CompSci 356: Computer Network Architectures

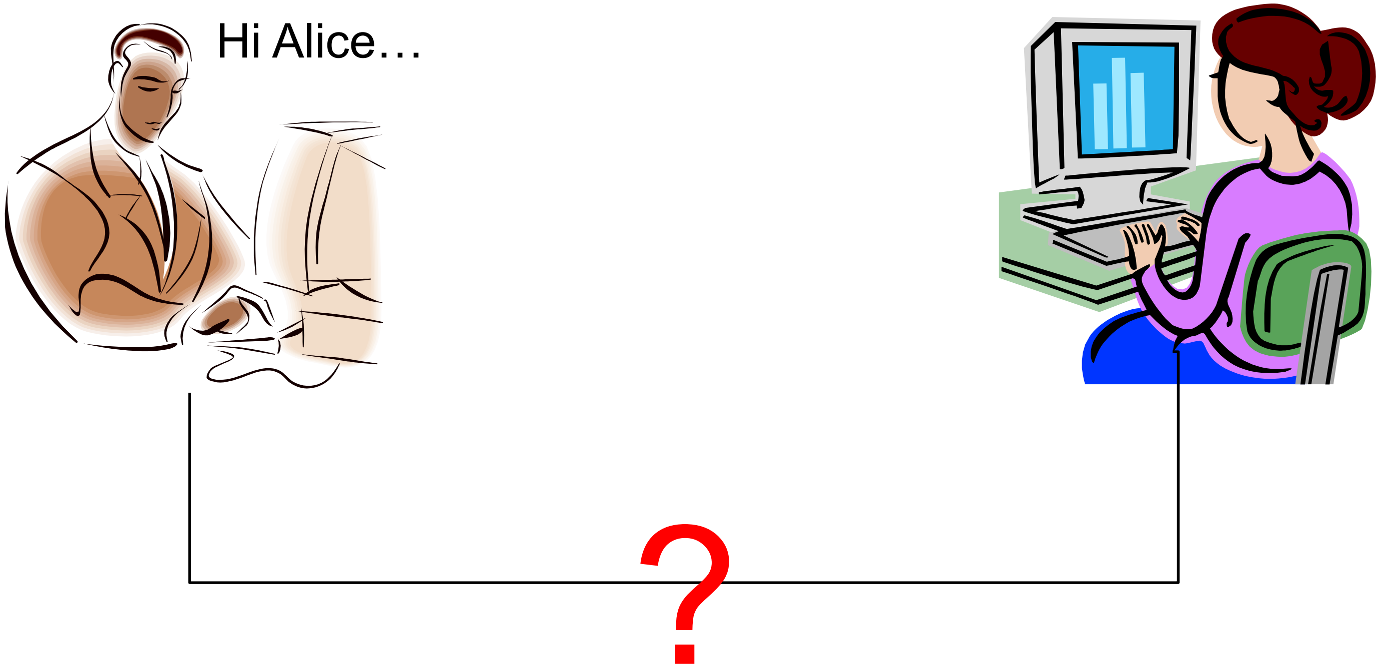
Lecture 4: Link layer: Encoding, Framing, and Error Detection
Ref. Chap 2.2, 2.3,2.4

Xiaowei Yang
xwy@cs.duke.edu

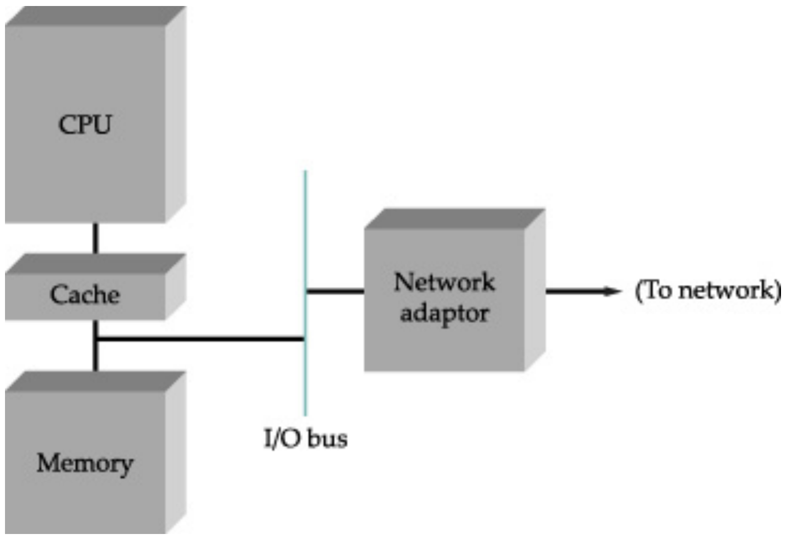
Overview

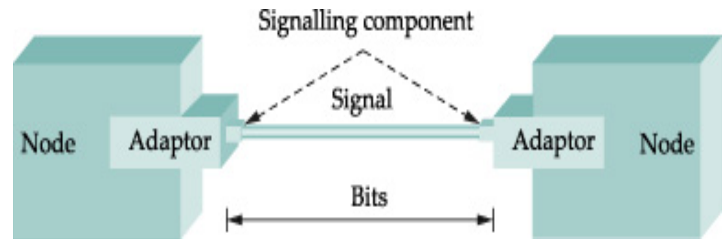
- Link layer functions
  - Encoding
  - Framing
  - Error detection

The simplest network is one link plus two nodes



Recap: Put bits on the wire

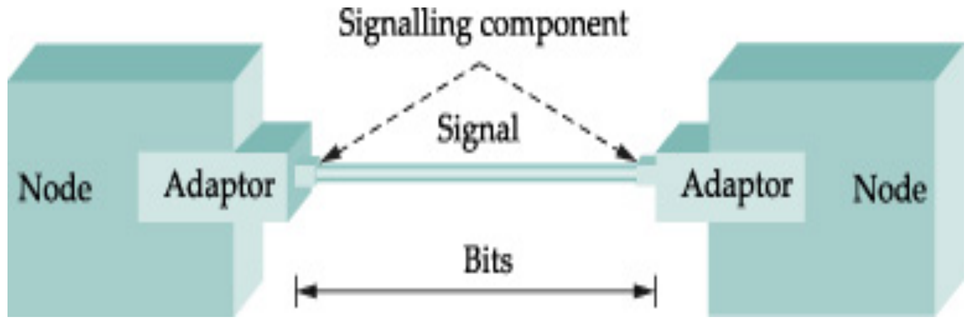




- Each node (e.g. a PC) connects to a network via a network adaptor.
- The adaptor delivers data between a node’s memory and the network.
- A device driver is the program running inside the node that manages the above task.

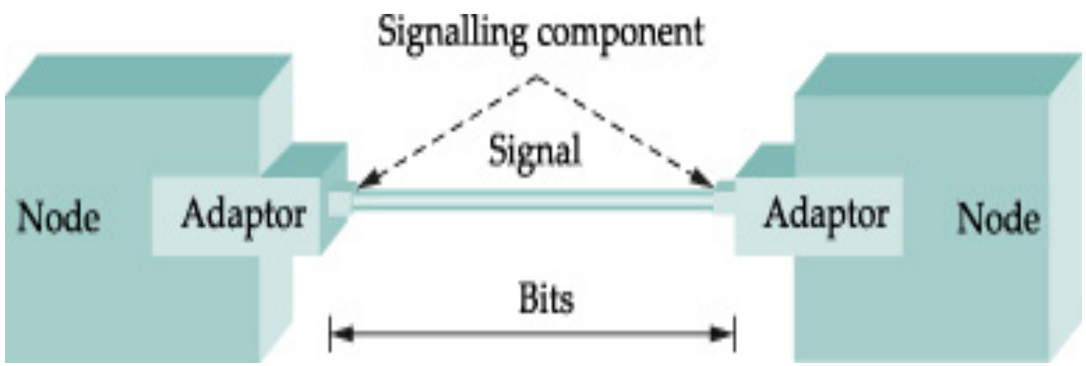
- At one end, a network adaptor encodes and modulates a bit into signals on a physical link.
- At the other end, a network adaptor reads the signals on a physical link and converts it back to a bit.

Metrics to describe a link



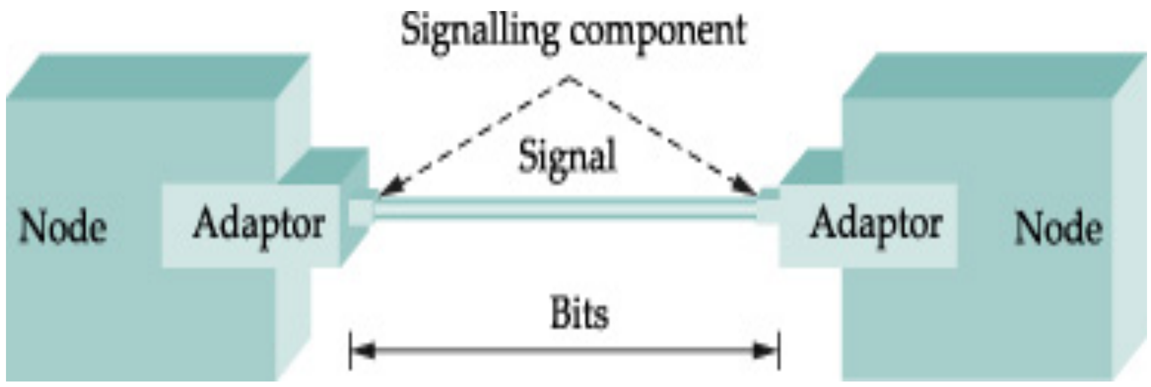
- Bandwidth
  - Why are some links slow/fast?
- Latency/delay
- Transmission delay (serialization)
  - Store and forward
- Delay * bandwidth product
- Throughput
  - How long does it take to send a file?

Link-layer functions



- Most functions are completed by adapters
  - Encoding
  - Framing
  - Error detection
  - Reliable transmission (next lecture)

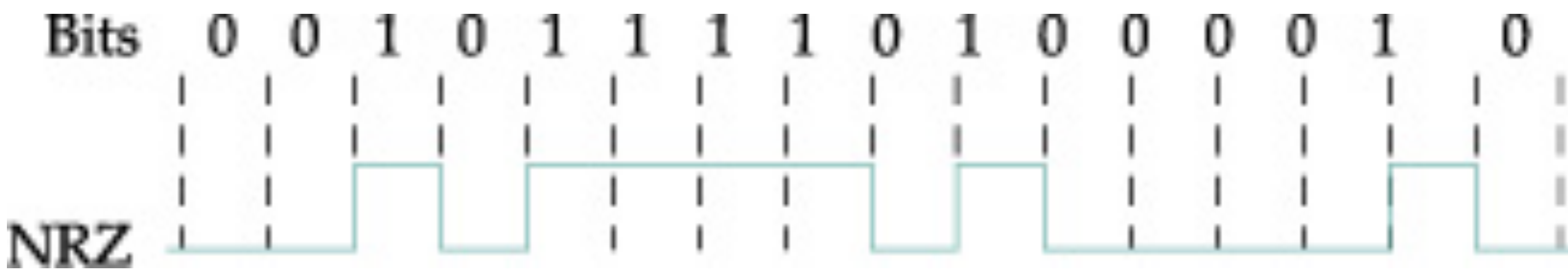
Encoding



- Implemented in hardware
- High and low signals, ignore modulation
- Simplest one: 1 to high, 0 to low

Non-return to zero

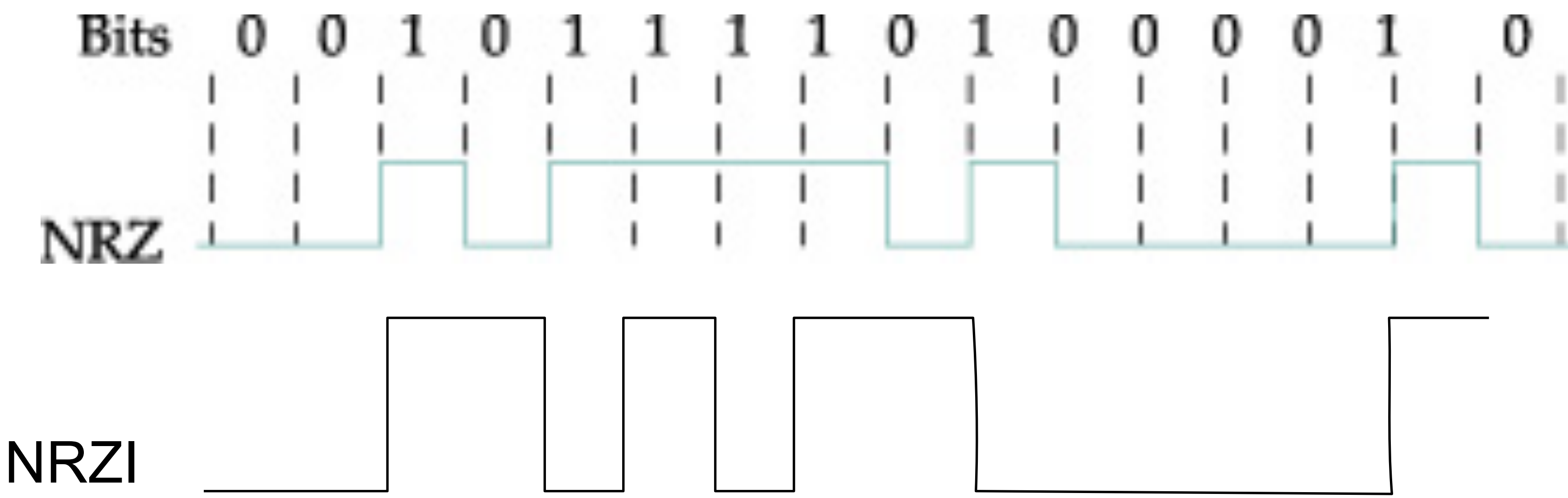
- 1 to high, 0 to low



- Not good for decoding
  - Baseline wander
  - Clock recovery

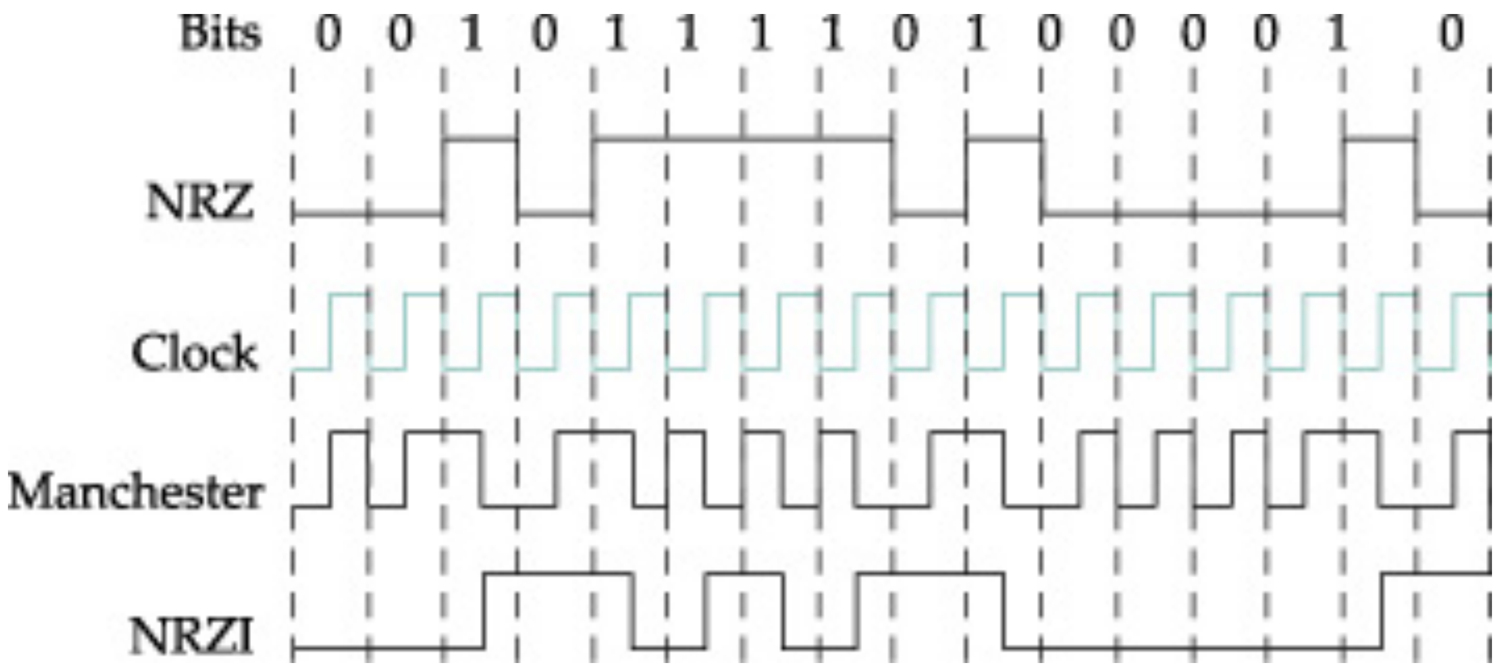
Solution 1: Nonreturn to zero inverted (NRZI)

- A transition from current signal encodes 1
- No transition encodes 0
- Does it solve all problems?
  - Not for consecutive 0s



Solution 2: Manchester encoding

- Clock XOR NRZ
  - 1: high → low; 0: low → high



  - Drawback: doubles the rate at which signals are sent
    - Baud rate: signal change rate
    - Bit rate = half of baud rate. 50% efficient

Final solution: 4B/5B

- Key idea: insert extra bits to break up long sequences of 0s or 1s

- 4-bit of data are encoded in a 5-bit code word
  - 16 data symbols, 32 code words
  - At most one leading 0, two trailing 0s
  - For every pair of codes, no more than three consecutive 0s

- 5-bit codes are sent using NRZI

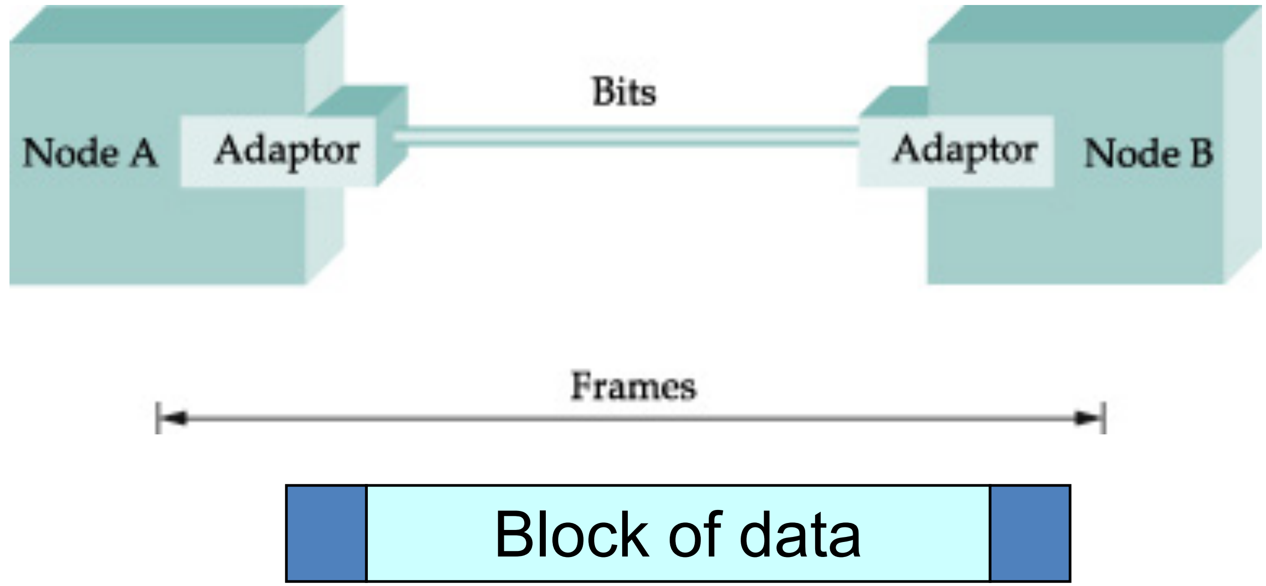
4-bit data symbol	5-bit code
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

- Exercise:
  - 00101101
- What's the high/low signal sequence?
- Efficiency?

Overview

- Link layer functions
  - Encoding
  - Framing
  - Error detection

Framing



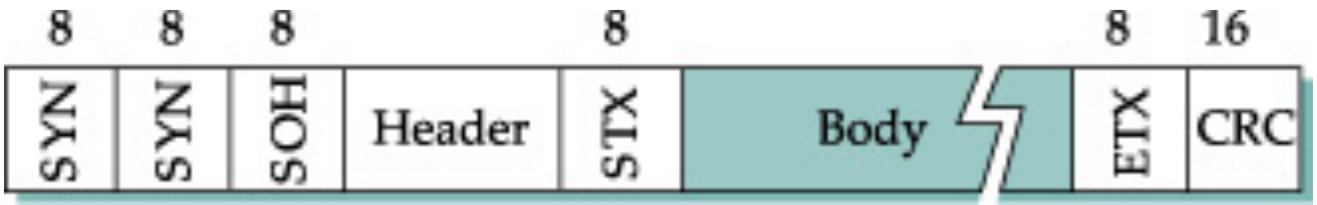
- Now we’ve seen how to encode bitstreams
- But nodes send blocks of data (frames)
  - A’s memory → adaptor → adaptor → B’s memory
- An adaptor must determine the boundary of frames

Variety of Framing Protocols

- Framing
  - Why is it an important task of an adaptor?
    - Frames may belong to different apps
    - Need to decide when to deliver them to apps

- Design choices
  - Byte-oriented protocols
    - Sentinel approach
    - Byte-counting approach
    - BISYNC, PPP, DDCMP
  - Bit-oriented protocols
  - Clock-based framing

Byte-oriented protocols: the sentinel approach



•Transmitted from the leftmost bit

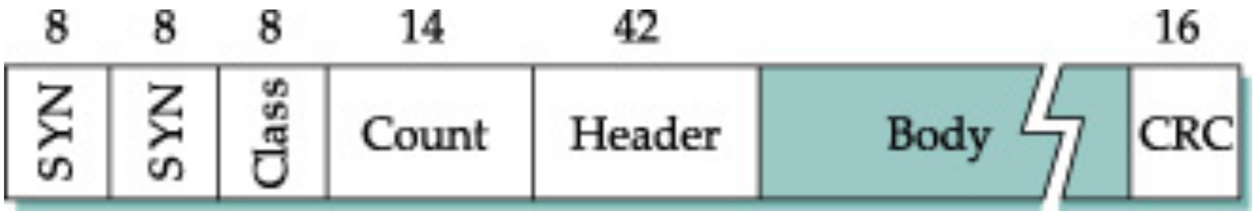
•Binary Synchronous Communication (BISYNC) by IBM in late 60s

- Frame: a collection of bytes (characters)
- SYN, ETX
  - What if special characters appear in a data stream?
    - Escape character: DLE
    - Character stuffing

Point-to-Point Protocol (PPP)

- Internet dialup access
  - RFC 1661, 1994
- Flag: 01111110;
- Address & Control: default
- Protocol: de-multiplexing
  - IP, Link Control Protocol, …,
- Checksum: two or four bytes
- Link Control Protocol
  - Set up and terminate the link
  - Negotiate other parameters
    - maximum receive unit

Byte-oriented protocols: the byte counting approach



•DDCMP by DECNET (Digital Data Communication Message Protocol)

- A byte count field
- The corruption of the count field
  - The sentinel approach: Corrupted ETX

Bit-oriented protocols

•High-level data link control (HDLC) protocol

- Frame: a collection of bits
- Beginning/ending sequence: 01111110
- Idle sequence:
  – 01111110 or idle flags 11111111
- Bit-stuffing for data
- Frames are of variable length

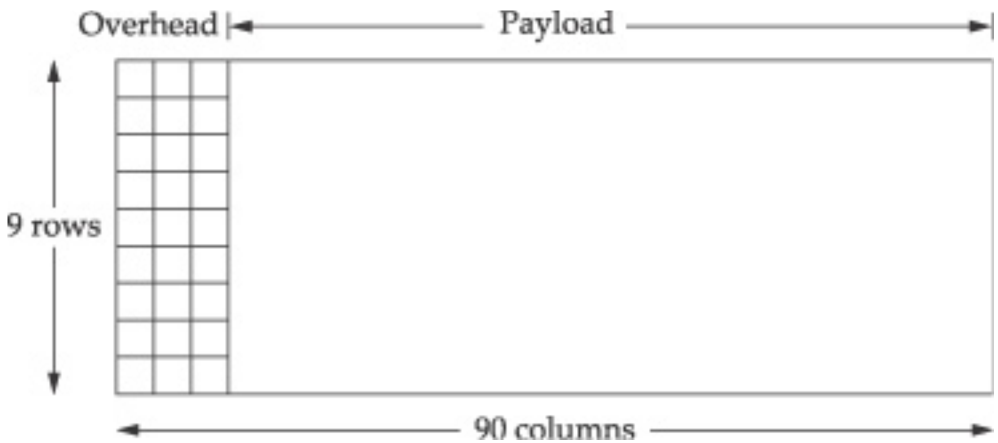
The bit-stuffing algorithm

- Bit-stuffing for data
  - Sender: inserts a 0 after every five consecutive 1's
  - Receiver: after five consecutive 1's,
    - If the next bit is 0, removes it
    - If the next bit is 1
      - If the next bit is 0 (i.e. the last 8 bits are 01111110), then frame ends
      - Else error; discard frame, wait for next 01111110 to receive

An exercise

- Suppose a receiver receives the following bit sequence
  - 011010111110101001111111011001111110
- What’s the resulting frame after removing stuffed bits? Indicate any error.

Clock-based Framing



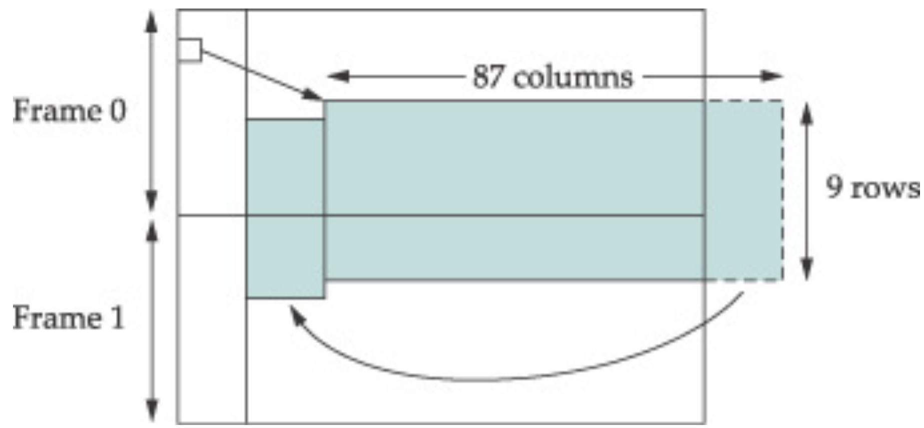
•STS-1/OC-1 frame

•51.840Mbps

•The slowest SONET link

- Synchronous Optical Network (SONET)
- Each frame is 125 us long, 810 bytes = 125 us * 51.84Mbps
- Clock synchronization
  - special pattern repeated enough times

Synchronized timeslots as placeholder



- Real frame data may float inside

Overview

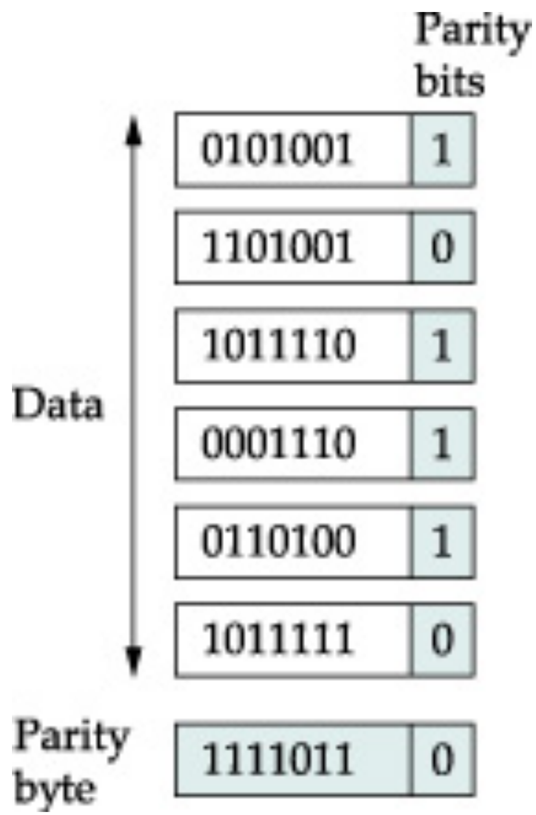
- Link layer functions
  - Encoding
  - Framing
  - Error detection

Error detection

- Error detection code adds redundancy
  - Analogy: sending two copies
  - Parity
  - Checksum
  - CRC

- Error correcting code

Two-dimensional parity



A sample frame of six bytes

- Even parity bit
  - Make an even number of 1s in each row and column
- Detect all 1,2,3-bit errors, and most 4-bit errors

Internet checksum algorithm

- Basic idea (for efficiency)
  - Add all the words transmitted and then send the sum.
  - Receiver does the same computation and compares the sums
- IP checksum
  - Adding 16-bit short integers using 1’s complement arithmetic
  - Take 1’s complement of the result
- Used by lab 2 to detect errors

1's complement arithmetic

- -x is each bit of x inverted
- If there is a carry bit, add 1 to the sum
- [-2^(n-1)-1, 2^(n-1)-1]
- Example: 4-bit integer
  - -5 + -2
  - +5: 0101; -5: 1010;
  - +2: 0010; -2: 1101;
  - -5 + -2 = 1010+1101 = 0111 + one carrier bit;
  - → 1000 = -7

Calculating the Internet checksum

- u_short cksum (u_short *buf, int count) {

```
register u_long sum = 0;

while  (count--) {
    sum += *buf++;
    if  (sum & 0xFFFF0000) {
        /* carry occurred. So wrap around */
        sum &= 0xFFFF;
        sum++;
    }
} // one’s complement sum
return ~(sum & 0xFFFF); // one’s complement of the sum
}
```

Verifying the checksum

- Adds all 16-bit words together, including the checksum
- 0: correct
- 1: errors

Remarks

- Can detect 1 bit error
- Not all two-bits
- Efficient for software implementation

Cyclic Redundancy Check

- Cyclic error-correcting codes
- High-level idea:
  - Represent an n+1-bit message with an n degree polynomial M(x)
  - Divide the polynomial by a degree-k divisor polynomial C(x)
  - k-bit CRC: remainder
  - Send Message + CRC that is dividable by C(x)

Polynomial arithmetic modulo 2

– B(x) can be divided by C(x) if B(x) has higher degree

– B(x) can be divided once by C(x) if of same degree

  - x^3 + 1 can be divided by x^3 + x^2 + 1
  - The remainder would be 0*x^3 + 1*x^2 + 0*x^1 + 0*x^0 (obtained by XORing the coefficients of each term)

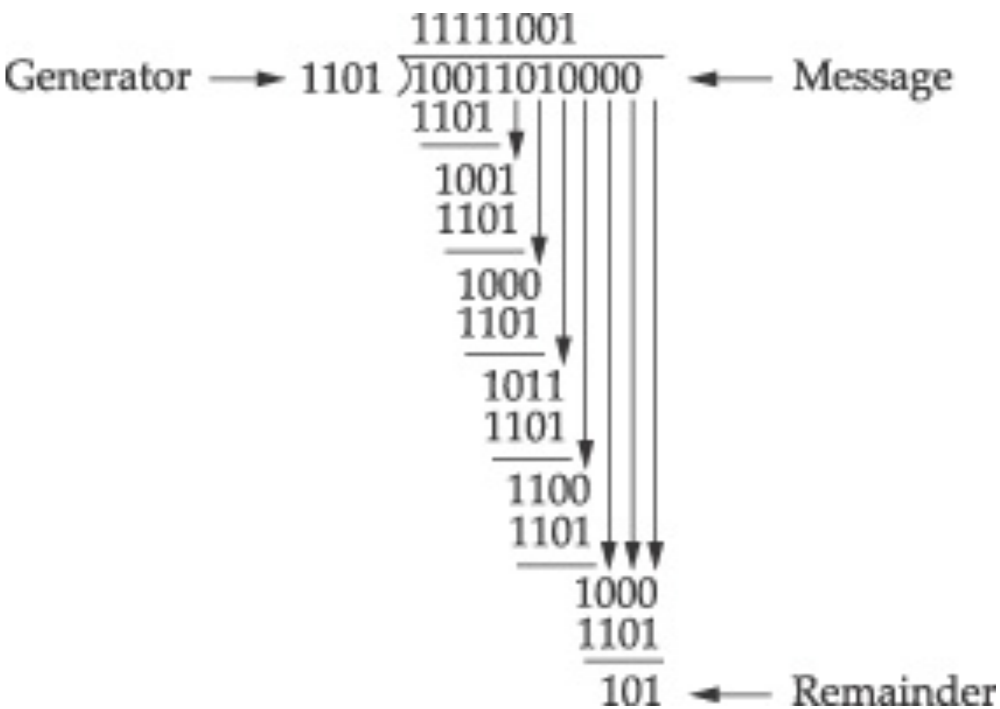
– Remainder of B(x)/C(x) = B(x) – C(x)

– Substraction is done by XOR each pair of matching coefficients

CRC algorithm

1. Multiply M(x) by x^k. Add k zeros to Message. Call it T(x)
2. Divide T(x) by C(x) and find the remainder
3. Send P(x) = T(x) – remainder
   - Append remainder to T(x)
- P(x) dividable by C(x)

An example



8-bit msg
– 10011010

Divisor (3bit CRC)
– 1101

Msg sent: 10011010101

How to choose a divisor

- Arithmetic of a finite field
- Intuition: unlikely to be divided evenly by an error
- Corrupted msg is $P(x) + E(x)$
- If $E(x)$ is single bit, then $E(x) = x^i$
- If $C(x)$ has the first and last term nonzero, then detects all single bit errors
- Find $C(x)$ by looking it up in a book

Summary

- Link layer functions
  - Encoding
    - NRZ, NRZI, Manchester, 4B/5B
  - Framing
    - Byte-oriented, bit-oriented, time-based
    - Bit stuffing
  - Error detection
    - Parity, checkshum, CRC