

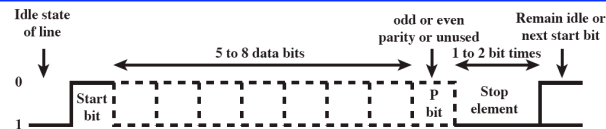
Asynchronous and Synchronous Transmission

- Timing problems require a mechanism to synchronize the transmitter and receiver
- Two solutions
 - Asynchronous
 - Synchronous

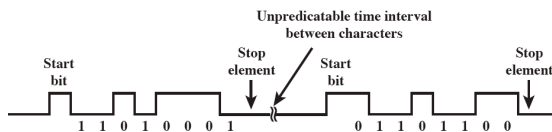
Asynchronous

- Data transmitted one character at a time
 - 5 to 8 bits
- Timing only needs maintaining within each character
- Resynchronize with each character

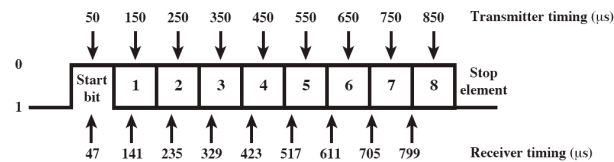
Asynchronous (diagram)



(a) Character format



(b) 8-bit asynchronous character stream



(c) Effect of timing error

CS420/520

Sequence 6

Asynchronous - Behavior

- In a steady stream, interval between characters is uniform
- In idle state, receiver looks for start bit
 - transition 1 to 0
- Next samples data bits
 - e.g. 7 intervals (char length)
- Then looks for next start bit...
 - Simple
 - Cheap
 - Overhead of 2 or 3 bits per char (~20%)
 - Good for data with large gaps (keyboard)

CS420/520 Axel Krings

Page 4

Sequence 6

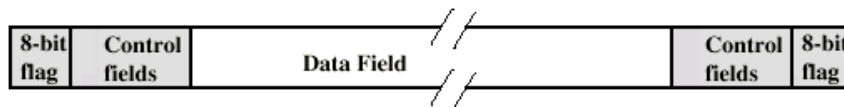
Synchronous - Bit Level

- Block of data transmitted without start or stop bits
- Clocks must be synchronized
- Can use separate clock line
 - Good over short distances
 - Subject to impairments
- Embed clock signal in data
 - Manchester encoding
 - Carrier frequency (analog)

Synchronous - Block Level

- Need to indicate start and end of block
- Use preamble and postamble
 - e.g. series of SYN (hex 16) characters
 - e.g. block of 11111111 patterns ending in 11111110
- More efficient (lower overhead) than async

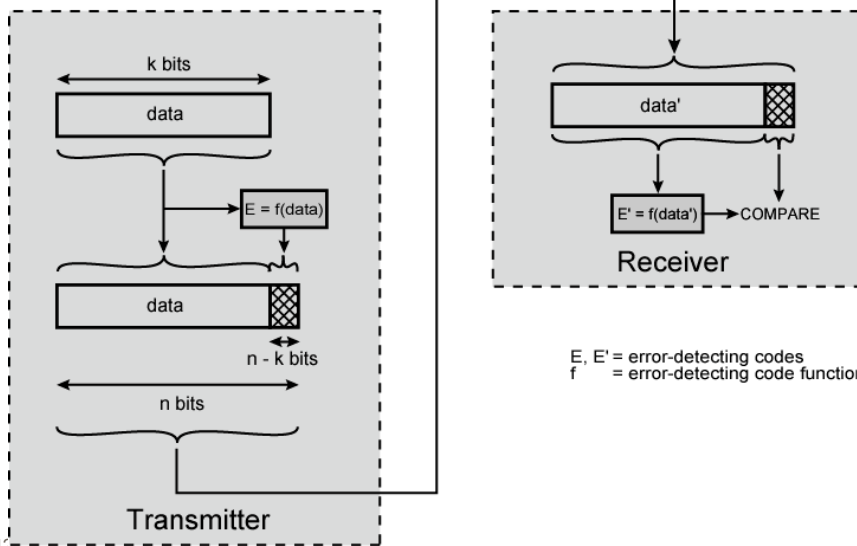
Synchronous (diagram)



Types of Error

- An error occurs when a bit is altered between transmission and reception
- Single bit errors
 - One bit altered
 - Adjacent bits not affected
- Burst errors
 - Length B
 - Contiguous sequence of B bits in which first, last and any number of intermediate bits are in error
 - Impulse noise
 - Fading in wireless
 - Effect is greater at higher data rates

Error Detection Process



Communication Techniques

- Error Control

- Concepts

- The concept behind error control is the prevention of delivery of incorrect messages (bits) to a higher level in the communication hierarchy.
 - The probability that one bit is in error is called the Bit Error Rate BER, e.g. $BER = 10^{-13}$

Communication Techniques

- There are two ways to manage Error Control
 - **Forward Error Control** - enough additional or redundant information is passed to the receiver, so it can not only detect, but also correct errors. This requires more information to be sent and has tradeoffs.
 - **Backward Error Control** - enough information is sent to allow the receiver to detect errors, but not correct them. Upon error detection, retransmitted may be requested.

Communication Techniques

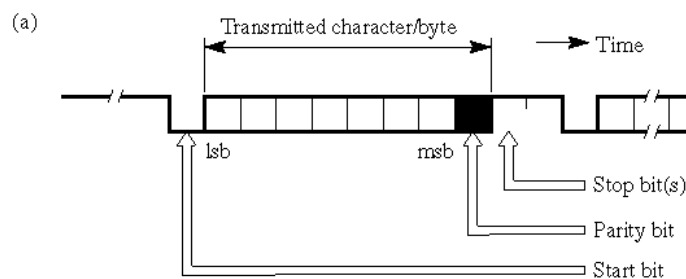
- code, code-word, binary code
- error detection, error correction
- Hamming distance
 - number of bits in which two words differ
- Widely used schemes
 - parity
 - check sum
 - cyclic redundancy check

Communication Techniques

- Parity

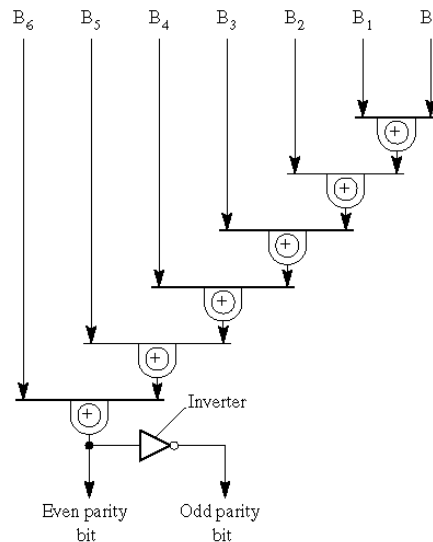
- extra bit at the end of a character (5-7 bits) specifying how many of the bits are 1's.
- The parity bit is said to be **even** if it is set to make the total number of 1's even, and **odd** if it is set to make the total number of 1's odd.
- Can detect all *odd* numbers of bit errors in the message.
- Typically used in asynchronous transmission, since timing and spacing between characters is uncertain.
- Requires one extra bit per characters (1/7 overhead)
- Can not** detect *even* numbers of bit errors -- error cancellation

Parity



Hal96 fig. 3.14

Parity



Hal96 fig. 3.14

Communication Techniques

- Combinatorial arguments

— Lets look at some probabilities associated with the detection of errors. We will classify three types of *received* messages based on their probabilities:

- *class 1*: P_1 - A frame arrives with no bit errors
- *class 2*: P_2 - A frame arrives with one or more undetected bit errors
- *class 3*: P_3 - A frame arrives with one or more detected bit errors and **no** undetected bit errors.

— In a simple system (no error detection), we only have Class 1 and 2 frames. If N_f is number of bits in a frame and P_B is BER for a bit then:

$$P_1 = (1 - P_B)^{N_f} \quad P_2 = 1 - P_1$$

Communication Techniques

— To calculate probabilities with error detection define:

- N_B - number of **B**its per character (including parity)
- N_C - number of **C**haracters per block
- N_F - number of bits per **F**rame = $N_B N_C$
- Notation: $\binom{N}{k}$ is read as “ N choose k ” which is the number of ways of choosing k items out of N .

$$\binom{N}{k} = \frac{N!}{k!(N-k)!}$$

— Note that the basic probability for P_1 does not change, and that P_3 is just what is left after P_1 and P_2

Communication Techniques

$$P_1 = (1 - P_B)^{N_B N_C}$$

$$P_B = \text{BER}$$

$$P_2 = \sum_{k=1}^{N_C} \binom{N_C}{k} \left[\sum_{j=2,4,\dots}^{N_B} \binom{N_B}{j} P_B^j (1 - P_B)^{N_B - j} \right]^k \left[(1 - P_B)^{N_B} \right]^{N_C - k}$$

$$P_3 = 1 - P_1 - P_2$$

Communication Techniques

- Parity Block Sum Check
 - As can be seen by this formula (as complex as it may appear), the probability of successfully detecting all errors that arrive is not very large.
 - All even numbers of errors are undetected
 - Errors often arrive in bursts so probability of multiple errors is not small
 - Can partially remedy situation by using a vertical parity check that calculates parity over the same bit of multiple characters. Used in conjunction with longitudinal parity check previously described.
 - Overhead is related to number of bits and can be large

Error Detection/Correction

- Cyclic Redundancy Checks (CRC)
 - Parity bits still subject to burst noise, uses large overhead (potentially) for improvement of 2-4 orders of magnitude in probability of detection.
 - CRC is based on a mathematical calculation performed on message. We will use the following terms:
 - M - message to be sent (k bits)
 - F - Frame check sequence (FCS) to be appended to message (n bits)
 - T - Transmitted message includes both M and $F \Rightarrow (k+n)$ bits)
 - G - a $n+1$ bit pattern (called generator) used to calculate F and check T

Error Detection/Correction

- Idea behind CRC
 - given k -bit frame (message)
 - transmitter generates n -bit sequence called frame check sequence (FCS)
 - so that resulting frame of size $k+n$ is exactly divisible by some predetermined number
- Multiply M by 2^n to shift, and add F to padded 0s

$$T = 2^n M + F$$

Error Detection/Correction

- Dividing $2^n M$ by G gives quotient and remainder

$$\frac{2^n M}{G} = Q + \frac{R}{G}$$

then using R as our FCS we get

remainder
is 1 bit less
than divisor

$$T = 2^n M + R$$

on the receiving end, division by G leads to

$$\frac{T}{G} = \frac{2^n M + R}{G} = Q + \frac{R}{G} + \frac{R}{G} = Q$$

Note:
mod 2 addition,
no remainder

Error Detection/Correction

- Therefore, if the remainder of dividing the incoming signal by the generator G is zero, no transmission error occurred.
- Assume T + E was received

$$\frac{T + E}{G} = \frac{T}{G} + \frac{E}{G}$$

since T/G does not produce a remainder, an error is detected only if E/G produces one

Error Detection/Correction

- example, assume G(X) has at least 3 terms
 - G(x) has 3 1-bits
 - detects all single bit errors
 - detects all double bit errors
 - detects odd #'s of errors if G(X) contains the factor (X + 1)
 - any burst errors < or = to the length of FCS
 - most larger burst errors
 - it has been shown that if all error patterns likely, then the likelihood of a long burst not being detected is $1/2^n$

Error Detection/Correction

- What does all of this mean?
 - A polynomial view:
 - View CRC process with all values expressed as polynomials in a dummy variable X with binary coefficients, where the coefficients correspond to the bits in the number.
 - $M = 110011$, $M(X) = X^5 + X^4 + X + 1$, and for $G = 11001$ we have $G(X) = X^4 + X^3 + 1$
 - Math is still mod 2
 - An error $E(X)$ is received, and undetected iff it is divisible by $G(X)$

Error Detection/Correction

—Common CRCs

- CRC-12 = $X^{12} + X^{11} + X^3 + X^2 + X + 1$
 - CRC-16 = $X^{16} + X^{15} + X^2 + 1$
 - CRC-CCITT = $X^{16} + X^{12} + X^5 + 1$
 - CRC-32 = $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Hardware Implementation:

$$G(X) = 1 + a_1 X + a_2 X^2 + \dots + a_{n-1} X^{n-1} + a_n X^n$$

