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
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)
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
Figure 6.1  Burst and Single-Bit Errors
Error Detection

- Regardless of design you will have errors, resulting in the change of one or more bits in a transmitted frame
- Frames
  - Data transmitted as one or more contiguous sequences of bits

\[ P_b \]
- Probability that a bit is received in error; also known as the bit error rate (BER)

\[ P_1 \]
- Probability that a frame arrives with no bit errors

\[ P_2 \]
- Probability that, with an error-detecting algorithm in use, a frame arrives with one or more undetected errors

\[ P_3 \]
- Probability that, with an error-detecting algorithm in use, a frame arrives with one or more detected bit errors but no undetected bit errors

- The probability that a frame arrives with no bit errors decreases when the probability of a single bit error increases
- The probability that a frame arrives with no bit errors decreases with increasing frame length
  - The longer the frame, the more bits it has and the higher the probability that one of these is in error
$E = f(data)$

$E' = f(data')$

$n - k$ bits

$k$ bits

$n$ bits

$E, E' = $ error-detecting codes

$f = $ error-detecting code function

**Figure 6.2 Error Detection Process**
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.
Error Detection/Correction

• Error Correction
  — What is needed for error correction?
    • Ability to detect that bits are in error
    • Ability to detect which bits are in error
  — Techniques include:
    • Parity block sum checking which can correct a single bit error
    • Hamming encoding which can detect multiple bit errors and correct less (example has hamming distance of 3 can detect up to 2 errors and correct 1)
      - 00000  00111  11100  11011
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
The simplest error detecting scheme is to append a parity bit to the end of a block of data. If any even number of bits are inverted due to error, an undetected error occurs.
Parity

Hal96 fig. 3.14
Parity

Hal96 fig. 3.14
Communication Techniques

• Combinatorial arguments
  — Probabilities associated with the detection of errors.
    • \( P_1 \) = prob. that a frame arrives with no bit errors
    • \( P_2 \) = prob. that, with an error-detection algo. in use, a frame arrives with one or more undetected bit errors
    • \( P_3 \) = prob. that, with an error-detection algo. in use, 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 bits per character (including parity)
  • \( N_C \) - number of characters per block
  • \( N_F \) - number of bits per frame = \( N_B N_C \)

  • 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

\[ P_1 = (1 - P_B)^{N_B N_C} \]

\[ P_B = \text{BER} \]

\[ P_2 = \sum_{k=1}^{N_C} \left( \begin{array}{c} N_C \\ k \end{array} \right) \left[ \sum_{j=2,4,...}^{N_B} \left( \begin{array}{c} N_B \\ j \end{array} \right) P_B^j (1 - P_B)^{(N_B - j)} \right]^k \left[ (1 - P_B)^{N_B} \right]^{N_C - k} \]

\[ P_3 = 1 - P_1 - P_2 \]
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
Figure 6.3 A Two-Dimensional Even Parity Scheme

(a) Parity calculation

| 0 1 1 1 0 | 1 |
| 0 1 1 1 0 | 1 |
| 0 1 0 0 0 | 1 |
| 0 1 0 1 1 | 1 |
| 0 0 0 1 0 | 0 |

(b) No errors

| 0 1 1 1 1 1 0 | 1 |
| 0 1 1 1 1 | 1 |
| 0 0 1 1 0 | 1 |
| 0 0 0 0 0 | 0 |
| 1 0 1 1 1 | 0 |
| 1 1 0 0 1 | 0 |

(c) Correctable single-bit error

| 0 1 1 1 1 1 0 | 1 |
| 0 1 1 1 1 | 1 |
| 0 0 1 1 0 | 1 |
| 0 0 0 0 0 | 0 |
| 1 0 1 1 1 | 0 |
| 1 1 0 0 1 | 0 |

(d) Uncorrectable error pattern
The Internet Checksum

- Error detecting code used in many Internet standard protocols, including IP, TCP, and UDP
- Ones-complement operation
  - Replace 0 digits with 1 digits and 1 digits with 0 digits
- Ones-complement addition
  - The two numbers are treated as unsigned binary integers and added
  - If there is a carry out of the leftmost bit, add 1 to the sum (end-around carry)
10 octet header – last 2 octets are checksum

00 01 F2 03 F4 F5 F6 F7 00 00

<table>
<thead>
<tr>
<th>Partial sum</th>
<th>0001 F203 F204</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial sum</td>
<td>F204 F4F5 1E6F9</td>
</tr>
<tr>
<td>Carry</td>
<td>E6F9 1 E6FA</td>
</tr>
<tr>
<td>Partial sum</td>
<td>E6FA F6F7 1DDF1</td>
</tr>
<tr>
<td>Carry</td>
<td>DDF1 1 DDF2</td>
</tr>
<tr>
<td>Ones compleent of the result</td>
<td>220D</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Partial sum</th>
<th>0001 F203 F204</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial sum</td>
<td>F204 F4F5 1E6F9</td>
</tr>
<tr>
<td>Carry</td>
<td>E6F9 1 E6FA</td>
</tr>
<tr>
<td>Partial sum</td>
<td>E6FA F6F7 1DDF1</td>
</tr>
<tr>
<td>Carry</td>
<td>DDF1 1 DDF2</td>
</tr>
<tr>
<td>Partial sum</td>
<td>DDF2 220D FFFF</td>
</tr>
</tbody>
</table>

(a) Checksum calculation by sender  
(b) Checksum verification by receiver

Figure 6.4  Example of Internet Checksum
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 $F = (k+n$ bits)
    • $G$ – is 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 $2^nM$ by $G$ gives quotient and remainder

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

then using $R$ as our FCS we get

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

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

remainder is 1 bit less than divisor
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 $< \text{or} = \text{to the length of FCS}$
    • 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

Figure 6.6 General CRC Architecture to Implement Divisor
\[(1 + A_1X + A_2X^2 + \ldots + A_{n-1}X^{n-k-1} + X^{n-k})\]
Hardware Implementation

Same thing, just another way of arranging it:

\[ G(X) = a_n X^n + a_{n-1} X^{n-1} + \ldots + a_2 X^2 + a_1 X + 1 \]

Note that the “+” in the shift register relates to mod-2 addition, i.e., XOR. The “x” here implies multiplication, i.e., if the term \( a_i \) is 1, the feedback loop is enabled, otherwise it is disconnected.
Forward Error Correction

• Correction of detected errors usually requires data blocks to be retransmitted

• Not appropriate for wireless applications:
  — The bit error rate (BER) on a wireless link can be quite high, which would result in a large number of retransmissions
  — Propagation delay is very long compared to the transmission time of a single frame

• Need to correct errors on basis of bits received

---

**Codeword**

- On the transmission end each \( k \)-bit block of data is mapped into an \( n \)-bit block (\( n > k \)) using a **forward error correction (FEC)** encoder
Figure 6.8 Error Correction Process
Block Code Principles

• Hamming distance
  — $d(v_1, v_2)$ between two $n$-bit binary sequences $v_1$ and $v_2$ is the number of bits in which $v_1$ and $v_2$ disagree
  — See example on page 203 in the textbook

• Redundancy of the code
  — The ratio of redundant bits to data bits $(n-k)/k$

• Code rate
  — The ratio of data bits to total bits $k/n$
  — Is a measure of how much additional bandwidth is required to carry data at the same data rate as without the code
  — See example on page 205 in the textbook
Figure 6.9  How Coding Improves System Performance
About limitations

• Do any of the above approaches protect from malicious manipulations?
  — Party (serial link):
    • e.g., flip 2 bits and ...
  — Internet Checksum (TCP/UDP/IP):
    • e.g., insert zeros, or insert multiple errors which sum to zero and ...
  — CRC (Ethernet):
    • add multiples of generator and ...
  — So, if you if you are worried about data integrity, use a signature, e.g., MD5, SHA-1or2
Summary

• Types of errors
• Error detection
• Parity check
  — Parity bit
  — Two-dimensional parity check
• Internet checksum
• Cyclic redundancy check
  — Modulo 2 arithmetic
  — Polynomials
  — Digital logic
• Forward error correction
  — Block code principles