6.450: Principles of Digital Communication 1

Digital Communication: Enormous and normally rapidly growing industry, roughly comparable in size to the computer industry.

Objective: Study those aspects of communication systems unique to those systems. Little focus on hardware or software (similar to other systems).

6.450 is part of 2 term sequence with 6.451 (Principles of Digital Communication 2) but it also stands alone.

Theory has had an unusually powerful impact on system design in digital communication.

Its basis, information theory, was developed in 1948 by Claude Shannon.

For 25 years it was an elegant mathematical theory and source of PhD problems. Now it is mainstream engineering and guides system development.

Still based on abstract ideas. The tools are simple but require deep understanding.

Complex relationship between modeling, theory, exercises, and engineering/design.

Use very simple models to understand ideas. This generates powerful general theorems plus insights into more complex models and thus reality.

Exercises aimed at understanding the principles - getting the right answer is not the point since the model is oversimplified.

Engineering deals with approximations and judgment calls based on multiple simple models (insights).

Since the theorems apply only to simple models, they don't apply directly to real systems.

Often the proof of the theorem is more important in understanding applications than the theorem.

The proof lets you know why the result works, and thus whether it will approximately work under more complex conditions.

That is, bottom line-itus does not work well in this subject.

Since the exercises apply only to simple models, they don't apply directly to real systems.

You have to understand the exercise at a gut level to see how to use the idea.

This is why you should discuss the exercises with other students - grinding out the answer by pattern matching and manipulation is not the point.

Everyday communication systems (the telephone system, the Internet) have incredible complexity.

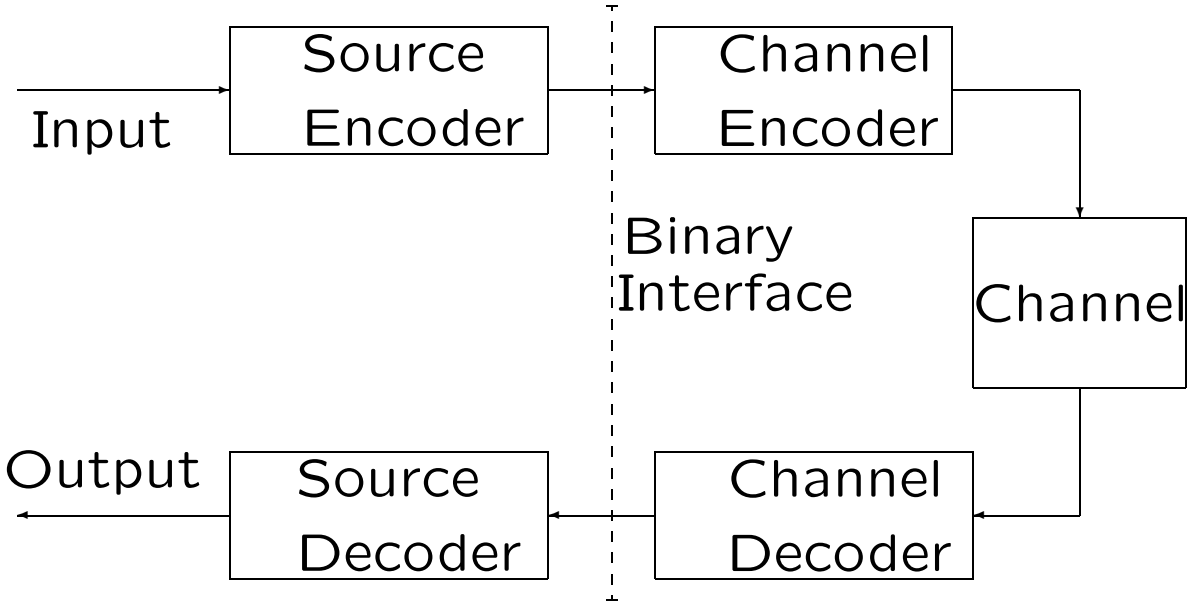
Must be designed and understood based on simple architectural principles.

Standardized interfaces and layering are key.

Most important interface here is between sources and channels.

Standard interface is binary data stream.

Channel input is a binary stream without meaning (from channel viewpoint).



REASONS FOR BINARY INTERFACE

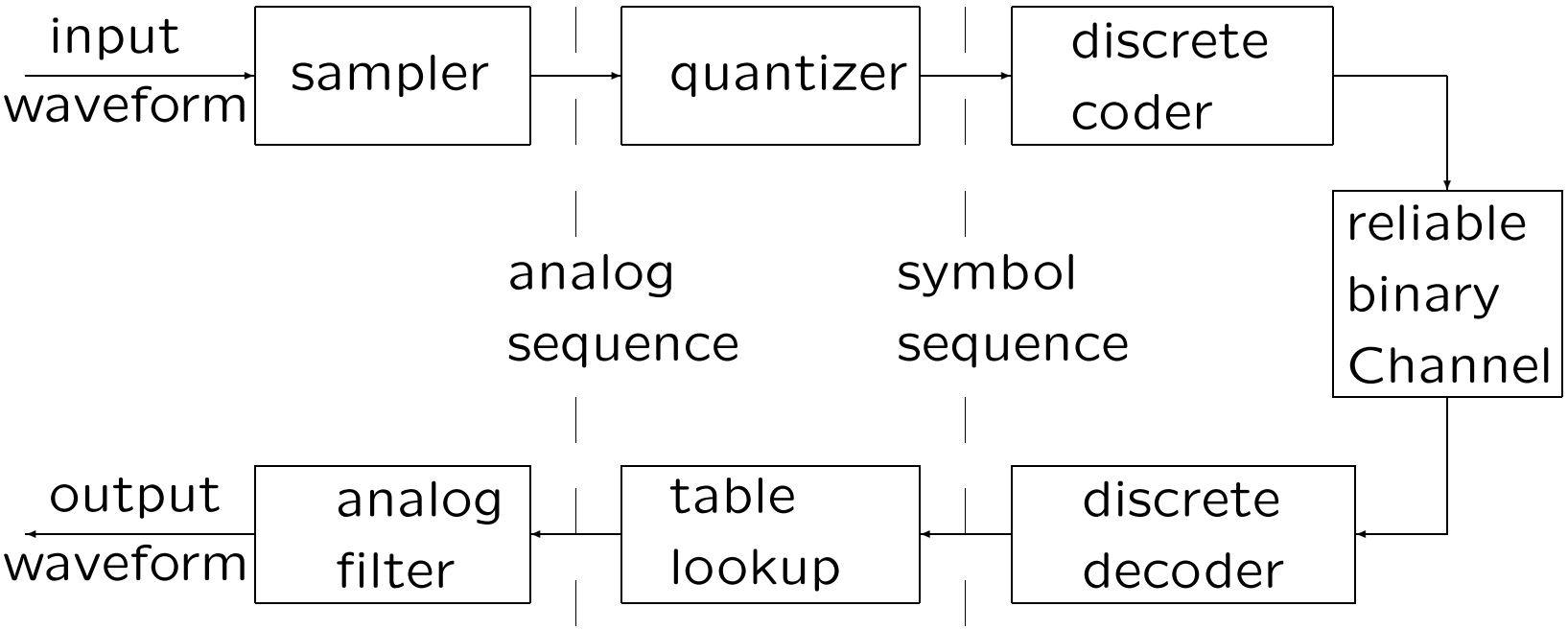
- Standardization (Simplifies implementation)
- Layering (Simplifies conceptualization)
- Loses nothing in performance (Shannon says)

Source Coding has 3 parts

- Analog waveform to analog sequence
- Quantizer (sequence to symbols)
- Symbols to bits

Decoding goes in opposite order

- bits to symbols
- symbols to sequence of numbers
- sequence to waveform



Layering of source coding

Binary interface: Source might produce irregularly timed packets (data) or bit stream (voice, video, etc.)

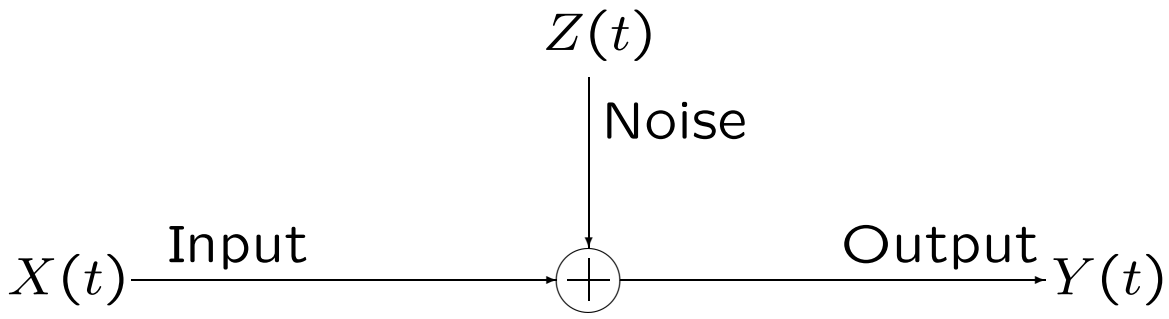
Channel accepts binary stream or packets, but queueing usually exists.

Queueing problems are separable and not covered here (see 6.263).

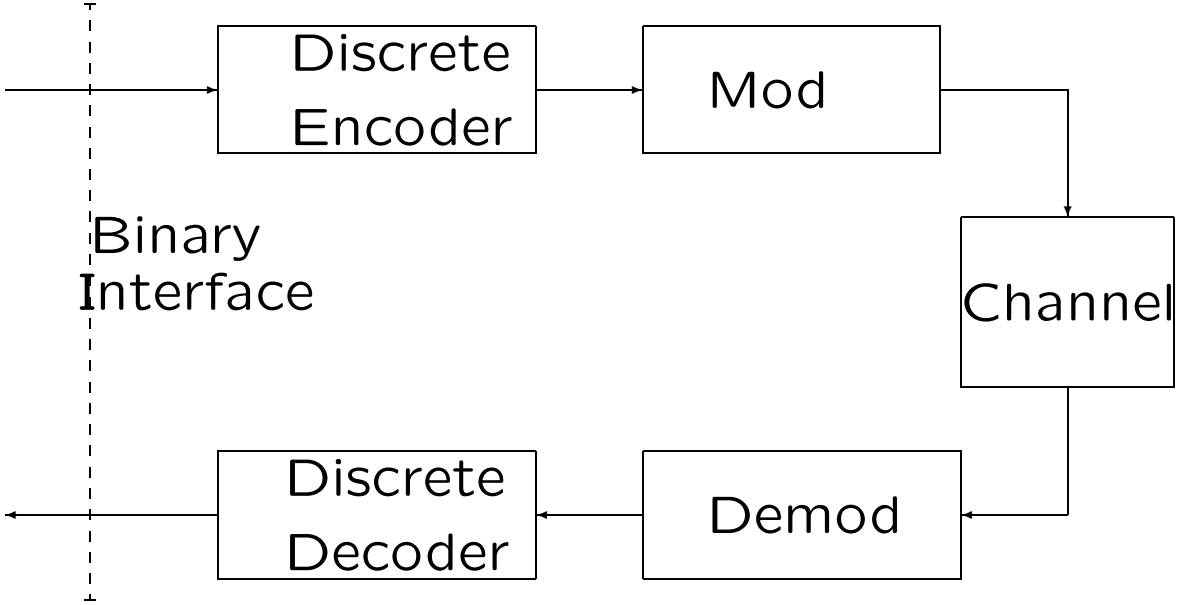
We study how to reduce the bit rate from sources and increase the maximum bit rate for channels.

Channel is given (not under control of designer).

Transmitter output is often a waveform. Input bits are converted to waveforms. At receiver, waveforms are converted to a bit stream.



Additive white Gaussian noise (AWGN) channel.



Separation of encoding into discrete coding and modulation.

Discrete encoder maps bits at one rate into bits at a higher rate. This makes it possible to correct channel errors

Example: $0 \rightarrow 000$; $1 \rightarrow 111$ corrects single errors.

Modulation maps bit sequences to waveforms.

Modulation usually separated into mapping to base-band waveforms, then modulation to passband.

Modern practice often combines all these layers into coded modulatiion.

For the additive WGN channel, with a bandwidth constraint, the capacity (in bits per second) is

$$C = W \log_2 \left(1 + \frac{P}{N_0 W}\right)$$

This is the ultimate, but it is essentially achievable in practice.

Wireless channels have added complications:

- Multiple physical paths.
- Relative path lengths change in time, causing multipath fading.
- Multiple users interfere with each other.

This subject stresses theory and mathematics more than most subjects on digital communication. Why?

There are too many communication systems to study them all, so principles are necessary.

Systems are becoming more and more complex. Simple models are the only hope for understanding.

The complexity of a communication system is of at least two types:

The computational complexity is the required number of elementary computational operations (per unit time).

The conceptual complexity concerns the difficulty of structuring or layering the system into simple components.

Example: A large telephone network has huge computational complexity but low conceptual complexity.

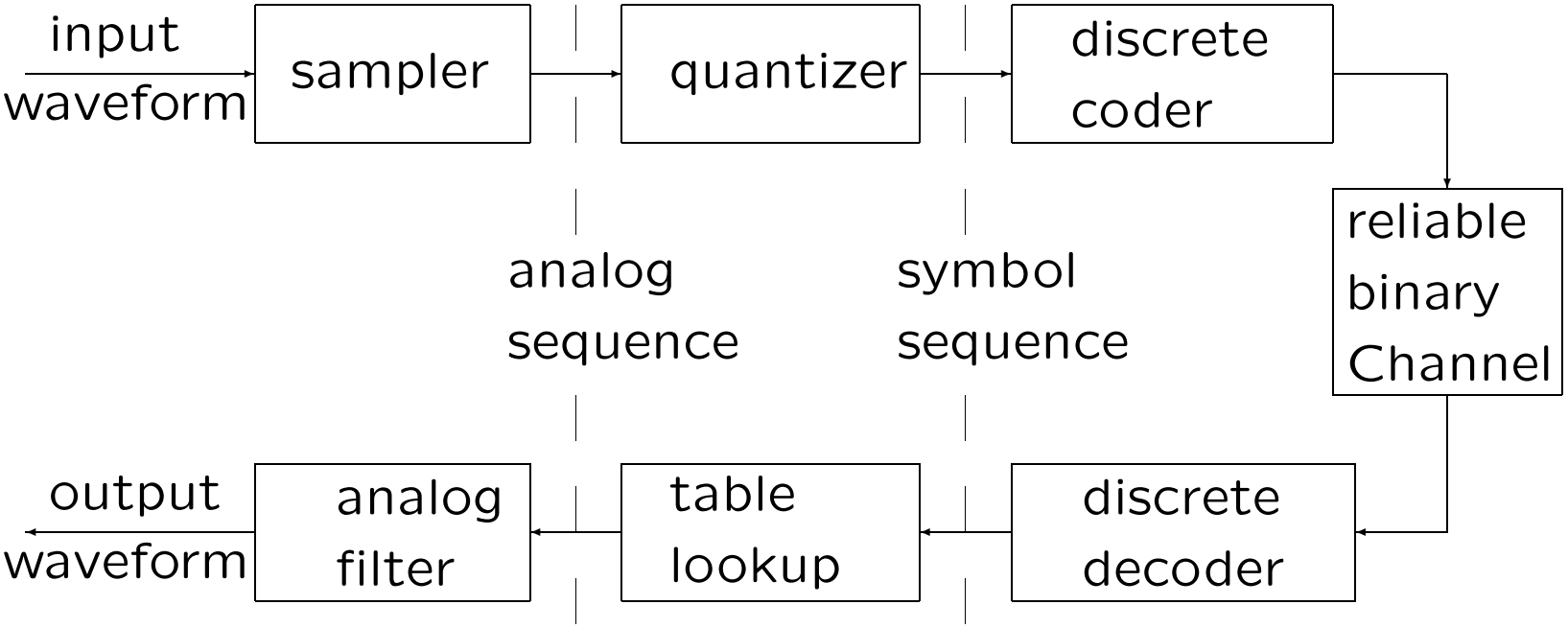
Microsoft word has low computational complexity but huge conceptual complexity.

Increasingly, computational complexity has little impact on cost because of chip technology. If you can understand it, you can build it.

Caveat 1: Low cost with high complexity requires large volume, long development time, and careful design.

Caveat 2: Complex systems are often not well thought through. They often don't work or are not robust.

Caveat 3: Special applications, since they involve small numbers, must extend the chips used in more general systems with relatively simple extensions.



Layering of source coding

Examples of analog sources are voice, music, video, images.

Restrict to waveform sources (voice, music)

Usually sampled or expanded into series, producing a sequence of real or complex numbers.

Sequence of numbers is encoded by quantization into sequence of symbols.

Both analog and discrete sources then require encoding of sequence of symbols.

DISCRETE SOURCE CODING

OBJECTIVE: Map sequence of symbols into binary sequence with unique decodability.

SIMPLEST APPROACH: Map each source symbol into an L-tuple of binary digits.

For an alphabet size of M, require $2^L \geq M$.

To avoid wasting bits, choose L as smallest integer satisfying $2^L \geq M$, i.e.,,

$$\log_2 M \leq L < \log_2 M + 1; \quad L = \lceil \log_2 M \rceil$$

Example (for alphabet $\{\alpha, a, A, \mathcal{A}, \overline{A}\}$):

$$\begin{aligned} \alpha &\rightarrow \ 000 \\ a &\rightarrow \ 001 \\ A &\rightarrow \ 010 \\ \mathcal{A} &\rightarrow \ 011 \\ \overline{A} &\rightarrow \ 100 \end{aligned}$$

This can be easily decoded.

As a practical example, the ASCII code maps letters, numbers, etc. into binary 8-tuples (bytes).

These are called *fixed length* codes.

MORE GENERAL FIXED LENGTH CODE:

Segment source sequence into blocks of n; encode n symbols as a unit.

There are M^n n-tuples of source letters.

Fixed-length source coding on n-tuples requires

$$L = \lceil \log_2 M^n \rceil$$

Rate $\overline{L} = L/n$ bits per source symbol (bpss)

$$\log_2 M \leq \overline{L} < \log_2 M + \frac{1}{n}$$

For large n, $\overline{L}$ approaches $\log_2 M$ from above.

Fixed-length coding requires $\log_2 M$ bpss.

VARIABLE LENGTH SOURCE CODES

Motivation: Probable symbols should have shorter codewords than improbable to reduce bpss.

A *variable-length source code* $\mathcal{C}$ encodes each symbol x in source alphabet $\mathcal{X}$ to a binary codeword $\mathcal{C}(x)$ of length $l(x)$.

For example, for $\mathcal{X} = \{a, b, c\}$

$$\begin{aligned} C(a) &= 0 \\ C(b) &= 10 \\ C(c) &= 11 \end{aligned}$$

Successive codewords of a variable-length code are transmitted as a continuing sequence of bits.

There are no commas; decoder must parse the received sequence.

Buffering might be a problem here.

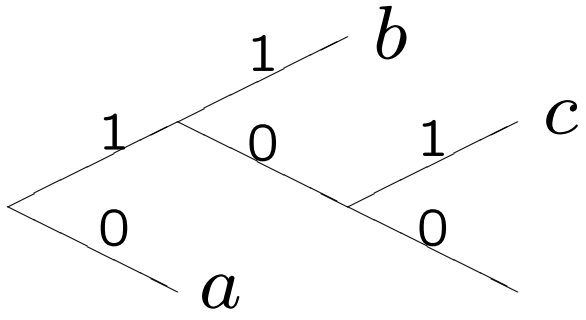
Requires *unique decodability* i.e., encoded bit stream must be uniquely parsed and source sequence recovered.

Assume initial synchronization.

PREFIX-FREE CODES

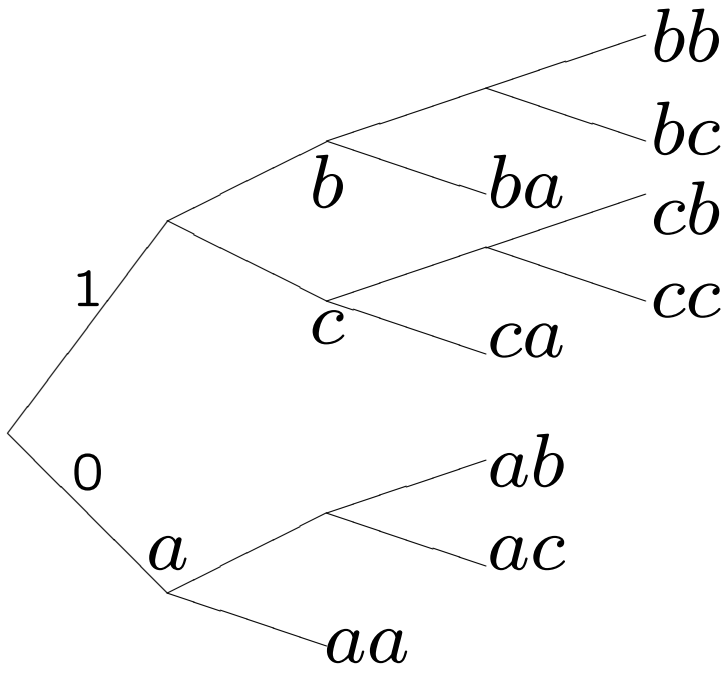
A code is prefix-free if no codeword is a prefix of any other codeword

A prefix-free code $\mathcal{C}$ can be represented by a binary code tree which grows from a root on the left to leaves on the right representing codewords.



$a \rightarrow 0$
$b \rightarrow 11$
$c \rightarrow 101$

Every codeword is at a leaf, but not all leaves are codewords. Empty leaves can be shortened. A full tree has no empty leaves.



Prefix-free codes are uniquely decodable.

Construct a tree for a concatenation of codewords.

To decode, start at the left, and parse whenever a leaf in the tree is reached.

THE KRAFT INEQUALITY

The Kraft inequality is a test on the existence of prefix-free codes with a given set of codeword lengths $\{l(x), x \in \mathcal{X}\}$.

Theorem (Kraft): Every prefix-free code for an alphabet $\mathcal{X}$ with codeword lengths $\{l(x), x \in \mathcal{X}\}$ satisfies

$$\sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1 \qquad (1)$$

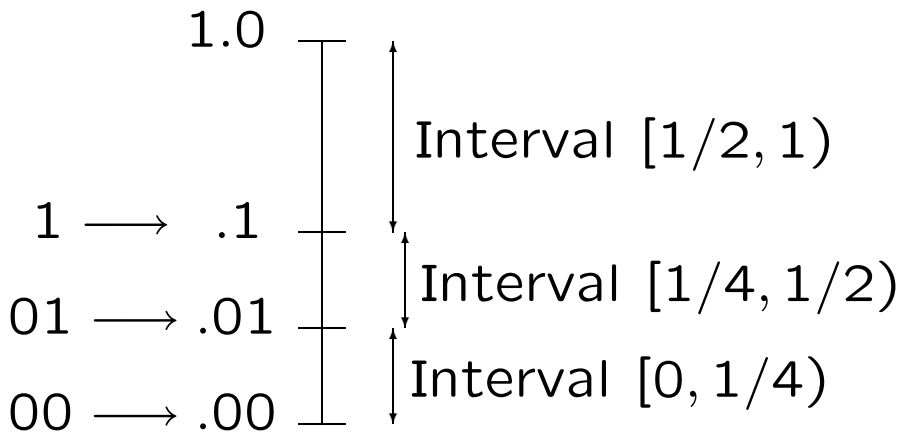
Conversely, if (1), then a prefix-free code with lengths $\{l(x)\}$ exists.

Moreover, a prefix-free code is full iff (1) is satisfied with equality.

We prove this by associating codewords with base 2 expansions i.e., 'decimals' in base 2.

Represent binary codeword $y_1, y_2, \ldots, y_m$ as

$$.y_1 y_2 \cdots y_m = y_1/2 + y_2/4 + \cdots + y_m 2^{-m}$$



If codeword x is a prefix of codeword y, then expansion x contains expansion y in its "approximation interval."

A code is a prefix code iff the base 2 expansion intervals are disjoint.

But the sum of disjoint expansion intervals is at most 1.

Code is full iff expansion intervals fill up [0, 1)

6.450 Principles of Digital Communication I
Fall 2009