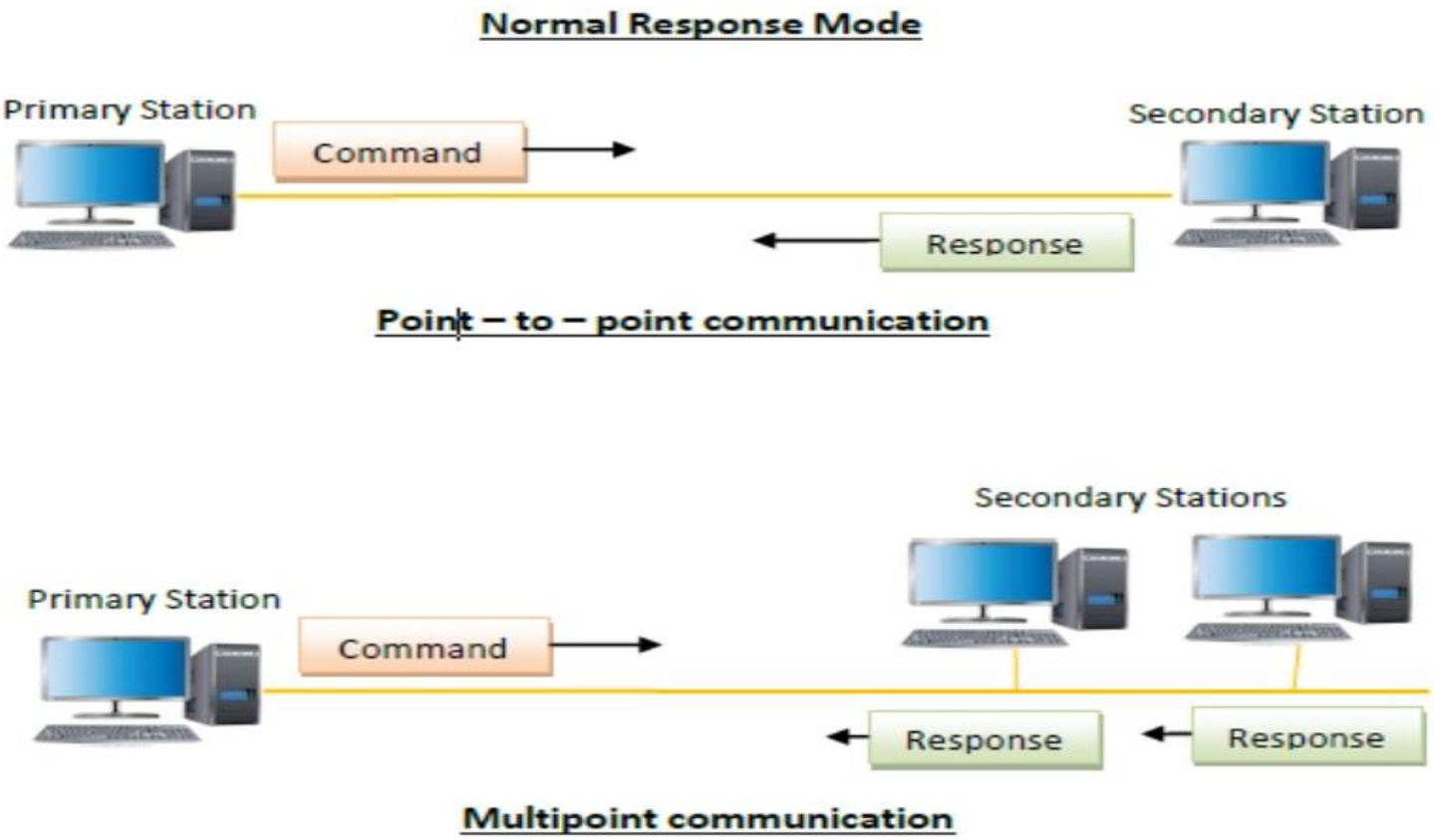
Các chế độ truyền tải (HDLC Transfer Modes)

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

Các chế độ truyền tải (HDLC Transfer Modes)

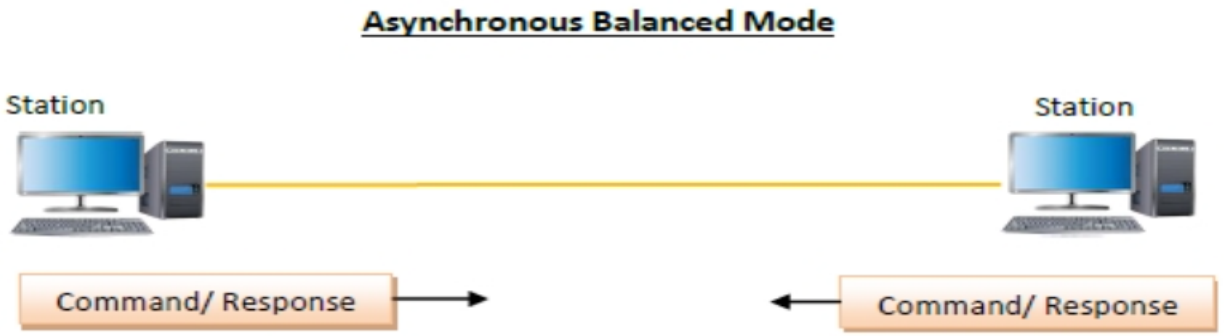
- Normal Response Mode (NRM)
  - Cấu hình **không cân bằng**
  - Primary khởi động cuộc truyền tải tới secondary
  - Secondary chỉ có thể truyền dữ liệu dưới dạng **các trả lời** cho các yêu cầu của primary
  - Được sử dụng trên các loại cáp nhiều sợi nối máy chính (Máy tính) với phụ (Terminal)
  - Máy tính đóng vai trò primary
  - Terminals đóng vai trò secondary

Normal Response Mode (NRM)



Các chế độ truyền tải (HDLC Transfer Modes)

- Asynchronous Balanced Mode (ABM)
  - Cấu hình cân bằng
  - Các trạm đều có thể khởi động một cuộc truyền tải mà không cần có phép
  - Được sử dụng rộng rãi

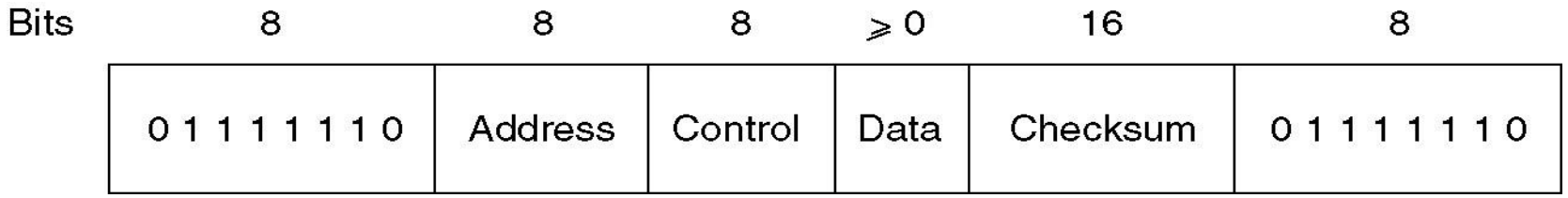


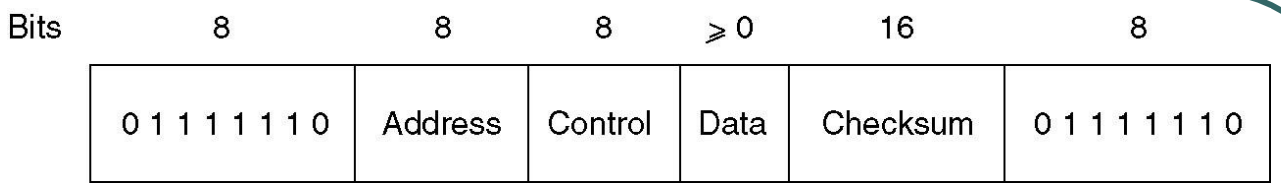
HDLC Transfer Modes (3)

- Asynchronous Response Mode (ARM)
  - Cấu hình không cân bằng
  - Secondary có thể khởi động một cuộc truyền tải mà không cần xin phép từ primary
  - Primary đảm bảo về đường truyền (khởi động, phục hồi, xóa nối kết)
  - Ít được dùng

Cấu trúc khung

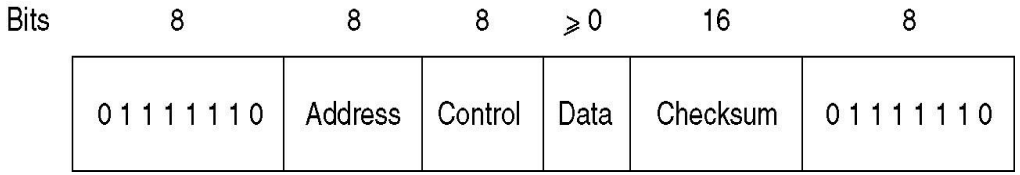
- Truyền tải đồng bộ (Synchronous transmission)
- Tất cả các cuộc truyền tải đều sử dụng khung
- Một dạng khung cho tất cả các loại dữ liệu và điều khiển





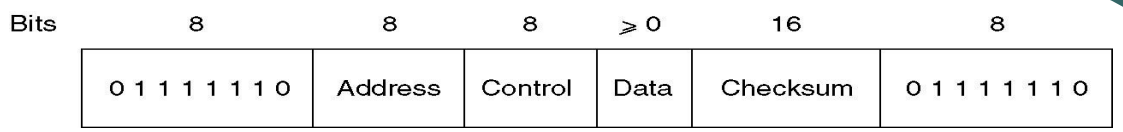
Cấu trúc khung

- **Flag (8 bit):** 01111110 , Sử dụng kỹ thuật bít độn
- **Address (8 bit):** Vùng ghi địa chỉ để **xác định máy phụ** được phép truyền hay nhận khung.
- **Control (8bit):** Được dùng để xác định loại khung:
  - Thông tin (Information),
  - Điều khiển (Supervisory)
  - Không đánh số (Unnumbered).
- **Information(128-1024 bytes):** Vùng chứa dữ liệu cần truyền.
- **Checksum:** vùng chứa mã kiểm soát lỗi
  - Phép chia đa thức: CRC-CCITT = $X^{16} + X^{12} + X^5 + 1$



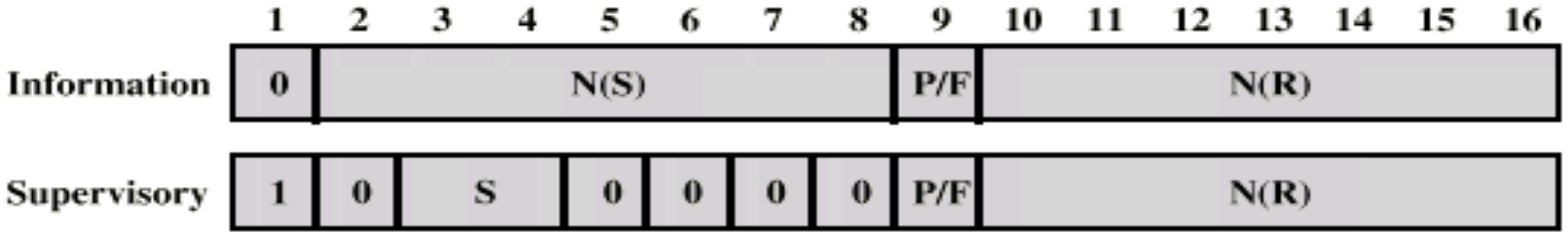
Control Field

- Khác nhau tùy thuộc vào kiểu khung
  - Information :
    - Khung chứa dữ liệu được truyền đi
    - Đồng thời chứa thông tin báo nhận (piggy-back)
  - Supervisory: khung báo nhận khi không còn dữ liệu để gởi ngược lại
  - Unnumbered: dùng để điều khiển nối kết

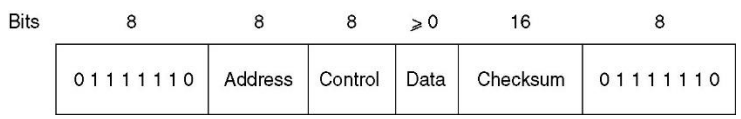


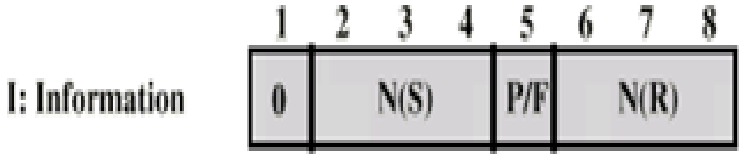
Control Field Diagram

(c) 8-bit control field format



(d) 16-bit control field format



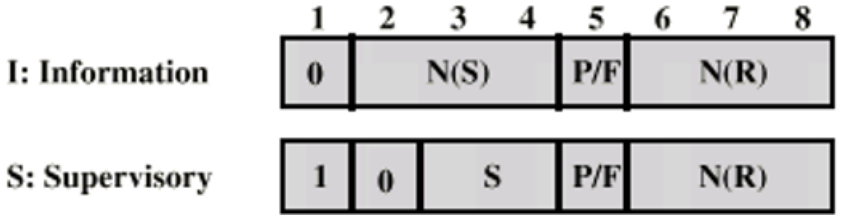


Poll/Final Bit

N(S) = Send sequence number
N(R) = Receive sequence number

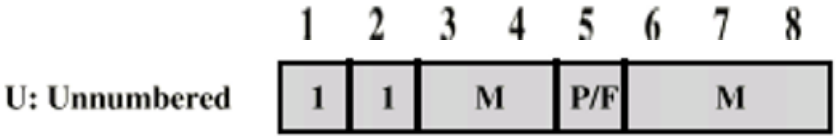
- Được sử dụng tùy thuộc vào ngữ cảnh
- Nếu là **khung lệnh**
  - Có ý nghĩa là Poll (?)
  - Giá trị 1 để yêu cầu bên kia trả lời
- Nếu là khung **trả lời**
  - Có ý nghĩa là Final
  - Khi máy phụ truyền tin lên máy chính (P=0 nếu còn dữ liệu gửi)
  - Giá trị 1 để biểu thị rằng nó kết thúc việc gởi

Supervisory function bits



SS=00	**RR** (Receive Ready), là khung báo nhận, thông báo sẵn sàng nhận dữ liệu, đã nhận tốt đến khung **Next-1**, và đang đợi nhận khung **Next**. Được dùng đến khi không còn dữ liệu gởi từ chiều ngược lại để vừa làm báo nhận (figgyback)
SS=01	**REJ** (Reject): đây là một khung báo **không nhận** (negative acknowledge), yêu cầu gởi lại các khung, từ khung Next.
SS=10	**RNR** (Receive Not Ready): thông báo **không sẵn sàng nhận tin**, đã nhận đến đến khung thứ Next-1, chưa sẵn sàng nhận khung Next
SS=11	**SREJ** (Selective Reject): **yêu cầu gởi lại một khung** có số thứ tự là **Next**

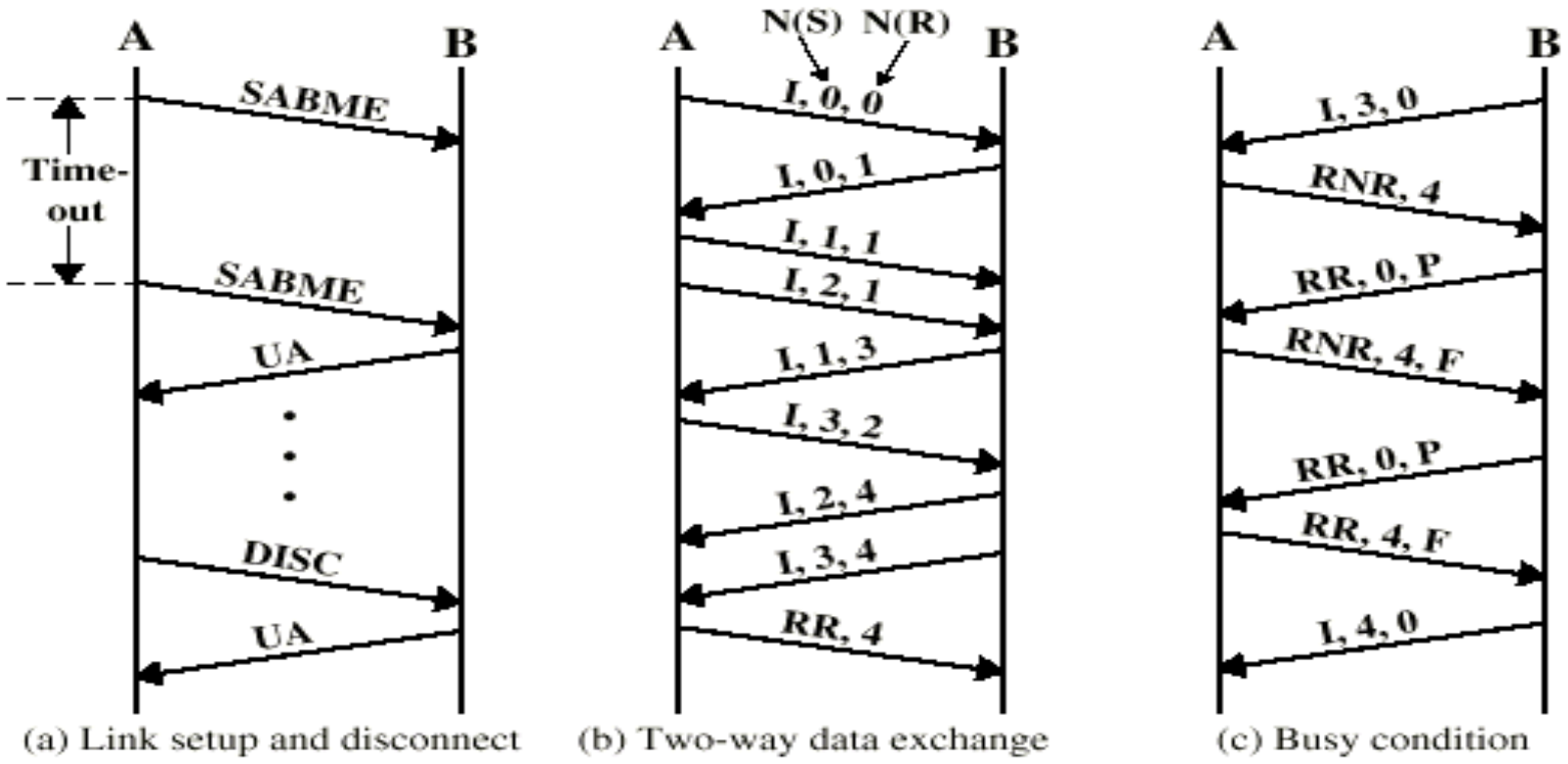
Unumbered Function Bits



1111**P**100	Lệnh này dùng để thiết lập chế độ truyền tải S**ABM** (Set Asynchronous Balanced Mode).
1100**P**001	Lệnh này dùng để thiết lập chế độ truyền tải S**NRM** (Set Normal Response Mode).
1111**P**000	Lệnh này dùng để thiết lập chế độ truyền tải S**ARM** (Set Asynchronous Response Mode).
1100**P**010	Lệnh này để yêu cầu xóa nối kết DISC (Disconnect).
1100**F**110	**UA** (Unumbered Acknowledgment). Được dùng bởi các **trạm phụ để báo với trạm chính** rằng nó đã **nhận và chấp nhận** các lệnh loại U ở trên.
1100**F**001	CMDR/FRMR (Command Reject/Frame Reject). Được dùng bởi trạm phụ để báo rằng nó **không chấp nhận** một lệnh mà nó đã nhận chính xác.

Một số kịch bản

N(S) = Send sequence number
N(R) = Receive sequence number

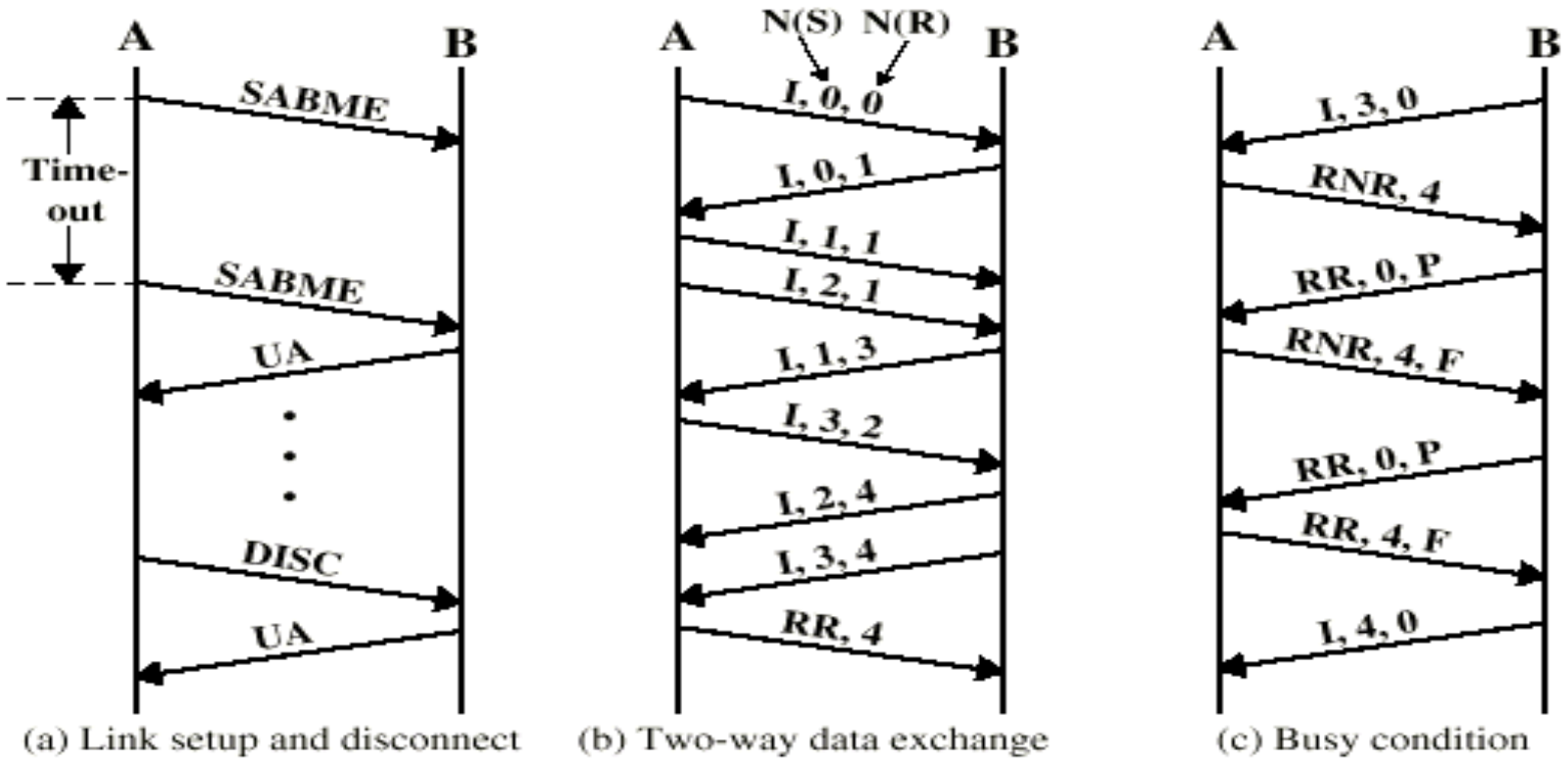


(a) Link setup and disconnect (b) Two-way data exchange (c) Busy condition

https://ebrary.net/206413/engineering/high_level_data_link_control_hdlc

Một số kịch bản

N(S) = Send sequence number
N(R) = Receive sequence number

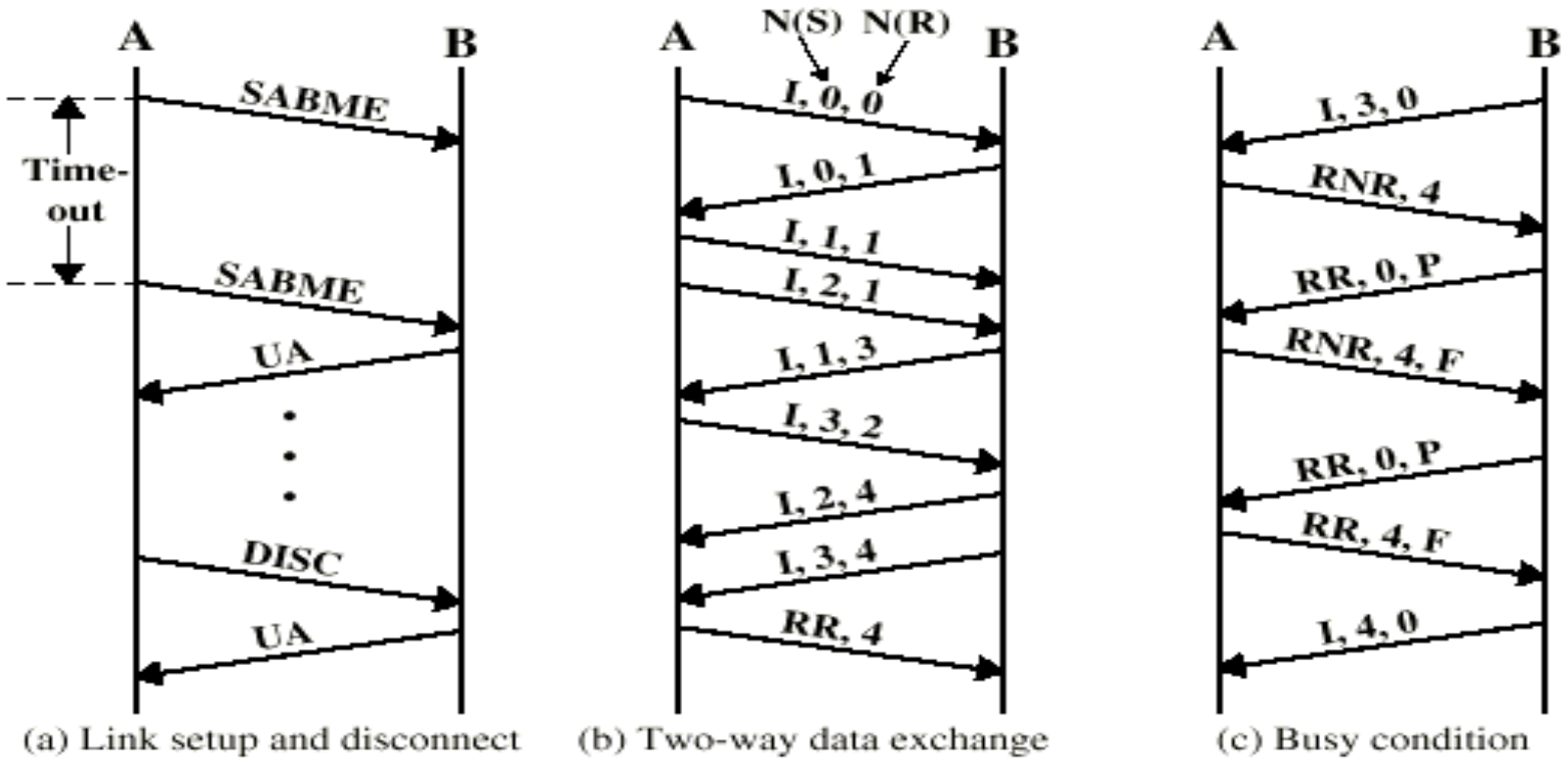


(a) Link setup and disconnect (b) Two-way data exchange (c) Busy condition

https://ebrary.net/206413/engineering/high_level_data_link_control_hdlc

Một số kịch bản

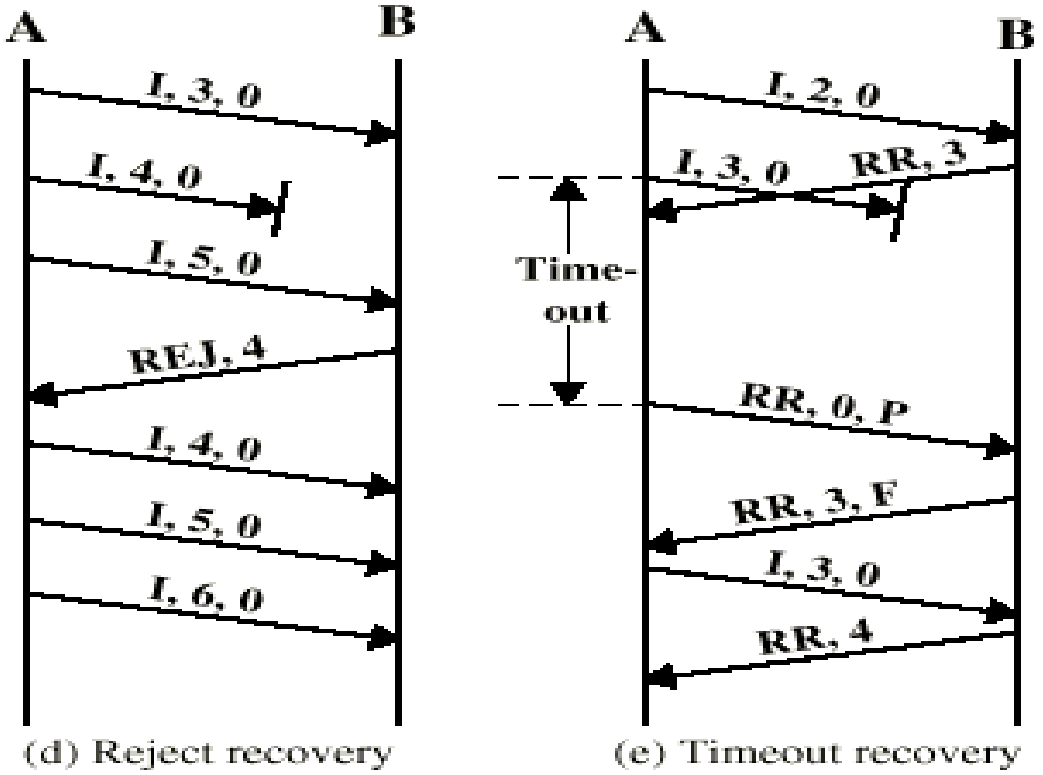
N(S) = Send sequence number
N(R) = Receive sequence number



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

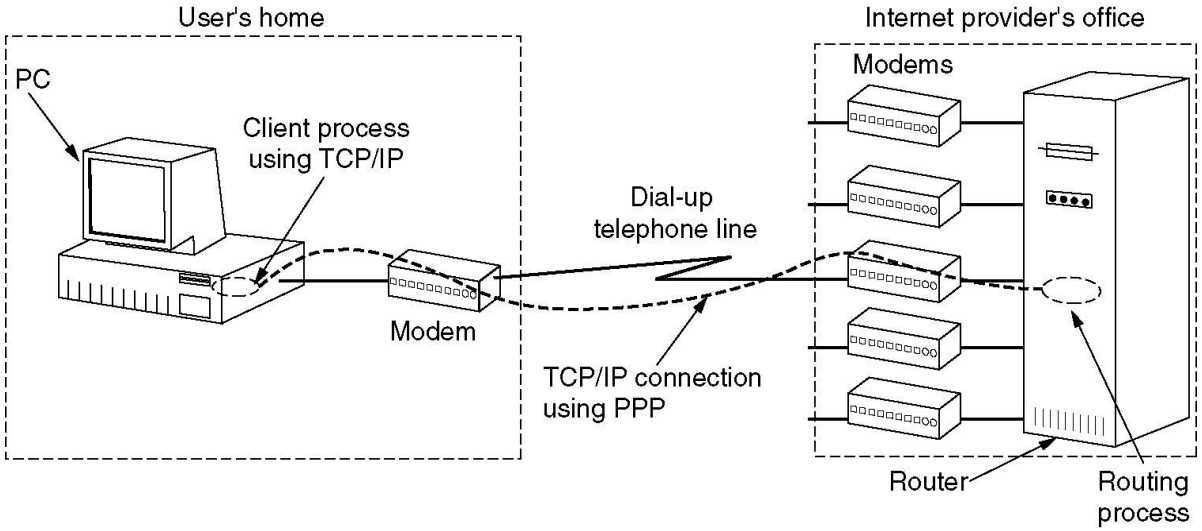
https://ebrary.net/206413/engineering/high_level_data_link_control_hdlc

Một số kịch bản



https://ebrary.net/206413/engineering/high_level_data_link_control_hdlc

Giao thức Điểm nối điểm (PPP- Point-to-Point Protocol)



- Sử dụng để truyền tải thông tin giữa các router trên mạng, hay giữa các máy tính người dùng với mạng của nhà cung cấp dịch vụ Internet (ISP)
- Được định nghĩa trong các RFC 1661, 1662, 1663.
- Hỗ trợ nhiều giao thức mạng vận hành trên nó, hỗ trợ chứng thực …

Giao thức Điểm nối điểm (PPP- Point-to-Point Protocol)

- PPP cung cấp 03 đặc tính:
  - Một phương pháp định khung cùng với phát hiện lỗi
  - Một giao thức điều khiển đường truyền (Link Control Protocol LCP): thiết lập giao tiếp, kiểm tra kênh, thỏa thuận các thông số, xoá kênh.
  - Một giao thức thương lượng các tùy chọn tầng mạng NCP (Network Control Protocol): thương lượng các tuỳ chọn tầng mạng dựa vào giao thức của tầng mạng được sử dụng.

Giao thức Điểm nối điểm (PPP- Point-to-Point Protocol)

- Xét trường hợp quay số kết nối máy tính đến Router của ISP:
  1. Máy tính quay số đến **Router của ISP** thông qua modem, khi Router tiếp nhận cuộc gọi -> một kết nối vật lý được hình thành
  2. Máy tính sẽ gởi các gói **LCP** để thoả thuận các thông số **PPP**
  3. Tiếp theo các gói tin của giao thức **NCP** được gởi đi để cấu hình tầng mạng (ví dụ: gán địa chỉ IP)
  4. Khi người dùng kết thúc, **NCP** xoá kết nối tầng mạng, giải phóng IP, tiếp theo giao thức **LCP** sẽ xoá kết nối của tầng liên kết dữ liệu, cuối cùng máy tính sẽ yêu cầu modem kết thúc cuộc gọi.

Giao thức Điểm nối điểm (PPP- Point-to-Point Protocol)

Khung trong PPP tương tự như khung HDLC

- **Flag (8 bit) có giá trị:** 01111110 , sử dụng kỹ thuật **độn byte** *(ESC: 01111101)*
- **Address (8 bit):** vùng địa chỉ, xác định địa chỉ máy đích (giá trị **1111111** đại điện cho tất cả các máy trạm)
- **Control (8bit):** có giá trị **0000011**, biểu thị giao thức không sử dụng cơ chế báo nhận, số thứ tự khung.
- **Trường protocol (1 hoặc 2 byte):** Xác định phần chứa trong data được định nghĩa bởi giao thức mạng nào (IP, SPX…)
- **Data:** dữ liệu truyền, chiều dài tối đa là 1500 bytes, LCP có thể thay đổi