CSC 458/2209 – Computer Networks

Handout # 4: Link Layer, Error Detection/Correction

Professor Yashar Ganjali
Department of Computer Science
University of Toronto

yganjali@cs.toronto.edu
http://www.cs.toronto.edu/~yganjali

Announcements

- Programming assignment 1 out next week (Sep 17th).
- To be completed in groups of *2-3* people.
- Due: Oct. 11th at 5PM.
  - Will have test runs during the last week.
- Tutorials
  - We have a tutorial on socket programming this Friday.
  - Next week the tutorial will be a review of this assignment.
- Links posted on class web page:
  - Socket programming
  - Coding guidelines
- Use Piazza if you have any questions.

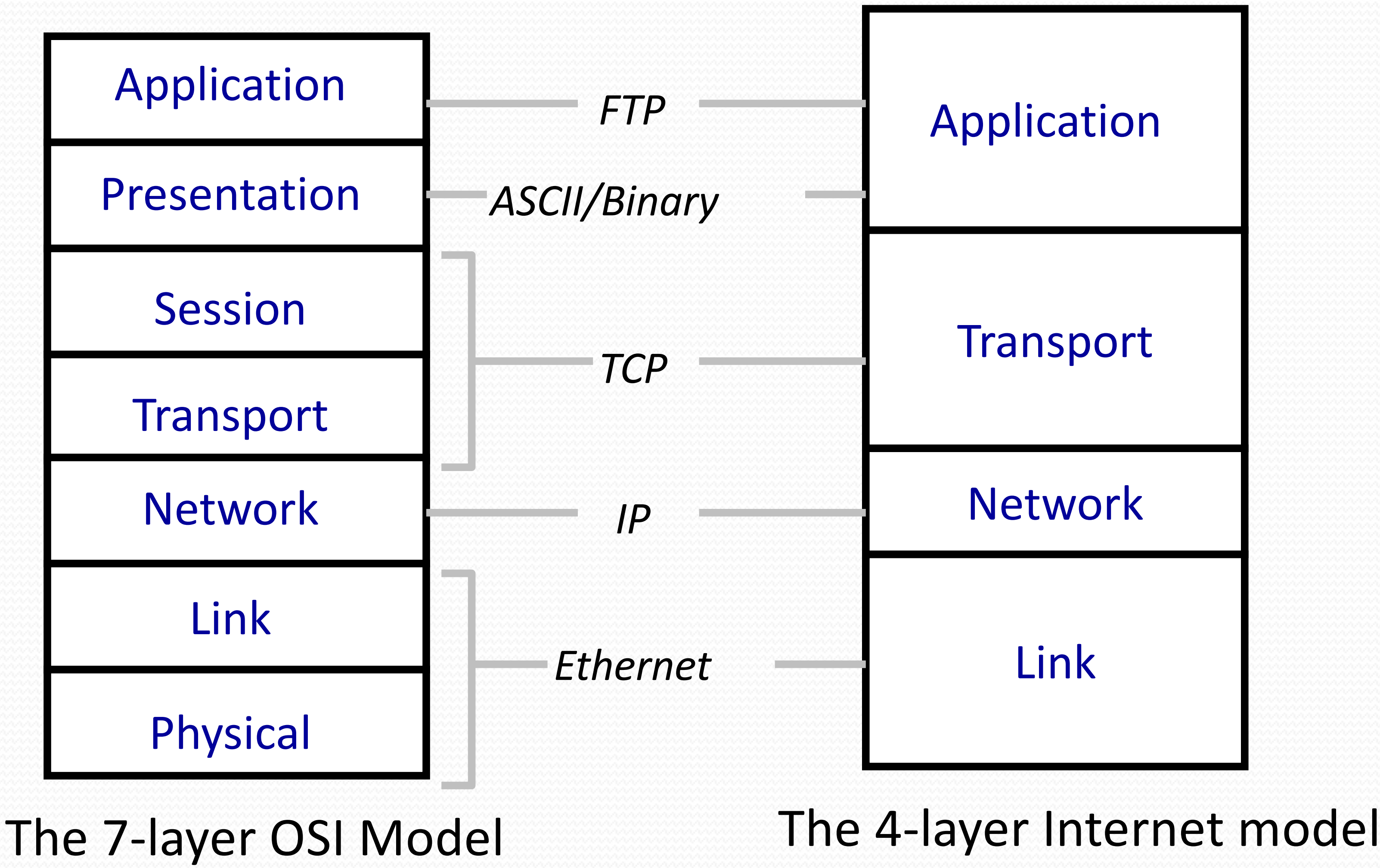
Announcements – Cont'd

- Reading for this week:
  - Chapter 1 of the textbook

- Next week: Chapter 2

Last Time …

Protocols, layering and reference models



Outline

Part 1. Physical/link layer

- Different types of media
- Encoding bits with signals
- Framing
- Model of a link

Part 2. Error detection and correction

- Hamming distance
- Parity, checksums, CRC, …

Part 1 – Physical/Link Layer

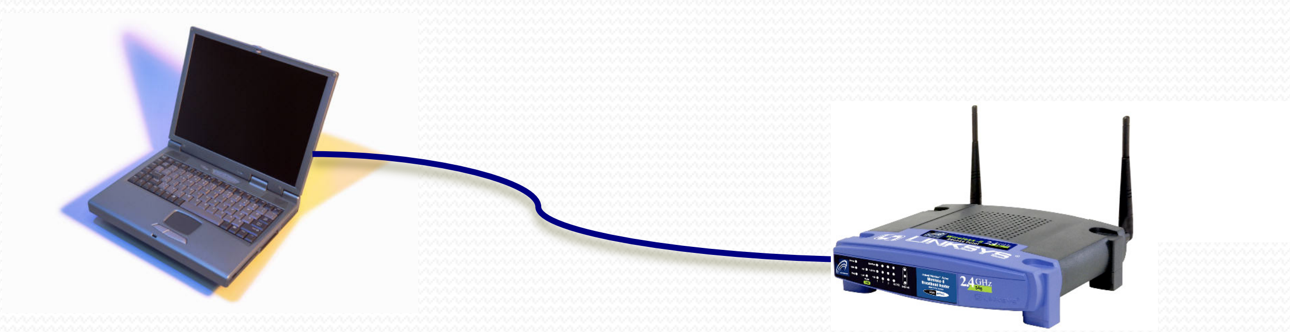
Focus:

<u>How do we send a message across a wire?</u>

The physical / link layers:

1. Different kinds of media
2. Encoding bits, messages
3. Model of a link

Application
Presentation
Session
Transport
Network
Data Link
Physical

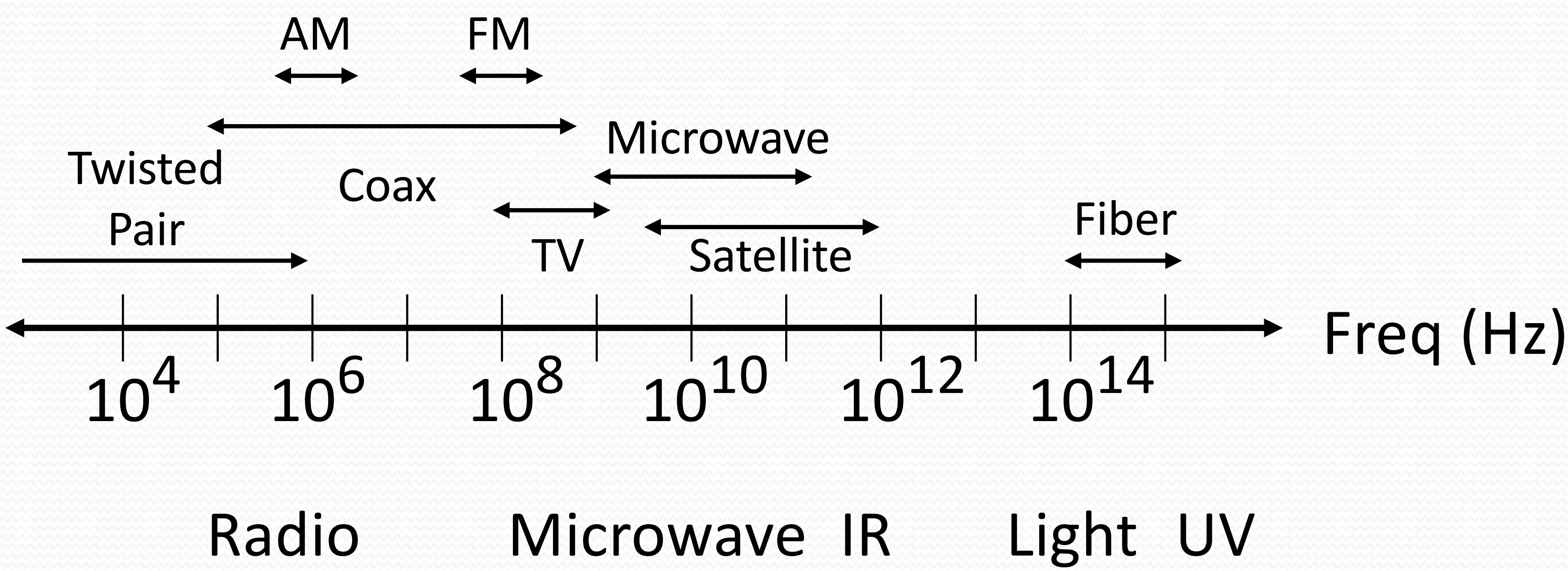


1. Different Types of Media

- Wire
  - Twisted pair, e.g., CAT5 UTP, 10 → 100Mbps, 100m
  - Coaxial cable, e.g, thin-net, 10 → 100Mbps, 200m
- Fiber
  - Multi-mode, e.g., 100Mbp/s, 600m
  - Single-mode, e.g., 100 → 2400 Mbps, 40km
- Wireless
  - Infra-red, e.g., IRDA, ~1Mbps
  - RF, e.g., 802.11 wireless LANs, Bluetooth (2.4GHz)
  - Microwave, satellite, cell phones, …

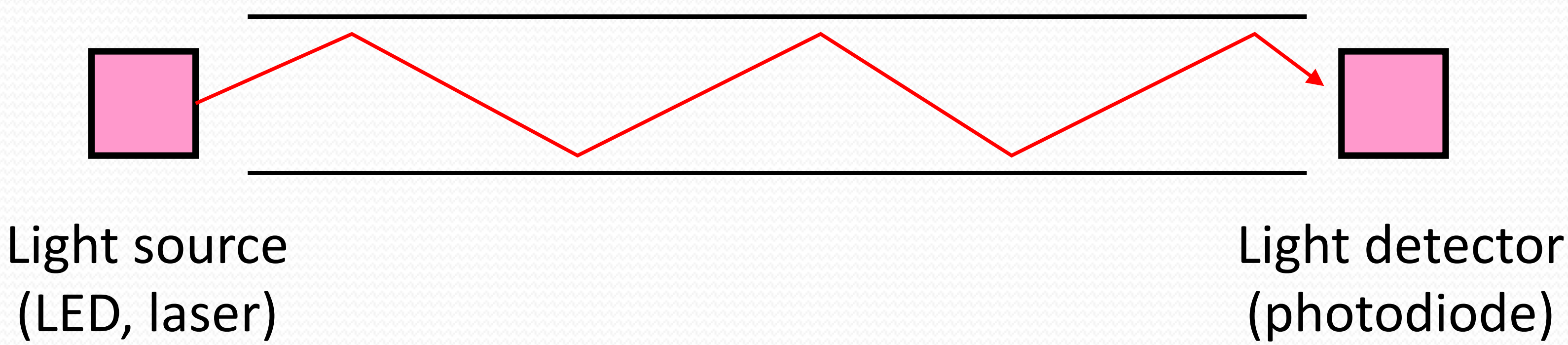
Wireless

- Different frequencies have different properties
- Signals subject to atmospheric/environmental effects



Fiber

- Long, thin, pure strand of glass
  - light propagated with total internal reflection
  - enormous bandwidth available (terabits)



- Multi-mode allows many different paths, limited by dispersion
- Chromatic dispersion if multiple frequencies

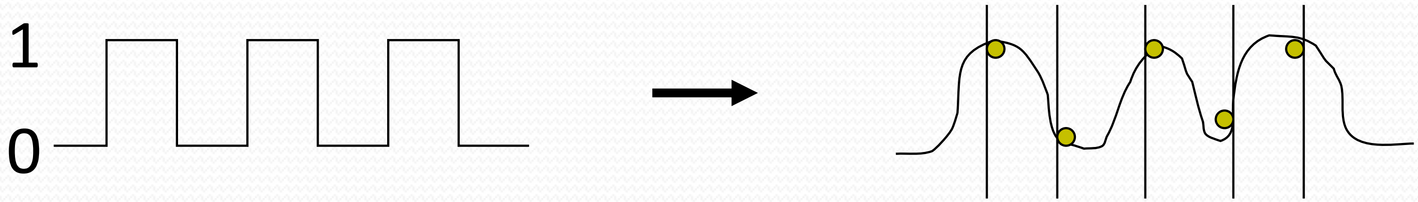
Bandwidth of a Channel

- EE: bandwidth (B, in Hz) is the width of the pass-band in the frequency domain
- CS: bandwidth (bps) is the information carrying capacity (C) of the channel

- Shannon showed how they are related by noise
  - Noise limits how many signal levels we can safely distinguish
  - Geekspeak: “cannot distinguish the signal from the noise”

2. Encoding Bits with Signals

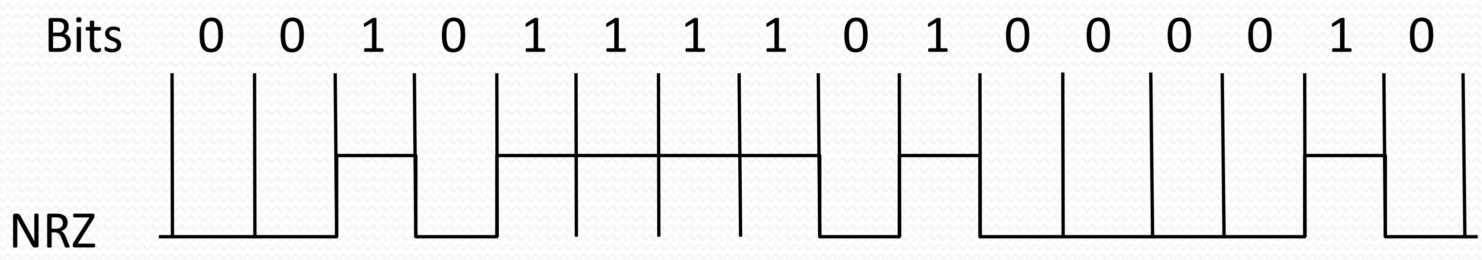
- Generate analog waveform (e.g., voltage) from digital data at transmitter and sample to recover at receiver



- We send/recover symbols that are mapped to bits
  - Signal transition rate = baud rate, versus bit rate
- This is baseband transmission …

NRZ and NRZI

- Simplest encoding, NRZ (Non-return to zero)
  - Use high/low voltages, e.g., high = 1, low = 0
- Variation, NRZI (NRZ, invert on 1)
  - Use transition for 1s, no transition for 0s



Clock Recovery

- **Problem**: How do we distinguish consecutive 0s or 1s?
- If we sample at the wrong time we get garbage …
- If sender and receiver have exact clocks no problem
  - But in practice they drift slowly
- This is the problem of clock recovery

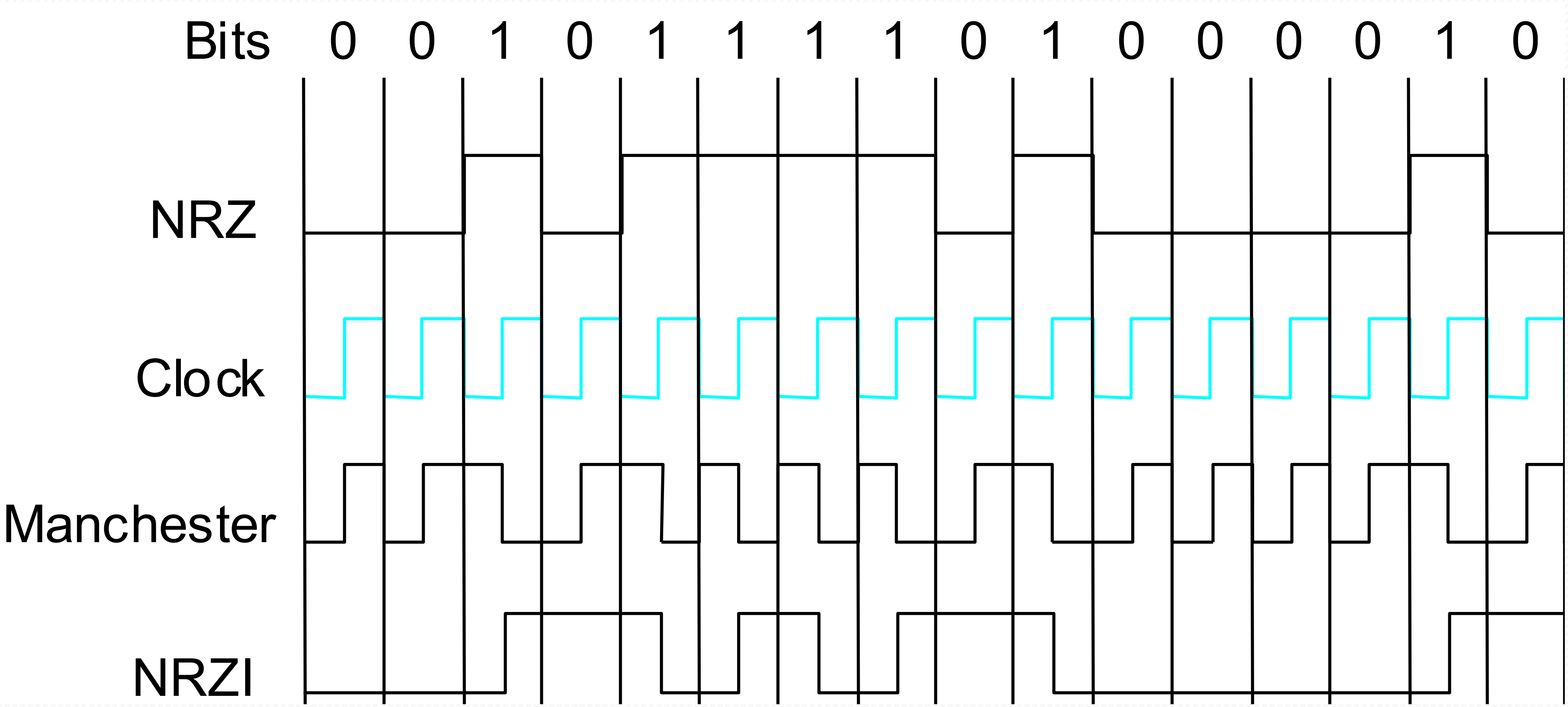
- **Possible solutions**:
  - Send separate clock signal → expensive
  - Keep messages short → limits data rate
  - Embed clock signal in data signal → other codes

Manchester Coding

- Make transition in the middle of every bit period
  - Low-to-high is 0; high-to-low is 1
  - Signal rate is twice the bit rate
  - Used on 10 Mbps Ethernet

- Advantage: self-clocking
  - clock is embedded in signal, and we re-sync with a phase-locked loop every bit
- Disadvantage: 50% efficiency

Coding Examples



4B/5B Codes

- We want transitions *and* efficiency …
- **Solution**: map data bits (which may lack transitions) into code bits (which are guaranteed to have them)
- 4B/5B code:
  - 0000 → 11110, 0001 → 01001, … 1111 → 11101
  - Never more than three consecutive 0s back-to-back
  - 80% efficiency
- This code is in LANs such as FDDI, 100Mbps Ethernet

3. Framing

- Need to send message, not just bits
  - Requires that we synchronize on the start of message reception at the far end of the link
  - Complete Link layer messages are called <u>frames</u>

- Common approach: Sentinels
  - Look for special control code that marks start of frame
  - And escape or “stuff” this code within the data region

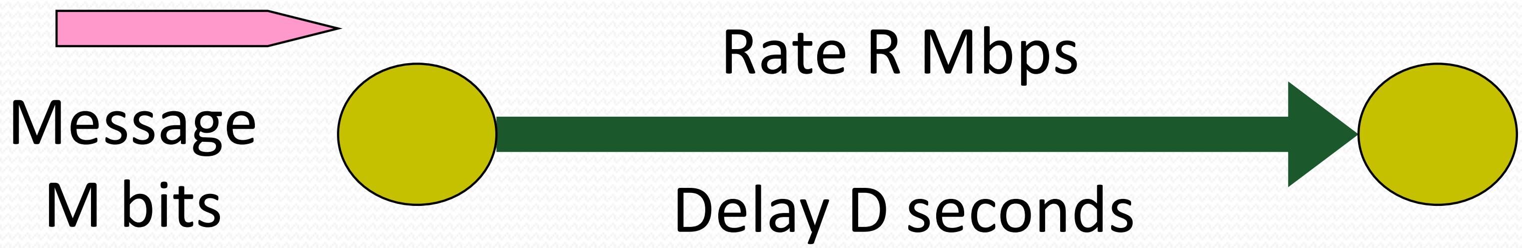
Example: Point-to-Point Protocol (PPP)

- IETF standard, used for dialup and leased lines

Flag 0x7E	(header)	Payload (variable)	(trailer)	Flag 0x7E

- Flag is special and indicates start/end of frame
- Occurrences of flag inside payload must be “stuffed”
  - Replace 0x7E with 0x7D, 0x5E
  - Replace 0x7D with 0x7D, 0x5D

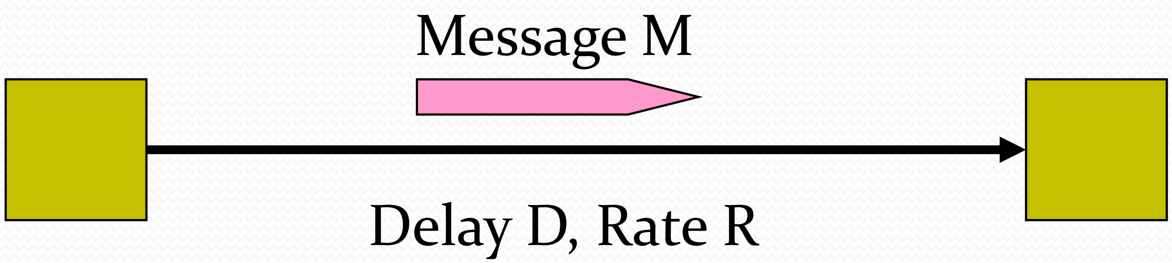
4. Model of a Link



- Abstract model is typically all we will need
  - What goes in comes out altered by the model
- Other parameters that are important:
  - The kind and frequency of errors
  - Whether the media is broadcast or not

Message Latency

- How long does it take to send a message?



- Two terms:
  - Propagation delay = distance / speed of light in media
    - How quickly a message travels over the wire
  - Transmission delay = message (bits) / rate (bps)
    - How quickly you can inject the message onto the wire
- Later we will see queuing delay …

Relationships

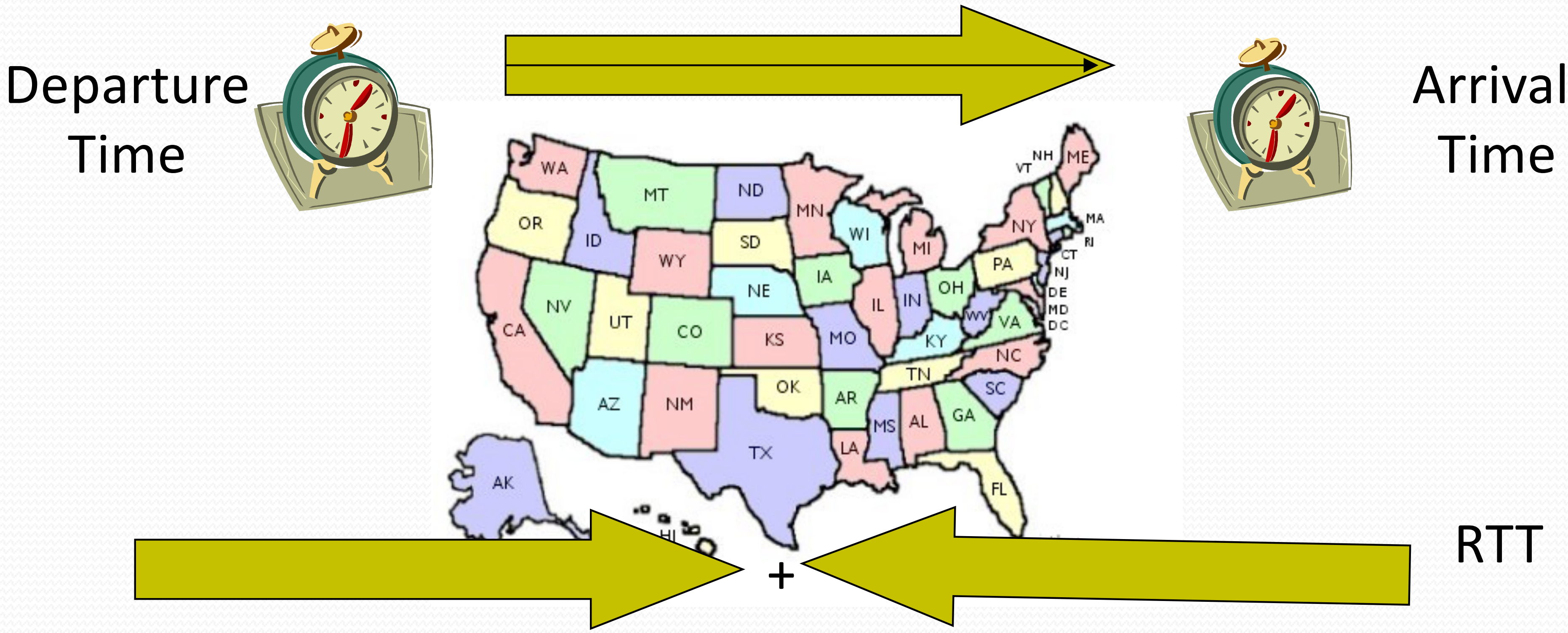
- Latency = Propagation + Transmit + Queue
- Propagation Delay = Distance/SpeedOfLight
- Transmit Time = MessageSize/Bandwidth

One-way Latency

- Dialup with a modem:
  - D = 10ms, R = 56Kbps, M = 1000 bytes
  - Latency = 10ms + (1000 x 8)/(56 x 1000) sec = 153ms!
- Cross-country with T3 (45Mbps) line:
  - D = 50ms, R = 45Mbps, M = 1000 bytes
  - Latency = 50ms + (1000 x 8) / (45 x 1000000) sec = 50ms!
- Either a slow link or long wire makes for large latency

Latency and RTT

- Latency is typically the one way delay over a link
  - Arrival Time - Departure Time



- The round trip time (RTT) is twice the one way delay
  - Measure of how long to signal and get a response

Throughput

- Measure of system's ability to "pump out" data
  - NOT the same as bandwidth
- Throughput = Transfer Size / Transfer Time
  - Eg, "I transferred 1000 bytes in 1 second on a 100Mb/s link"
    - BW?
    - Throughput?
- Transfer Time = SUM OF
  - Time to get started shipping the bits
  - Time to ship the bits
  - Time to get stopped shipping the bits

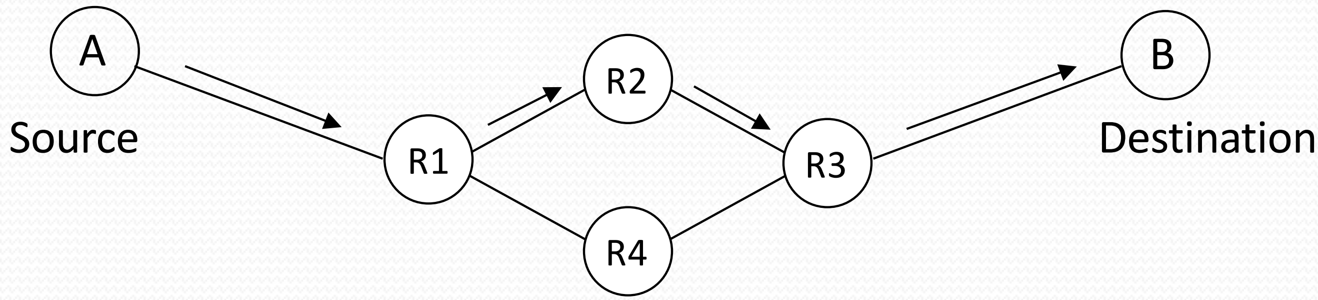
Messages Occupy "Space" On the Wire

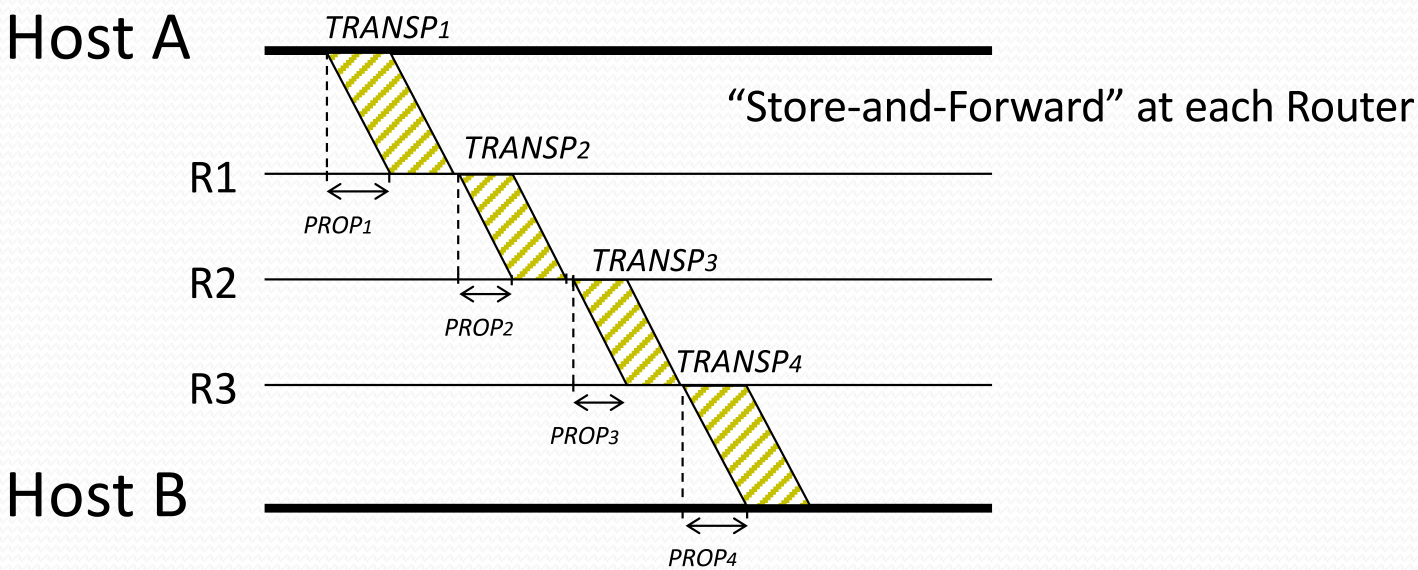
- Consider a 1b/s network.
  - How much space does 1 byte take?
- Suppose latency is 16 seconds.
  - How many bits can the network "store"
  - This is the BANDWIDTH-DELAY Product
  - Measure of "data in flight."
  - 1b/s * 16s = 16b
- Tells us how much data can be sent before a receiver sees any of it.
  - Twice B.D.P. tells us how much data we could send before hearing back from the receiver something related to the first bit sent.
  - Implications?

A More Realistic Example

BDP = 50ms * 100Mbps = 5Mb = 625KB

Packet Switching

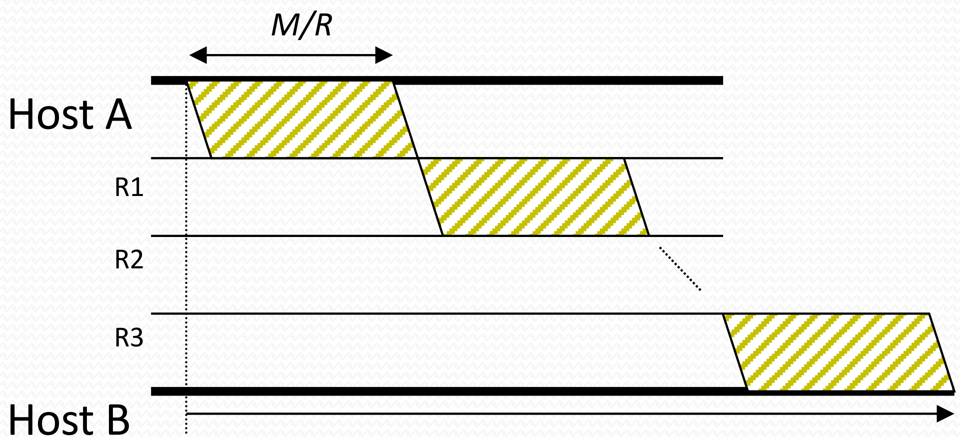




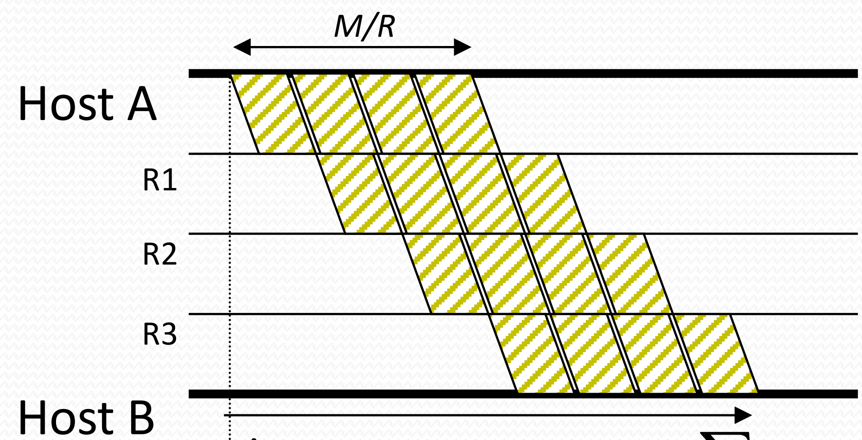
$$\text{Minimum end to end latency} = \sum_i (TRANSP_i + PROP_i)$$

Packet Switching

- *Why not send the entire message in one packet?*



$$\text{Latency} = \sum_i (PROP_i + M / R_i)$$

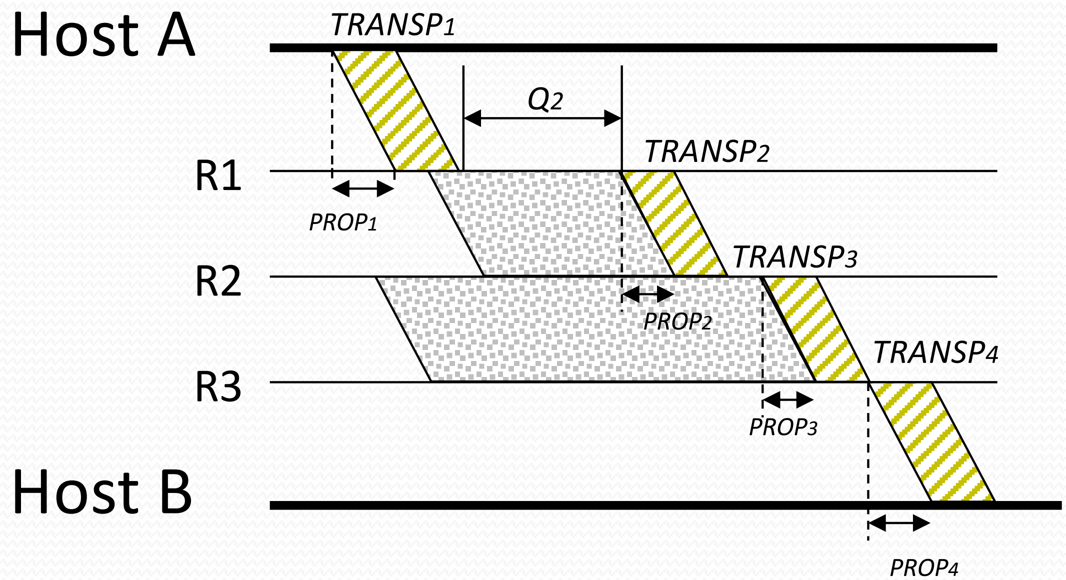


$$\text{Latency} = M / R_{\min} + \sum_i PROP_i$$

Breaking message into packets allows parallel transmission across all links, reducing end to end latency. It also prevents a link from being "hogged" for a long time by one message.

Packet Switching – *Queueing Delay*

Because the egress link is not necessarily free when a packet arrives, it may be queued in a buffer. If the network is busy, packets might have to wait a long time.



How can we determine the queueing delay?

$$\text{Actual end to end latency} = \sum_i (TRANSP_i + PROP_i + Q_i)$$

Part 1: Key Concepts

- We typically model links in terms of bandwidth and delay, from which we can calculate message latency.
- Different media have different properties that affect their performance as links.
- We need to encode bits into signals so that we can recover them at the other end of the channel.
- Framing allows complete messages to be recovered at the far end of the link.

Outline

Part 1. Physical/link layer

- Different types of media
- Encoding bits with signals
- Framing
- Model of a link

Part 2. Error detection and correction

- Hamming distance
- Parity, checksums, CRC, …

Part 2 – Error Detection and Correction

- Focus: How do we detect and correct messages that are garbled during transmission?

- The responsibility for doing this cuts across the different layers

Application
Presentation
Session
Transport
Network
Data Link
Physical

Errors and Redundancy

- Noise can flip some of the bits we receive
  - We must be able to detect when this occurs!
  - Why?
  - Who needs to detect it? (links, routers, OSs, or apps?)
- Basic approach: add redundant data
  - Error detection codes allow errors to be *recognized*
  - Error correction codes allow errors to be *repaired* too

Motivating Example

- A simple error detection scheme:
  - Just send two copies. Differences imply errors.
- **Question**: Can we do any better?
  - With less overhead
  - Catch more kinds of errors
- **Answer**: Yes – stronger protection with fewer bits
  - But we can't catch all inadvertent errors, nor malicious ones
- We will look at basic block codes
  - K bits in, N bits out is a (N, K) code
  - Simple, memoryless mapping

Detection vs. Correction

- Two strategies to correct errors:
  - Detect and retransmit, or Automatic Repeat reQuest. (ARQ)
  - Error correcting codes, or Forward Error Correction (FEC)
- Satellites, real-time media tend to use error correction
- Retransmissions typically at higher levels (Network+)
- **Question**: Which should we choose?

Detect or Correct?

- Advantages of Error Detection
  - Requires smaller number of bits/overhead.
  - Requires less/simpler processing.

- Advantages of Error Correction
  - Reduces number of retransmissions.

- Most data networks today use error detection, not error correction.

Retransmissions vs. FEC

- The better option depends on the kind of errors and the cost of recovery
- Example: Message with 1000 bits, Prob(bit error) 0.001
  - Case 1: random errors
  - Case 2: bursts of 1000 errors
  - Case 3: real-time application (teleconference)

Encoding to Detect Errors

- We use codes to help us detect errors.
- The set of possible messages is mapped by a function onto the set of codes.
- We pick the mapping function so that it is easy to detect errors among the resulting codes.
- Example: Consider the function that duplicates each bit in the message. E.g. the message 1011001 would be mapped to the code 11001111000011, and then transmitted by the sender. The receiver knows that bits always come in pairs. If the two bits in a pair are different, it declares that there was a bit error.
- Of course, this code is quite inefficient…

The Hamming Distance

- Errors must not turn one valid codeword into another valid codeword, or we cannot detect/correct them.
- Hamming distance of a code is the smallest number of bit differences that turn any one codeword into another
  - e.g, code 000 for 0, 111 for 1, Hamming distance is 3
- For code with distance d+1:
  - d errors can be detected, e.g, 001, 010, 110, 101, 011
- For code with distance 2d+1:
  - d errors can be corrected, e.g., 001 → 000

Hamming Distance

Number of bits that differ between two codes

e.g. 1 0 0 1 0 1 0 1

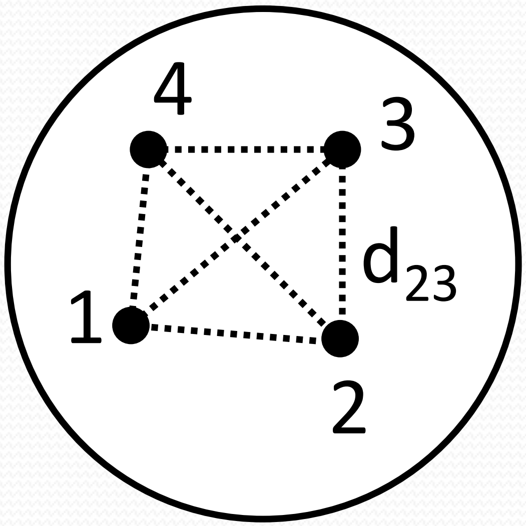
1 0 1 1 1 0 0 1

0 0 1 0 1 1 0 0 → HD=3

In our example code (replicated bits), all codes have at least two bits different from every other code. Therefore, it has a Hamming distance of 2.

Hamming Distance

Set of codes



HD = min (d_{ij})

To reliably detect a d-bit error: HD ≥ d+1

To reliably correct a d-bit error: HD ≥ 2d+1

Parity

- Start with n bits and add another so that the total number of 1s is even (even parity)
  - e.g. 0110010 → 01100101
  - Easy to compute as XOR of all input bits
- Will detect an odd number of bit errors
  - But not an even number
- Does not correct any errors

2D Parity

- Add parity row/column to array of bits
- Detects all 1, 2, 3 bit errors, and many errors with >3 bits.
- Corrects all 1 bit errors

0101001	1
1101001	0
1011110	1
0001110	1
0110100	1
1011111	0
1111011	0

Checksums

- Used in Internet protocols (IP, ICMP, TCP, UDP)
- **Basic Idea**: Add up the data and send it along with sum
- **Algorithm**:
  - *checksum* is the 1s complement of the 1s complement sum of the data interpreted 16 bits at a time (for 16-bit TCP/UDP checksum)
- **1s complement**: flip all bits to make number negative
  - Consequence: adding requires carryout to be added back

CRCs (Cyclic Redundancy Check)

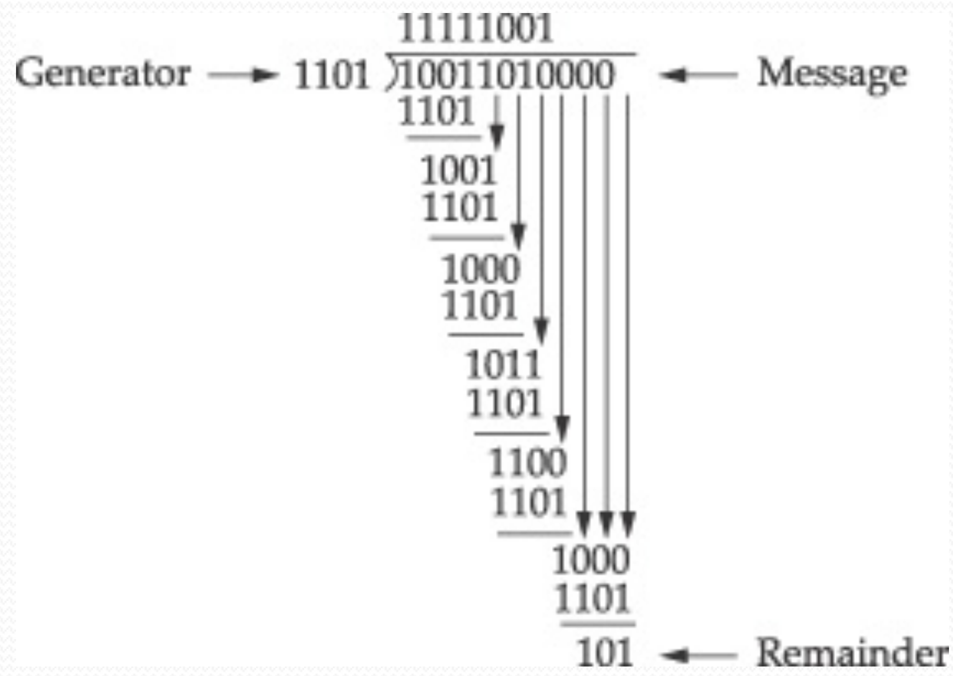
- Stronger protection than checksums
  - Used widely in practice, e.g., Ethernet CRC-32
  - Implemented in hardware (XORs and shifts)

- **Algorithm**: Given ***n*** bits of data, generate a ***k*** bit check sequence that gives a combined ***n*** + ***k*** bits that are divisible by a chosen divisor ***C***(***x***)

- Based on mathematics of finite fields
  - "numbers" correspond to polynomials, use modulo arithmetic
  - e.g, interpret 10011010 as $x^7 + x^4 + x^3 + x^1$

Example

- Message: 10011010
- Generator: 1101
- Divide 10011010000 by 1101
- Remainder: 101
- Message to be sent: 10011010101



Part 2: Key Concepts

- Redundant bits are added to messages to protect against transmission errors.
- Two recovery strategies are retransmissions (ARQ) and error correcting codes (FEC)
- The Hamming distance tells us how much error can safely be tolerated.