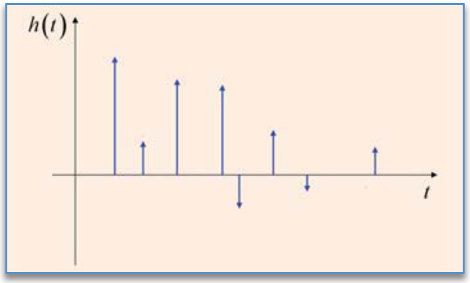
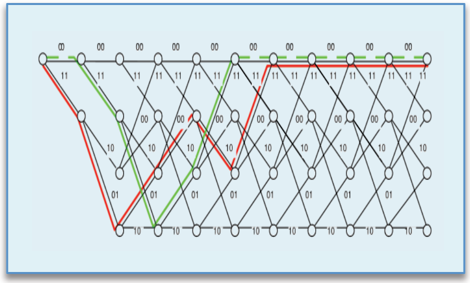
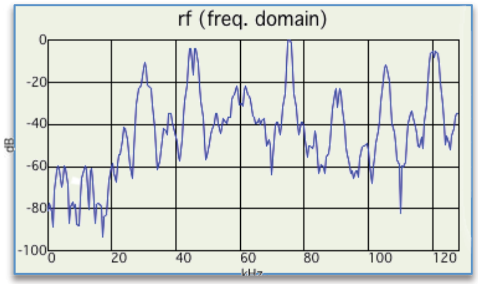
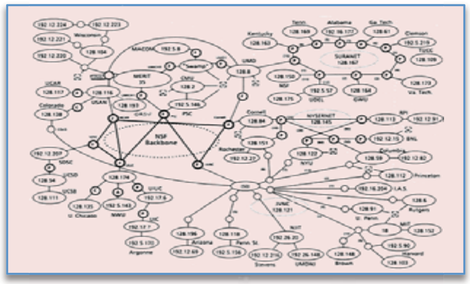




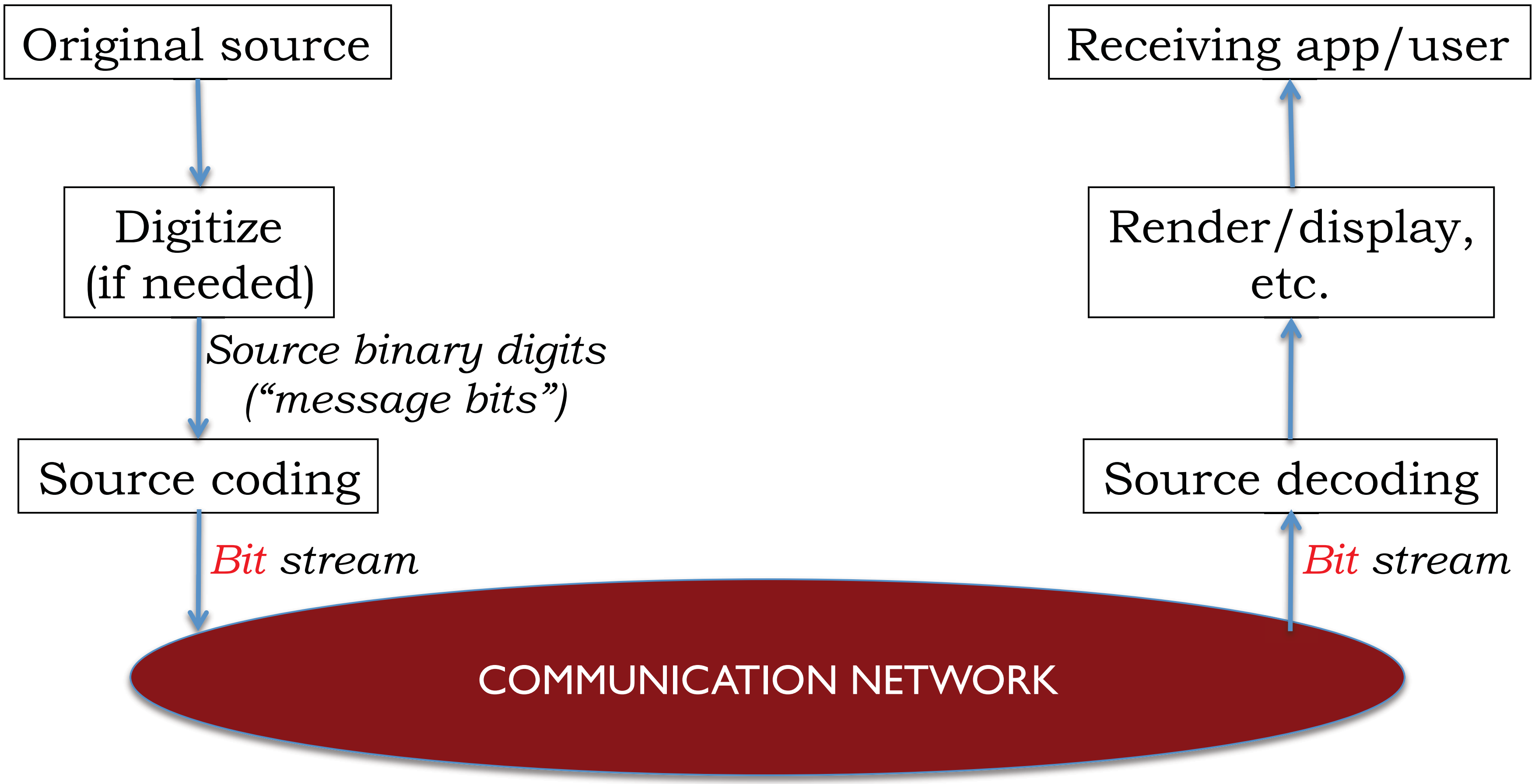
INTRODUCTION TO EECS II

DIGITAL COMMUNICATION SYSTEMS

6.02 Fall 2012
Lecture #3

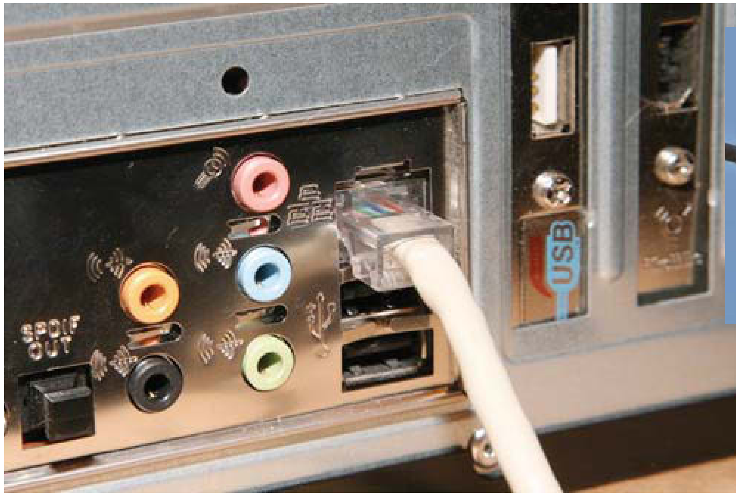
- Communication network architecture
- Analog channels
- The digital abstraction
- Binary symmetric channels
- Hamming distance
- Channel codes

The System, End-to-End



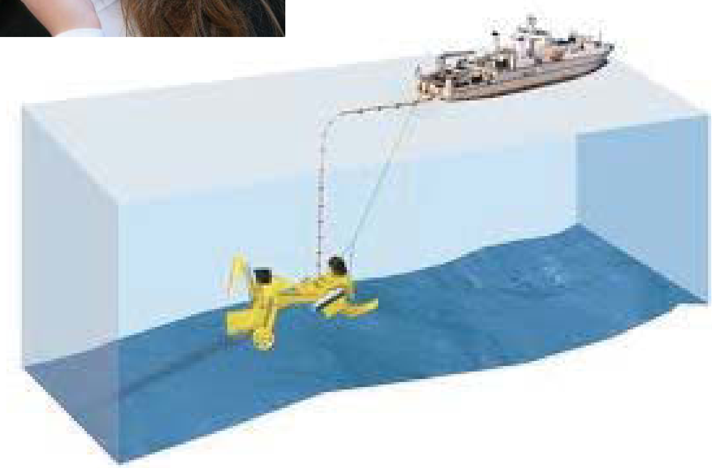
- The rest of 6.02 is about the colored oval
- Simplest network is a single physical communication link
- We'll start with that, then get to networks with many links

Physical Communication Links are Inherently *Analog*





Analog = continuous-valued, continuous-time



Voltage waveform on a cable
Light on a fiber, or in free space
Radio (EM) waves through the atmosphere
Acoustic waves in air or water
Indentations on vinyl or plastic
Magnetization of a disc or tape
…

or … Mud Pulse Telemetry, anyone?!

"This is the most common method of data transmission used by MWD (Measurement While Drilling) tools. Downhole a valve is operated to restrict the flow of the drilling mud (slurry) according to the digital information to be transmitted. This creates pressure fluctuations representing the information. The pressure fluctuations propagate within the drilling fluid towards the surface where they are received from pressure sensors. On the surface, the received pressure signals are processed by computers to reconstruct the information. The technology is available in three varieties - *positive* pulse, *negative* pulse, and *continuous wave*."

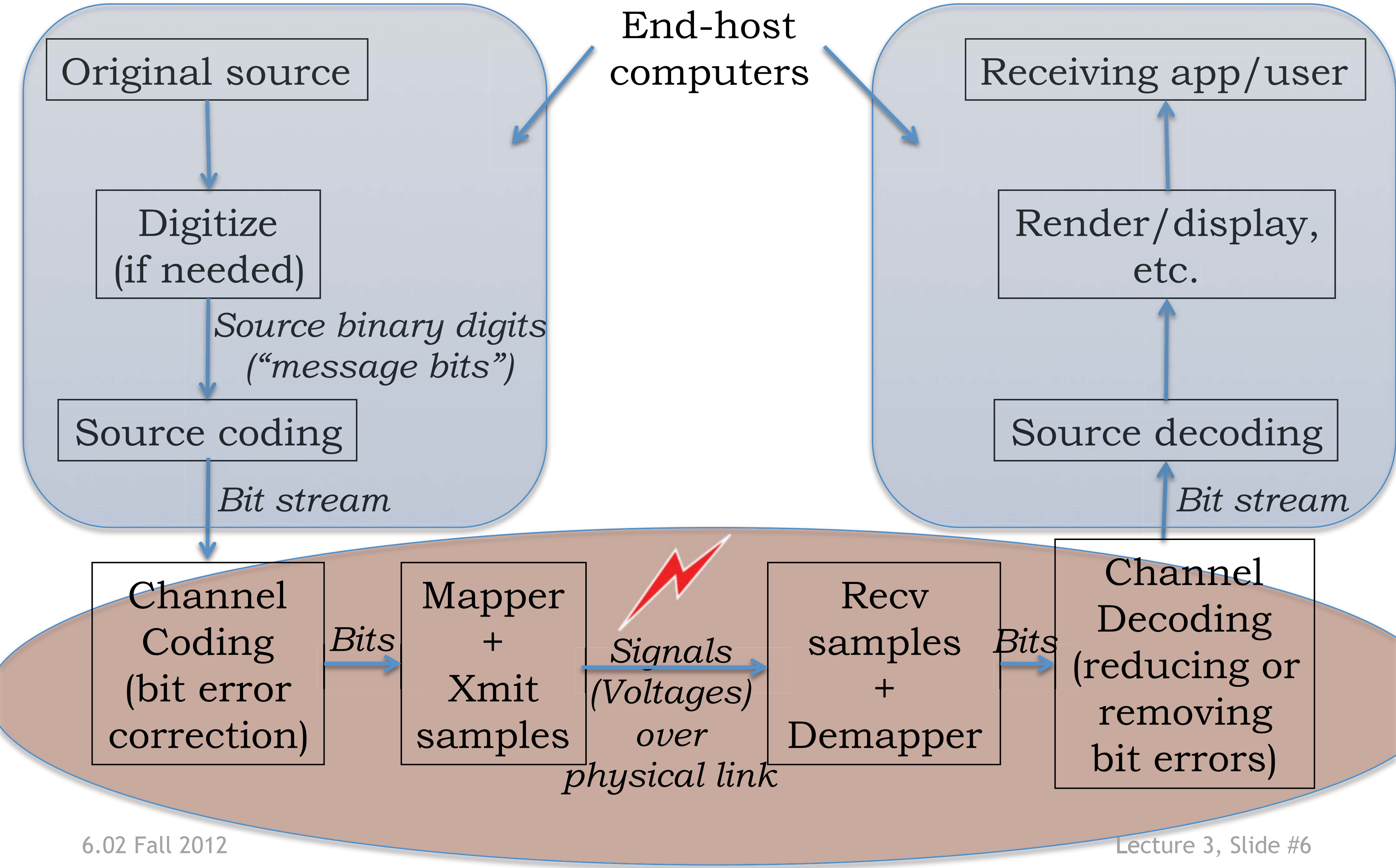
(from Wikipedia)

or … Mud Pulse Telemetry, anyone?!

"This is the most common method of data transmission used by MWD (Measurement While Drilling) tools. Downhole a valve is operated to restrict the flow of the drilling mud (slurry) according to the ***digital*** information to be transmitted. This creates pressure fluctuations representing the information. The pressure fluctuations propagate within the drilling fluid towards the surface where they are received from pressure sensors. On the surface, the received pressure signals are processed by computers to reconstruct the information. The technology is available in three varieties - *positive* pulse, *negative* pulse, and *continuous wave*."

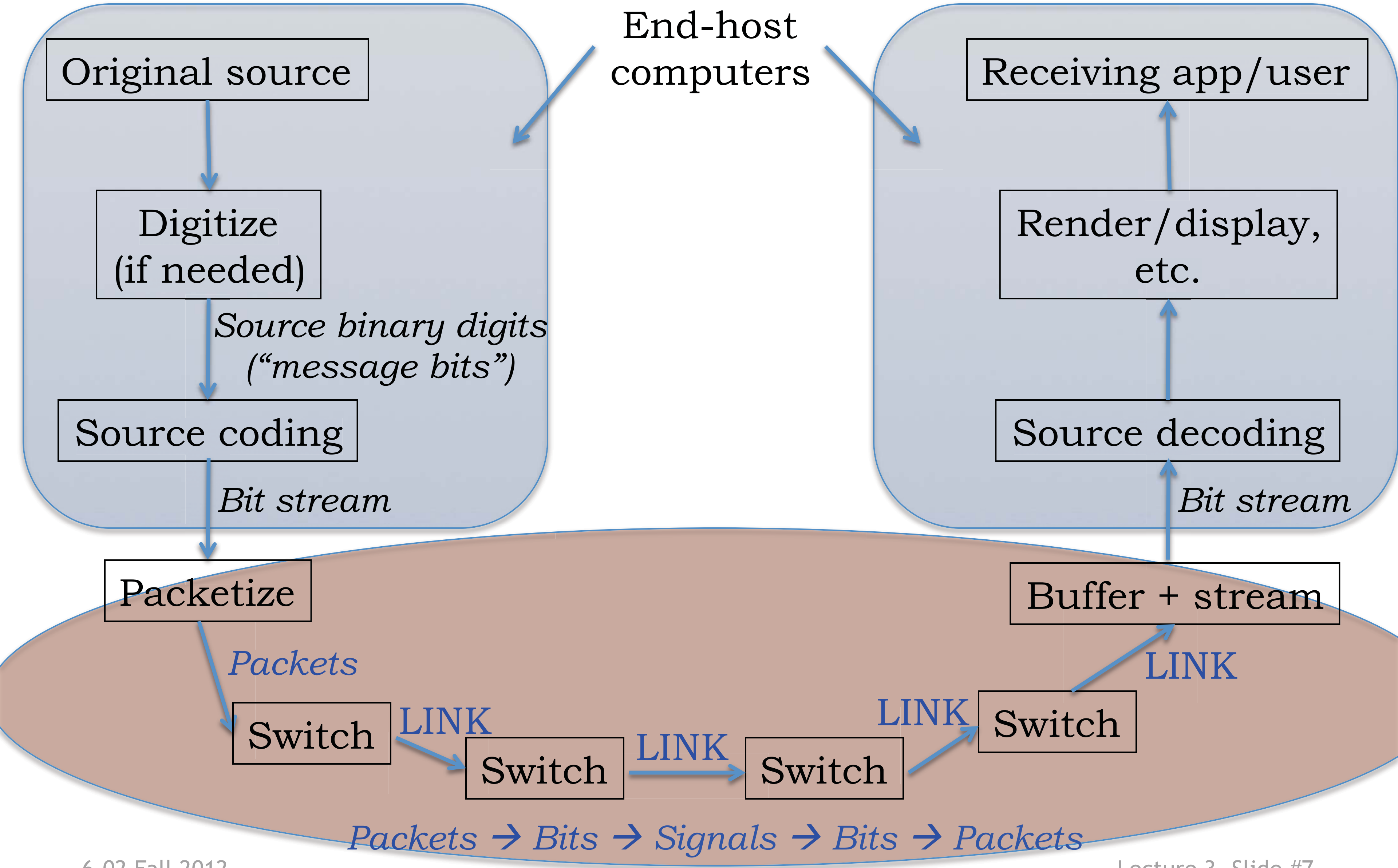
(from Wikipedia)

Single Link Communication Model



Network Communication Model

Three Abstraction Layers: Packets, Bits, Signals



Digital Signaling: Map Bits to Signals

Key Idea: "Code" or map or modulate the desired bit sequence onto a (continuous-time) analog signal, communicating at some bit rate (in bits/sec).

To help us extract the intended bit sequence from the noisy received signals, we'll map bits to signals using a fixed set of discrete values. For example, in a *bi-level signaling* (or *bi-level mapping*) scheme we use two "voltages":

V0 is the binary value "0"

V1 is the binary value "1"

If V0 = -V1 (and often even otherwise) we refer to this as ***bipolar*** signaling.

At the receiver, process and sample to get a "voltage"

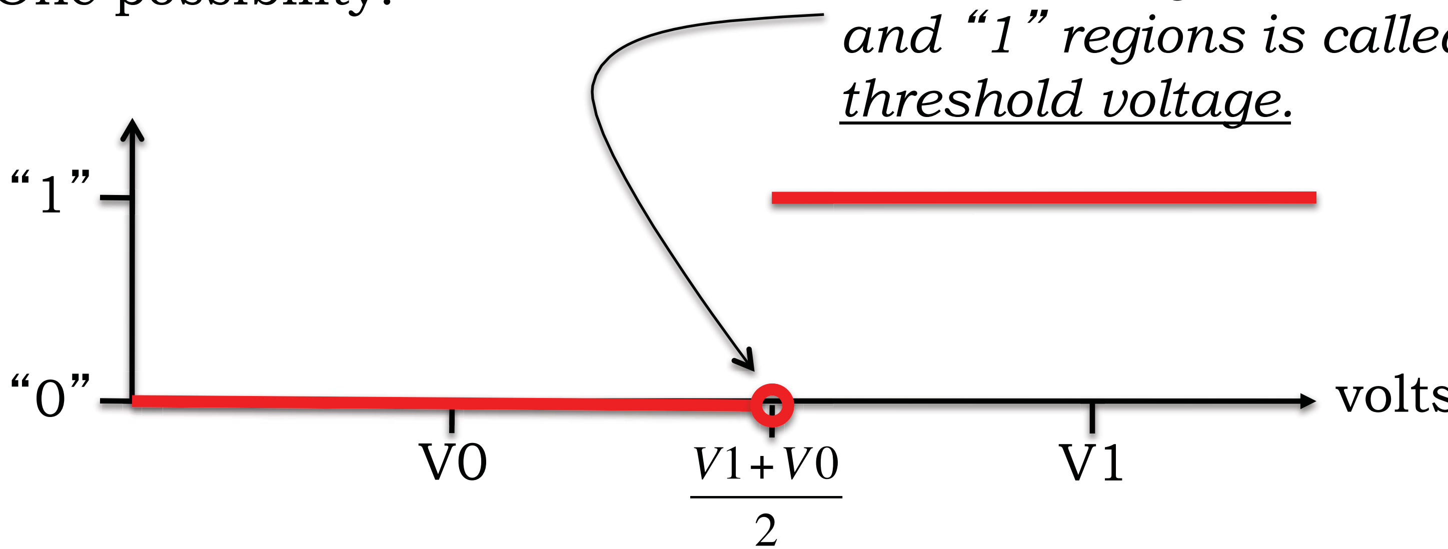
- Voltages near V0 would be interpreted as representing "0"
- Voltages near V1 would be interpreted as representing "1"
- If we space V0 and V1 far enough apart, we can tolerate some degree of noise --- but there will be occasional errors!

Digital Signaling: Receiving

We can specify the behavior of the receiver with a graph that shows how incoming voltages are mapped to "0" and "1".

One possibility:

The boundary between "0" and "1" regions is called the <u>threshold voltage</u>.

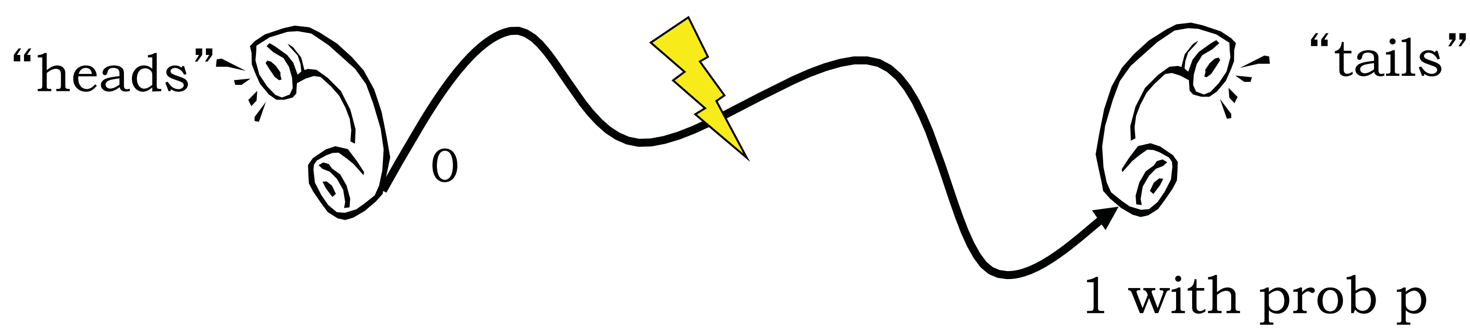


If received voltage between $V0$ & $\frac{V1+V0}{2}$ → "0", else "1"

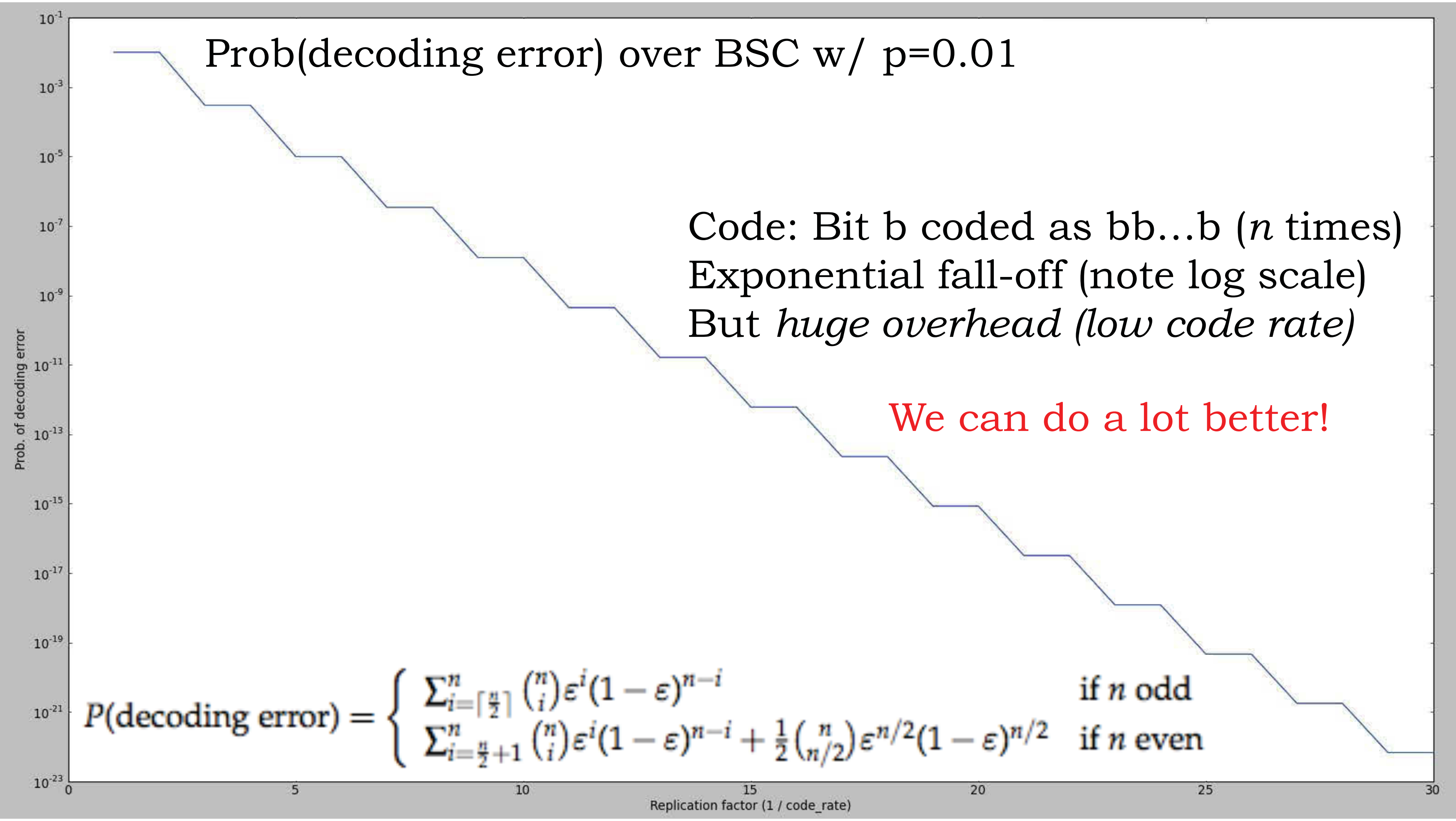
Bit-In, Bit-Out Model of Overall Path: Binary Symmetric Channel

Suppose that during transmission a "0" is turned into a "1" or a "1" is turned into a "0" with probability p, independently of transmissions at other times

This is a *binary symmetric channel* (BSC) --- a useful and widely used abstraction



Replication Code to reduce decoding error



$$P(\text{decoding error}) = \begin{cases} \sum_{i=\lceil \frac{n}{2} \rceil}^{n} \binom{n}{i} \varepsilon^i (1-\varepsilon)^{n-i} & \text{if } n \text{ odd} \\ \sum_{i=\frac{n}{2}+1}^{n} \binom{n}{i} \varepsilon^i (1-\varepsilon)^{n-i} + \frac{1}{2}\binom{n}{n/2} \varepsilon^{n/2}(1-\varepsilon)^{n/2} & \text{if } n \text{ even} \end{cases}$$

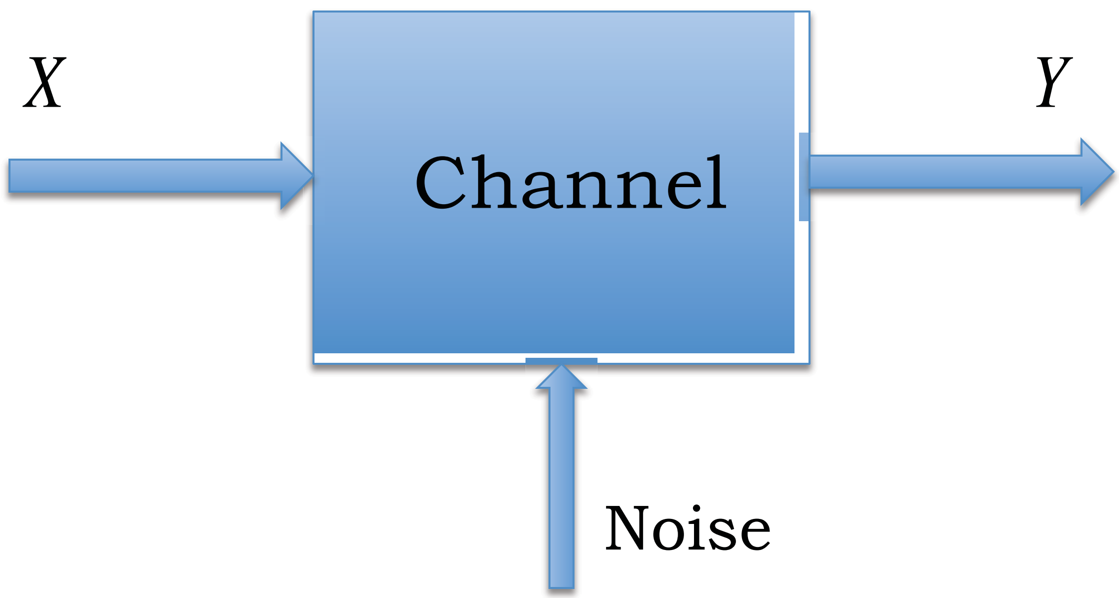
Replication factor, *n* (1/code_rate)

Mutual Information

$$I(X;Y) = H(X) - H(X \mid Y)$$

How much is our uncertainty about X reduced by knowing Y?

Evidently a central question in communication or, more generally, inference. Thank you, Shannon!



Evaluating conditional entropy and mutual information

To compute conditional entropy:

$$H(X \mid Y = y_j) = \sum_{i=1}^{m} p(x_i \mid y_j) \log_2 \left(\frac{1}{p(x_i \mid y_j)} \right)$$

$$H(X \mid Y) = \sum_{i=1}^{m} H(X \mid Y = y_j) p(y_j)$$

$$H(X,Y) = H(X) + H(Y \mid X)$$

$$= H(Y) + H(X \mid Y)$$

because

$$p(x_i, y_j) = p(x_i) p(y_j \mid x_i)$$

$$= p(y_j) p(x_i \mid y_j)$$

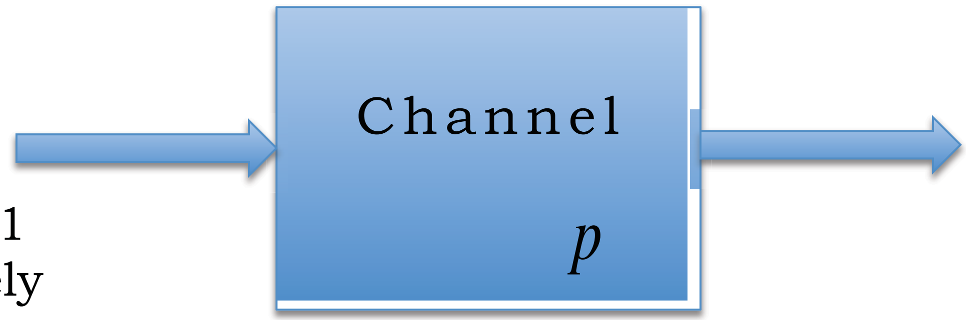
so

$$\boxed{I(X;Y) = I(Y;X)} \rightarrow \text{mutual information is symmetric}$$

e.g., Mutual information between input and output of binary symmetric channel (BSC)

$X \in \{0,1\}$

Assume 0 and 1 are equally likely



$Y \in \{0,1\}$

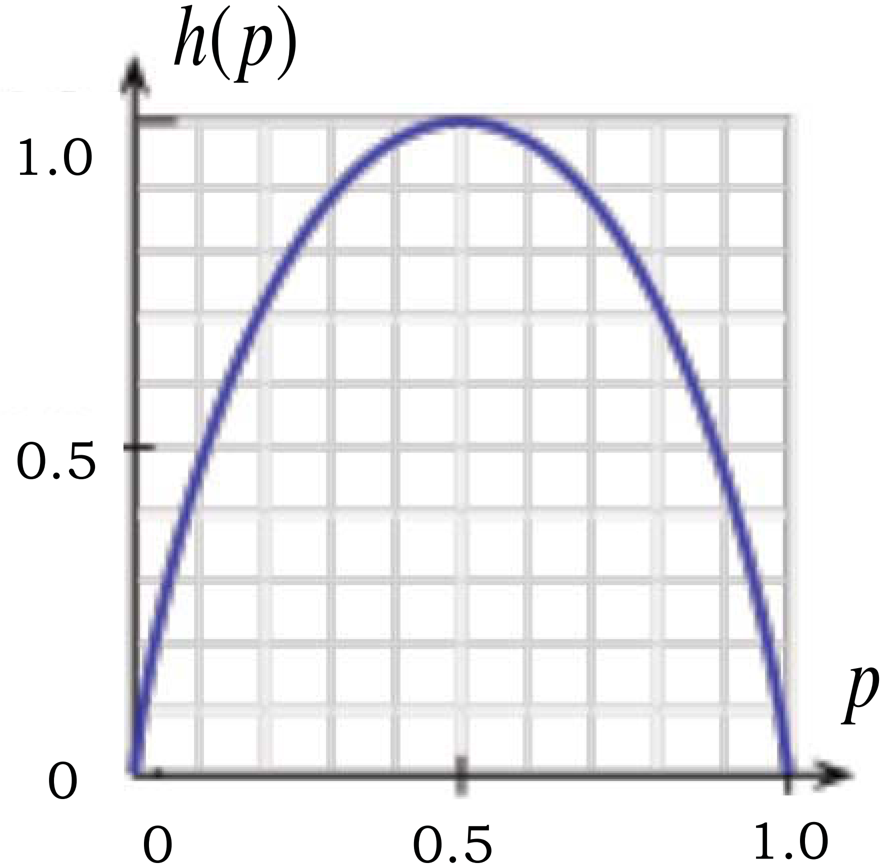
With probability p the input binary digit gets flipped before being presented at the output.

$$\boxed{I(X;Y)} = I(Y;X) = H(Y) - H(Y \mid X)$$
$$= 1 - H(Y \mid X=0)p_X(0) - H(Y \mid X=1)p_X(1)$$
$$= \boxed{1 - h(p)}$$

Binary entropy function $h(p)$

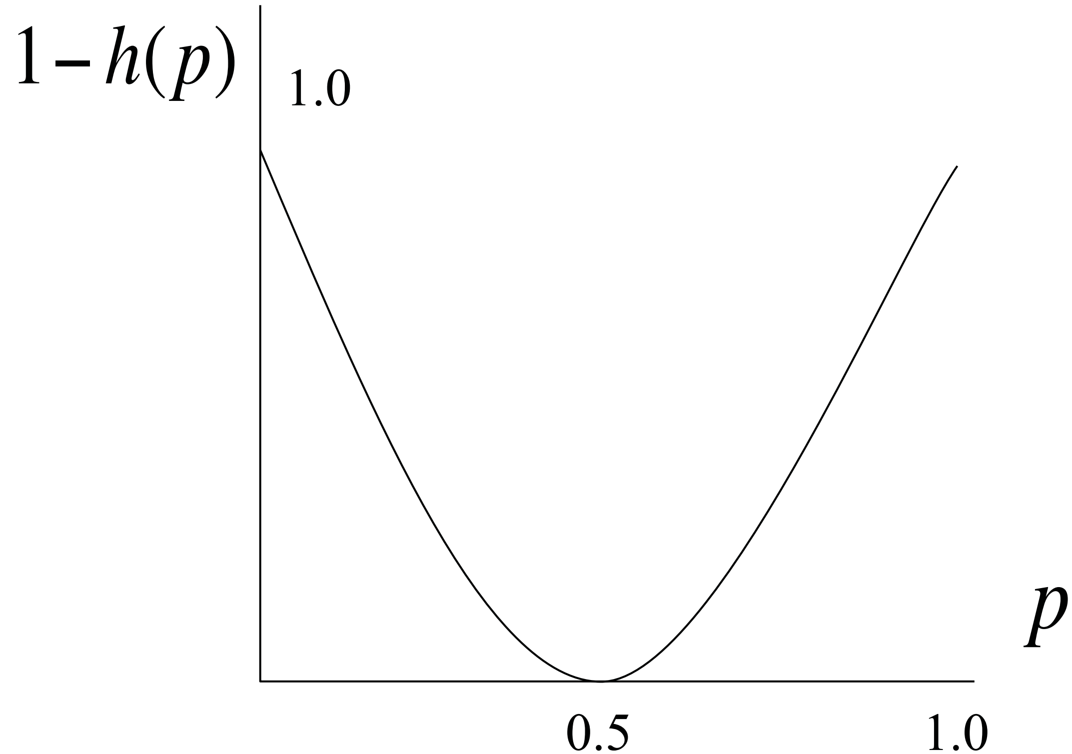
Heads (or C=1) with probability p

Tails (or C=0) with probability $1-p$



$$H(C) = -p\log_2 p - (1-p)\log_2(1-p) = h(p)$$

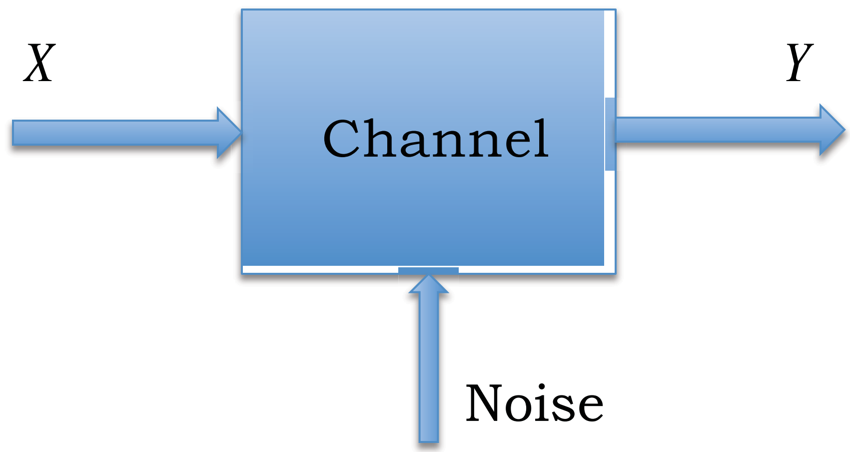
So mutual information between input and output of the BSC with equally likely inputs looks like this:



For low-noise channel, significant reduction in uncertainty about the input after observing the output.

For high-noise channel, little reduction.

Channel capacity



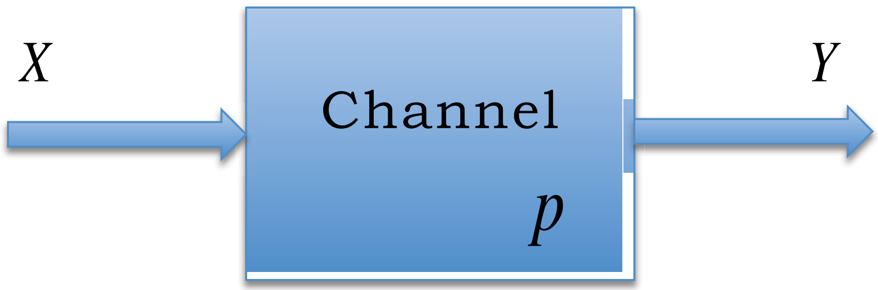
To characterize the *channel*, rather than the input and output, define

$$C = \max I(X;Y) = \max\{H(X) - H(X \mid Y)\}$$

where the maximization is over all possible distributions of X.

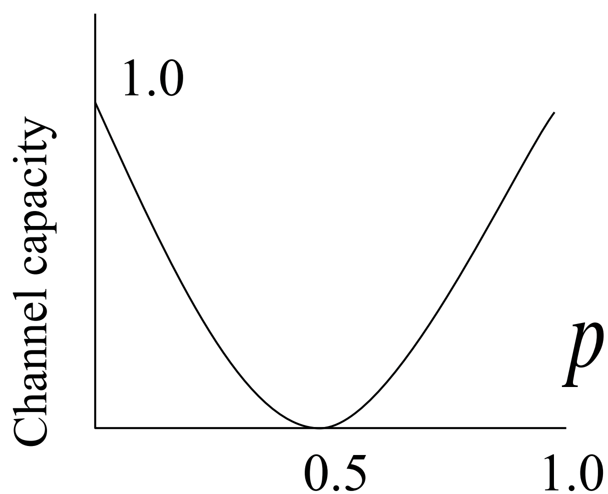
> This is the most we can expect to reduce our uncertainty about X through knowledge of Y, and so must be *the most information we can expect to send through the channel on average, per use of the channel*. Thank you, Shannon!

e.g., capacity of the binary symmetric channel



Easiest to compute as $C = \max\{H(Y) - H(Y \mid X)\}$, still over all possible probability distributions for X. The second term doesn't depend on this distribution, and the first term is maximized when 0 and 1 are equally likely at the input. So invoking our mutual information example earlier:

➔ $\boxed{C = 1 - h(p)}$



What channel capacity tells us about how fast and how accurately we can communicate …

The magic of asymptotically error-free transmission at any rate $R < C$

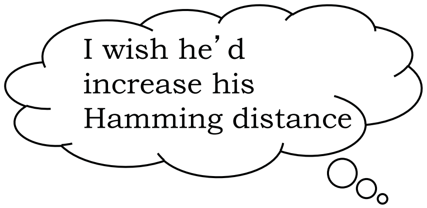
Shannon showed that one can theoretically transmit information (i.e., message bits) at an average rate $R < C$ per use of the channel, with arbitrarily low error.

(He also showed the converse, that transmission at an average rate $R \geq C$ incurs an error probability that is lower-bounded by some positive number.)

The secret: Encode blocks of k message bits into n-bit codewords, so $R = k / n$, with k and n very large.

Encoding blocks of k message bits into n-bit codewords to protect against channel errors is an example of

channel coding



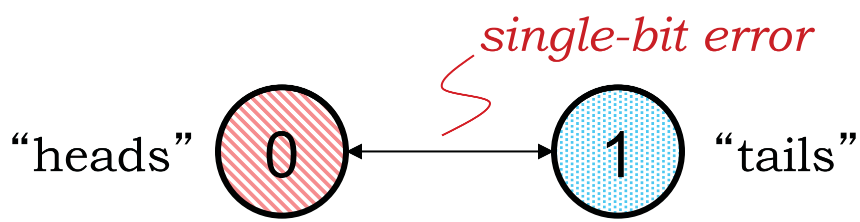
Hamming Distance

Image from *Green Eggs and Ham*, by Dr. Seuss, removed due to copyright restrictions.

The number of bit positions in which the corresponding bits of two encodings of the same length are different

The Hamming Distance (HD) between a valid binary codeword and the same codeword with *e* errors is *e*.

The problem with no coding is that the two valid codewords ("0" and "1") also have a Hamming distance of 1. So a single-bit error changes a valid codeword into another valid codeword...

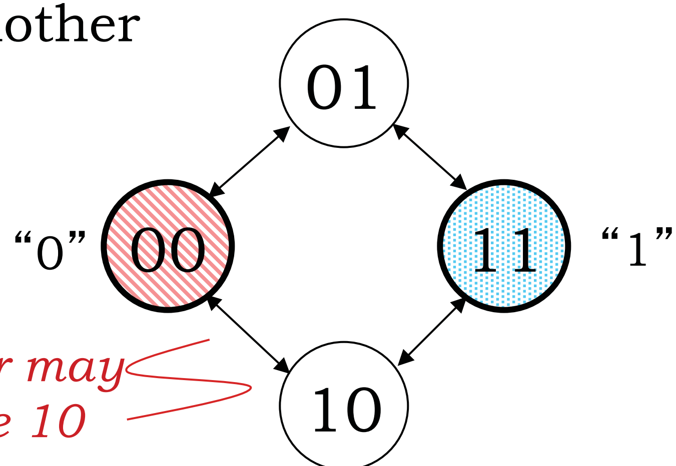


What is the Hamming Distance of the replication code?

Idea: Embedding for Structural Separation

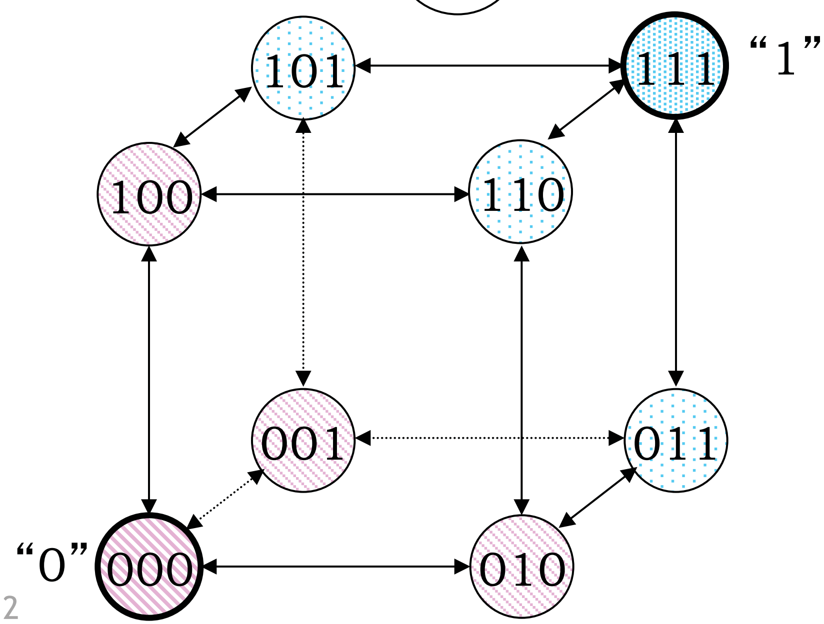
Encode so that the codewords are "far enough" from each other

Likely error patterns shouldn't transform one codeword to another



single-bit error may cause 00 to be 10 (or 01)

Code: nodes chosen in hypercube + mapping of message bits to nodes



If we choose 2^k out of 2^n nodes, it means we can map all k-bit message strings in a space of n-bit *codewords*. The *code rate* is k/n.

Minimum Hamming Distance of Code vs. Detection & Correction Capabilities

If d is the minimum Hamming distance between codewords, we can detect **all** patterns of <= (d-1) bit errors

If d is the minimum Hamming distance between codewords, we can correct **all** patterns of $\left\lfloor \frac{d-1}{2} \right\rfloor$ or fewer bit errors

How to Construct Codes?

0000000	1100001	1100110	0000111
0101010	1001011	1001100	0101101
1010010	0110011	0110100	1010101
1111000	0011001	0011110	1111111

Want: 4-bit messages with single-error correction (min HD=3)

How to produce a code, i.e., a set of codewords, with this property?

A Simple Code: Parity Check

- Add a parity bit to message of length k to make the total number of "1" bits even (aka "even parity").
- If the number of "1"s in the received word is *odd*, there there has been an error.

0 1 1 0 0 1 0 1 0 0 1 1 → original word with parity bit
0 1 1 0 0 0 0 1 0 0 1 1 → single-bit error (detected)
0 1 1 0 0 0 1 1 0 0 1 1 → 2-bit error (not detected)

- Minimum Hamming distance of parity check code is 2
  - Can detect all single-bit errors
  - In fact, can detect all odd number of errors
  - But cannot detect even number of errors
  - And cannot correct any errors

Binary Arithmetic

- Computations with binary numbers in code construction will involve Boolean algebra, or algebra in “GF(2)” (Galois field of order 2), or modulo-2 algebra:

$$0+0=0, \qquad 1+0=0+1=1, \qquad 1+1=0$$

$$0*0=0*1=1*0=0, \qquad 1*1=1$$

Linear Block Codes

Block code: k message bits encoded to n code bits, i.e., each of 2^k messages encoded into a unique n-bit combination via a *linear transformation*, using GF(2) operations:

$$C=D.G$$

C is an n-element row vector containing the codeword
D is a k-element row vector containing the message
G is the kxn *generator matrix*
Each codeword bit is a specified linear combination of message bits.

Key property: Sum of any two codewords is *also* a codeword → necessary and sufficient for code to be linear. (So the all-0 codeword has to be in any linear code --- why?)

More on linear block codes in recitation & next lecture!!

Minimum HD of Linear Code

- (n,k) code has rate k/n
- Sometimes written as (n,k,d), where d is the minimum HD of the code.
- The "weight" of a code word is the number of 1's in it.
- The minimum HD of a linear code is the minimum weight found in its nonzero codewords

Examples: What are n, k, d here?

{000, 111} (3,1,3). Rate= 1/3.

{0000, 1100, 0011, 1111} (4,2,2). Rate = ½.

{1111, 0000, 0001}

{1111, 0000, 0010, 1100}

Not linear codes!

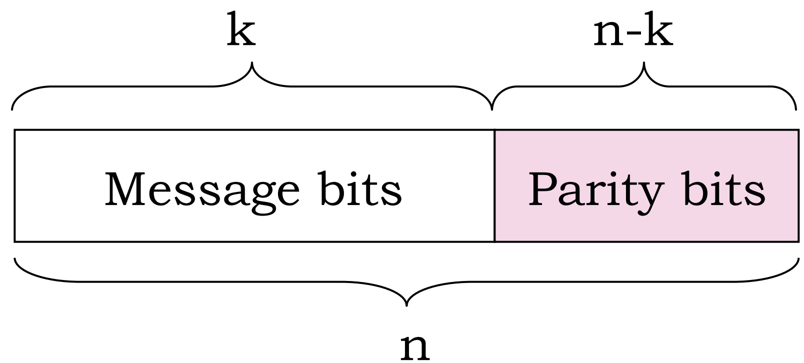
0000000	1100001	1100110	0000111
0101010	1001011	1001100	0101101
1010010	0110011	0110100	1010101
1111000	0011001	0011110	1111111

(7,4,3) code. Rate = 4/7.

The HD of a linear code is the number of "1"s in the non-zero codeword with the smallest # of "1"s

(n,k) Systematic Linear Block Codes

- Split data into *k*-bit blocks
- Add (*n*-*k*) parity bits to each block using (*n*-*k*) linear equations, making each block *n* bits long



The entire block is the called the "code word in systematic form"

- Every linear code can be represented by an equivalent systematic form
- Corresponds to choosing G = [I | A], i.e., the identity matrix in the first k columns