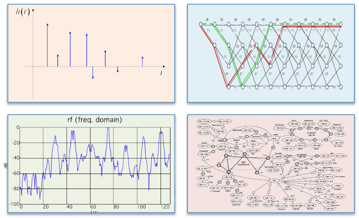


INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

6.02 Fall 2011
Lecture #1

- Motivation
- Information & entropy
- Huffman codes

Duties	Name	Email at mit.edu	Office	Phone
Lectures (office hrs by appointment)	Hari Balakrishnan	hari	32-G940	x3-8713
	George Verghese	verghese	10-140K	x3-4612
Recitations	Paul Ampadu	ampadu	...	x...
	Karl Berggren	berggren	36-219	x4-0272
	Sidhant Misra	sidhant	...	...
TAs (check lab hours link)	Mukul Agarwal	magar	--	--
	Jason Cloud	jcloud	--	--
	Shuo Deng	shuodeng	--	--
	Lyla Fischer	fischerl	--	--
	Rui Li	rui	--	--
	Ruben Madrigal	madrigal	--	--
	Surapap Rayanokorn	surapap	--	--
	Xiawa Wang	xiawaw	--	--

6.02
Fall 2011

Home
Announcements

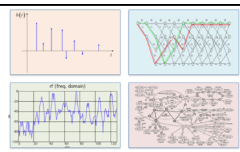
Handouts
»Lectures
»PSets
»Tutorial Problems

*MIT cert required
* On-line grades
* PSets:
1,
* Help queue
* Lab Hours
* Staff only

Course info
Course calendar
Course description

SW installation
Python
Numpy
Matplotlib

Previous terms



INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

Week of September 5, 2011

http://mit.edu/6.02

- This week's to-do list:
  - Wed: First Lecture
  - Thu: First Recitation
- Next week's to-do list:
  - Wed: PSet #1 due
  - Thu, Fri: Lab checkoff with your interviewer
- The first meeting of 6.02 will be at 2p in room **34-101** on Wednesday, 9/7. Consult the Course Calendar for a detailed schedule of lectures, recitations, labs and quizzes. Recitation meetings start Thursday, 9/8.
- **Recitation assignments:** As a starting point, please attend the section assigned to you by the Registrar.
- Please take a moment to read the Course Info page which describes course mechanics and policies.

See Announcements to read previous messages.

Questions?

- Email 6.02-staff@mit.edu
- Or better still, sign up for 6.02 Piazza at http://piazza.com/class#fall2011/602
- BTW: PSet #1 is now online, due by 9/15 at 6 am ("midnight" Wed 9/14)

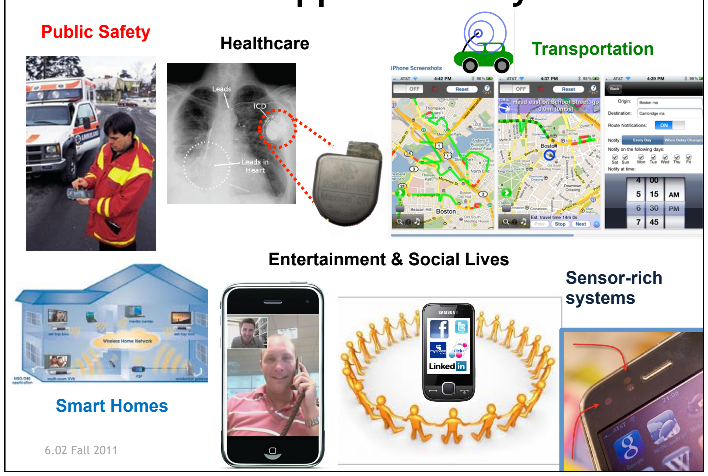
Greatest Engineering Achievements OF THE 20TH CENTURY

National Academy of Engineering (http://www.greatachievements.org/)

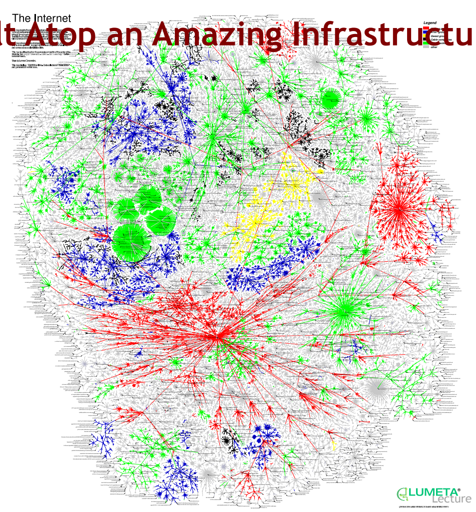
Welcome!
How many of the 20th century's greatest engineering achievements will you use today? A car? Computer? Telephone? Explore our list of the top 20 achievements and learn how engineering shaped a century and changed the world.

1. Electrification
2. Automobile
3. Airplane
4. Water Supply and Distribution
5. Electronics
6. Radio and Television
7. Agricultural Mechanization
8. Computers
9. Telephone
10. Air Conditioning and Refrigeration
11. Highways
12. Spacecraft
13. Internet
14. Imaging
15. Household Appliances
16. Health Technologies
17. Petroleum and Petrochemical Technologies
18. Laser and Fiber Optics
19. Nuclear Technologies
20. High-performance Materials

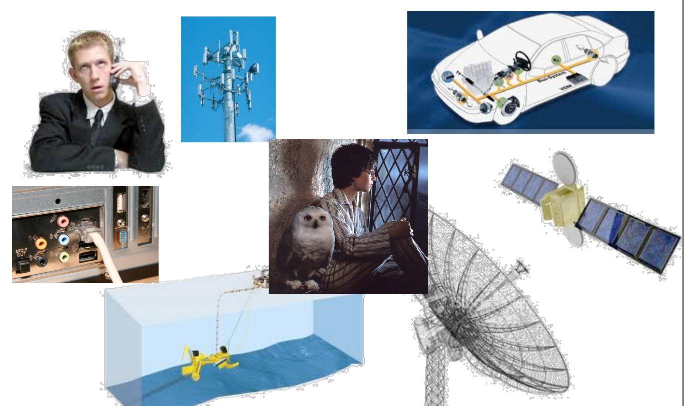
Networked Apps are Everywhere



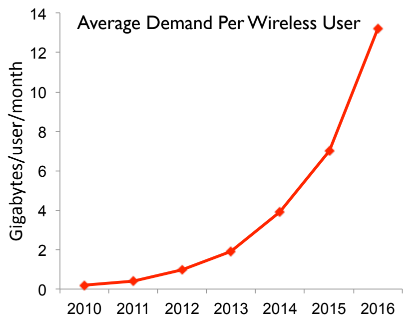
Built Atop an Amazing Infrastructure



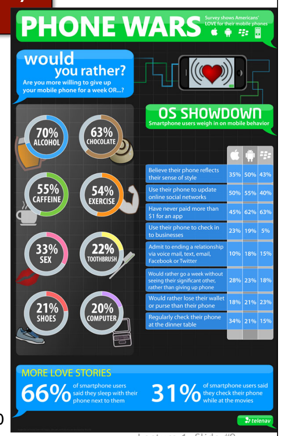
Physical Communication Channels



5B mobile phones in the world
Exceeds #people with shoes, toilets, electricity!
1B with "broadband" capability

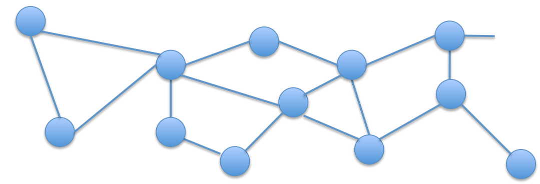


Projection: Rysavy Research, 2010



Survey by TeleNav

6.02 Roadmap



Understanding information, encoding messages
- Source coding for compression
- Communication abstractions: packets, bits, signals

Point-to-point communication channels:
- *Bits*: Noise, bit errors, error correction
- *Signals*: Frequency content & response, LTI model, modulation/demodulation

Multi-node and multi-hop networks:
- *Packets*: MAC protocols, packet switching, routing, reliable transport

Engineering Goals

If you were asked to design a communication network, what would your most important goals be?

1. Reliability
2. Efficient sharing (→ cost-effectiveness)
3. Scalability
4. Energy-efficiency, …

INFORMATION

Information Resolves Uncertainty

Information is a mathematical quantity that depends on the probability of occurrence of a particular event, which we might think of as a sequence of one or more *symbols*.

Nice: *"It was 75 degrees F in Boston on Jan 30"*

Awful: *"It was 30 degrees F in Boston on Jan 30"*

Which statement conveys more information?

High probability of event → less information
Low probability of event → more information

Measuring Information

Based on work by Hartley, Claude Shannon, the father of information theory, defined the information, *I*, associated with an event (message) of probability p as

$$I = \log_2\left(\frac{1}{p}\right)$$

The unit of measurement is the bit (binary digit: "0" or "1").

1 bit corresponds to p = ½, e.g., the probability of a heads or tails when flipping a fair coin.

This lines up with our intuition: we can encode the result of a single coin flip using just 1 bit: say "1" for heads, "0" for tails. Encoding 25 flips of a fair coin requires 25 bits.

Information in Equi-Probable Events

Q: Suppose we have N equi-probable events. How much information have you learned if tell you that a specific event occurred?

A: $I = \log_2 (1 / (1/N)) = \log_2 N$ bits.

Q: Suppose we have N equi-probable events. How much information have you learned if tell you that one of M equally probable events occurred from this set of N events?

A: P(the event that occurred being one of M events) = M/N
Therefore, $I = \log_2 (1 / (M/N)) = \log_2 (N/M)$ bits.

Information: A measure of the uncertainty of an event.

Examples

We're drawing cards at random from a standard 52-card deck:

Q. If I tell you the card is a ♤, how many bits of information have you received?

A. We've gone from N=52 possible cards down to M=13 possible cards, so the amount of info received is $\log_2(52/13) = 2$ bits.

This makes sense, we can encode one of the 4 (equally probable) suits using 2 bits, e.g., 00=♡, 01=♢, 10=♧, 11=♤.

Q. If instead I tell you the card is a 7, how much info?

A. N=52, M=4, so info = $\log_2(52/4) = \log_2(13) = 3.7$ bits

Example (cont'd.)

Q. If I tell you the card is the 7 of spades, how many bits of information have you received?

A. We've gone from N=52 possible cards down to M=1 possible cards, so the amount of info received is $\log_2(52/1)$ = 5.7 bits.

Note that if the events are *independent,* then information is additive (5.7 = 3 + 2.7)!

But additivity holds only when the separate pieces of information are independent: P(A and B) = P(A)P(B)

Expected Information: Entropy

Now consider a message transmitting the outcome of an event that has a set of possible outcomes, where we know the probability of each outcome.

Formally, model a random variable X with possible values $\{x_1, \ldots, x_n\}$ and their associated probabilities $p(x_1), \ldots, p(x_n)$.

The *entropy* H of a discrete random variable X is the expected value of the information content of X:

$$H(X) = E(I(X)) = \sum_{i=1}^{n} p(x_i)\log_2\left(\frac{1}{p(x_i)}\right)$$

Okay, why do we care about entropy?

Entropy tells us the average amount of information that must be delivered in order to resolve all uncertainty.

Shannon showed that entropy is a *lower bound* on the number of bits that must, on average, be used to encode our messages.

Achieving the entropy bound is the "gold standard" for an encoding: entropy gives us a metric to measure encoding effectiveness.

SOURCE CODES (Or, COMPRESSION)

Fixed-length Encodings

An obvious choice for encoding equally probable outcomes is to choose a fixed-length code that has enough sequences to encode the necessary information

- 96 printing characters → 7-bit ASCII
- Unicode characters → UTF-16
- 10 decimal digits → 4-bit BCD (binary coded decimal)

Fixed-length codes have some advantages:

- They are "random access" in the sense that to decode the n^{th} message symbol one can decode the n^{th} fixed-length sequence without decoding sequence 1 through n-1.
- Table lookup suffices for encoding and decoding

Improving on Fixed-length Encodings

$choice_i$	p_i	$log_2(1/p_i)$
"A"	1/3	1.58 bits
"B"	1/2	1 bit
"C"	1/12	3.58 bits
"D"	1/12	3.58 bits

The expected information content in a choice is given by the entropy:
= (.333)(1.58) + (.5)(1) + (2)(.083)(3.58) = 1.626 bits

Can we find an encoding where transmitting 1000 choices requires 1626 bits on the average?

The "natural" fixed-length encoding uses two bits for each choice, so transmitting the results of 1000 choices requires 2000 bits.

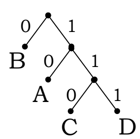
Variable-length encodings

(David Huffman, MIT 1950)

Use shorter bit sequences for high probability choices, longer sequences for less probable choices

$choice_i$	p_i	*encoding*
"A"	1/3	10
"B"	1/2	0
"C"	1/12	110
"D"	1/12	111

BC A BA D
011010010111



Huffman Code Tree

Expected length
=(.333)(2)+(.5)(1)+(2)(.083)(3)
= 1.666 bits

Transmitting 1000 choices takes an average of 1666 bits… better but not optimal

Another Variable-length Code (not!)

Here's an alternative variable-length for the example on the previous page:

Letter	*Encoding*
A	0
B	1
C	00
D	01

Why isn't this a workable code?

The expected length of an encoded message is

(.333+.5)(1) + (.083 + .083)(2) = 1.22 bits

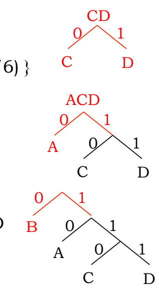
which even beats the entropy bound ☺

Huffman's Coding Algorithm

- Begin with the set S of symbols to be encoded as binary strings, together with the probability p(s) for each symbol s in S. The probabilities sum to 1 and measure the frequencies with which each symbol appears in the input stream. In the example from the previous slide, the initial set S contains the four symbols and their associated probabilities from the table.
- Repeat the following steps until there is only 1 symbol left in S:
  - Choose the two members of S having lowest probabilities. Choose arbitrarily to resolve ties.
  - Remove the selected symbols from S, and create a new node of the decoding tree whose children (sub-nodes) are the symbols you've removed. Label the left branch with a "0", and the right branch with a "1".
  - Add to S a new symbol that represents this new node. Assign this new symbol a probability equal to the sum of the probabilities of the two nodes it replaces.

Huffman Coding Construction

- Initially S = { (A, 1/3) (B, 1/2) (C, 1/12) (D, 1/12) }
- First iteration
  - Symbols in S with lowest probabilities: C and D
  - Create new node
  - Add new symbol to S = { (A, 1/3) (B, 1/2) (CD, 1/6) }
- Second iteration
  - Symbols in S with lowest probabilities: A and CD
  - Create new node
  - Add new symbol to S = { (B, 1/2) (ACD, 1/2) }
- Third iteration
  - Symbols in S with lowest probabilities: B and ACD
  - Create new node
  - Add new symbol to S = { (BACD, 1) }
- Done



Huffman Codes – the final word?

- Given static symbol probabilities, the Huffman algorithm creates an optimal encoding when each symbol is encoded separately and symbols are from an iid distribution. (Optimal ≡ no other encoding will have a shorter expected message length)
- Huffman codes have the biggest impact on average message length when some symbols are substantially more likely than other symbols.
- You can improve the results by adding encodings for symbol pairs, triples, quads, etc. From example code:
  Pairs: 1.646 bits/sym, Triples: 1.637, Quads 1.633, …
  But the number of possible encodings quickly becomes intractable.
- Symbol probabilities change message-to-message, or even within a single message.
- Can we do adaptive variable-length encoding?
  - Tune in next time!