# COS 461: Networks and Distributed Computing

- Prof. Ed Felten
  - felten@cs.princeton.edu
- http://www.cs.princeton.edu/courses/cs461
- requirements
  - programs (in Java)
  - course project
  - midterm, final exams
- now … on to business

# Networking on a Shoestring

- have two wires between points A and B
- need to communicate (in both directions)
  - use one wire in each direction
- how to do it?

## Signaling

- use voltage to represent ones and zeroes
- two problems
  - no common ground level
  - timing

## Ground

- take average of recent voltage level
  - 0 if much below average
  - 1 if much above average

# Timing

- ideal: synchronized clocks

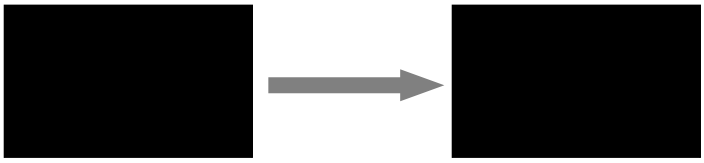
- reality: can't synchronize; clocks drift

# Timing

- strategy: run at approximately same speed
  - » How close do we have to be? (Answer later.)
- on seeing a transition, receiver re-adjusts its clock to the nearest half-tick
- adjust clock forward or backward, whichever is shorter

## Choosing Clock Speed

- electrical effects flatten and spread waveform as it travels down wire



- run as fast as possible without missing any transitions

## Coping with Clock Drift

- for correctness, receiver must not drift by more than half a cycle between transitions.
- if max time between transitions is K cycles, clock speed must be within factor of 1+1/2K.
- if no limit on K, no tolerance for drift

## Forcing Transitions

- idea: encode data before transmitting
  - code forces frequent transitions
- example: Manchester encoding

- tolerates 25% clock drift, but "wastes" half of bandwidth

## Other Encodings

- waste less bandwidth, but tolerate less drift
  - 5/4 encoding in book
  - generally a good trade
- other advantages of forcing transitions
  - keeps baseline voltage from drifting
  - detects broken wires

# Framing

- need to divide data stream into packets
  - variable length better than constant length
    - » avoid wasting bandwidth
    - » doesn't add much complexity
  - two approaches
    - » length field
    - » end marker

# Length Field Approach

- first 16 bits of packet give the length

- problem: error recovery

## End-Marker Approach

- special value marks end of packet
  - say it's 11111111
- solves synchronization problems
- but what if 11111111 occurs in data?
- solution: bit stuffing
  - if sender sees seven 1's in a row, insert a 0
  - receiver deletes a 0 that follows seven 1's

## Error Control

- real wires don't always transmit data correctly
  - electrical glitches
  - physical stresses and damage
- dealing with errors
  - detection
  - recovery

## Error Detection

- sender computes checksum, appends to packet
- receiver verifies checksum
- properties of a good checksum
  - always signal error if only a few bits corrupted
  - low probability of coincidental match if many bits corrupted
  - signals error if signal stuck on 0 or 1

## Checksums in Practice

- internet scheme
  - (roughly) take sum of 16-bit words in message
  - not very good, but fast to compute in software
- CRC (cyclic redundancy code)
  - based on polynomial arithmetic in finite fields
  - implement in hardware with shift register and a few XOR gates

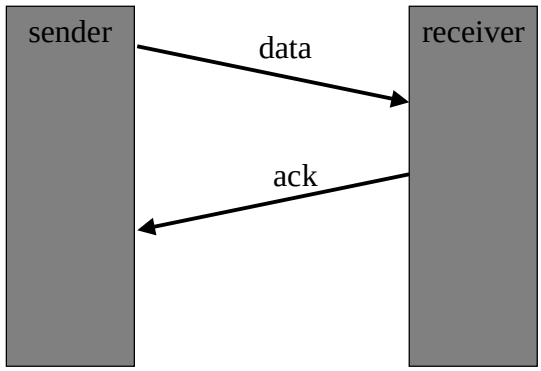
## Error Correcting Codes

- code message redundantly
- detect/correct errors that affect a few bits
- seldom used in practice
  - good at correcting a few bad bits, but errors tend to come in bunches
  - must be ready to recover from uncorrectable errors anyway
  - if errors are rare, better to handle other ways

## Recovering from Errors

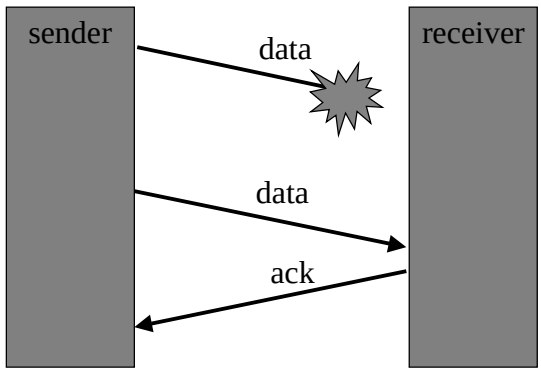
- first try: ask sender to retransmit message
  - but: if wire breaks, sender thinks things are OK
- second try: acknowledgements
  - receiver tells sender packet arrived safely
  - if no acknowledgement within a timeout period, sender retransmits packet

# Retransmission Scenario 1

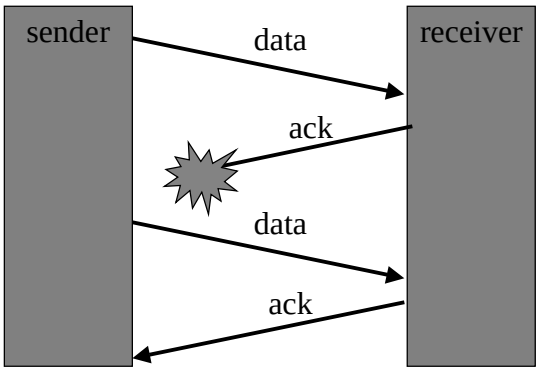





# Retransmission Scenario 2



# Retransmission Scenario 3






# Details

- sequence number to identify packets
  - how big are sequence numbers?
  - one bit is enough (alternating bit protocol)
- sender remembers packet until ack
- packet type distinguishes ack from data
  - or piggyback ack on data packet
- fancier schemes in another lecture

# Review

- speed limited by wire physics
- approximately synchronized clocks
- encoding to force frequent transitions
- sentinel value marks end of packet
  - bit stuffing if sentinel occurs in data
- checksum to detect errors
- timeout and retransmission to recover from errors