1985

جامعة محمد بوضياف - المسيلة

Université Mohamed Boudiaf - M'sila

Computer Networks

Lecture 3 : Data link layer

By : Prof. Lamri SAYAD

2026

Agenda

- Introduction
- Error detection and correction
  - Information Theory
  - Error Introduction
  - Block Coding
  - Hamming Distance
  - Linear Block Codes
  - Checksum
  - CRC - Cyclic Redundancy Check
- Multiple Access Protocols
- Data Link Layer Addressing
- ARP

Goals

Chapter goals:

- ❑ understand principles behind data link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - reliable data transfer, flow control

- ❑ instantiation and implementation of various link layer technologies

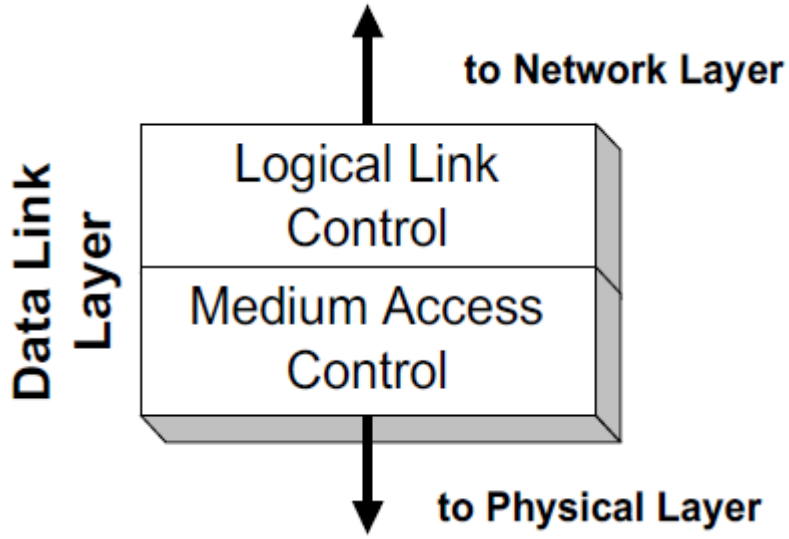
Introduction

Some terminologies

- **Data-link layer** has the responsibility of transferring **data** from one **node** to an **adjacent node** over a **link.**
- PDU is a frame, encapsulates packets
- Nodes are hosts, routers, bridges, switches,…
- Links are the communication channels that connect adjacent nodes along communication path.
  - wired links
  - wireless links

Data link layer : sub-layers

- In any broadcast network, the stations must ensure that only one station transmits at a time on the shared communication channel
- The protocol that determines who can transmit on a broadcast channel are called Medium Access Control (MAC) protocol
- The MAC protocols are implemented in the MAC sublayer which is the lower sublayer of the data link layer
- The higher portion of the data link layer is often called Logical Link Control (LLC)



Functions and services of the data link layer

1. Framing
2. Physical addressing
3. Flow control
4. Error control
5. Medium access control

Functions and services of the data link layer

- **Responsible for moving frames from one hop (node) to the next**
- **Framing**
  - Divides the stream of bits received from network layer into data units called frames
- **Physical addressing**
  - Known also as the MAC or link address
  - Data Link Layer adds a header to the frame to define the sender and receiver of the frame.
  - If the frame for a system outside the sender's network the r**eceiver address** : is the address of the connecting device that connects the network to next one (Router/switch).
  - Ethernet uses 6-bytes (48-bits) physical address that printed on the NIC

Functions and services of the data link layer

- **Flow control**
  - if the data rate at the receiver is less than the data produced by the sender, the data link layer imposes a flow control to avoid overwhelming the receiver
- **Error control**
  - Add mechanisms to detect and retransmit damaged or lost frames.
  - Prevent also duplication of frames.
  - Error control is normally achieved through a trailer added to the end of frame.
- **Medium access control**
  - When two or more devices are connected to the same link, data link layer protocols are necessary to determine which device has the control over the link at a given time

Functions and services of the data link layer

Service	Sublayer
Flow Control	LLC or DLC
Framing	MAC
Physical Addressing	
Error Control	
Access Control	

Framing

Framing

- What is framing?
  - **Framing** is a core function of the **Data Link Layer**.
  - **Framing** is the process of dividing a continuous stream of bits received from the physical layer (at the receiver side) into identifiable units called **frames**.
- The **Physical Layer** sends a **continuous stream of bits**, but the receiver must know:
  - Where a message **starts**
  - Where it **ends**
- Framing provides this structure.

Framing

- Why Framing Is Necessary?
  - Without framing, a receiver would see only streams of bits such as :

    10101100101011001001010110101 0

  - There would be **no way to know**:
    - where one message ends
    - where the next begins
  - Framing solves this by adding **structure**:

Header	Data (payload)	Trailer

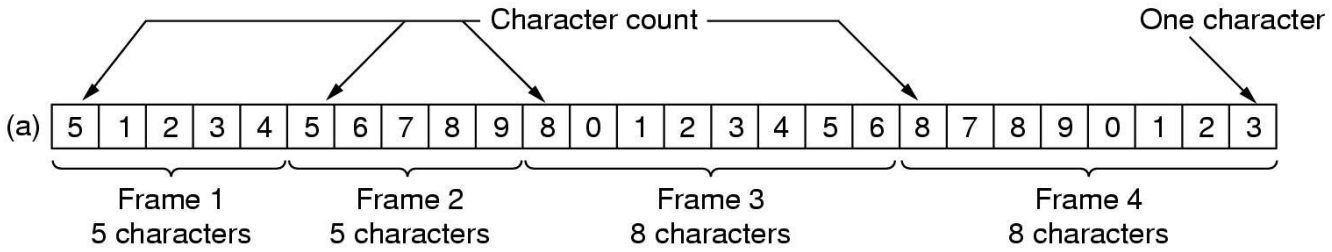
  - The **header and trailer** help identify frame boundaries and carry control information.

Framing

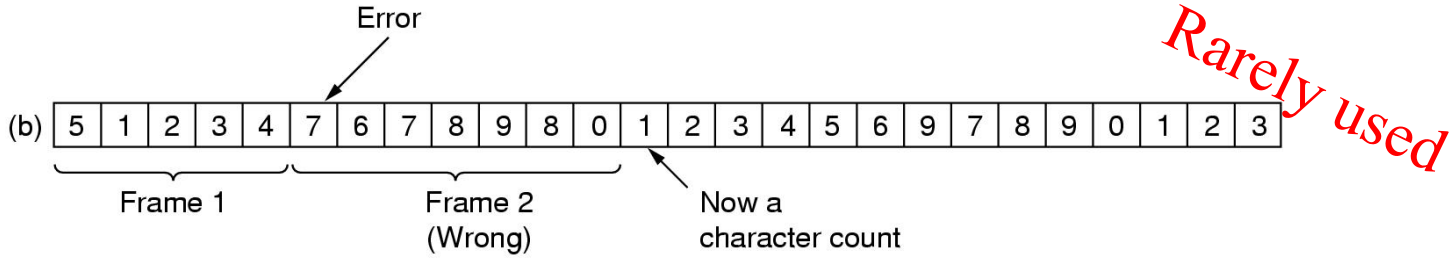
- **Framing Techniques**: There are **four main framing methods** used in data link protocols.
  - Character count
  - Starting and ending characters with character stuffing
  - Starting and ending flags with bit stuffing
  - Physical layer coding violations

Framing

- Character (Byte) count
  - The **frame header contains a field indicating the number of bytes in the frame**.



  - **Problem :** If the length field becomes corrupted, the receiver loses synchronization and cannot find the next frame boundary.



Rarely used

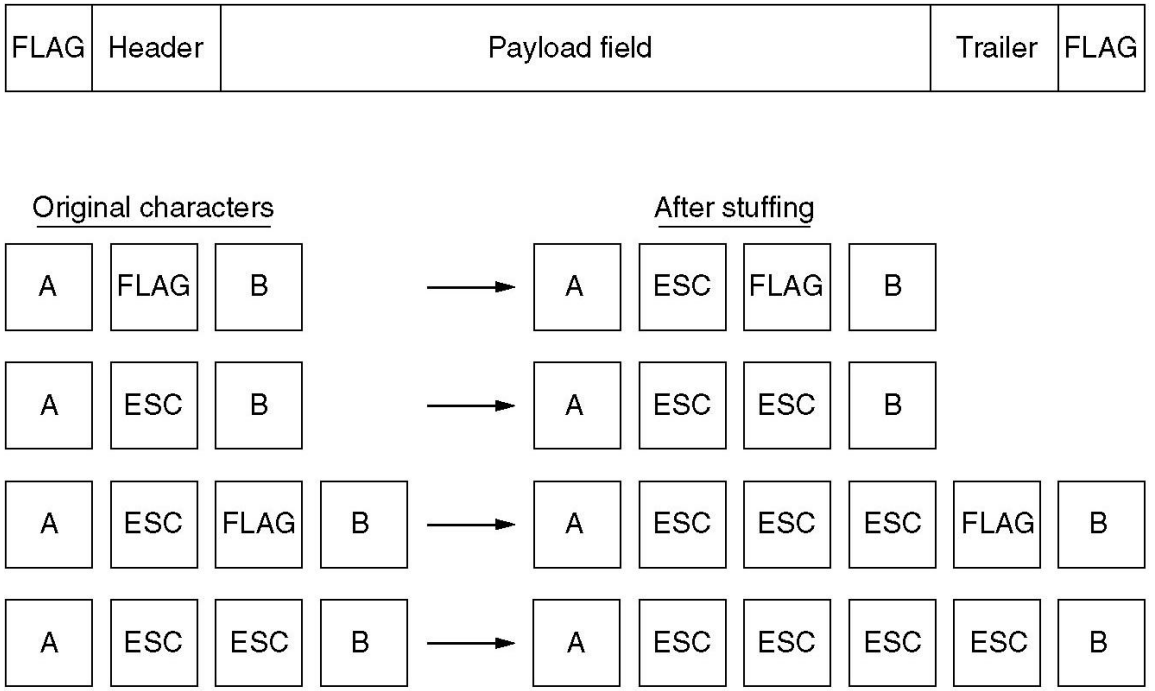
Framing

- Character stuffing : Used in **character-based protocols**.
  - Frame starts / ends with same char: FLAG

a) **Problem :** What if the **data itself contains the FLAG character**?

b) **Solution: Byte Stuffing**
   - An **escape character (ESC)** is inserted before the flag appearing inside data.

(b) 4 examples of byte sequences before and after stuffing.



Framing

- Bit stuffing : Used in **bit-oriented protocols** like **HDLC**.
  - Special bit pattern for start / end of frame
    01111110
- **Problem** : The same bit sequence could appear in data.
- **Solution**: After **five consecutive 1s**, the sender inserts a **0**.
- **Receiver rule**:
  - After five 1s, remove the next 0.
  - This ensures the flag pattern appears **only as a frame delimiter**.

Framing

- Bit stuffing : Example

(a) The original data.

(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

(b) The data as they appear on the line.

(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0

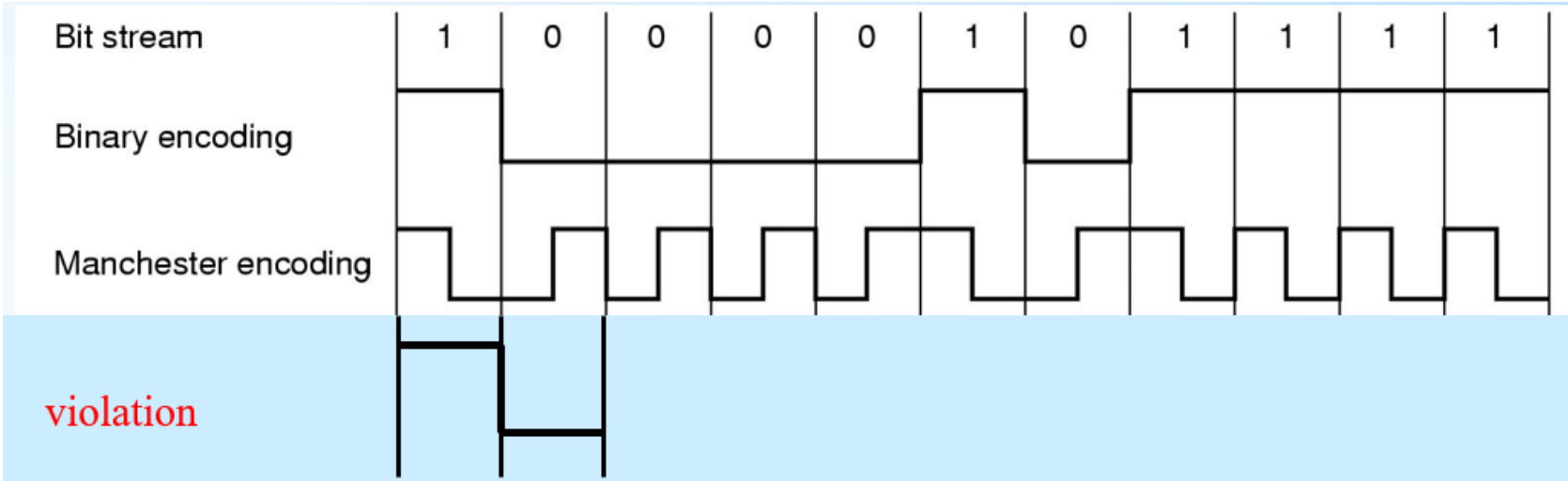
Stuffed bits

(c) The data as they are stored in receiver's memory after destuffing.

(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Framing

- Physical layer coding violation
  - Some encoding schemes contain **unused signal patterns**.
  - These unused patterns can mark **frame boundaries**.
  - e.g. Manchester encoding: transition in the middle of a slot



  - Use no transition in a slot (= coding violation) as start of frame

Error control :
Error detection and correction

Error control

- Error control
  - How does a sender know that all packets are correctly received?
    - ACK packet by receiver
  - How does a sender detect packets lost or not recognized at receiver?
    - Timer at sender: resend packet
  - How to handle duplicates or out of order received packets?
    - Sequence number in packet and ACK packet
  - Optimization: NACK packet
    - Inform sender that something strange happened

Error Detection and Correction

Error Introduction

- An electromagnetic signal is subject to interference from heat, magnetism, and other forms of electricity.

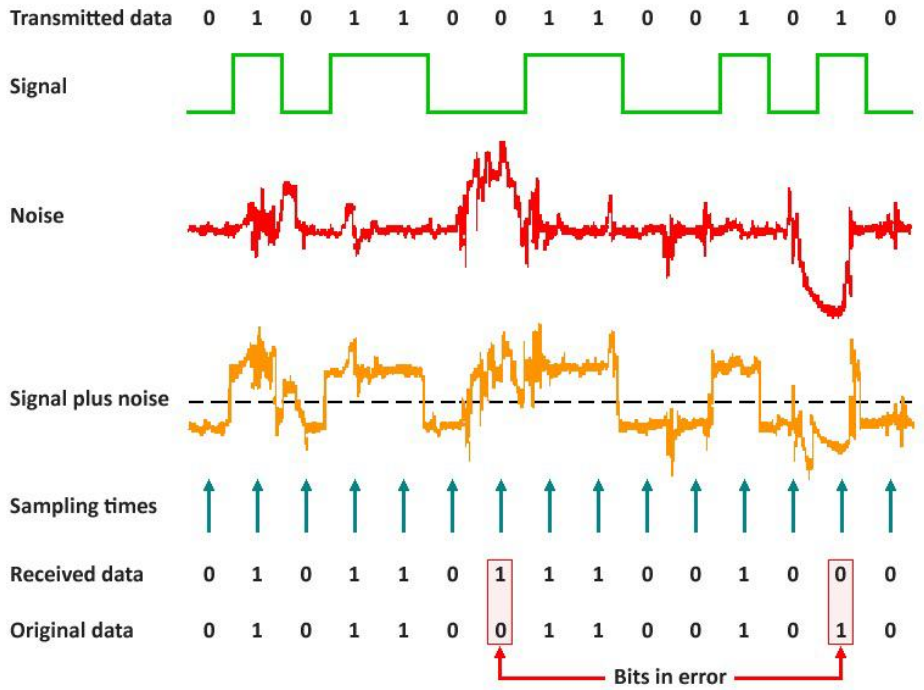
Error is result of

Attenuation,

Distortion,

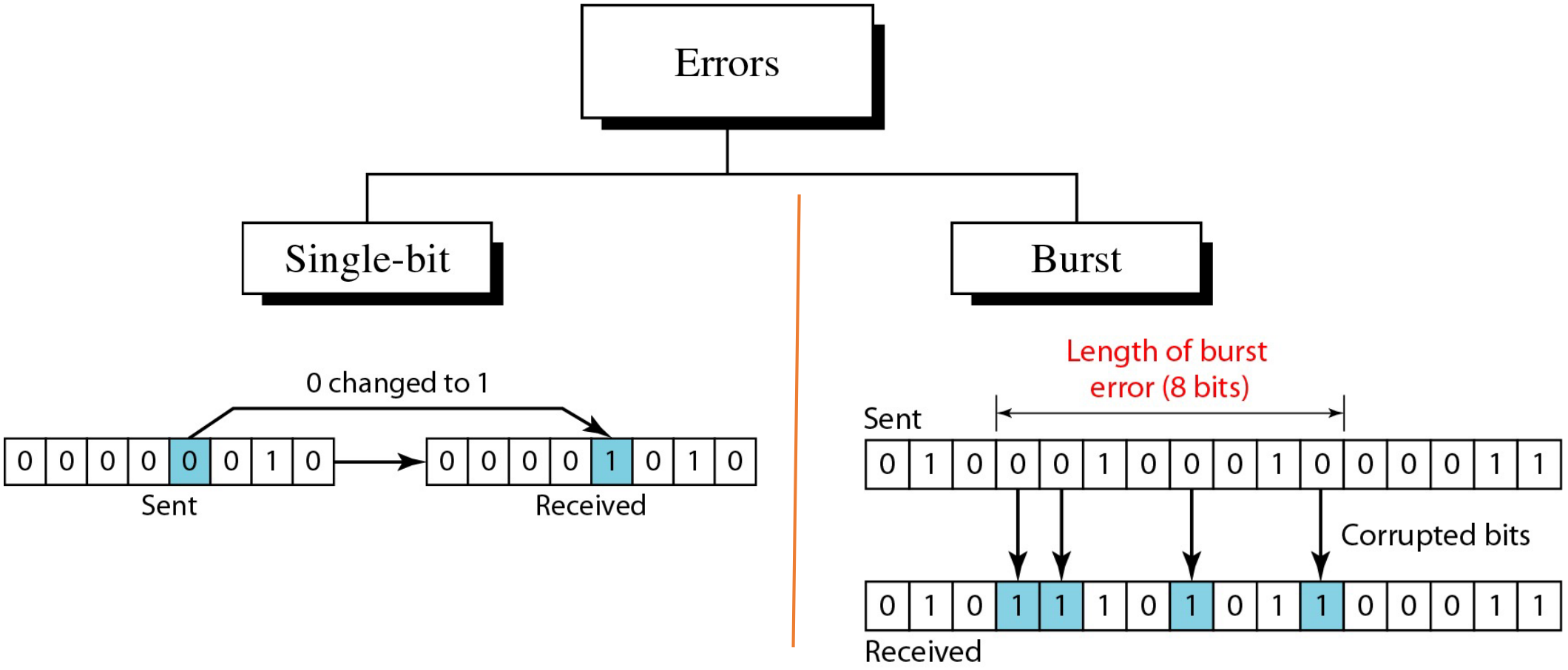
Noise sum.

Noise Problems:



Error Detection and Correction

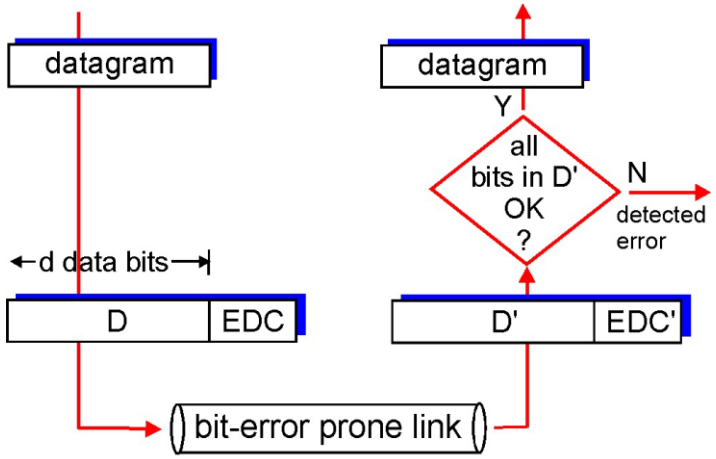
Error Introduction: Single-Bit & Burst Error



Error Detection and Correction

Error Introduction: Redundancy & Error Control

- **Error Control** uses the concept of redundancy, which means adding extra (redundant) bits for detecting errors at the destination.
- **Types of Error Control** :
  - **Error Detection**: Checks if an error has occurred? yes or no ➔ Request a retransmission.
  - **Error Correction**: include enough redundant information to enable the receiver to deduce what the transmitted data must have been even in the presence of errors.



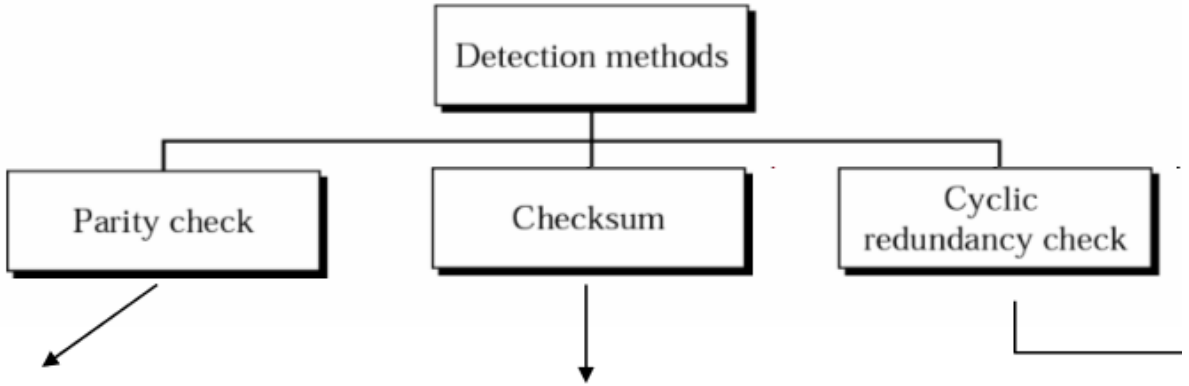
EDC= Error Detection and Correction bits (redundancy)

D = Data protected by error checking, may include header fields

- Larger EDC field yields better detection and correction

Error detection codes

Error detection methods



Single Parity

- detects all error involving an odd # of errors

2-D Parity

- detects & corrects 1-bit errors
- detects all 2- and 3- bit errors
- detects some 4-bit errors

Internet Checksum

- detects all errors involving an odd # of bits
- detects most errors involving an even # of bits

Error Detection – Single Parity Bit

- Adds one single extra bit to the frame (in the trailer field – at the end of the frame)
- Ensures that the total number of 1's in the frame is even or odd.
- **Types of Parity bits**
  - **Even parity bit:** If the number of 1's in a given set of bits is even, the parity bit value is set to 0.
  - **Odd Parity bit:** If the number of 1's in a given set of bits is odd, the parity bit value is set to 0.

- **Even parity bit:** How to compute the parity bit
  1. Take D data bits (data word),
  2. Add one check bit (or parity bit) P
  3. Sum up the D bits
  4. P = (Sum modulo 2) or XOR

- Always transmit according to the rule
- On receipt, if the rule is violated, word (frame) is invalid (corrupted)

Error Detection – Parity Bit

- How many errors will simple parity detect/correct?
  - **Answer** : odd number of transmission errors
- What about larger errors?
  - 2D Parity

Checksums

- Idea: sum up data in N-bit words
  - Widely used in, e.g., TCP/IP/UDP

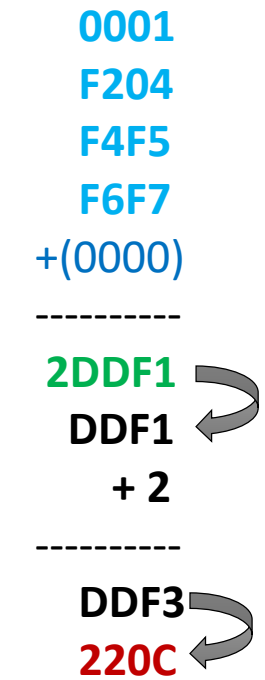
- Stronger protection than parity

Internet Checksum

- “The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words …” – RFC 791

- **Sending**: Dataword : 0001 F204 F4F5 F6F7
  1. Arrange data in 16-bit words
  2. Put zero in checksum position,
  3. Add datawords and the checksum,
  4. Add any **carryover** back to get 16 bits
  5. Negate (complement) to get sum

Codeword : 0001 F204 F4F5 F6F7 220C



Internet Checksum

Receiving: Codeword : 0001 F204 F4F5 F6F7 220C

1. Arrange data in 16-bit words
2. Checksum will be non-zero,
3. Add datawords and the checksum,
4. Add any carryover back to get 16 bits
5. Negate (complement) the result and check it it 0

Dataword : 0001 F204 F4F5 F6F7

Cyclic Redundancy Check (CRC)

- Even stronger protection
  - Given *n* data bits, generate *k* check bits such that the *n+k* bits are evenly divisible by a generator C
- The catch:
  - It's based on mathematics of finite fields, in which "numbers" represent polynomials
  - e.g, 10011010 is $x^7 + x^4 + x^3 + x^1$
- What this means:
  - We work with binary values and operate using modulo 2 arithmetic

Cyclic Redundancy Check (CRC)

- Send Procedure:
  1. Extend the *n* data bits with *k* zeros (*k* is the order of the polynomial)
  2. Divide by the generator value C
  3. Keep remainder, ignore quotient
  4. Adjust *k* check bits by remainder

- Receive Procedure:
  1. Divide and check for zero remainder

Cyclic Redundancy Check (CRC)

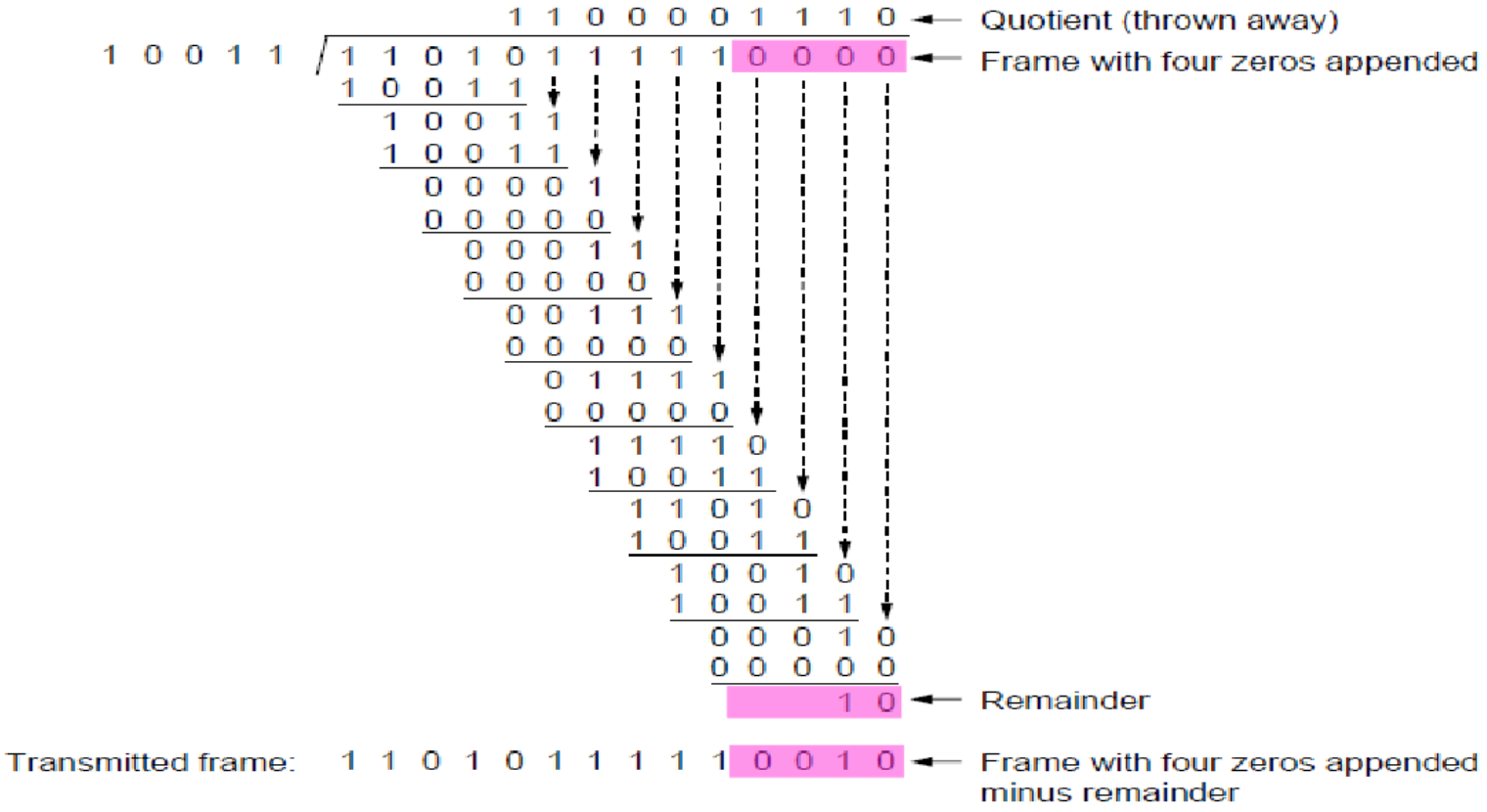
Data bits: 1101011111

Check bits:

C(x)=x^4+x^1+1

C = 10011

k = 4



Error correction codes

Error Correction

- Error detection only **identifies errors**.
- Error control ensures **correct data delivery**.

- **Methods of Error Correction** :
  - **Forward Error Correction (FEC)**:
    - Coding with redundancy.
    - The sender adds **extra redundant bits** so the receiver can **correct errors without retransmission**.
  - **Retransmission**:
    - Instead of correcting errors, the receiver asks the sender to retransmit corrupted frames.
    - This requires acknowledgment messages.

Error Correction

- Error Correction codes are
  - Hamming Codes
  - Binary Convolution Code
  - Reed Solomon Code
  - Low-Density Parity-Check Code
- All of these codes add redundancy to the information that is sent.
- A frame consists of :
  - *m* data (i.e., message) bits
  - *r* redundant (i.e. check) bits.
- Code rate :
  - fraction of the codeword that carries information that is not redundant
  - $Code\ rate = \frac{m}{(m+r)}$

Block Coding

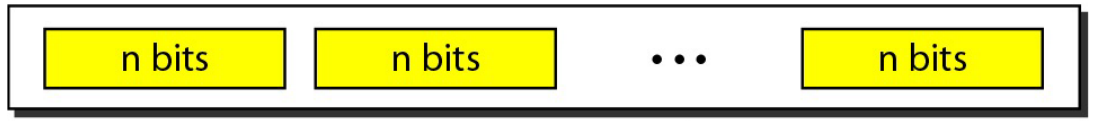
- Divide the message into blocks, each of *k* bits, called datawords.
- Add *r* redundant bits to each block to make the length *n = k + r*.
- The resulting *n*-bit blocks are called codewords.
- Select valid codewords.

- Example: 4B/5B block coding

k = 4 and n = 5.

2^k = 16 datawords and 2^n = 32 codewords.

2^k Datawords, each of k bits



2^n Codewords, each of n bits (only 2^k of them are valid)

Block Coding : Hamming Distance

- Some codewords are *valid;* others are *invalid*
- Hamming distance between two codewords is the number of bits that must be flipped to change from one to the other
  - If Hamming distance is *d* then *d* single bit errors needed to change one word to the other
- **Another definition** : The number of bit positions in which two codewords differ is called the **Hammingdistance** (Hamming, 1950).
- Examples: Hamming distance between :
  - ”Karolin-1" and "Kerstin-1" is 3;
  - 2173896 and 5233796 is 4;
  - 10101 and 11110 is 3;
  - 000 and 101 is 2.
- Hamming distance of a **code** is the minimum Hamming distance between two valid codewords.
- The error-detecting and error-correcting properties of a block code depend on its Hamming distance.

Block Coding : Hamming Distance

- The Hamming distance between two binary words *n* and *m* is the number of differences between corresponding bits.

  d(n,m) = n **XOR** m

- Detecting one single-bit error requires a code with a distance equal to 2. How does this generalize?
- Correcting one single-bit error requires a code with a distance equal to 3; how does this generalize?

Example for Table 1 of C(2,3) $d_{min} = 2$

d(000, 011) =2 d(000, 101) =2 d(000, 110) =2

d(011, 101) =2 d(011, 110) =2 d(101, 110) =2

This code guarantees detection of only a single error.

Datawords	*Codewords*
00	000
01	011
10	101
11	110

Example for Table 2 C(2,5) has $d_{min}=3$

This code can detect up to 2 errors or correct 1 error

Dataword	*Codeword*
00	00000
01	01011
10	10101
11	11110

Block Coding : Hamming Distance
Generalization

- **Error detection**:
  - For a coding of distance d+1, up to d errors will always be detected
- **Error correction**:
  - For a coding of distance 2d+1, up to d errors can always be corrected by mapping to the closest valid codeword

Linear Block Coding: Definition

- Almost all block codes used today belong to a subset called **linear block codes**.
- A Linear Block Code is a code in which the eXclusive OR (addition modulo-2) of two valid codewords creates another valid codeword.
- Examples for Tables C(2,3) and C(2,5) belong to the class of linear block codes:
  1. The scheme C(2,3) is a linear block code because the result of XORing any codeword with any other codeword is a valid codeword. For example, the XORing of the second and third codewords creates the fourth codeword: 011 XOR 101 = 110.
  2. The scheme C(2,5) is also a linear block code. We can create all four codewords by XORing two other codewords.

Datawords	*Codewords*
00	000
01	011
10	101
11	110

Dataword	*Codeword*
00	00000
01	01011
10	10101
11	11110

Linear Block Coding 1: Simple Parity-Check Code (with Detection)

- A codeword consists of n bits of which k are data bits and r are check bits n=k+r.
- A Simple Parity-check Code is a single-bit error-detecting code in which n=k+1 with dmin = 2. The extra bit called parity bit is selected to make the number of 1s in the codeword even (or odd).
- Example for Simple Parity Code k=4, r=1, n=5
  - The addition of the 4 bits of the data word is the parity bit modulo 2 (or XOR of the data bits).
  - If the number of 1s are even or 0, the result is 0;
  - if the number of 1s are odd the result is 1.

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	00000	1000	10001
0001	00011	1001	10010
0010	00101	1010	10100
0011	00110	1011	10111
0100	01001	1100	11000
0101	01010	1101	11011
0110	01100	1110	11101
0111	01111	1111	11110

Hamming Code

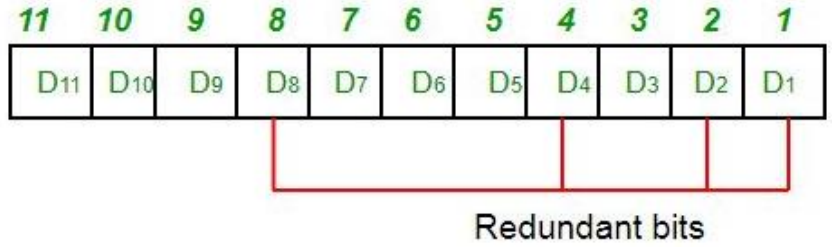
- Algorithm of **Hamming codes (with even parity bits)**
  1. Write the bit positions starting from 1 in binary form (1, 10, 11, 100, etc).
  2. All bit positions that are a power of 2 are marked as parity bits (1, 2, 4, 8, etc).
  3. All the other bit positions are marked as data bits.
  4. Each parity bit covers all bits where the bitwise AND of the parity position and the bit position is non-zero.
  5. Since we check for even parity, set a parity bit to 1 if the total number of ones in the positions it checks is odd.
  6. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

Position	R8	R4	R2	R1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1

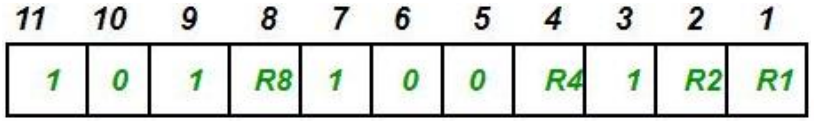
R1 -> 1,3,5,7,9,11
R2 -> 2,3,6,7,10,11
R3 -> 4,5,6,7
R4 -> 8,9,10,11

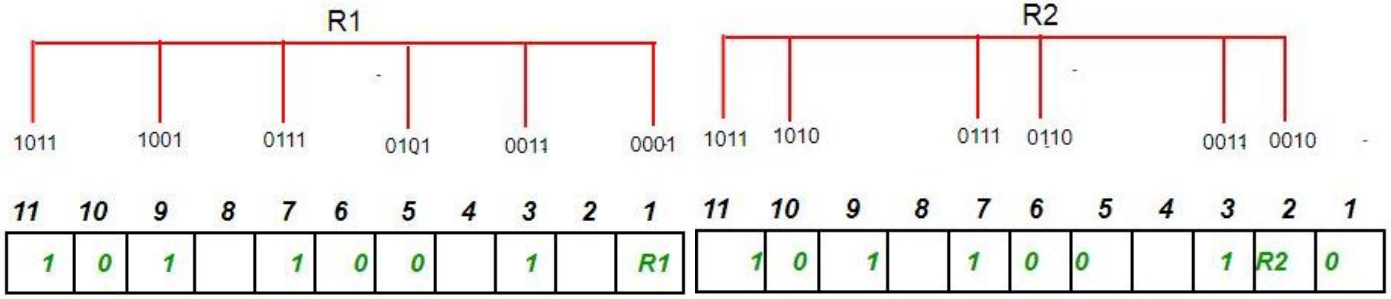
Hamming Code : Example

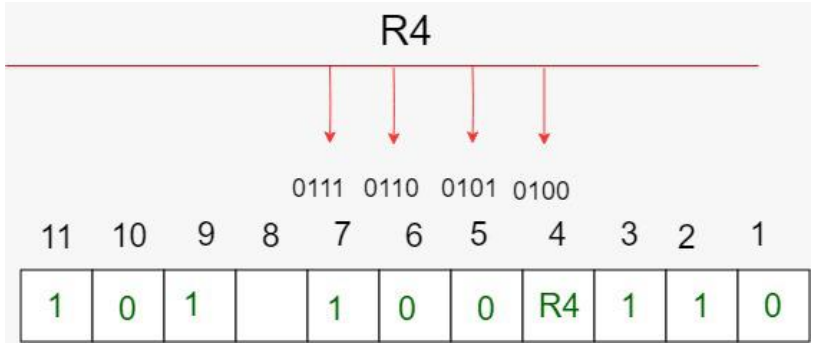
- Determining the Position of Redundant Bits

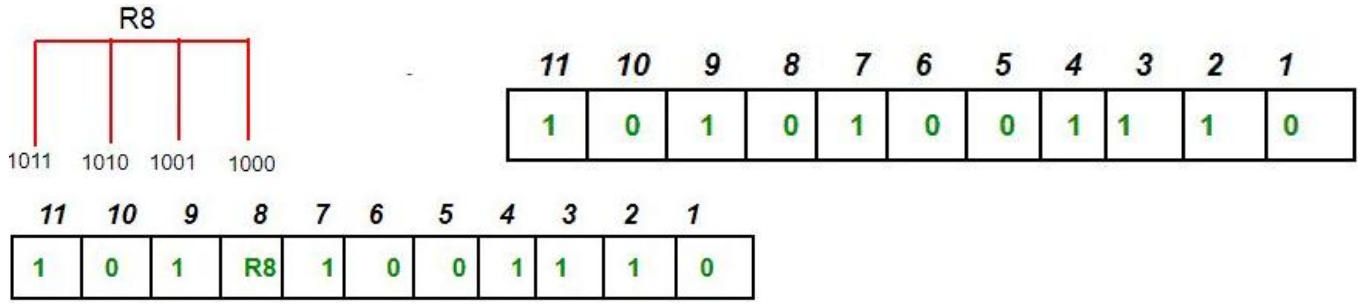


- Example : Suppose the data to be transmitted is 1011001









ARP : Address Resolution Protocol

ARP

- **What is ARP ?**

ARP (Address Resolution Protocol) is a protocol used to:

- Map an IP address (Layer 3) to a MAC address (Layer 2).

In simple terms:

- ARP answers the question : “What is the MAC address of this IP address?”

- **Why do we need ARP ?**
- Devices communicate using MAC addresses (Data Link Layer)
- But applications use IP addresses (Network Layer)
- So before sending a frame, a device must know: Destination IP → Destination
- ARP performs this translation.

ARP : How it works ?

- **Scenario :** Host A wants to send data to Host B. It knows:
  - Destination IP (e.g., 192.168.1.10)
  - But does NOT know the MAC address
- Step 1: ARP Request (Broadcast)
  - Host A sends : "Who has IP 192.168.1.10? Tell me your MAC address.“
  - Sent as a broadcast frame (Destination MAC = FF:FF:FF:FF:FF:FF) to the LAN hosts
- Step 2: ARP Reply (Unicast)
  - Host B responds:"I am 192.168.1.10, my MAC is AA:BB:CC:DD:EE:FF“
  - Sent directly to Host A (unicast)
- Step 3: ARP Table Update
  - Host A stores the mapping : 192.168.1.10 → AA:BB:CC:DD:EE:FF
  - This is saved in the ARP cache (table).
- Step 4: Data Transmission

Flow Control

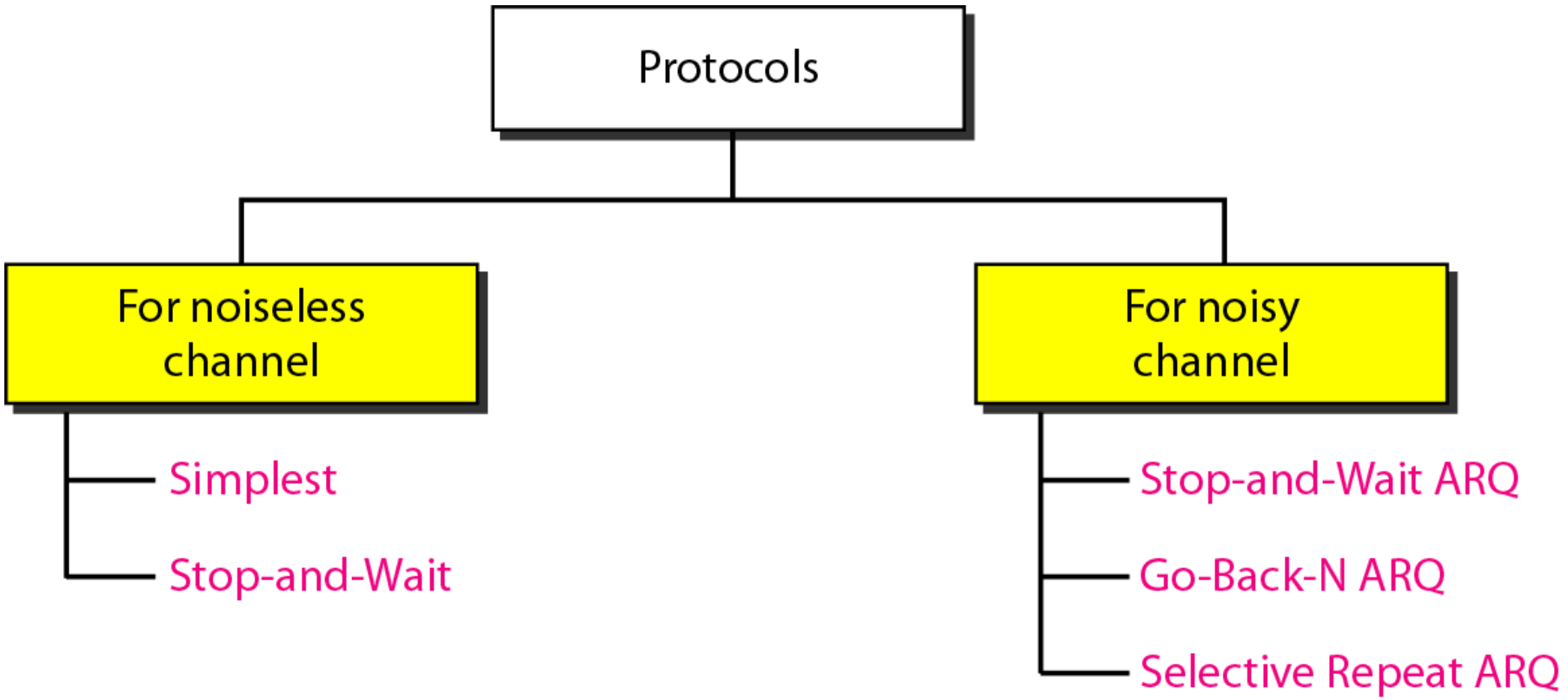
Duties of the data link layer : Flow control

- Flow control
  - How to prevent a sender to overload the receiver with packets
    - Receiver gives permission to send more packets
  - Ensuring the sending entity does not overwhelm the receiving entity
    - Preventing buffer overflow
  - Mechanisms:
    - Implicit: ACK packet implies permission
    - Explicit: communicate window size

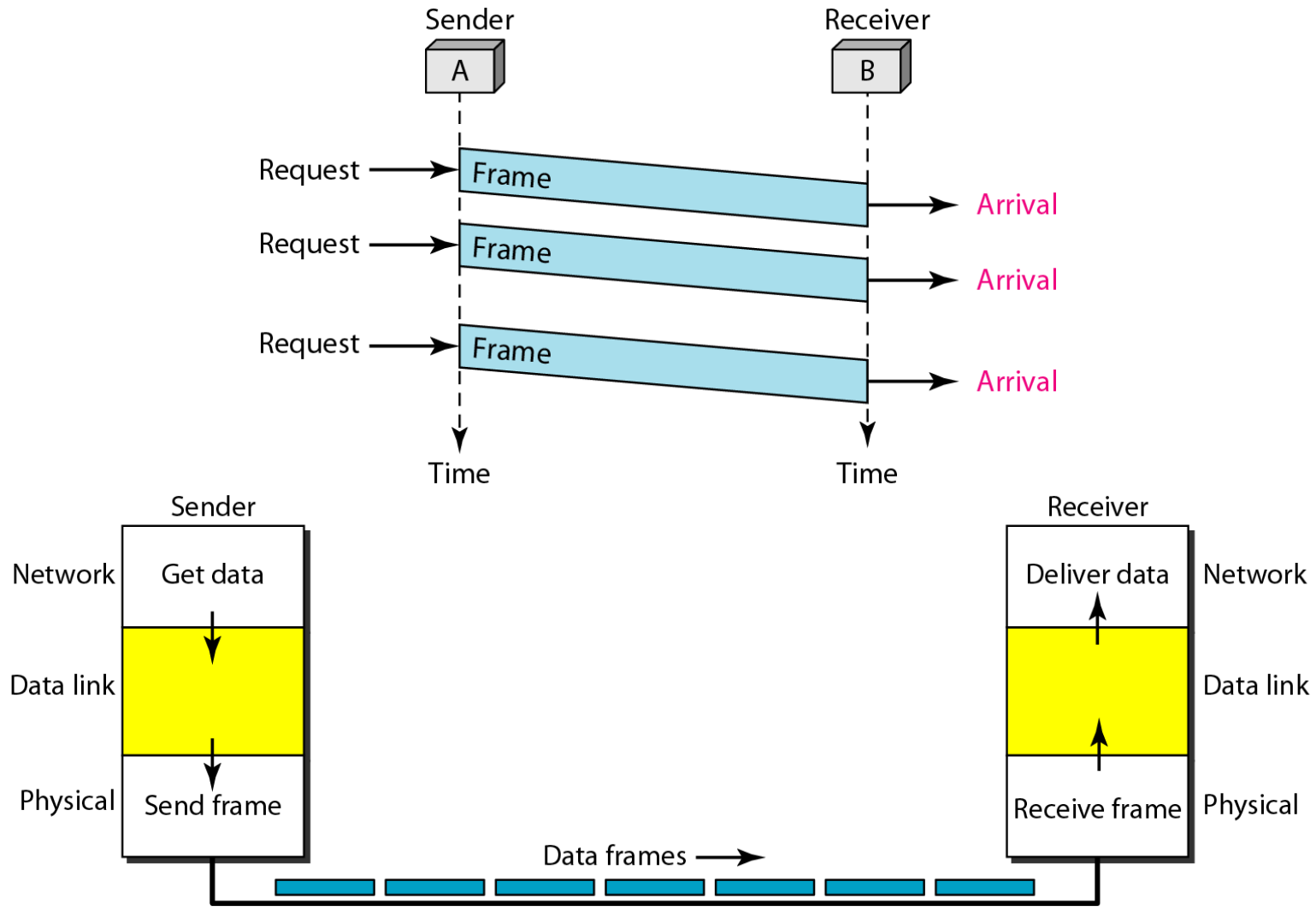
Note

Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.

Taxonomy of some protocols



Flow diagram of the Simplest protocol



The simplest protocol

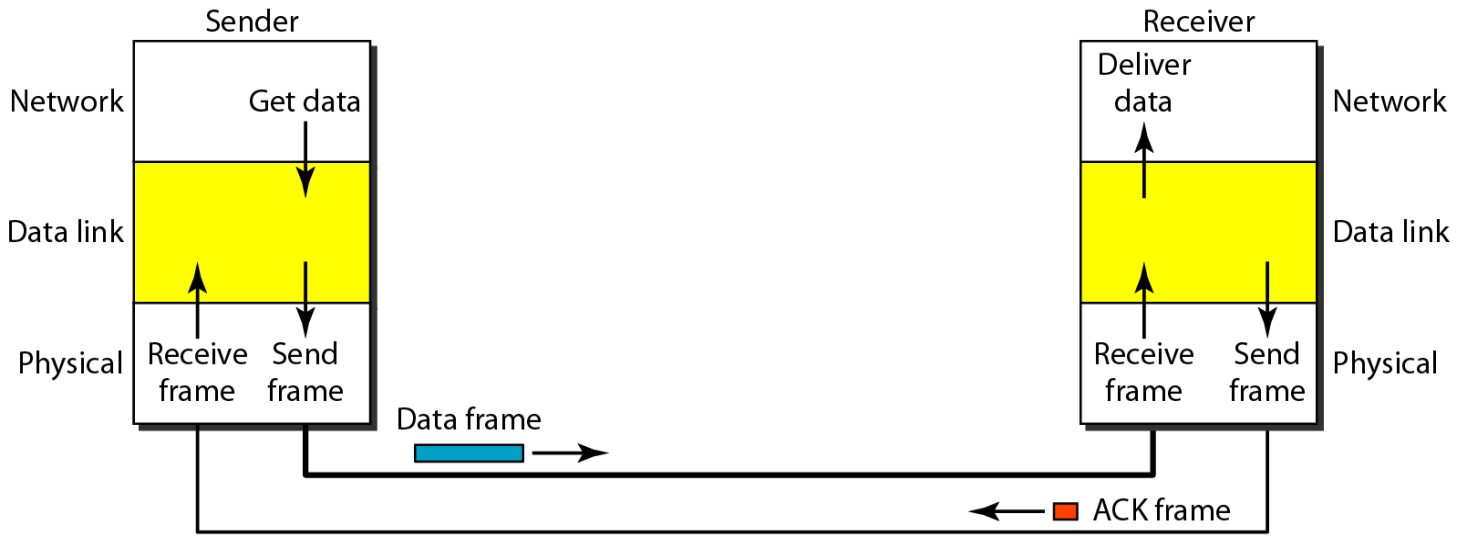
Sender

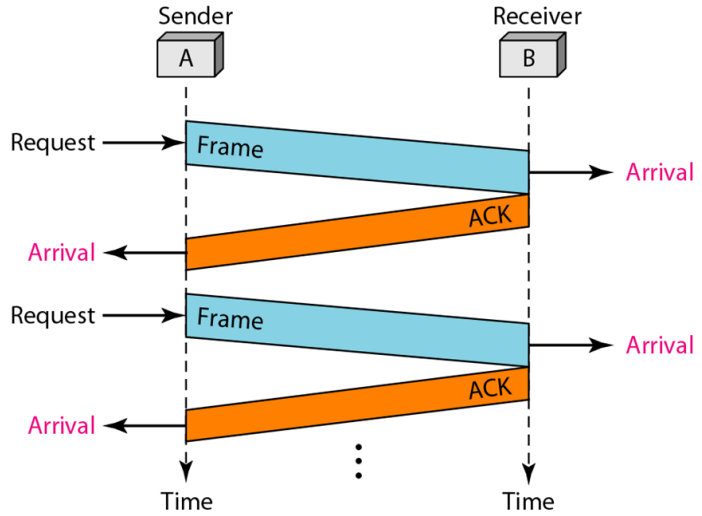
```
1  while(true)                          // Repeat forever
2  {
3    WaitForEvent();                    // Sleep until an event occurs
4    if(Event(RequestToSend))           //There is a packet to send
5    {
6       GetData();
7       MakeFrame();
8       SendFrame();                    //Send the frame
9    }
10 }
```

Receiver

```
1  while(true)                          // Repeat forever
2  {
3    WaitForEvent();                    // Sleep until an event occurs
4    if(Event(ArrivalNotification))     //Data frame arrived
5    {
6       ReceiveFrame();
7       ExtractData();
8       DeliverData();                  //Deliver data to network layer
9    }
10 }
```

Design of Stop-and-Wait Protocol





Sender-site algorithm for Stop-and-Wait Protocol

```
1  while(true)                          //Repeat forever
2  canSend = true                       //Allow the first frame to go
3  {
4    WaitForEvent();                    // Sleep until an event occurs
5    if(Event(RequestToSend) AND canSend)
6    {
7       GetData();
8       MakeFrame();
9       SendFrame();                    //Send the data frame
10      canSend = false;                //Cannot send until ACK arrives
11   }
12   WaitForEvent();                    // Sleep until an event occurs
13   if(Event(ArrivalNotification)      // An ACK has arrived
14    {
15      ReceiveFrame();                 //Receive the ACK frame
16      canSend = true;
17    }
18 }
```

Receiver-site algorithm for Stop-and-Wait Protocol

```
1  while(true)                          //Repeat forever
2  {
3    WaitForEvent();                    // Sleep until an event occurs
4    if(Event(ArrivalNotification))     //Data frame arrives
5    {
6       ReceiveFrame();
7       ExtractData();
8       Deliver(data);                  //Deliver data to network layer
9       SendFrame();                    //Send an ACK frame
10   }
11 }
```

NOISY CHANNELS

We discuss two protocols in this section that use error control.

Stop-and-Wait Automatic Repeat Request
Go-Back-N Automatic Repeat Request

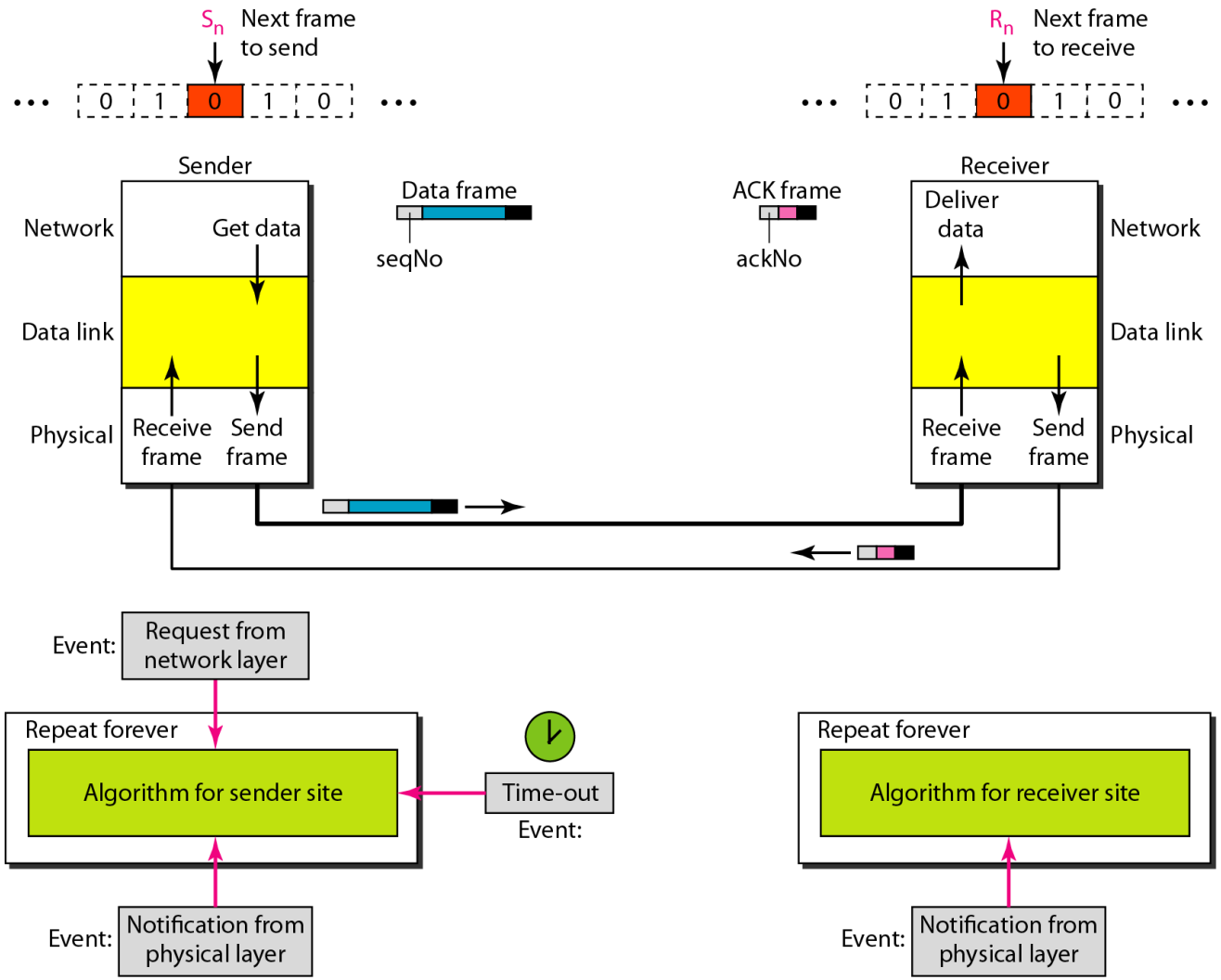
Notes

Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires.

In Stop-and-Wait ARQ, we use sequence numbers to number the frames. The sequence numbers are based on modulo-2 arithmetic.

In Stop-and-Wait ARQ, the acknowledgment number always announces in modulo-2 arithmetic the sequence number of the next frame expected.

Design of the Stop-and-Wait ARQ Protocol



Sender-site algorithm for Stop-and-Wait ARQ

```
1  Sn = 0;                                  // Frame 0 should be sent first
2  canSend = true;                          // Allow the first request to go
3  while(true)                              // Repeat forever
4  {
5    WaitForEvent();                        // Sleep until an event occurs
6    if(Event(RequestToSend) AND canSend)
7    {
8       GetData();
9       MakeFrame(Sn);                      //The seqNo is Sn
10      StoreFrame(Sn);                     //Keep copy
11      SendFrame(Sn);
12      StartTimer();
13      Sn = Sn + 1;
14      canSend = false;
15   }
16   WaitForEvent();                        // Sleep
```

(continued)

Sender-site algorithm for Stop-and-Wait ARQ

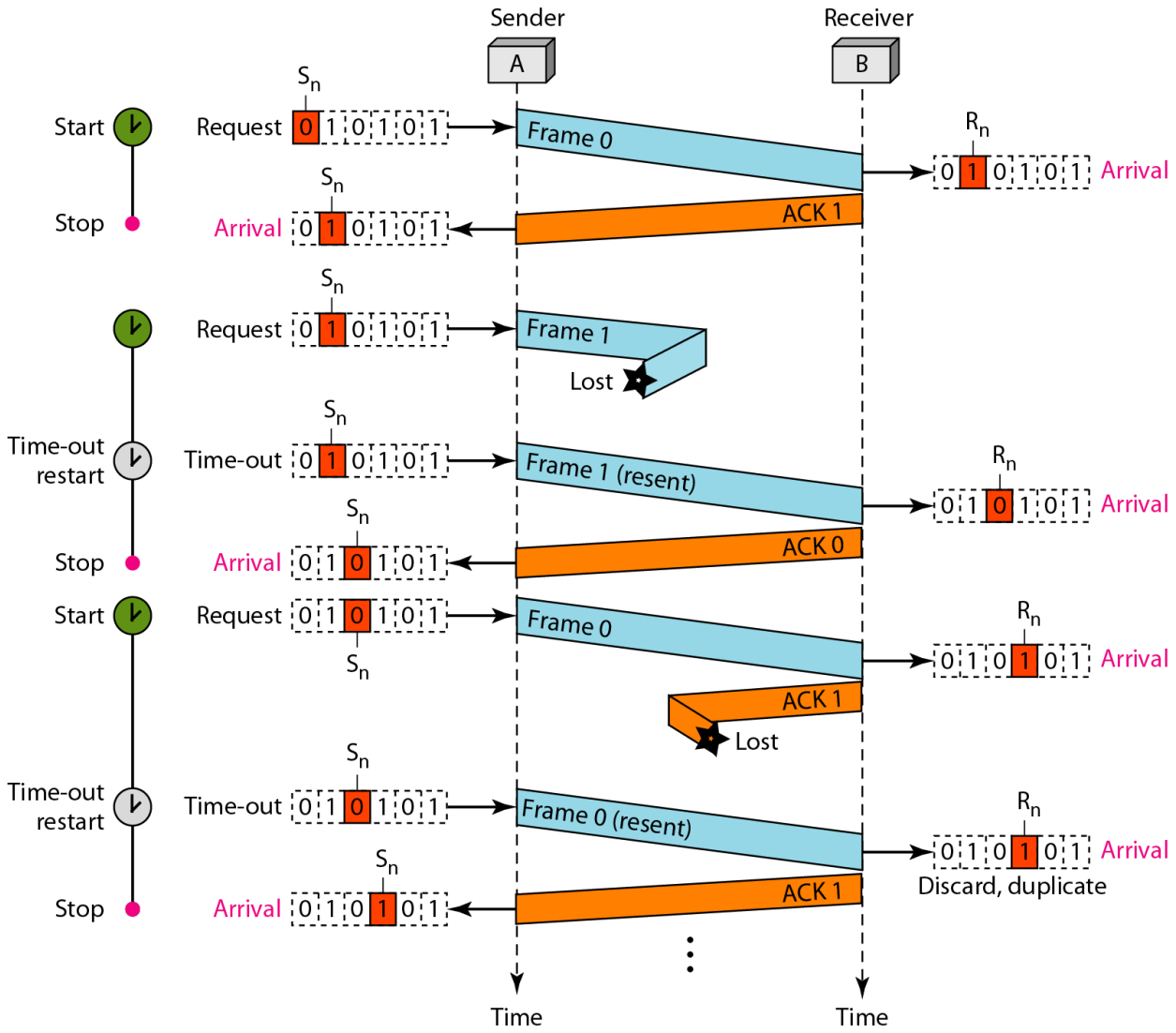
(continued)

```
17      if(Event(ArrivalNotification)       // An ACK has arrived
18      {
19        ReceiveFrame(ackNo);              //Receive the ACK frame
20        if(not corrupted AND ackNo == Sn) //Valid ACK
21          {
22            Stoptimer();
23            PurgeFrame(Sn-1);             //Copy is not needed
24            canSend = true;
25          }
26       }
27
28      if(Event(TimeOut)                   // The timer expired
29      {
30       StartTimer();
31       ResendFrame(Sn-1);                 //Resend a copy check
32      }
33  }
```

Receiver-site algorithm for Stop-and-Wait ARQ Protocol

```
 1  Rn = 0;                          // Frame 0 expected to arrive first
 2  while(true)
 3  {
 4    WaitForEvent();                // Sleep until an event occurs
 5    if(Event(ArrivalNotification))  //Data frame arrives
 6    {
 7       ReceiveFrame();
 8       if(corrupted(frame));
 9          sleep();
10       if(seqNo == Rn)             //Valid data frame
11       {
12        ExtractData();
13         DeliverData();            //Deliver data
14         Rn = Rn + 1;
15       }
16        SendFrame(Rn);             //Send an ACK
17    }
18  }
```

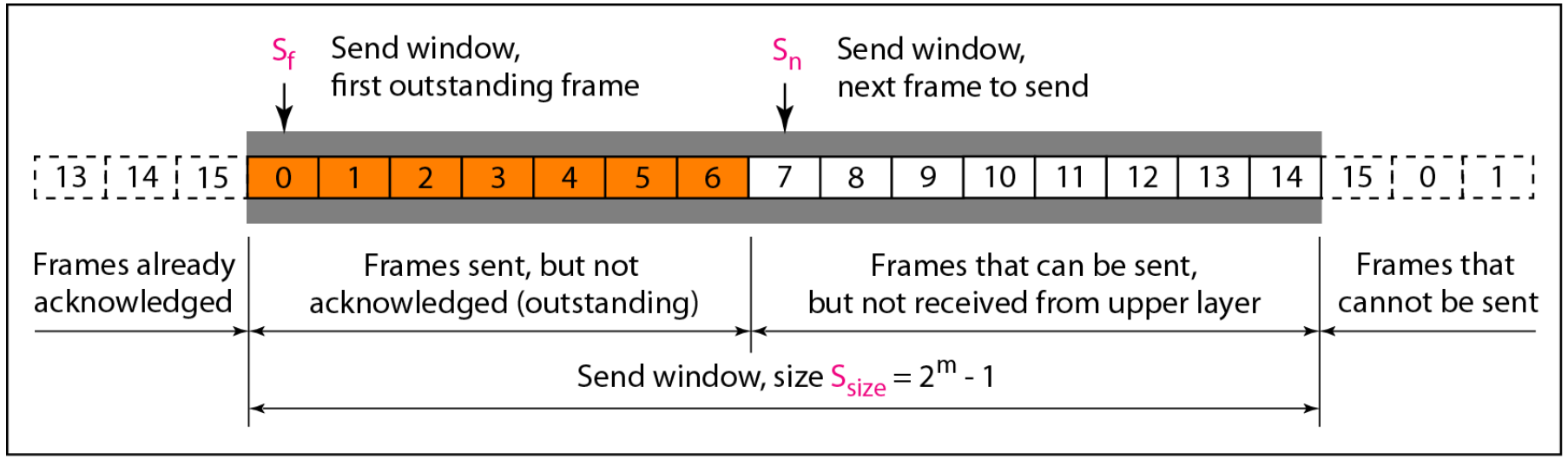
Flow diagram



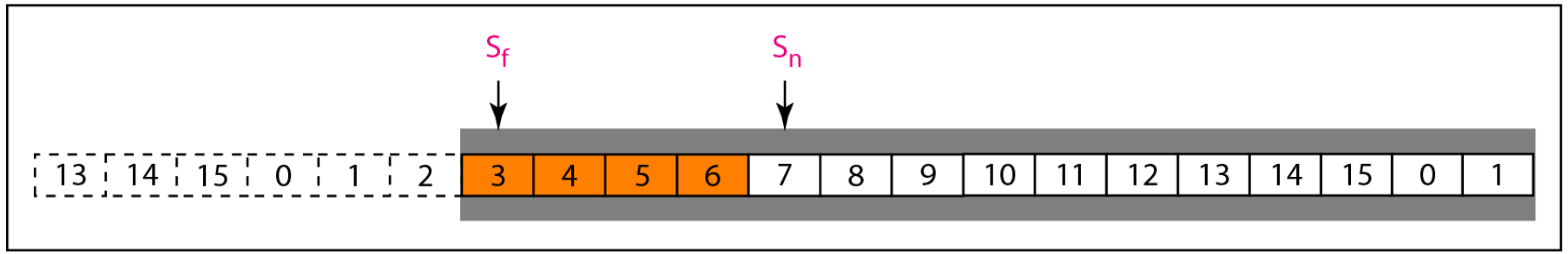
Note

In the Go-Back-N Protocol, the sequence numbers are modulo 2^m, where m is the size of the sequence number field in bits.

Send window for Go-Back-N ARQ

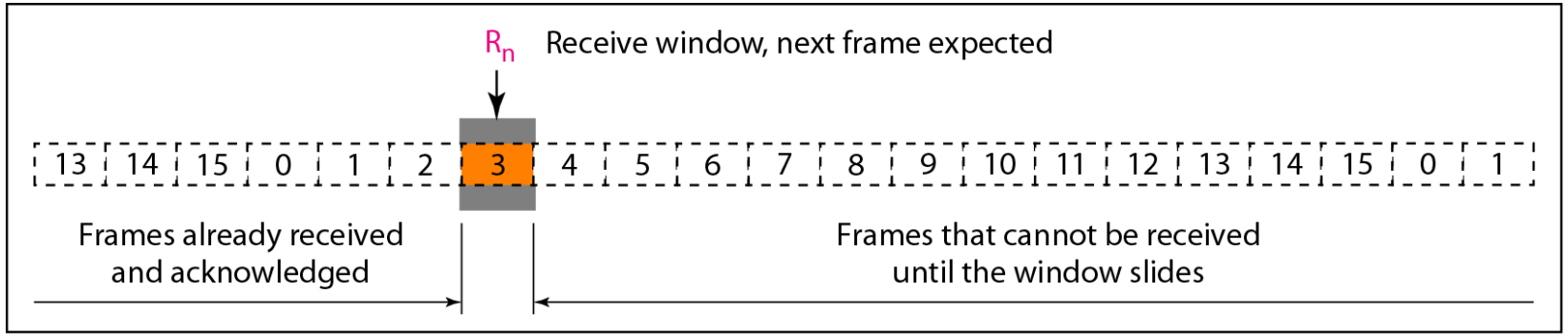


a. Send window before sliding

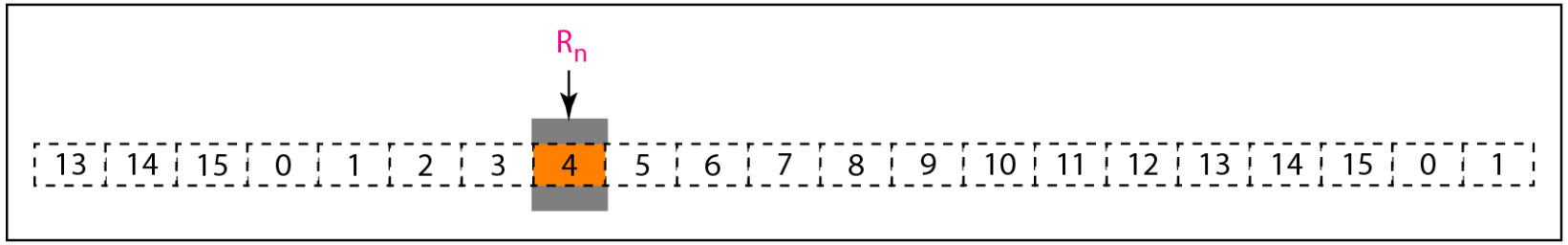


b. Send window after sliding

Receive window for Go-Back-N ARQ



a. Receive window

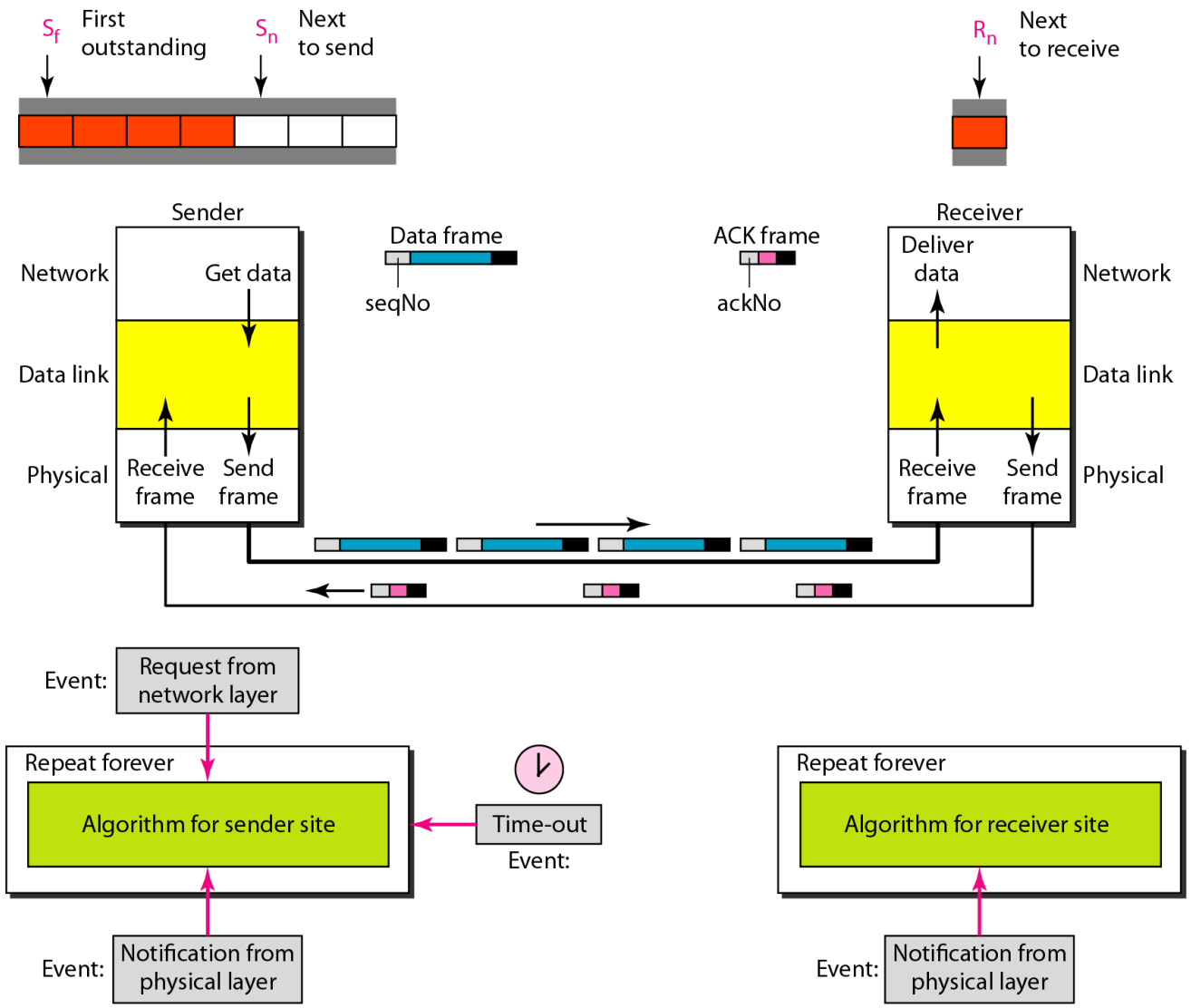


b. Window after sliding

Note

The receive window is an abstract concept defining an imaginary box of size 1 with one single variable R_n.
The window slides
when a correct frame has arrived; sliding occurs one slot at a time.

Design of Go-Back-N ARQ



Go-Back-N sender algorithm

```
1  Sw = 2^m - 1;
2  Sf = 0;
3  Sn = 0;
4
5  while (true)                          //Repeat forever
6  {
7   WaitForEvent();
8    if(Event(RequestToSend))            //A packet to send
9    {
10      if(Sn-Sf >= Sw)                  //If window is full
11            Sleep();
12      GetData();
13      MakeFrame(Sn);
14      StoreFrame(Sn);
15      SendFrame(Sn);
16      Sn = Sn + 1;
17      if(timer not running)
18           StartTimer();
19   }
20
```

(continued)

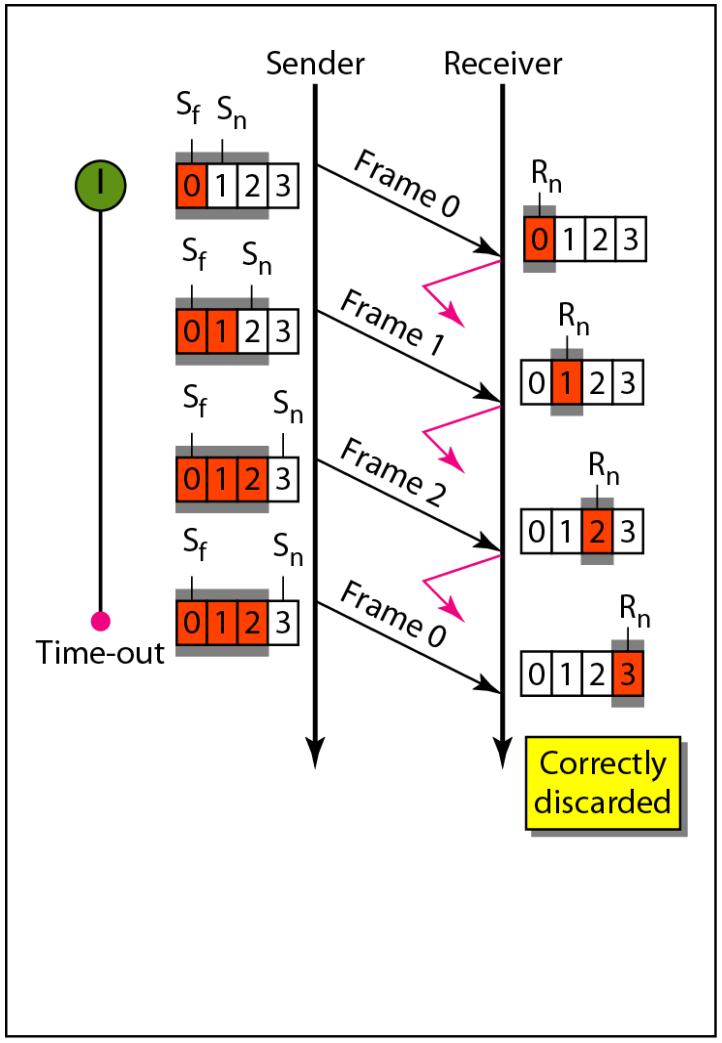
Go-Back-N sender algorithm *(continued)*

```
21    if(Event(ArrivalNotification))  //ACK arrives
22    {
23       Receive(ACK);
24       if(corrupted(ACK))
25            Sleep();
26       if((ackNo>Sf)&&(ackNo<=Sn))  //If a valid ACK
27       While(Sf <= ackNo)
28        {
29         PurgeFrame(Sf);
30         Sf = Sf + 1;
31        }
32        StopTimer();
33    }
34
35    if(Event(TimeOut))               //The timer expires
36    {
37     StartTimer();
38     Temp = Sf;
39     while(Temp < Sn);
40      {
41       SendFrame(Sf);
42       Sf = Sf + 1;
43      }
44    }
45  }
```

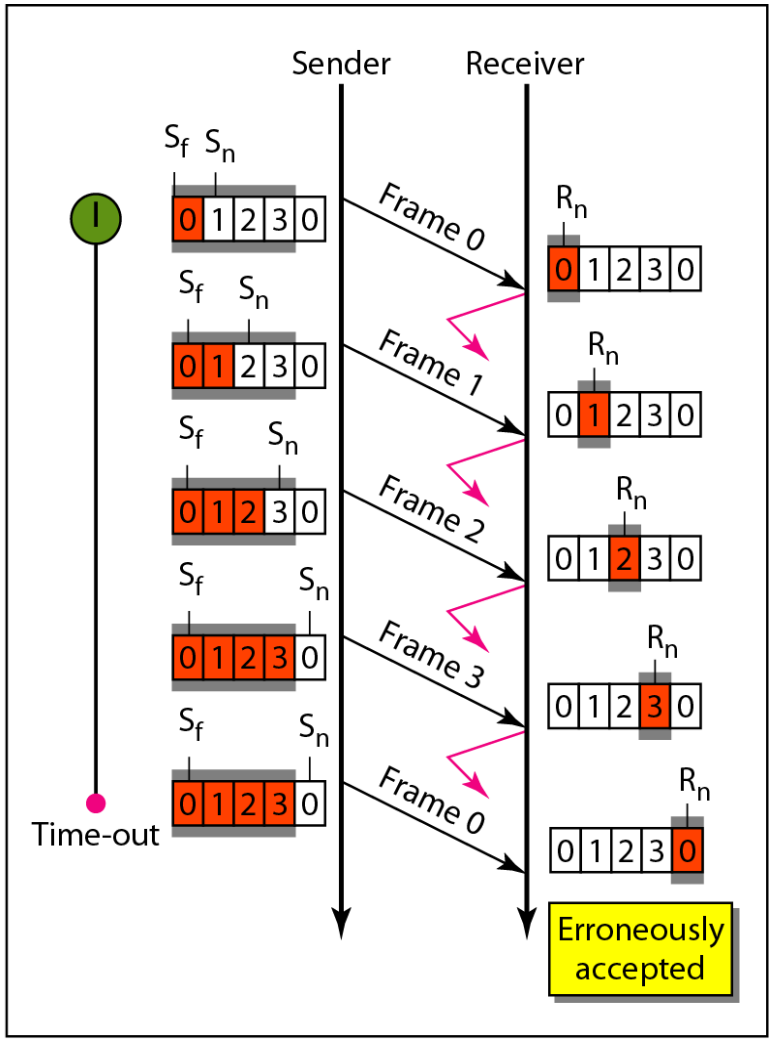
Go-Back-N receiver algorithm

```
 1  Rn = 0;
 2
 3  while (true)                          //Repeat forever
 4  {
 5    WaitForEvent();
 6
 7    if(Event(ArrivalNotification)) /Data frame arrives
 8    {
 9       Receive(Frame);
10       if(corrupted(Frame))
11            Sleep();
12       if(seqNo == Rn)                  //If expected frame
13       {
14         DeliverData();                 //Deliver data
15         Rn = Rn + 1;                   //Slide window
16         SendACK(Rn);
17       }
18    }
19  }
```

Window size for Go-Back-N ARQ



a. Window size < 2^m



b. Window size = 2^m

Multiple Access Protocols

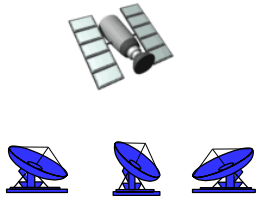
Multiple access links, protocols

two types of “links”:

- point-to-point
  - PPP for dial-up access
  - point-to-point link between Ethernet switch, host
- *broadcast (shared wire or medium)*
  - old-fashioned Ethernet
  - upstream HFC
  - 802.11 wireless LAN

shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)



shared RF (satellite)

humans at a cocktail party (shared air, acoustical)

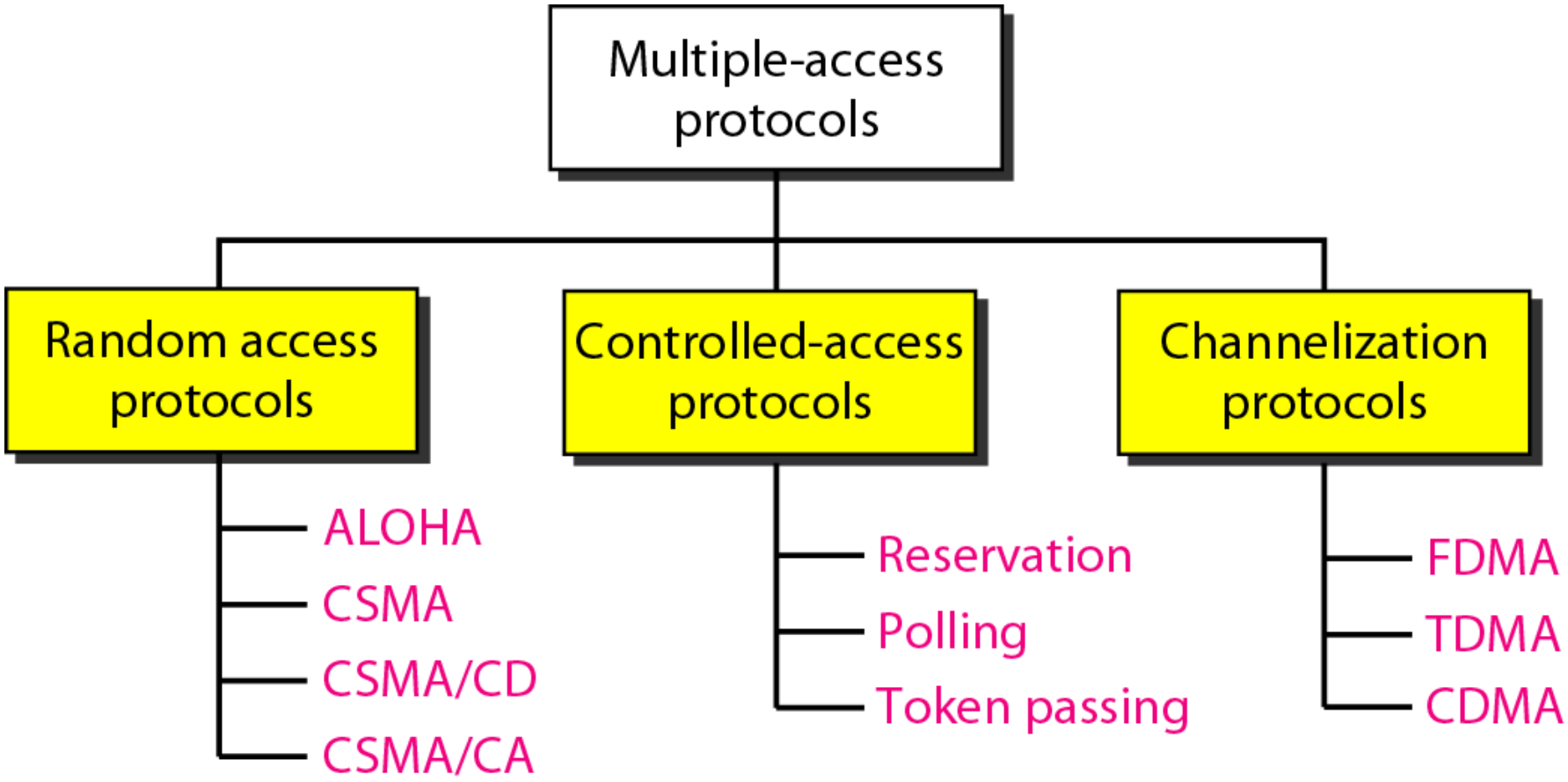
Multiple-access protocols

- When more than two nodes send at the same time , the transmitted frames collide.
- All collide frames are lost and the bandwidth of the broadcast channel will be wasted.
- We need multiple –access protocol to coordinate access to multipoint or broadcast link (Nodes or stations are connected to or use a common link)
- Multiple access protocols are needed in wire and wireless LANs and satellite networks

Problem of controlling the access to the medium is similar to the rules of speaking in an assembly :

- *Given everyone a chance to speak*
- *Don't speak until you are spoken to*
- *Don't monopolize the conversation*
- *Raise your hand if you have a question*
- *Don't interrupt when someone speaking*

Strategies for Media Access



RANDOM ACCESS

In random access or contention methods,

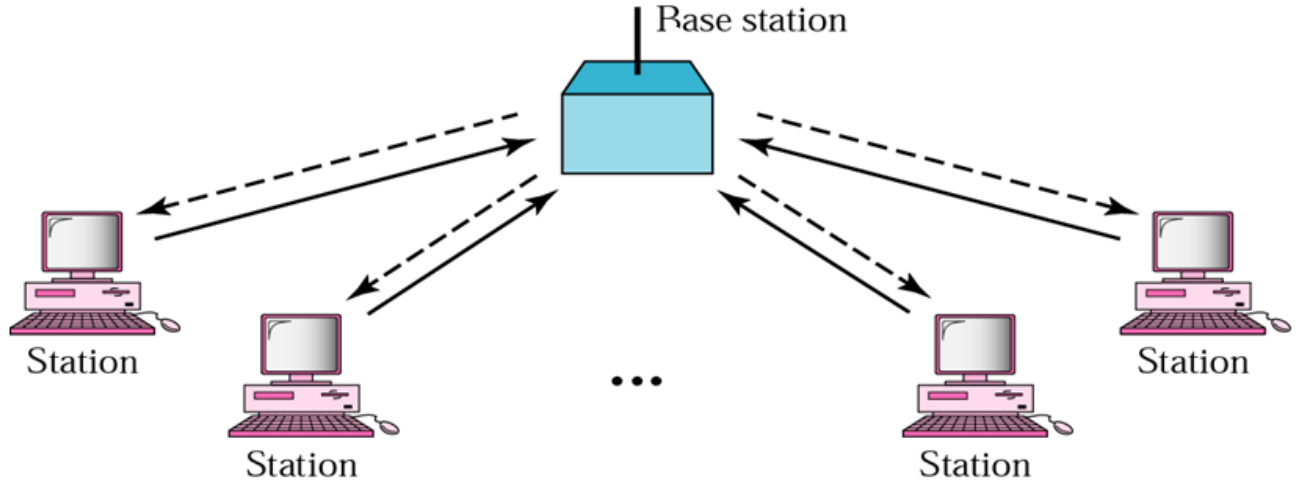
- **No station is superior to another station and none is assigned the control over another.**
- **No station permits , or does not permit, another station to send.**
- **A station that has data to send use a procedure defined by the protocol to make a decision on whether or not to send. This decision depends on the state of the medium(idle or busy)**

Random access Methodes:

- ALOHA: uses MA(Multiple Access) No carrier sense
- Carrier Sense Multiple Access ; CSMA
- Carrier Sense Multiple Access with Collision Detection (CSMA/CD)
- Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)

Pure ALOHA

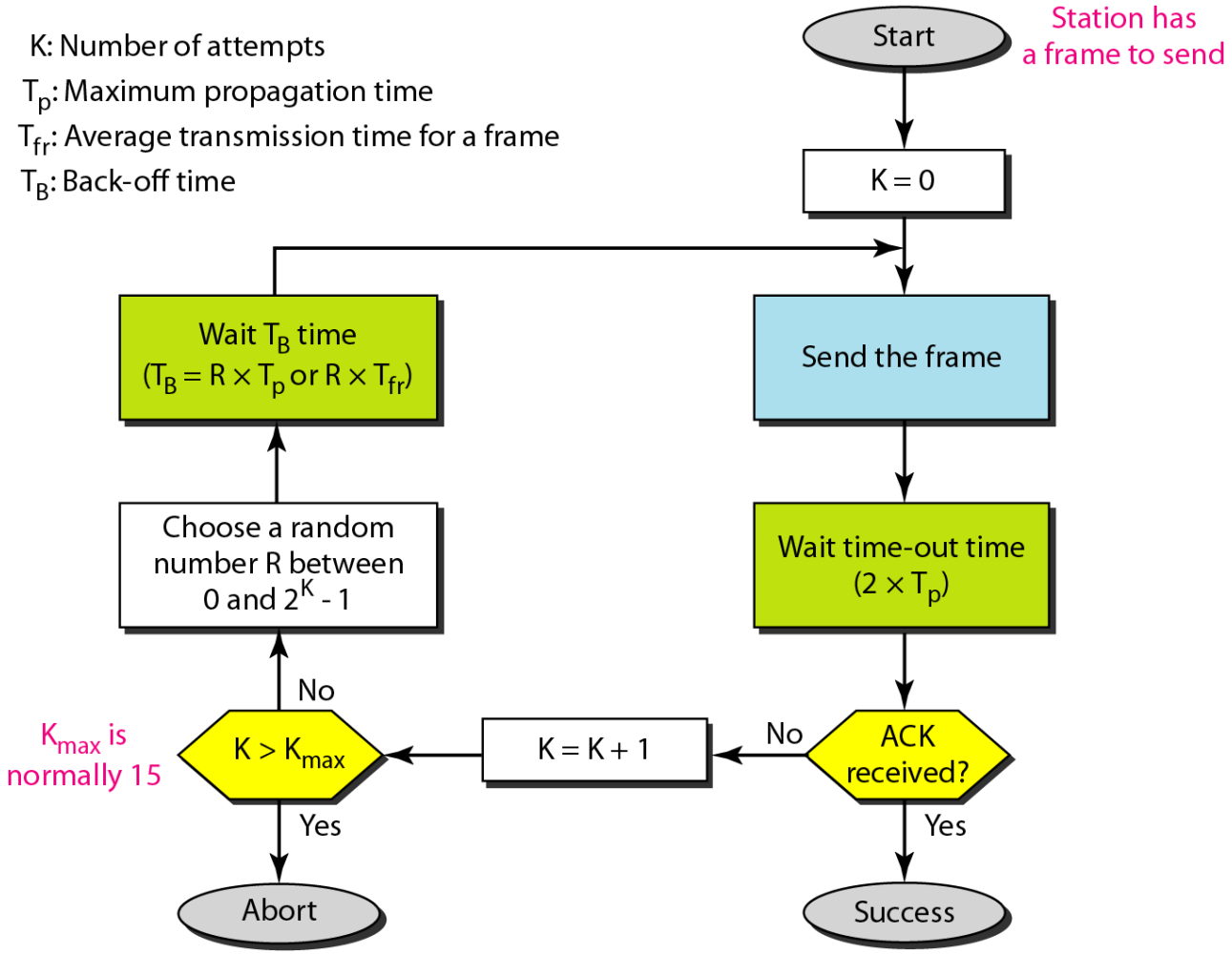
- *The earliest random access method developed at the Univ. of Hawaii in the early 1970s*
- *Designed for a radio (wireless) LAN*
- *Simple method. Each station sends a frame whenever it has a frame to send.*
- *Since there is only one channel to share, there is the possibility of collision between frames from different stations.*



Pure ALOHA

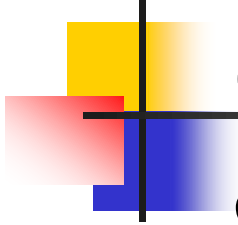
- *Each station sends a frame whenever it has a frame to send.*
- *It relies on acknowledgments from the receiver.*
- *If the ACK dose not arrive after a time-out period, the station resend the frame.*

Figure 12.4 Procedure for pure ALOHA protocol



Carrier sense Multiple Access(CSMA)

- ***CSMA**: listen before transmit. If channel is sensed busy, defer transmission*
- ***Persistent CSMA**: retry immediately when channel becomes idle (this may cause instability)*
- ***Non persistent CSMA**: retry after random interval*
- ***Note**: collisions may still exist, since two stations may sense the channel idle at the same time (or better, within a "vulnerable" window = round trip delay)*



Carrier Sense Multiple Access collision detection (CSMA/CD)

- ***CSMA/CD**: carrier sensing and deferral like in CSMA. But, collisions are detected within a few bit times.*
- *Transmission is then aborted, reducing the channel wastage considerably.*
- *Typically, **persistent** retransmission is implemented*
- *Collision detection is **easy in wired LANs** (eg, E-net): can measure signal strength on the line, or code violations, or compare tx and receive signals*
- *Collision detection **cannot be done in wireless LANs** (the receiver is shut off while transmitting, to avoid damaging it with excess power)*
- *CSMA/CD can approach channel utilization =1 in LANs (low ratio of propagation over packet transmission time)*

CSMA/CD (collision detection)

CSMA/CD: carrier sensing, deferral as in CSMA

- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage

- collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength
- human analogy: the polite conversationalist

Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission If NIC senses channel busy, waits until channel idle, then transmits
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame !
4. If NIC detects another transmission while transmitting, aborts and sends 48-bit jam signal
5. After aborting, NIC enters *exponential backoff:* after *m*th collision, NIC chooses *K* at random from $\{0,1,2,\ldots,2^m-1\}$. NIC waits $K \cdot 512$ bit times, returns to Step 2

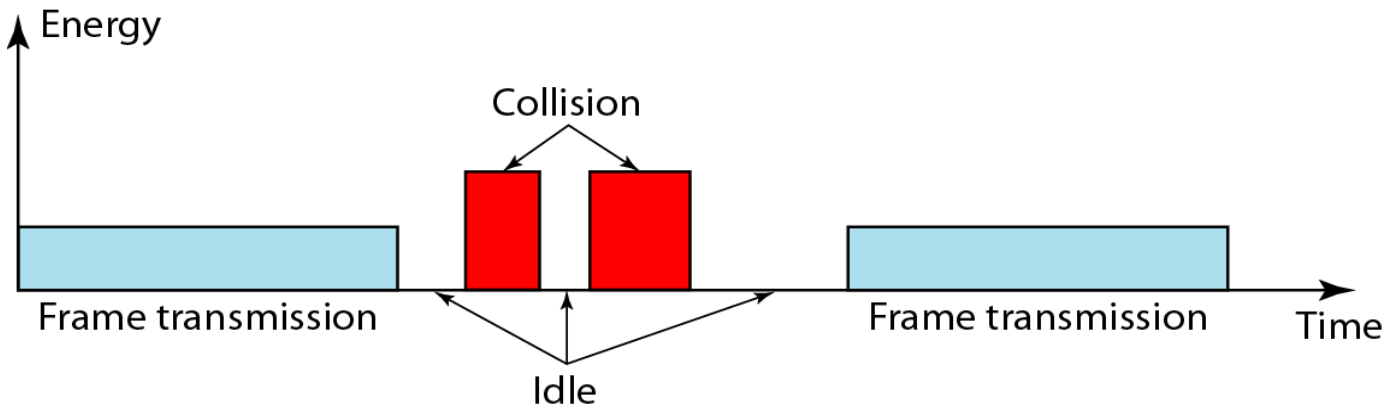
collision detection

How the station detects a collision?

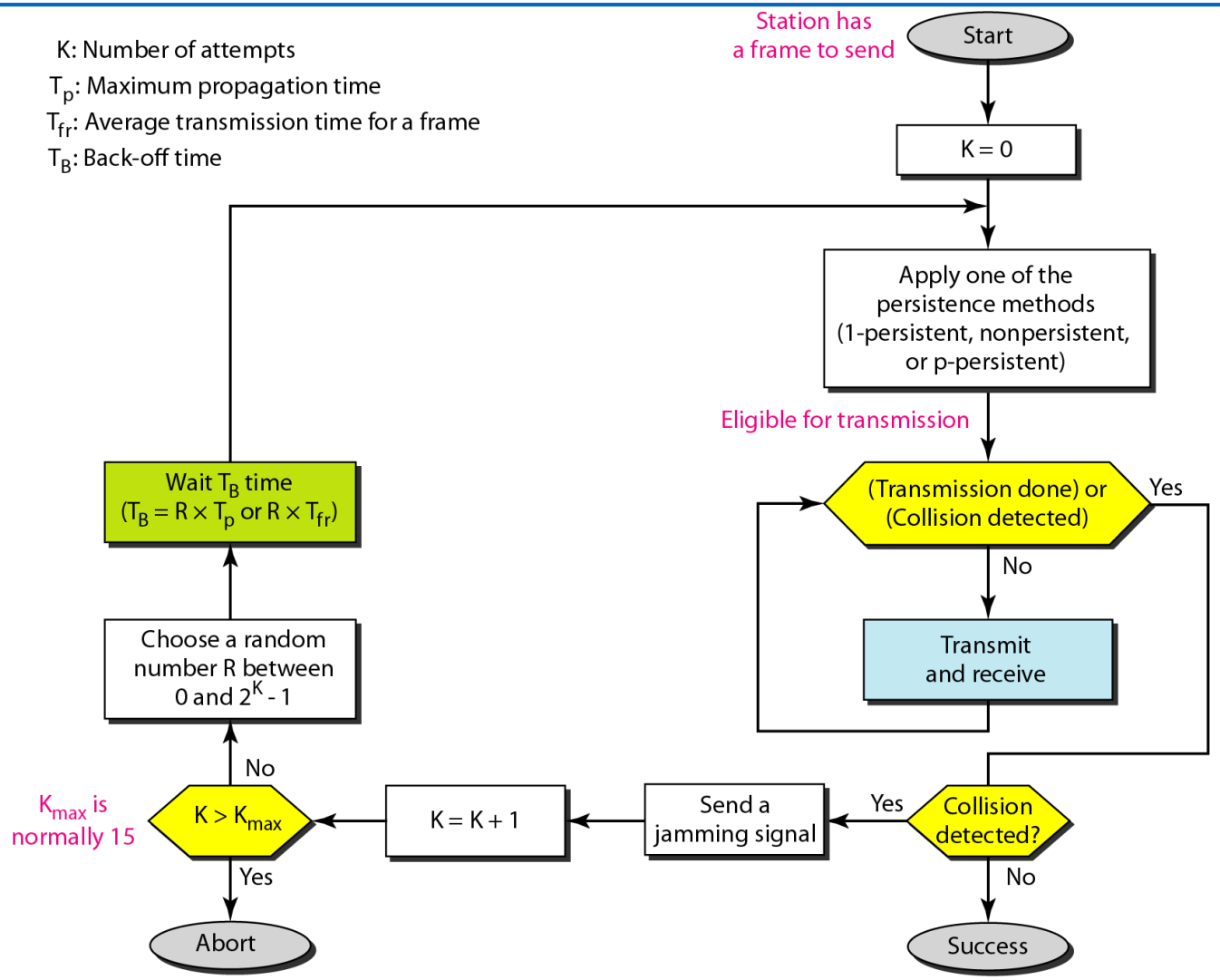
- Detecting voltage level on the line
- Detecting energy level .

Energy in channel can have three values: zero, normal, **and abnormal.**

- at zero level, the channel is idle
- At the normal level, a station has successfully captured the channel and is sending its frame.
- At the abnormal level, there is a collision and the level of the energy twice the normal level.



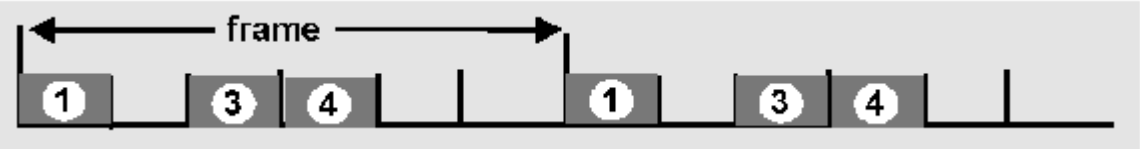
Flow diagram for the CSMA/CD



Channel Partitioning MAC protocols: TDMA

TDMA: time division multiple access

- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle

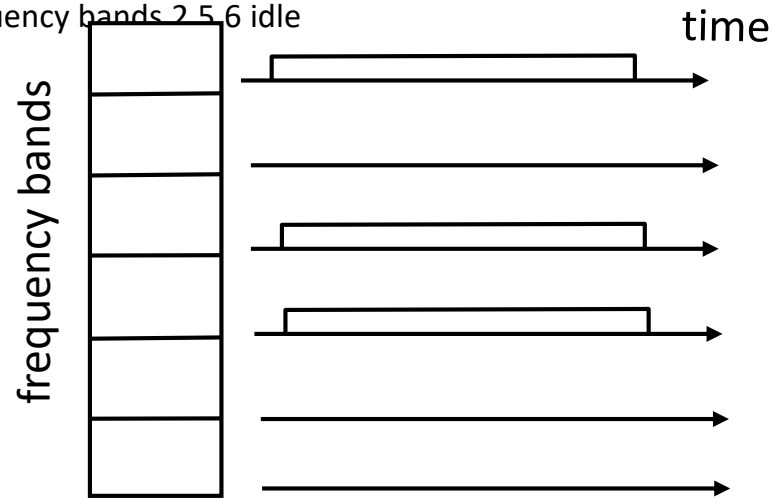


- TDM (Time Division Multiplexing): channel divided into N time slots, one per user; inefficient with low duty cycle users and at light load.
- FDM (Frequency Division Multiplexing): frequency subdivided.

Channel Partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



- TDM (Time Division Multiplexing): channel divided into N time slots, one per user; inefficient with low duty cycle users and at light load.
- FDM (Frequency Division Multiplexing): frequency subdivided.