

Reliable Data Transfer between Two Machines

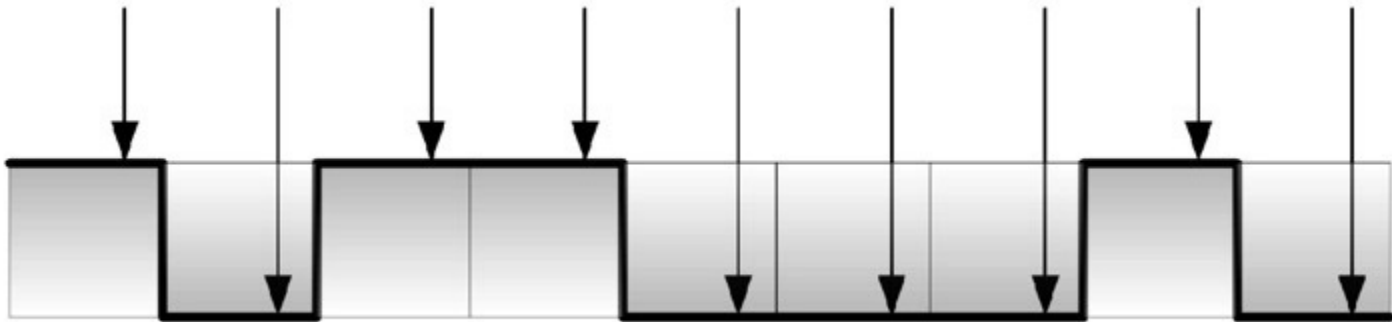
Issues

- Synchronization
 - How does the receiver know when the sender is sending data?
- Encoding
 - Representing data with signals
- Error detection
 - How to detect if what is received is same as what is sent
- Error control
 - How to ensure the receiver gets the correct data
- Flow control
 - How to ensure the sender does not swamp the receiver (fast sender, slow receiver)

Synchronization

Need for synchronization

- Basic steps in transmitting digital data
 - Transmitter sends a stream of 0 or 1 bits
 - Receiver samples incoming signal **once per bit time** to see if it is a 0 or 1
 - Typically at the center of the bit



- Transmitter sends one bit (0 or 1) every 1 millisecond (say)
 - Say there is a clock that ticks every 1 millisecond
 - Transmitter puts a 1 or 0 on the line at each tick of the clock
 - bit 1 on tick 1, bit 2 on tick 2, ...
- What should the receiver do to get the pattern correctly?
 - Need to know when to start counting (when will bit 1 arrive)
 - A-priori agreed or sender has to tell (how?)
 - Need to know when to look for the next successive bits (duration of one bit)

Framing

- Break up the bit pattern into multiple frames
- Issues
 - Needs identifier for each frame to distinguish between frames
 - Needs technique to identify where a frame ends and the next one begins
 - Frame synchronization problem

Frame Synchronization

- When data is transferred from the transmitter to the receiver, unless steps are taken to provide synchronization, the receiver may start interpreting the data erroneously
- Two common approaches:
 - Asynchronous Transmission
 - Synchronous Transmission

Asynchronous Transmission

- Transmitter and Receiver has separate clocks
- Data rate and frame format negotiated a-priori
- Data are transmitted one character at a time (5-8 bits)
 - Timing or synchronization must only be maintained within each character
 - Start bit at the beginning of each new character to resynchronize (start the receiver clock at the right time)
- There may be gaps between two successive character
- When no character is being transmitted, the line between transmitter and receiver is in an *idle* state
 - Start, stop, and parity bits.

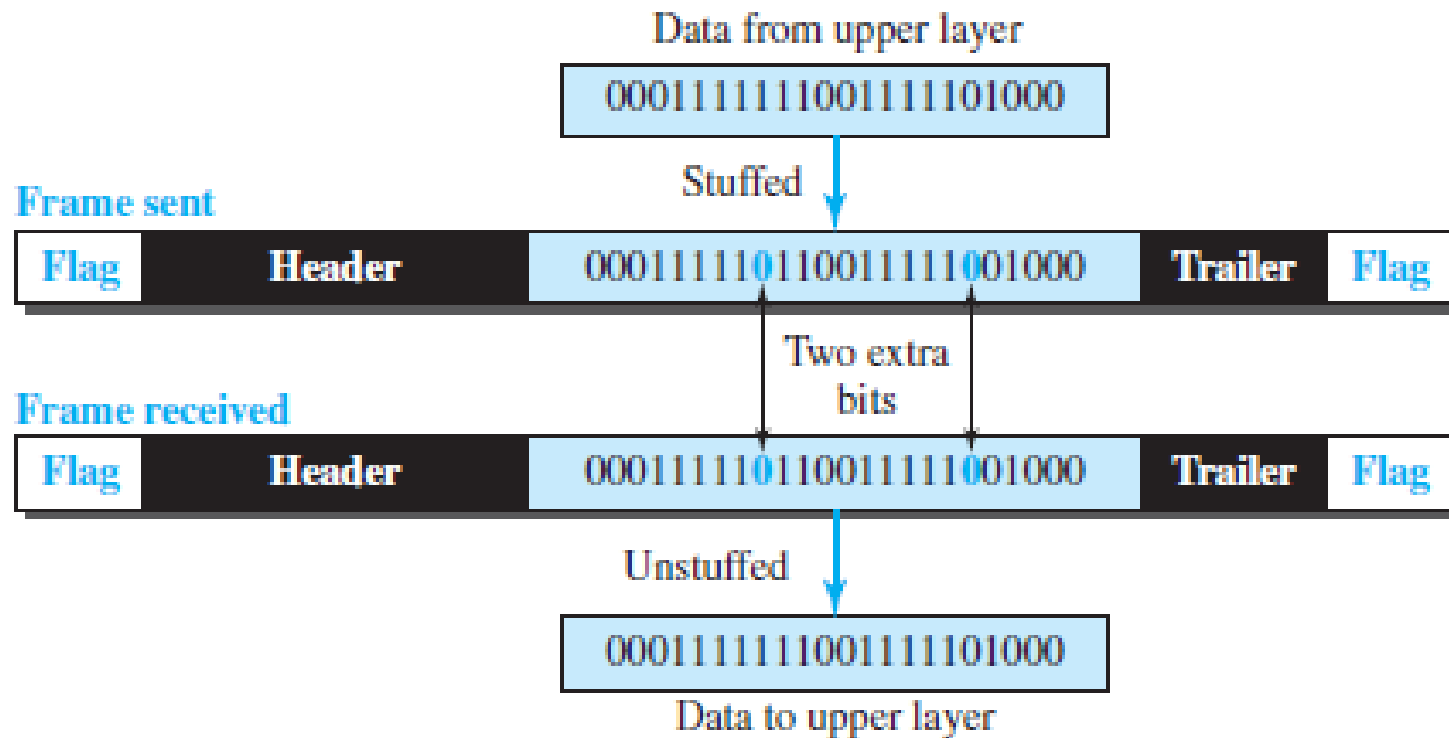
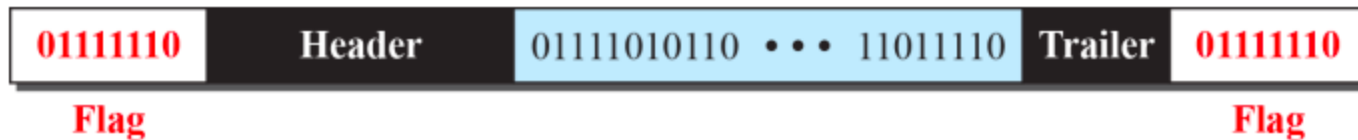


Synchronous Transmission

- Receiver clock is synchronized with transmitter clock
- A block of bits is transmitted in a steady stream without start and stop codes
 - The block may be arbitrarily long
 - To prevent timing drift between transmitter and receiver, clock signal is embedded in the data signal or separate clock signal sent
- There is another level of synchronization required, so as to allow the receiver to determine the beginning and end of a block of data
 - Every block begins with a **preamble** bit pattern, and generally ends with a **postamble** bit pattern

Synchronous Transmission (contd.)

- A typical synchronous frame format:
 - 8-bit flag (preamble)
 - Control fields
 - Data field
 - Control fields
 - 8-bit flag (postamble)
- Special inter-frame fill pattern is sent when no frame to send to maintain synchronization
- What if data itself contains the preamble/postamble pattern?
 - Use [bit-stuffing](#)



- For sizable blocks of data, synchronous transmission is far more efficient than asynchronous mode
 - Asynchronous transmission requires 20% or more overhead
 - The control information, preamble and postamble in synchronous transmission are typically less than 100 bits

Encoding Techniques

Data Encoding Techniques

- Data is transmitted by propagation and processing of signals
- Encoding – specifying how data is represented by signals
 - Digital data over digital signal
 - Digital data over analog signal

Encoding Digital Data with Digital Signals

- Digital signal
 - Uses discrete, discontinuous, voltage pulses
 - Each pulse is a signal element
 - Binary data is encoded into signal elements
- Data
 - Bit string of 0's and 1's
 - Each bit is present for a duration T (**bit interval**)
 - Data rate = $1 / T$ bps

Some Terminology

- Unipolar
 - All signal elements have same sign
- Polar
 - Two logic states represented by +ve/-ve voltage
- Duration or length of a bit
 - Time taken for transmitter to emit the bit
- Modulation/Signaling rate
 - Rate at which the signal level changes

Issues in Encoding

- Signal Spectrum
 - Lack of high frequencies reduces required bandwidth
 - Lack of dc component allows ac coupling via transformer, providing isolation reducing interference
- Clocking issues
 - Synchronizing transmitter and receiver is essential
 - External clock is one way used for synchronization
 - Synchronizing mechanism based on signal is also used & preferred (over using an external clock)

- Error detection
 - Can be built into signal encoding sometimes
- Cost and complexity
 - Higher signal rate (& thus data rate) lead to higher costs
 - Some codes require signal rate greater than data rate

Some Encoding Schemes

- Nonreturn to Zero-Level (NRZ-L)
- Nonreturn to Zero Inverted (NRZI)
- Bipolar-AMI (Alternate Mark Inversion)
- Pseudoternary
- Manchester
- Differential Manchester

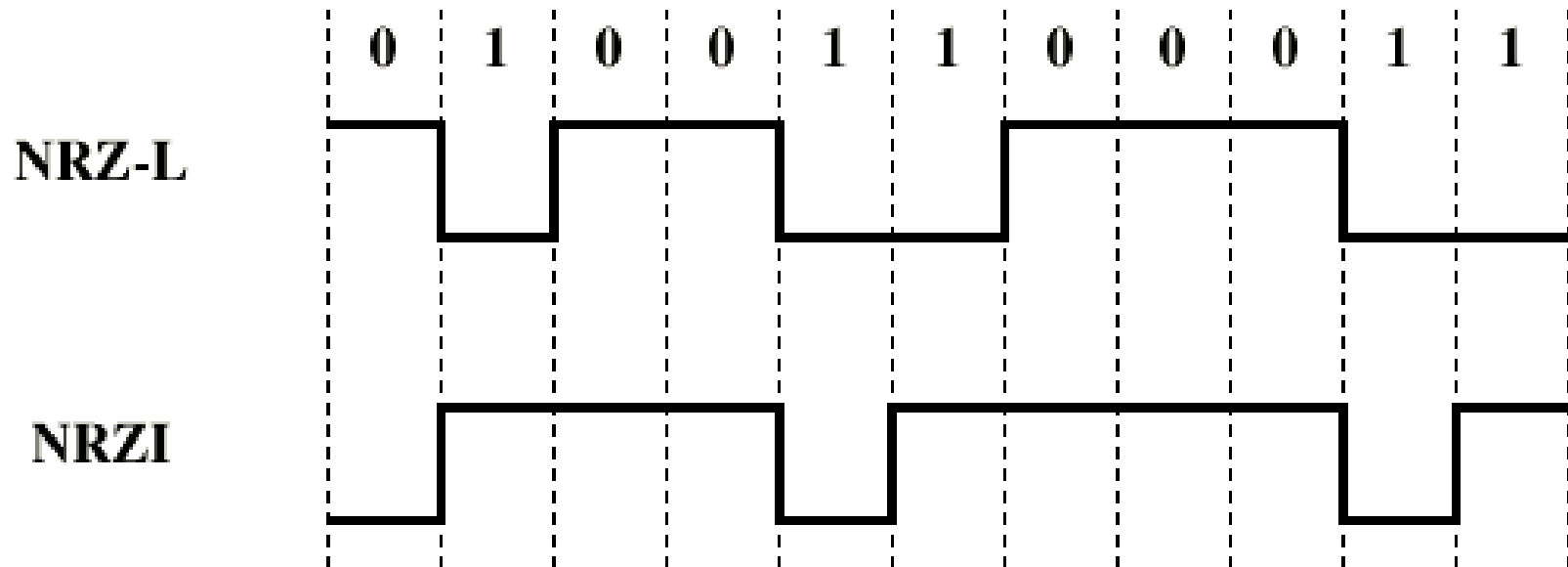
Nonreturn to Zero-Level (NRZ-L)

- Two different voltages for 0 and 1 bits
- Voltage held constant during bit interval
- Example:
 - 0 = High
 - 1 = Low

Nonreturn to Zero Inverted (NRZ-I)

- *Nonreturn to zero inverted on ones*
- Voltage held constant during bit interval
- Data encoded as presence or absence of signal transition at beginning of bit time
- Transition (low to high or high to low) denotes a binary 1
- No transition denotes binary 0
- An example of differential encoding (Data represented by changes rather than levels)

NRZ



NRZ pros and cons

- Pros
 - Easy to engineer
 - Makes good use of bandwidth
- Cons
 - Presence of DC component
 - Lack of synchronization capability, not self-clocking
 - Long string of 1's (or 0's) for NRZ-L or long string of 0's for NRZI will give constant voltage levels may cause loss of receiver synchronization

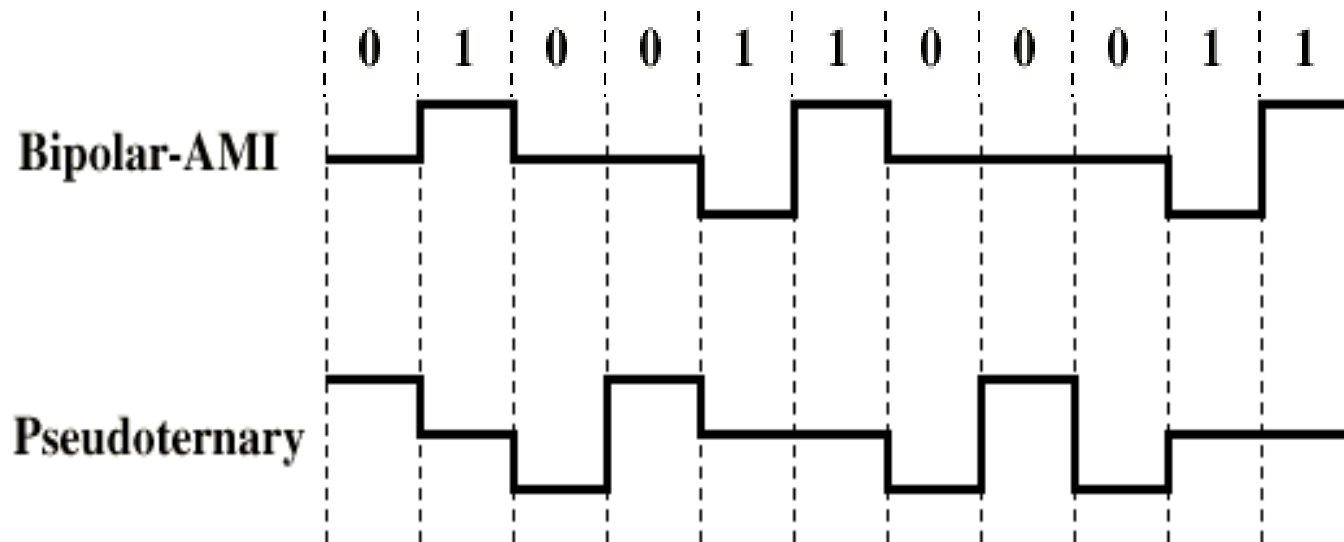
Multilevel Binary

- Use more than two levels
- Bipolar-AMI
 - Bit 0 represented by no line signal
 - Bit 1 represented by positive or negative pulse
 - Pulses for bit 1 alternate in polarity
 - No loss of sync if a long string of 1's happens (long string of 0's still a problem)
 - No net dc component
 - Lower bandwidth
 - Some error detection builtin

Pseudoternary

- Opposite of Bipolar-AMI
- Bit 1 represented by absence of line signal
- Bit 0 represented by alternating positive and negative
- No advantage or disadvantage over bipolar-AMI

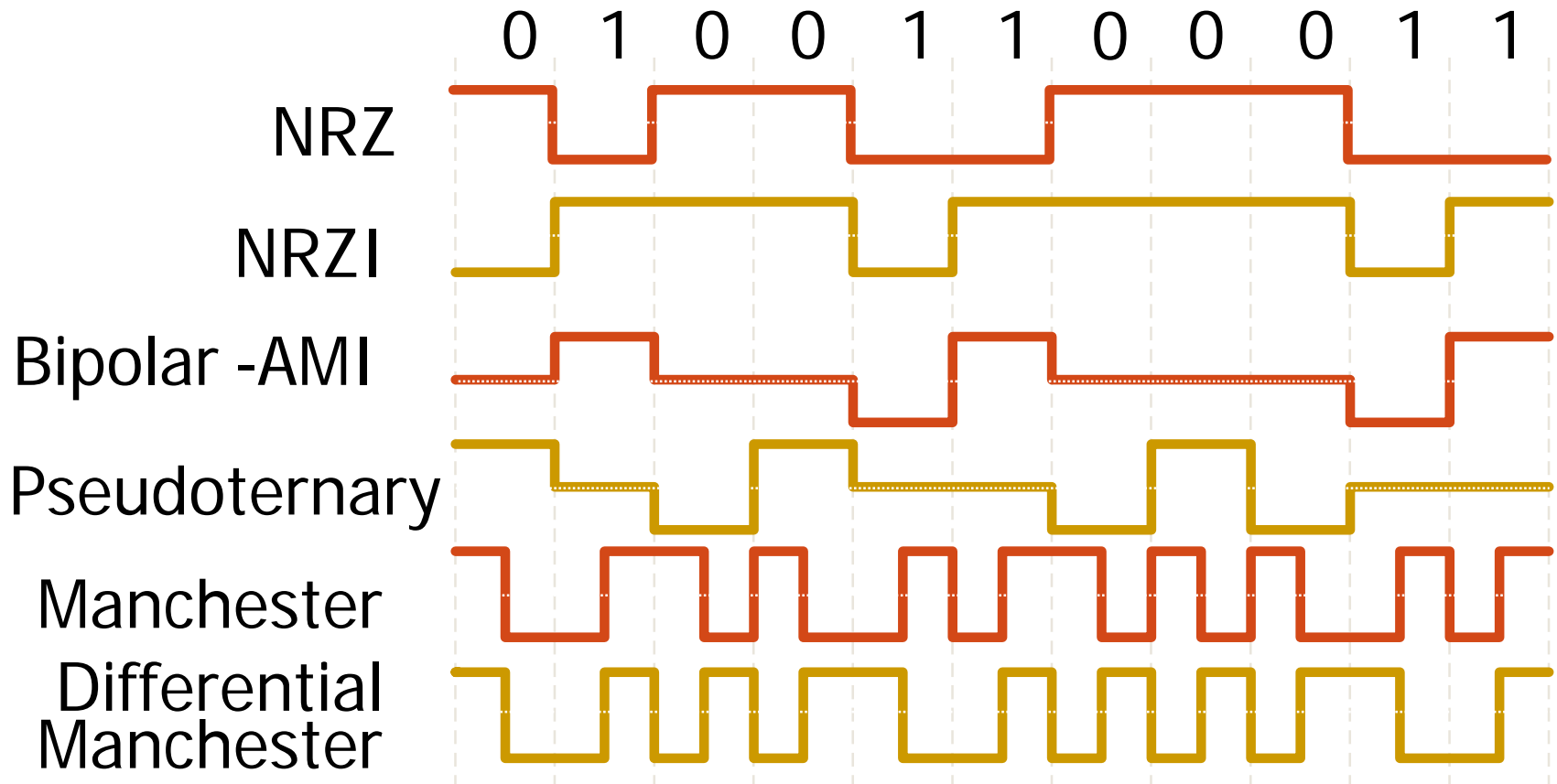
Bipolar-AMI and Pseudoternary



Biphase

- Manchester
 - Transition in middle of each bit period
 - Transition serves as clock and data
 - Low to high represents 1
 - High to low represents 0
- Differential Manchester
 - Midbit transition is for clocking only (always there)
 - Transition at start of a bit period represents 0
 - No transition at start of a bit period represents 1
 - Differential encoding scheme

Digital data, Digital signal

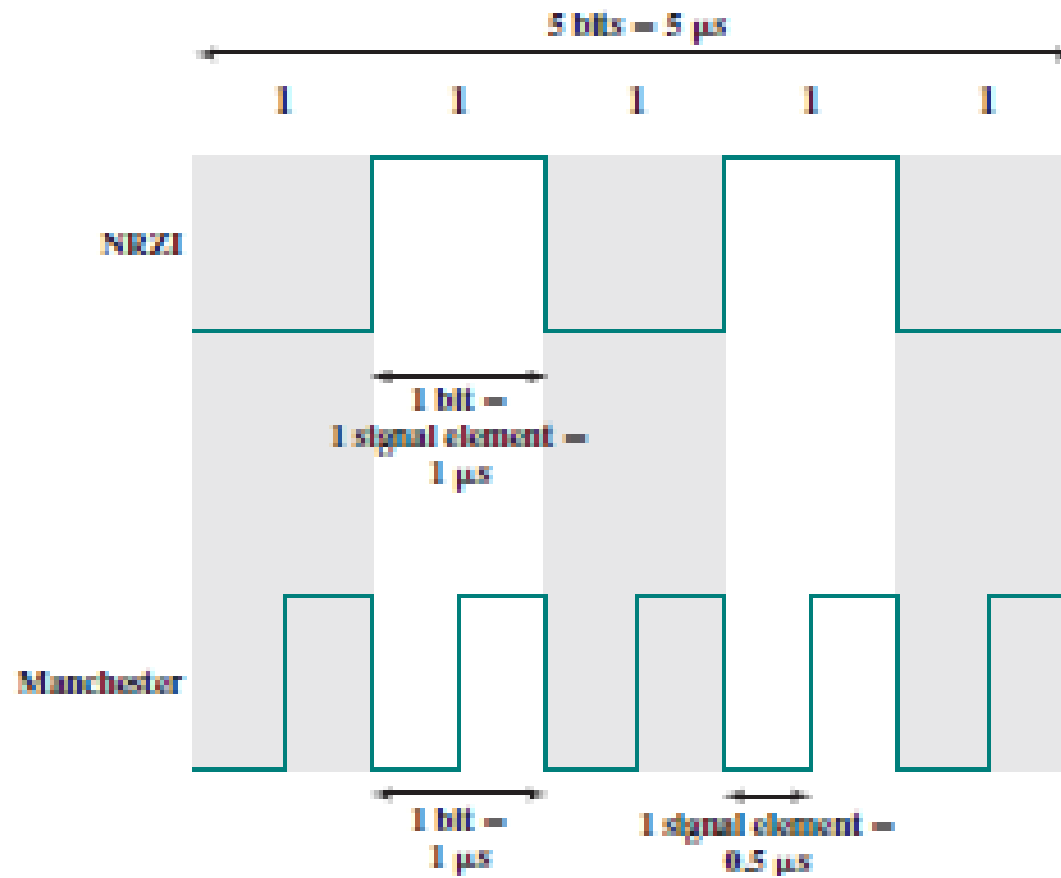


Biphase Pros and Cons

- Con
 - At least one transition per bit time and possibly two
 - Maximum modulation rate is twice that of NRZ
 - Requires more bandwidth
- Pros
 - Synchronization on mid bit transition (self clocking)
 - No dc component
 - Error detection
 - Absence of expected transition points to error in transmission

Modulation Rate

- Modulation rate/ Signaling rate of an encoding scheme = rate of signal transitions in the channel
- Higher the modulation rate, higher the bandwidth required
- Depends on the actual data stream
 - Should see the maximum modulation rate possible for some data stream
- Another way to characterize : No. of signal transitions per data bit (average/maximum)



| | Minimum | 101010... | Maximum |
|-------------------------|------------------|------------------|--------------------|
| NRZ-L | 0 (all 0s or 1s) | 1.0 | 1.0 |
| NRZI | 0 (all 0s) | 0.5 | 1.0 (all 1s) |
| Bipolar-AMI | 0 (all 0s) | 1.0 | 1.0 |
| Pseudoternary | 0 (all 1s) | 1.0 | 1.0 |
| Manchester | 1.0 (1010...) | 1.0 | 2.0 (all 0s or 1s) |
| Differential Manchester | 1.0 (all 1s) | 1.5 | 2.0 (all 0s) |

- So there is a tradeoff
 - Biphase is good for synchronization but needs high signaling rate (higher bandwidth)
 - Non-Biphase codes need lower signaling rate (so lower bandwidth) but may cause receiver to go out of sync
- Can we try to get good clock synchronization at low signaling rate?
 - Scrambling
 - $x\text{B}/y\text{B}$ codes (for different x, y values)

Scrambling

- Replace sequences that will cause long spells of constant voltage with filling sequences that introduces sufficient transitions
- Receiver should replace filling sequences with original data sequence
- Example: Bipolar-AMI with 8-zeroes substitution (B8ZS)
 - For any octet of all 0's
 - If last voltage pulse before the octet is +ve, replace with 000+−0−+
 - If last voltage pulse before the octet is −ve, replace with 000−+0+−
 - How does the receiver detect scrambling has been done?
 - Hint: Is this a valid sequence for Bipolar-AMI?

8B/10B Encoding

- Other similar schemes like 4B/5B, 5B/6B, 64B/66B (Block Codes)
- Two levels of encoding
 - Bit pattern encoded with another bit pattern
 - Encoded bit pattern encoded again over digital/analog signals (as studied)
- Data split into 8-bit Octets
- Each octet is replaced with a predefined 10-bit code
- Each 10-bit code has one of the following
 - 5 0's and 5 1's
 - 6 0's and 4 1's
 - 4 0's and 6 1's
- Ensures that in the final encoded string, not more than 5 0's or 5 1's can occur simultaneously
 - Ensures enough transitions for clock synchronizations

- Also ensures DC-balance
 - In a string of at least 20 bits, difference in count of 0's and 1's is at most 2
 - This requires introducing *running disparity* bits (we will not cover)
- Requires less b/w, can go over longer distances
- Reduces data rate (as more no. of bits transmitted)
- Used in many protocols, Ex: Gigabit Ethernet

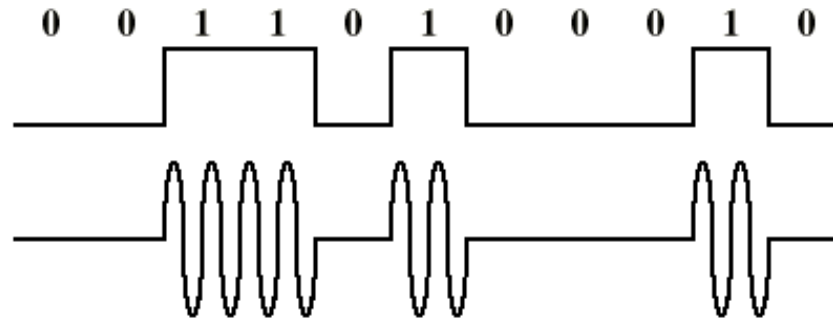
Encoding Digital Data with Analog Signals

- Modem (modulator-demodulator) is used to convert digital data to analog signal and vice versa
- Carrier signal: basic signal that is changed (modulated)
- Three basic modulation techniques are used:
 - Amplitude shift keying (ASK)
 - Frequency shift keying (FSK)
 - Phase shift keying (PSK)

- Amplitude Shift Keying
 - Values represented by different amplitudes of a carrier signal
 - Usually, one amplitude is zero
 - i.e. presence and absence of carrier is used
 - Susceptible to noise
 - Used over optical fiber (but usually with some digital encoding before ASK)

- Frequency Shift Keying
 - Values represented by different frequencies (near carrier)
 - Less susceptible to noise than ASK
 - Constrained by bandwidth of channel
- Phase Shift Keying
 - Phase of carrier signal is shifted to represent data

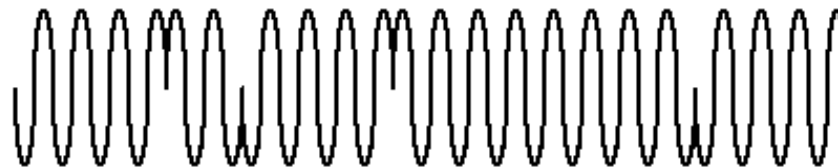
Modulation Techniques



(a) Amplitude-shift keying



(b) Frequency-shift keying



(c) Phase-shift keying

Error Detection

Error Detection & Correction

- Error – bit pattern sent and bit pattern received are different. Possible causes
 - Framing errors
 - Noise...
- Error detection – detects error but may or may not correct
- Error correction – detects and corrects errors
 - Usually requires larger number of extra bits with data bits, not usually done in network protocols, we will skip

Error Detection Techniques

- Basic Principle
 - Transmitter: For a given bit stream M , additional bits (called error-detecting bits or check bits) are calculated as a function of M and appended to the end of the data bits in M
 - Receiver: On receiving the bit stream, separates the data bits, performs the same calculation on data bits, and compares the two results. A detected error occurs if there is a mismatch.
- Common Methods
 - Parity Check
 - Cyclic Redundancy Codes
 - Checksum

Parity Check

- One extra “parity” bit is added to each word
 - Odd parity: bit added so as to make # of 1's odd
 - Even parity: makes total # of 1's even
- Detects any odd number of bit errors (error in 1bit, 3 bits, 5 bits ...), but can be fooled by any even number of errors
- Simple and easy to implement
- Not very robust against noise

CRC

- Powerful error detection method, easily implemented in hardware
- Message (M) to be transmitted is appended with extra frame checksum bits (F), so that bit pattern transmitted (T) is perfectly divisible by a special “generator” pattern (P) - (divisor)
- At destination, divide received message by the same P. If remainder is nonzero, there is an error

- Let
 - $T = (k+n)$ -bit frame to be transmitted, $n < k$
 - $M = k$ -bit message, the first k bits of T
 - $F = n$ -bit FCS, the last n bits of T
 - $P = n+1$ bits, generator pattern (predetermined divisor)
- The concept uses modulo-2 arithmetic
 - no carries/borrows; $\text{add} \equiv \text{subtract} \equiv \text{XOR}$

- Extend M with n '0's to the right ($\equiv 2^n M$)(shift left by n bits)
- Divide extended message by P to get R ($2^n M / P = Q + R/P$)
- Add R to extended message to form T ($T = 2^n M + R$)
- Transmit T
- At receiver, divide T by P. Nonzero remainder \Rightarrow error

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

Note:
Remainder
R=F=FCS
in these
examples

Note: $R+R=0$ in mod-2 arithmetic

$$0 \oplus 0 = 0$$

$$1 \oplus 1 = 0$$

- $M = 110011$, $P = 11001$, $R = 4$ bits
- Append 4 zeros to M , we get 1100110000

$$\begin{array}{r}
 - 2. \qquad \qquad \qquad 100001 \\
 11001 \overline{) 1100110000} \\
 \oplus 11001 \\
 \hline
 00001 \\
 \oplus 00000 \\
 \hline
 00010 \\
 \oplus 00000 \\
 \hline
 00100 \\
 \oplus 00000 \\
 \hline
 01000 \\
 \oplus 00000 \\
 \hline
 10000 \\
 \oplus 11001 \\
 \hline
 1001
 \end{array}$$

- 3. $\therefore T = 110011$ 1001

For each stage of division, if the number of dividend bits equals number of divisor P bits, then $Q=1$, otherwise $Q=0$

- Can view CRC generation in terms of polynomial arithmetic also
- Any bit pattern \equiv polynomial in dummy variable X with the bits as coefficients as shown in the following example:
 - e.g., $M = 110011 \equiv 1 \cdot X^5 + 1 \cdot X^4 + 0 \cdot X^3 + 0 \cdot X^2 + 1 \cdot X + 1 \cdot X^0$
 $\therefore M(X) = X^5 + X^4 + X + 1$

- Commonly used polynomials, $P(X)$
 - CRC-16 = $X^{16} + X^{15} + X^2 + 1$
 - CRC-CCITT = $X^{16} + X^{12} + X^5 + 1$
 - CRC-32 = $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^{12} + X^4 + X^2 + X + 1$

- CRC can detect
 - All single-bit errors
 - All double-bit errors, as long as $P(X)$ has a factor with at least three terms (as long as P has at least three 1s)
 - Any odd number of errors, as long as $P(X)$ contains a factor $(X+1)$
 - Any burst error for which the length of the burst is less than or equal to the length of the FCS
 - Many other larger burst errors
- Strong error detection scheme, easy implementation in hardware