

# Introduction

Purpose

by Erol Seke

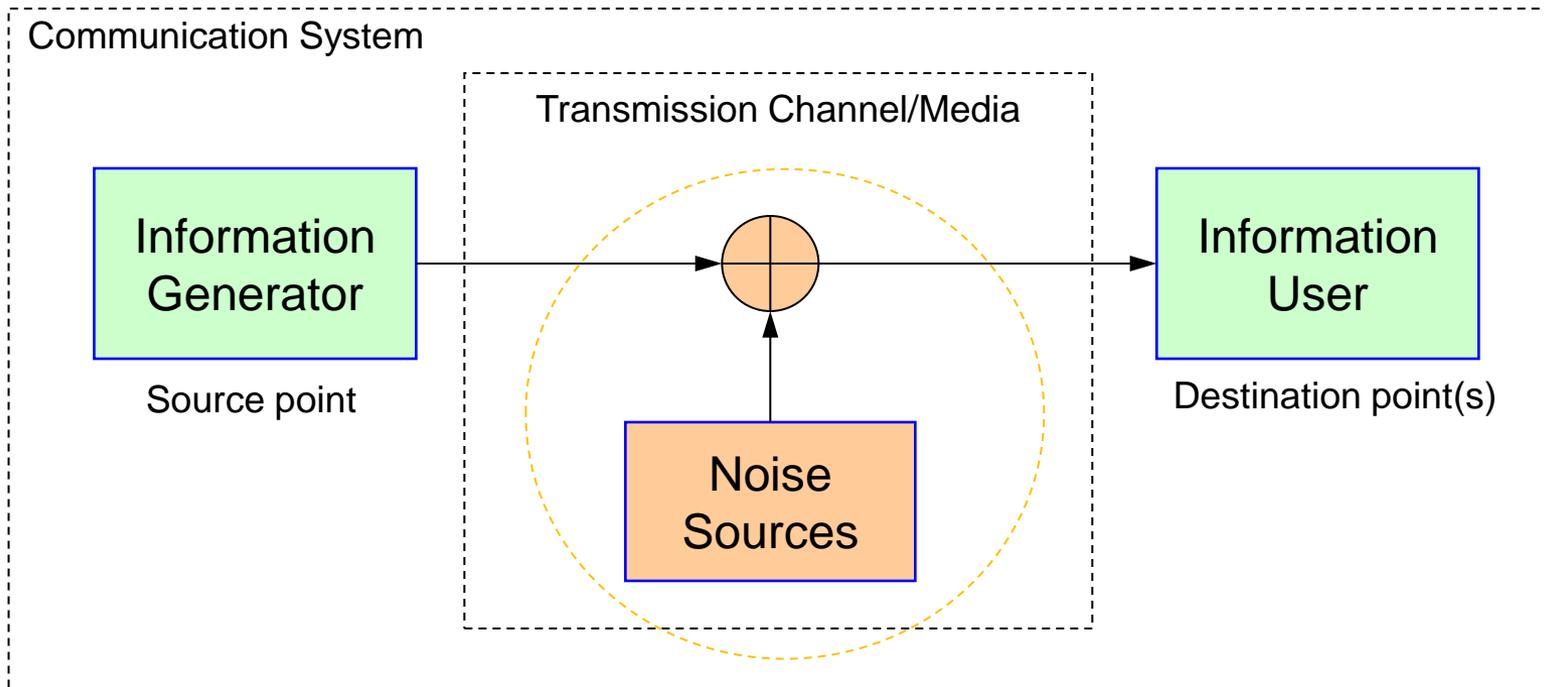
For the course “**Communications**”



**ESKİŞEHİR OSMANGAZI UNIVERSITY**

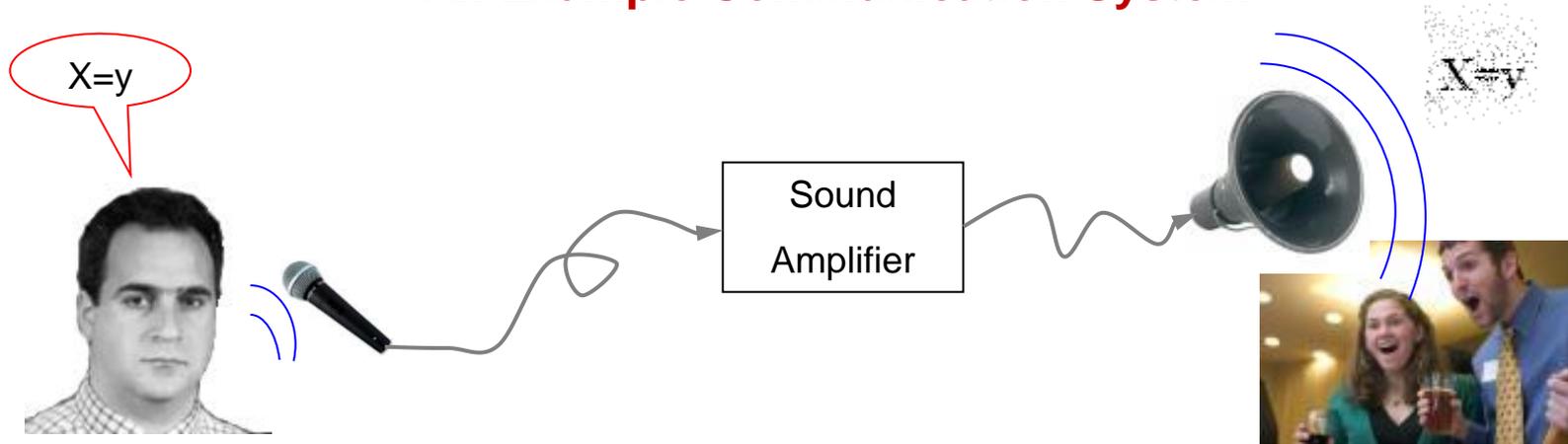
# The Goal

Transfer information from source point to one or more destinations correctly  
(using least amount of **resources**, in most cases)



**Resources** = material, voltage, spectrum, health, circuit, ... = time, money

## An Example Communication System



### Some other examples of (electronic) source, channel and destination

Microphone – Twisted pair of wire - Amplifier

Modem - Twisted pair of telephone line - Modem

Fax scanner – Telephone system – Fax printer

Computer – Ethernet cable - Computer

Computer's data storage medium – Fiber optic network – Another computer's storage

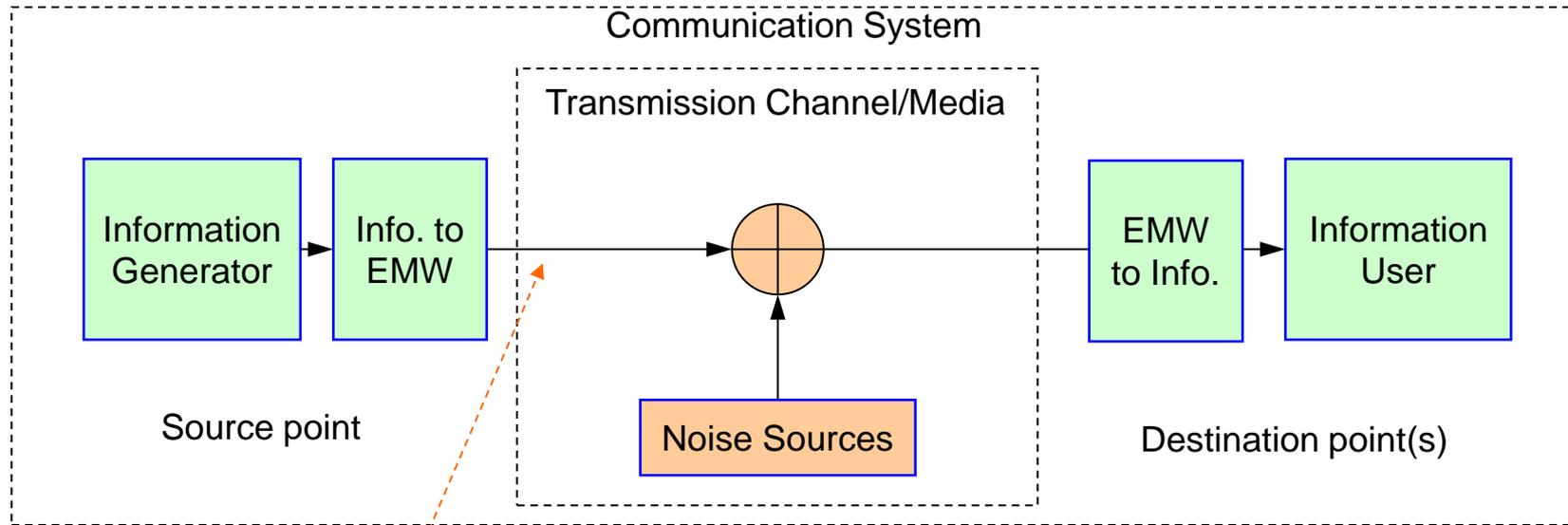
Digital data generator – Magnetic disk – Digital data user

Radio transmitter – Air/Space – Radio receiver

Digital TV data from satellite – Atmosphere – Digital TV receiver

TV Remote controller – Air – IR sensor/receiver on TV

## Another Block Diagram



$$\psi(r, t) = A \cos(kr - \omega t + \varphi) \quad (\text{basic travelling wave})$$

$t$  : time

$r$  : distance from source

$A$  : amplitude

$k$  : spatial frequency

$\omega$  : temporal frequency

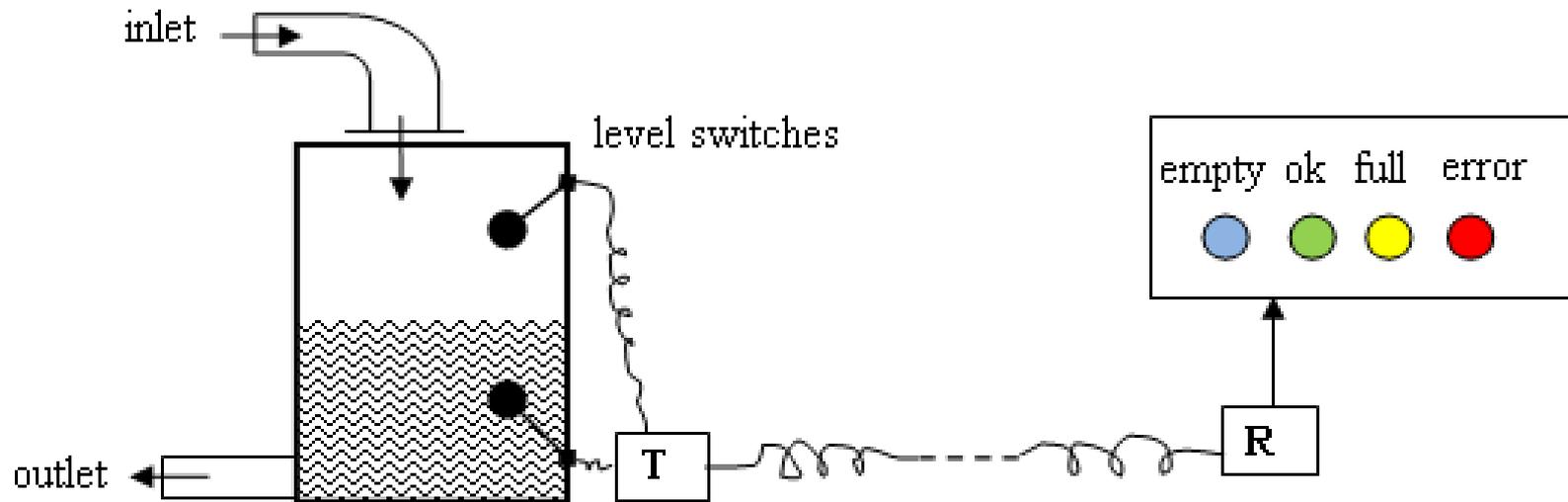
Magnetic wave is always perpendicular to electric field and travel direction

**It turns out that we need to control value(s) of one or more of these in order to carry information**

We can control some properties of electric field, magnetic field occurs automatically

We can also control polarity of the wave using superposition of multiple waves

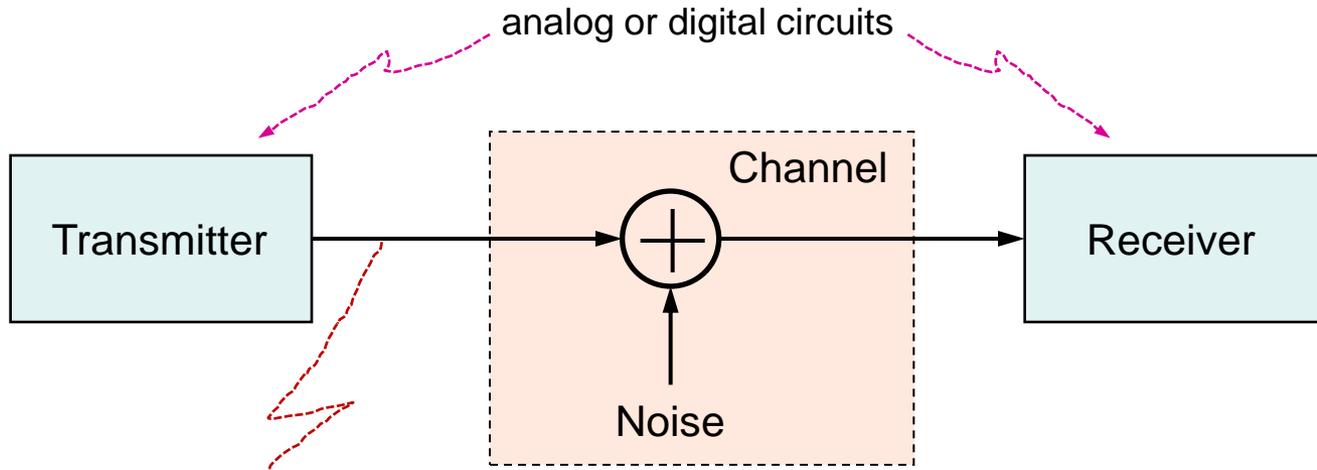
## An Example Digital Communication System



There are only 4 messages/values to sent

**Consider:** An analog system doing a similar job

# Analog / Digital Electronic Communication



Transmitted Signal

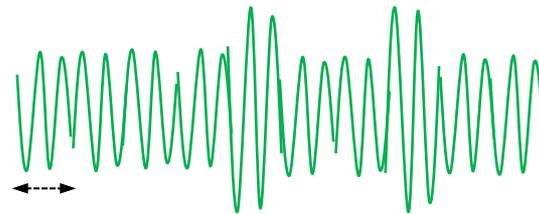
Analog Communication

Digital Communication

(infinite number of possible values)



Signal's all values are important at every point and cannot be completely repaired when damaged



$T_s$

Finite number of symbols represented by finite number of waveforms within each  $T_s$  ( $T_s$  can be variable but finite)

"digital is any signal that is discrete in time and discrete in magnitude"  
Oppenheim & Shaeffer (do you agree?)

# Digital Communication

## Advantages :

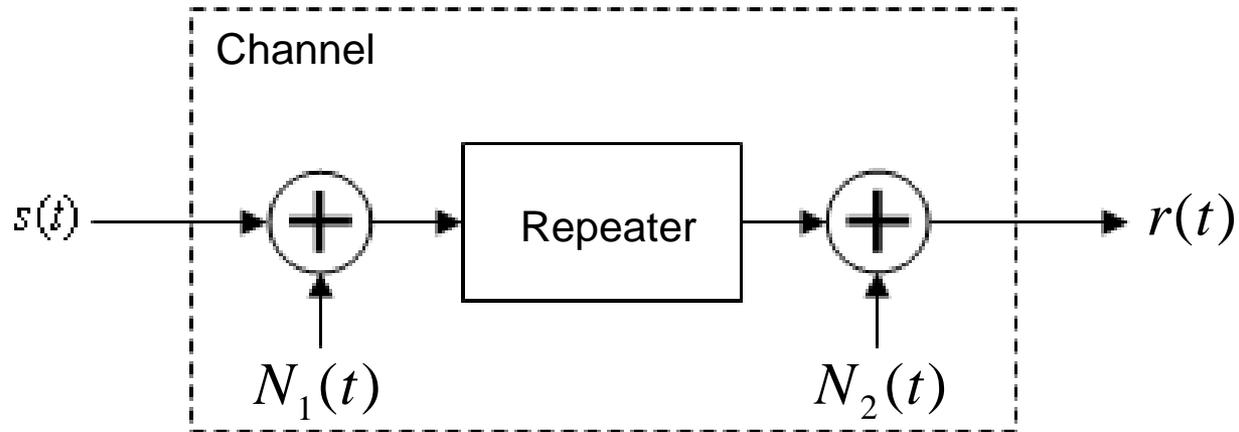
- Mathematical/Logical Processing on the data is possible
- Therefore : higher protection against noise
- More flexible when performed using reconfigurable / reprogrammable elements
- ?

## Disadvantages : (against analog communication)

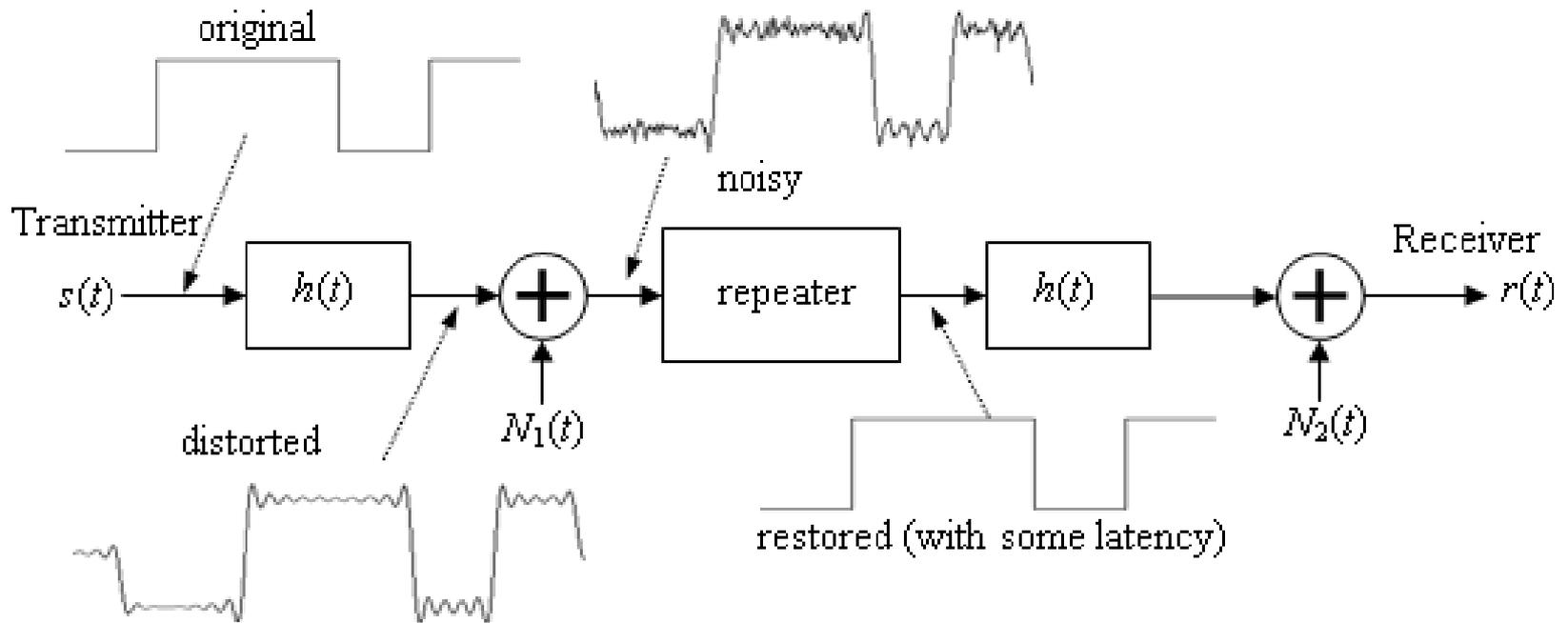
- Complexity is higher
- Higher speed devices are required
- Sometimes analog signals need to be converted/deconverted using ADC/DAC
- ?

## An Advantage of Digital Communication

Signal can be restored and resent halfway between transmitter and receiver

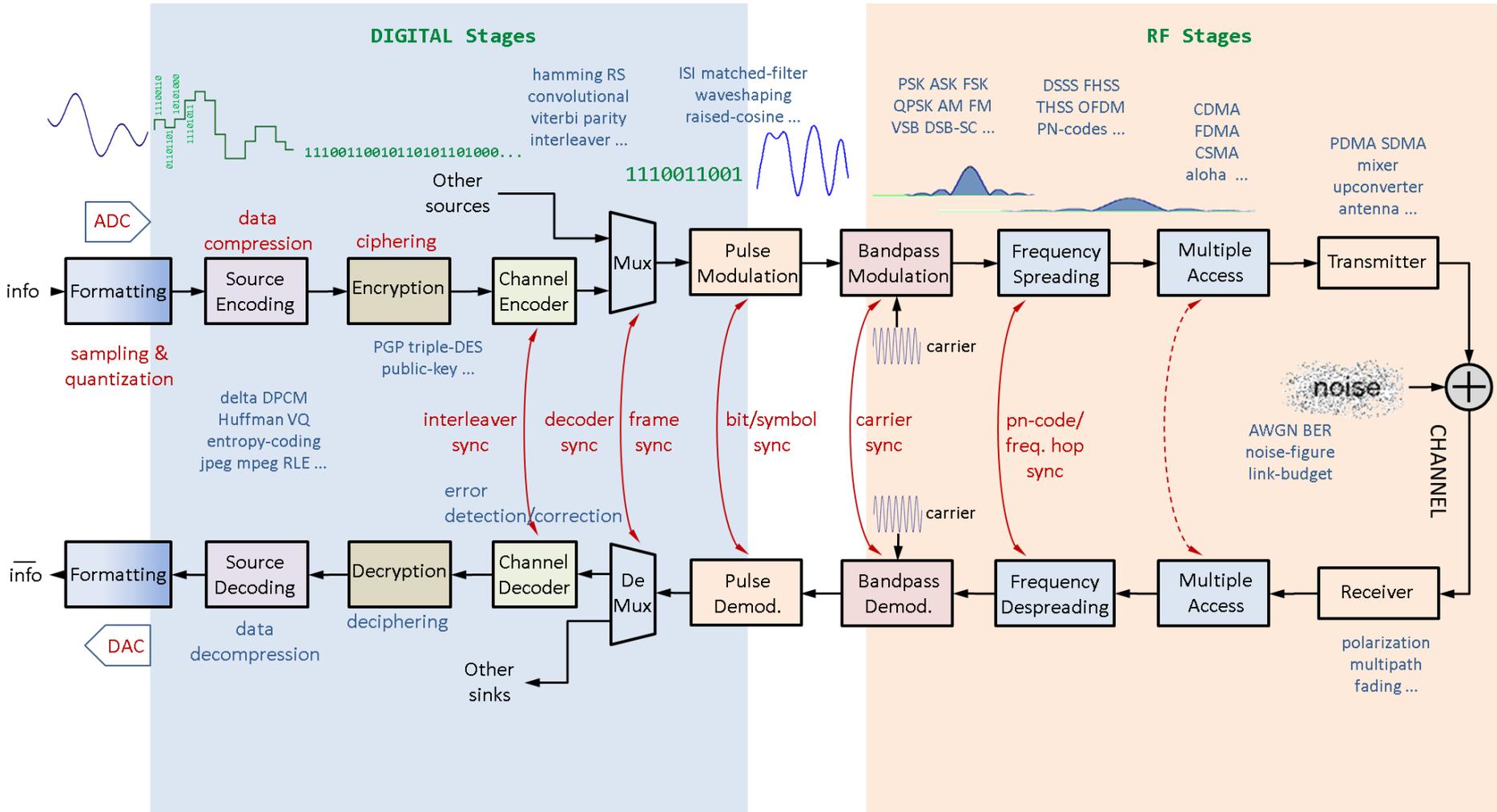


## An Advantage of Digital Communication



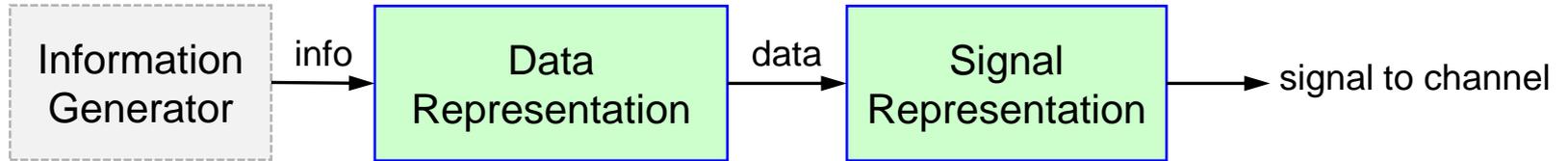
So that the signal is received with minimum (or no) error (but with additional delay)

# General Communication System



we will get back to this schema time-to-time.

# Information, Data and Signal



## Examples

idea → words → speech/voice

states → bits → electrical signals

voice → electrical signals → electrical signals

several **representation** changes may occur before the channel-signal output to channel

idea → words → speech/voice → electrical signals → bits → electrical signals

we are interested in signals-to-signals and states-to-signals paths in this course

# Simple Example

States

day



represented by 0



represented by

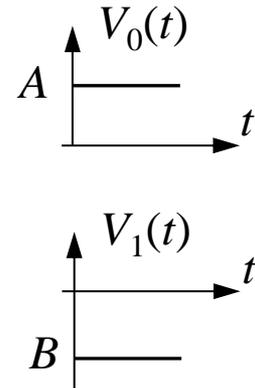
night



represented by 1



represented by



**Fact 1** : if it is always 'night', then nobody needs to share this information, that is there is no information to share

**Fact 2** : Information user must know what the signals mean (speak same language/symbols/signals etc)

## Simple Example

**A sentence** : "The sun will rise tomorrow"

**meaning** : The star that the earth rounds around will continue to exist and earth will continue to spin and no catastrophic event will occur to prevent that and our side of the earth will complete a considerable part of its rotation cycle, facing towards the sun. (probability=1)

The opposite of the above event has the probability of 0.

It turns out that there is no point of sharing this sentence as it does not contain any information unless the sentence has some epic meaning. For other meanings, of course, both sides must *speak the same language*.

**So**, what is *information*?

# Information, Data and Signal

**Fact** : In order for an event to be counted as informative event, its probability must be between (0,1) **excluding** both ends

**So**: To have a probability within (0,1) a complementing probability (opposite of the event) must exist

- \* So that the occurring event might change in the future
- \* So that the representative data might change in the future
- \* So that the representative signal might change in the future
- \* So that we cannot use constant/static/periodic signals.  
Something in the signal must change in time (the changes will represent information)



If there is no time of event, there is no <put anything here>.

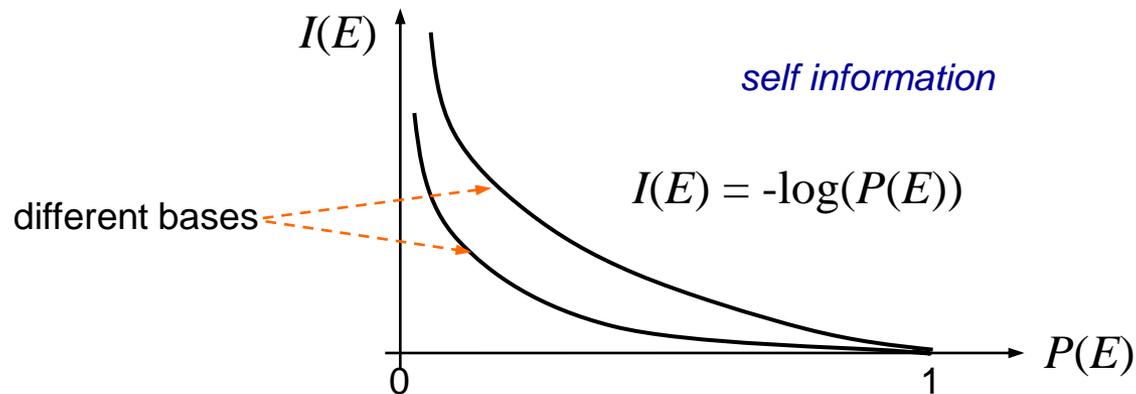
# Information

$P(E)$

$I(E)$

"Stocks will drop 0.5% tomorrow"    low information (happens everyday)

"Stocks will drop 25% tomorrow"    high information (rarely happens)



information is [unit]less quantity.

But in order to compare quantities we use the base of the logarithm as if it is a unit

$$I(E) = -\log_2(P(E)) \quad [\text{bits}] \quad (\text{information value in bits})$$

# Example

Expected grades in Communications Course (approximately)

AA : 5%	$I_{AA} = -\log_2(P_{AA}) = -\log_2(0.05) \sim 4.32$ bits
BA : 10%	
BB : 15%	
CB : 20%	
CC : 20%	$I_{CC} = -\log_2(P_{CC}) = -\log_2(0.2) \sim 2.32$ bits
DC : 15%	
DD : 5%	.. so on
FF : 10%	

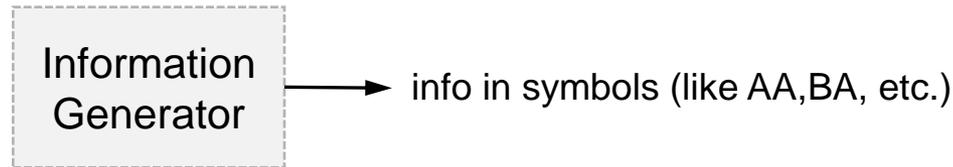
**Meaning** : When someone said "I got an AA", he/she actually transferred 4.32 bits worth of information to us.

**Question**: How much information does he/she transfer by telling all the grades?

**Answer**:  $\text{SumOf}(\text{All\_Info}) = \text{Info\_Student1} + \text{Info\_Student2} + \dots$   
 $= \text{Number\_of\_students} \times \text{Average\_Info\_Per\_Grade?}$

**Question**: What is the Average\_Info\_Per\_Grade?

# Average Information Per Source Output



Since we know the probabilities, we can calculate

$$I_{avg} = \sum_{n=1}^{N_{sym}} p_n I_n \quad (\text{weighted average})$$

$N_{sym}$  : the number of possible grades (8 in our example)

$$I_{avg} = - \sum_{n=1}^{N_{sym}} p_n \log_2(p_n)$$

we give it a special name : *entropy of the source*

which depends only on the symbol probabilities

and denote it as  $H(z)$  where  $z = \{p_n, n = 1, \dots, N_{sym}\}$

( in our example  $z = \{0.05, 0.1, 0.15, 0.2, 0.2, 0.15, 0.05, 0.1\}$  )

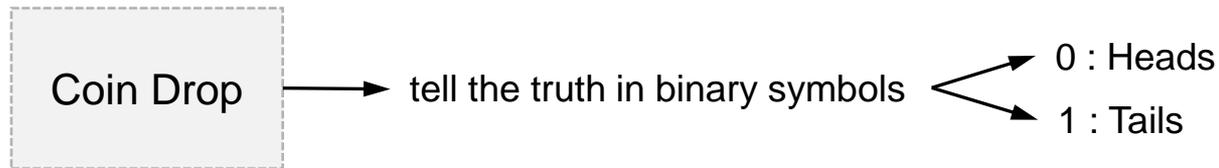
# Examples

We have 2 possible events : H, T with equal probabilities (like a coin drop)

$$I_H = -\log_2(0.5) = 1 \quad \text{and} \quad I_T = -\log_2(0.5) = 1 \quad \text{bit}$$

$$H(z) = I_{avg} = \sum_{n=1}^2 p_n I_n = 0.5 \times 1 + 0.5 \times 1 = 1 \quad \text{bit per symbol}$$

H can be represented by binary 0      T can be represented by binary 1



**Question:** What if the coin is not a fair one (probs are not equal)?

example :  $z = \{0.25, 0.75\}$

$$I_H = -\log_2(0.25) = 2 \quad I_T = -\log_2(0.75) \cong 0.415 \quad \text{bits}$$

oops, what does it mean to have 0.415 bits ?

# Examples

We have 8 possible symbols with equal probabilities of 0.125 each

$$I_s = -\log_2(0.125) = 3 \text{ bits for each symbol (logical)}$$

These can well be { 000, 001, 010, 011, 100, 101, 110, 111 }

or { 0, 1, 2, 3, 4, 5, 6, 7 } or { a, b, c, d, e, f, g, h } or ...

**The point is** : the symbols do not need to be represented in binary  
(although their info can be measured in bits)

**However** : we prefer binary since we use it all the time (in all digital systems).  
But that does not prevent us to create symbols like "01011" which  
might conveniently be represented by 01011 bit sequence.

**Question**: What if the symbol information values are not integers?

**Answer**: No problem. That all depends on what we want to do with  
them or how we represent them.

# Extensions

Extensions are constructed by having symbols together (side by side or in a bag)

example  $A = \{0, 1\}$

then  $B = \{000, 001, 010, 011, 100, 101, 110, 111\}$  (fixed length)

is a 3<sup>rd</sup> extension of binary alphabet  $A$

**Why ?** : To have more symbols, to have more efficient representations

Probabilities of newly created symbols are

$$u = \{p_{000}, p_{001}, p_{010}, p_{011}, p_{100}, p_{101}, p_{110}, p_{111}\}$$

$$p_{abc} = p_a p_b p_c$$

example  $z = \{0.25, 0.75\}$

$$p_{011} = p_0 p_1 p_1 = 0.25 \times 0.75 \times 0.75 \cong 0.14$$

# Extensions

Neither extensions nor original alphabet needs to have fixed length codes

example alphabet constructed of fixed length extensions of binary alphabet symbols

$$B = \{000, 001, 010, 011, 100, 101, 110, 111\}$$

example alphabets constructed from variable length extensions of binary alphabet symbols

$$C = \{00, 01, 011, 1011, 101, 11001, 110, 111\}$$

$$D = \{0, 1, 10, 11, 100, 101, 110, 111\}$$

We can have infinite number of alphabets representing the same source symbol-set

**Question** : So, What are their differences, advantages, disadvantages etc?

# Coding : Representations with Other Symbol Sets

Symbol	Code-1	Code-2	Code-3	Code-4	Code -5	Code -6	
$S_1$	000	0	1	1	0	00	
$S_2$	001	1	01	10	01	01	
$S_3$	010	10	001	100	011	10	...
$S_4$	011	11	0001	1000	0111	110	
$S_5$	100	100	00001	10000	01111	111	

↑
fixed length
}
variable length codes

**Coding** Representing symbols (or a sequence of symbols) from a symbol set with symbols (or a sequence of symbols) from another set

example

abc... → 123...

it is also good to have

123... → abc...

**Question:** Why are we doing it?

**Answer:** For efficient representation

# Average Code Length

$p_i$	Symbol	Code-1	Code-2	Code-3	Code-4	Code -5	Code -6	
0.36	$S_1$	000	0	1	1	0	00	
0.18	$S_2$	001	1	01	10	01	01	
0.17	$S_3$	010	10	001	100	011	10	...
0.16	$S_4$	011	11	0001	1000	0111	110	
0.13	$S_5$	100	100	00001	10000	01111	111	

$$L_{avg} = \sum_{n=1}^{N_{sym}} p_n l_n = 3 \text{ bits for Code-1}$$

$$L_{avg} = \sum_{n=1}^{N_{sym}} p_n l_n = 2.29 \text{ bits for Code-6}$$

so, using Code-6 is better

$$H(z) = 2.216$$

**Why** not use Code-2 then? It looks like it will result a shorter average code length

**Because** Code-2 is not uniquely decodable when transferred consecutively

123... ~~→~~ abc...

# Unique Decodability

Let us have an information source generating symbols from the alphabet

$$A = \{s_1, s_2, s_3, s_4, s_5\}$$

with the probabilities of  $u = \{0.36, 0.16, 0.17, 0.16, 0.13\}$

Assume that the source has generated the sequence of  $s_1 s_2 s_3 s_1 s_1 s_5 s_4$

Coding the symbols with Code-2, we would have : 0, 1, 10, 0, 0, 100, 11

or a binary sequence of : **01100010011**

We would like to decode the sequence **01100010011** back to  $s_1 s_2 s_3 s_1 s_1 s_5 s_4$

remembering that we do not have symbol separators, we immediately see that it is impossible to decode it back to original

So, the Code-2 is not uniquely decodable (that means it is nearly useless)

# Corollary

We need to have uniquely decodable codes with lower (than original) average code-lengths

Let us examine the previous code table again

$p_i$	Symbol	Code-1	Code-2	Code-3	Code-4	Code-5	Code-6	
0.36	$S_1$	000	0	1	1	0	00	
0.18	$S_2$	001	1	01	10	01	01	
0.17	$S_3$	010	10	001	100	011	10	...
0.16	$S_4$	011	11	0001	1000	0111	110	
0.13	$S_5$	100	100	00001	10000	01111	111	

Code-1 : uniquely decodable,  $L_{avg} = 3$ , fixed-length

Code-2 : **not** uniquely decodable,  $L_{avg} = ?$ , variable-length, not instantaneous\*

Code-3 : uniquely decodable,  $L_{avg} = x$ , variable-length

Code-4 : uniquely decodable,  $L_{avg} = x$ , variable-length, not instantaneous\*

Code-5 : uniquely decodable,  $L_{avg} = x$ , variable-length, not instantaneous\*

Code-6 : uniquely decodable,  $L_{avg} = 2.29$ , variable-length

The code is considered **instantaneous** if the symbols can be determined when their last bits are received

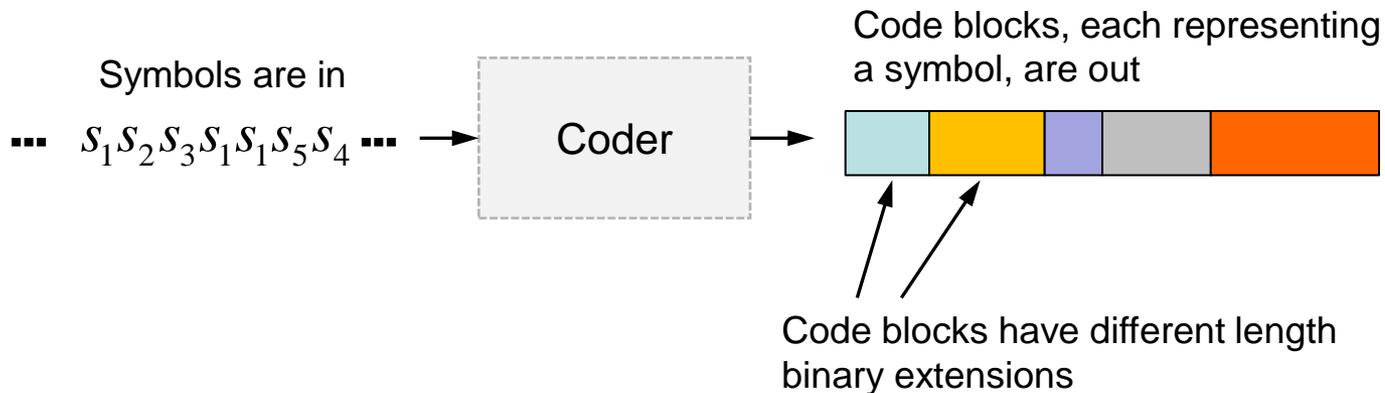
# Minimum Average Code Length

We see that we can have infinite number of Codes that are uniquely decodable.  
We also need to have efficient representation (smaller average code length)

**Question** : Is there a way to find a code with minimum average code length?

**Answer**: Yes for block codes

*block code* : symbol-to-binarycode representation  
(implying that there are other (non-block) codes as well)



# Example

Symbols	Probs	New symbols /code
00	0.49	?
01	0.21	?
10	0.21	?
11	0.09	?

We would like to determine a code for each symbol, which, for the given probabilities, best represents the self-information of the symbol.

**A method** : divide the pre-ordered set of probabilities into two so that sum of probabilities on both sides are as close as possible. Continue doing that until there is only one in each division

$p_i$	prefixes		
0.49	0		
0.21	1	0	
0.21	1	1	0
0.09	1	1	1

## Generated code table

00	0
01	10
10	110
11	111

Now we have code for each input symbol to replace with

Code is variable length and uniquely decodable. The method is called *Shannon-Fano* coding\*

\*:it is actually code-table preparation

prefix sides with 0 or 1  
and do it again

## Example

$p_i$	symbol	
0.7	0	cannot do anything here in terms of efficient representation $L_{avg} = 1$
0.3	1	

But : Shannon states that "wider the extension, better the representation"

Let us find 2<sup>nd</sup> extension of the ensemble  $(A, z)$  where  $z = \{0.7, 0.3\}$

$p_i$	symbol	code	
0.49	00	0	$L_{avg} = \sum_{i=1}^4 P(s_i) L(s_i)$ $L_{avg} = 0.49 \times 1 + 0.21 \times 2 + 0.21 \times 3 + 0.09 \times 3 = 1.81$ a single input bit is now represented by $1.81 / 2 = 0.905$ bits now.
0.21	01	10	
0.21	10	110	
0.09	11	111	

Let us now test this argument with the 2<sup>nd</sup> extension of the 2<sup>nd</sup> extension (or the 4<sup>th</sup> extension of the original binary alphabet)

## Example

Probs	code table	
0.2401	0000	00
0.1029	0001	010
0.1029	0010	0110
0.1029	0011	0111
0.1029	0100	1...
0.0441	.	1...
0.0441	.	1...
0.0441	.	1...
0.0441	.	1...
0.0441	.	1...
0.0441	.	1...
0.0441	.	1...
0.0189	.	1...
0.0189	.	1...
0.0189	.	1...
0.0189	.	1...
0.0081	1111	1...

$$L_{avg} = 3.5948 \text{ bits/symbol}$$

a single input bit is now represented by  
 $3.5948 / 4 = 0.8987$  bits

we see that it is getting **better**

The entropy is  $H(u) = 3.5252$

so, we still have room for improvement

It is guaranteed that extensions of  $n > 4$  will  
 have better representations

**But** cannot be lower than  $H$

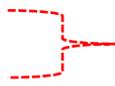
**hmw:** complete the table

# Huffman Code Table

It is proven that Huffman's code generates smallest ACL among dictionary-based statistical block codes

## Example

	<u>Probs</u>
$s_1$	0.49
$s_2$	0.21
$s_3$	0.21
$s_4$	0.09

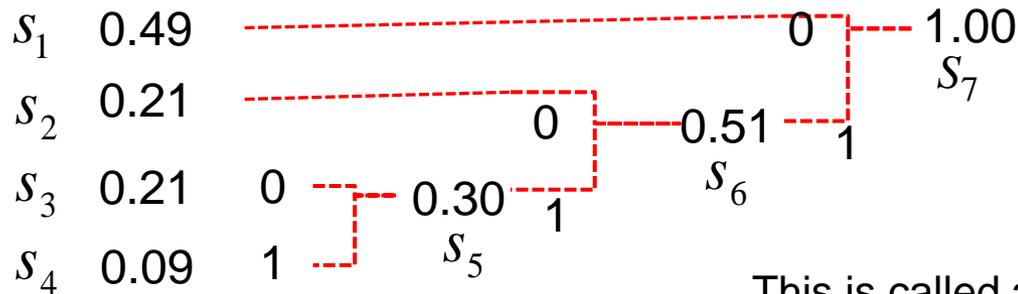
 Let these symbols with smallest probabilities be a single symbol  $s_5$

Its probability would be 0.30

But they are actually two symbols and when its code is seen at the decoder we need to have a bit to differentiate them

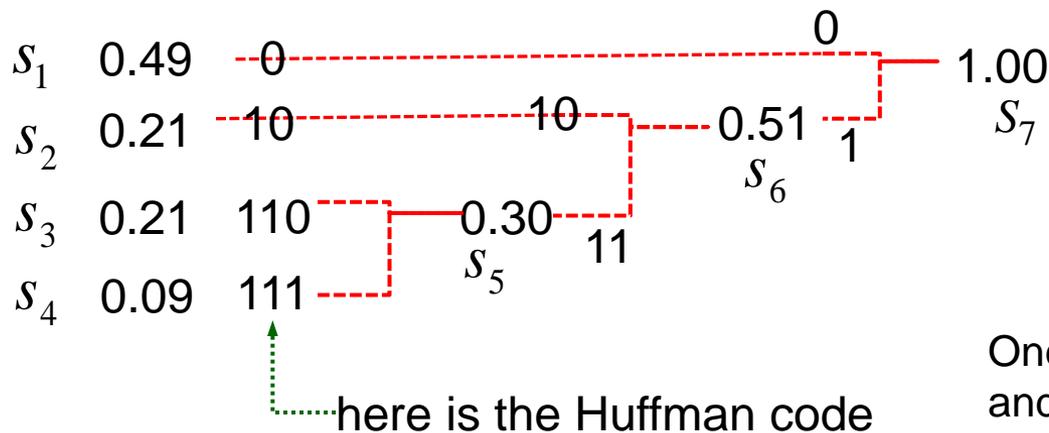
		additional bit	
$s_3$	0.21	0	 $s_5$ 0.30
$s_4$	0.09	1	

Now we have three symbols. Continue combining symbols with smallest probs and prepending differentiating bit marks.



This is called a Huffman tree

Now, from right to left, follow the paths for each symbol and find assigned bits to them by appending each bit to the right (LSB)



One can create the tree first and assign bits later

We see that the ACL is same as the code found using Shannon-Fano. It is guaranteed that Huffman method generates shorter or same length codes

$$L_{avg} = \sum_{i=1}^4 P(s_i)L(s_i) \quad 0.49 \times 1 + 0.21 \times 2 + 0.21 \times 3 + 0.09 \times 3 = 1.81$$

Here is an Example where two methods generate different code lengths

		SF	Huf
$s_1$	0.36	00	0
$s_2$	0.18	01	100
$s_3$	0.17	10	101
$s_4$	0.16	110	110
$s_5$	0.13	111	111

$$H(z) \leq L_{avgHuf} \leq L_{avgSF}$$

$$H(z) = 2.216$$

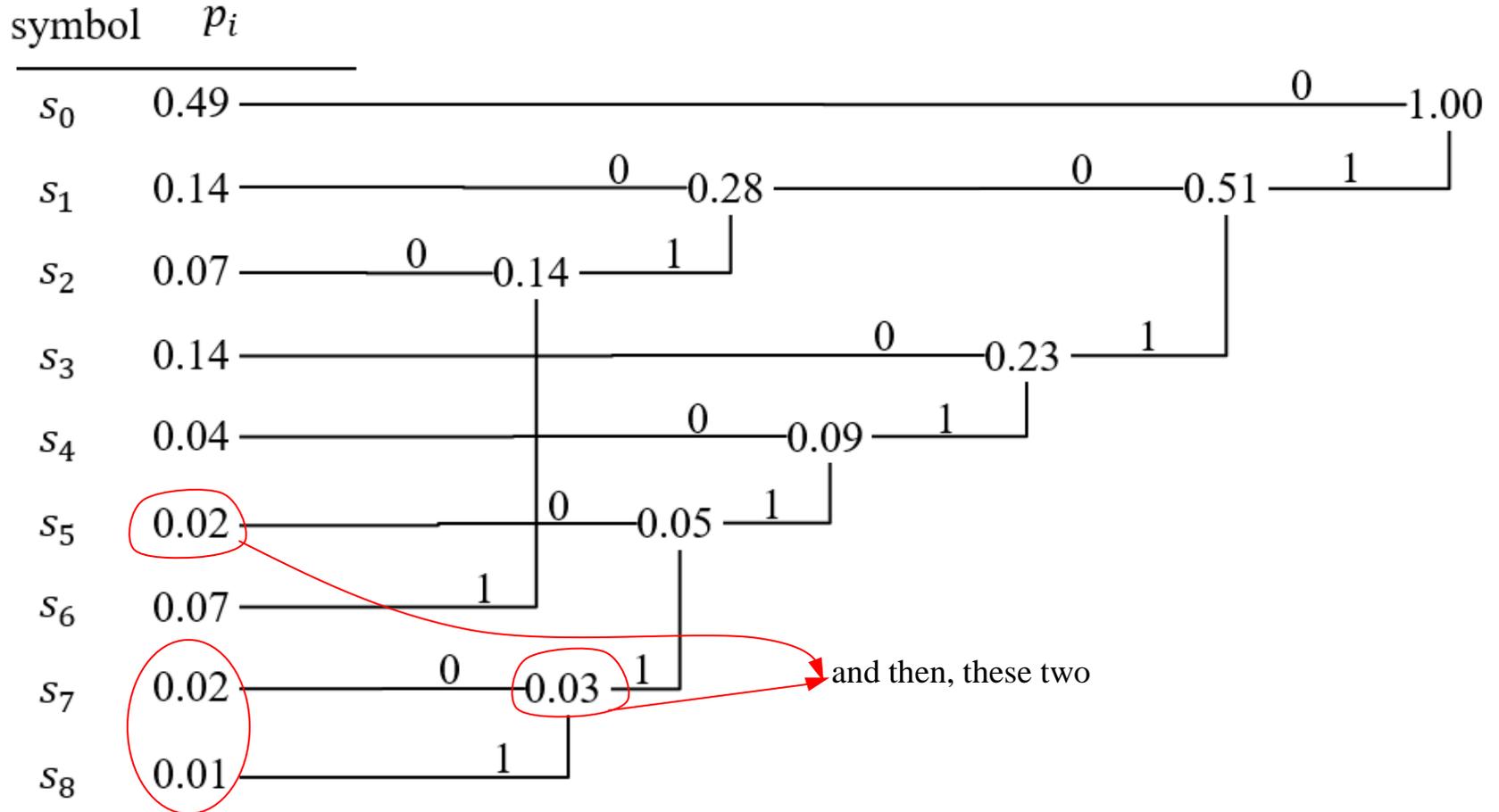
$$L_{avgHuf} = 2.28$$

$$L_{avgSF} = 2.29$$

Hmw: check validity

# Example

$$u = \{0.49, 0.14, 0.07, 0.14, 0.04, 0.02, 0.07, 0.02, 0.01\}$$

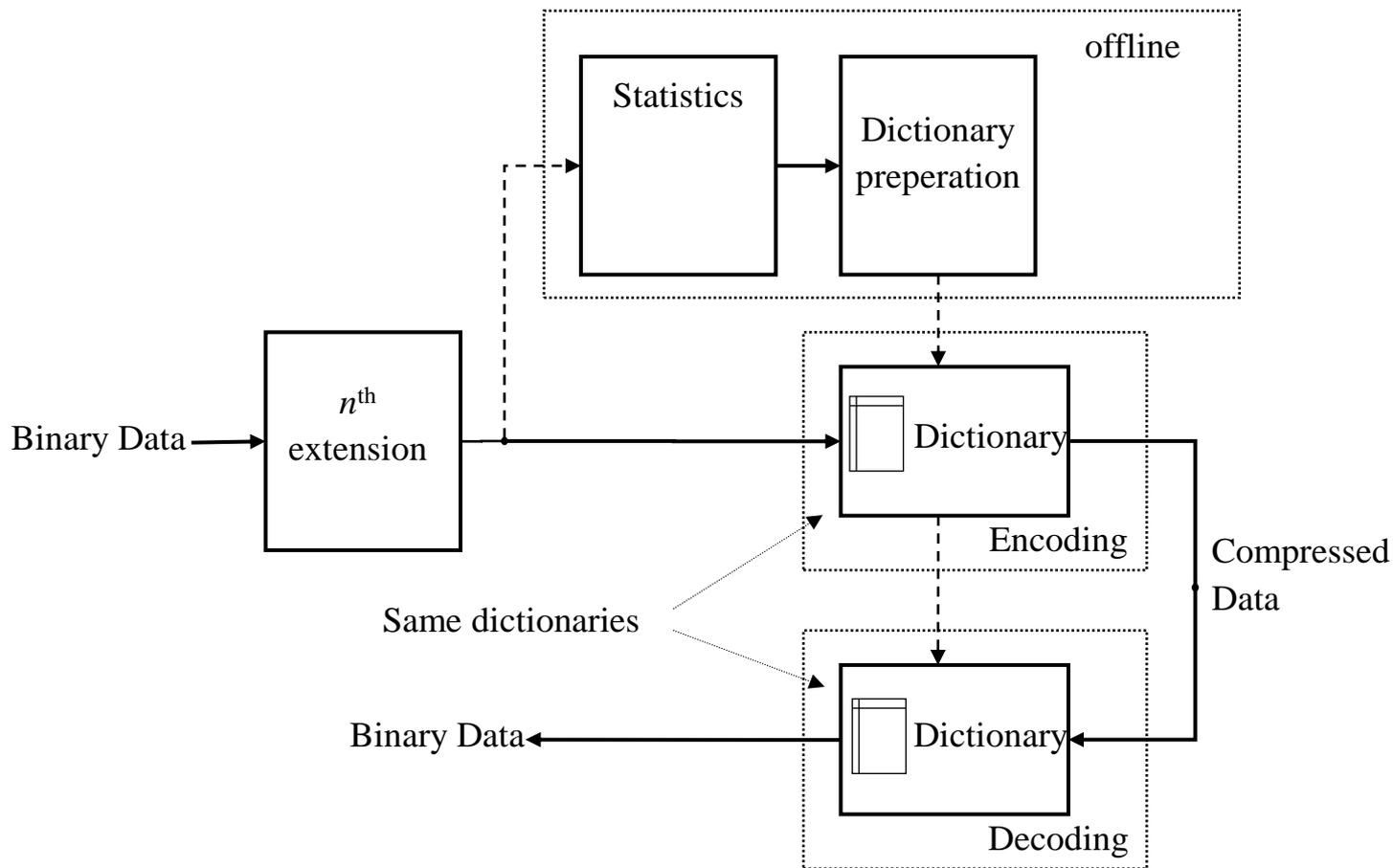


start by combining these two smallest

and then, these two

# Flow of Dictionary Coding

(Compression/Decompression)



# Summary

## What we have seen/learned

1. Information is the entity we need to communicate
2. We measure the information using probabilities of events
3. Average information of a source is the weighted (by their probabilities) self-information of events
4. Average Info Per source output is called Entropy of the source
5. Unique decodability is a requirement but instantaneous decodability is not.
6. Block codes are the simplest but not the only coding method.
7. Efficiency of block coding can be improved by extensions.
8.  $ACL \geq \text{Entropy}$
9. Encoded information must be represented by electrical waveforms for electronic-communication

## Related

$\Sigma\Delta$ -Coding, Channel Coding, Waveform representations

## Next

Demonstration of speed of electromagnetic waves

**END**