Hamming Code

• Gives a method for constructing a code with a distance of 3
  – Uses $n = 2^k - k - 1$, e.g., $n=4$, $k=3$
  – Put check bits in positions $p$ that are powers of 2, starting with position 1
  – Check bit in position $p$ is parity of positions with a $p$ term in their values

• Plus an easy way to correct [soon]
Hamming Code (2)

• Example: data=0101, 3 check bits
  – 7 bit code, check bit positions 1, 2, 4
  – Check 1 covers positions 1, 3, 5, 7
  – Check 2 covers positions 2, 3, 6, 7
  – Check 4 covers positions 4, 5, 6, 7

1 2 3 4 5 6 7
Hamming Code (3)

• Example: data=0101, 3 check bits
  – 7 bit code, check bit positions 1, 2, 4
  – Check 1 covers positions 1, 3, 5, 7
  – Check 2 covers positions 2, 3, 6, 7
  – Check 4 covers positions 4, 5, 6, 7

\[ \begin{align*}
0 & 1 \\
1 & 2 & 3 & 4 & 5 & 6 & 7
\end{align*} \]

\[ p_1 = 0 + 1 + 1 = 0, \quad p_2 = 0 + 0 + 1 = 1, \quad p_4 = 1 + 0 + 1 = 0 \]
Hamming Code (4)

- To decode:
  - Recompute check bits (with parity sum including the check bit)
  - Arrange as a binary number
  - Value (syndrome) tells error position
  - Value of zero means no error
  - Otherwise, flip bit to correct
Hamming Code (5)

• Example, continued

\[ \begin{array}{ccccccc}
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{array} \]

\[ p_1 = \]
\[ p_2 = \]
\[ p_4 = \]

Syndrome =
Data =
Hamming Code (6)

• Example, continued

\[ \begin{array}{cccccccc}
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{array} \]

\[ p_1 = 0+0+1+1 = 0, \quad p_2 = 1+0+0+1 = 0, \]
\[ p_4 = 0+1+0+1 = 0 \]

Syndrome = 000, no error
Data = 0 1 0 1
Hamming Code (7)

• Example, continued

\[ \rightarrow 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \]

\[ 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \]

\[ p_1 = \quad p_2 = \]
\[ p_4 = \]

Syndrome =

Data =
Hamming Code (8)

- Example, continued

\[ \rightarrow 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \]
\[ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \]

\[ p_1 = 0 + 0 + 1 + 1 = 0, \quad p_2 = 1 + 0 + 1 + 1 = 1, \]
\[ p_4 = 0 + 1 + 1 + 1 = 1 \]

Syndrome = 1 1 0, flip position 6
Data = 0 1 0 1 (correct after flip!)
Other Error Correction Codes

- Codes used in practice are much more involved than Hamming

- Convolutional codes (§3.2.3)
  - Take a stream of data and output a mix of the recent input bits
  - Makes each output bit less fragile
  - Decode using Viterbi algorithm (which can use bit confidence values)
Other Codes (2) – LDPC

• Low Density Parity Check (§3.2.3)
  – LDPC based on sparse matrices
  – Decoded iteratively using a belief propagation algorithm
  – State of the art today

• Invented by Robert Gallager in 1963 as part of his PhD thesis
  – Promptly forgotten until 1996 ...

Source: IEEE GHN, © 2009 IEEE
Detection vs. Correction

• Which is better will depend on the pattern of errors. For example:
  – 1000 bit messages with a bit error rate (BER) of 1 in 10000

• Which has less overhead?
Detection vs. Correction

• Which is better will depend on the pattern of errors. For example:
  – 1000 bit messages with a bit error rate (BER) of 1 in 10000

• Which has less overhead?
  – It still depends! We need to know more about the errors
Detection vs. Correction (2)

1. Assume bit errors are random
   – Messages have 0 or maybe 1 error

   • Error correction:
     – Need ~10 check bits per message
     – Overhead:

   • Error detection:
     – Need ~1 check bits per message plus 1000 bit retransmission 1/10 of the time
     – Overhead:
2. Assume errors come in bursts of 100
   – Only 1 or 2 messages in 1000 have errors

• Error correction:
  – Need >>100 check bits per message
  – Overhead:

• Error detection:
  – Need 32? check bits per message plus 1000 bit resend 2/1000 of the time
  – Overhead:
Detection vs. Correction (4)

• Error correction:
  – Needed when errors are expected
  – Or when no time for retransmission

• Error detection:
  – More efficient when errors are not expected
  – And when errors are large when they do occur
Error Correction in Practice

• Heavily used in physical layer
  – LDPC is the future, used for demanding links like 802.11, DVB, WiMAX, LTE, power-line, ...
  – Convolutional codes widely used in practice

• Error detection (w/ retransmission) is used in the link layer and above for residual errors

• Correction also used in the application layer
  – Called Forward Error Correction (FEC)
  – Normally with an erasure error model
  – E.g., Reed-Solomon (CDs, DVDs, etc.)
• Multiplexing is the network word for the sharing of a resource

• Classic scenario is sharing a link among different users
  – Time Division Multiplexing (TDM)
  – Frequency Division Multiplexing (FDM)
Time Division Multiplexing (TDM)

- Users take turns on a fixed schedule
Frequency Division Multiplexing (FDM)

• Put different users on different frequency bands

Overall FDM channel
TDM versus FDM

- In TDM a user sends at a high rate a fraction of the time; in FDM, a user sends at a low rate all the time.
TDM versus FDM (2)

- In TDM a user sends at a high rate a fraction of the time; in FDM, a user sends at a low rate all the time.
TDM/FDM Usage

• Statically divide a resource
  – Suited for continuous traffic, fixed number of users

• Widely used in telecommunications
  – TV and radio stations (FDM)
  – GSM (2G cellular) allocates calls using TDM within FDM
Multiplexing Network Traffic

- Network traffic is **bursty**
  - ON/OFF sources
  - Load varies greatly over time

![Graph showing bursty network traffic](image)
Multiplexing Network Traffic (2)

- Network traffic is **bursty**
  - Inefficient to always allocate user their ON needs with TDM/FDM
Multiplexing Network Traffic (3)

- **Multiple access** schemes multiplex users according to their demands – for gains of statistical multiplexing.

Two users, each need $R$

Together they need $R' < 2R$
Multiple Access

- We will look at two kinds of multiple access protocols
  1. Randomized. Nodes randomize their resource access attempts
     - Good for low load situations
  2. Contention-free. Nodes order their resource access attempts
     - Good for high load or guaranteed quality of service situations
Topic

- How do nodes share a single link? Who sends when, e.g., in WiFi?
  - Explore with a simple model

- Assume no-one is in charge; this is a distributed system
Topic (2)

- We will explore random **multiple access control** (MAC) protocols
  - This is the basis for **classic Ethernet**
  - Remember: data traffic is bursty
ALOHA Network

• Seminal computer network connecting the Hawaiian islands in the late 1960s
  – When should nodes send?
  – A new protocol was devised by Norm Abramson ...
ALOHA Protocol

• Simple idea:
  – Node just sends when it has traffic.
  – If there was a collision (no ACK received) then wait a random time and resend
• That’s it!
ALOHA Protocol (2)

- Some frames will be lost, but many may get through…
- Good idea?
ALOHA Protocol (3)

• Simple, decentralized protocol that works well under low load!

• Not efficient under high load
  – Analysis shows at most 18% efficiency
  – Improvement: divide time into slots and efficiency goes up to 36%

• We’ll look at other improvements
Classic Ethernet

- ALOHA inspired Bob Metcalfe to invent Ethernet for LANs in 1973
  - Nodes share 10 Mbps coaxial cable
  - Hugely popular in 1980s, 1990s
CSMA (Carrier Sense Multiple Access)

• Improve ALOHA by listening for activity before we send (Doh!)
  – Can do easily with wires, not wireless

• So does this eliminate collisions?
  – Why or why not?
CSMA (2)

- Still possible to listen and hear nothing when another node is sending because of delay
CSMA (3)

- CSMA is a good defense against collisions only when BD is small
CSMA/CD (with Collision Detection)

• Can reduce the cost of collisions by detecting them and aborting (Jam) the rest of the frame time
  – Again, we can do this with wires

[Diagram showing three devices connected with Xs and a Jam! icon at the end of each line]
CSMA/CD Complications

• Want everyone who collides to know that it happened
  – Time window in which a node may hear of a collision is 2D seconds
CSMA/CD Complications (2)

- Impose a minimum frame size that lasts for 2D seconds
  - So node can’t finish before collision
  - Ethernet minimum frame is 64 bytes
CSMA “Persistence”

• What should a node do if another node is sending?

• Idea: Wait until it is done, and send
CSMA “Persistence” (2)

• Problem is that multiple waiting nodes will queue up then collide
  – More load, more of a problem
CSMA “Persistence” (3)

• Intuition for a better solution
  – If there are N queued senders, we want each to send next with probability 1/N

Send p=½

Whew

Send p=½
Binary Exponential Backoff (BEB)

• Cleverly estimates the probability
  – 1st collision, wait 0 or 1 frame times
  – 2nd collision, wait from 0 to 3 times
  – 3rd collision, wait from 0 to 7 times ...

• BEB doubles interval for each successive collision
  – Quickly gets large enough to work
  – Very efficient in practice
Classic Ethernet, or IEEE 802.3

- Most popular LAN of the 1980s, 1990s
  - 10 Mbps over shared coaxial cable, with baseband signals
  - Multiple access with “1-persistent CSMA/CD with BEB”
Ethernet Frame Format

- Has addresses to identify the sender and receiver
- CRC-32 for error detection; no ACKs or retransmission
- Start of frame identified with physical layer preamble
Modern Ethernet

• Based on switches, not multiple access, but still called Ethernet
  – We’ll get to it in a later segment