

# Information Theory & Coding...

Module 6

# At the end you will know

- ▶ What is information and how its important.
- ▶ What is data compression – it's aspect; entropy.
- ▶ Some coding techniques.
- ▶ Data transmission through channel and its reception – our goal to achieve.

# Information Theory

- ▶ Invented by C.E. Shannon in late 1940s.
- ▶ Information theory deals with mathematical modeling and analysis of a communication system rather than physical sources and channels.
- ▶ In particular it helps us to find –
  - The amount of information contained in the signal.
  - The irreducible complexity below which the signal cant be compressed.
  - The ultimate transmission rate for reliable communication over a noisy channel.

# Information Theory

Information theory answers two fundamental questions:

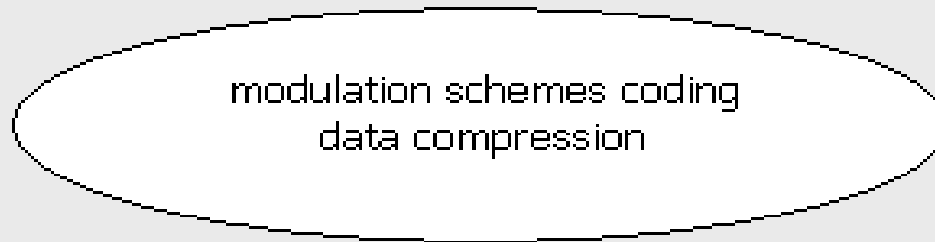
- ▶ What is the ultimate data compression?  
Answer: The Entropy  $H$ .
- ▶ What is the ultimate transmission rate?  
Answer: Channel Capacity  $C$ .

The concept that *increasing transmission rate over a channel increases the error rate*. Shannon showed that this is not true as long as rate is *below Channel Capacity*.

Shannon has further shown that random processes have an irreducible complexity below which they can not be compressed.

data  
compression  
limit

Entropy



data  
transmission  
limit

Channel  
Capacity

- ▶ We assume a *discrete memory less source* , memory less in the sense that the symbol emitted at any point of time is independent of previous choices.
- ▶ The source output as emitted during every signaling interval is modeled as a discrete random variable,  $S$ , which takes on symbols from a fixed finite alphabet  $\xi = \{s_0, s_1, s_2 \dots s_{K-1}\}$  with probabilities

$$P(S=s_k) = p_k \quad k=0,1,2,3,\dots$$

and

$$\sum_{k=0}^{K-1} p_k = 1$$

# Information

- ▶ For an event  $S = s_k$  with the symbol  $s_k$  emitted by the source with probability  $p_k$ . Then if  $p_k=1$  and  $p_i = 0$  and for all  $i \neq k$ , then there is no information because we know what the message is going to be from the source.
- ▶ But if  $p_k$  is low, then there is more surprise and so the information when  $s_k$  is emitted by the source than the symbol  $s_i$  which is having the higher probability to be emitted.
- ▶ The amount of information is related to the inverse of the probability of occurrence as given by

$$I(s_k) = \log\left(\frac{1}{p_k}\right) = -\log(p_k)$$

# Information

- ▶ So,  $I(s_k) > I(s_i)$   
for  
 $p_k < p_i$
- ▶ When  $p_k = 1/2$ , we have  $I(s_k) = 1$ , i.e., *one bit of information that we gain* when one of *two* possible and equally likely events occur.
- ▶ And  $I(s_k) \geq 0; 0 \leq p_k < 1$



# Entropy...

- ▶ The *amount of potential information contained in a signal* is termed the *entropy*, usually denoted by  $H$ , which is defined as follows:

$$H(X) = - \sum_X P(X) \log P(X)$$

- ▶ We can essentially think of this as a weighted average of  $\log 1/P(X)$ . The quantity  $1/P(X)$  expresses the amount of “surprise” in an event  $X$ —i.e., if the probability is low, then there is a lot of surprise, and consequently a lot of information is conveyed by telling you that  $X$  happened.
- ▶ It's a measure of the *average information content per source symbol*.
- ▶ The reason we are taking the log of the surprise is so that the total amount of surprise from independent events is additive. Logs convert multiplication to addition, so

$$\log \frac{1}{P(X_1 X_2)} = \log \frac{1}{P(X_1)} + \log \frac{1}{P(X_2)}.$$

# Entropy

- ▶ Thus, entropy essentially measures the “average surprise” or “average uncertainty” of a random variable.
- ▶ If the distribution  $P(X)$  is highly peaked around one value, then we will rarely be surprised by this variable, hence it would be incapable of conveying much information.
- ▶ If on the other hand  $P(X)$  is uniformly distributed, then we will be most surprised on average by this variable, hence it could potentially convey a lot of information.

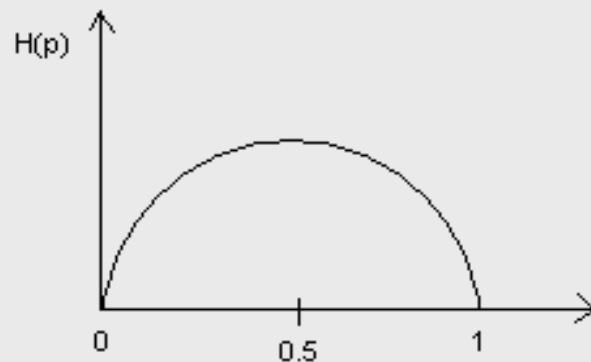
## ➤ Properties of Entropy

▶ **(P1)**  $H(x) \geq 0$

$$0 \leq p(x) \leq 1 \rightarrow \log [ 1/p(x) ] \geq 0$$

▶ **[E]**  $x = \begin{pmatrix} 1 & p \\ 0 & 1-p \end{pmatrix}$

$$\begin{aligned} H(x) &= -p \log p - (1-p) \log(1-p) \\ &= H(p) - \text{the entropy function.} \end{aligned}$$



# Entropy

- ▶ Limiting factor:
  - *Entropy is 0*, if and only if  $p_k=1$  for some  $k$  and remaining probabilities in the set are all 0 – this means there is *no uncertainty* in the information.
  - *Entropy is  $\log_2 K$* , if and only if  $p_k=1/K$  for all  $k$  i.e., all the symbols in the alphabet are equi-probable; this upper bound on entropy corresponds to *maximum information*.

# Example

- ▶ Consider a discrete memory less source with source alphabet  $\xi = \{s_0, s_1, s_2\}$  with respective probabilities  $p_0 = 1/4$  ;  $p_1 = 1/4$  &  $p_2 = 1/2$ .
- ▶ The entropy of the source becomes –

$$\begin{aligned} H(\xi) &= p_0 \log_2\left(\frac{1}{p_0}\right) + p_1 \log_2\left(\frac{1}{p_1}\right) + p_2 \log_2\left(\frac{1}{p_2}\right) \\ &= \frac{1}{4} \log_2(4) + \frac{1}{4} \log_2(4) + \frac{1}{2} \log_2(2) \\ &= \frac{3}{2} \end{aligned}$$

# Information Rate

- ▶ If a source of message generates messages at the rate  $r$  messages per second then the information rate is defined as

$R = rH$  = average no of bits of information / second.

where,  $H$  stand for entropy of the source.

# Source Coding...

- ▶ From a transmitters point of view data of the source is compressed in a source encoder.
- ▶ For efficient compression we need the knowledge of entropy or the statistics of the source.
- ▶ For an example, we can assign short code to frequent source codes and longer code to infrequent one – this is called variable length code.

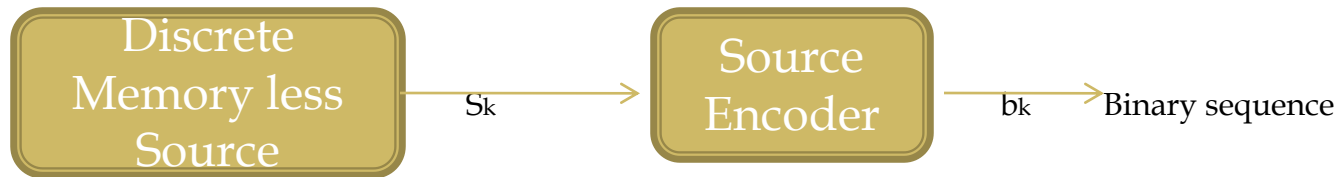
# Requirement of Efficient Source Coding...

- ▶ The primary requirement for a efficient source encoder is that –
  - The code words produced by the encoder is in binary form.
  - The source code is uniquely decodable, so that the original source sequence can be reconstructed perfectly from the encoded binary sequence.



# Coding Efficiency

- ▶ We have the following source encoder –



the above source is having an alphabet with  $K$  different symbols and  $k$  -th symbol  $s_k$  occurs with probability  $p_k$ ,  $k=0,1,2,3\dots K-1$ .

let the binary codeword assigned to symbol  $s_k$  by the encoder have length  $l_k$ , measured in bits.

# Coding Efficiency...

- So the average codeword length  $\bar{L}$  of the source encoder as – 
$$\bar{L} = \sum_{k=0}^{K-1} p_k l_k$$

in physical terms it represents the average no of bits per source symbol.

if  $L_{\min}$  the minimum possible value for  $\bar{L}$  , then coding efficiency of the source encoder is given by-

$$\eta = \frac{L_{\min}}{\bar{L}}$$

with  $\bar{L} \geq L_{\min}$  we have  $\eta \leq 1$  . encoder is said to be efficient when efficiency approaches unity.

# Shannon's First Theorem

- ▶ Shannon's first theorem on coding efficiency gives the idea to calculate  $L_{\min}$ . It is stated as –
  - *Given a discrete memory less source of entropy  $H(\xi)$ , the average codeword length  $\bar{L}$  for any distortion less source coding scheme is bounded as*

$$\bar{L} \geq H(\xi)$$

So, according to the source coding theorem, the entropy represents a *fundamental limit on the average no of bits per source symbol* necessary to represent a discrete memory less source in that it can be made as small as, but *no smaller than the entropy*, so with  $L_{\min} = H(\xi)$ ,

$$\eta = \frac{H(\xi)}{\bar{L}}$$

# Data Compaction

- ▶ For efficient transmission redundant data should be removed from the signal prior to transmission.
- ▶ This operation with no loss of information is *performed in digital form* and is called as *data compaction or lossless data compression*.
- ▶ This operation provides the source output with average no of bits/symbol and also original data could be reconstructed from these data with no loss of information.
- ▶ So short descriptions are provided to the most frequent outcomes and longer for the less frequent ones.

# Prefix Coding...

- ▶ Any sequence made up of the initial part of the code is called the prefix of the code and A *Prefix Code* is defined as a code in which no code word is the prefix of any other code word.
- ▶ Example of Prefix Code:

Source Symbol	Probability	Code I	Code II	Code III
s0	0.5	0	0	0
s1	0.25	1	10	01
s2	0.125	00	110	011
s3	0.125	11	111	0111

# example

- ▶ *Code I* : its not a prefix code since the bit 0, the codeword for  $s_0$ , is a prefix of 00 ( $s_2$ ). Again the bit 1, the codeword for  $s_1$  is the prefix of 11 ( $s_3$ ).
- ▶ *Code III*: its also not a prefix code. Justify...
- ▶ *Code II*: it's a prefix code for the given condition that no code is the prefix of any other.

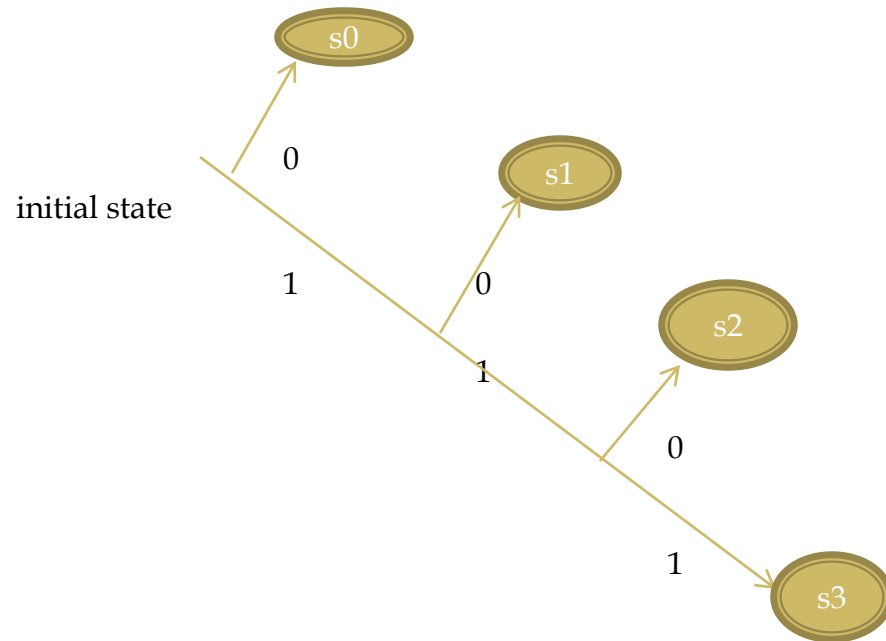
# Decoding...

- ▶ The important property of this code is that its *uniquely decodable*.
- ▶ To decode a sequence of code words generated from a prefix source code, the source decoder *starts at the beginning of the sequence and decodes one code word at a time*.
- ▶ This way it sets up a decision tree which is a graphical portrayal of code words in the particular code.

- ▶ The first received bit moves the decoder to the terminal state  $s_0$ , if its 0, or else to a second decision point if its 1.
- ▶ In the latter case, the second bit moves the decoder one step further down the tree, i.e., either to  $s_1$  if its 0 or to a third decision point if its 1. and so on.....
- ▶ Once each terminal state emits its symbol the decoder goes back to its initial state.
- ▶ The end of the prefix code is always recognizable. Hence, *decoding could be accomplished as soon as the binary sequence representing the source symbol is fully received.* So its also called the instantaneous code.



# Decoding Tree...



# Problem...

- ▶ Decode the sequence 1011111000 and verify it will produce  $s_1s_3s_2s_0s_0\dots$

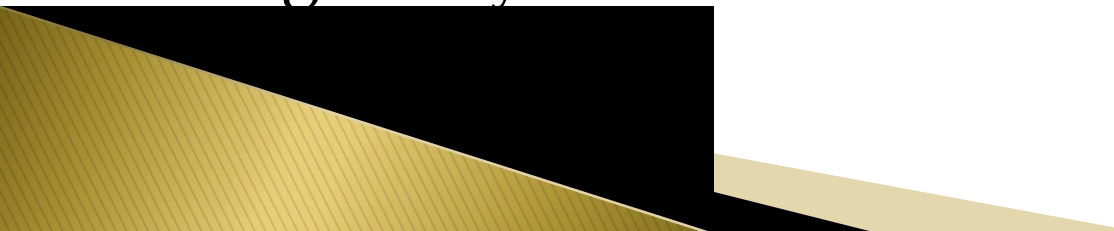
- ▶ we can prove

$$\lim_{n \rightarrow \infty} \frac{1}{n} \overline{L_n} = H(\xi)$$

- ▶ So, by making the order  $n$  of an extended prefix source encoder large enough, we can make the code faithfully represent the discrete memory less source as close as desired.
- ▶ In other words the average code word length of an extended prefix code can be made as small as the entropy of the source provided the extended code has a high enough order.
- ▶ But the disadvantage of extended code is increased decoding complexity.

# Hoffman Coding

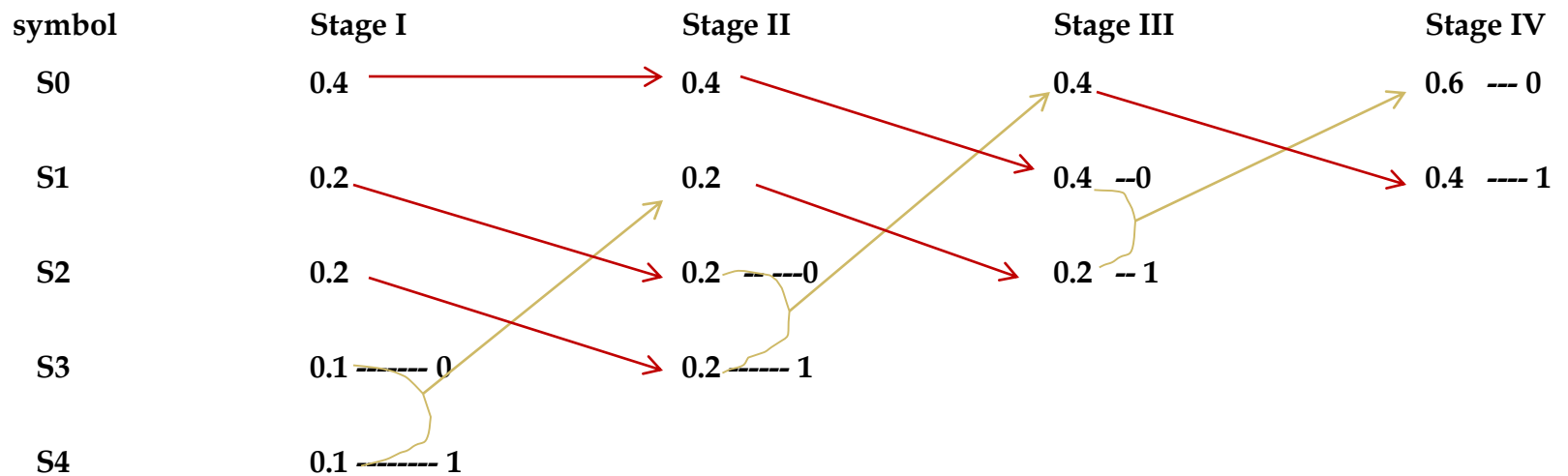
- ▶ Its one important class of prefix code.
- ▶ The basic idea behind the Hoffman coding is to assign to each of an alphabet a sequence of bits roughly equal to the amount of information conveyed by it.
- ▶ The end result is a source code whose average code word length approaches the fundamental limit set by the entropy of the memory less source namely entropy.

- ▶ For given set of symbols with different probabilities there is a reduction process applied here to generate the code.
  - ▶ The reduction process is continued in a step by step manner until we are left with a final set of two source symbols and they are assigned with 0 and 1.
  - ▶ From this trivial last stage we then go back to the original symbol to find its code word.
- 

# Huffman Encoding process...

- ▶ The source symbols are listed in order of decreasing probability. The two source symbols of lowest probability are assigned a 0 and a 1 – this is the splitting phase.
- ▶ These two source symbols are regarded as being combined to form a new source symbol with probability equal to the sum of two original probabilities (thus it in effect reduces the no of source symbols by one).
- ▶ This probability value of the new symbol is placed in the list according to its value.
- ▶ This *process is continued until* we are left with a final list of source symbols of only two for which we assign 0 and 1.

# Hoffman Coding process...



# Contd...

Symbol	Probability	Code word
s0	0.4	00
s1	0.2	10
s2	0.2	11
s3	0.1	010
s4	0.1	011



average codeword length is therefore –

$$\begin{aligned}\bar{L} &= 0.4(2) + 0.2(2) + 0.2(2) + 0.1(3) + 0.1(3) \\ &= 2.2\end{aligned}$$

and entropy –

$$\begin{aligned}H(\xi) &= 0.4 \log_2\left(\frac{1}{0.4}\right) + 0.2 \log_2\left(\frac{1}{0.2}\right) + 0.2 \log_2\left(\frac{1}{0.2}\right) + \\ &\quad 0.1 \log_2\left(\frac{1}{0.1}\right) + 0.1 \log_2\left(\frac{1}{0.1}\right) \\ &= 0.52877 + 0.46439 + 0.46439 + 0.33219 + 0.33219 \\ &= 2.12193\end{aligned}$$

efficiency –

$$\eta = \frac{H(\xi)}{\bar{L}} = 2.1293 / 2.2 = 0.9678$$

### 13.2.1 Shannon-Fano Coding

One basic technique by which we may generate such a more "efficient" code is the Shannon-Fano algorithm. An example of the application of this algorithm is given in Fig. 13.2. Here we consider that there are 8 possible messages  $m_1$  through  $m_8$  with probabilities as given. We have ordered the messages in order of decreasing probability. In column I we divide the message into two parts such that the sum of the probabilities of each group are the same. Thus,  $m_1$  in one partition has the probability  $\frac{1}{2}$  and the sum of the probabilities of all the other messages in the second

Message	Probability	I	II	III	IV	V	No of bits/message
$m_1$	1/2	0					1
$m_2$	1/8	1	0	0			3
$m_3$	1/8	1	0	1			3
$m_4$	1/16	1	1	0	0		4
$m_5$	1/16	1	1	0	1		4
$m_6$	1/16	1	1	1	0		4
$m_7$	1/32	1	1	1	1	0	5
$m_8$	1/32	1	1	1	1	1	5

Fig. 13.2 An example of the application of the Shannon-Fano algorithm.

partition is also  $\frac{1}{2}$ . We assign the bit 0 to the message in one partition and we assign the bit 1 to all the messages in the other partition. This process of dividing groups in the same partition into two partitions each with equal sums of probabilities is continued, until each message finds itself alone in a partition. At each partitioning, one group has a zero assigned while a 1 is assigned to the other. In the example of Fig. 13.2 five partitionings are required. We find that  $m_1$  is represented by the single bit 0,  $m_2$  is represented by the 3 bit code 100, etc., up to  $m_8$  whose code is the 5 bit word 11111.

We now find that with this code, the average information per message interval is

$$\begin{aligned}
 H &= \sum_{i=1}^{\infty} p_i \log_2 \frac{1}{p_i} = \frac{1}{2} \log_2 2 + 2 \times \frac{1}{8} \log_2 2^3 + 3 \times \frac{1}{16} \log_2 2^4 + 2 \times \frac{1}{32} \log_2 2^5 \\
 &= 2 \frac{5}{16}
 \end{aligned} \tag{13.12}$$

We find further that the average number of bits per message is

$$\begin{aligned}
 \text{aver. bits/message} &= 1\left(\frac{1}{2}\right) + 3\left(\frac{1}{8}\right) + 3\left(\frac{1}{8}\right) + 4\left(\frac{1}{16}\right) + 4\left(\frac{1}{16}\right) + 4\left(\frac{1}{16}\right) + 5\left(\frac{1}{32}\right) + 5\left(\frac{1}{32}\right) \\
 &= 2 \frac{5}{16}
 \end{aligned} \tag{13.13}$$

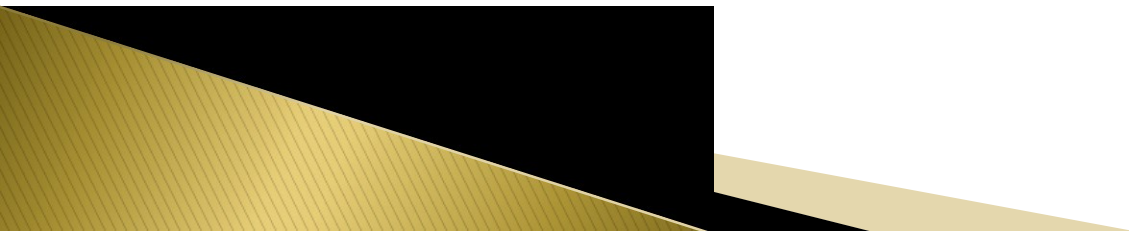
## ► Shannon's Second Theorem –

- Let a discrete memory less source with an alphabet of  $S$  have entropy  $H$  and produce symbols once in every  $T_s$  seconds i.e. information rate  $R=H/T_s$ . Let a discrete memory less channel have capacity  $C$  Then, if

$$R \leq C$$

then there exists a coding scheme for which the source output could be transmitted over the channel and be reconstructed with an arbitrarily small probability of error.

- Its also called the *channel coding theorem* and is valid for a discrete memory less channel.
- It specifies the channel capacity as the fundamental limit on the rate at which the transmission of reliable error free messages can take place over a discrete memory less channel.



# Capacity of a Gaussian Channel

- ▶ According to Shannon Hartley Theorem, the channel capacity of white, band limited Gaussian channel is given by-

$$\underline{C = B \log_2 (1 + S/N) \text{ bits/sec}}$$

where,  $B$  = channel band width (BW)

$S$  = signal power

$N$  = total noise in the channel BW.

i.e.,  $N = \eta B$  with  $\eta/2$  is power spectral density



# Application...

- ▶ Generally physical channels we use are at least approximately Gaussian.
- ▶ According to the expression we have for Gaussian channel the channel capacity could be increased if
  - We increase SNR of the signal, or
  - We increase BW of the channel.
- ▶ But for a given channel with a given noise distribution its not possible to increase the signal power infinitely because of the design issue.
- ▶ We can think of increasing the channel BW infinitely, but for a channel with infinite BW we find the channel capacity is limited to  $1.44 S/\eta$ .

# BW-SNR Trade-off

- ▶ For a *Gaussian noise less channel* ( $S/N = \infty$ ) has an *infinite capacity*. On the other hand when the *channel capacity* does increase, it *does not become infinite as the BW becomes infinite*, because with an increase in BW the *noise power also increases*.
- ▶ We find for a fixed signal power in the presence of white Gaussian noise the channel capacity approaches an upper limit with increasing BW given as  $1.44 S/\eta$ .



- ▶ Also we find that
  - if  $S/N = 7$  and  $B=4\text{kHz}$  then  $C=12 \times 10^3$ .
  - And if we increase SNR to  $S/N = 15$  and  $B$  is decreased to  $3\text{kHz}$  then also channel capacity remains same.
  - Thus with a  $3\text{kHz}$  BW the noise power will be  $\frac{3}{4}$  as large as with  $4\text{kHz}$ . Thus signal power has to be increased by a factor of  $(\frac{3}{4} \times \frac{15}{7})=1.6$ . i.e., 25% reduction in BW requires a 60% increase in signal power.