Chapter 6 – Digital Data Communication Techniques

CSE 3213
Fall 2011

10/2/2011 1:15 PM

1

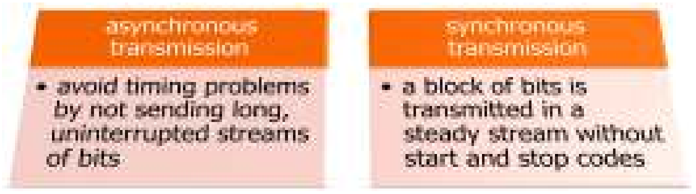
Asynchronous and Synchronous Transmission

- Receiver samples the medium at the center of each bit time.
- Transmitter's and receiver's clocks may not be precisely aligned.
- Example (discussed in class)

2

Asynchronous and Synchronous Transmission (2)

- timing problems require a mechanism to synchronize the transmitter and receiver
  - —receiver samples stream at bit intervals
  - —if clocks are not precisely aligned, drifting will sample at wrong time after sufficient bits are sent
- two solutions to synchronizing clocks:



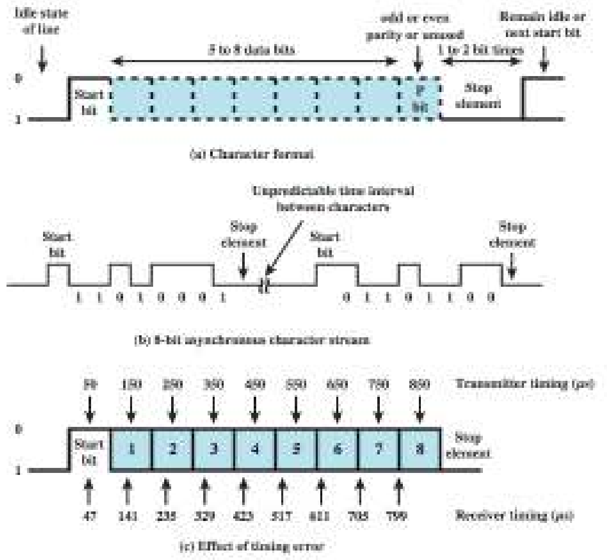
asynchronous transmission
- avoid timing problems by not sending long, uninterrupted streams of bits

synchronous transmission
- a block of bits is transmitted in a steady stream without start and stop codes

Asynchronous Transmission

- ➢ data are transmitted one character at a time
  - ➢ each character is 5 to 8 bits in length
  - ➢ receiver has the opportunity to resynchronize at the beginning of each new character
- simple and cheap
  - ➢ requires overhead of 2 or 3 bits per character (~20%)
- the larger the block of bits, the greater the cumulative timing error
- good for data with large gaps (keyboard)

Asynchronous Transmission



(a) Character format

(b) 8-bit asynchronous character stream

(c) Effect of timing error

Asynchronous Trx Behavior

- Idle state = binary 1
- In idle state, receiver looks for transition from 1 to 0
- Start bit = binary 0
- Then samples next 5 – 8 intervals (character length)
- Stop element = binary 1 (min length = 1, 1.5 or 2 bits)
  No maximum length specified for stop element (why?)
- Then looks for next 1 to 0 for next char

6

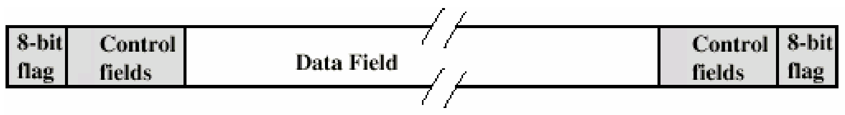
Synchronous Trx - Bit Level

- 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 (digital signals)
  - —Carrier frequency, phase (analog signals)

7

Synchronous Trx - 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 asynchronous transmission



8

Types of Error

- an error occurs when a bit is altered between transmission and reception
  - —binary 1 is transmitted and binary 0 is received or binary 0 is transmitted and binary 1 is received

single bit errors

- isolated error that alters one bit but not nearby bits
- caused by white noise

burst errors

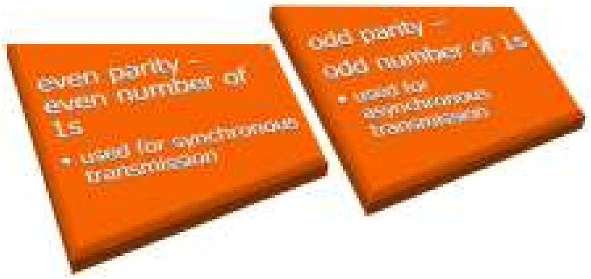
- contiguous sequence of *B* bits where first and last bits and any number of intermediate bits are received in error
- caused by impulse noise or by fading in wireless
- effects greater at higher data rates

Error Detection

- regardless of design there will be errors
- can detect errors by using an error-detecting code added by the transmitter
  - code is also referred to as *check bits*
- recalculated and checked by receiver
- there is still chance of undetected error

Parity Check

- the simplest error detecting scheme is to append a parity bit to the end of a block of data

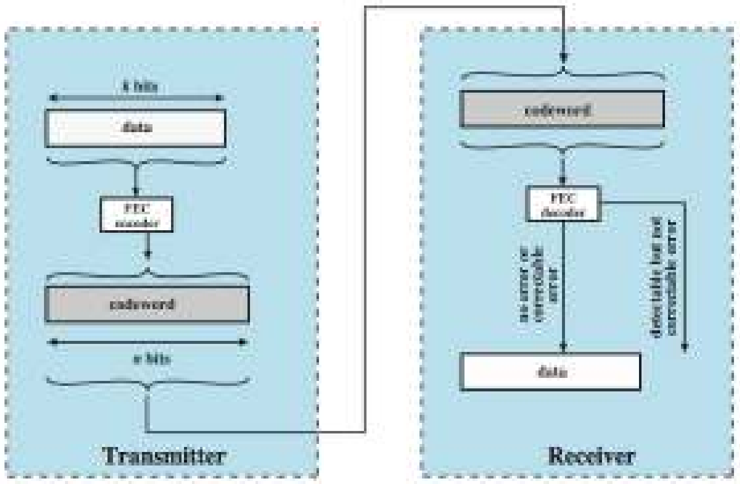


even parity – even number of 1s
- used for synchronous transmission

odd parity – odd number of 1s
- used for asynchronous transmission

➢ if any even number of bits are inverted due to error, an undetected error occurs

Error Detection Process



Cyclic Redundancy Check (CRC)

- One of the most common and powerful error-detecting codes.
- Given *k* bits of data,
  generate a sequence F of *j* bits (FCS)
  using a predetermined divisor P of (*j*+1) bits
- Transmit a frame of *k+j* bits (data + FCS) which will be exactly divisible by divisor P
- Receiver divides frame by divisor P
  - —If no remainder, assume no error

13

CRC (continued)

- Simpler but equivalent method: receiver repeats the steps the sender did.
  - —If getting the same FCS, assume no error.

3 equivalent procedures:

- Modulo-2 arithmetic
- Polynomials
- Digital logic (not covered)

Examples (discussed in class)

14

Error Correction

- Correction of detected errors usually requires whole data block to be retransmitted (chap. 7)
- But not appropriate for wireless applications
  - —Bit error rate is high
    - Lots of retransmissions
  - —Propagation delay can be long (satellite) compared with frame transmission time
    - Would result in retransmission of frame in error plus many subsequent frames
- Need to correct errors on basis of bits received

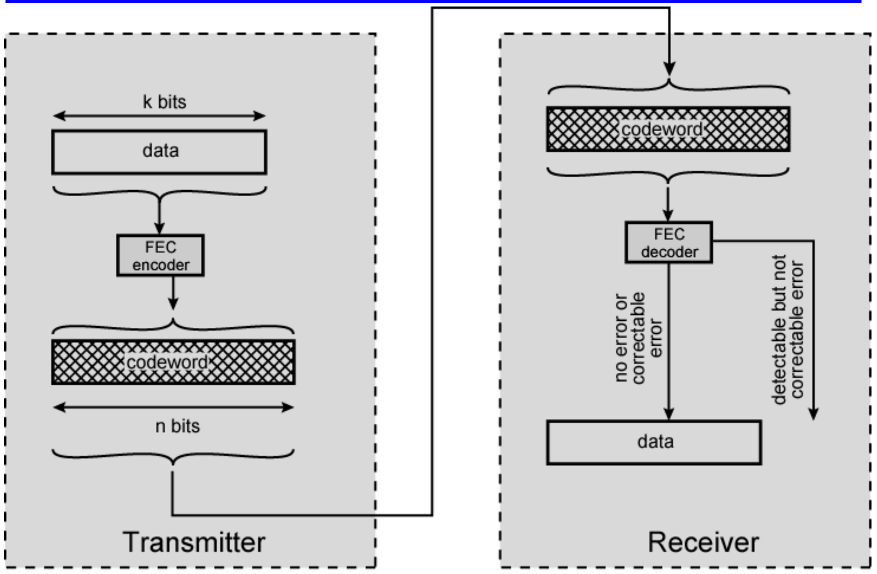
15

How Error Correction Works

- Add redundancy to transmitted message
- Can deduce original in face of certain level of error rate
- Example: block error correction code
  - —In general, add $(n - k)$ bits to end of block
    - Gives n bit block (codeword)
    - All of original k bits included in codeword
  - —Some FEC map k bit input onto n bit codeword such that original k bits do not appear

16

Error Correction Process Diagram



Error Correction Process

- Each *k* bit block mapped to an *n* bit block (*n*>*k*)
  - — Codeword
  - — Forward error correction (FEC) encoder
- Codeword sent
- Received bit string similar to transmitted but may contain errors
- Received code word passed to FEC decoder
  - — If no errors, original data block output
  - — Some error patterns can be detected and corrected
  - — Some error patterns can be detected but not corrected
  - — Some (rare) error patterns are not detected
    - Results in incorrect data output from FEC

18

Design Considerations for Block Code

- For given values of *n* and *k*, want the largest possible value of d_{min}
- To increase d_{min} increase the number of extra bits.
- Reduce the number of extra bits to reduce bandwidth needed
- Easy to encode/decode, minimal overheads (memory, time)

Reading

- Chapter 6, Stallings' book
- Homework: Solve problems 6.13 and 6.15 in the textbook.