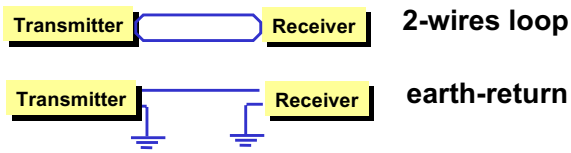
Data Transmission

Surasak Sanguanpong
nguan@ku.ac.th
http://www.cpe.ku.ac.th/~nguan
Last updated: May 19, 1999

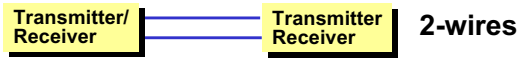
Applied Network Research Group Department of Computer Engineering, Kasetsart University

Communication mode

- **Simplex**



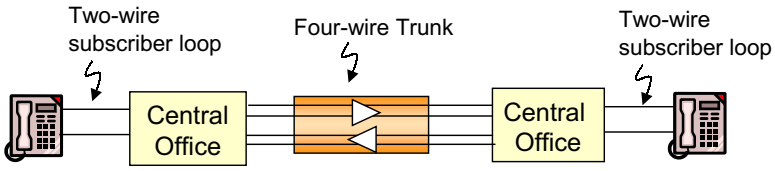
- **Half-duplex**



- **Full-duplex**

Simplex means communication runs in one direction. The examples include TV and radio broadcasting or pager. For half-duplex, both end devices can send and receive (they must alternate). Full-duplex uses two-way communications, but a device can send and receive simultaneously.

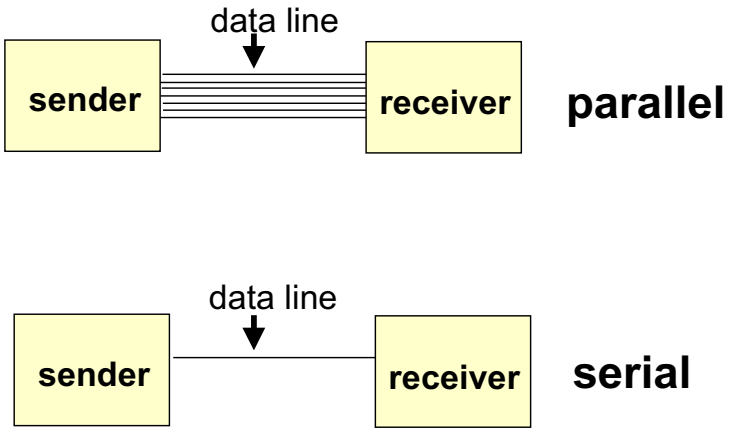
Two-Wire VS. Four-Wire Circuit



- Two-Wire :
  - share the same two lines for both transmitting and receiving
  - two-wire circuit is converted to four-wire circuit by the hybrid circuit
- Four-Wire :
  - physically seperated pairs of wires for transmitting and receiving

In two-wire circuits, the transmitted and received signals share the same two wires. The subscriber loop in telephone system make use of two-wire circuits. The PSTN trunks are typically four-wire circuits. The two-wire connection must be converted to a four-wire circuit for interfacing to a trunk circuit. The conversion is performed by the hybrid circuit.

Parallel v.s. Serial



A groups of bits is transmitted simultaneously in parallel transmission over a bundled of lines in a cable. Parallel transmissions are normally used where distance between the two devices are short.

Many problems found on parallel transmission over a long distance:

- Using multiples lines is expensive.
- Thicker wires are required to reduce signal degradation.
- Wires resistance may cause the bits drift and arrive at slightly different times.
- NEXT problems

Serial transmission uses just one line, transmits all the bits along it one after another. It is cheaper and more reliable than parallel transmission over a long distances.

Serial Transmission Mode

- **Asynchronous transmission**
  - avoid timing problem by not sending long stream of bits
  - data are transmitted one character at a time
  - synchronization is maintained within each character with start and stop codes
- **Synchronous transmission**
  - a block of bits is transmitted in a steady stream without start and stop codes
  - synchronization is maintain by a separate clock line or embedding clocking info in the signal

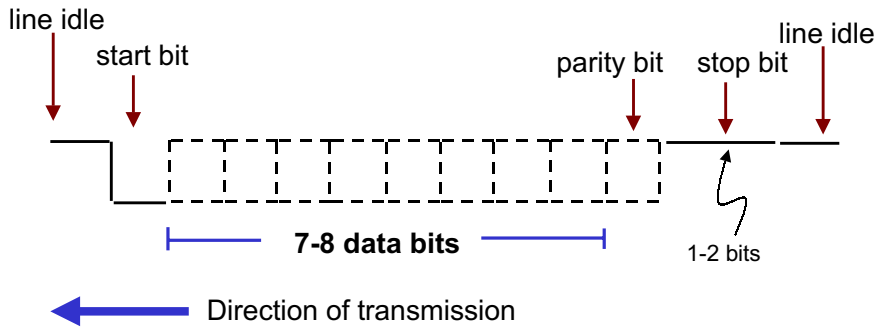
Two ways are used in synchronization : *Asynchronous* and *Synchronous* transmission.

Synchronization between transmitter and receiver is determined by whether the transmitter and receiver clock's are independent (asynchronous) or synchronized (synchronous).

With asynchronous transmission, it's not necessarily to send bit timing to the receiver. Receiver must produce a clock of its own to interpret each received bit. The sender sends group of bits at any time and the receiver never knows when they will arrive.

With synchronous transmission, the transmitter and the receiver must operate in synchronism using the same clock. The transmitter generates a clock which can be either transmitted to the receiver over a separate channel or be provided in the transmitted data.

Asynchronous Transmission



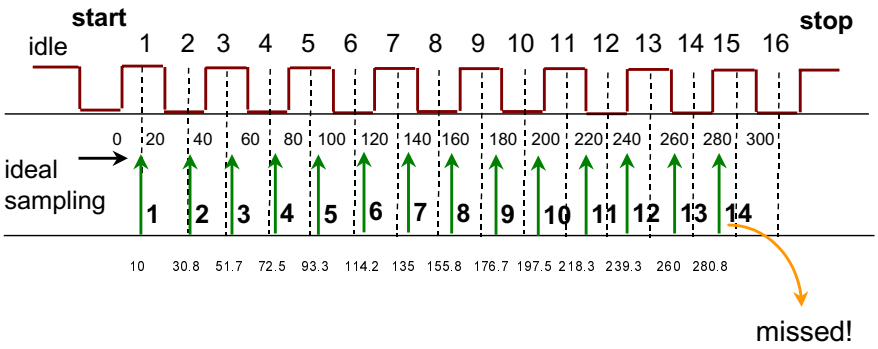
Bits are sent in a small group (usually byte) independently. When no data is transmitted, the line will be in an *idle state* which equivalent to binary 1.

The 7-8 bits that make up a character are encapsulated between *start bit* which a value of binary 0 and stop bits which is a binary 1.

The parity bit is followed the data and to be set by the transmitter such that the total number of ones in the character, including the parity bit is *even parity* or *odd parity* or *none* (no parity bit).

Stop bits may have a length of 1, 1.5, or 2 bits. Note that stop bits is the same as the idle, the transmitter will continue to transmit the stop bits until it is ready to send next character.

Bit missed



20 ms bit duration and 48Hz receiver clock

Clock independence in asynchronous transmission limited a number of transmitted data bit in each transmission.

Assumed that data is transmitted asynchronously using bits of 20 ms duration and the receiver clock is operating at 48 Hz.

Receiver clock period = 1/48Hz = 20.83 ms

It is clear that after the first bit is received, the receiver clock slips by approximately 0.83 ms on each successive bit. The problem therefore resolves into finding how many such slips are necessary to slide beyond half of one received bit period, or 10 ms :

10/0.83 = 12

Therefore the 12th clock pulse after the first one which was perfectly centered on bit 1, that is the 13th actual clock pulse, occur just on the trailing edge of the 13th bit. This means that bit 13 may or may not be successfully received. Hence the 14th bit is completely missed.

Synchronous Transmission :

- **But their clock is somehow be synchronized!**
  - provides a separate clock line, or
  - embedded clocking information in the digital signal
- **Need another level of synchronization**
  - flag to determine the beginning and end of the block, normally called preamble and postamble

Two categories of synchronous transmission :

- Byte-Oriented Protocols
- Bit-Oriented Protocols

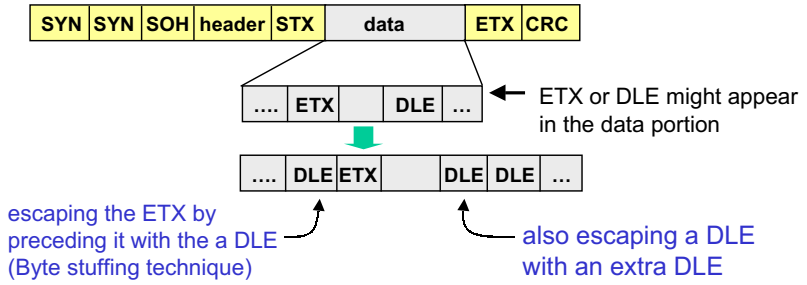
The sender and receiver must synchronize their clocks so the receiver knows where new byte begin. A special bit-transition pattern is embedded in the digital signal that provides the timing between sender and receiver.

Many encoding techniques might be used for embedding timing information such as bipolar encoding, or Manchester encoding.

Synchronous communication can be subdivided into two categories: byte-oriented protocols or bit-oriented protocol . Byte-Oriented protocols use ASCII characters such as **SYN**, **SOH** and **ETX** to control the transmission of data blocks. In Bit-Oriented protocols, data is transmitted as a steady stream of bits. A special flag 01111110 is used to delimited each frame.

Byte-Oriented protocol

- **View each frame as a collection of bytes**
  - exemplified by the BISYNC (Binary Synchronous Protocol) protocol developed by IBM in the late 1960s
  - also DDCMP used in DECNET, IMP-IMP in ARPANET
- **BISYNC example :**



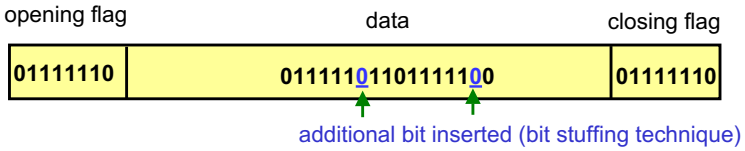
Byte-oriented transmission is used to send blocks of characters (frame) such as files of ASCII characters. Each block use a starting flag to achieve synchronization

Two or more control characters known as synchronous idle or **SYN** characters are inserted before each block of characters by the sender. These characters are used to synchronize a block of characters. The frame contents will be terminated with **ETX** character.

In BISYNC, when binary data is being transmitted, data transparency mechanism allows such data to be sent without being misinterpreted. This could be overcame by escaping the **ETX** character by preceding it with a **DLE** character whenever it appears in the body of a frame. If a **DLE** exists in the frame body, it is also escaped by preceding with an extra **DLE**

Bit-Oriented protocol

- **View each frame as a collection of bits**
  - SDLC (Synchronous Data Link Control Protocol) developed by IBM is an example
  - later standardized by the OSI as the HDLC (High-Level Data Link Control Protocol)
  - PPP is an example of HDLC variation



01111110 might occur in the the body of frame
any time five consecutive 1s have been sent,
a 0 is insert before sending the next bit

The block of data is treated as a sequence of bits. A special bit pattern signals the beginning of a block known as a opening flag (preamble). The same flag is also used as a closing flag (postamble).

To achieve data transparency, the flag is not allow to occur in the body of the frame. The transmitter circuit detects a sequence of five consecutive 1s, then automatically inserts an additional binary 0. The technique is known as bit stuffing. On the receiving side, should five consecutive 1s arrives, the receiver makes its decision based on the next incoming bit. :

- if the next bit is a 0, it must have been stuffed, and so the receiver removes it
- if the next bit is a 1, the receiver looks at the next bit :
  - if it sees 0, then it is the closing flag
  - if it sees 1, (01111111) then there must have been an error and the frame will be discarded.