Digital Communication

Professor Surendra Prasad
Department of Electrical Engineering
Indian Institute of Technology Delhi

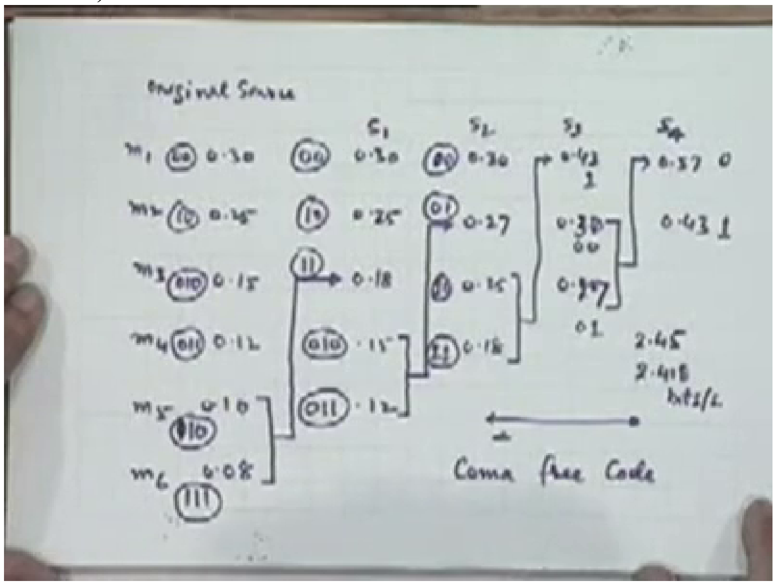
Lecture No 35

On the Possibility of Error Free Communication over a Noisy Channel

(Refer Slide Time 01:03)

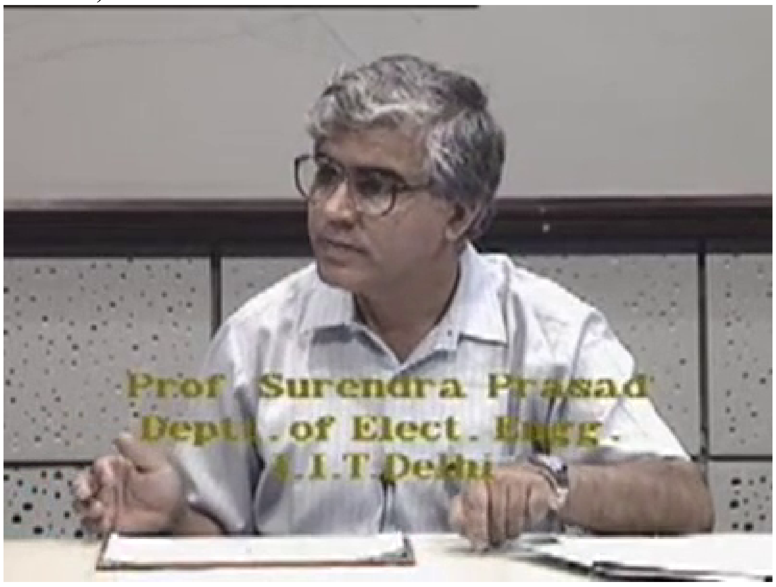
So if you recollect we were talking about Huffman coding procedure which is almost an instantaneous source coding procedure

(Refer Slide Time 01:12)



for discrete sources. And this is a summary of the procedure that we discussed yesterday. I am just displaying the whole thing again for your information. Do you have any question regarding this procedure? It is understood? I think it is a very simple procedure.

(Refer Slide Time 01:28)



(Professor – student conversation starts)

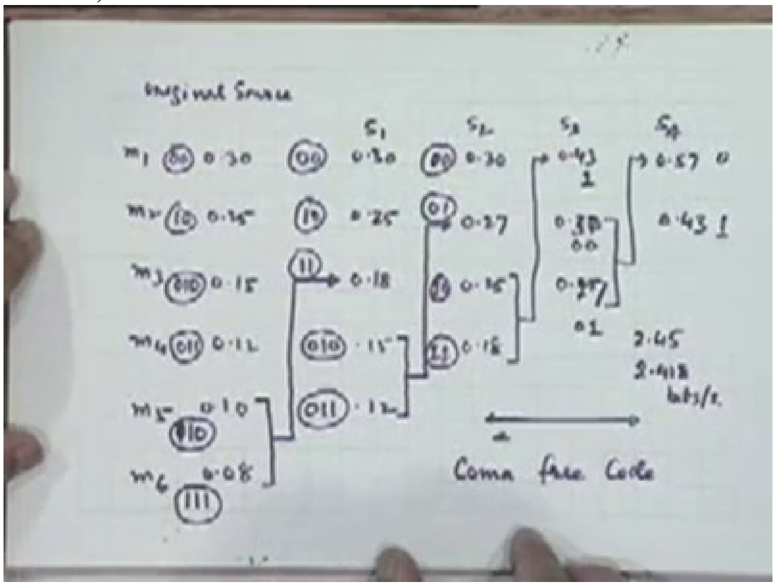
Student: Sir, one question is there. If there are only 2 messages m 1, m 2 and probabilities are not equal...

Professor: We will take care of that case in a few minutes.

(Professor – student conversation ends)

But the basic procedure is

(Refer Slide Time 01:38)

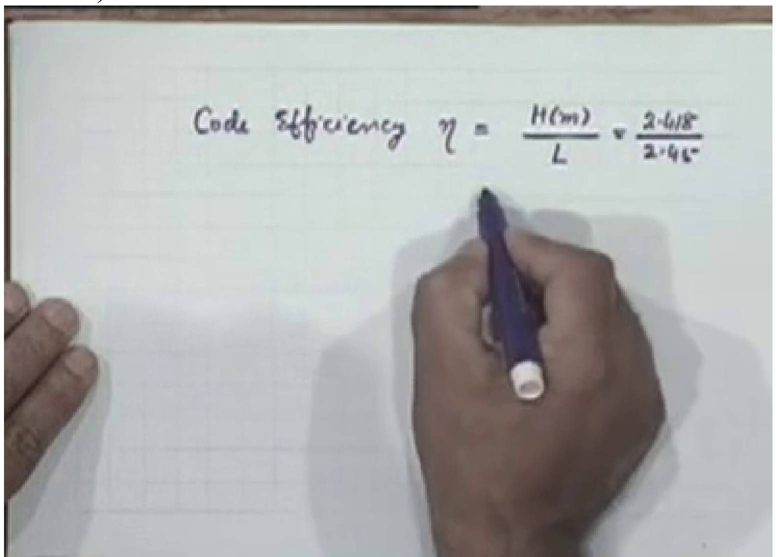


you have a dictionary of some message symbols in the source, each having some a priori probability of occurrence. You arrange these messages or symbols in the order of probability and then you keep on 0:01:57.8 of smaller sizes, smaller dictionaries by clubbing the two lowest of these message sequences, message symbols in this list. And eventually you are left with only 2 messages with very nearly equal probabilities. At that stage you start doing bit mapping and rather arbitrarily at this stage, and then keep on tracing back for distinguishing between those sources which were, those equivalent sources which were the result of clubbing two aligned sources or not two sources I should use a word messages or symbols, right?

And this procedure is quite efficient as seen from this example. As I said one could verify that the entropy of this source is 2 point 4 5 bits per symbol where as the act/actual, sorry the entropy is 2 point 4 1 8 bits per symbol whereas the average length of this code that one has finally come up with is 2 point 4 5 bits per symbol, right? Because some messages have length 2 are represented by 2 bits and some are represented by 3 bits and if you take average depending on their probabilities, using their probabilities, the average value turns out to be 2 point 4 5 bits per symbol.

And one can define a quantity which is a measure of how good your coding procedure is. And it is called, it is right, it is called code efficiency usually denoted by the symbol eta and as I said last time also it is the ratio of entropy of the source to the average length of the code that you finally got. And for this particular case this will be 2 point 4 1 8 upon 2 point 4 5

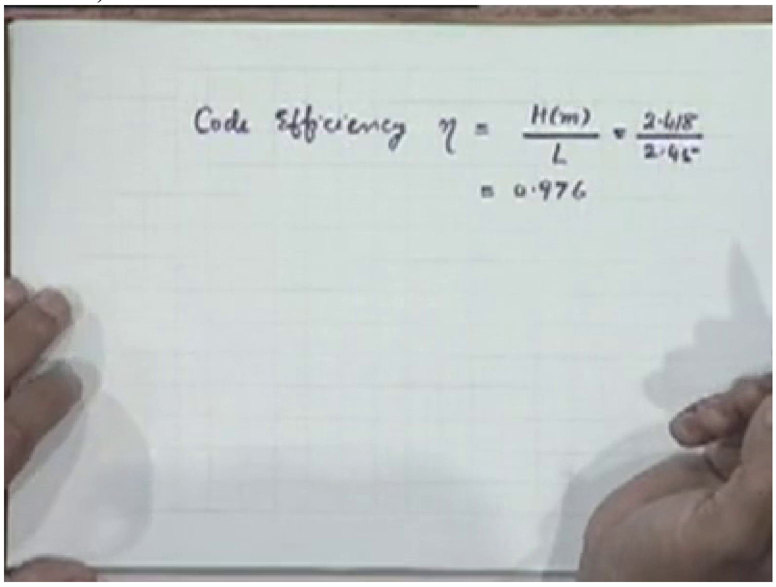
(Refer Slide Time 04:00)



Code Efficiency $\eta = \frac{H(m)}{L} = \frac{2.418}{2.45}$

which is of the order of point 9 7 or so, right, that is

(Refer Slide Time 04:06)



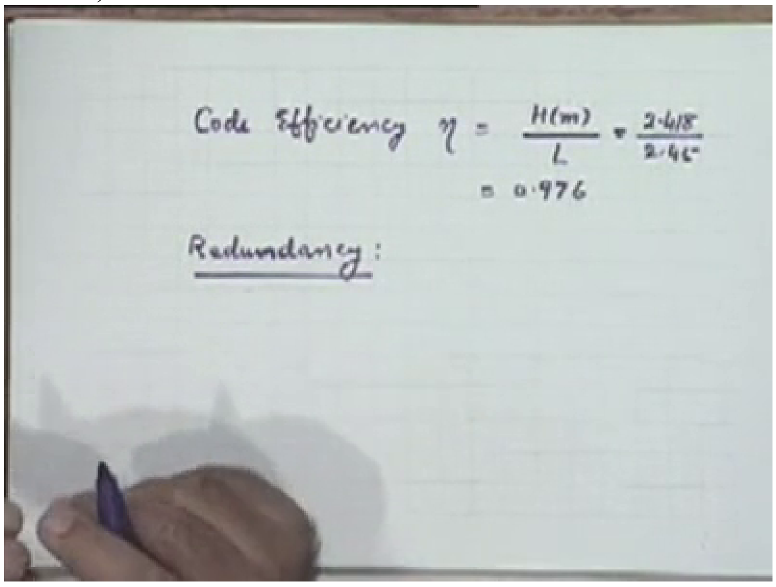
Code Efficiency $\eta = \frac{H(m)}{L} = \frac{2.418}{2.45}$

$= 0.976$

97 point 6 percent efficiency. Obviously the maximum value of efficiency is unity or 100 percent.

We also define a complementary quantity which is called the redundancy of the code, that is the

(Refer Slide Time 04:24)



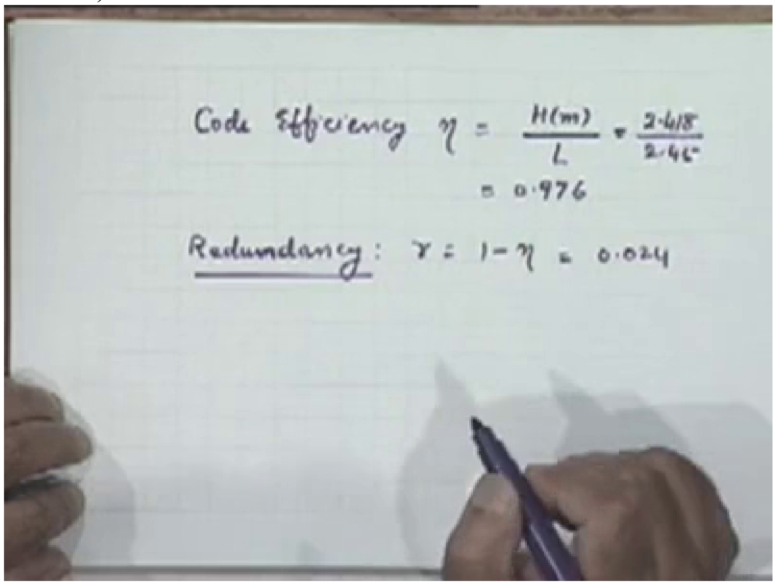
Code Efficiency $\eta = \frac{H(m)}{L} = \frac{2.418}{2.45}$

$= 0.976$

Redundancy :

transmitted signal or the transmitted message is still having some extraneous information, some extraneous value and that is basically defined as 1 minus eta

(Refer Slide Time 04:45)



Code Efficiency $\eta = \frac{H(m)}{L} = \frac{2.418}{2.45}$

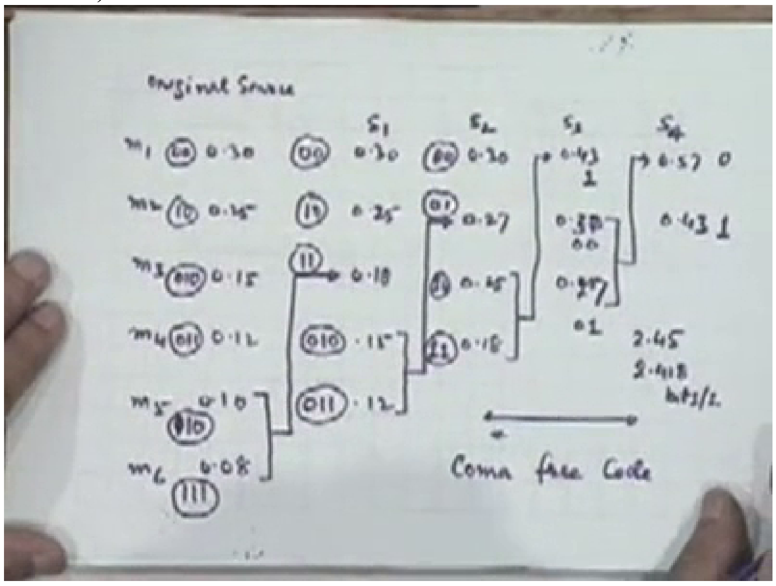
$= 0.976$

Redundancy: $\gamma = 1-\eta = 0.024$

which in this case will be point zero 2 4, Ok. So Huffman coding is a good coding procedure and there are other such coding procedures available which can do nearly as well. Well it is one of the more popular procedures because of its simplicity.

Now there will be situations like the one that was just pointed by Varun where it may not be sufficiently good

(Refer Slide Time 05:21)

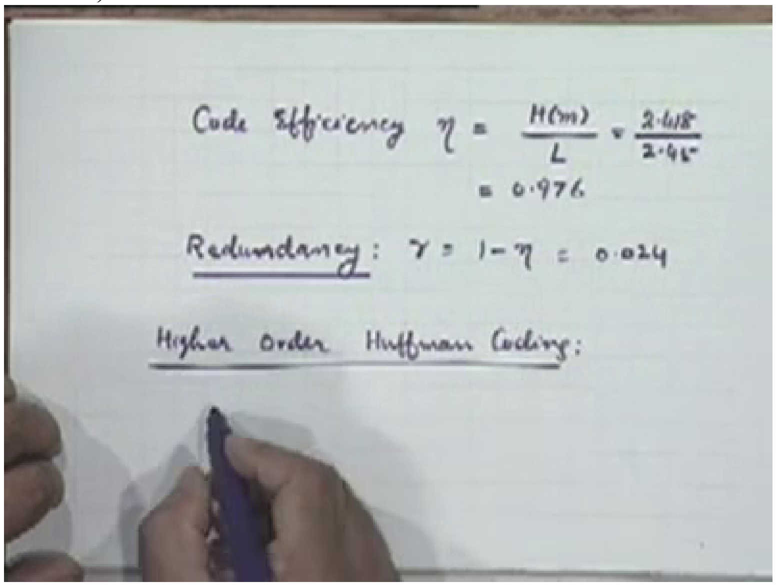


to carry out this coding on a single symbol basis. The Huffman procedure carries out this coding on a single symbol basis. That is, as each symbol is emitted you will immediately associate a corresponding code for it.

For example m 1 is emitted by the source, it will be represented by the string zero zero. If m 4 is emitted it will be represented instantaneously by the string zero 1 1.And this is unlike the source coding argument that we discussed yesterday earlier on in which one had to wait for an infinite sequence of emissions before one started the coding procedure, right?

So that was one extreme where the coding delay was infinite. This is another extreme where the coding delay is zero, no coding delay, right. Because you have a code immediately available to you, as each symbol is emitted you can just carry out that mapping and you have the encoded value. Now it is sometimes not good enough to do this kind of a thing; that is instantaneous coding. In this particular case it turned out to be good enough, right. In which case would be worthwhile to do a compromise of the two, right? One need not wait for infinitely long message emissions. One could take 1 or 2 or 3 message emissions rather than only one and then code this together, right and one could get considerable advantage by doing, introducing the small amount of coding delay and achieve a code efficiency of nearly unity, right?
Even if it was difficult to do so on a single symbol basis, to demonstrate this, incidentally this kind of Coding, Huffman coding, one can have Huffman coding procedure based on a bit of coding delay of this kind, these are called higher order Huffman coding procedures,

(Refer Slide Time 07:29)



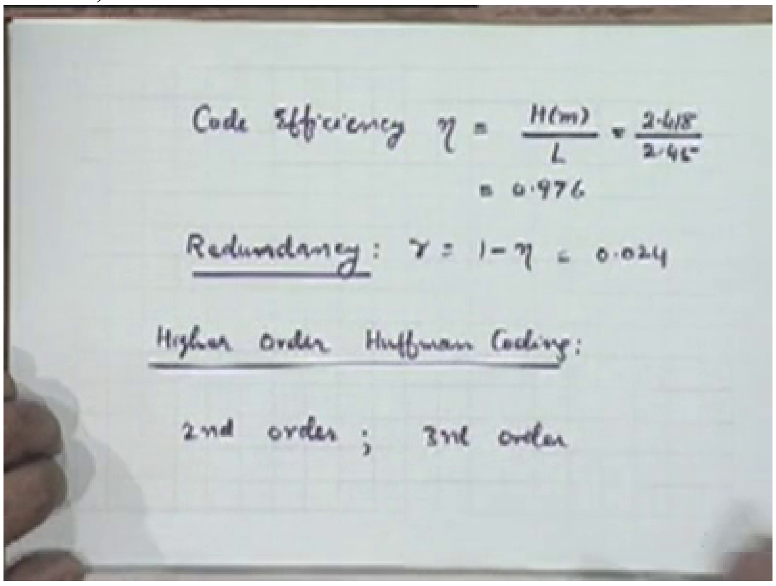
Code Efficiency $\eta = \frac{H(m)}{L} = \frac{2.418}{2.45}$
$= 0.976$

Redundancy: $\gamma = 1 - \eta = 0.024$

Higher Order Huffman Coding:

Ok. For example I could have a second order Huffman code, or a third order Huffman code. Usually one does not go

(Refer Slide Time 07:38)



Code Efficiency $\eta = \frac{H(m)}{L} = \frac{2.618}{2.45}$

$= 0.976$

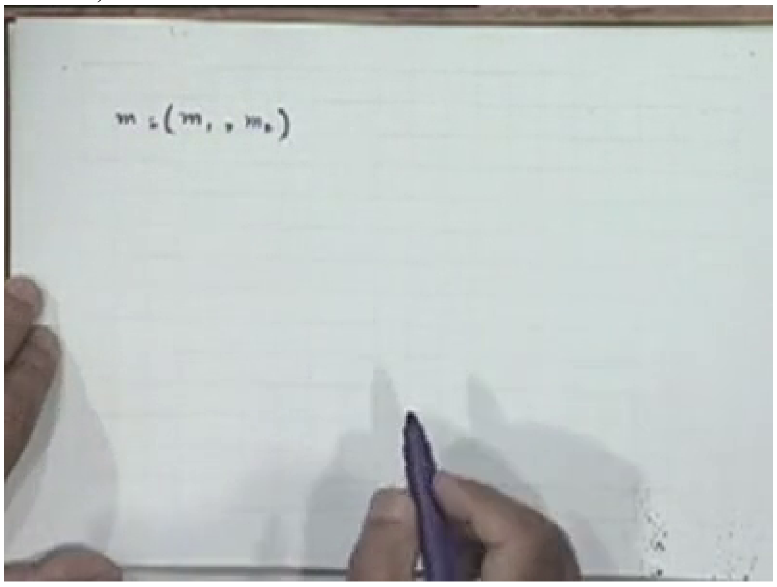
Redundancy: $\gamma = 1 - \eta = 0.024$

Higher Order Huffman Coding:

2nd order ; 3rd order

beyond the few values like this, 1, 2 or 3. One does not in any case go up to very large values like one has to do for the ideal case that we discussed, Ok. Let me illustrate what advantage we can get out of going for higher order Huffman coding by taking a very simple example where our message source is comprised of a very

(Refer Slide Time 08:05)

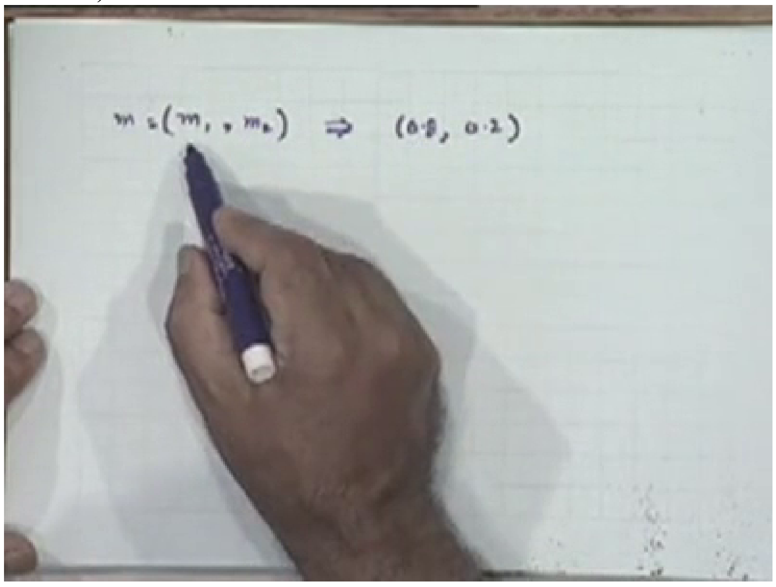


simple kind with only 2 outputs, m 1 and m 2, right.

And if they are equi-probable then we do not have to go through Huffman coding, right? For example, if each of them m 1 as well as m 2 is emitted by equal probability of point 5 each, the obvious thing to do is to allocate 0 and 1 arbitrarily to them and that is the best one can do. Because it is 1, the entropy of the source is 1 bit per symbol and that also will represent, require 1 bit and therefore you have code efficiency of unity. One does not have to do anything beyond that. Is it clear?

On the other hand let us take the situation where the probabilities associated with the two symbols are very unbalanced. Let us say point 8 and point 2 respectively. So point 8 is the probability of emission

(Refer Slide Time 08:57)

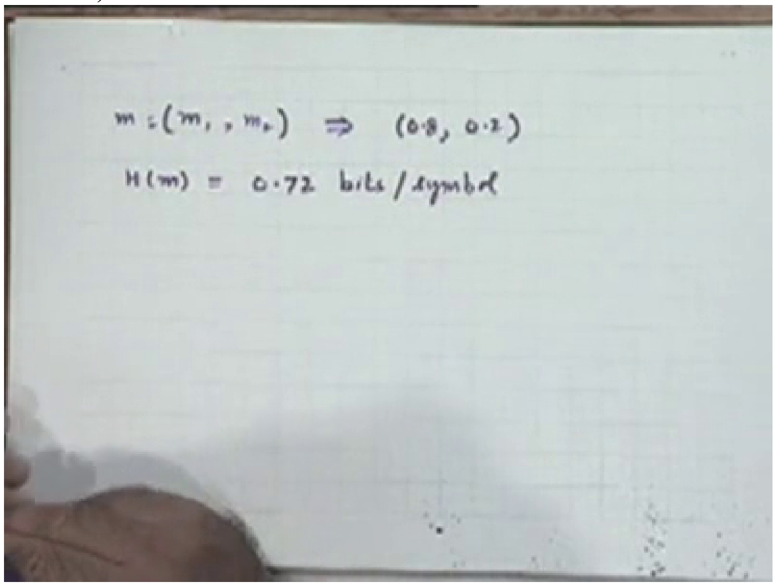


$m : (m_1, m_2) \Rightarrow (0.8, 0.2)$

of m sub 1 and the point 2 is the corresponding probability for m sub 2. Ok and it is very easy to check that the entropy now is not 1 bit per symbol as it would have been if both had equal probability of point 5 each, right? But it is obviously less than 1 because equal probabilities case give you the largest, maximum entropy. It turns out to be point 7 2 bits per symbol. You just carry out the computation and verify that.

And now if you do a

(Refer Slide Time 09:36)

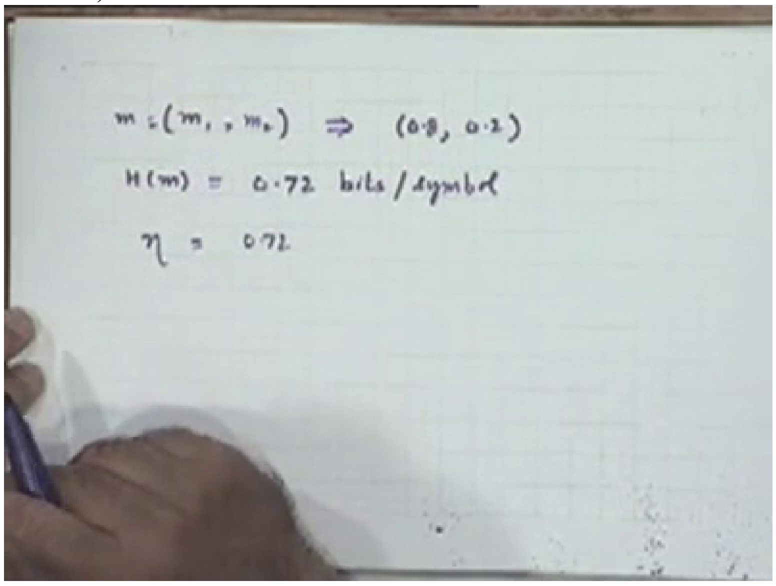


$m : (m_1, m_2) \Rightarrow (0.8, 0.2)$

$H(m) = 0.72$ bits / symbol

straight forward allocation of zero and 1 to m 1 and m 2 what is your code efficiency? Point 7 2, right?

(Refer Slide Time 09:45)



$m: (m_1, m_2) \Rightarrow (0.8, 0.2)$

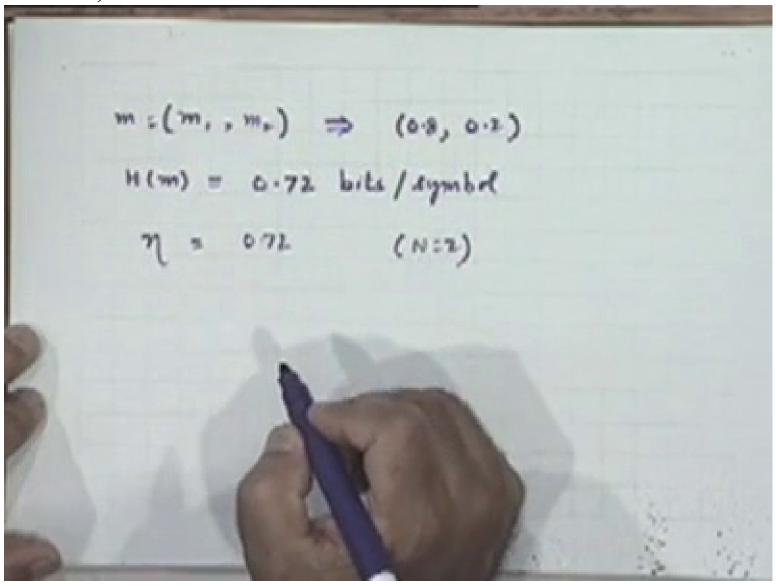
$H(m) = 0.72$ bits / symbol

$\eta = 0.72$

Because your code length is 1 bit but the average information being conveyed by this code, per transmission is only point 7 2 bits because that is the entropy of the source, alright. So efficiency is only point 7 2 or 72 percent. Now this is where it would be worthwhile investigating or going for a higher order Huffman coding procedure, right? And the higher order Huffman coding procedure now lets a few bits, few message outcomes accumulate before you think of coding.

For example, let us consider the second order case, alright? Now that is, we will take N equal to 2.

(Refer Slide Time 10:29)



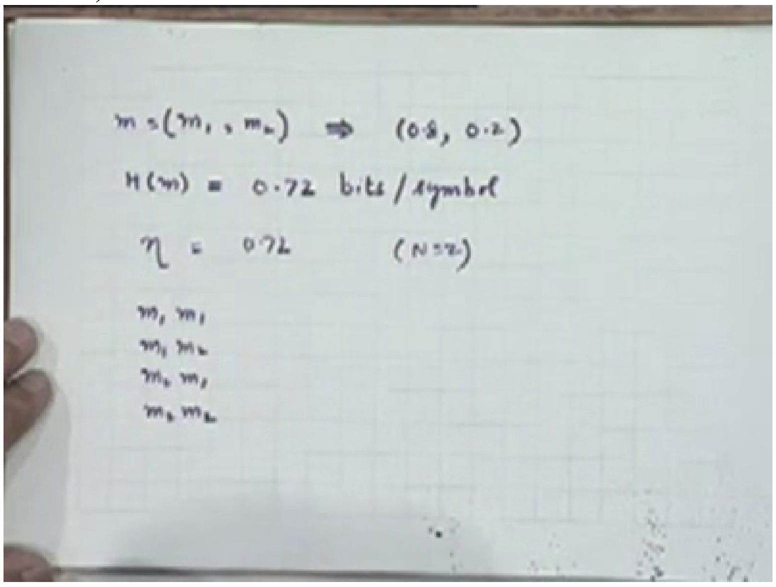
$m: (m_1, m_2) \Rightarrow (0.8, 0.2)$

$H(m) = 0.72$ bits / symbol

$\eta = 0.72 \qquad (N=2)$

We now think of a new message source whose outcomes are successive, two successive outcomes of this source, right? And obviously there are 4 possible outcomes with different probabilities. These are m 1, m 1; m 1, m 2; m 2, m 1; m 2 m 2, right?

(Refer Slide Time 11:03)



$m = (m_1, m_2) \Rightarrow (0.8, 0.2)$

$H(m) = 0.72$ bits / symbol

$\eta = 0.72 \qquad (N=2)$

$m_1 m_1$

$m_1 m_2$

$m_2 m_1$

$m_2 m_2$

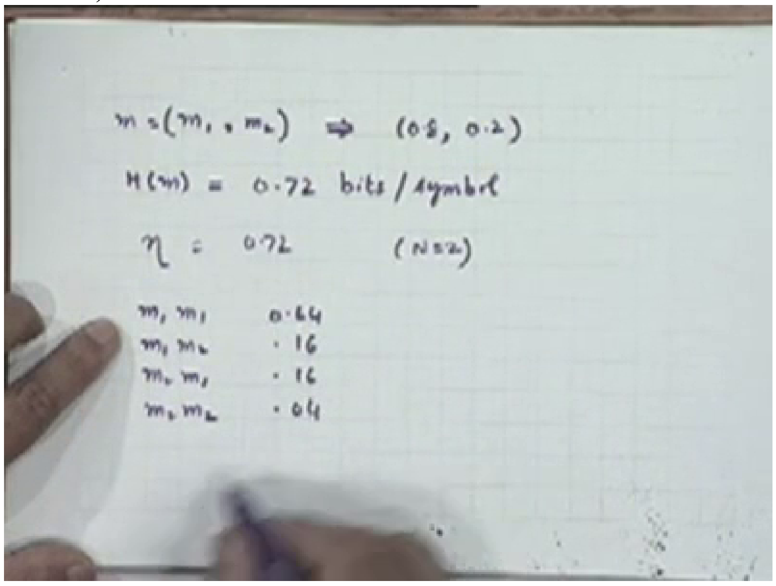
First could be m 1, second could be m 1, like that and so on. And if we assume that successive emissions are independent of each other then we can compute the probabilities of each of these 4 message values that we have now, 4 message symbols you can think of these as a new source with 4 symbols as its output, as its dictionary, Ok and its corresponding probabilities are, what will be the probability of this sequence? Point 6 4

(Professor – student conversation starts)

Student: point 1 6

Professor: point 1 6, point 1 6 and point zero 4. Right

(Refer Slide Time 11:43)



$m = (m_1, m_2) \Rightarrow (0.8, 0.2)$

$H(m) = 0.72$ bits / symbol

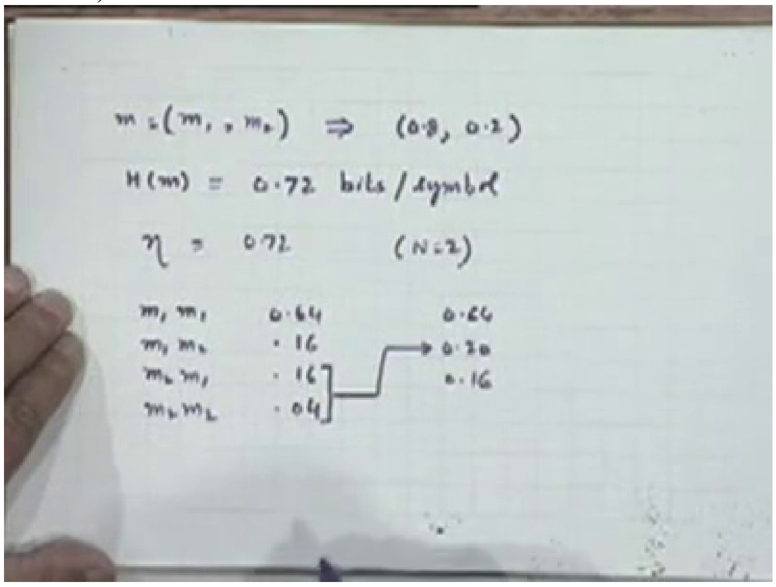
$\eta = 0.72 \quad (N = 2)$

$m_1 m_1$	0.64
$m_1 m_2$	.16
$m_2 m_1$	.16
$m_2 m_2$	.04

and of course I have written that out again in the order of their decreasing probability and one goes through the rest of the procedure in the same way. Shall I do it?
(Professor – student conversation ends)

Alright, although I do not think it is necessary but since it is a small one, you can do that. The next one will be we will club these two to get point 2 0 and this is point 1 6, this one comes here;

(Refer Slide Time 12:14)



$m = (m_1, m_2) \Rightarrow (0.8, 0.2)$

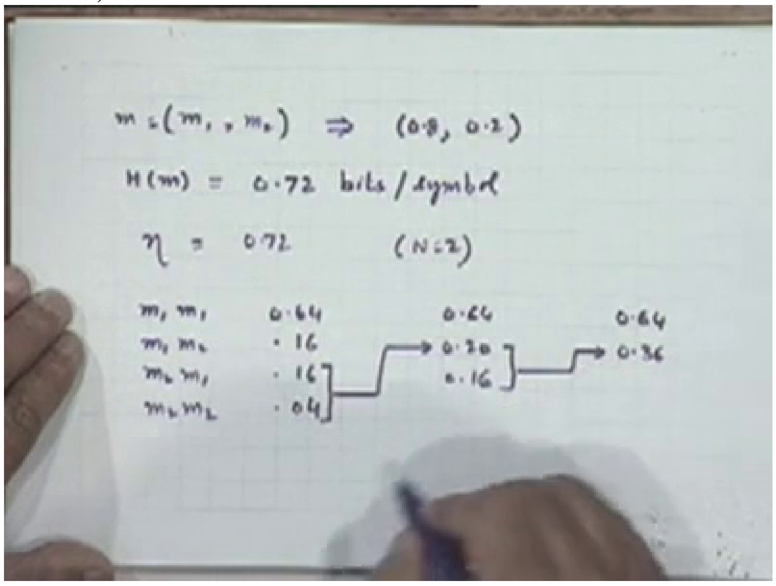
$H(m) = 0.72$ bits / symbol

$\eta = 0.72 \quad (N = 2)$

$m_1 m_1$	0.64	0.64
$m_1 m_2$	.16	0.20
$m_2 m_1$	.16	0.16
$m_2 m_2$	.04	

these two are clubbed together again. Now you could get,

(Refer Slide Time 12:27)



$m = (m_1, m_2) \Rightarrow (0.8, 0.2)$

$H(m) = 0.72$ bits/symbol

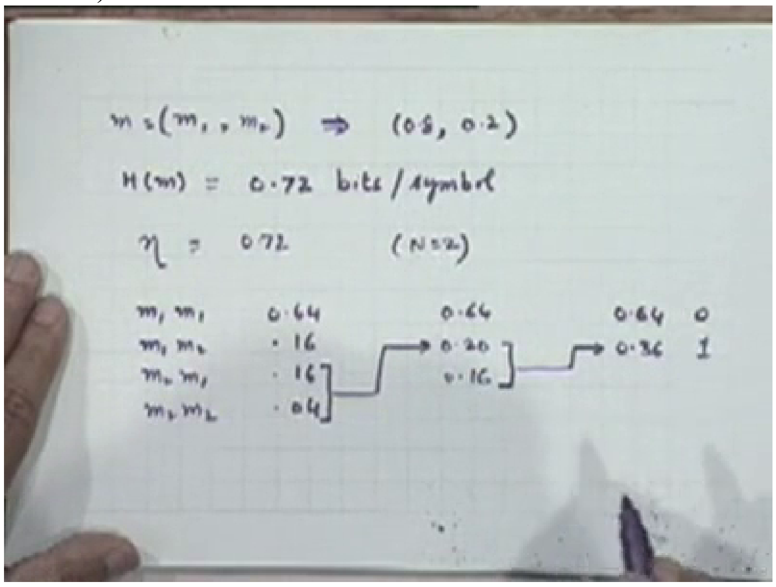
$\eta = 0.72 \quad (N=2)$

$m_1 m_1$	0.64	0.64	0.64
$m_1 m_2$	.16	0.20	0.36
$m_2 m_1$	.16	.16	
$m_2 m_2$	.04		

alright. Now we have still not got them really equal or very close to equal but they are better than what we started with here, point 8 and point 2, right the discrepancy is less here. If you go for the third order situation you will find that the eventual discrepancy is even lower, right? That is the advantage of waiting for some time in some cases.

So one can start with bit allocation of some kind. For example I could do a zero here and a 1 here,

(Refer Slide Time 13:06)



$m = (m_1, m_2) \Rightarrow (0.8, 0.2)$

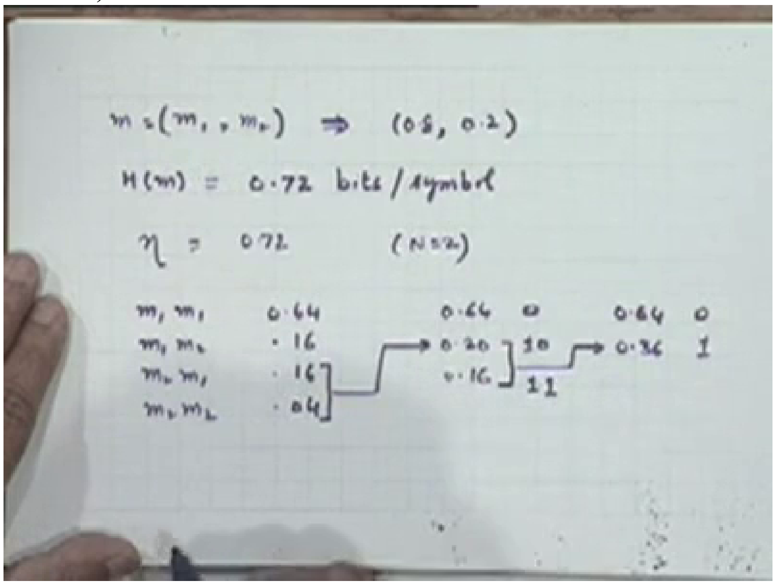
$H(m) = 0.72$ bits/symbol

$\eta = 0.72 \quad (N=2)$

$m_1 m_1$	0.64	0.64	0.64	0
$m_1 m_2$	.16	0.20	0.36	1
$m_2 m_1$	.16	.16		
$m_2 m_2$	.04			

and at this stage this will become zero. This will become 1 zero and 1 1,

(Refer Slide Time 13:13)



$m = (m_1, m_2) \Rightarrow (0.8, 0.2)$

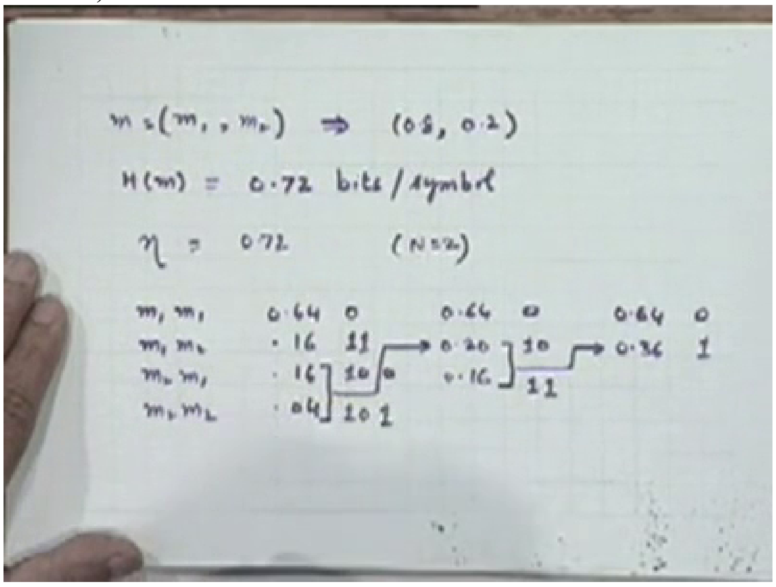
$H(m) = 0.72$ bits/symbol

$\eta = 0.72 \quad (N=2)$

$m_1 m_1$	0.64	0.64	0	0.64	0
$m_1 m_2$	.16	0.20	10	0.36	1
$m_2 m_1$	.16	0.16	11		
$m_2 m_2$	.04				

this will be zero and this point 1 6 will be 1 1, right, this point 1 6 will be 1 zero zero right? And this will become 1 zero 1,

(Refer Slide Time 13:30)



$m = (m_1, m_2) \Rightarrow (0.8, 0.2)$

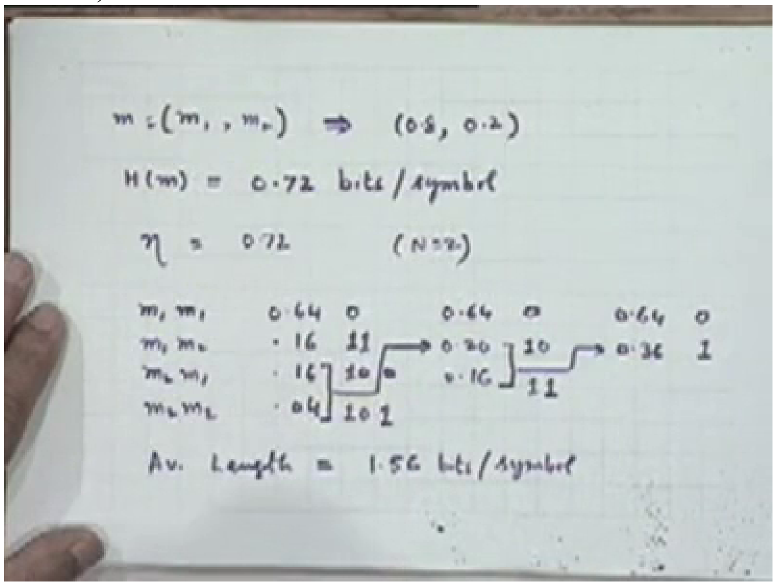
$H(m) = 0.72$ bits/symbol

$\eta = 0.72 \quad (N=2)$

$m_1 m_1$	0.64	0	0.64	0	0.64	0
$m_1 m_2$	.16	11	0.20	10	0.36	1
$m_2 m_1$	.16	100	0.16	11		
$m_2 m_2$	.04	101				

so this will be the final code. m 1, m 1 will be represented by zero, this by 1 1 and so on.Ok So you will wait for 2 successive messages and then carry out the coding. So you won't be now instantaneous coding, you will be coding with unit delay, right? And one can check that the average length of this code, if you were to go just by these probabilities, point 6 4 into 1, plus point 1 6 into 2 plus point 1 6 into 3 plus point zero 4 into 3, that will turn out to be 1 point 5 6 bits per symbol of this source.

(Refer Slide Time 14:23)



$m = (m_1, m_2) \Rightarrow (0.8, 0.2)$

$H(m) = 0.72$ bits/symbol

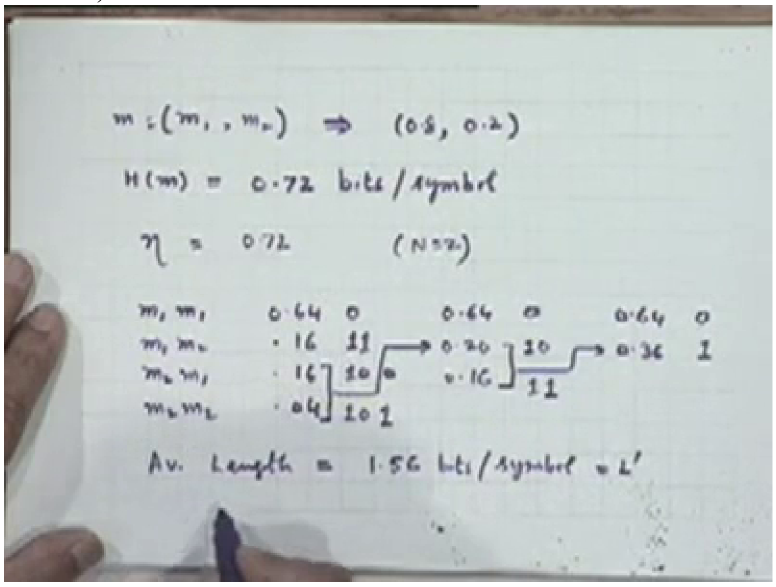
$\eta = 0.72 \quad (N=2)$

$m_1 m_1$	0.64	0	0.64	0	0.64	0
$m_1 m_2$	.16	11	0.20	10	0.36	1
$m_2 m_1$	.16	100	.16	11		
$m_2 m_2$	.04	101				

Av. Length = 1.56 bits/symbol

But this source is conveying how many symbols actually? Of the original source? 2 symbols, right? So of the original, let me cal this L prime.

(Refer Slide Time 14:34)



$m = (m_1, m_2) \Rightarrow (0.8, 0.2)$

$H(m) = 0.72$ bits/symbol

$\eta = 0.72 \quad (N=2)$

$m_1 m_1$	0.64	0	0.64	0	0.64	0
$m_1 m_2$	.16	11	0.20	10	0.36	1
$m_2 m_1$	.16	100	.16	11		
$m_2 m_2$	.04	101				

Av. Length = 1.56 bits/symbol = L'

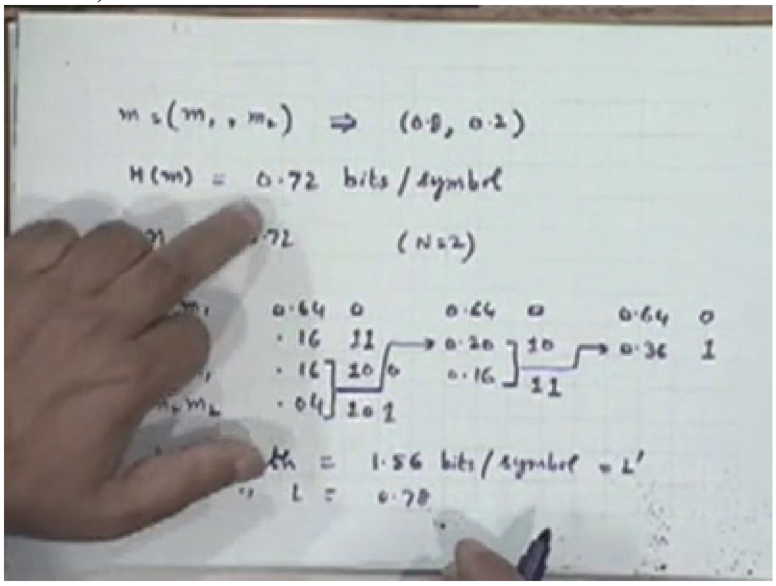
So the average length L with respect to the original source is, point 7 8. And now what is your code efficiency?

(Professor – student conversation starts)

Student: point ninety five.

Professor: It is point 7 2

(Refer Slide Time 14:49)


m = (m₁ , m₂) ⇒ (0.8, 0.2)
H(m) = 0.72 bits / symbol
(N=2)
0.64 0
.16 11
.16 100
.04 101
0.64 0
0.20 10
0.16 11
0.64 0
0.36 1
= 1.56 bits / symbol = L'
L = 0.78

upon point 7 8, right which turns out to be about point 9 or so. So is it true? The entropy was point 7 2 and the average length is very close to that now, with point 7 8, right? So our efficiency has gone above 90 percent.

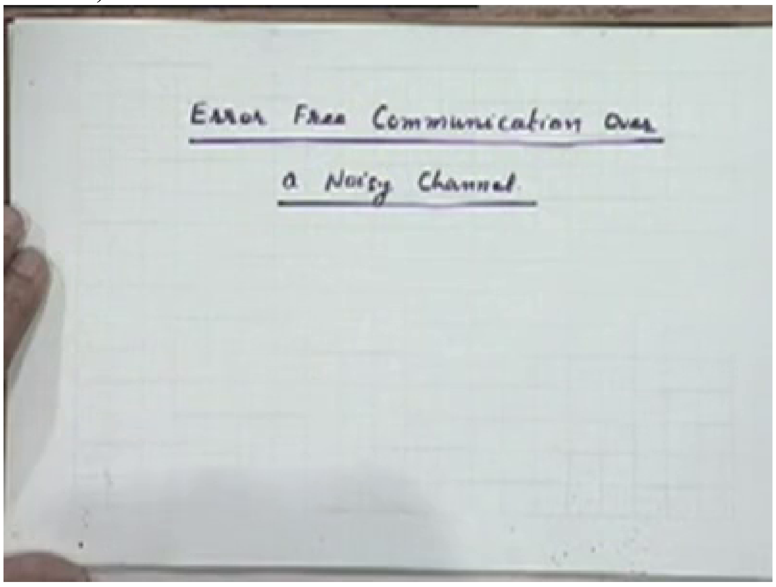
(Professor – student conversation ends)

One can check if you take n equal to 3, you will get 98 to 99 percent efficiency which is very good. So sometimes it is worthwhile introducing a bit of coding delay to get better efficiency, representation efficiency. Remove as much as redundancy as possible in the source emissions. Is it Ok? Please speak out if you have any doubts left? Ok. No question. Nikhil is it Ok, fine. So that is as far as what I wanted to cover as source codings. This is a very brief glimpse into what source coding is all about. It is very, fairly vast subject on its own. And there are various kinds of sources and various kinds of source coding procedures to take care of the sources.

But in the short time that we have available with us, I only wanted to give you an idea of what source coding is all about and what we can achieve by doing source coding, Ok. But the basic purpose you should not forget. The basic purpose is to come up with an efficient digital representation of your source outputs, Ok, so that we do not burden the channel unnecessarily by transmitting redundant information. Basically that is the idea of source coding, right? Because channel time is important and expensive because it is composed of a lot of infrastructure and lot of equipment which costs money. So you have to efficiently use it.

To efficiently use it, we must make sure that we avoid sending redundant information unless we are doing it deliberately for some other reason, right. But per se we would not like redundancy to be transmitted because it means waste of channel capacity. Ok Now with this background about source coding I would now like to briefly motivate you about Shannon's result regarding error-free communication, the possibility of error-free communication over a noisy channel. This is what we talked about when we started discussing information theory. You recollect that discussion?

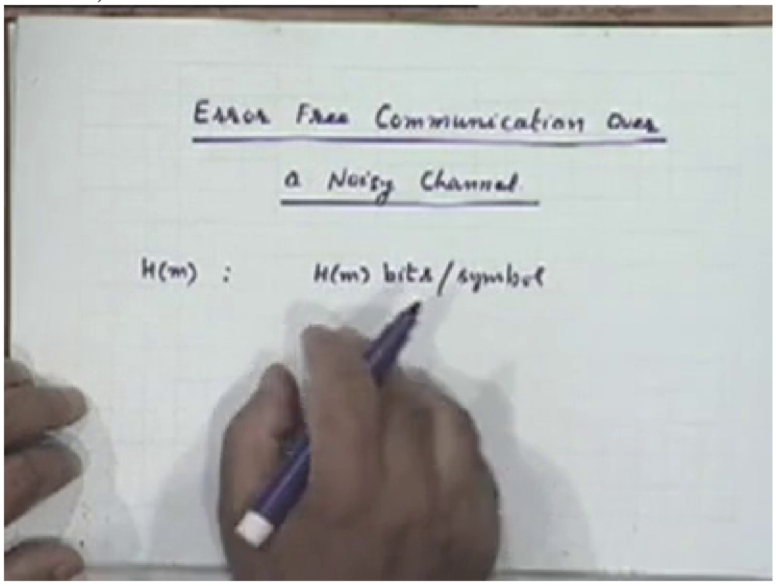
(Refer Slide Time 18:00)



Error Free Communication Over a Noisy Channel

See what we have learnt till now in information theory is that if you have a source with entropy H m we can transmit by appropriate source coding procedures, we can encode it into a digital representation which require on an average H m bits per message symbol, right? One can carry out an encoding

(Refer Slide Time 18:35)

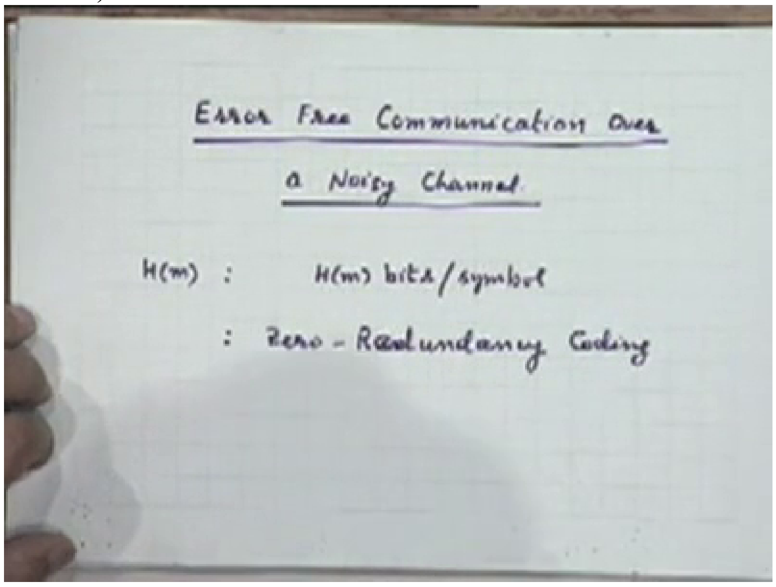


Error Free Communication Over a Noisy Channel

H(m) : H(m) bits / symbol

procedure and if encoding procedure is good, we can hope to achieve this kind of representation of the message and we could of course simply transmit it on the channel. If you do that, we call this zero redundancy coding.

(Refer Slide Time 19:02)



Error Free Communication Over a Noisy Channel

H(m) : H(m) bits / symbol

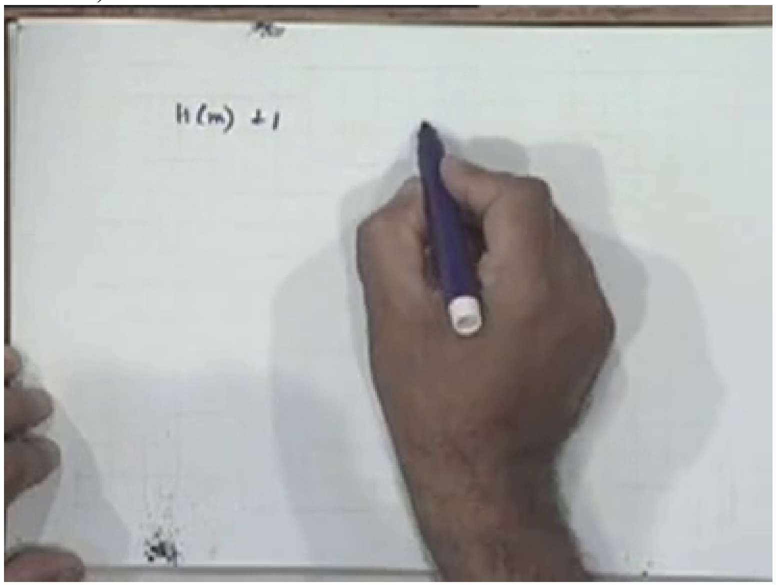
: Zero - Redundancy Coding

Now this can be very risky also. While it is good from the representation point of view, from not burdening the channel unnecessarily with redundant information point of view but it can be very risky thing to do in real life. Why? Because real life channel is going to be noisy. Now anything that you transmit now, something that goes wrong because of noise, there is no way to know whether what you have received is what was transmitted or the result of distortion or degradation due to noise. Because there was absolutely no redundancy, right? You expect whatever you receive to be hard information, right? Whether it is 1s or zeroes, you will expect its face value because it was supposed to be the true representation of the source, right.

Whether it was, whether it is really the true value of the source that you are receiving or whether it is the result of that having undergone some changes because of an error probability in your receiver, right there is no means to know, right. So there is therefore this problem that if you carry out this zero redundancy coding, the noisy channel is going to create problems for you. That means it is not possible in such a situation to have error-free communication, if you stick to zero redundancy coding, and it follows therefore, it is more or less obvious that use of or introduction of certain amount of redundancy is going to help us combat these noise problems.

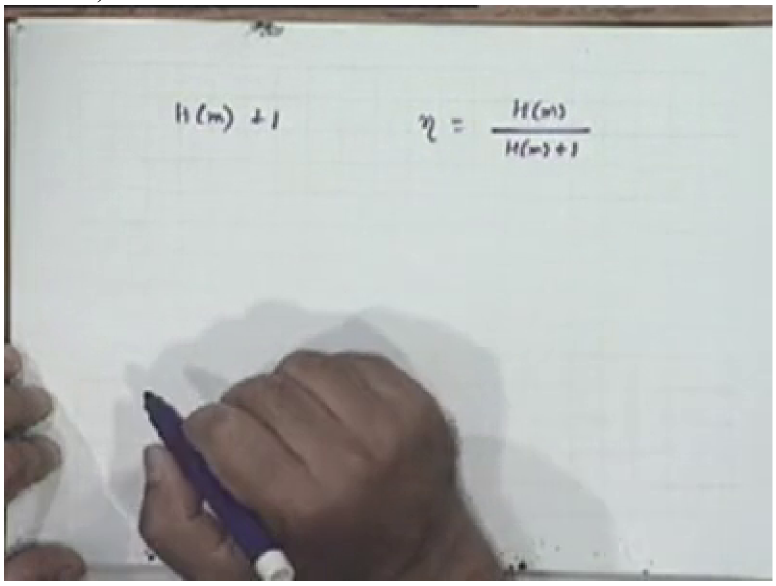
For example you are all familiar with the use of parity check codes, right, in various kinds of systems including communication systems, Ok. Suppose you are using that kind of parity check coding and suppose you are doing this parity check coding on a source with entropy H m, typically what you are doing is to every bit, to every word of the source output you are adding one extra bit, right and your code

(Refer Slide Time 21:32)



efficiency therefore becomes, alright?

(Refer Slide Time 21:40)



$H(m) + 1$

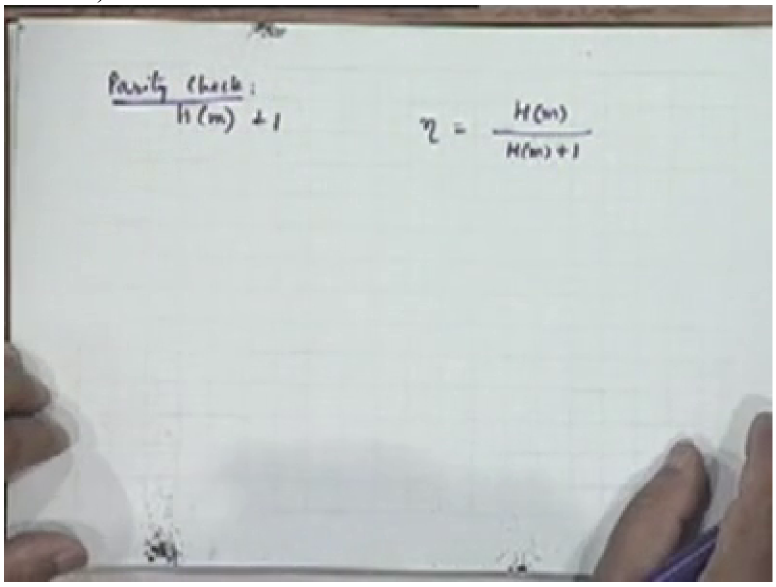
$$\eta = \frac{H(m)}{H(m)+1}$$

(Professor – student conversation starts)

Student: This is for what?

Professor: This is for parity check coding. You just add one parity check bit, right? That is what the parity check coding is all about.

(Refer Slide Time 21:50)



Parity check:

$H(m) + 1$

$$\eta = \frac{H(m)}{H(m)+1}$$

Add 1 additional bit to every word, to every message word that you have, Ok. If average, if length of the message is H m bits, you are adding one extra bit making it H m plus 1 bits right, that one bit is not really the original information. It has been deliberately added by you as a redundant bit. It is not really conveying any new information but we know that at the cost of going from 100 percent efficiency to less than 100 percent efficiency, at the, the advantage that you will get is the enability to be able to say when errors occur. You would be able to at least detect that errors have occurred.

Student: Some kind of error

Professor: Some kinds of errors have occurred, right, which ability was totally missing in the original situation, right? Is it clear?

(Professor – student conversation ends)

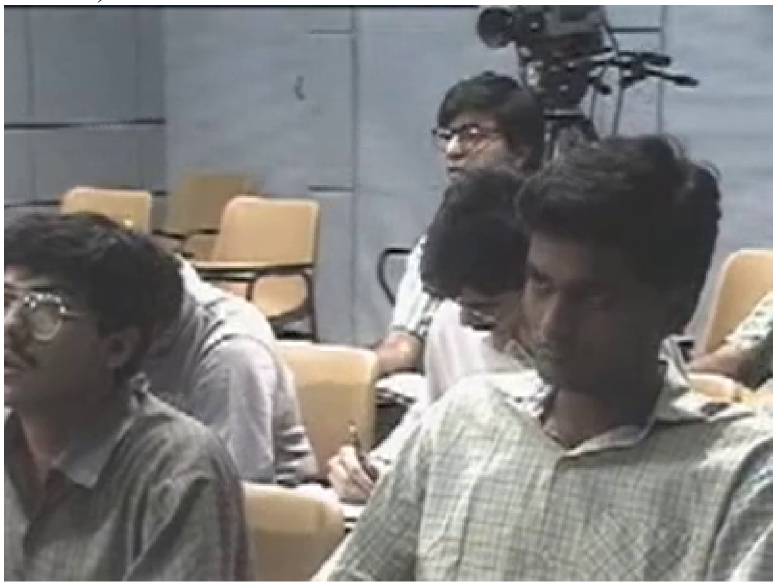
So as eta decreases or redundancy, what will be redundancy in this case, 1 minus this which will become 1 by H m plus 1? So from zero redundancy we have got a finite value of redundancy. As gamma increases we get some means to combat noise. We already know about parity check code so therefore I took this as an example. In general, the moral is that one can increase the immunity against channel noise by introducing certain amount of redundancy in your transmission, right?

Immunity against channel noise can be increased via introduction of redundancy, Ok. And it is this which brings in the possibility of achieving error-free communication over a noisy channel. Let us discuss how.

(Professor – student conversation starts)

Student: Sir, the noise would be, in the message signal the bits are flipped 0:24:07.4. Instead

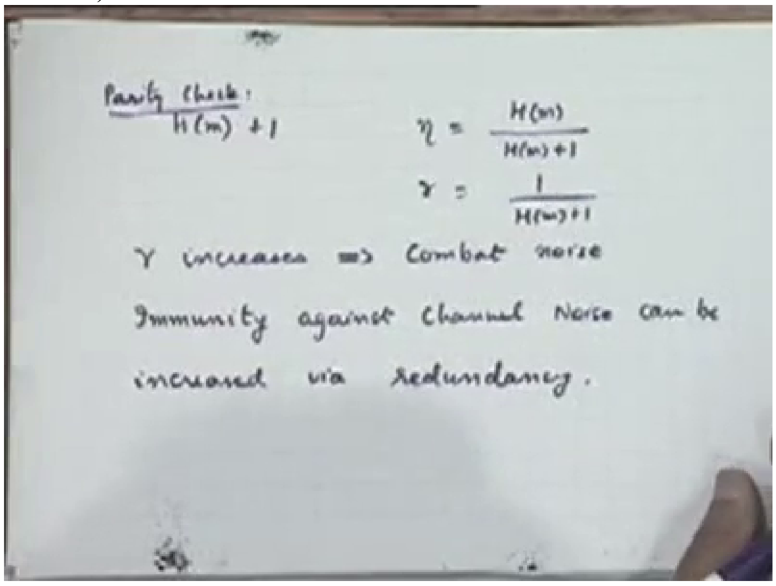
(Refer Slide Time 24:09)



of the message signals if we suppose the parity bit is flipped, message signal is transmitted Ok.

Professor: I have not yet, I have not yet, I am not discussing the mechanism of this coding and decoding, right? Actually, just to answer your question very briefly an appropriate code will be able to detect where the error has occurred independent of whether it is in the message bits or in the parity bits, Ok. But that we are not going to dwell at the moment, Ok. A good well-designed code will be able to make, we will be able to find out where errors have occurred and what those values are, right provided it is a proper error correcting code. But if it is only parity check code there is only so much you can do with it. You cannot do very much. You can only detect errors in fact. You cannot even find out where the error has occurred. You can only detect the occurrence of the error, right? Nothing more than that.

(Refer Slide Time 25:00)



Parity Check:
$H(m) + 1$

$$\eta = \frac{H(m)}{H(m)+1}$$

$$\gamma = \frac{1}{H(m)+1}$$

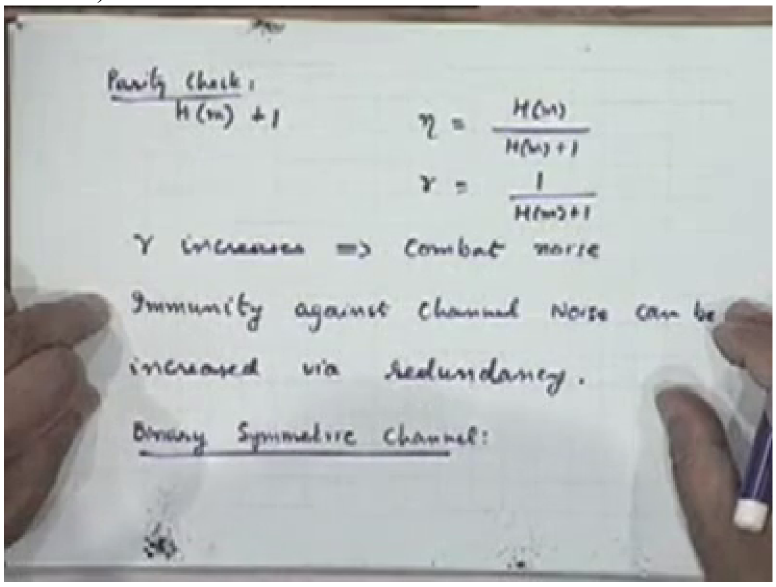
γ increases $\Rightarrow$ Combat noise

Immunity against Channel Noise can be increased via redundancy.

(Professor – student conversation ends)

But anyway we are not discussing that at the moment. We are only discussing what we can do, what we can eventually hope to achieve by doing this kind of coding procedure, right? And I want to, actually speaking; introduce the concept of channel capacity to you, which is a fairly slightly abstract concept which I want you to understand. Let us take the example of what we sometimes call a binary symmetric channel. Before going to that, please remember what is it that I have said which I want to further discuss. What I have said is that I can get

(Refer Slide Time 25:54)



Parity Check:

H(m) + 1

$\eta = \frac{H(m)}{H(m)+1}$

$\gamma = \frac{1}{H(m)+1}$

γ increases => Combat noise

Immunity against Channel Noise can be increased via Redundancy.

Binary Symmetric Channel:

some kind of immunity against channel noise by introducing redundancy.

The question is how much redundancy one has to introduce, Ok. That question is I am going to illustrate slightly by taking an example of what is called a binary symmetric channel, Ok. A channel, a physical channel that we have been discussing in the first course in communication as well as in this course, is one which is comprised of the transmitter, the physical medium and the receiver.

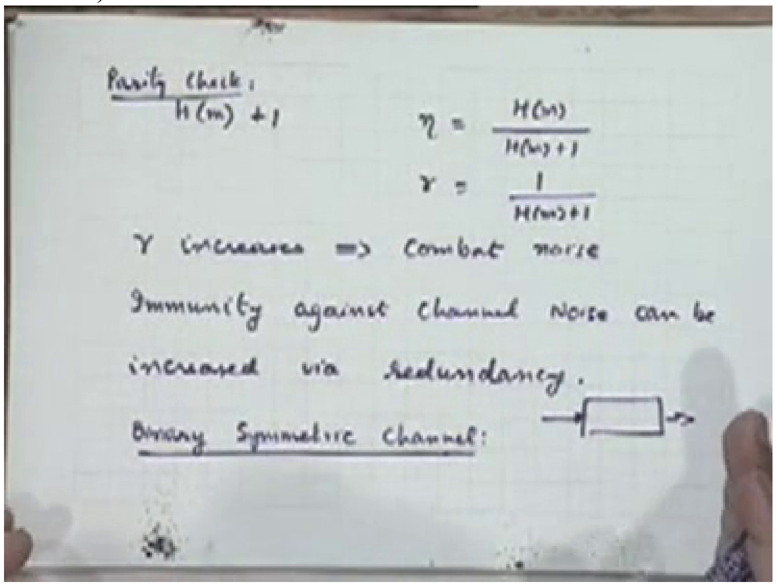
(

The thing is disturbing. Can you shut the door? Is the door open? Then it is fine

)

Let us return to this binary symmetric channel. This is a model that is used for representing physical channels in a binary situation, in a binary signal transmission situation, Ok where, see you may have a specific digital modulation scheme, you may have some physical channel with a certain kind of noise, let us say Gaussian noise and then you finally have at the receiver the matched filter followed by sampler followed by your decision device. But if you look at this, if you regard this whole system as a black box, what is the input to this? It is a sequence of

(Refer Slide Time 27:52)



Parity Check:

$H(m) + 1$

$$\eta = \frac{H(m)}{H(m)+1}$$

$$r = \frac{1}{H(m)+1}$$

r increases $\Rightarrow$ Combat noise

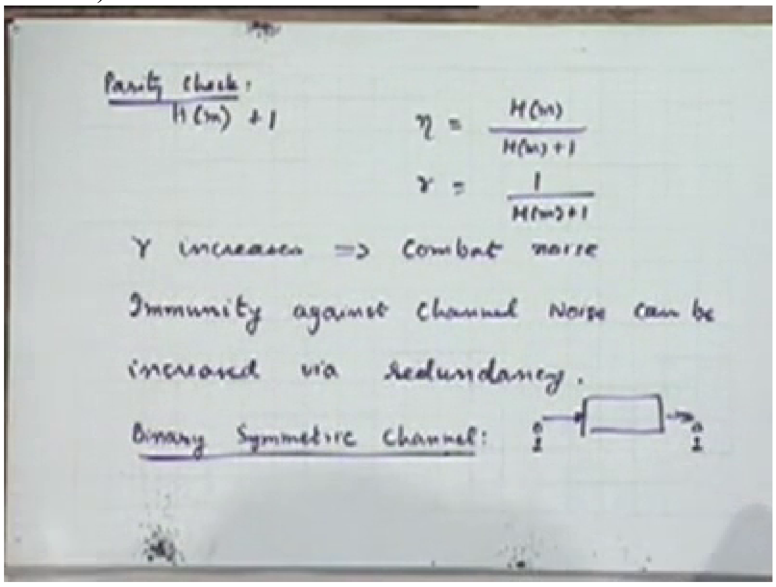
Immunity against Channel Noise can be increased via Redundancy.

Binary Symmetric Channel:

zeroes and 1s, right? It is a digital system, digital communication system, the input is let us say the sequence of zeroes and 1s if you are assuming binary sources as the input, right?

Similarly the output, the final output, the very final output is again the sequence of zeroes and 1s, right? So every zero or 1 that is transmitted here is received as, hopefully

(Refer Slide Time 28:18)



Parity Check:

$H(m) + 1$

$$\eta = \frac{H(m)}{H(m)+1}$$

$$r = \frac{1}{H(m)+1}$$

r increases $\Rightarrow$ Combat noise

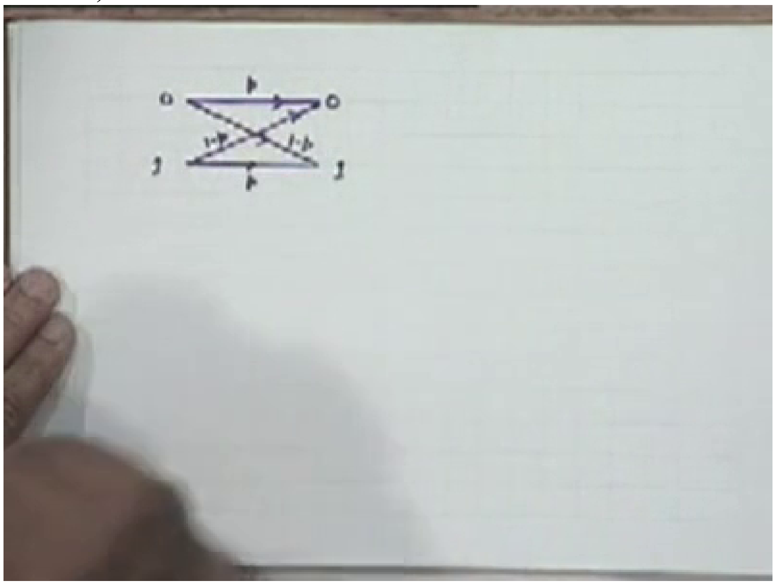
Immunity against Channel Noise can be increased via Redundancy.

Binary Symmetric Channel:

corresponding zero and 1 here except when errors occur in which case a zero may be flipped as 1 and 1 may be flipped as a zero, right. So when you model the channel or the communication system as a black box in this sense, looking at only the absolute inputs the very first input that you are giving in terms of binary values and the final outputs in terms of binary values, that resulting model is described by various kinds of models and one of the models is what is called as binary symmetric model for this conjecture. 0:28:54.8

And what it says is Ok, in the binary symmetric channel the model is, a zero will be received as a zero, let me write it separately, usually it is represented like this. A zero will be received as a zero with a certain probability p, and will be received as a 1 with the probability 1 minus p. And if it is a symmetric model, the same will be true for1. If 1 will be received as a 1 with a probability p and as a zero with probability

(Refer Slide Time 29:34)

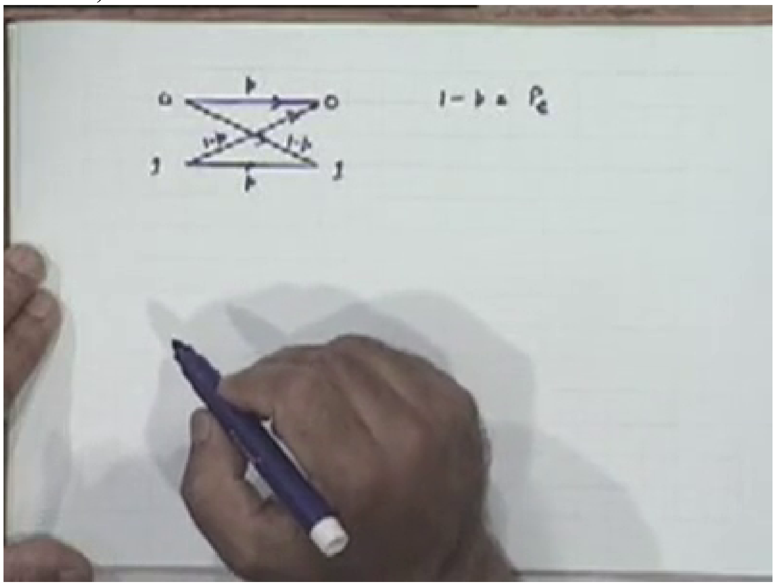


1 minus p. Obviously 1 minus p is then equivalent to the error probability calculation that we have done so far, right, for various systems. So the value of p and 1 minus p will depend on what specific system configuration we have, how much noise we have in the system, how much bandwidth we are using and all those things.

The exact design of the system will dictate the value of p but a single parameter finally is sufficient to describe the overall performance of the channel, the overall performance of the system in terms of inputs and outputs. And that parameter is p. Of course calculation of this parameter further requires you to go deeper into the communication system and understand how to calculate it like we have done for various situations, Ok.

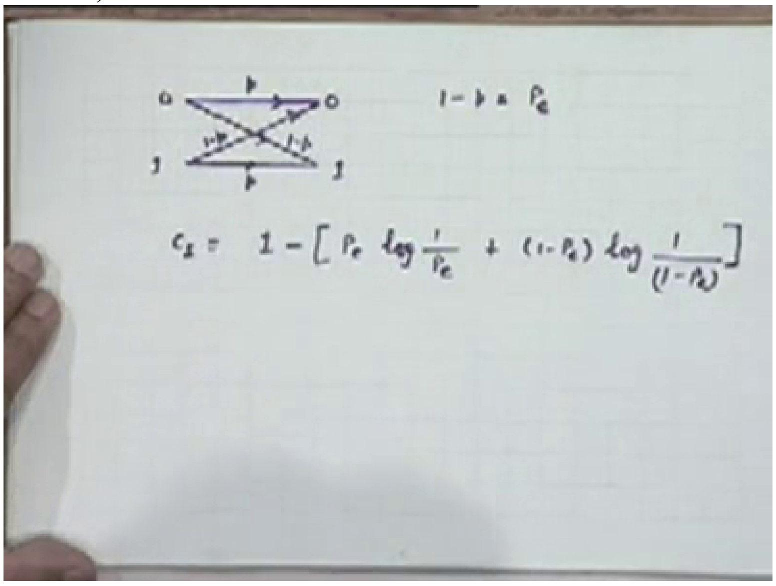
Now for this kind of a channel which is described by this parameter p or describe by parameter p sub e which is error probability associated with it,

(Refer Slide Time 30:37)



right? It can be shown that we have a channel capacity C sub s which is given by this formula, sorry log,

(Refer Slide Time 31:07)



$$C_s = 1 - \left[P_e \log \frac{1}{P_e} + (1 - P_e) \log \frac{1}{(1 - P_e)} \right]$$

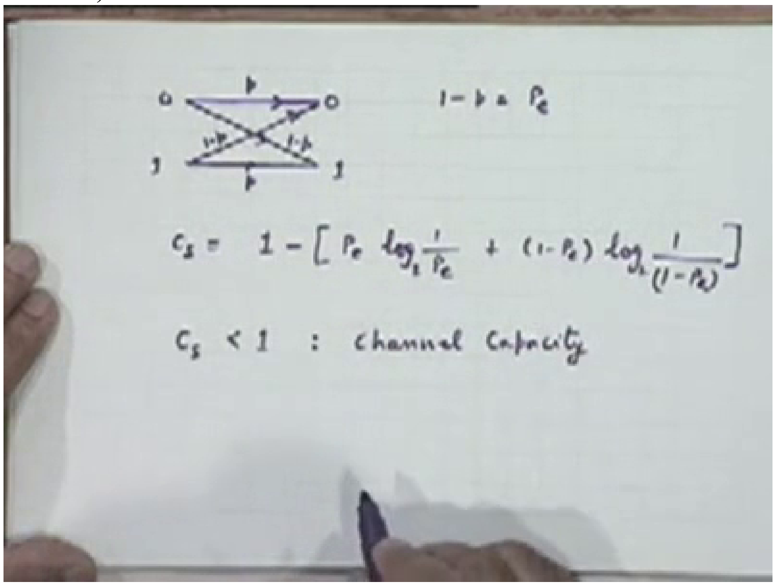
if we have time, we will even prove this formula later. At the moment I just want to introduce a concept. This C s is a number; it is very easy to verify from this expression that it would be less than 1. And that is precisely what is, what can be calculated to be the channel capacity of the binary symmetric model, associated with the binary symmetric model.

(Professor – student conversation starts)

Student: Sir, log to base?

Professor: Log to the base 2,

(Refer Slide Time 31:42)



$1 - p = P_e$

$$C_s = 1 - \left[P_e \log_2 \frac{1}{P_e} + (1-P_e) \log_2 \frac{1}{(1-P_e)} \right]$$

$C_s < 1$: Channel Capacity

Ok.

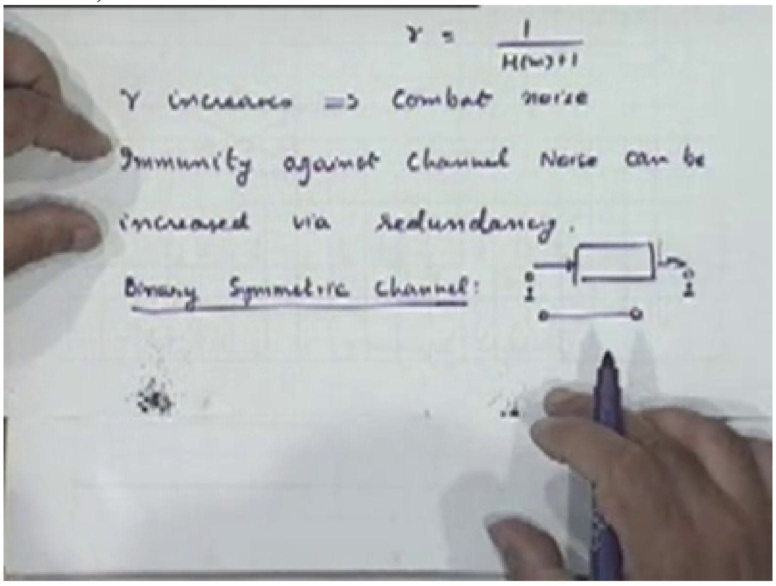
Student: Sir what do you mean by channel capacity?

Professor: That is precisely what we are discussing, Ok.

(Professor – student conversation ends)

Remember at what I point I introduced, I digressed into this. The point was I wanted to discuss the immunity against random noise can be increased by introducing redundancy

(Refer Slide Time 32:12)



$$\gamma = \frac{1}{H(w)+1}$$

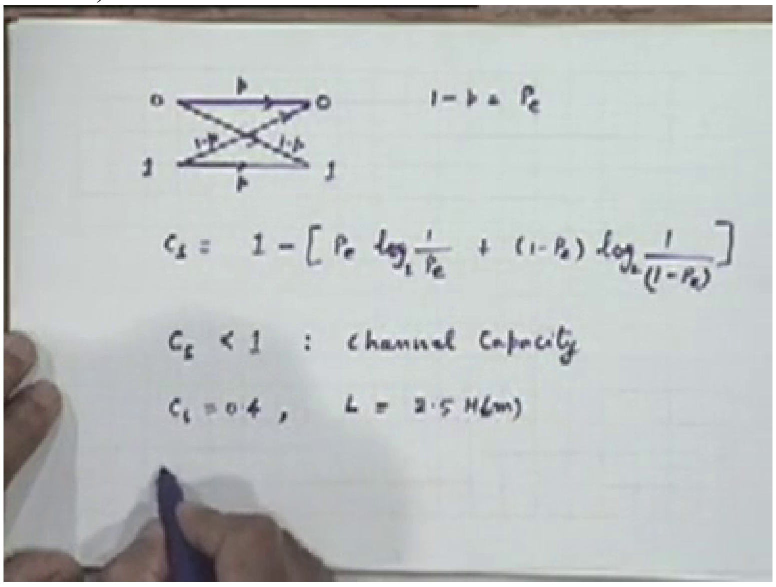
γ increases $\Rightarrow$ Combat noise

Immunity against Channel Noise can be increased via redundancy.

Binary Symmetric Channel:

and the question I am addressing now is how much redundancy one has to introduce? And the answer is that depends on the channel capacity, right? This number of channel capacity associated with the specific channel model. For example if you have a channel capacity let us say point 4, Ok, then you would require an average codelength equal to 2 point 5 times H m. Because,

(Refer Slide Time 32:45)



$1 - p = P_e$

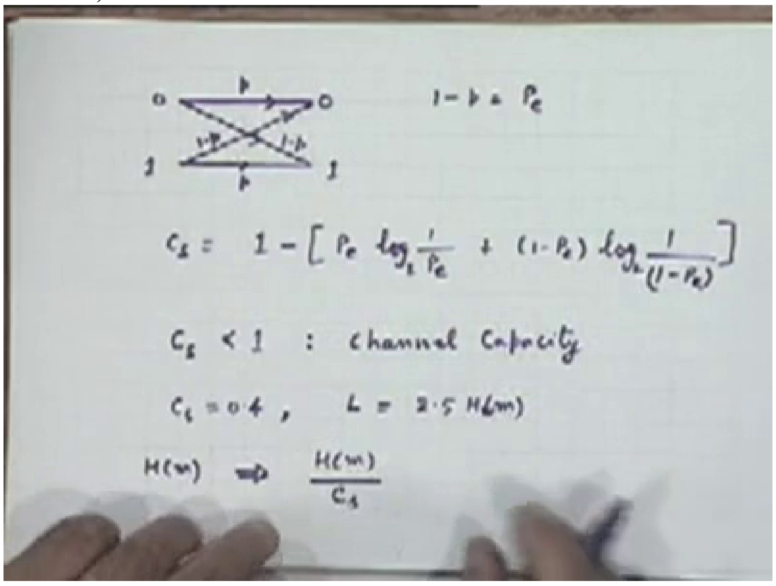
$C_s = 1 - \left[P_e \log_2 \frac{1}{P_e} + (1-P_e) \log_2 \frac{1}{(1-P_e)} \right]$

$C_s < 1$: Channel Capacity

$C_s = 0.4$, $L = 2.5\, H(m)$

well the result is that if you have a source with entropy H m and a channel with capacity C sub s then you must use on an average so many redundant bits, an average codelength

(Refer Slide Time 33:07)



$1 - p = P_e$

$C_s = 1 - \left[P_e \log_2 \frac{1}{P_e} + (1-P_e) \log_2 \frac{1}{(1-P_e)} \right]$

$C_s < 1$: Channel Capacity

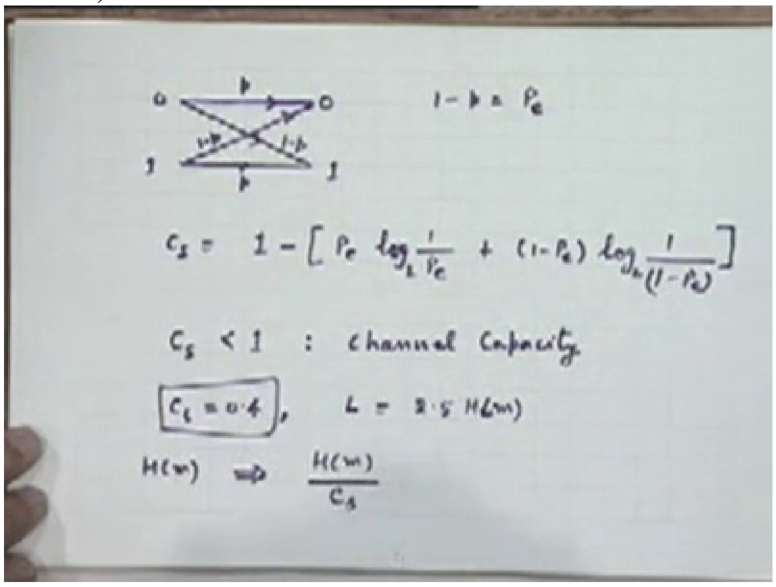
$C_s = 0.4$, $L = 2.5\, H(m)$

$H(m) \Rightarrow \frac{H(m)}{C_s}$

of this value, H m divided by C sub s. That is how this 2 point 5 H m comes. Ok. This tells me that for every bit of the message, every bit of the message symbol that I transmit in this particular example, I must transmit 1 point 5 redundant bits, right?

So this notion of channel capacity tells me how many redundant bits I must add to my basic information which I am interested in sending. What I am interested in sending is an average information rate of H m bits per symbol, right? But if I want some kind of immunity against channel then that tells me that I must adopt at least 1 point 5 bits to every single bit of the message symbol that I transmit so as to hopefully get some kind of immunity, reasonable immunity or possibility of error free communication on a channel of this capacity.

(Refer Slide Time 34:16)


0
p
0
1-p
1-p
1
p
1

$1 - p = P_e$

$$C_s = 1 - \left[P_e \log_2 \frac{1}{P_e} + (1-P_e) \log_2 \frac{1}{(1-P_e)} \right]$$

$C_s < 1$: Channel Capacity

$\boxed{C_s = 0.4}$, $L = 2.5\, H(m)$

$H(m) \Rightarrow \frac{H(m)}{C_s}$

If the capacity was point 8, then obviously we will come down. The amount of redundancy that we need to add becomes the need for having additional redundancy also correspondingly comes down.

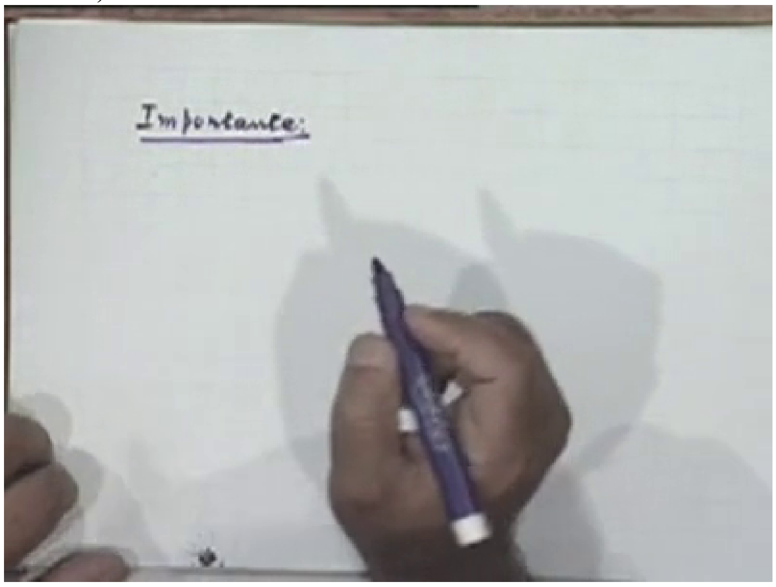
If C s is 1, you have a perfect channel, right? No redundancy is needed and obviously that will be hopefully a situation with, what kind of situation?

(Professor – student conversation starts)
Student: p equal to zero
Professor: p equal to zero that is no noise is present in the system, right?
(Professor – student conversation ends)

So somehow we have, the amount of redundancy that we end up using, we require to use depends on this notion of channel capacity. This is just through an example I wanted to illustrate what channel capacity will eventually tell us. How does it come into the picture is something I will take up now. But suppose I eventually, I want to understand what channel capacity is, this is how, this is the kind of thing I can do with it in system design, Ok. This kind of conclusion I can draw from this.

Now the importance of what Shannon said was the following. I am just trying to recollect for you now what we discussed on the very first day when I started discussing information theory.

(Refer Slide Time 35:41)

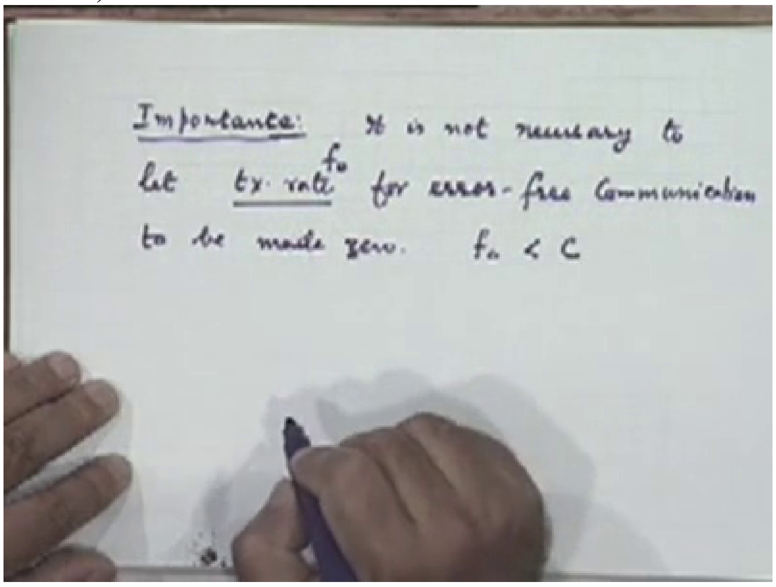


Do you remember we discussed the trade-off between error probability and signal power and data rate, right? Our conventional classical communication theory that we have read so far tells us that if I want zero error probability or error free communication, either I must increase my average transmission power to very large values, infinite values, or I must decrease my data rate or information rate down to zero, isn't it, so as to make the average power, average energy per symbol very, very large, infinite and therefore eventually get the same zero error probability situation, right.

This is where Shannon came and changed things. He said that it is not necessary that your average information rate be brought down to zero, Ok. It is possible to get error-free communication even when the average information rate is finite and your transmitted power is also finite. The only thing you have to make sure is the average information rate that you transmit is no more than the channel capacity. It is a number, a constant number which he called the channel capacity, which is a function of the transmitted power as well as the, let us say, the noise that you have in the channel, the total noise that you are dealing with in the channel.

So importance of Shannon result is precisely this, that it is not really necessary to let the transmission rate or the information rate for the error-free communication to be made zero. What is really necessary is that suppose I denote this error-free 0:38:01.3 transmission rate by say F, F sub naught. It is only necessary that F naught be kept below

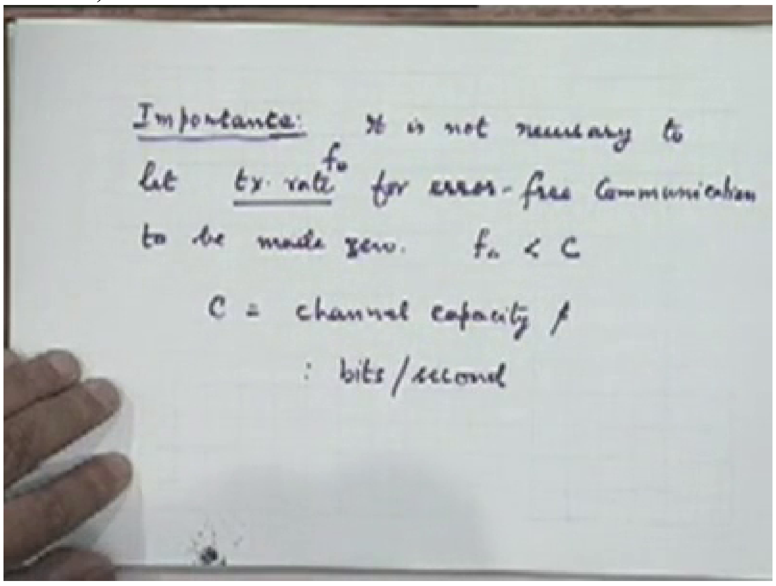
(Refer Slide Time 38:08)



Importance: It is not necessary to let $\overset{f_0}{\text{tx. rate}}$ for error-free Communication to be made zero. $f_0 < C$

a constant value C and that constant value is your channel capacity measured in bits per second. These are units of channel capacity,

(Refer Slide Time 38:30)



Importance: It is not necessary to let $\overset{f_0}{\text{tx. rate}}$ for error-free Communication to be made zero. $f_0 < C$

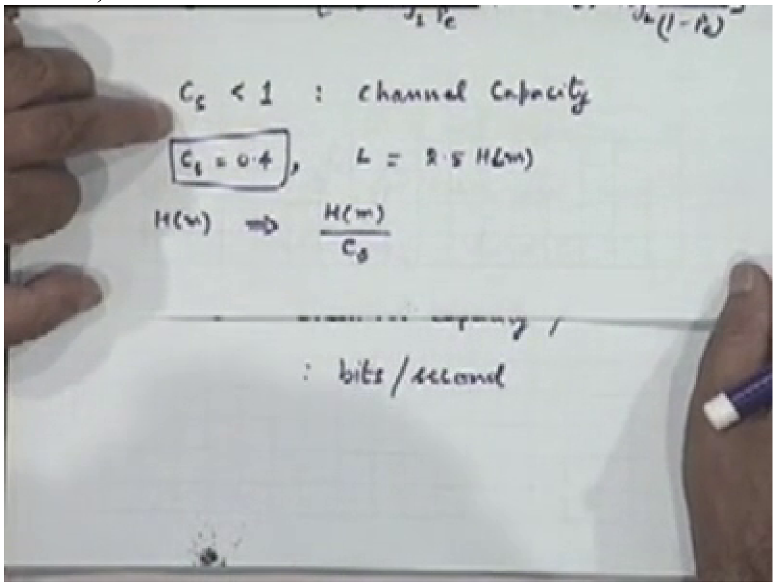
C = channel capacity /

: bits/second

Ok.

Now if you combine these two information, here I said C s, this number

(Refer Slide Time 38:43)



$C_s < 1$: Channel Capacity

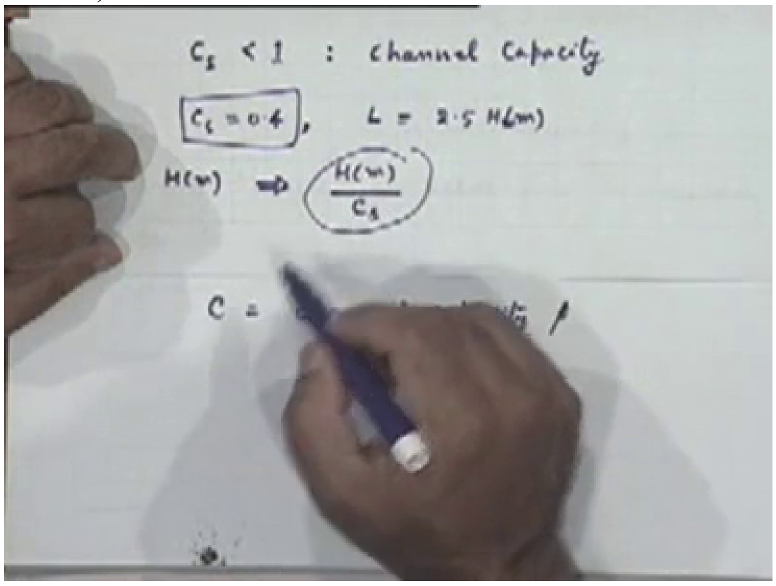
$\boxed{C_s = 0.4}$, $L = 2.5\, H(m)$

$H(m) \Rightarrow \frac{H(m)}{C_s}$

: bits / second

C s I defined to be a number less than 1, right, what does this represent? That for every symbol that I transmit I cannot, every symbol I have to transmit I have to, the number of effective bits that I can transmit per symbol is essentially point 4, that is what it means. Because the point 6 part of it has to be redundant bits. That is the meaning of that. So when I say I need to add so many redundant bits, my total

(Refer Slide Time 39:16)



$C_s < 1$: Channel Capacity

$\boxed{C_s = 0.4}$, $L = 2.5\, H(m)$

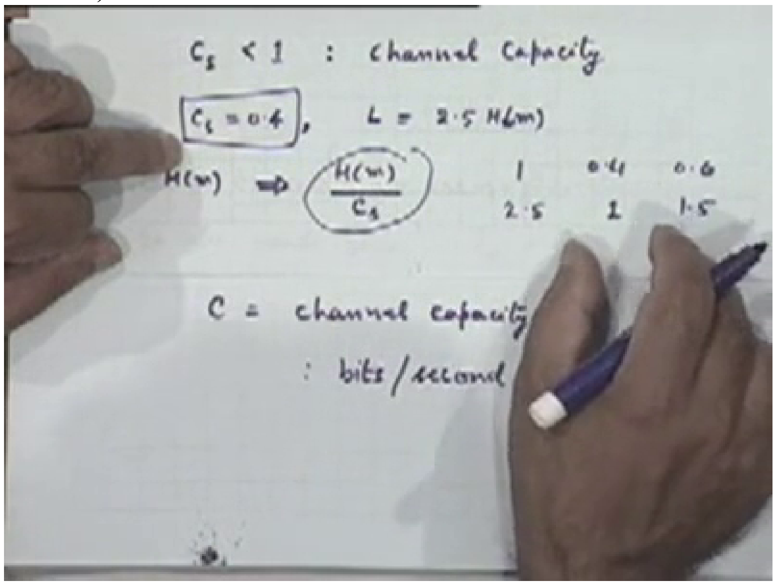
$H(m) \Rightarrow \frac{H(m)}{C_s}$

$C =$

number of bits have to be so many, right, in other words one can imply to mean that for finally for, if you transmit 1 bit, out of that point 4 can be your information and point 6 has to be the redundant part.

For example here we discussed if you transmit 2 point 5, then 1 is your information bit and 1 point 5 are your redundant bits.

(Refer Slide Time 39:42)



$C_s < 1$: Channel Capacity

$\boxed{C_s = 0.4}$, $L = 2.5\,H(m)$

$H(m) \Rightarrow \frac{H(m)}{C_s}$

1	0.4	0.6
2.5	1	1.5

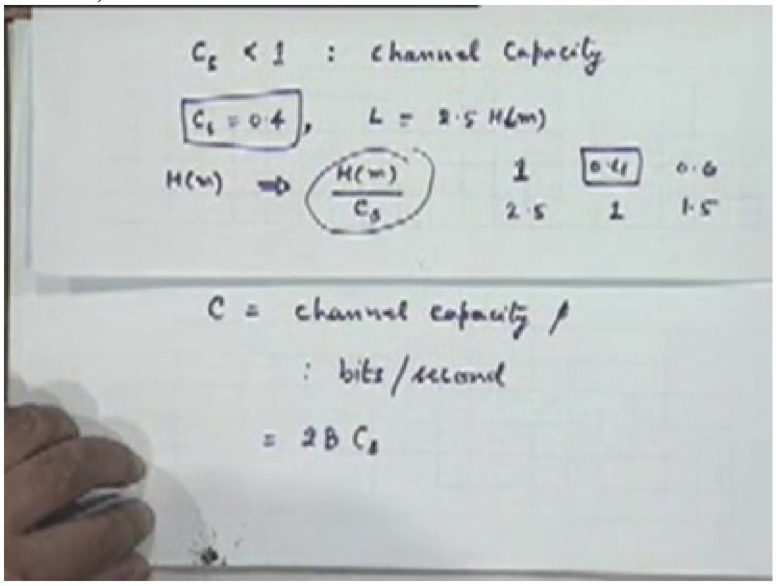
C = channel capacity

: bits / second

Now if you take the fractions, right, this is what it means. If you transmit, if you are transmitting out a single bit, out of that single bit the information being by that bit is only point 4 bits, right? The rest of it is redundant information. That is another way of looking at this capacity of the channel, this measure of the capacity of the channel which is dimensionally less than 1, which is dimensionally constant and it is less than 1. But if I want to convert into bits per second how will I do that?

That will depend on how many symbols per second I can transmit on that channel, right, which I already know from other theories. From, for example, my Nyquist theory, that tells me how many symbols per second I can transmit on a particular channel. For example if the channel has a bandwidth B and it is a band limited channel, I do not have, I can hope to transmit symbols without inter-symbol interference at a maximum rate of 2 B per second, right? And therefore I can combine these two results and say that this will be equal to 2 B times C s,

(Refer Slide Time 40:53)



$C_s < 1$: channel capacity

$\boxed{C_s = 0.4}$, $L = 2.5\, H(m)$

$H(m) \Rightarrow \frac{H(m)}{C_s}$ $\quad 1 \quad \boxed{0.4} \quad 0.6$

$\quad\quad\quad\quad\quad\quad\quad 2.5 \quad 1 \quad 1.5$

C = channel capacity /

: bits/second

$= 2B\, C_s$

Ok.

If I transmit 2 B symbols per second and every symbol is allowed to carry only this much information, that is what channel capacity really means. It tells the, what is the maximum amount of information that every single symbol that you transmit on the channel is allowed to have, right? Then this is a measure of my overall channel capacity. It is slightly abstract notion but if you think about it, it is very easy to understand what we are talking about.

(Professor – student conversation starts)

Student: Sir this bandwidth and the source, when you are after encoding and all that

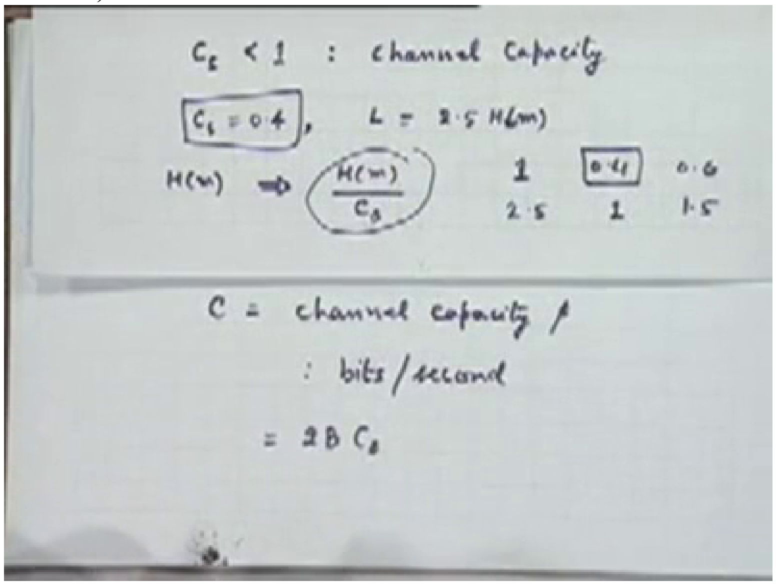
Professor: Yes

(Refer Slide Time 41:31)

Student: So the symbol rate will be 2 B C s or only C s, converted into 2 B?

Professor: No, channel capacity is

(Refer Slide Time 41:39)



$C_s < 1$: Channel Capacity

$\boxed{C_s = 0.4}$, $L = 2.5\,H(m)$

$H(m) \Rightarrow \dfrac{H(m)}{C_s}$ 1 $\boxed{0.4}$ 0.6

2.5 1 1.5

C = channel capacity /

: bits/second

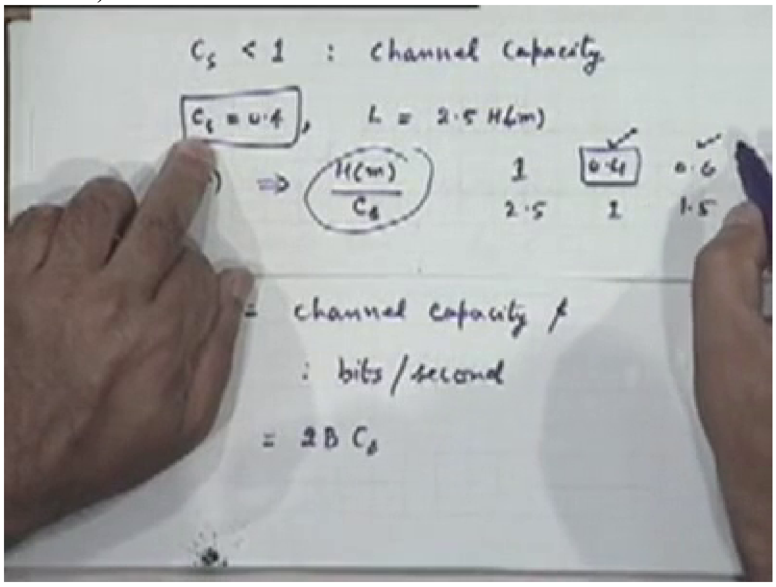
$= 2B\,C_s$

absolutely independent of source. It has nothing to do with source. It is an attribute of only the channel. Just like the entropy was an attribute only of the source, it told me how much average information is coming out of its source, right? Similarly channel capacity is an attribute purely of the channel. It tells me for that kind of a channel, which has this kind of a noise, and on which this much power is being transmitted, what kind of redundancy must be introduced? That means if you are transmitting a symbol what should be its composition in terms of its information part and redundant part.

(Professor – student conversation ends)

Every symbol that is transmitted, for example in this case must not carry more than point 4 bits of information part and the rest should be

(Refer Slide Time 42:30)



$C_s < 1$: Channel Capacity

$\boxed{C_s = 0.4}$, $L = 2.5\,H(m)$

$\Rightarrow \frac{H(m)}{C_s}$ $\quad \frac{1}{2.5} \quad \frac{\boxed{0.4}}{1} \quad \frac{0.6}{1.5}$

= channel capacity /

: bits/second

$= 2B\,C_s$

redundant information. So it is an attribute only of the channel. It has nothing to do with the source. Please try to understand this. One, entropy is an attribute only of the source. Two, the channel capacity is an attribute only of the channel. That is why it is called channel capacity, Ok.

(Professor – student conversation starts)

Student: Does C s depend 0:42:53.8?

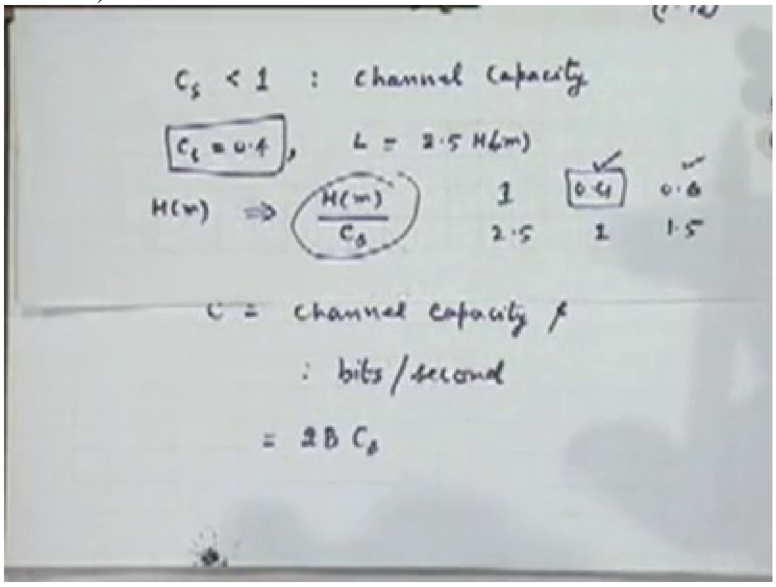
Professor: C s in terms of?

Student: Probability of error.

(Refer Slide Time 42:56)

Professor: That is right. It depends on, I said

(Refer Slide Time 43:02)



$C_s < 1$: Channel Capacity

$\boxed{C_s = 0.4}$, $L = 2.5\,H(m)$

$H(m) \Rightarrow \frac{H(m)}{C_s}$ $\quad \frac{1}{2.5} \quad \frac{\boxed{0.4}}{1} \quad \frac{0.6}{1.5}$

C = Channel capacity /

: bits / second

$= 2B\,C_s$

the kind of noise we have on the system. It depends on the kind of signal power you are transmitting. And for the binary symmetric model, all that is embedded in your error probability.

Student: That is, it also depends on signal power.

Professor: That is right; error probability depends on signal power, noise power, everything

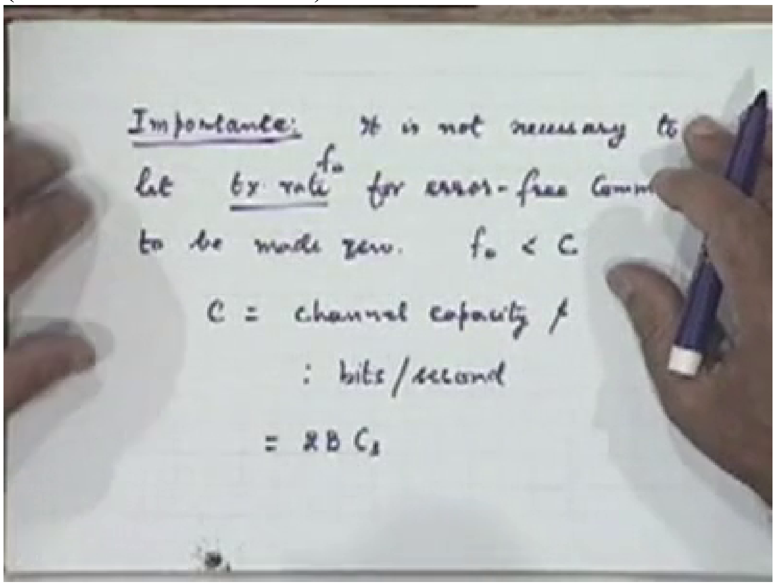
Student: 0:43:19.4 Channel capacity also depends on signal power?

Professor: Obviously. It will depend on everything that you have in the communication system, only thing is for the binary symmetric channel, the only parameter that you need to finally calculate, to calculate this number is this error probability. You must somehow calculate that number first and then you are able to calculate your channel capacity. So it is an indirect way of doing things. Have you understood the concept?

Student: How do you make sure that no error will be there?

Professor: Now we are coming to that. This is what Shannon said.

(Refer Slide Time 43:52)



Importance: It is not necessary to let **bit rate** f_0 for error-free comm
to be made zero. $f_0 < C$

C = channel capacity /
: bits/second
$= 2BC_s$

Shannon said that if you make sure that your data rate is less than, your information rate is less than this number, this constant number which is this, right which will depend on the channel, the kind of channel you have then it is theoretically possible to expect error-free transmission by some means or another, Ok. That is what Shannon said. And I would like to, like to try to justify how this kind of argument can be given.

(Professor – student conversation ends)

This is quite, quite against our normal expectation, isn't it? Because normal expectation is, no matter what you do, if you want zero error probability or error-free communication you need to have either infinite signal power or you have

(Professor – student conversation starts)

Student: Zero

Professor: Zero error rate, zero data rate or information rate, Ok

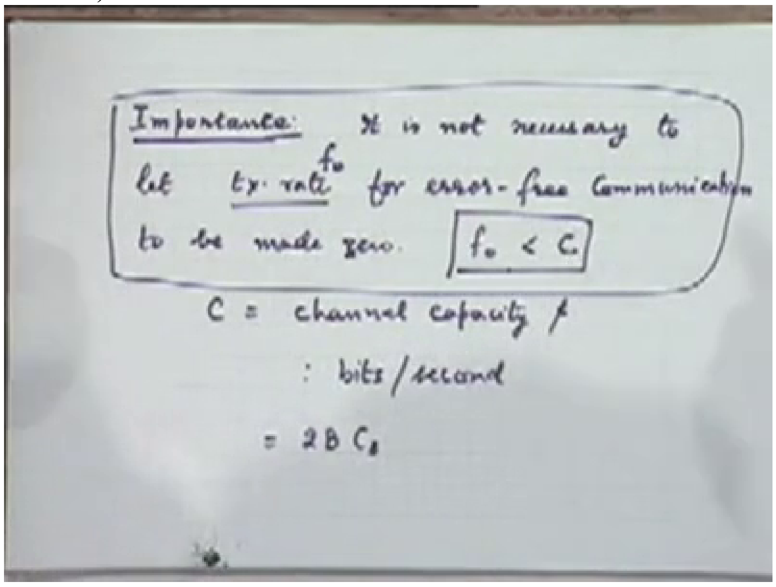
Student: Has he given any exact model of this?

Professor: Can you repeat your question please?

Student: Given model of the channel?

Professor: No this result, this result is independent of what the value of channel capacity is. The value of the channel capacity will depend on the specific channel, right? But this result will be absolutely general, right. This number will be different for different channels.

(Refer Slide Time 45:16)



Importance: It is not necessary to let $\overset{f_0}{\text{tx. rate}}$ for error-free Communication to be made zero. $\boxed{f_0 < C}$

C = channel capacity

: bits/second

$= 2B\,C_s$

The capacity of the channel will depend on what channel you are working with, what kind of modulation scheme you have, what is the physical path, what is bandwidth, right, all this will determine what C is. But once you have that value C, then this is the result to use.

Student: Sir

Professor: Yes

Student: The maximum rate we have on the channel of bandwidth B is 2 B, right?

Professor: Symbol rate

Student: Defining the Symbol rate

Professor: Right

Student: And we define the, because C s is nothing but the efficiency.

(Refer Slide Time 45:45)

So out of....

Professor: C s is not efficiency.

Student: It will effectively turn out to that.

Professor: Ok you can put it that way, yes.

Student: So effectively out of 2 B bits the number of information bits are 2 B C s.

Professor: 2 b symbols. Please distinguish between symbols and bits.

Student: Fine. It is 2 B into C s.

Professor: Right.

Student: And the remaining is redundant information.

Professor: That is right.

Student: That is how you can say.

Professor: That is right. The, one of the answers he has already given, that this result is coming because we have redundancy, right. Because everything we are now transmitting is not pure information. Along with information we have agreed to waste our transmission capacity to transmit also

Student: Redundancy

Professor: redundancy, right. Now the question is how does redundancy also create this kind of situation. That is what we need to understand, right? And that is what we would like to understand next.

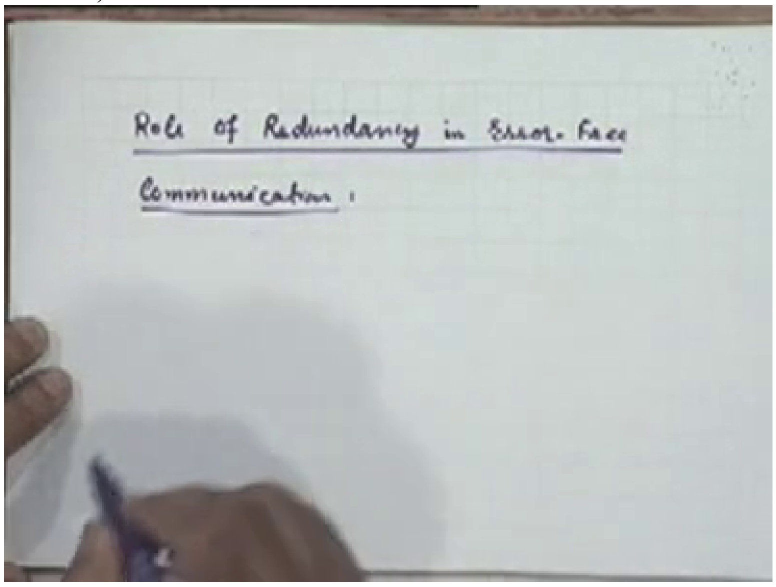
(Professor – student conversation ends)

How we can get from introductional redundancy the possibility of error-free communication? This is what we would like to understand next. So let us look at that.

It is clear that if at all this result has come from somewhere; it has come from redundancy, transmission of redundancy, redundant information, isn't it? Because this is the only additional thing we are doing which we are not thinking of doing earlier, right. So somehow transmission of redundant information has a major role to play in achieving, in getting Shannon's results. So let us see how redundancy can lead to this kind of a result. Mind you my argument is going to be purely intuitive. This is the result which can be, Shannon proved mathematically, very rigorously. But we have no time and scope for doing that mathematical argument, right.

We will only discuss intuitively how this result can be justified, right. We will only look into justification for this result, nothing more than that. And that justification will be easy to understand if you understand the role of redundancy in error-free communication.

(Refer Slide Time 48:07)

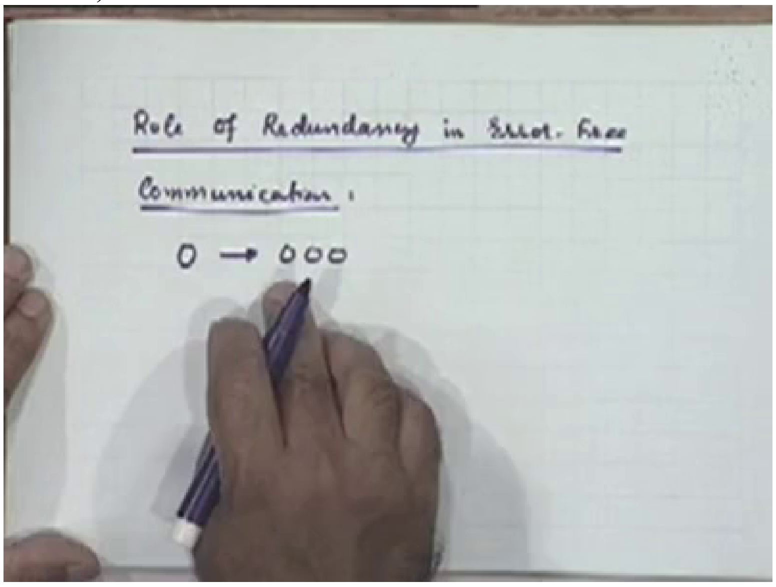


Role of Redundancy in Error-Free Communication :

And to do that I will take a very simple kind of introducing, simple method of introducing redundancy. Suppose I have a very simple kind of a source, outputs are zeroes and 1s. And I introduce redundancy in the following way.

Every zero that comes along, I switch 1 bit of output that is coming out, I add 2 redundant information bits, 2 redundant bits, both equal to whatever has come out. For example if zero comes out,

(Refer Slide Time 48:44)

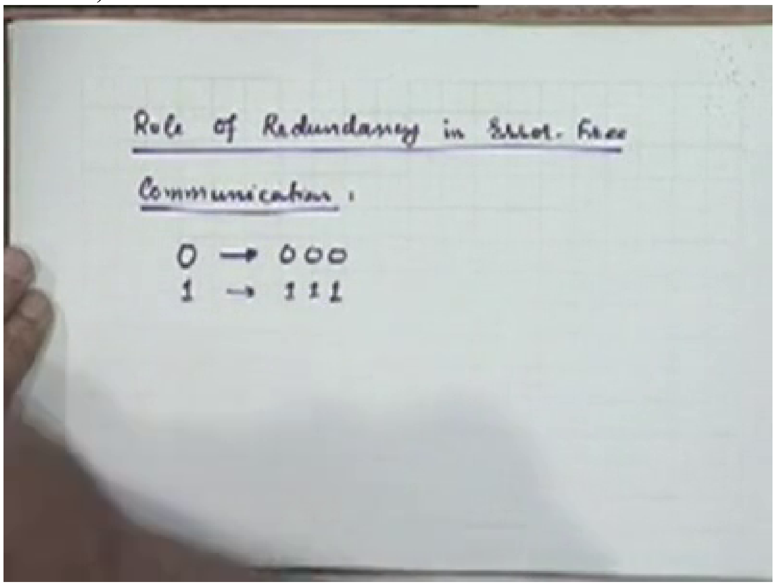


Role of Redundancy in Error-Free Communication:

$0 \rightarrow 000$

I transmit not one bit but, one zero, but transmit three successive zeroes, right? Similarly if 1 is emitted by the source, I transmit three successive 1s.

(Refer Slide Time 48:58)



Role of Redundancy in Error-Free Communication:

$0 \rightarrow 000$
$1 \rightarrow 111$

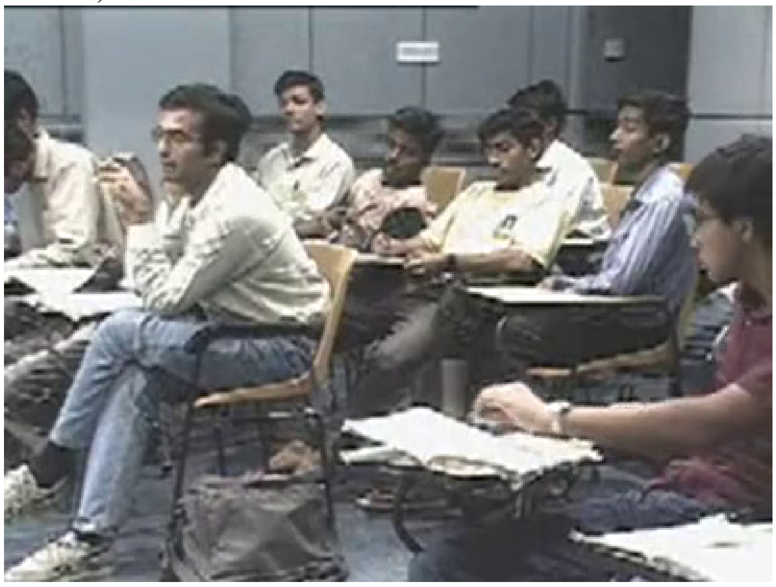
This is called redundancy by just repeating, Ok. Just keep on repeating a few times, may be 3 times, may be 5 times whatever bit or symbol that has been emitted by the source. Very simple way of introducing redundancy.

Now if I do that, you can already see something. That at the receiver what you will receive is not necessarily zero zero zero or 1 1 1 right. Something may change because of noise. You may make a wrong decision because of noise, because of your finite error probability. May be zero zero zero will become zero zero 1, or zero 1 zero, or 1 zero zero, right. It can also become zero 1 1 but the probability of that occurrence is going to be smaller than one of the bits only going in error, right? So therefore as a receiver I will now also have a decoder corresponding to this encoding scheme, which in this simple case we can predict what it should be. It should be a majority logic. The decoder should be essentially a majority logic system which will decide the correct symbol based on the number of major symbols that you get here.

So if you get two zeroes out of 3, you would say a zero was transmitted. If you get two 1s out 3, you will say a 1 has been transmitted, right. So there is already, you can see the possibility of reducing your error probability as far as this zero and 1 is concerned. So I can correct how many errors? Up to 1 error can be corrected by this simple repeated transmission scheme.

(Professor – student conversation starts)
Student: If there are more than one error then it will be...
Professor: If there are more than one errors there is no way to, you will make a wrong decision, right? But since we are working on the assumption that more than one error will have much smaller, or much smaller possibility of occurrence. Of course this does not lead you to Shannon's results straightaway.
Student: Sir, it is absolutely valid then after transmission suppose we have an error, is it due to some

(Refer Slide Time 51:08)



disturbance within the surrounding or the channel?

Professor: Whatever may be the reason.

Student: Some spike 0:51:12.0 in and nearby channel may cause disturbances, chances of getting two errors simultaneously, 0:51:20.3 is that gap really that low?

Professor: Well if it is a well-designed system, yes. You know, if you have a problem in which you have to worry about manmade disturbances then your communication channel model has to be different from additive white Gaussian noise channel model. At the moment we are discussing additive white Gaussian channel noise model. But if...

Student: Channel 0:51:45.5 is through a longer distance, only for...

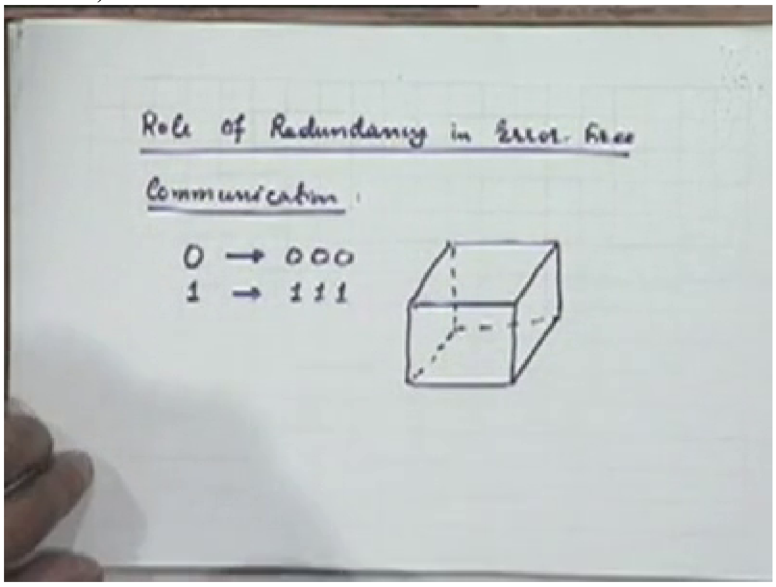
Professor: This result was derived only for additive white Gaussian noise channel, right which is the most common model which one has to worry about for most kinds of channels. But yes, there are situations where your kind of noise disturbances are there, then the results are different. But corresponding results exist also for those cases, Ok.

(Professor – student conversation ends)

We must understand the limitations under which we have this result. Ok, so the moral is that we can increase these repetitions and our capacity to correct errors will also improve. For example instead of doing a repeated transmission of only 3 times, I do a repeated transmission 5 times. How many errors can I correct? Two errors, right? I can correct 2 errors and so on. By introducing more redundancy, I can correct more number of errors and that is what we can understand from this.

Now let me try to put the same thing, try to understand the same thing through a geometrical picture. And that picture is going to be very crucial for us. Let me draw that picture here. Let me consider

(Refer Slide Time 53:16)



Role of Redundancy in Error Free Communication :

$0 \rightarrow 000$

$1 \rightarrow 111$

the same code in which a zero is represented by three zeroes and 1 is represented by three 1s. Now how many different possible sequences I can receive at the receiver? 8 possible different sequences, right, corresponding to 3 received bits that I have. I can represent these 8 bits as 8 vertices of this cube, right.

Let us say the origin is sequence 0 0 0, and I will number the vertices carefully. Let us say each axis here, I have 3 axes, x, y and z, I can only change one bit along each axis. Let us say, this particular bit will be affected, I think this can be 0 1 0, this could be 0 0 1 and this could be 1 0 0, right? So this bit is affected along this axis, this bit is affected along this axis and the middle bit is affected along this axis. So by using this logic what will be the representation here?

Is it obvious?

(Professor – student conversation starts)

Student: 0 1 1

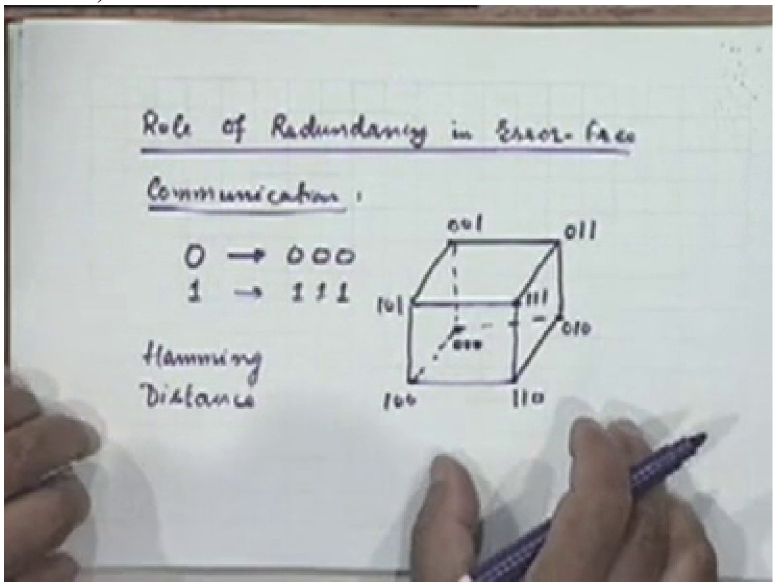
Professor: This will be 1 1 0, this will be 1 0 1 and this will be 1 1 1, right? So what do you see? That the 2 sequences which formed your codewords, you can think of these as your

codewords, right? Corresponding to an emission of zero, you are transmitting this word, this can be thought of as a codeword. Similarly 1 1 1 is a codeword, right? These lie on diagonally opposite vertices. That is one way of looking at it.
Student: What do we mean diagonally opposite?
Professor: Isn't it obvious? One vertex is here. The other is here. Do I need to explain that? I hope not. Ok.
(Professor – student conversation ends)

The other way of looking at it is which are the, which are the received words which will be decoded as 0 0 0, this one, this one and this one, right. They correspond to a single error occuring for transmission of 0 0 0. If this is transmitted and a single error occurs, where would you land? Either here or here, that is one of the neighboring vertices which are at a, what you call, a unit Hamming distance. Hamming distance is the number of places in which two digital words differ, right? So we say the distance between them is unity, distance between these two is also unity, distance between these two is also unity so these are, this distance is called Hamming distance.

Hamming was a gentleman who was

(Refer Slide Time 56:21)



well-known in both logic theory as well as information theory. There are also some codes by his name, Hamming codes, alright. Similarly if you transmit 1 1 1, the words that will be possibly received after errors or single errors will be these, again neighboring vertices, right?

And the reason why we could do single error correction was because all neighboring vertices with single errors, all vertices with single error are at a distance of 1 from the actual vertex, from the true vertex, true code vertex, right?

Whereas if you take a distance 2, for example what is the distance 1 1 1 and 0 1 0?

(Professor – student conversation starts)

Student: 2

Professor: It is 2, because you wll have to go like that or this, right? There are 2 bits which are different.

Student: Sir why is the the 0:57:21.2

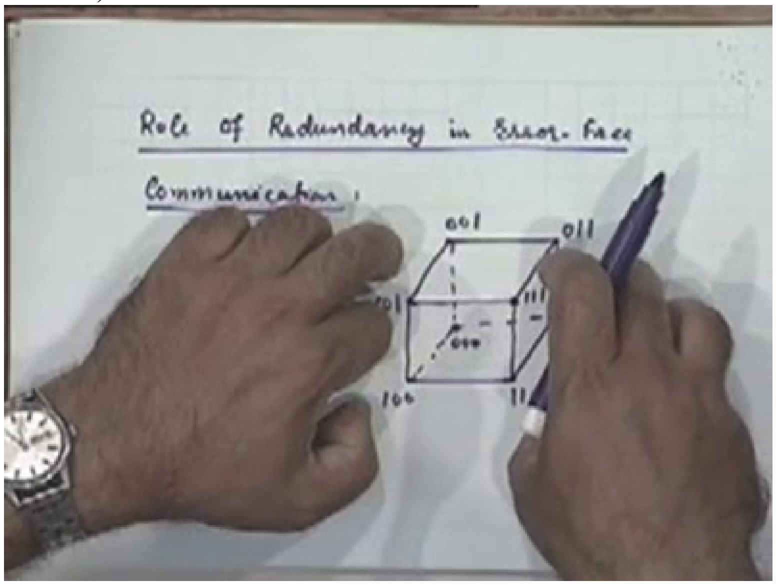
Professor: I am coming to that. Have you understood what I have said so far?

Student: 0:57:28.6 nearest neighbor.

(Refer Slide Time 57:28)

Professor: Yes, be a little patient. I am coming to something from here. I mean if you have not understood the picture itself,

(Refer Slide Time 57:36)



I can answer that question related to that. But what is the motivation of this picture is precisely I am coming to, right? Ok.

(Professor – student conversation ends)

Another way of looking at it is, that if I draw what I call two Hamming spheres, you can think of this as some kind of an abstract space, it is not real space in which this cube is existing because the only points which are really of interest to us are these points. These intermediate points which are drawn by lines do not really exist in this space, right? Because there is no way I can really get to one of these points. If I have what is called the Hamming space in which, along each axis I can only take two values, 0 and 1, right, essentially I have, I call this as Hamming cube, it is consisting of 8 vertices, full stop.

This Hamming cube is essentially a collection of these 8 vertices. Now I can therefore think of role of redundancy in this way. It is precisely because of two reasons I can do this error correction. One, that neighboring vertices were not used as codewords. If you picked up one vertex as a codeword, then you left out these neighboring vertices unassigned. They were not assigned as codewords, clear? Similarly you picked up this as a codeword, these neighboring vertices were not assigned as codewords, Ok. Number two, second point is that if I draw a sphere of radius unity around each of these points, around this point, around this point, they will be non-overlapping spheres. Now these are not again Euclidean spheres. What is a sphere? It is a locus of points at a unit distance from the center. In the Hamming space, therefore the sphere is essentially a collection of those points which are at a unit distance from this, from the center.

Which are these points? This point, this point and this point constitutes the Hamming sphere around this center. This point, this point and this point constitute the Hamming sphere around this center. And they do not overlap. In the sense they do not have a common point. These two sets do not have a common point. So the second way of looking at it is that Hamming sphere of unit radius around each of the codewords do not overlap, right? It is these two reasons which help us achieve this error correction. If you were doing that 5 repetition code, you will now be working with Hamming cube in 5 dimensions, not 3 dimensions. And in that 5 dimensional space, Hamming spheres of radius 2 around each of the two codewords would be non-overlapping, right. We will start from this picture next time to justify Shannon's result. Thank you.