Indian Institute of Technology Kanpur

National Programme on Technology Enhanced Learning (NPTEL)

Course Title

Error Control Coding: An Introduction to Linear Block Codes

Lecture – 1C

Introduction to Error Control Coding-III

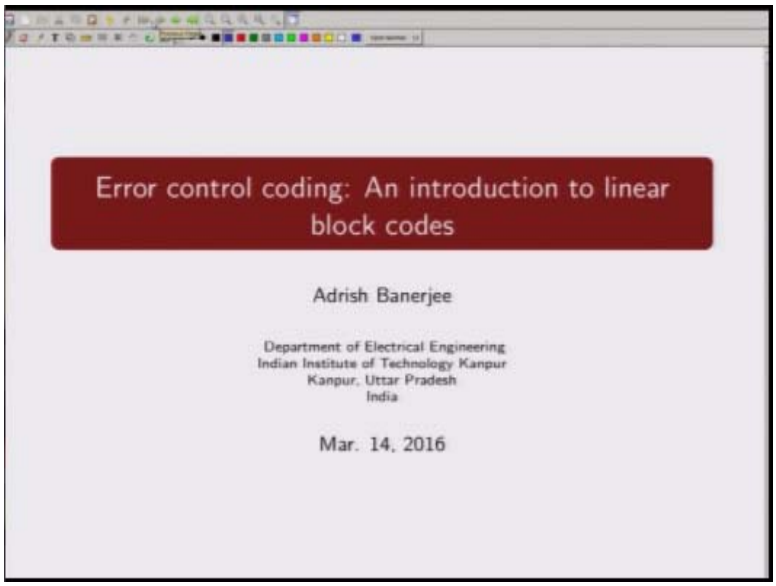
by

Prof. Adrish Banerjee

Department of Electrical Engineering, IIT Kanpur

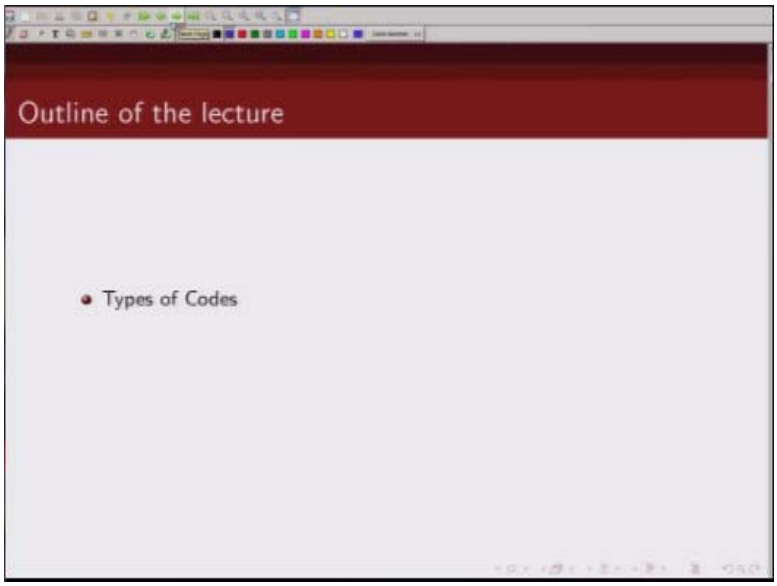
Welcome to the course on error control coding.

(Refer Slide Time: 00:16)



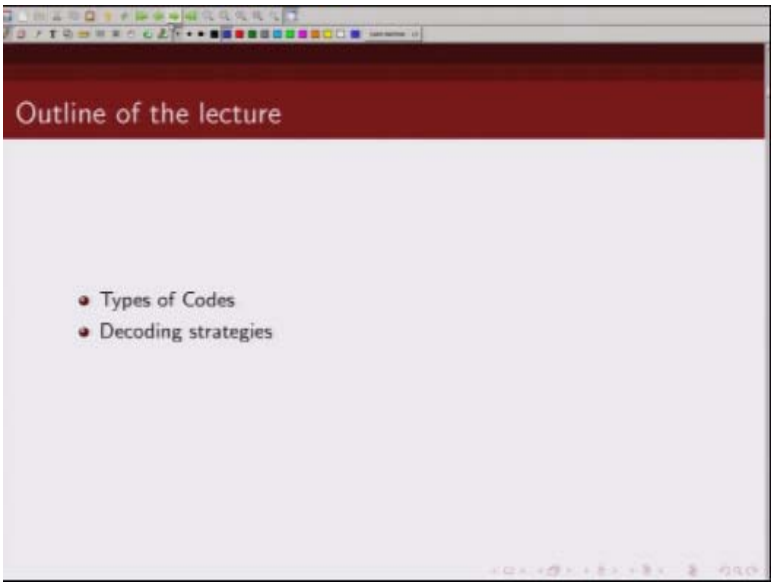
An introduction to linear block codes, I am Adrish Banerjee, in this lecture we are going to conclude our discussion on interaction and so in this lecture I will first describe what is a difference between block course and convolutional codes, and then we will talk about some very simple decoding strategies and finally we will explain what we mean by forwarded correction automatic repeat request and hybrid ARQ.

(Refer Slide Time: 00:50)



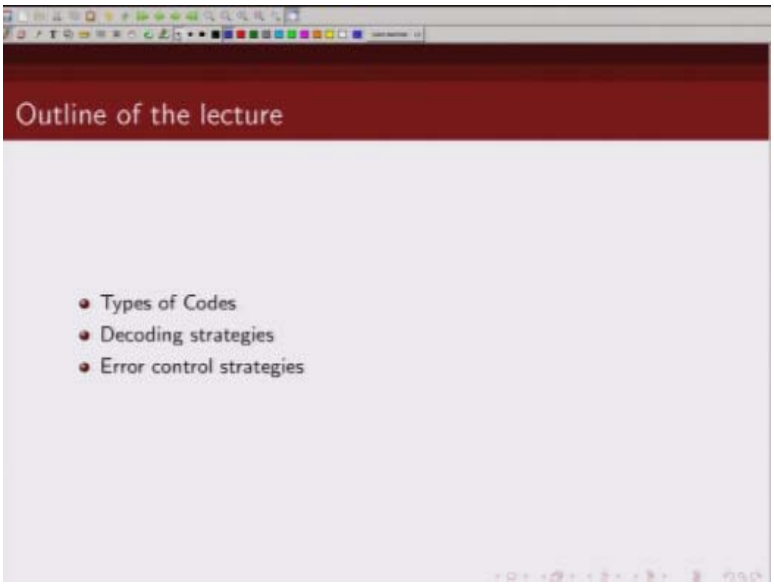
So as I said we will first describe, so error correcting course can be broadly classified into two classes, block course and convolutional course. We will describe what is meant by block code and what is meant by convolutional code and will bring out a difference and similarities between the two.

(Refer Slide Time: 01:11)



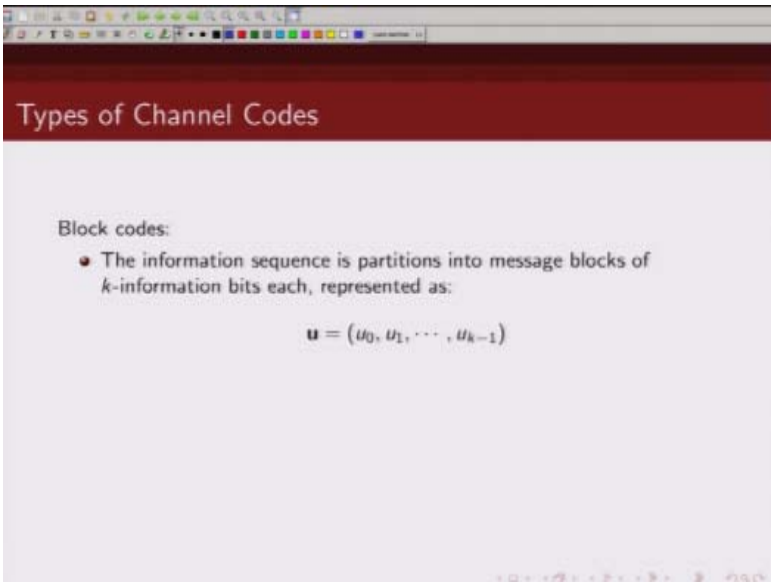
Then we will talk about various decoding strategies.

(Refer Slide Time: 01:14)



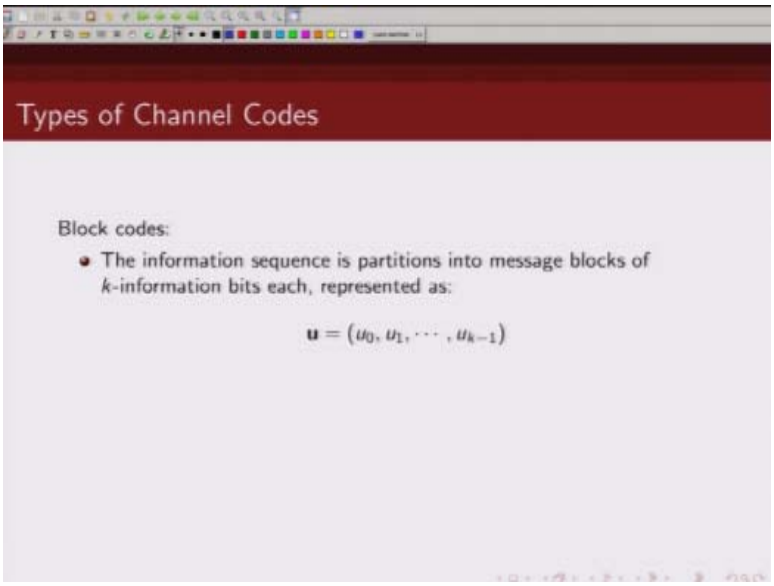
And finally we will talk about what we mean by forward error correction, hybrid ARQ, and automatic repeat request.

(Refer Slide Time: 01:25)



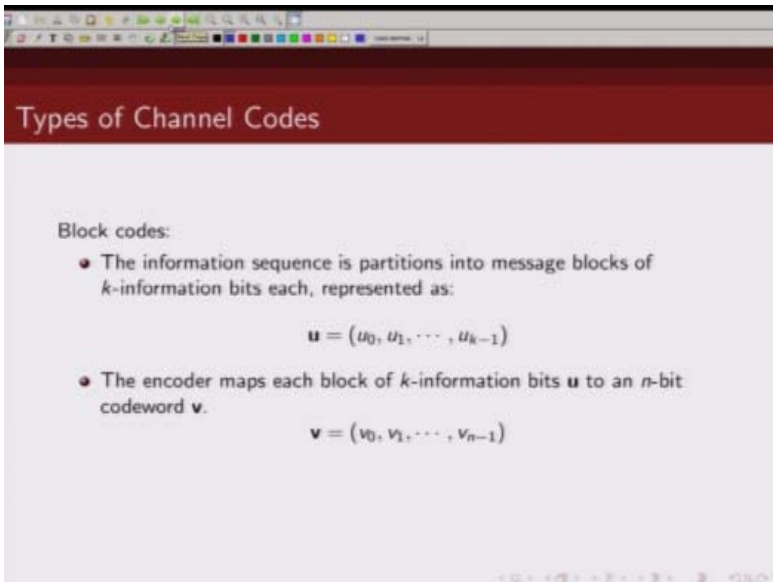
So we start with what is block codes, so as the name suggests in block codes we take a block of K bits and map it to an n bit code word so our information sequence is passed into

(Refer Slide Time: 01:44)



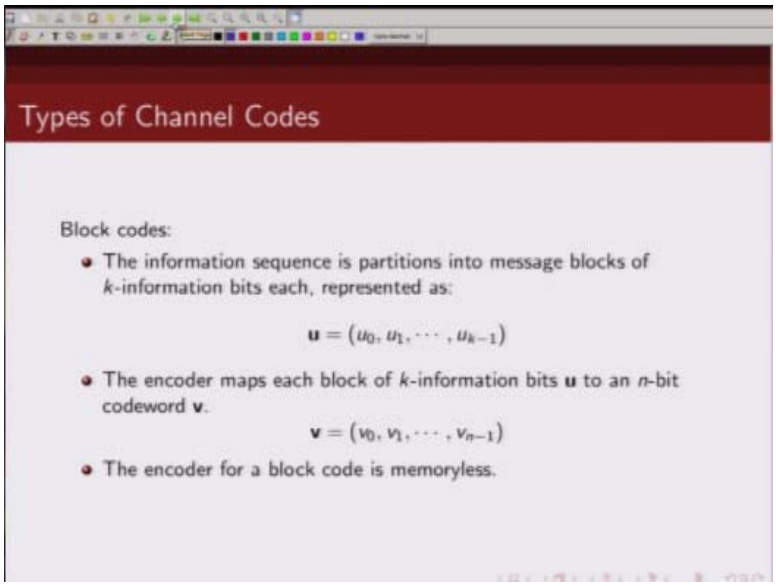
Blocks of k bits and we take this block of k bits and map it to block of n bits.

(Refer Slide Time: 01:54)



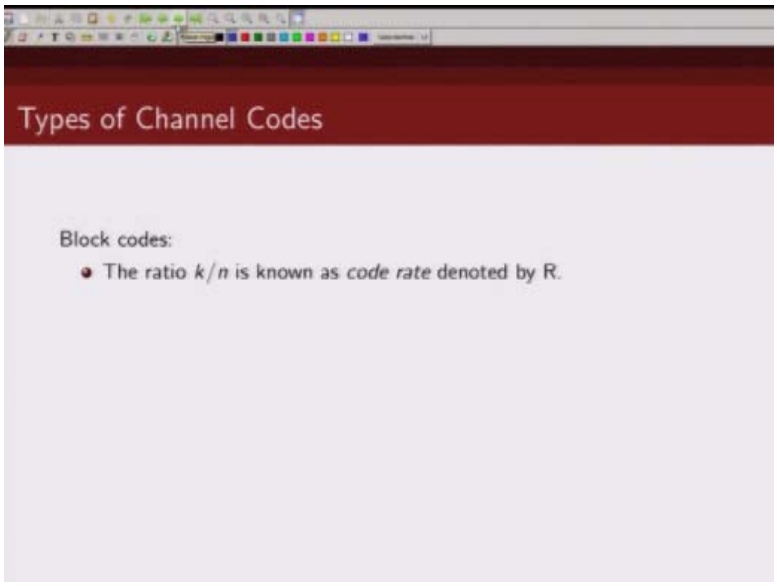
So we denote our information sequence by u so this is a k bit sequence u_0, u_1,to u_{k-1} and our encoder is going to map this k bits into an n bit sequence which is denoted by v.

(Refer Slide Time: 01:54)



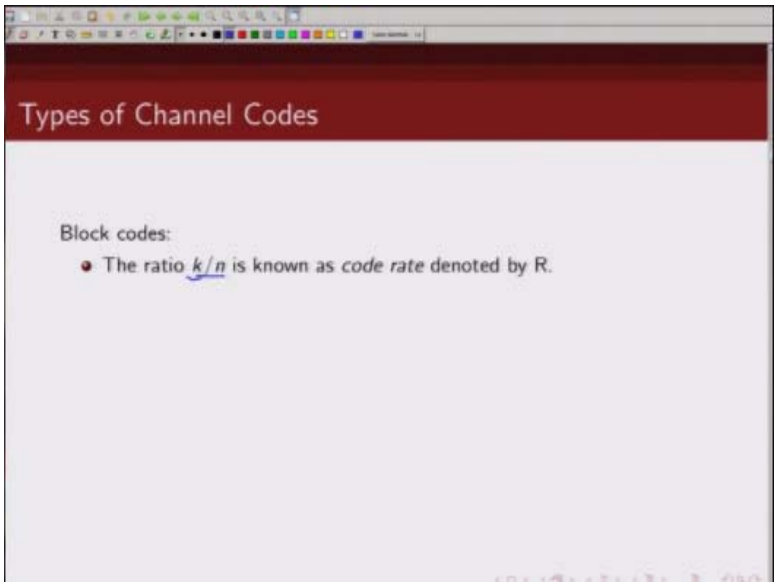
Now in block codes the encoder is memory less. What do we mean by that, so when we encode a block of K bits our output depends on only on that current block of k bits, it does not depend on what were the previous blocks of data, it only depends, output only depends on the current k bits so that is one property of block codes which makes it different from convolutional codes, block codes are memory less.

(Refer Slide Time: 02:51)



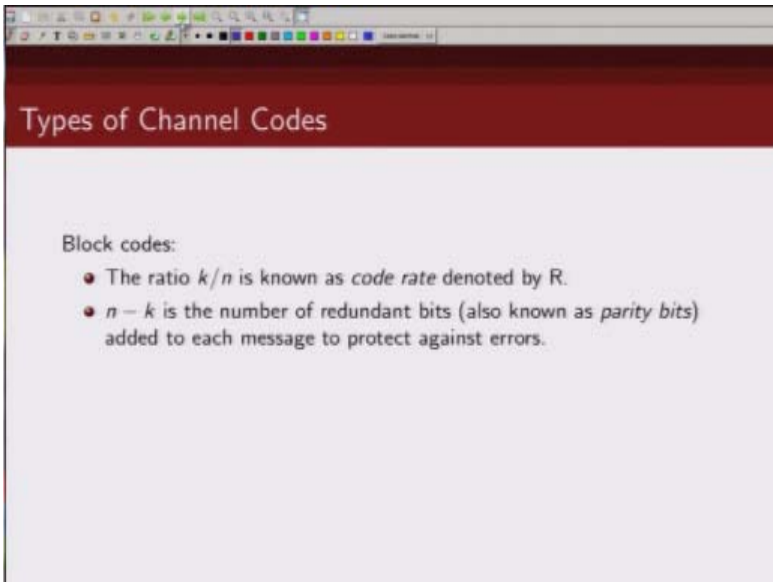
As we mentioned in the previous lectures we define our code rate to be.

(Refer Slide Time: 02:57)



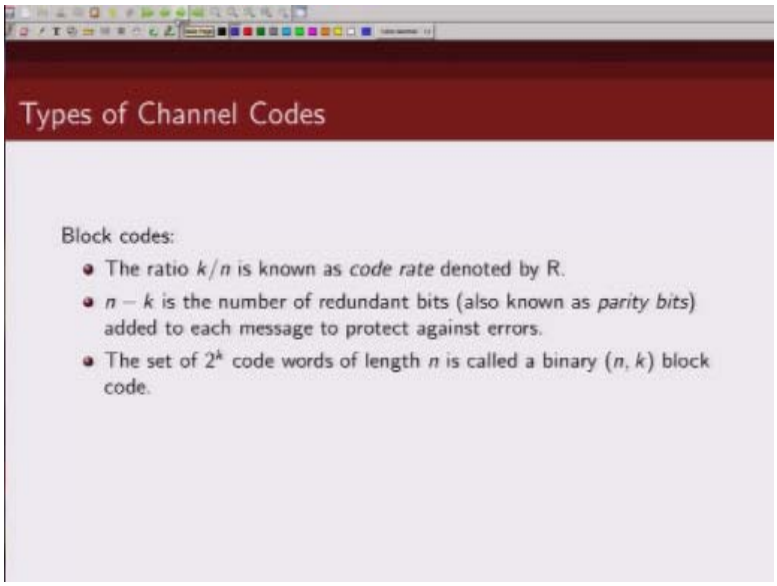
The ratio of number of information bits to number of coded bits. So the ratio of information bits to coded bit is basically, will be denoted by code rate.

(Refer Slide Time: 03:13)



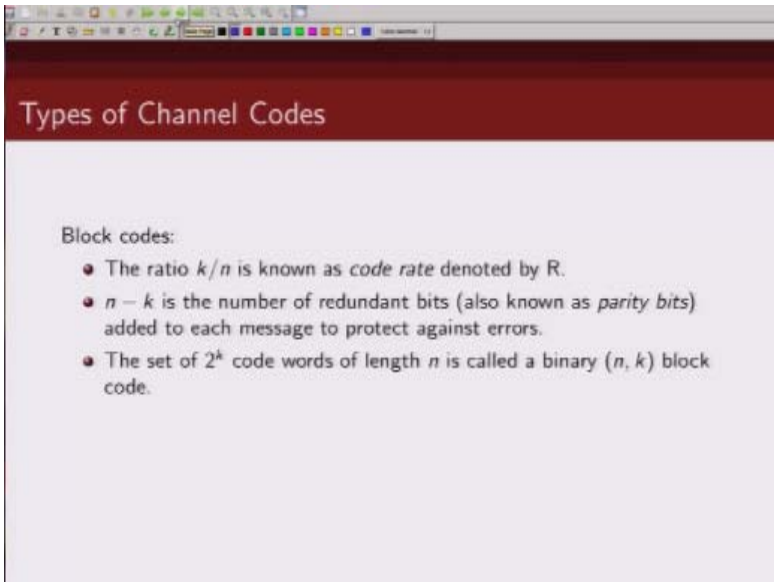
And typically denoted by R, case number of information bits n is the number of coded bits. So n- k is number of redundant bits that we are adding to our information bits and these are also known as parity bits.

(Refer Slide Time: 03:32)



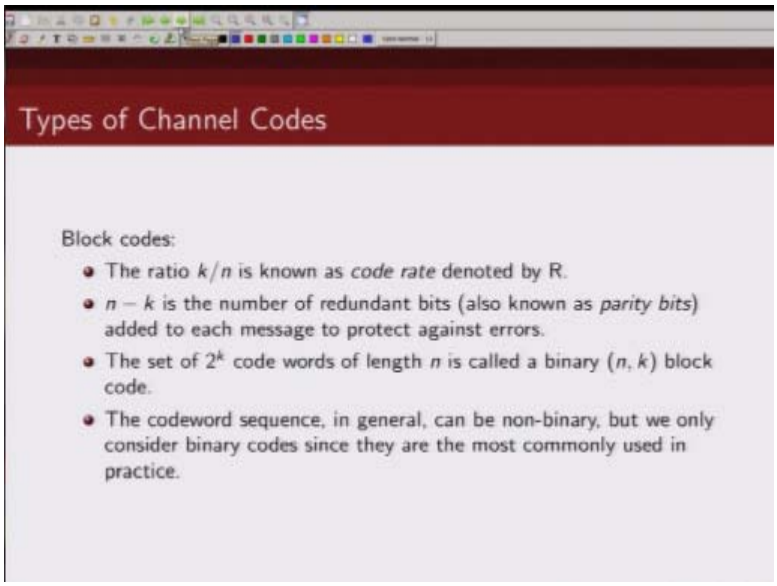
If you are considering without loss of generality we will basically considered in these set of lectures binary code words so our information sequence consists of 0's and 1's, similarly our code sequence also consists of 0's and 1's, since we are considering a block of K bits and binary code words so number of code words is basically 2^k.

(Refer Slide Time: 04:00)



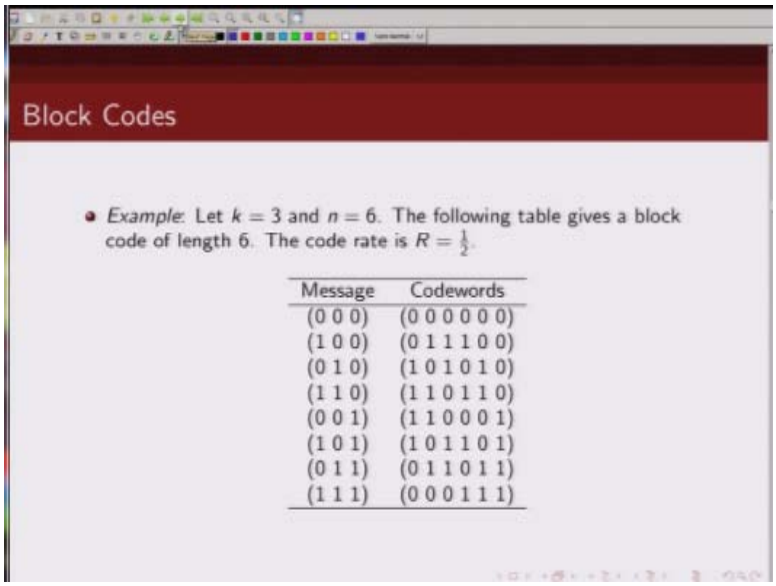
So a binary n, k block code consists of 2^k code words each of length n.

(Refer Slide Time: 04:08)



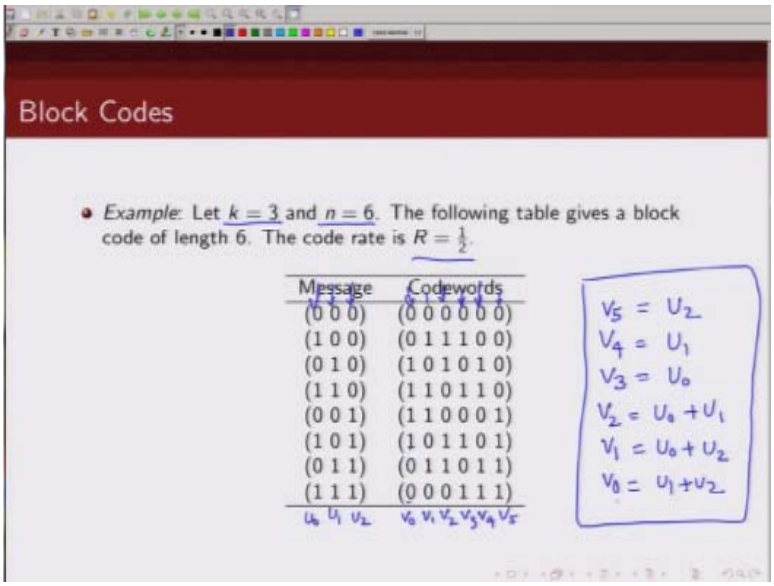
Now these code words need not be binary however it is the same theory mostly applies to non binary code words as well so we will restrict our discussion to binary code words.

(Refer Slide Time: 04:25)



So let us consider an example of a linear block code, so in this example.

(Refer Slide Time: 04:33)



Our number of information bits is 3, the number of coded bits is 6 so the code rate which is ratio of number of information bits to number of coded bits is 3/ 6 which is half. So what we have here is basically our message bits, now k is 3 that means there are 2^3 which is 8 code words and these are basically from 000 to 111 these are the 8 code words, now the message these are the 8 message bits and corresponding to these messages bits these are the eight code words.

Now 000 is mapped to all 0 sequence, 100 is mapped to 011 100 likewise other sequences have been mapped. So let us look at how we have mapped, how have we found out the message parity bits for this particular code word? So let us look at each of the columns of these code words, so let us look at this column first which is 0000 1111, so how was this column, how did we map to get this column, if you look at information bits this column is nothing but same as this information bit.

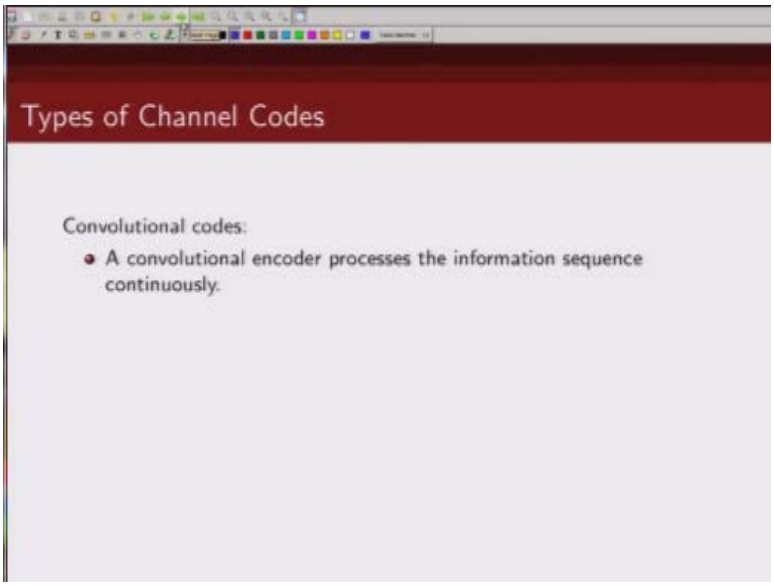
Can see 0000 1111, similarly look at this one this column is same as this column 001100 11 and this column is same as this column, so in other words this bit of the code word is same as this bit of the information sequence, this bit of the code word is same as this bit of the information sequence, this bit of the code word is same as this bit of the information sequence.

Now let us look at this one, so if we do, you exhort of these two, look at this, it gives you 0 + 0 is 0, 1 + 0 is 1, 0 + 1 is 1, 1 + 1 is 0, we are talking about binary addition over binary field so 0 + 0 is 0, 0 + 1 is 1, 1 + 0 is 1, and 1 + 1 is 0, its modeler to addition. So 1+ 1 is 0 this is 0 + 0 is 0, 1 + 0 is 1, 0 + 1 is 1 and 1 + 1 is 0. So if we let us say the denote this by u_0, u_1, u_2 and we denote these by v_0 ,v_1 ,v_2,v_3,$v_4$and v_5 what we have found out so far is v_5 is same as u_2, v_4 is same as u_1, v_3 is same as u_0 and what is v_2 ?

v_2 was u_0+ u_1, now let us look at v_1 if we look at these two u_0 + u_2 so 0 + 0 is 0, 1 + 0 is 1, 0, 0 + 0 is 0, 1 + 0 is 1,0 + 1 is 1, 1 + 1 is 0, 0 + 1 is 1 and 1 + 1 is 0, so v_1 is nothing but u_0 + u_2 okay. Now let us looks, look at last this one v_0 what is v_0, we can see that this is same as u_1 + u2 this is u_1 + u_2 so 0 + 0 is 0, 0 + 0 is 0, 1 + 0 is 1, 1 + 0 is 1, 0 + 1 is 1, 0 + 1 is 1, 1 + 1 is 0 and 1 + 1 is 0, so this is how we have map our information bits into our coded bits okay.

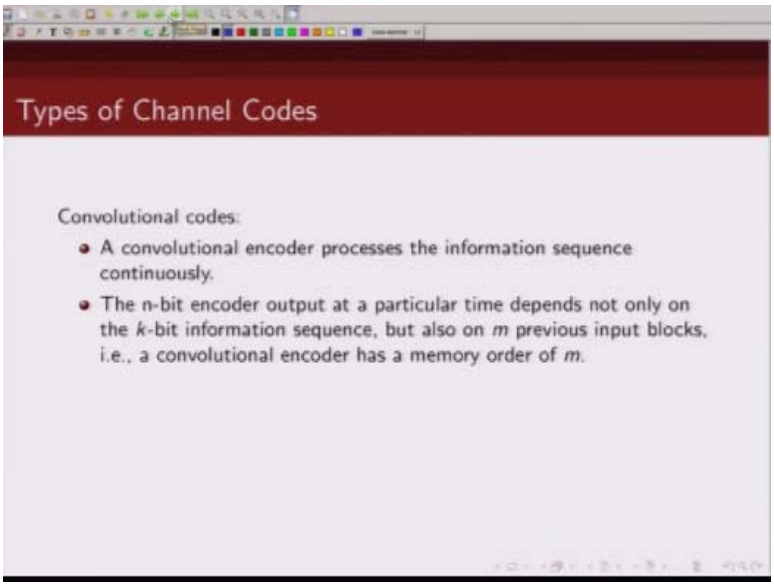
So again to recap in block codes we take, we partition our information sequence into blocks of K bits and we map these k bits into blocks of n bits and this mapping is memory less, in other words how we map these k bits does not depends on the how we have mapped the previous blocks of k bits.

(Refer Slide Time: 10:24)



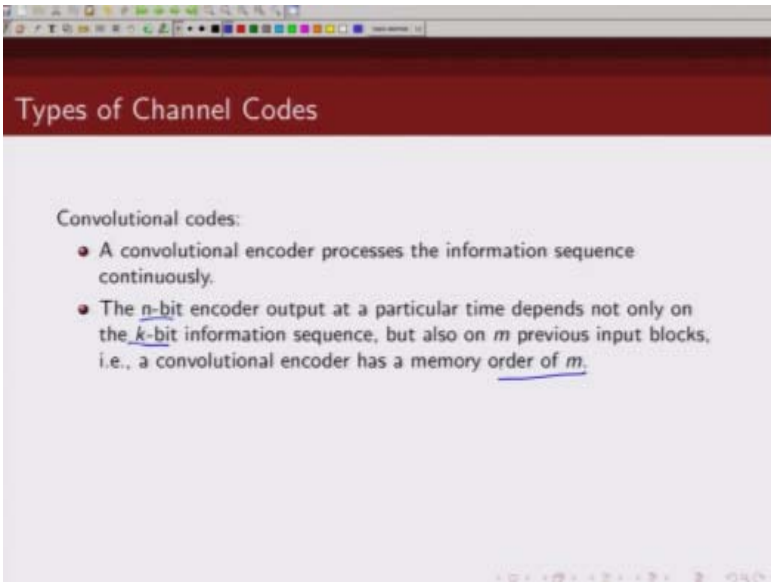
Okay so let us now contrast it with what are convolutional codes and how are they different from conventional codes? So in block codes we power for information sequence into blocks of data and we handle them block-by-block, whereas in a convolutional code you can process your information sequence in a continuous fashion.

(Refer Slide Time: 10:24)



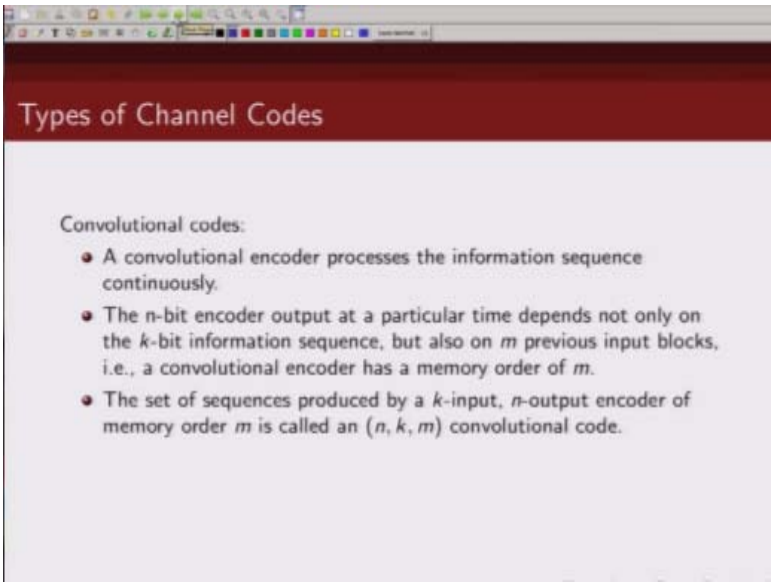
The second difference is the encoding in conditional code is with memory, in other words the current output not only depends on current input but it also depends on past inputs and outputs okay, so unlike block codes in convolutional codes output depends on past inputs and outputs.

(Refer Slide Time: 11:18)



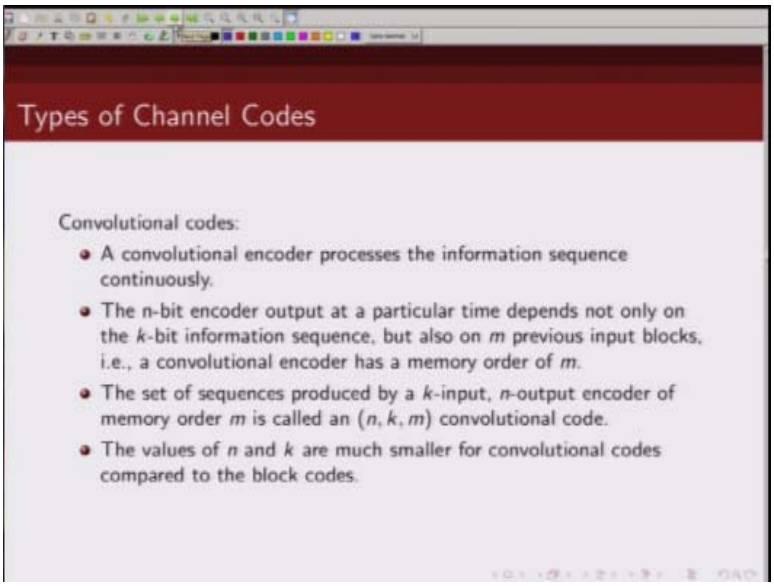
So if we have an n, k convolutional codes where k is the number of information bits, n is the number of coded bits we have another parameter if you recall our memory order which signifies basically how many past bits or how many, what is the past information that has been used to generate the current output.

(Refer Slide Time: 11:52)



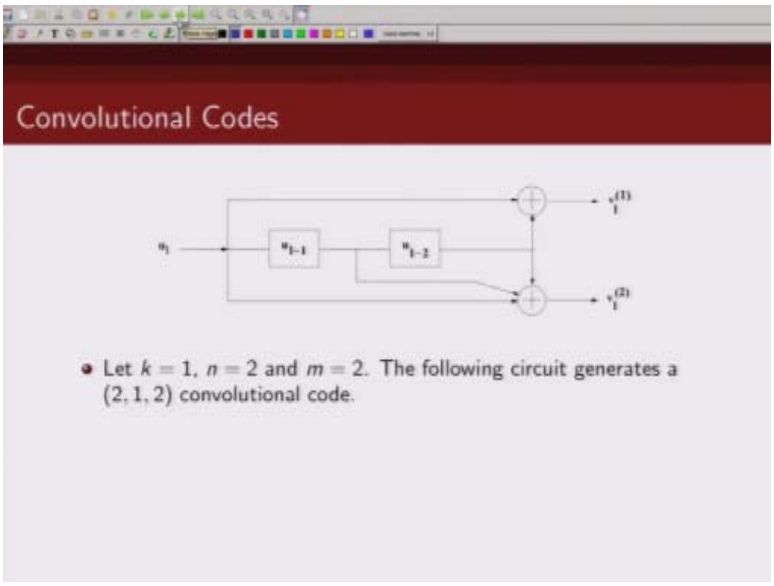
So we define a convolutional code not only by these parameters n and k but another parameter which basically denotes the memory of the encoder.

(Refer Slide Time: 11:52)



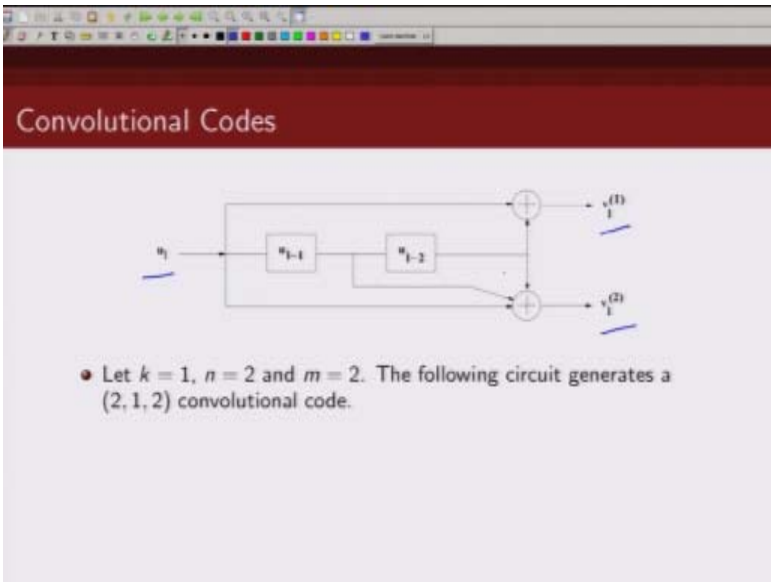
Another subtle difference in case of convolutional codes typically the values of K and n are much smaller compared two values of k and n for block codes.

(Refer Slide Time: 12:23)



So let us take an example now of a convolutional codes, so here we have

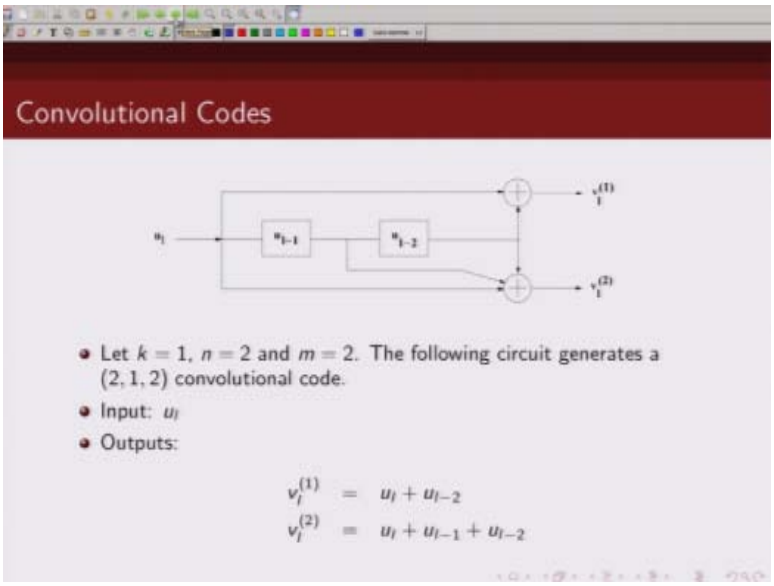
(Refer Slide Time: 12:30)



One input and two outputs, the input we are denoting by u_l output we are denoting by v_l^1 v_l^2. Now note here each of the outputs here not only depends on the current input which is u_1 but it also depends on these past values, it also depends on what u_{l-1} was what u_{l-2} was, so this is an example of memory order two.

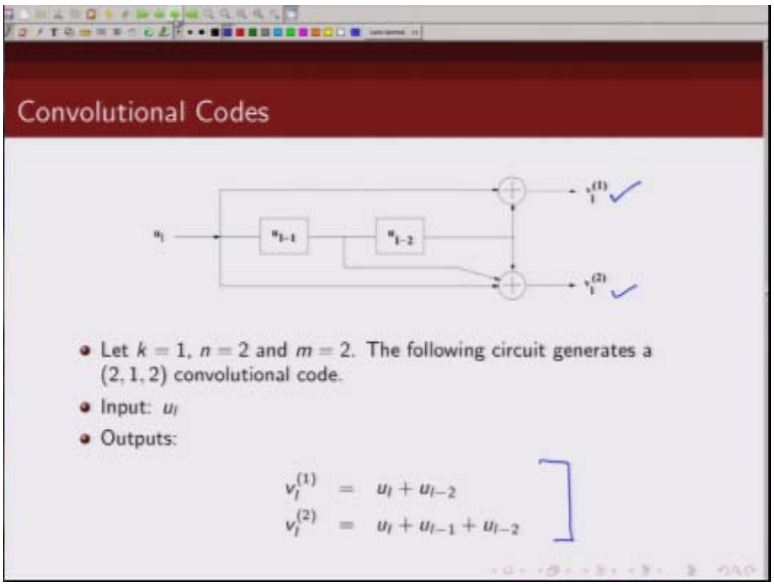
So the current input current output not only depends on current input but also depends on past two values of the input, so this is an example of a 2, 1, 2 convolutional code n is to the two outputs k is 1, one input and memory order is two because the output depends on past two values of information sequence.

(Refer Slide Time: 13:36)



So you can see here.

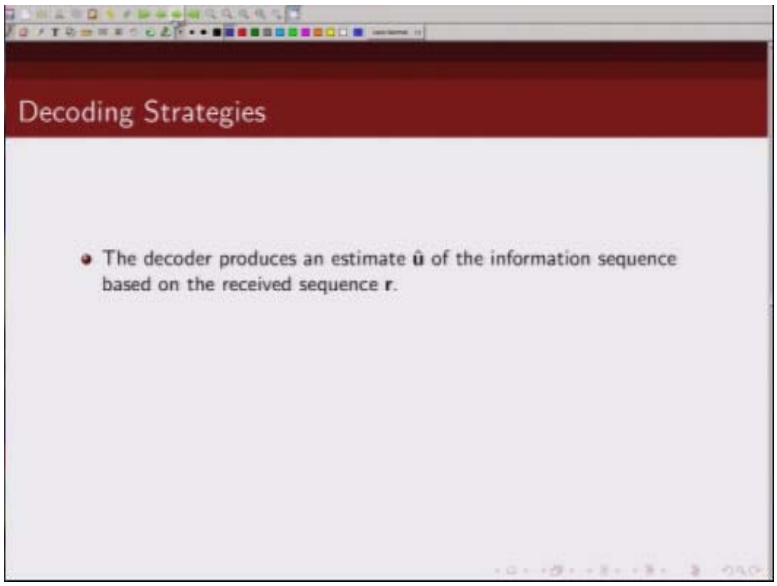
(Refer Slide Time: 13:38)



The first input which is v1, $v_l^{(1)}$ it is basically $u_l + u_{l-2,}$ see in other words it depends on the current input and what was the input, past two values basically, and similarly this one depends on current input, past input, one past input and the this u_{l-2} so this is basically how, so you can see the difference here the output not only depends on current input but it also depends on past inputs, similarly here basically can see.

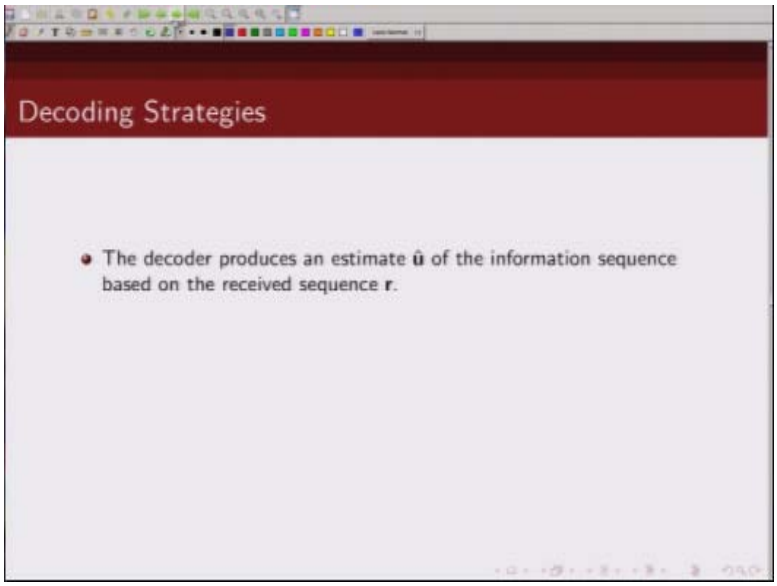
The in a convolutional code output depends on past inputs and outputs okay, so that is one of the major difference between convolutional codes and block codes.

(Refer Slide Time: 14:43)



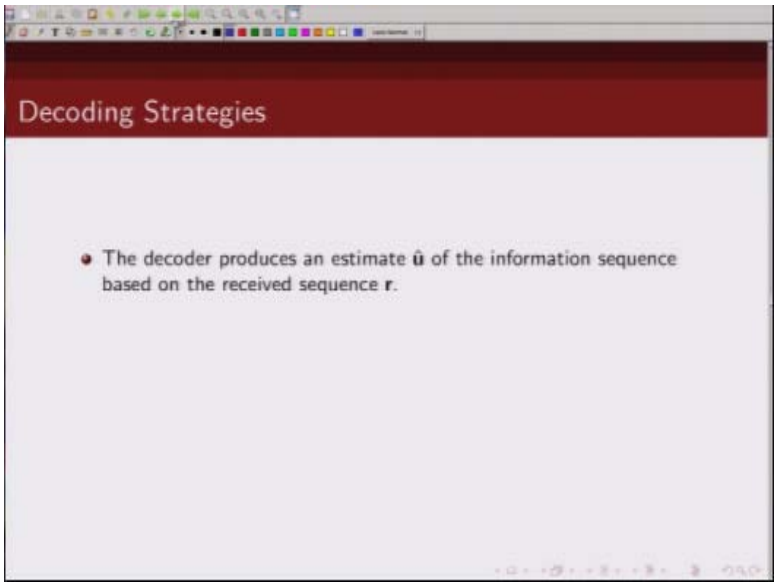
Now let us move to the topic of what sort of decoding strategy should we employee when we want to decode a code so

(Refer Slide Time: 14:58)



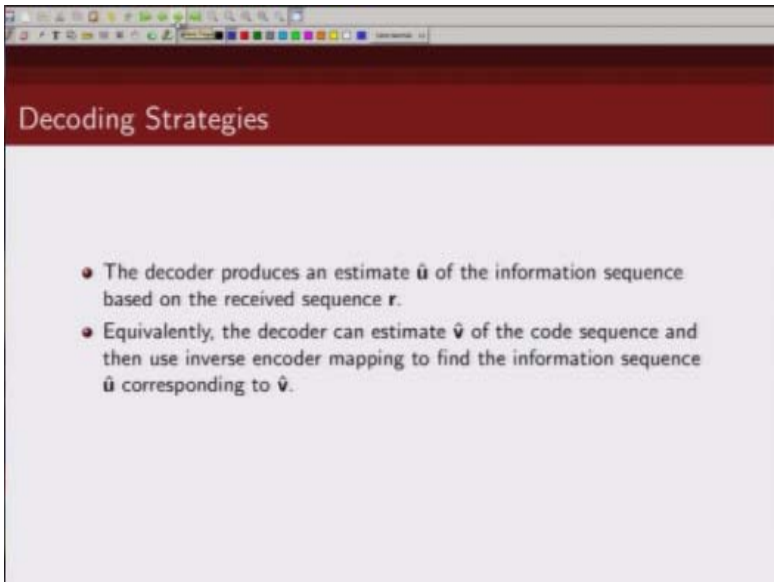
Now as I said a decoded objective is, it takes our input the demodulated signal R, and it has to produce an estimate of the information sequence û right so the decoder

(Refer Slide Time: 15:18)



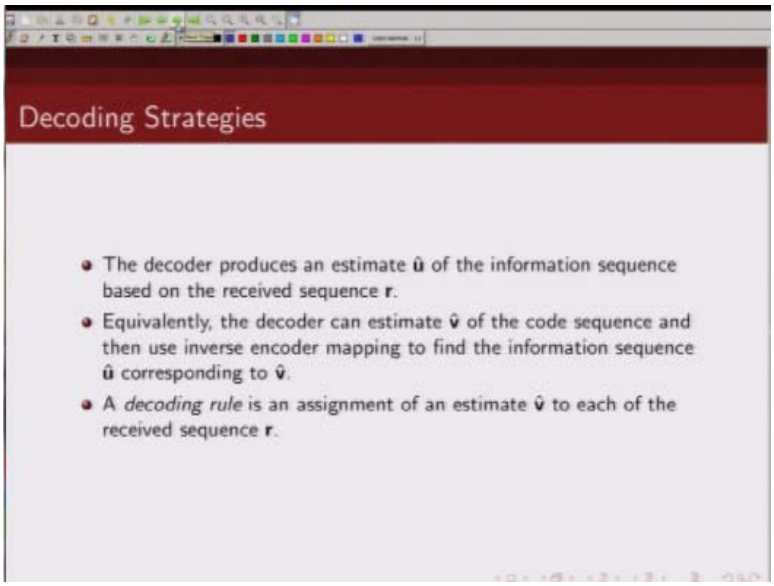
Produces an estimate of the information sequence based on what it has received, soft demodulate output which is r.

(Refer Slide Time: 15:29)



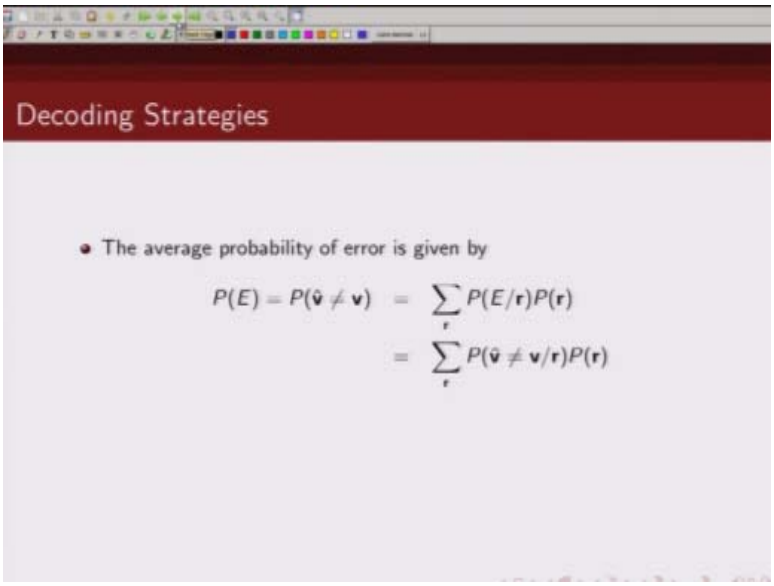
Now we can see this estimation of the information sequence problem is equivalent to estimating the code sequence because there is one to one mapping from a particular code word to the information sequence. So we can say equivalently the problem that decoder has to estimate is it has to estimate the code sequence given a receive sequence r, because there is one to one mapping from the message bits to the code bits.

(Refer Slide Time: 16:08)



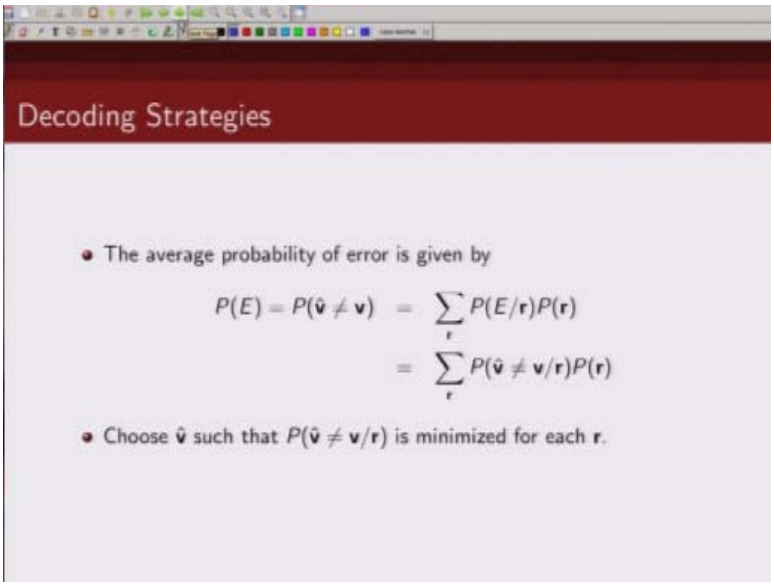
So what is a decoding strategy or what is a decoding rule? A decoding rule is nothing but given a receive sequence r we are trying to estimate what our code sequence transmit code sequence one so we are trying to estimate v′ or û from receive sequence r, so we have to decide how what rule or what logic should we use when we get receive sequence r how do we assign that receive sequence r to any particular code word.

(Refer Slide Time: 16:45)



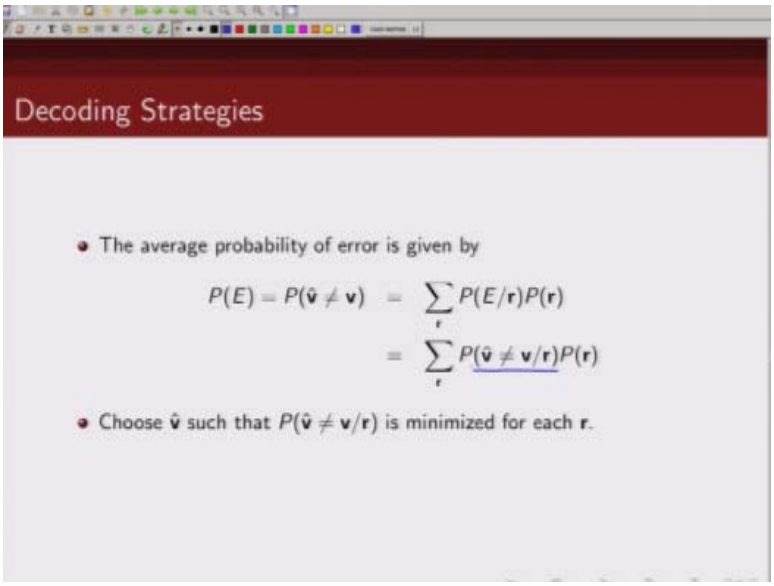
Now one of the policies which we can use is basically to minimize probability of error, now when does an error occur? When my decoded sequence is not same as my transmitter signals so my probability of error is given by probability then when my estimated sequence which I denote by v′ is not same as v, so this can be written as probability of error given r receive sequence multiplied by probability of the receive sequence r and sum over all possible receive sequence, an error is nothing but when v is not same as v′ so I can write this equation in this particular form.

(Refer Slide Time: 17:37)



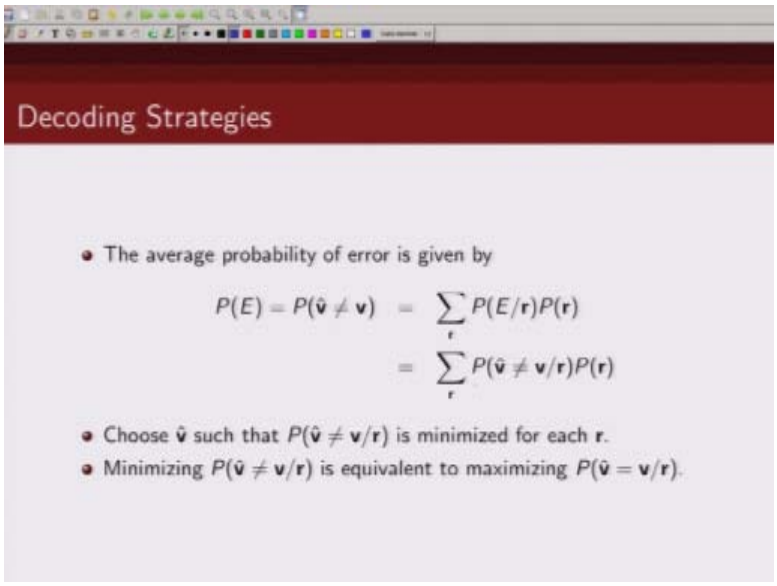
So if I want to minimize probability of error I will have to minimize this so my decoding rule should be such that this is minimized, so there are two terms in this one is P(r), and another is just term. Now whatever v′ I choose that does not change P(r) so the choice of decoding rule does not change my P(r) so in other words if I have to minimize probability of error I should choose my v′ in such a way such that.

(Refer Slide Time: 18:20)



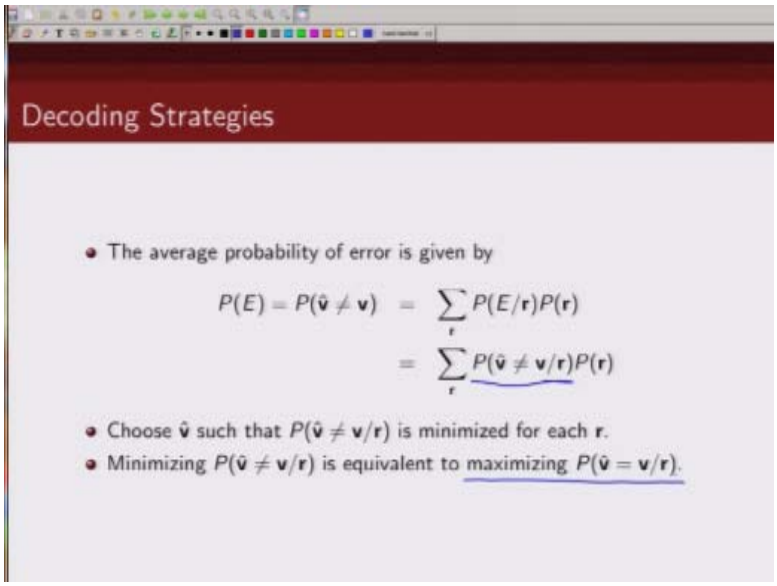
This is minimized, for each receive sequence r this term should be minimized okay.

(Refer Slide Time: 18:31)



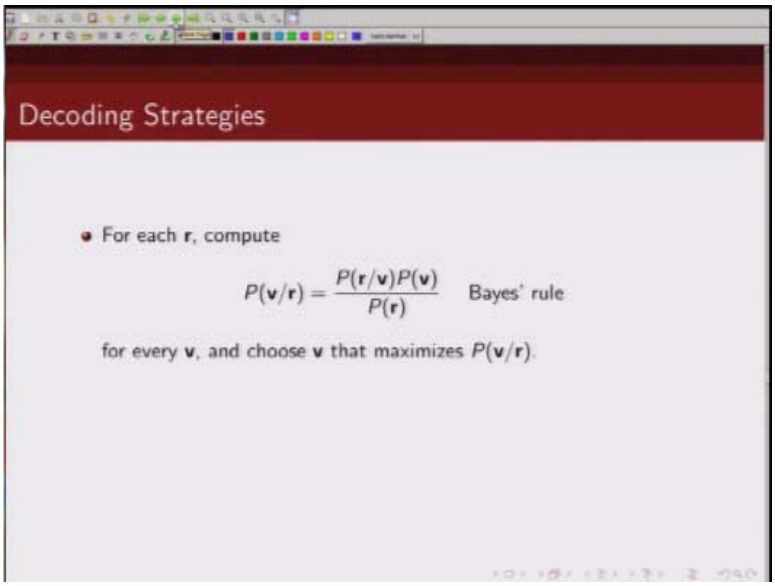
Now minimizing this term

(Refer Slide Time: 18:34)



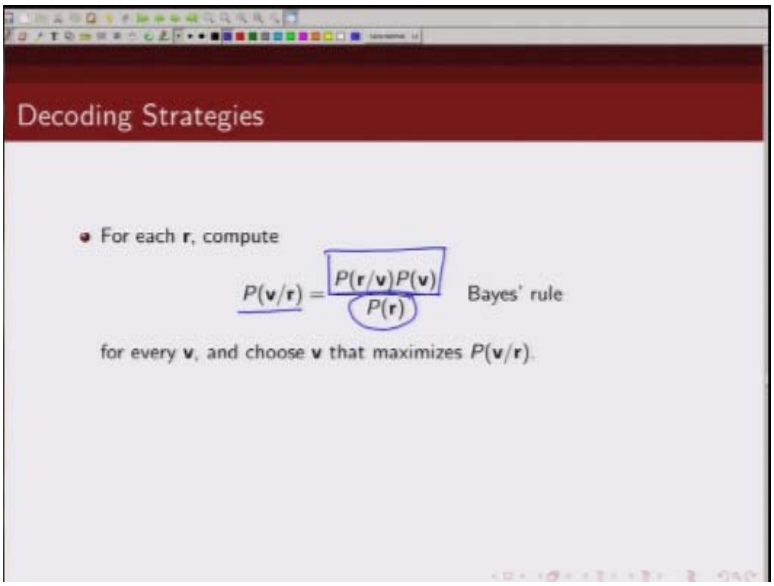
Minimizing this term is same as maximizing this term, correct. Minimizing the probability v′ is not same as v given r is equivalent to maximizing the probability that v′ is equal to v given r okay.

(Refer Slide Time: 19:00)



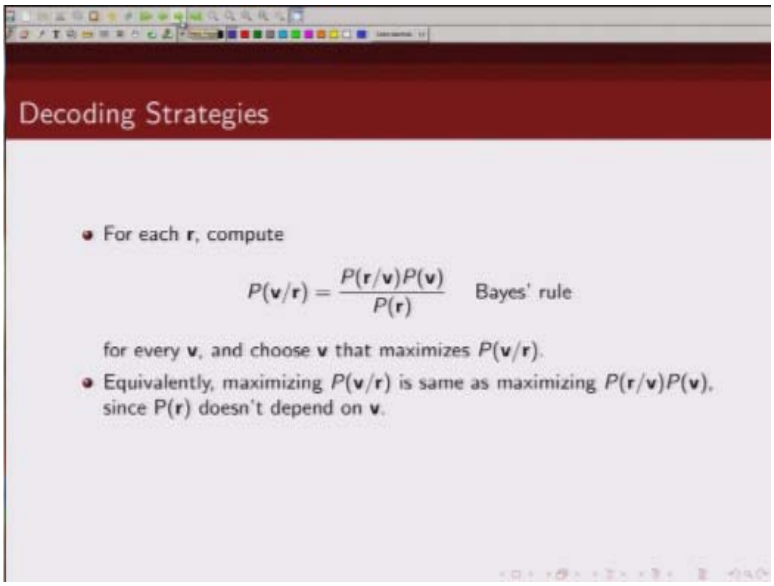
So we have to maximize this now using Bayes' rule we can write P(v/r) as P(r/\v) multiplied by P(v)/P(r)and this has to be maximized for every basically v, so we should choose our v such that this thing is maximized, now again choice of v.

(Refer Slide Time: 19:29)



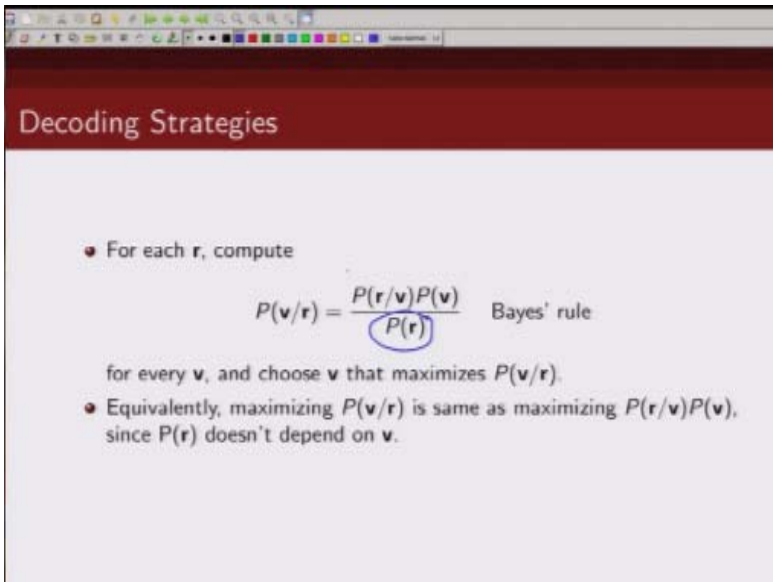
Does not change this, so we can write our probability to maximize the so to maximize this then becomes maximizing this quantity.

(Refer Slide Time: 19:48)



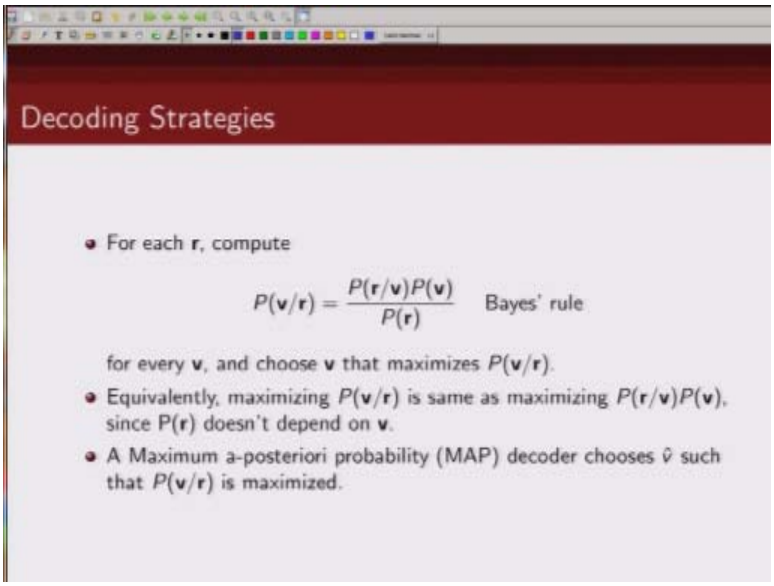
So we can say maximizing this is nothing but maximizing this quantity because

(Refer Slide Time: 19:58)



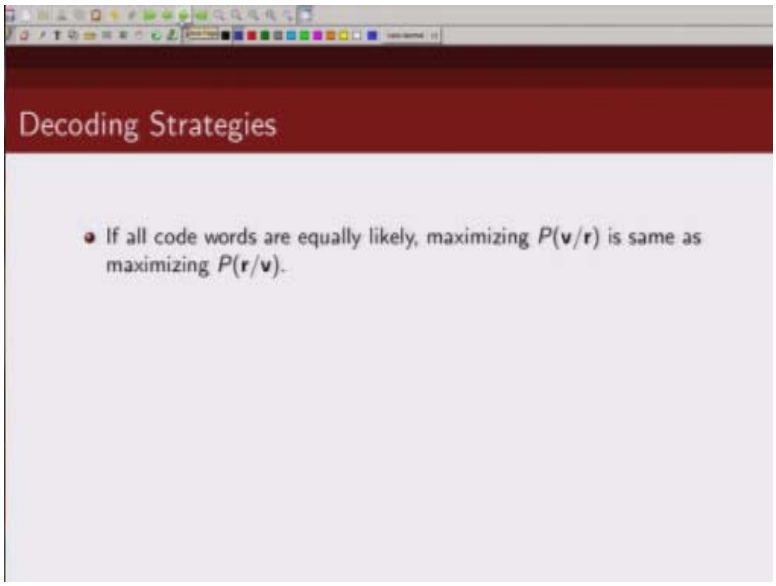
This quantity does not depend on choice of v okay, so if you want to minimize probability of error, we want to maximize this, we want to maximize this quantity

(Refer Slide Time: 20:17)



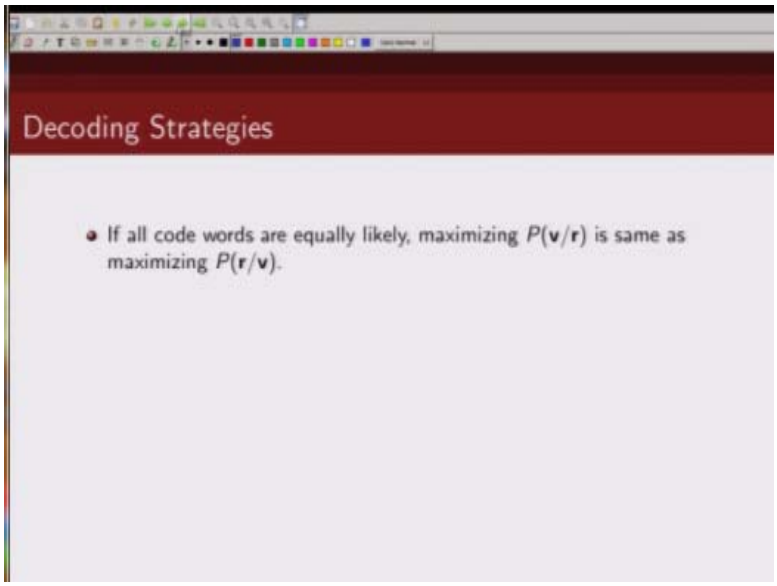
And a map decoder, a maximum a-posterior probability decoder is the one which will do exactly that so it will choose a v′ such that this is, this probability is maximized.

(Refer Slide Time: 20:34)



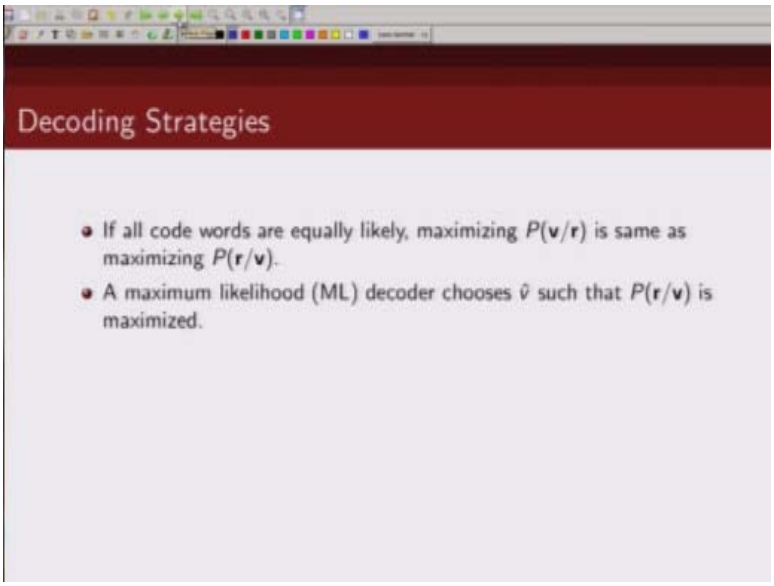
Now what happens if all code words are equally likely to happen? If all code words are equally likely to happen then look at this term, probability of V will be same so in that case maximizing probability of v given r is same as maximizing P(r) given v.

(Refer Slide Time: 20:56)



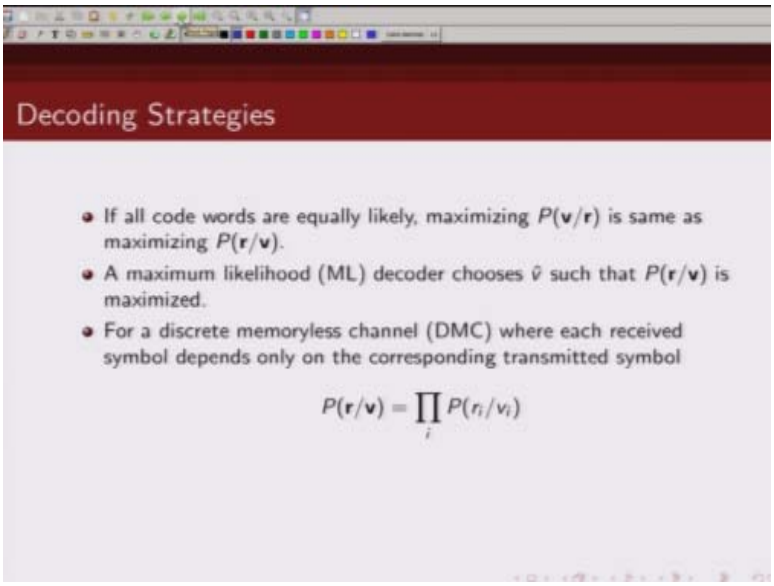
So that is what we are saying, if all code words are equally likely then maximizing P(v/r) is same as maximizing this likelihood ratio, a likelihood function P(r/v)

(Refer Slide Time: 21:13)



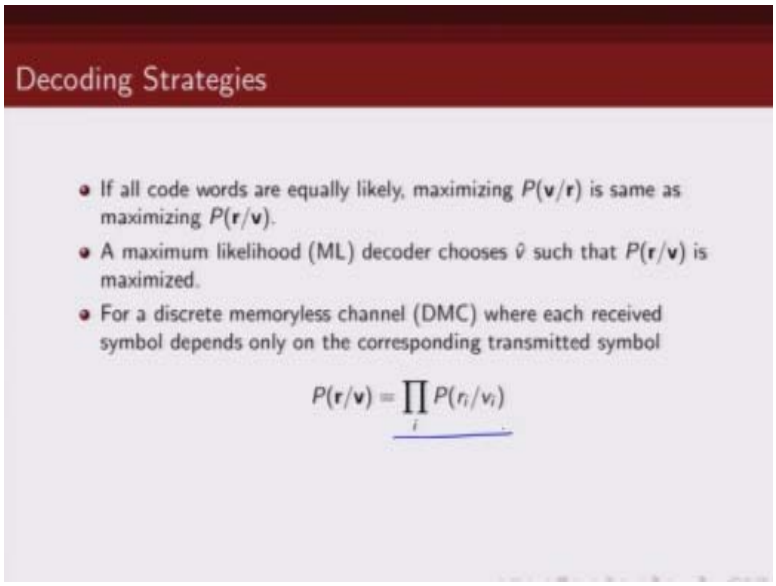
So a maximum likelihood decoder is the one which will choose v′ such that this quantity if maximized.

(Refer Slide Time: 21:25)



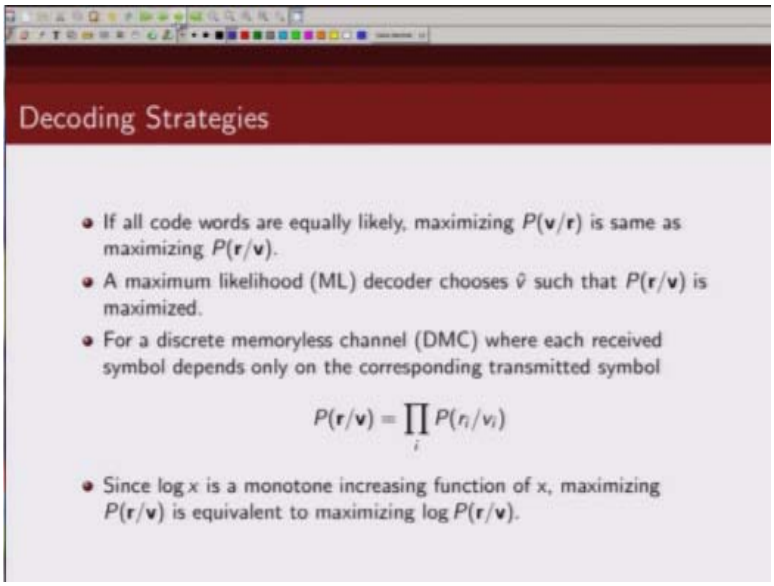
Now if we consider that our channel is discreet memory less channel, in other words we can write the probability for a discreet memory less channel we can write probability of receive sequence r given transmit sequence v we can write it as product.

(Refer Slide Time: 21:44)



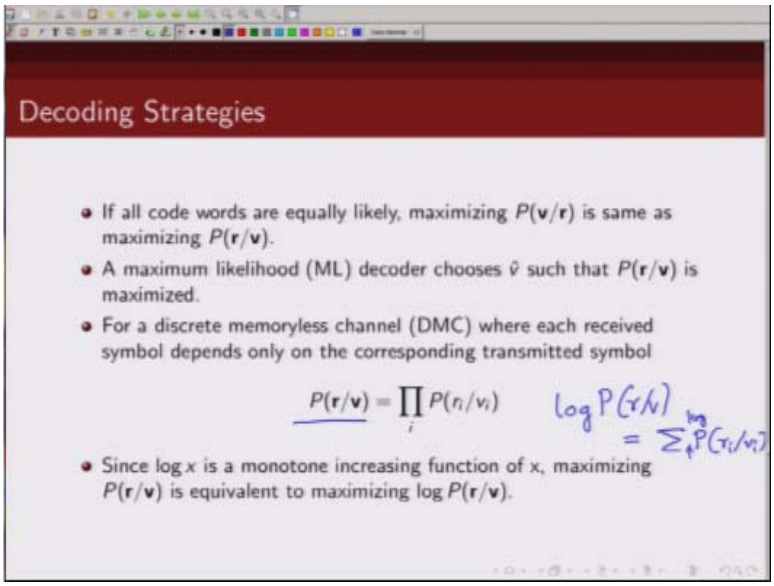
Of these individual probabilities, so if that happens

(Refer Slide Time: 21:50)



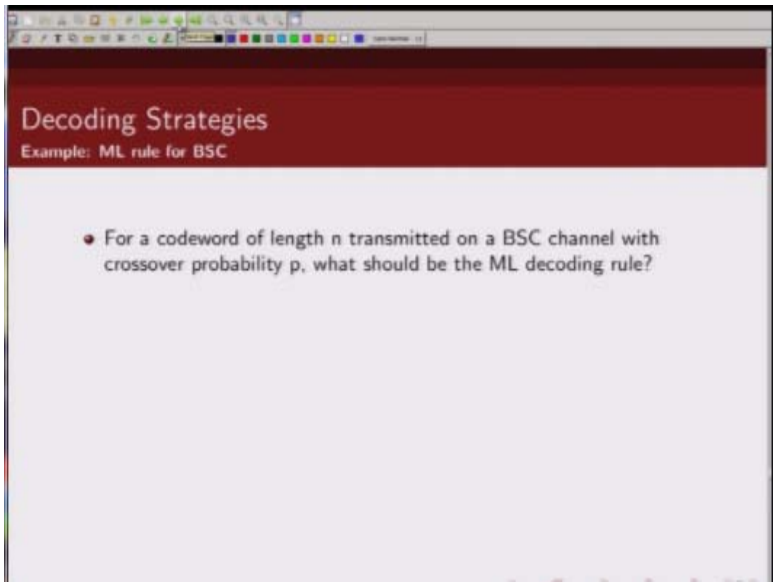
Then we can further simplify our maximizing criteria, so you want to maximize this is same as maximizing this, now since log of x is a monotone increasing function effects we can say maximizing.

(Refer Slide Time: 22:11)



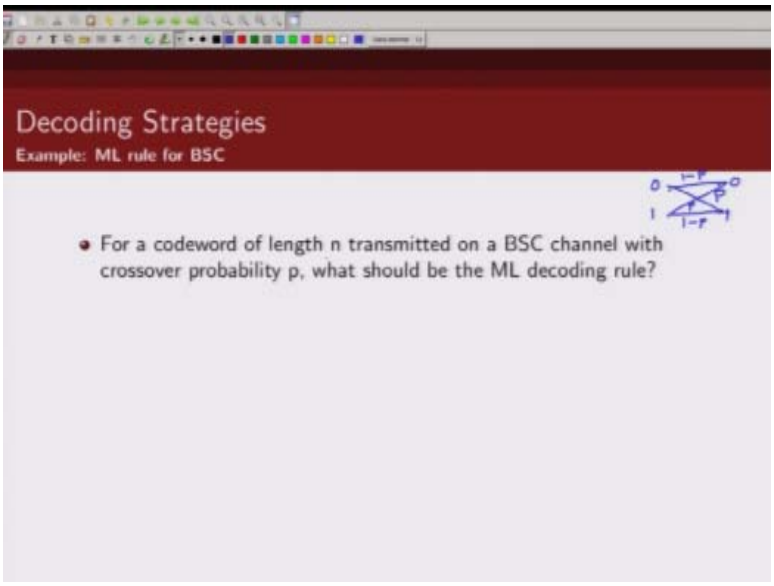
This probability is same, is equivalent to maximizing log P(r/v). Now if we do that then this product becomes Σ okay so then we can basically write this as basically then log P(r/v) will become basically Σ and this will be basically of course this will there will be some log term here, log term here, and this is basically much easier to compute okay.

(Refer Slide Time: 22:53)



So let us take an example, we are interested in finding what would be the maximum likelihood decoding rule for a binary symmetric channel? Now recall what is a binary symmetric channel?

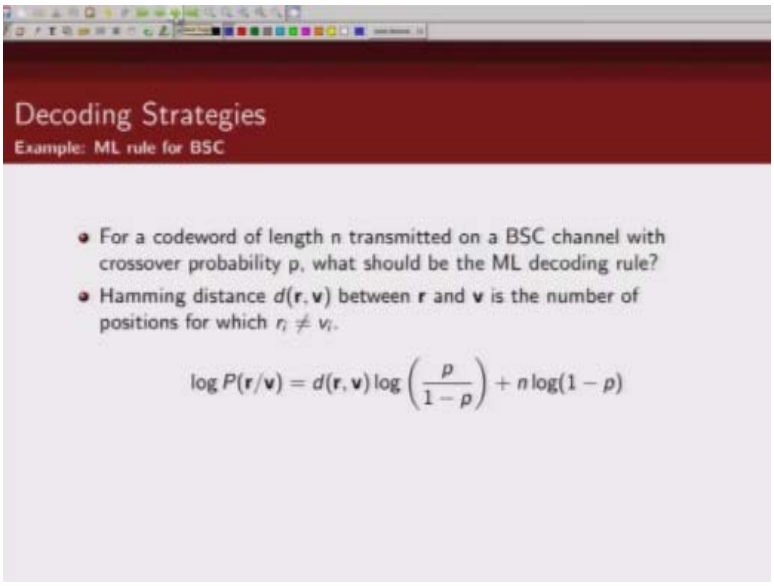
(Refer Slide Time: 23:10)



There are two inputs this is 0 and 1, two outputs 0 and 1 with probability 1 –P. I receive my bits correctly and there is a crossover probability of P, okay so the question I am asking is if I have a code word of length n which is transmitted over a binary symmetric channel whose crossover probability is P what should be my maximum likelihood decoding rule?

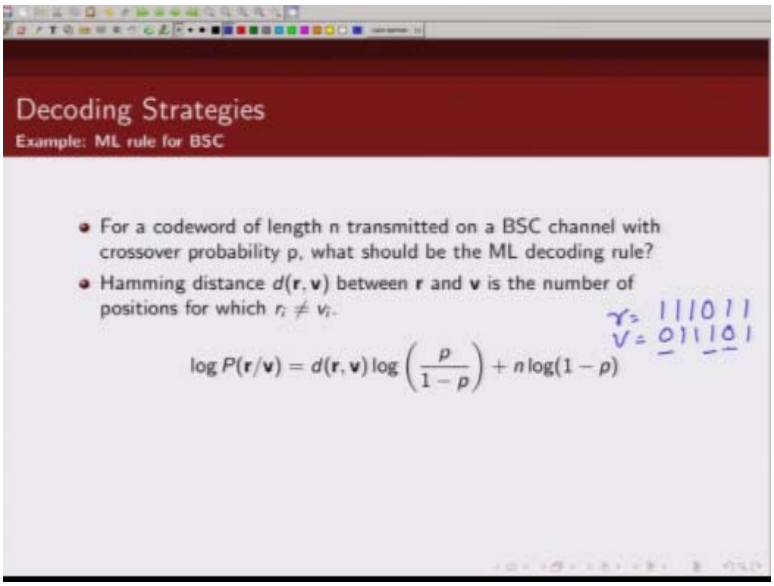
So how do I solve it, as we just saw in the previous slide maximizing probability of art, for maximum likelihood decoder we have to maximize the probability of P(r/v) which is equivalent to maximizing log of P(r/v), so let us try to come, compute what is log (r/v) okay.

(Refer Slide Time: 24:23)



Now before I calculate P(r/v) let me introduce another term which is called Hamming distance. Now what is hamming distance between two code words? So hamming distance between two code words or two end tupples, let us call it hamming distance between r and v both are n bit vector basically so hamming distance between r and v is defined as number of positions in which r and v are differing.

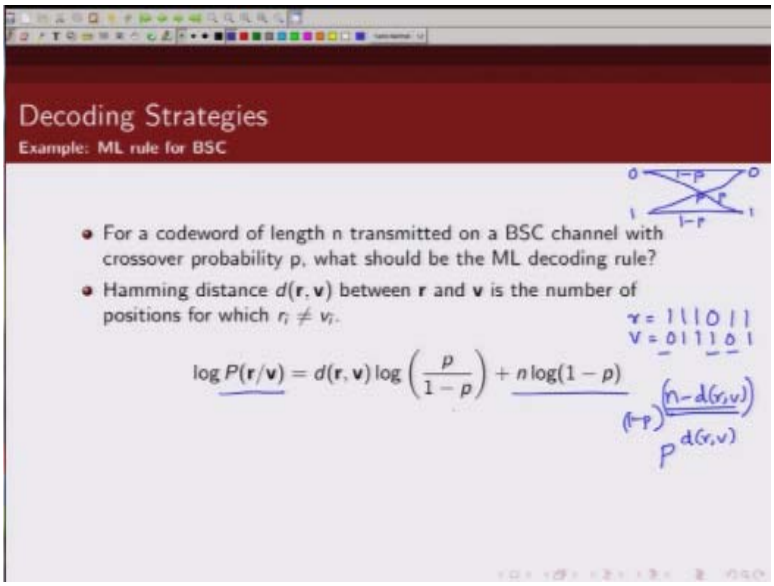
(Refer Slide Time: 24:58)



So for example if let us say r is 111011 and v is 011101 then what is the Hamming distance? It is differing in the first location one it is not differing here, not differing here, it is differing here that is two, it is differing in this location that is three, it is not differing in this location, so r and v differs in three locations, one is this location, other is this location and third is this location, so the Hamming distance between r and v is three in this case okay.

Now when we are sending an n-bit code word over a binary symmetric channel what happens, some of the bits will get flipped with probability crossover probability P let us denote those number of flip bits by d.

(Refer Slide Time: 25:58)



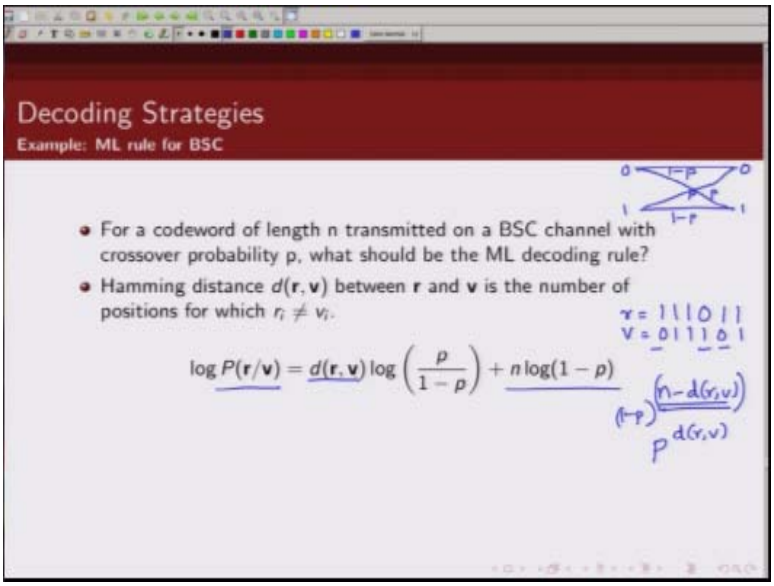
So hamming distance between two r and v will specify the locations where r and v are not same and when r and v is not same that means those are locations where error have occurred so number of positions that got flipped as a result of sending this code word or binary symmetric channel that is denoted by this d(r, v) and the remaining number of bits which did not get changed that is basically n–d(r, v) so these many bits did not get changed and these many bits got flipped.

So what is the probability that d bits got flipped that is given by $P^{d(r, v)}$ and what's the probability that n –d bits were received correctly that is given by 1 –P raised to the power this quantity. So we can write P(r/v) as $P^{d} x1\text{-}^{n-d,}$ now if we take log on both sides then this will basically become n – d log(1-P)+D times log(P). Now we take terms containing d r v out so what we will get is d r v log(p)/ 1-P+ n times log(1-P).

So to maximize this probability we have to choose our v′ such that this is maximized. Now look closely at both of these terms, let us first look at this term does this term depend on selection of v, no it depends on n which is code word length, it depends on cross of probability P, so whatever v we choose it does not change this probability, so in other words to maximize this then we will have to maximize this first term.

Now look at this term closely, typically the crossover probability will be smaller then half, if that happens.

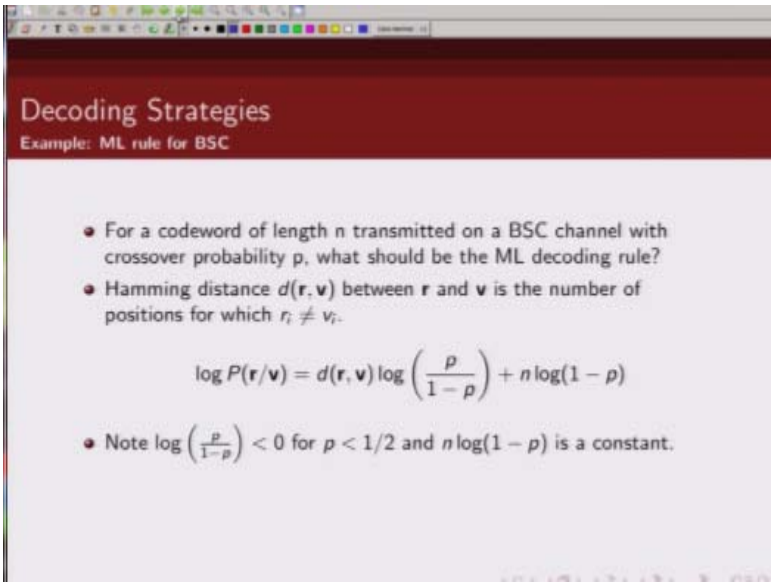
(Refer Slide Time: 29:01)



What happens to this ratio P/1-P this will be sum ratio between 0 and 1 and what happens to log of a number which is between zero and one, that is a negative quantity so what we get then is to maximize this we have to maximize – d (r, v) correct. So a maximum likelihood decoder will choose a v such that –d(r, v) is maximized, in other words we should choose a code word v in such a way such that d(r, v) is minimized.

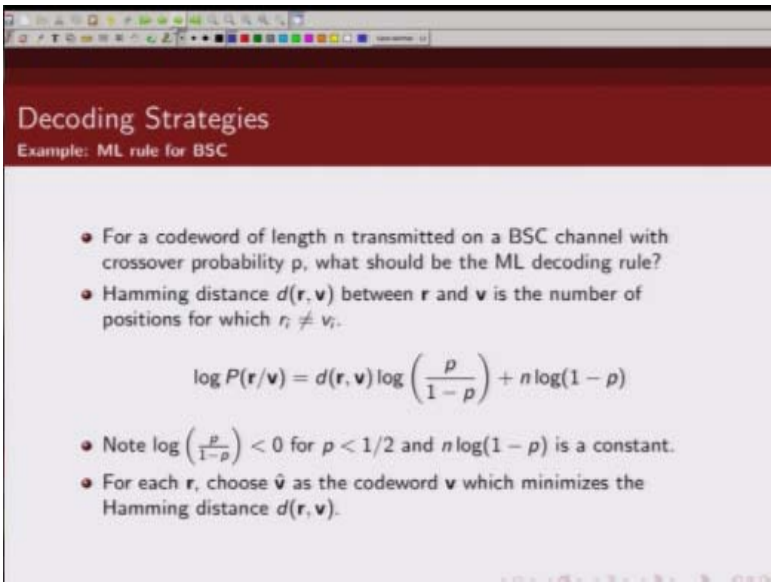
When d (r, v) is minimized then only-d(r, v) will be maximized.

(Refer Slide Time: 30:00)



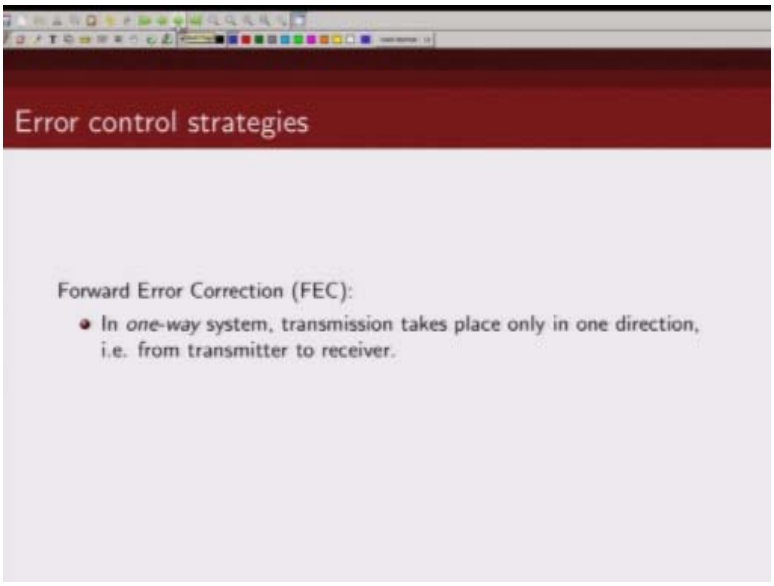
So that is what we are seeing here log(P1-P) < 0 so this will be a negative quantity, when you want to maximize a negative quantity this term should be as small as possible and this term does not depend on selection of v.

(Refer Slide Time: 30:20)



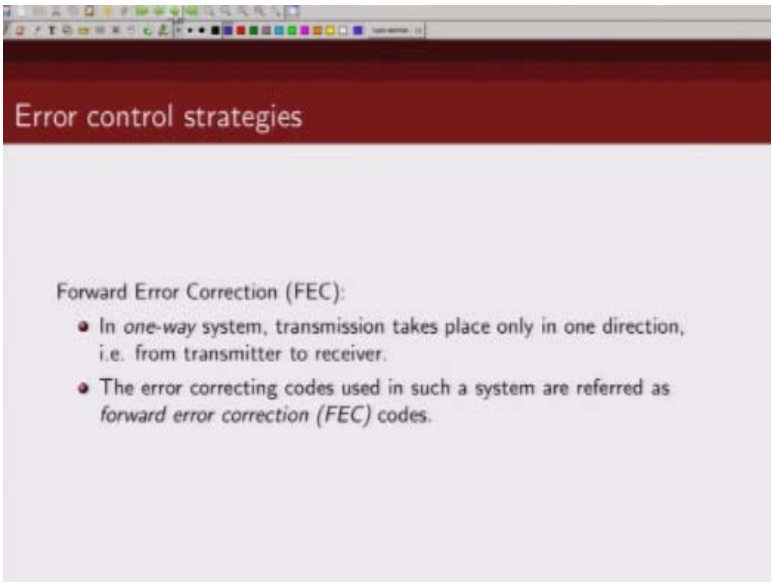
So this gives us a decoding maximum likelihood decoding rule for binary symmetric channel and what is that? We should choose a v such that d(r, v) is minimized, in other words we should choose a code word v such that hamming distance between the code word v and the receive sequence is minimized and that makes sense and that is our maximum likelihood decoding rule.

(Refer Slide Time: 30:55)



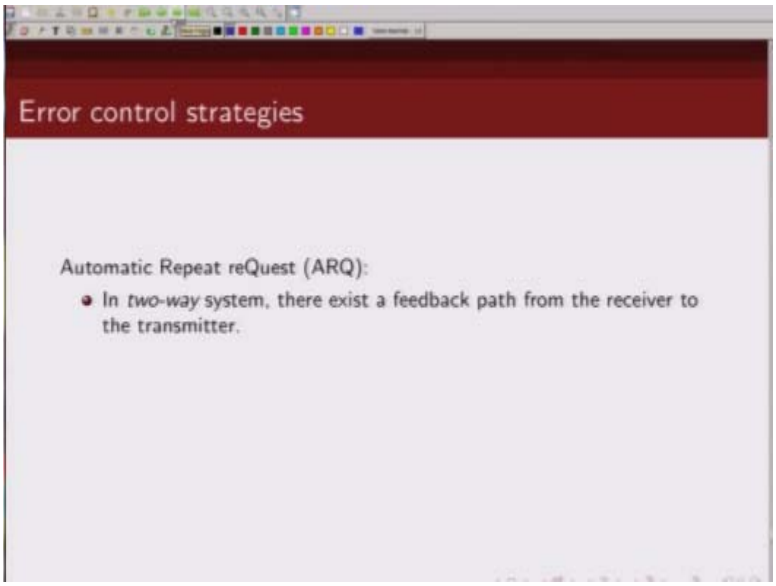
So finally I am going to conclude this lecture with definition of few error correcting strategies. The first one which I am going to describe is what is known as FEC Forward Error Correction, so in systems where there is no feedback from the receiver to the transmitter we are calling those systems as one-way system where transmission happens only in one direction from transmitter to receiver. In those systems.

(Refer Slide Time: 31:32)



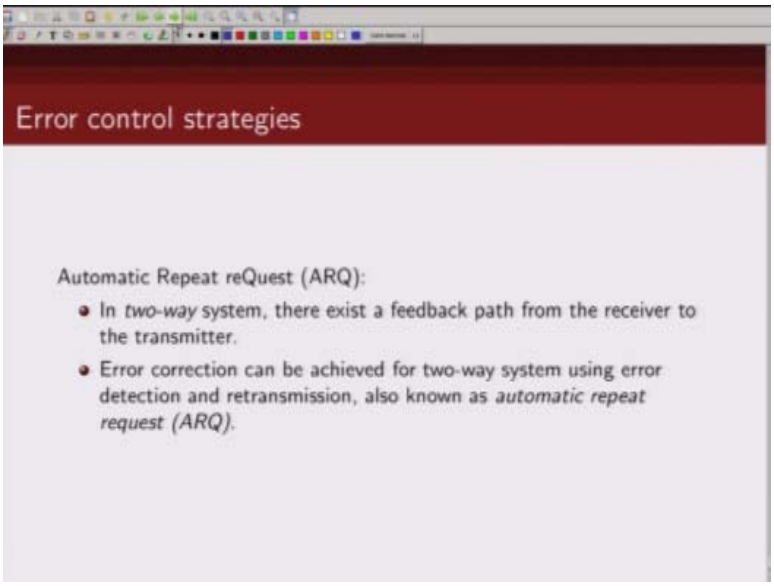
The error-correcting codes that are used are known as FEC, so when you hear this term FEC code it basically means basically when we are sending, so this is the error correcting code used for when we are using transmission one way from transmitter to receiver. Now in some cases we have a mechanism of feedback from the receiver to back to the transmitter.

(Refer Slide Time: 32:05)



So in those cases where there exists a feedback from the receiver to transmitter we are calling this these systems as two-way systems.

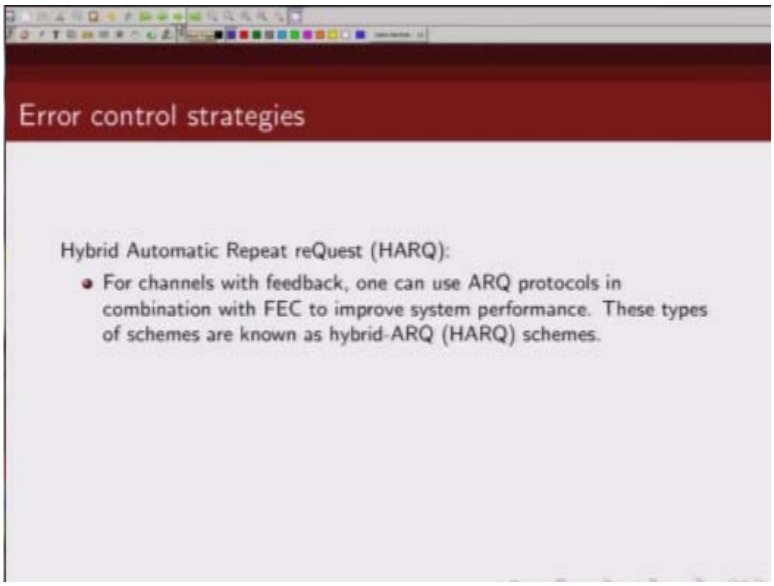
(Refer Slide Time: 32:16)



So for these systems basically the error-correcting strategy which is used is what is known as automated repeat request. Now how does this works? So use initially sent some coded packets where you just use parity bits for error detection. So you send your information bits and some true parity bits for error detection at the receiver using those parity bits the receiver will try to judge whether there is any error in the received packet, if it finds that there are errors it will send a negative acknowledgement and again you will retransmit the same packet or some additional parity bits.

So that is basically same packet basically so that is your automatic repeat request scheme. Now in this automatic repeat request scheme the idea is you are sending an uncoded packet with some bits for error detection so you are not really sending any bits for error correction, so only so this is typically useful if the links are very good, you just are sending uncoded packet with some bits for error deduction and occasionally when the packers are not received correctly then you ask for retransmission.

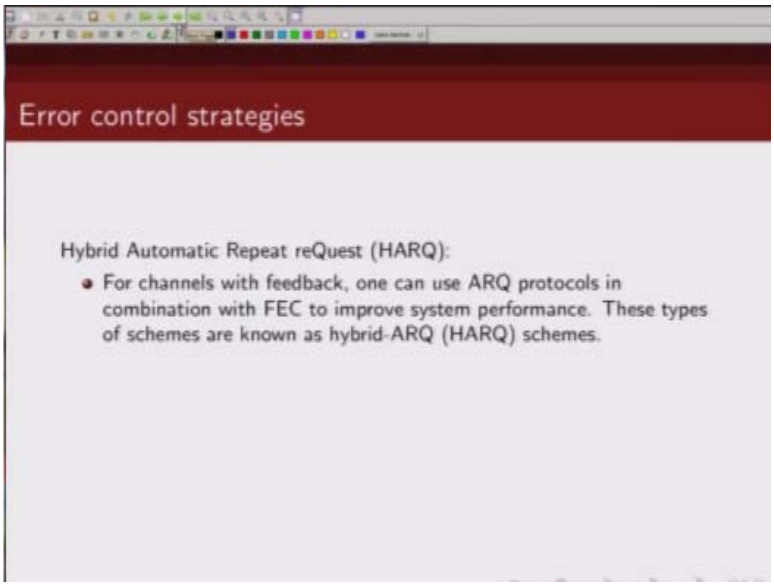
(Refer Slide Time: 32:16)



A strategy that combines both forward error correction and ARQ is known as hybrid ARQ system. In this you send coded packets from transmitter to receiver, now if these coded packets are not received correctly by the receiver the receiver will basically send a negative acknowledgement and then you will send -- resend the same packet or you will try to send some additional parity bits and using those additional bits you will try to now decode the original packet.

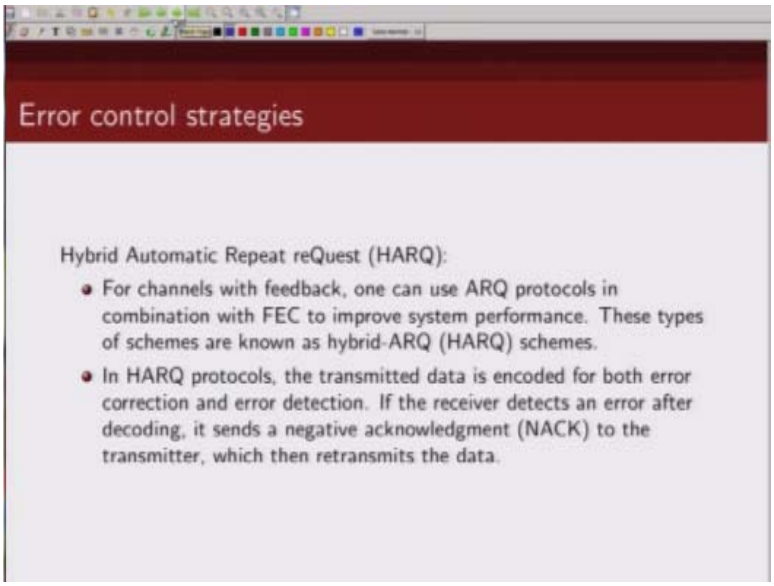
So hybrid ARQ is the combination of forward error correcting scheme and automatic repeat request scheme.

(Refer Slide Time: 34:38)



Typically it is a

(Refer Slide Time: 34:38)



In a communication system you will see a combination of what forward error correct schemes and hybrid ARQ scheme used. So with this I am going to conclude this lecture, thank you.

Acknowledgement
Ministry of Human Resource & Development

Prof. Satyaki Roy
Co-ordinator, NPTEL IIT Kanpur

NPTEL Team
Sanjay Pal
Ashish Singh
Badal Pradhan
Tapobrata Das
Ram Chandra
Dilip Tripathi
Manoj Shrivastava
Padam Shukla
Sanjay Mishra
Shubham Rawat
Shikha Gupta
K. K. Mishra
Aradhana Singh
Sweta

Ashutosh Gairola
Dilip Katiyar
Sharwan
Hari Ram
Bhadra Rao
Puneet Kumar Bajpai
Lalty Dutta
Ajay Kanaujia
Shivendra Kumar Tiwari

an IIT Kanpur Production