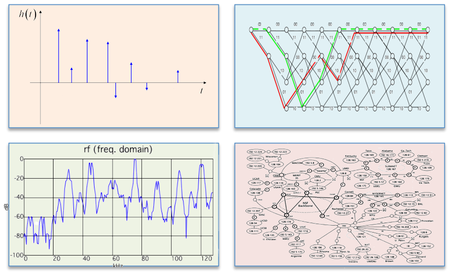


INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

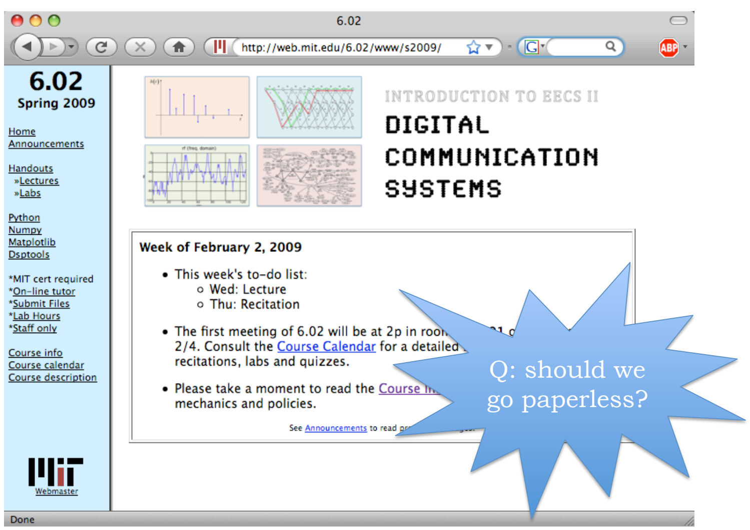
6.02 Spring 2009
Lecture #1

- Introductions, where to find info
- Engineering goals for comm systems
- Analog woes, the digital abstraction
- Basic recipes for sending info

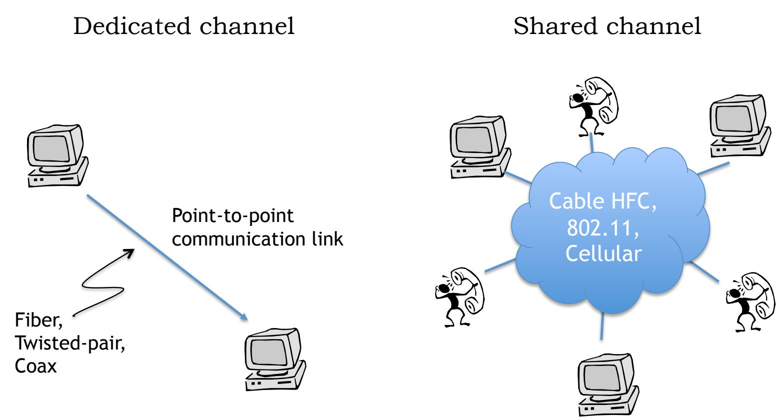
Staff Introductions

Duties	Name	Email at mit.edu	Office	Phone
Lectures	Hari Balakrishnan	hari	32-G940	x3-8713
	Chris Terman	cjt	32-G790	x3-6038
	Jacob White	white	36-817	x3-2543
Recitations	Vladimir Stojanovic	vlada	38-260	x4-4913
	Chris Terman	cjt	32-G790	x3-6038
	Mythili Vutukuru	mythili	32-G982	x3-7341
TAs	Micah Brodsky	micahbro	--	--
	Yajun Fang	yjfang	--	--
	Philip Godoy	godoy	--	--
	Yunjie Ma	yunjiema	--	--
	Lavanya Sharan	l_sharan	--	--
Admin	Chris Terman	cjt	32-G790	x3-6038

http://web.mit.edu/6.02/www/s2009

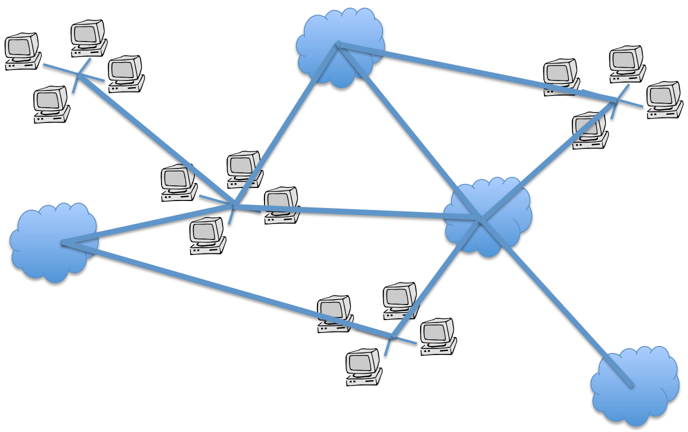


Digital Communication Links



Digital Communication Networks

Hierarchical, multi-hop networks



Design Criteria

- Engineering involves making design decisions & tradeoffs, so we'll have to figure out

  What's important, relative priorities
  How to measure success (design metrics)

- Communications System design criteria:

  Reliability
  Scalability
  Performance
  Cost

System Reliability

- Design engineers:
  - Low MTBF components, failure prediction
  - Easy to identify and fix problems
    - Remote observability and controllability
  - Replace/expand/evolve network incrementally
  - Defend against malicious users
- Redundancy
  - No single point of failure
  - "Fail soft" – degradation, not failure
  - Automated adaptation to component failures
- Users:
  - High availability
  - Accurate delivery of messages
    - Failing that, detection of failure and meaningful feedback

System Scalability

- Enable incremental build-out
  - increase in usage involves incremental costs (both at edges and in interior of network)
  - Address bottlenecks without fundamental changes
- Economies of scale
  - Larger number of users → less cost/user
  - "lose money on each customer, but make it up in volume"
- Slow growth of scale factors (N = number of users)

  $2^N > N^2 > N \log(N) > N > \log(N) >$ constant

  Decentralized rather than centralized

System Performance

- Design Engineers
  - Utilization
  - Minimize protocol overhead
  - Quality of Service (performance tiers)
- User
  - Throughput (guaranteed minimum)
    - Opportunistic improvements
  - Latency (guaranteed maximum)
    - One-way, round-trip
  - Isochrony?

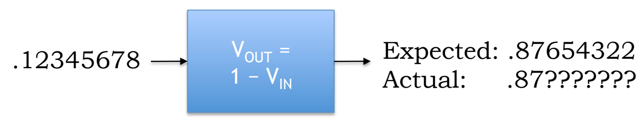
System Costs

- NRE (non-recurring expenses, ie, one-time costs)
- Basic infrastructure
- Per connection
- Per message transported
- Economies of scale, amortization

Why Digital?

- Seems like sending information using, say, voltages on a wire would be straightforward. Suppose the output range of the transmitter was 0V to 1V.
  - Sending char N of 128 possible chars: xmit (N/128)V.
  - Sending int N of 2^{16} possible ints: xmit (N/65536)V.
  - Sending int N of 2^{32} possible ints: xmit (N/4294967296)V
  - Sending music: xmit analog waveform
- What's the problem?
  - Nothing! At least in the ideal world…
  - In the real world where we live, it's a different story

Analog Woes

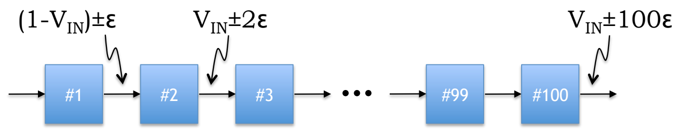


The actual value of V_{OUT} depends on many factors:

- Manufacturing tolerance of internal components
- Environmental factors (temp, power supply voltage)
- External influences (EM effects that affect voltages)
- How long we're willing to wait
- How much we're willing to spend

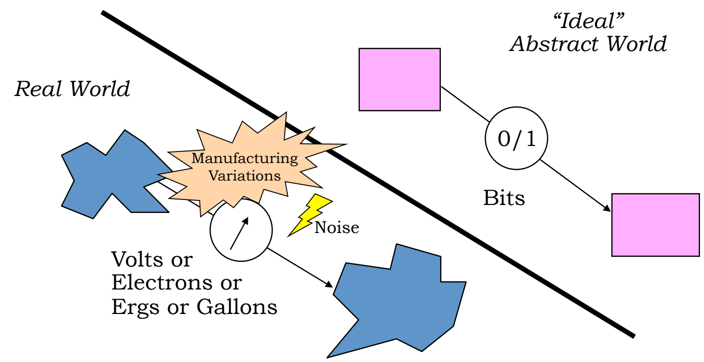
Truth in advertising: $V_{OUT} = (1 - V_{IN}) \pm \varepsilon$ ← If we call it ε maybe it'll seem small ☺

Analog Errors Accumulate



- If, say, ε = 1%, then result might be 100% off (urk!)
- Accumulation is good for money, bad for errors
- As system builders we want to guarantee output without having to worry about exact internal details
  - Bound number of processing stages in series OR
  - Figure out a way to eliminate errors at each processing stage. So how do we know *which part of the signal is message and which is error?*

The Digital Abstraction

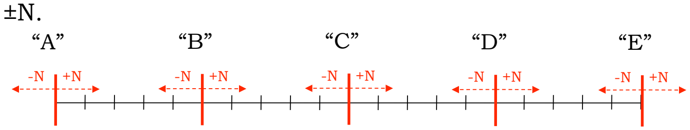


Keep in mind that the world is not digital, we would simply like to engineer it to behave that way. Furthermore, we must use real physical phenomena to implement digital designs!

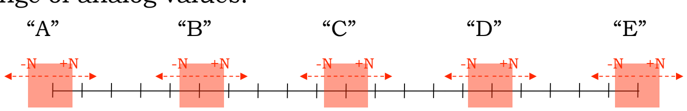
Digital Signaling: Transmitting

To ensure we can distinguish signal from noise, we'll encode information using a fixed set of discrete values called *symbols*.

Given a bound N on the size of possible errors, if the analog representations for the symbols are chosen to be at least 2N apart, we should be able to detect and eliminate errors of up to ±N.

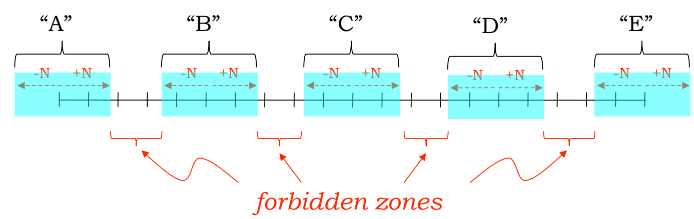


Since we will use non-ideal components in the transmitter, we allow each transmitted symbol to be represented by a (small) range of analog values.



Digital Signaling: Receiving

Since the channel/wire are imperfect and we will use non-ideal components in the receiver, we require the receiver to accept a larger range of analog values for each symbol.



The *forbidden zones* between symbols are ranges of received values that are not required to be mapped to a specific symbol (i.e., the receiver is allowed to map voltages in the forbidden zones however it wants – it's not even required to be monotonic or deterministic for these inputs). Necessary?

Forbidden Zones

- Forbidden zones are an essential element of the digital abstraction.
- How to digitize analog signal:
  - Build an analog-in, digital-out comparator that determines if the input is > a specific threshold (e.g,. using a high-gain opamp if we're using voltages)
  - Set the threshold in the middle of the forbidden zone
  - Maps larger-than-required range of inputs to valid outputs, but that's okay
- Since manufacturing and environment effects will cause the exact threshold to vary slightly receiver-to-receiver, we use the forbidden zone to give us the "elbow room" we need to make low-cost, high-speed receivers.

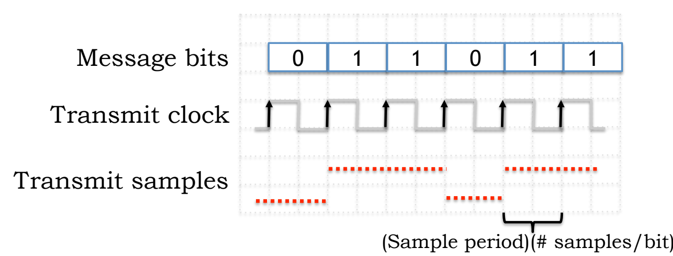
Communicating Information

- For simplicity, we'll transmit messages that are sequences of binary digits (aka bits). We'll label the digits 0 and 1.
- We'll work in *discrete time*, i.e., we'll transmit sequences of samples with a specific time interval between samples (aka the sample period).
  - Analog samples may be any value in the signaling range
  - Digitized samples may have only specific discrete values
- Transmitter: given a sequence of bits, generate a sequence (often longer) of analog samples
- Receiver: given a sequence of analog samples, recover original sequence of bits.

Engineering Choices

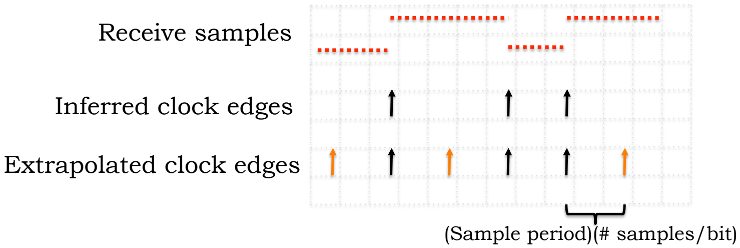
- Choose analog signaling ranges for 0 and 1.
  - How much noise? How big does f.z. have to be?
  - For example, transmit voltage sample $v \le -0.5V$ for 0 and $v \ge 0.5V$ for 1. Use 0V as receiver threshold.
- Choose sample period
  - Smaller is usually better
  - Smaller is harder, more costly, more power-hungry
- Choose how many samples to send for each bit
  - Too few: not enough info for receiver to make decisions
  - Too many: waste of transmission capacity
- Making choices involves *engineering tradeoffs*
  - Determined by goals, priorities
  - This is what makes engineering fun!

Transmitting Information



- Periodic events are timed by a clock signal
  - Sample period is controlled by the sample clock
  - Transmit clock is a submultiple of the sample clock
- Can receiver do its job if we only send samples and not the transmit clock?
  - Save a wire and the power needed to drive clock signal

Clock Recovery @ Receiver



- Receiver can infer presence of clock edge every time there's a transition in the received samples.
- Using sample period, extrapolate remaining edges
  – Now know first and last sample for each bit
- Choose "middle" sample to determine message bit

Two Issues for Recitation

- Don't want receiver to extrapolate over too long an interval
  – Differences in xmit & rcv clock periods will eventually cause receiver to mis-sample the incoming waveform
  – Fix: ensure transitions every so often, even if transmitting all 0's or all 1's (key idea: recoding)
- If recovered message bit stream represents, say, 8-bit blocks of ASCII characters, how does receiver determine where the blocks start?
  – Need out-of-band information about block starts
  – Fix: use special bit sequences to periodically synchronize receiver's notion of block boundaries. These sync sequences must be unique (i.e., distinguishable from ordinary message traffic).

Summary

- Design goals: reliability, scalability, performance, cost
- Analog signaling has issues
  – Real-world circuits & channels introduce errors
  – Errors accumulate at each processing step
- Digital Abstraction
  – Convention for analog signaling that lets us distinguish message from errors; restore signal at each step
  – Noise margins and forbidden zones
  – Recover digital data by comparing against threshold
- Receiver design
  – We don't send xmit clock so receiver does clock recovery
  – Determine bit from samples in "middle" of bit cell