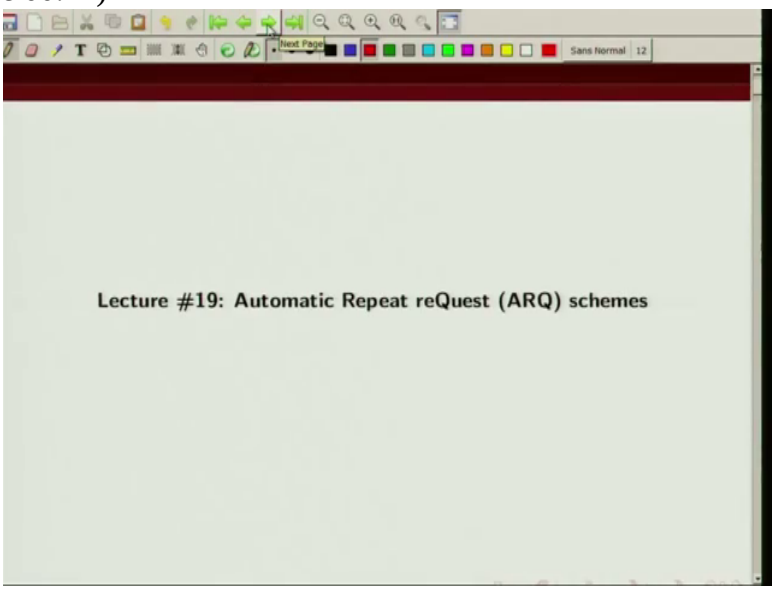
An Introduction to Coding Theory
Professor Adrish Banerji
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
Module 08
Lecture Number 32
Automatic Repeat reQuest (ARQ) schemes

(Refer Slide Time 00:13)

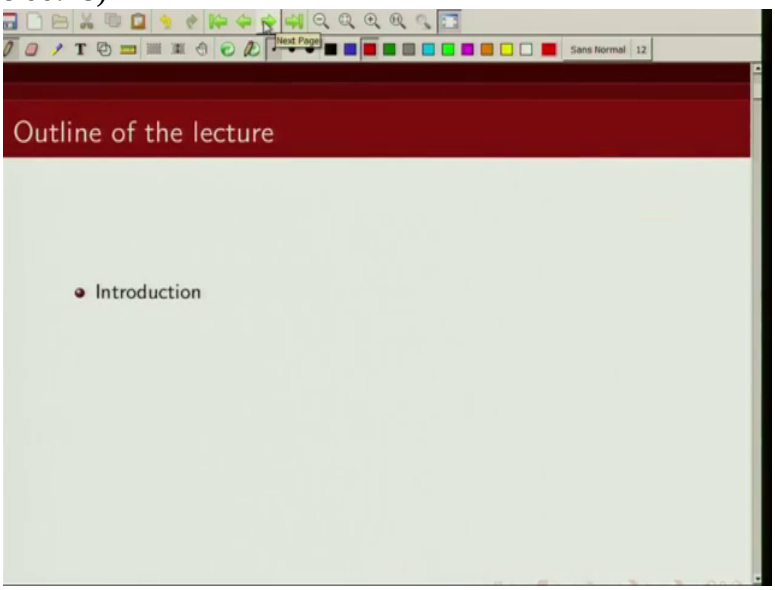
Welcome to the course on An Introduction to Coding Theory. I am Adrish Banerji. And today we are going to talk

(Refer Slide Time 00:22)



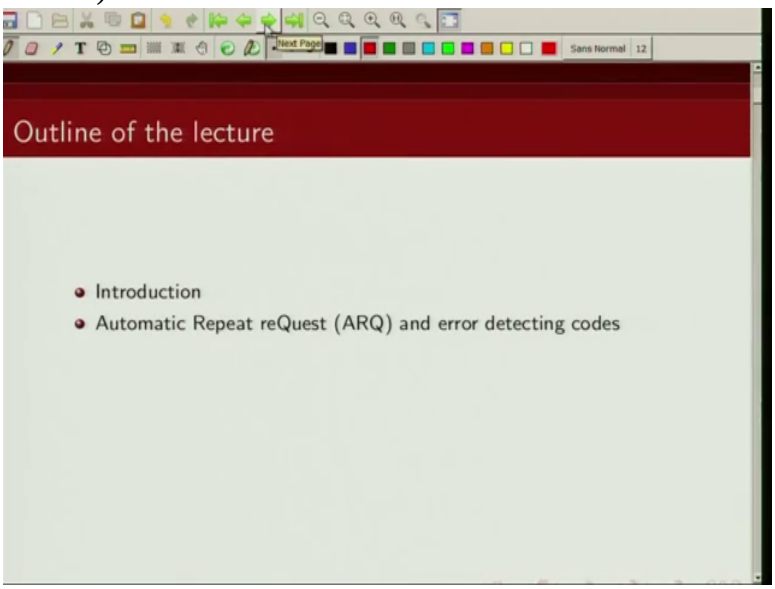
about Automatic Repeat Request schemes.

(Refer Slide Time 00:28)



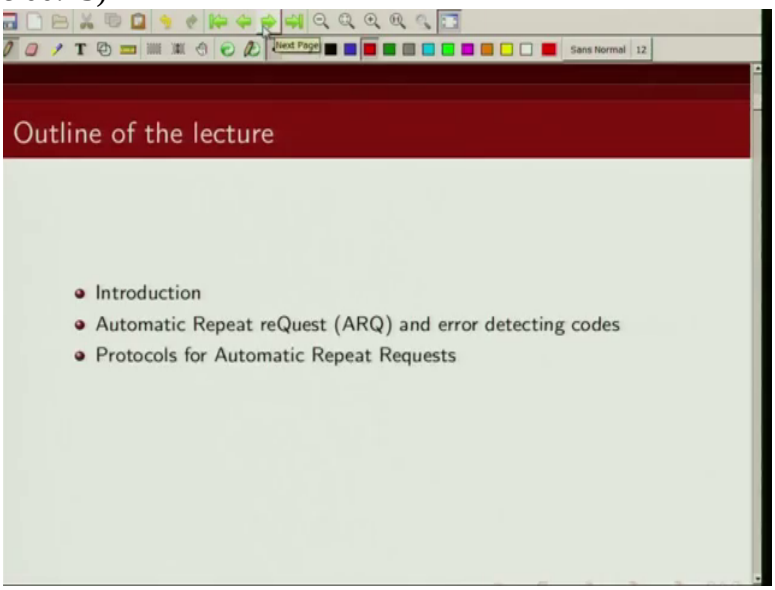
So this is the outline of the talk today. I am first going to briefly describe what is the A R Q scheme

(Refer Slide Time 00:37)



and then I will talk about the role of A R Q codes in A R Q schemes.

(Refer Slide Time 00:45)

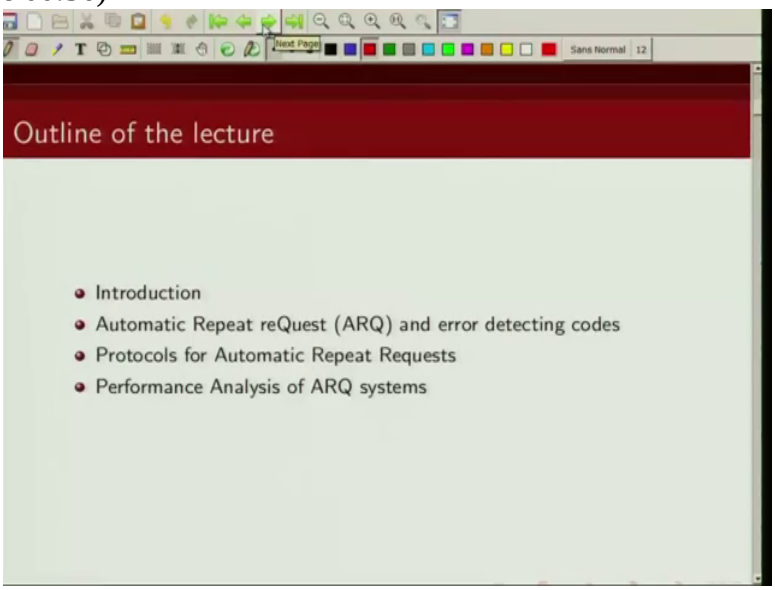


Subsequently I will be talking about 3 different protocols for

(Refer Slide Time 00:51)

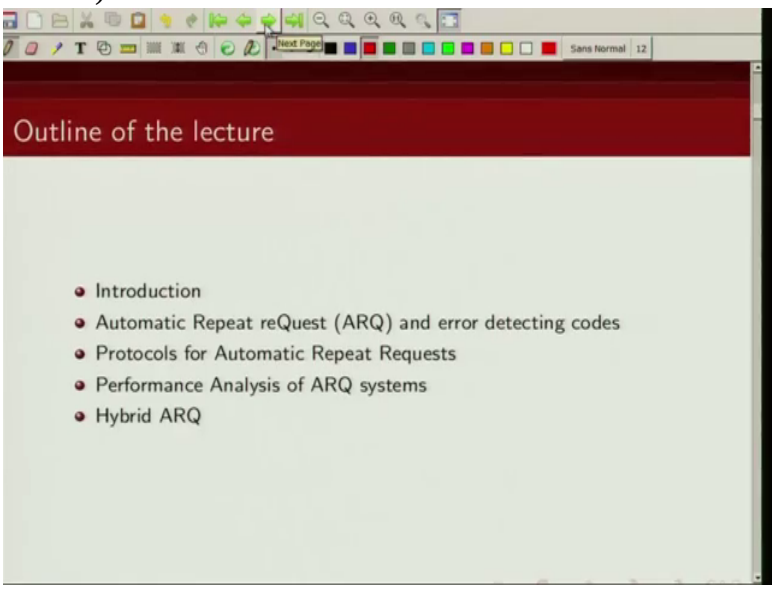
retransmission that we are going to use. And then I am

(Refer Slide Time 00:56)



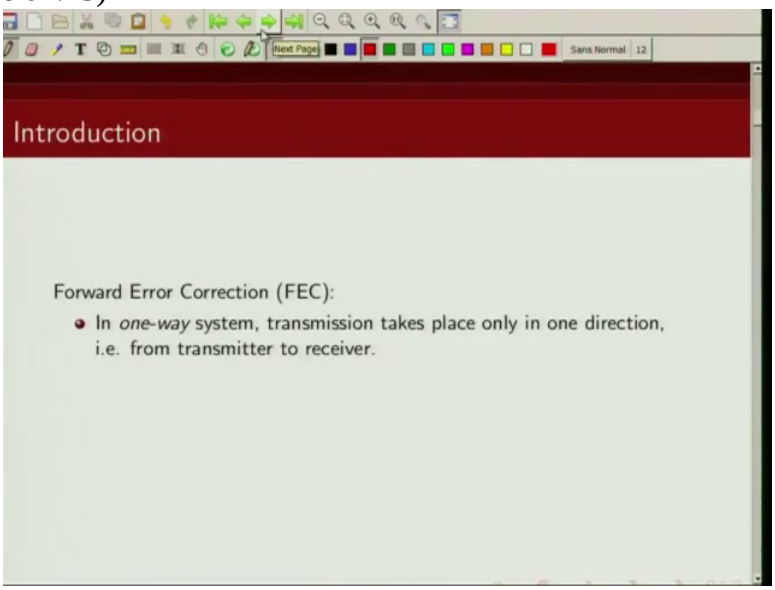
going to analyze the performance of those 3 protocols and I am going to analyze in terms of efficiency of those protocols as well as reliability of those A R Q protocols

(Refer Slide Time 01:11)



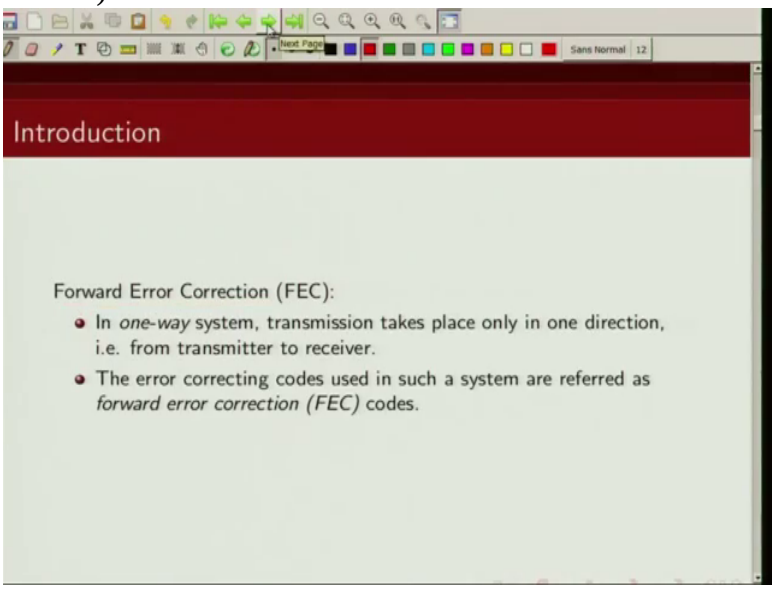
and finally I am going to talk about hybrid A R Q which is the combination of forward error correction and automatic Repeat Request schemes.

(Refer Slide Time 01:23)



So if you recall in the first lecture we talked about there are two possible strategies for error correction. One is forward error correction. So channels which are one way, so for example there is only one way communication from transmitter to receiver; so in those schemes, in those kinds of channels we used forward error correction schemes. Now

(Refer Slide Time 01:52)

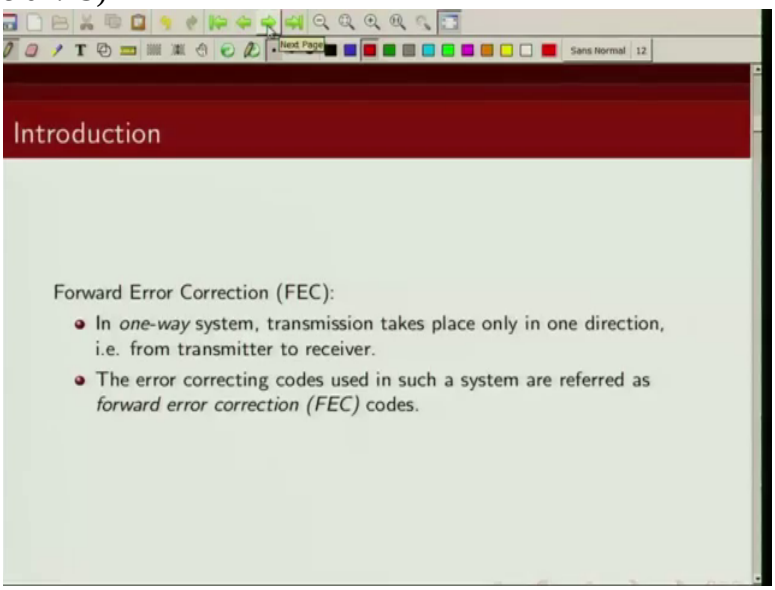


what do we do in forward error correcting schemes? So we encode our information sequence using an error correcting code and send it over the communication link.

(Refer Slide Time 02:04)

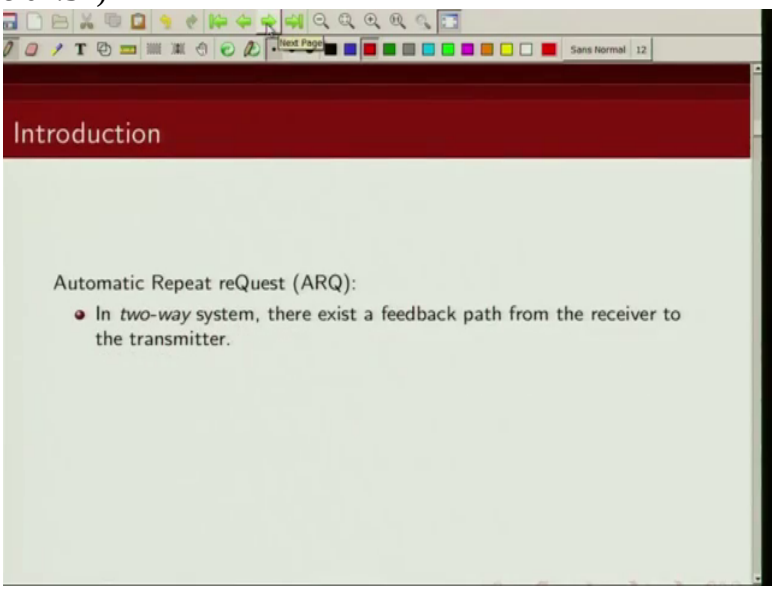
The receiver will try to detect and correct errors based on the redundant bits that we have transmitted. If the packet is not received correctly in a forward error correcting scheme, there is no mechanism for receiver to let the transmitter know that it has not received correctly because it is a one way channel. And the error correcting code used in these channels are known

(Refer Slide Time 02:29)



as forward error correction codes.

(Refer Slide Time 02:34)



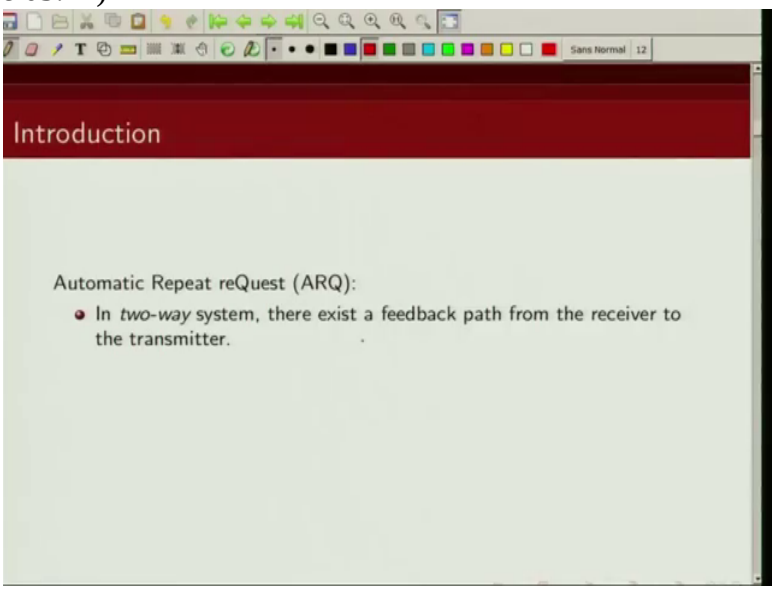
Now channels where there is a feedback from the receiver

(Refer Slide Time 02:40)

to the transmitter, so the receiver, if it receives the information sequence, we can encode the information sequence using error correcting codes which we may use for error detection. Now once we transmit this packet to the receiver, the receiver will try to detect if there is a parity violation. Now if the receiver is able to detect that the bits are not received correctly, the receiver is going to send a negative acknowledgment to the transmitter because it is a two way channel, so through this feedback channel the receiver will inform the transmitter that it has not received the packet correctly and then the transmitter will re-transmit the packet. Now if the packet which was transmitted is received error free then the receiver will send the

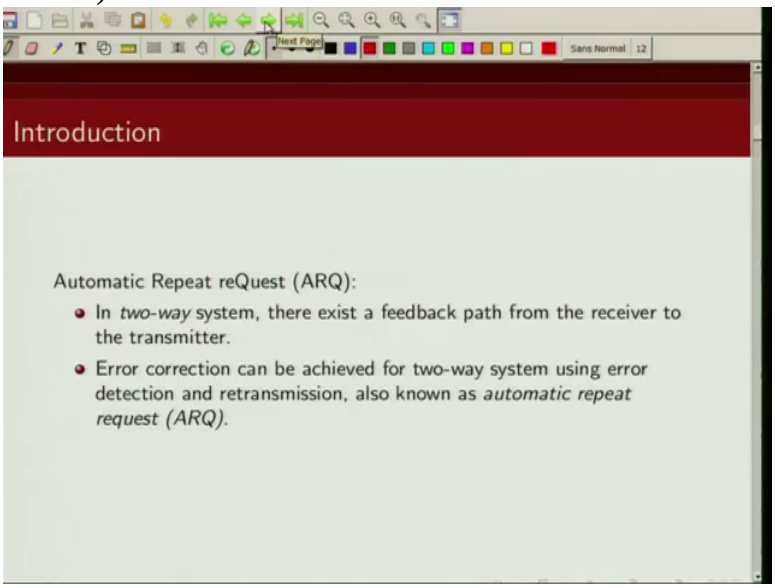
acknowledgement saying I have received the packet correctly and then the transmitter will send a new packet. So in a two way communication channel there exists

(Refer Slide Time 03:44)



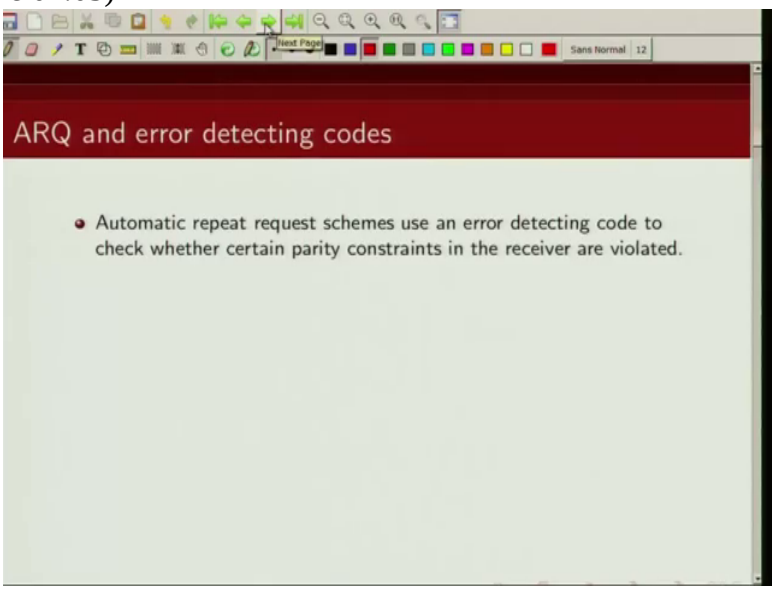
a feedback path from the receiver to the transmitter

(Refer Slide Time 03:48)



and the error correcting code that we use for such a communication channels are known as automatic repeat request which are, which is also known by this acronym A R Q.

(Refer Slide Time 04:05)

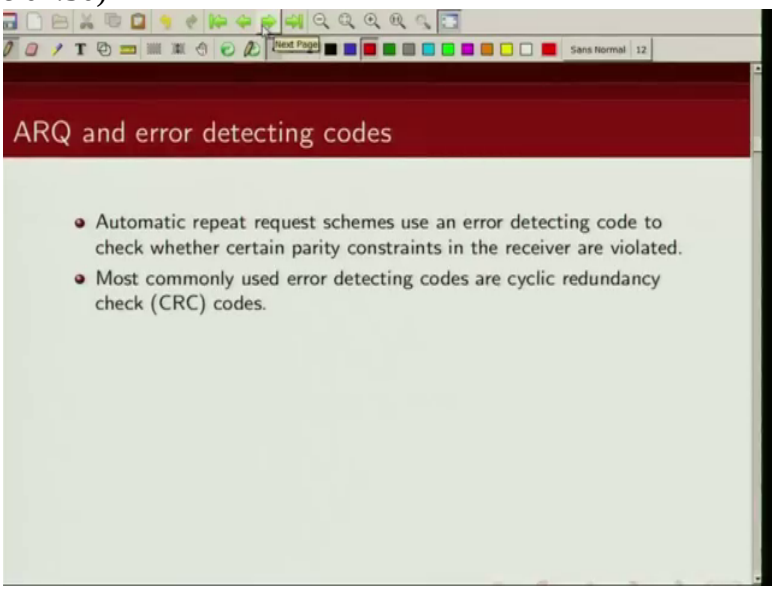


Now what is the role of error detecting code in A R Q schemes? So an automatic repeat request scheme uses an error detecting code to find out whether the packet that we transmitted over the communication channel, whether the packet has been

(Refer Slide Time 04:24)

received correctly or not. So what we do at the transmitter is we encode this information sequence using an error detecting code and we transmit over this communication channel. Now the receiver, uh this error detecting code will try to find out whether the packet has been received correctly or not using those additional parity bits.

(Refer Slide Time 04:50)

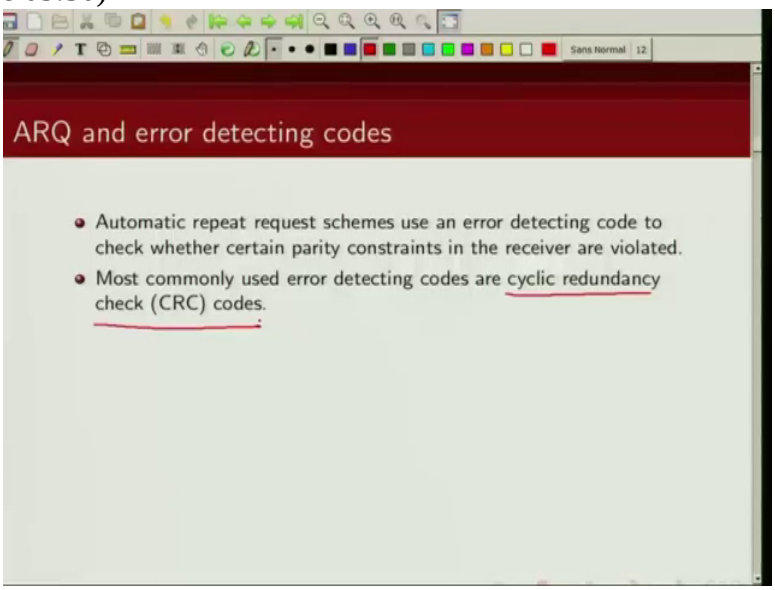


Now one of the most commonly used error detecting code in A R Q code application is what is known as C R C, cyclic redundancy check codes. C R C codes are error detecting codes which are very popularly used for A R Q applications. So what is done is we

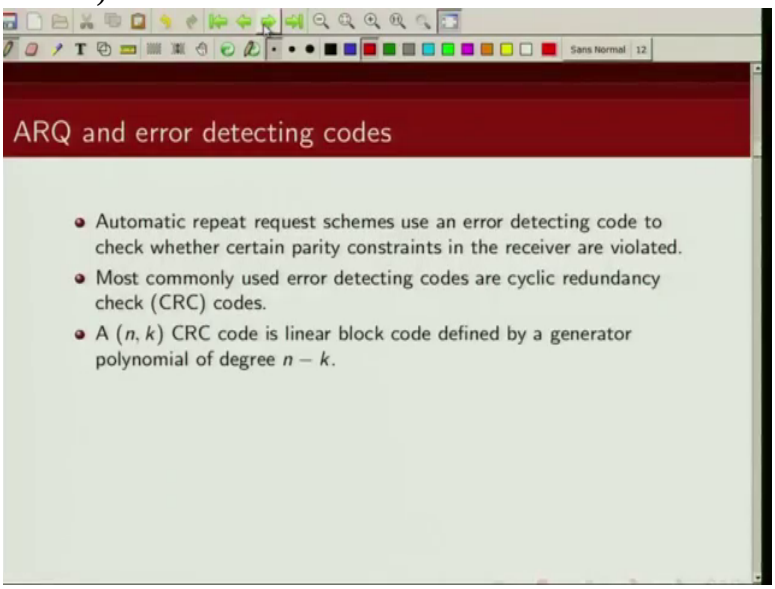
(Refer Slide Time 05:11)

encode our information sequence using this C R C code and send it over the communication channel and then using those additional parity bits, at the receiver we will try to find out whether the bits have been received correctly or not.

(Refer Slide Time 05:30)

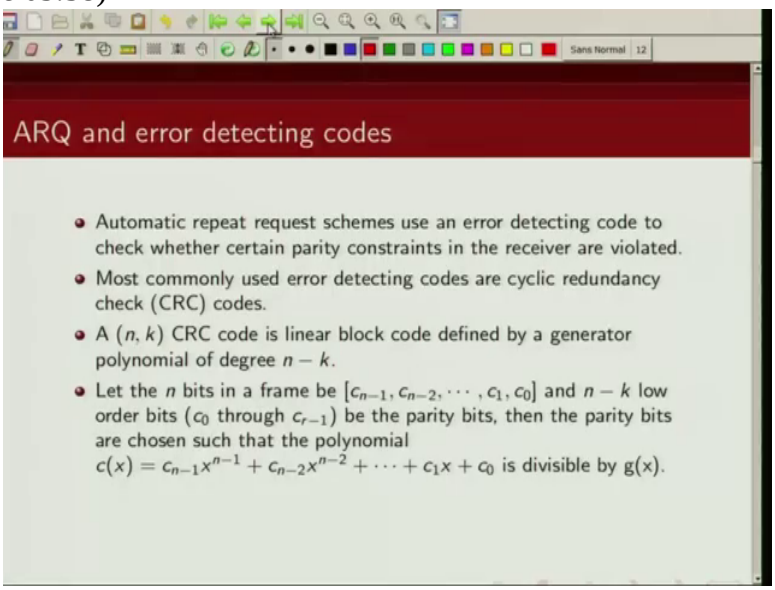


(Refer Slide Time 05:32)



So it is an n k code, linear block code. So block size is, code length is n. Information sequence length is k. So number of parity bits is n minus k. So a C R C code is defined by generator polynomial of degree n minus k which is number of parity bits that we are using.

(Refer Slide Time 05:58)

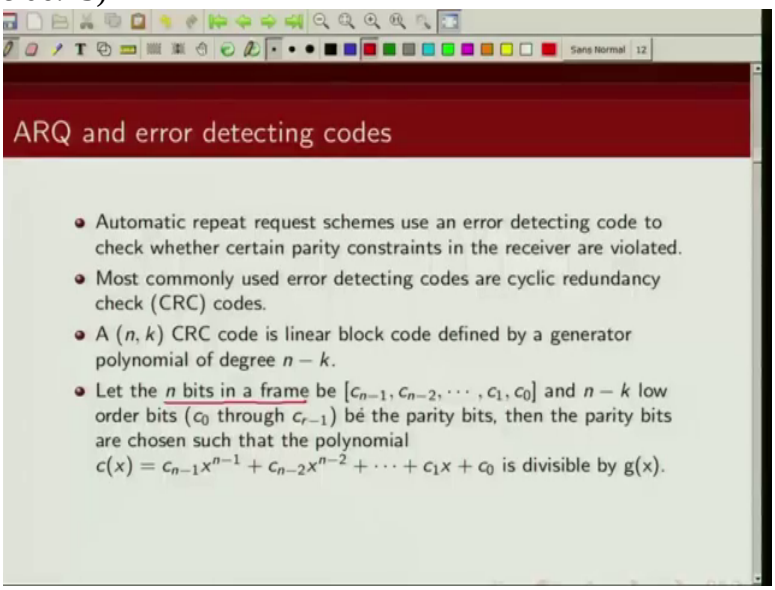


So let the n bits in a frame, now we are going to use interchangeably the word packet, frame. Essentially we are sending a block of

(Refer Slide Time 06:11)

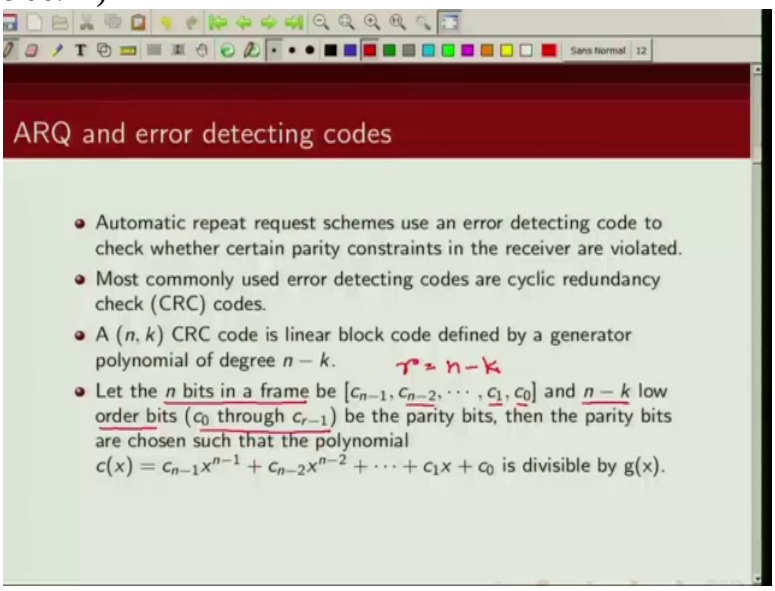
n bits. So let us denote

(Refer Slide Time 06:15)



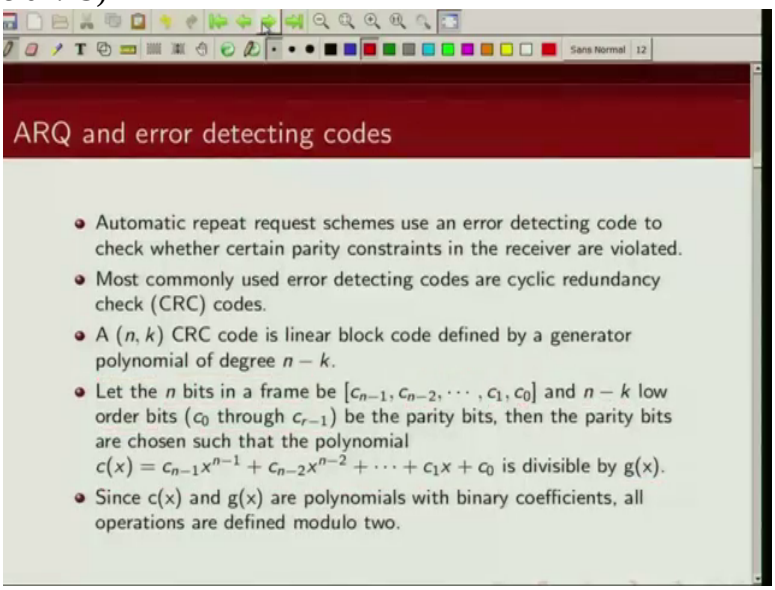
the block of n bits by c n minus one, c n minus 2, c n minus 3, c 1, c 0. So this is a block of n bits and let the n minus k low order bits that means from c 0 to c r minus 1, what is r, r is essentially my n minus k,

(Refer Slide Time 06:41)



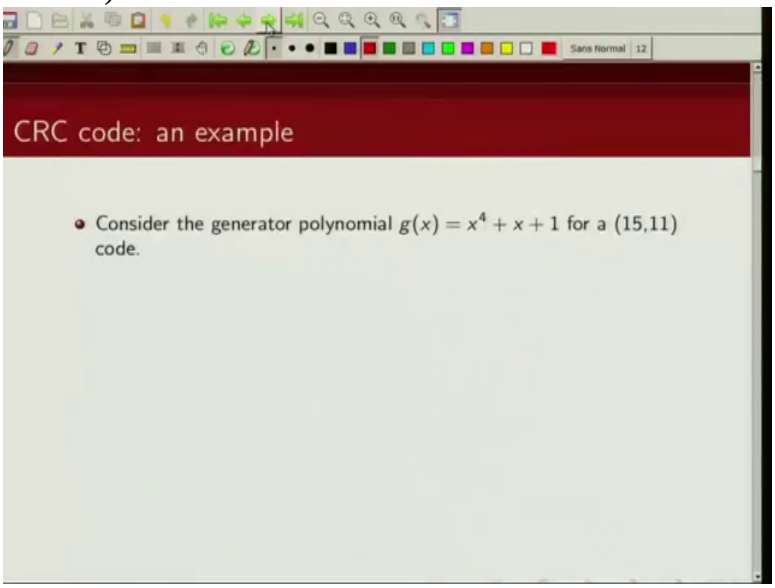
Ok. So let us assume that the low order n minus k bits are my parity bits. So then in C R C code, the parity bits are chosen in such a way such that this code polynomial is divisible by the generator polynomial of the C R C code. So we are going to choose this c 0, c 1, c 2, c 3, c r minus 1 in such a way such that this code polynomial c x is divisible by this generator polynomial of the C R C code of degree n minus k. Now let us take an example

(Refer Slide Time 07:29)



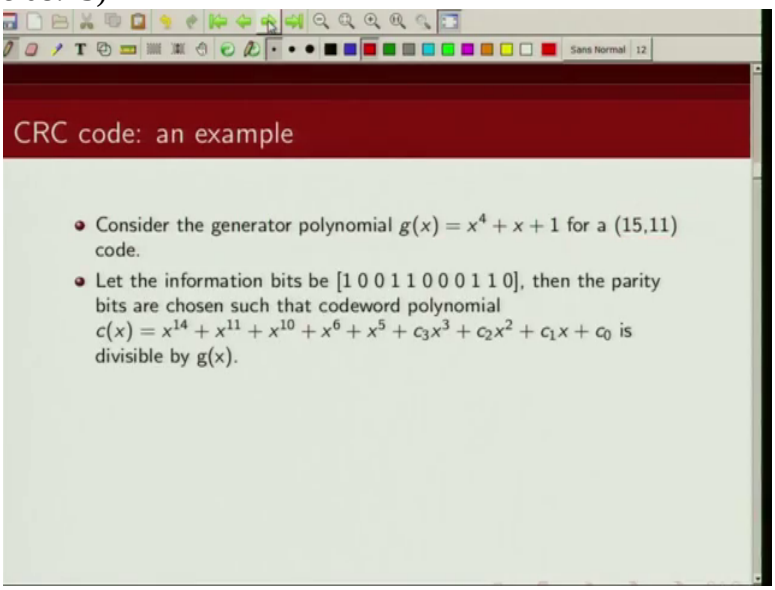
to illustrate this. And remember when we talking about, so these code polynomial as well as generator polynomials; they are polynomial with binary coefficients. So we are talking about all binary operations. So these operations are defined over modulo 2.

(Refer Slide Time 07:52)



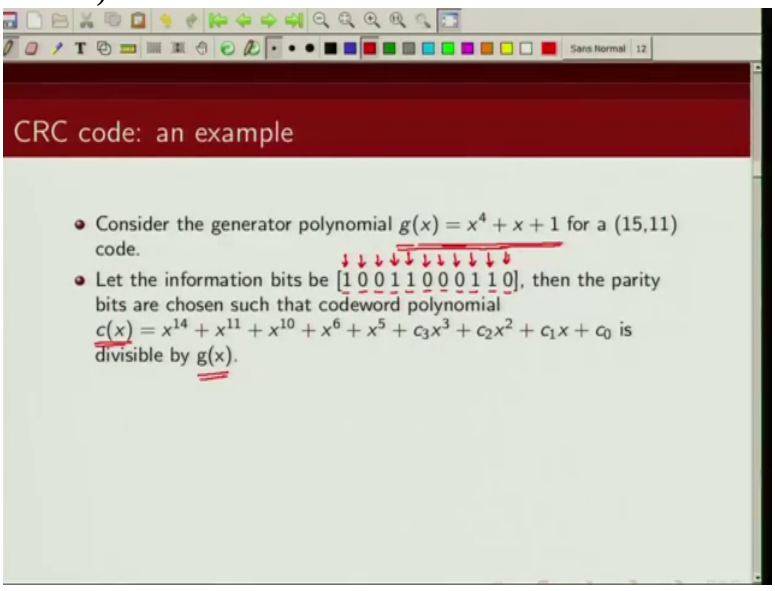
Let us take an example of a C R C code. So we consider a C R C code which has 4 parity bits whose generator polynomial is given by x to power 4 plus x plus 1. So this is a uh, generator polynomial for C R C code and our n k code parameters are given by 15 11. So n is 15 and k is 11 and there are 4 parity bits.

(Refer Slide Time 08:25)



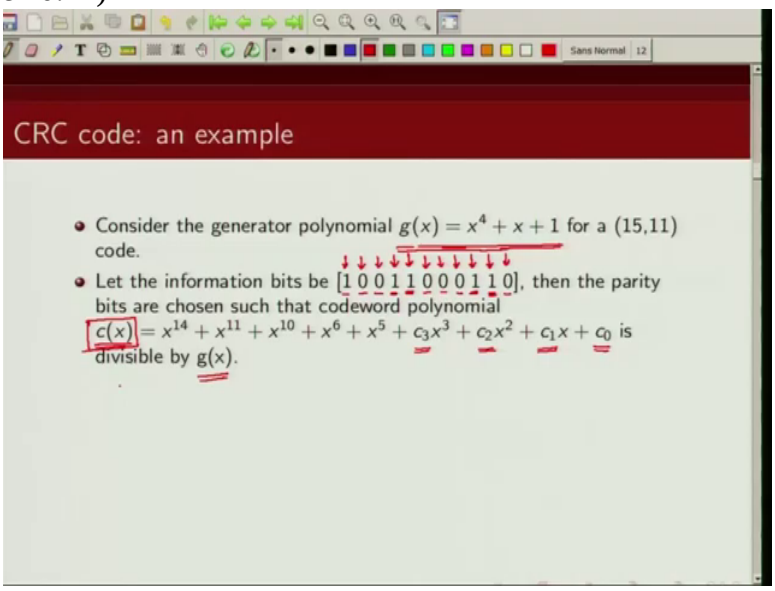
So let us assume those 11 bits are given by this. 1 0 0 1 1 0 0 0 1 1 0. So these are the 11 information bits. Now we want to encode these information bits using a C R C code which will add 4 parity bits whose generator polynomial is given by this. v x is x 4 plus x plus 1. Now remember we need to choose 4 parity bits in such a way such that this code polynomial c of x is divisible by this generator polynomial g of x. So we are going to write this information sequence in this polynomial form. So this is my c14, c 13, c 12, c 11, c 10, c 9, c 8, c 7, c 6, c 5, c 4 and then

(Refer Slide Time 09:37)



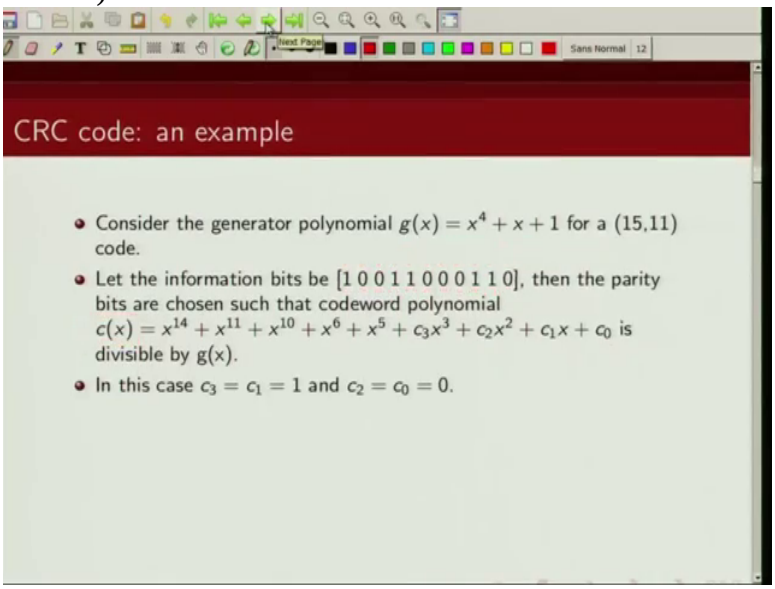
c 3, c 2, c 1, c 0 these are my parity bits. So remember this is x 14 plus x 11 plus x 10 plus x 6 plus x 5 plus c 3 x 3 plus c 2 x square plus c 1 x plus c 0. This is my code polynomial. Now I need to choose this c 3, c 2, c 1, c 0 in such a way such that this code polynomial c of x

(Refer Slide Time 10:14)



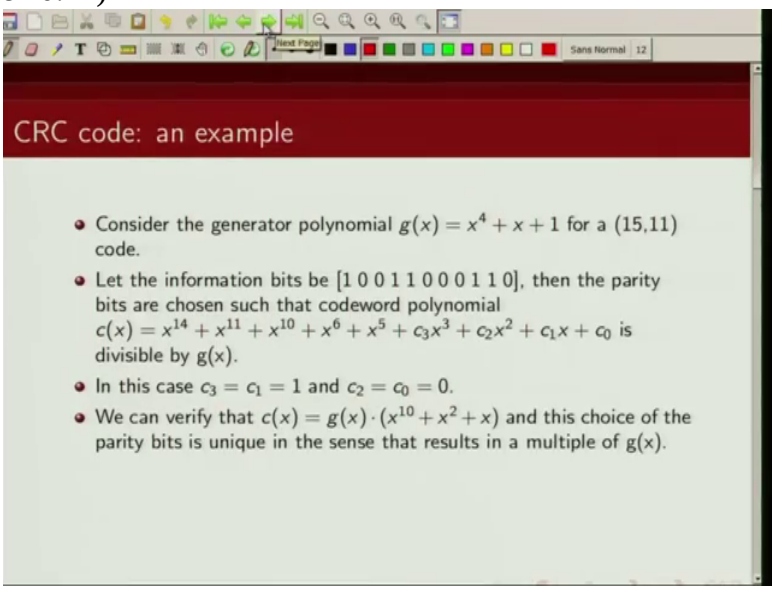
is divisible by g of x, this generator polynomial for this C R C code.

(Refer Slide Time 10:23)



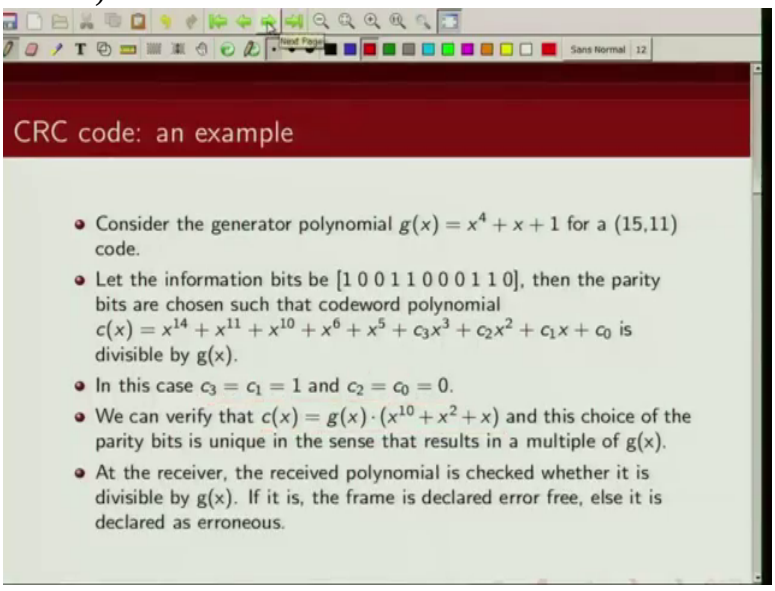
So in this case we can verify that if we choose c 3 and c 1 to be 1, and c 2 and c 0 to be zero, then this code polynomial is divisible by g of x

(Refer Slide Time 10:42)



and we can see that in this case if we choose our c 3, c 2, c 1, c 0 in this fashion our c of x is divisible by g of x and this is basically product of g of x into x to power 10 plus x square plus x. And this is a unique choice of these parity bits, c 0, c 1, c 2; c 3 which will ensure that c of x is divisible by g of x.

(Refer Slide Time 11:15)

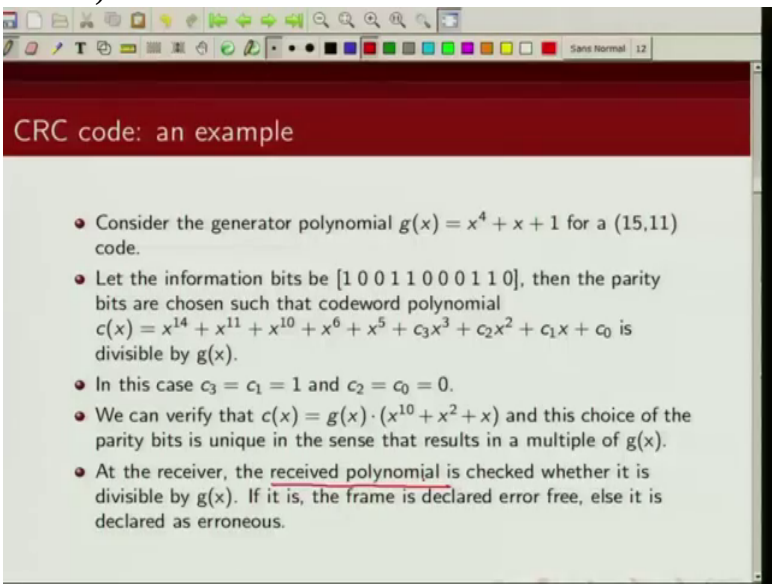


Now at the receiver what we are going to do is once we get the received polynomial we are going to check whether there are errors or not.

(Refer Slide Time 11:26)

How are we going to check? We are going to divide this received polynomial by g of x. If it is divisible by g of x that means there are either no error or it has converted into some other codeword and the errors are undetected, Ok. Now if this received polynomial

(Refer Slide Time 11:49)

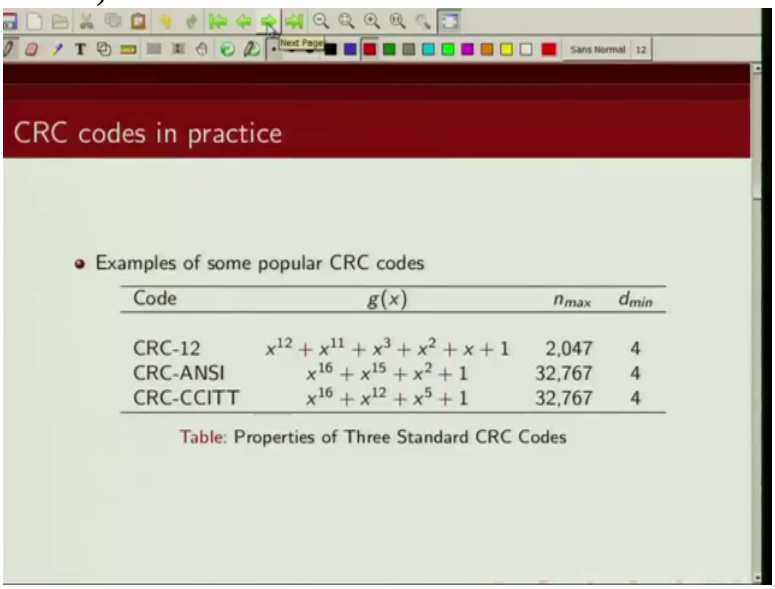


is divisible by g of x then we declare that the frame is error free or else if it is not divisible by g of x, we know that there is an error in the received sequence. So this is how we are going to make use of this C R C for error

(Refer Slide Time 12:09)

detection

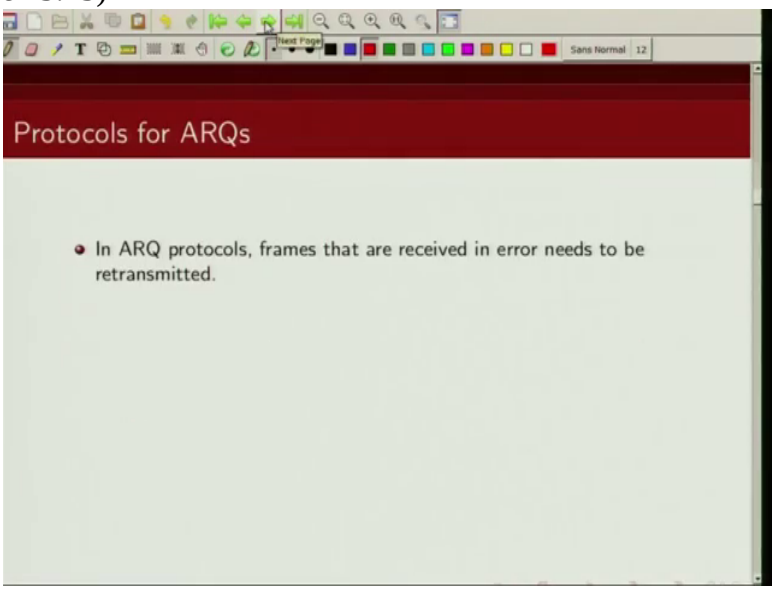
(Refer Slide Time 12:12)



I have listed some of the popular C R C codes. So this C R C 12 code, its generator polynomial is given by this, x 12 plus x 11 plus x 3 plus x square plus x plus 1. The maximum n that it can handle is two thousand forty seven and its minimum distance is 4. So it can detect 3 errors. Another popular code is C R C ANSI code which has, this is 16 bit C R C code. Generator polynomial given by x 16 plus x 15 plus x square plus 1 and can handle n of thirty two thousand seven sixty seven. Its minimum distance is also 4. Another 16 bit C R C code which is popular is C R C CC I TT code and again its generator polynomial is given by x 16 plus x 12 plus x 5 plus 1. Its minimum distance is also 4 and it can take maximum n of thirty two thousand seven sixty seven. So if you subtract from this n, the number of parity

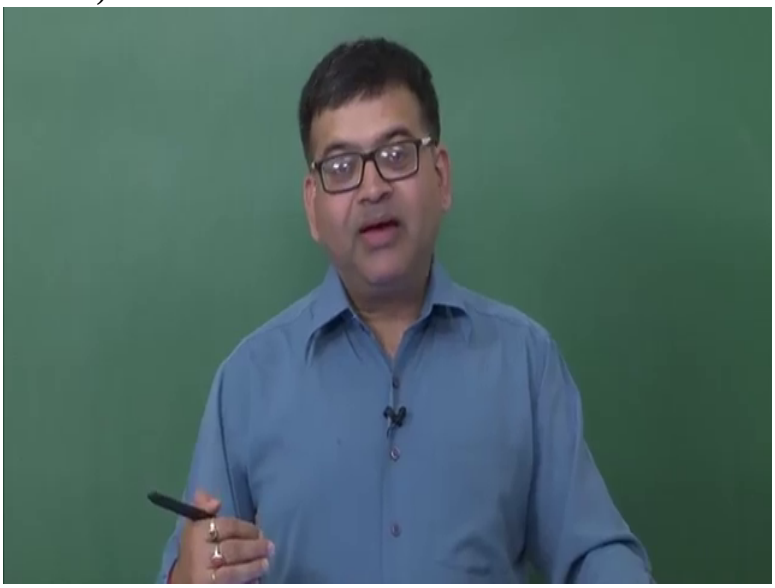
bits in this case its 12, in this case it is 16, in this case it is 16 you will get the size of the information sequence that you are encoding using these C R C codes.

(Refer Slide Time 13:43)



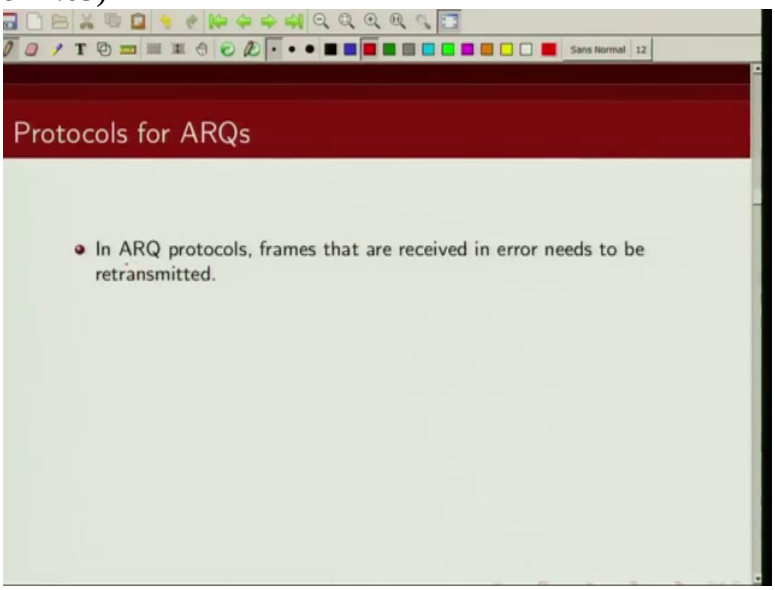
Now that we have explained what is an A R Q scheme

(Refer Slide Time 13:49)



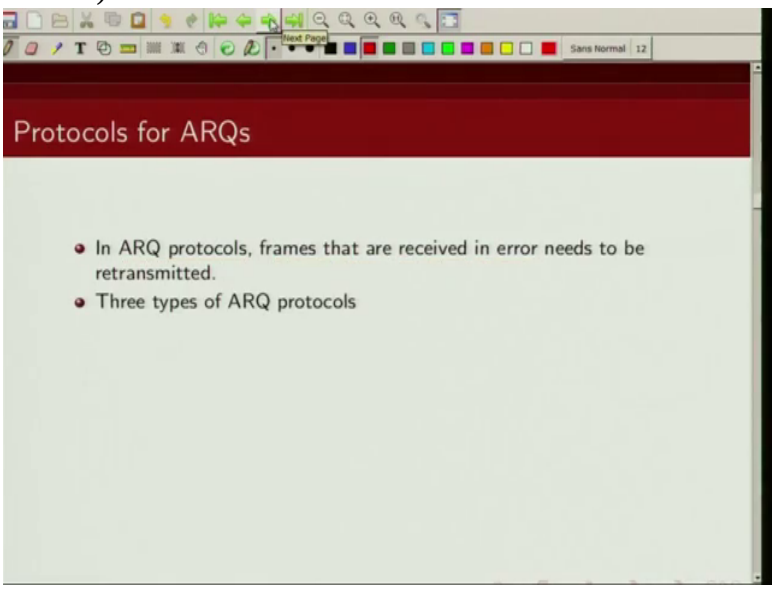
and the role played by error detecting code, let us talk about what are the different protocols that we can employ when a packet is detected to be in error? What are the different ways in which we can do retransmission? So as you know

(Refer Slide Time 14:09)



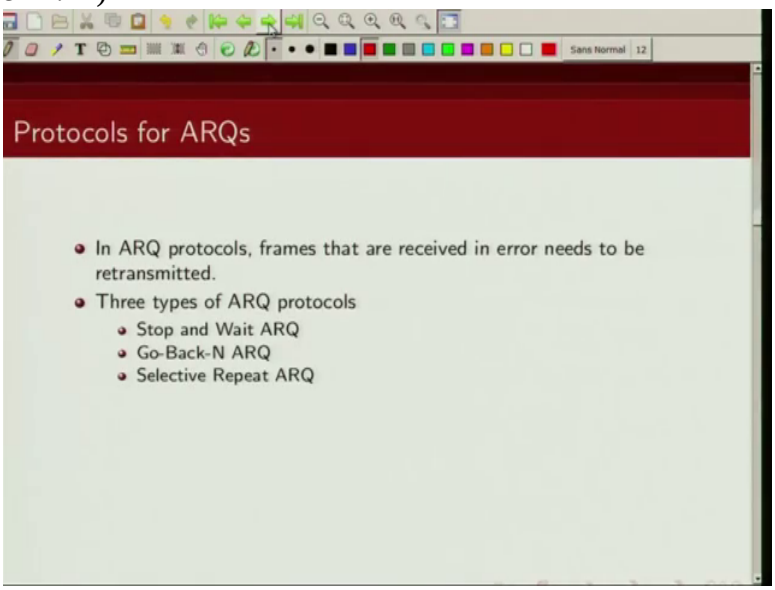
in A R Q protocols if the frames are received in error, you need to retransmit those frames or packets.

(Refer Slide Time 14:18)



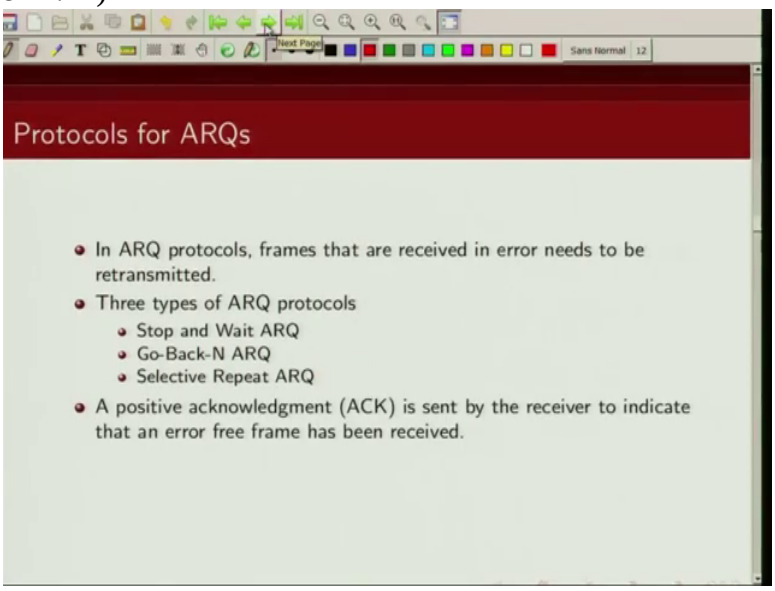
So we are going to talk about 3 different protocols for A R Q,

(Refer Slide Time 14:24)



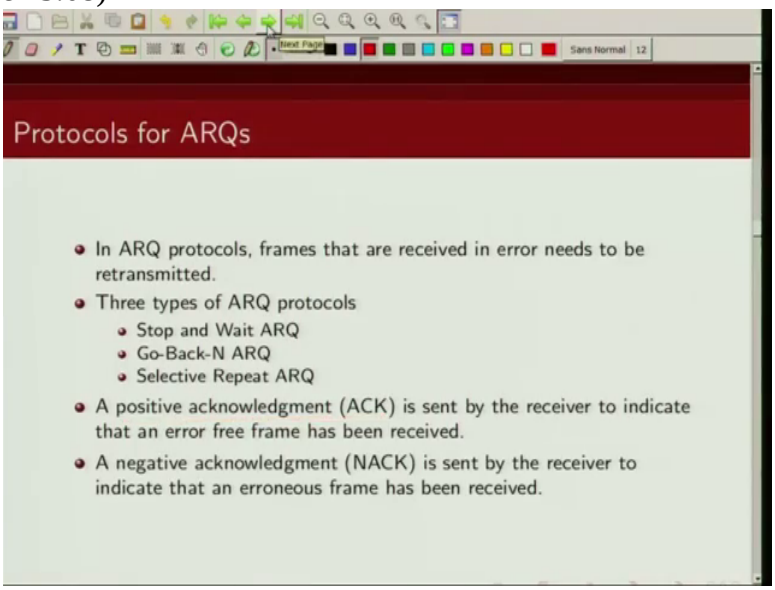
the first one is called Stop and Wait, the second is known as Go Back N and the third is known as Selective Repeat A R Q scheme. We are going to talk about each one of them in detail and we are also going to do performance analysis of each of these A R Q protocols.

(Refer Slide Time 14:47)



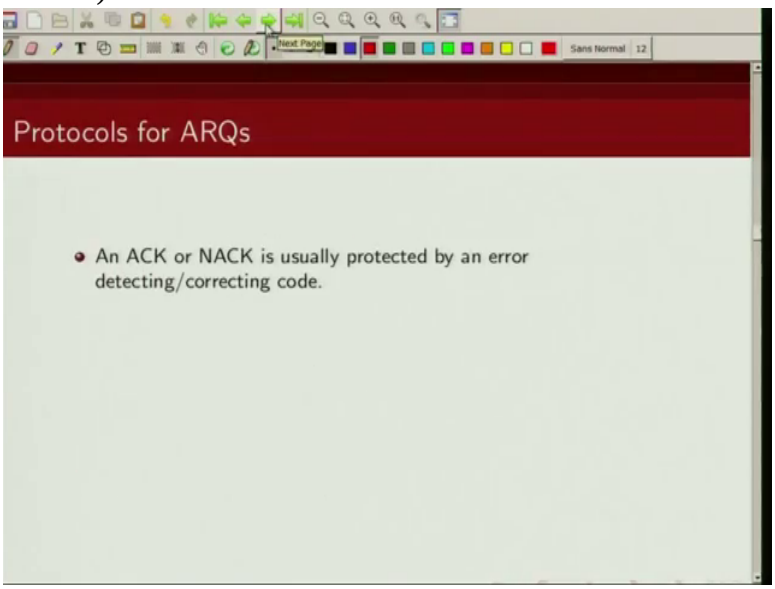
Now the way these A R Q protocols work is basically uh, if the packet is received error free or there are undetected errors, the receiver is going to send a positive acknowledgement to indicate that the packet has been successfully received otherwise

(Refer Slide Time 15:08)



it is going to send a negative acknowledgment and transmitter is then going to re-transmit some packets, Ok.

(Refer Slide Time 15:20)

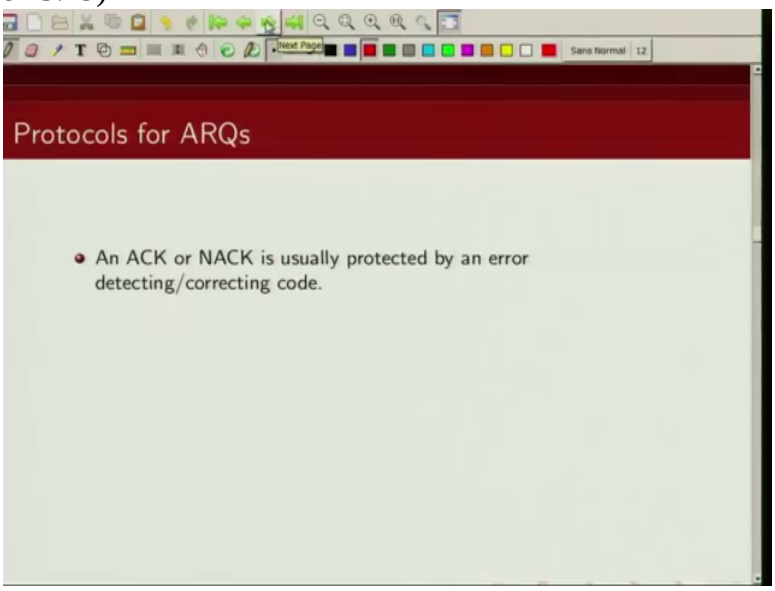


Now when we are sending this acknowledgment and negative acknowledgment from the receiver to the transmitter through the feedback channel we assume

(Refer Slide Time 15:31)

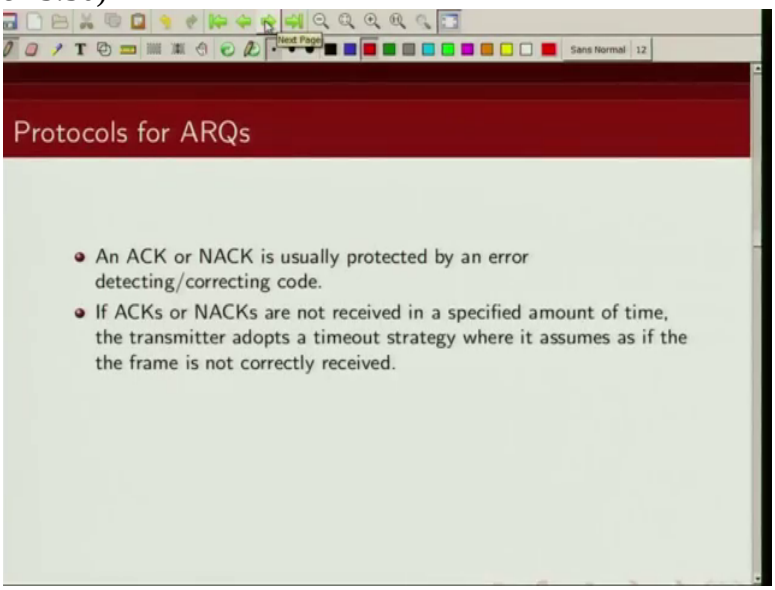
that these have been protected using some error correcting and error detecting code. So in our analysis we will assume that there is no error in sending these acknowledgment and negative acknowledgment signals from the receiver

(Refer Slide Time 15:49)



to the transmitter.

(Refer Slide Time 15:50)

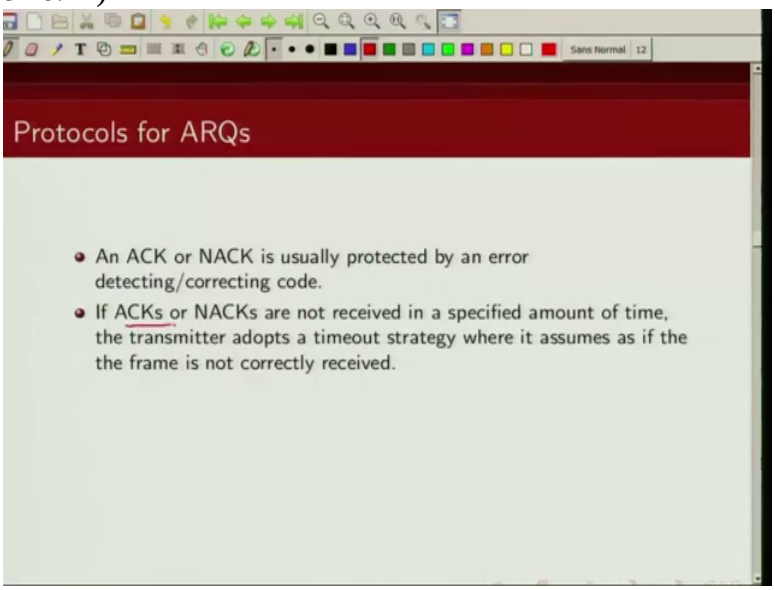


We will also assume that if the transmitter does not receive

(Refer Slide Time 15:56)

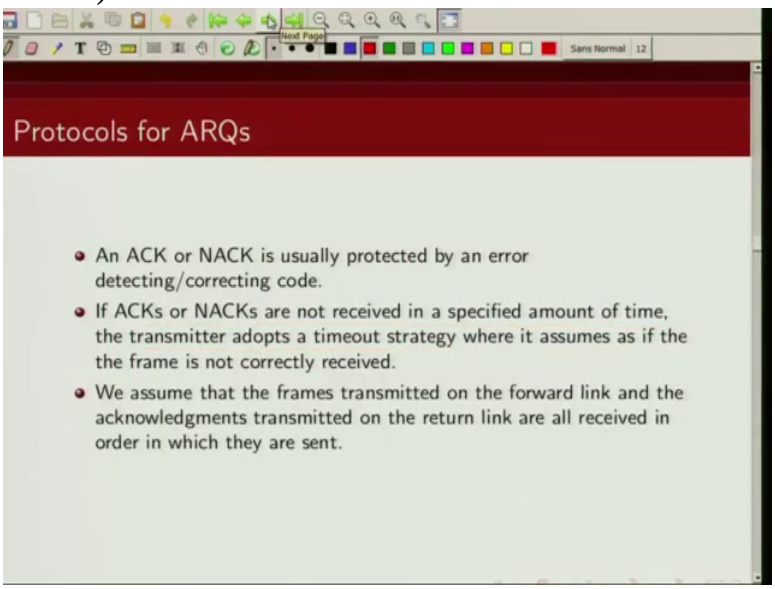
acknowledgment or negative acknowledgment in time, so there is a timeout, so we fix up a time, if we receive a acknowledgment or negative acknowledgment of a packet we have transmitted, after we have transmitted that packet, if we do not receive an ACK or an NACK between, within a particular time instance we will assume that the packet has not been successfully received, Ok.

(Refer Slide Time 16:24)



So if ACKs and NACKs are not received in specified amount of time, the transmitter adopts a timeout strategy where it will assume as if the frame is not correctly received. The another

(Refer Slide Time 16:37)

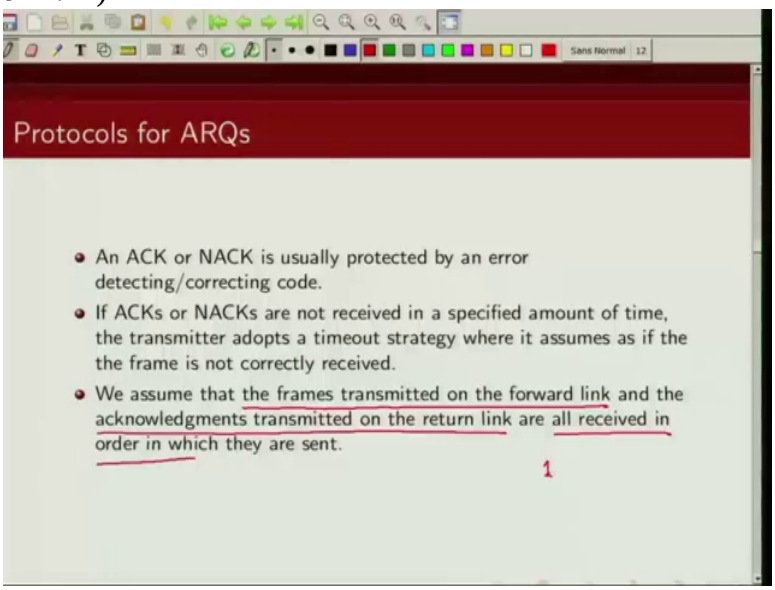


assumption that we are going to make is that the all the frames that we transmit over the forward link as well as the acknowledgment that are sent over the reverse link , they are all received in order. So they are all received in order. So the packets that we are sending first, the receiver is going to receive them first,

(Refer Slide Time 16:59)

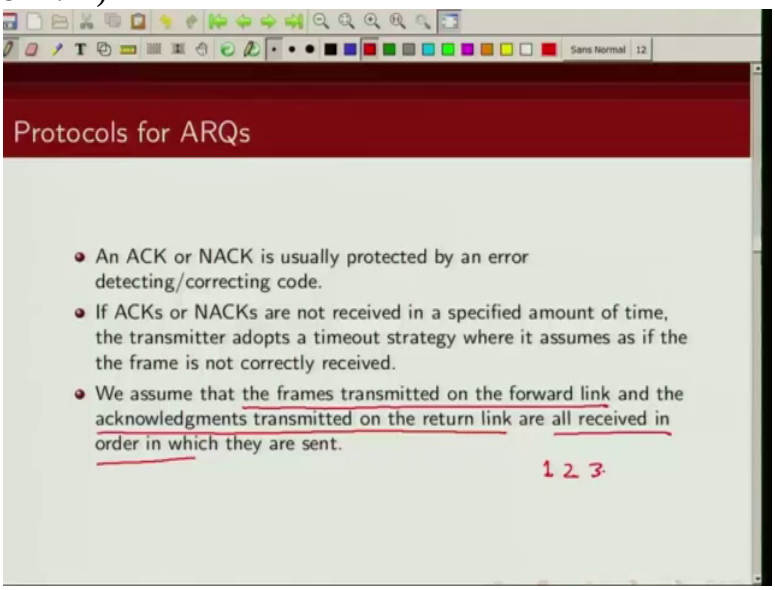
the transmitter is going to receive the NACK and ACK of those packets first which have transmitted earlier. That's what I mean when I said all are received in order. So if I send packet number 1

(Refer Slide Time 17:12)



first, then 2 first and 3 first,

(Refer Slide Time 17:14)

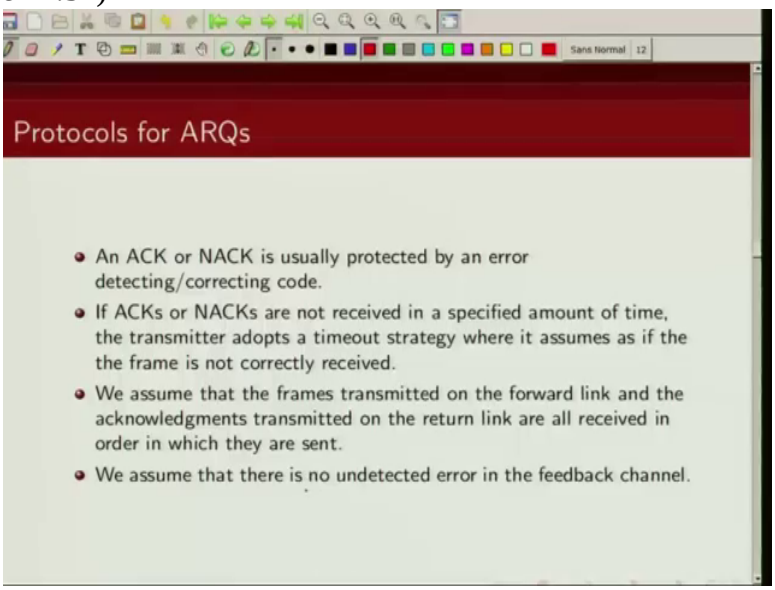


the receiver will also receive packet 1 first then 2, and then 3. And similarly the transmitter will first receive the NACK or

(Refer Slide Time 17:23)

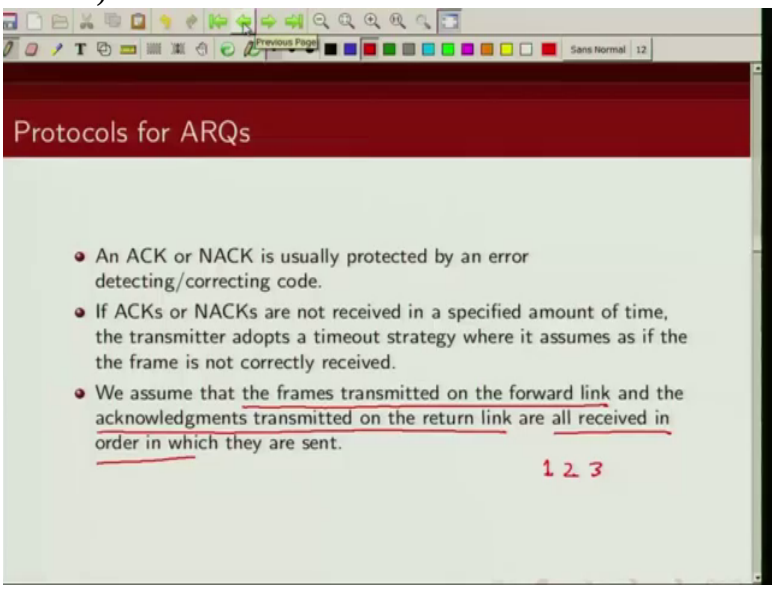
ACK of packet 1, then NACK and ACK of packet 2, and then NACK or ACK of packet 3. That's what I mean when I say all are received in order.

(Refer Slide Time 17:37)



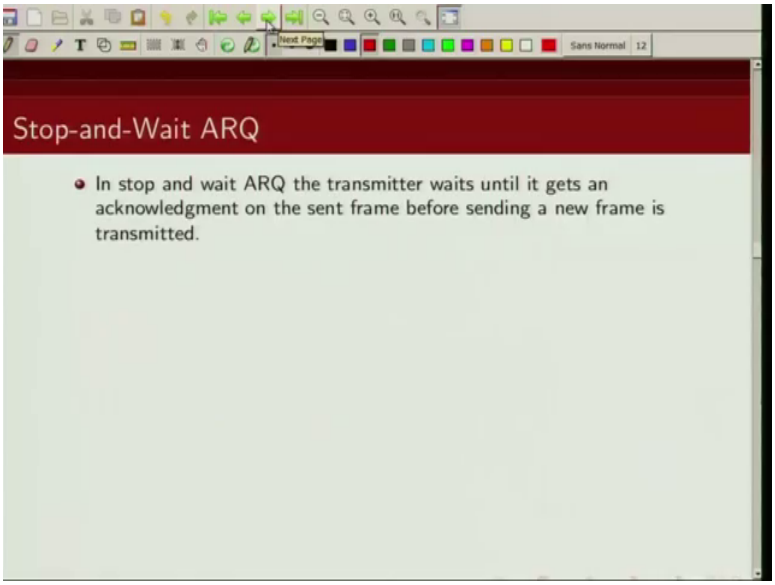
We also assume that there is no undetected error in the feedback channel. So

(Refer Slide Time 17:44)



the NACK and ACK that are sent are received correctly by the transmitter.

(Refer Slide Time 17:51)

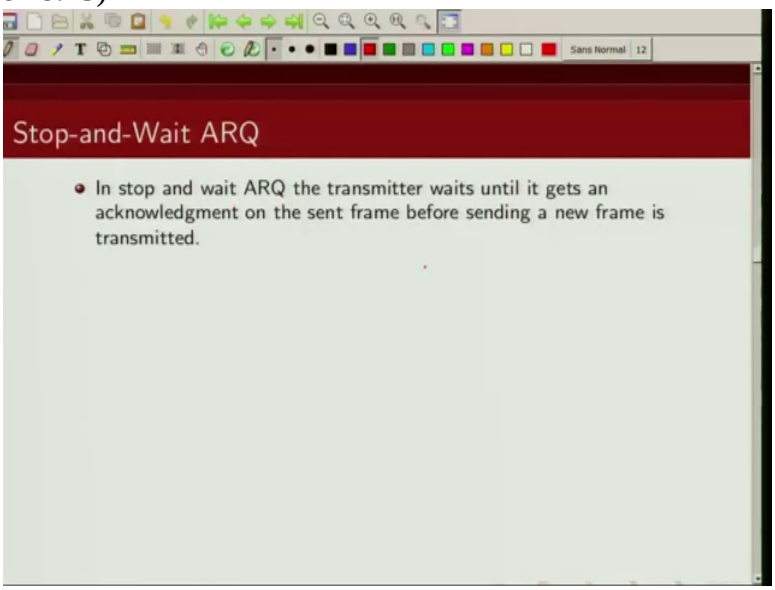


Now let us talk about each of these protocols in detail. The first protocol that we are going talk about is known as Stop and Wait A R Q. So as the name suggests, we transmit the packet and wait

(Refer Slide Time 18:06)

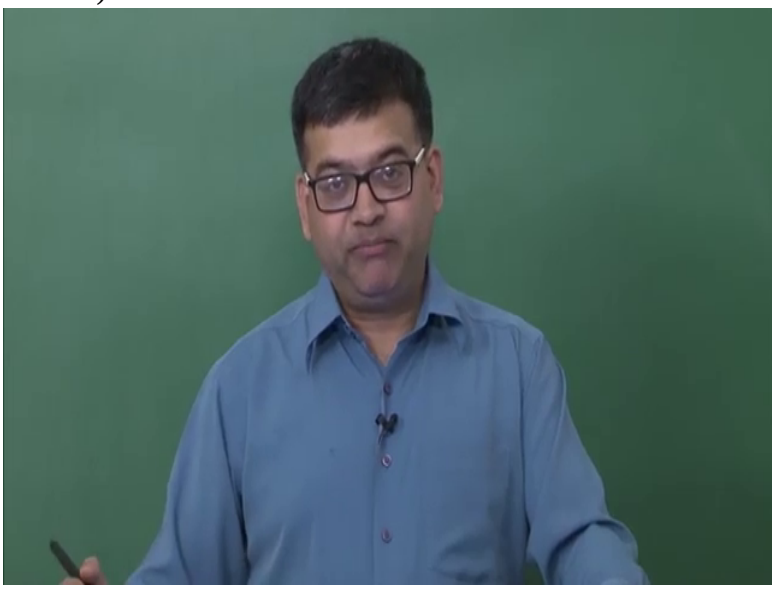
until we get acknowledgement of that particular packet or the frame. So in Stop and Wait

(Refer Slide Time 18:13)



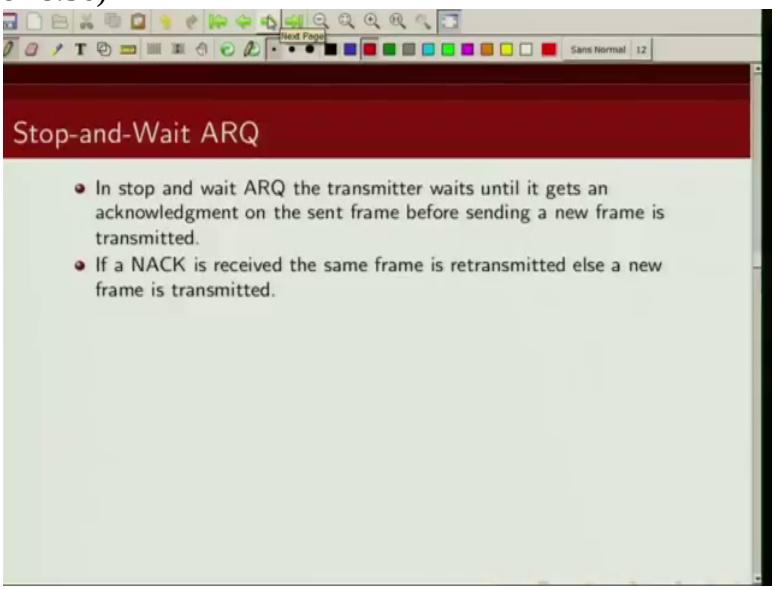
A R Q, the transmitter waits until it gets the acknowledgment of the sent frame before sending a new frame.

(Refer Slide Time 18:24)



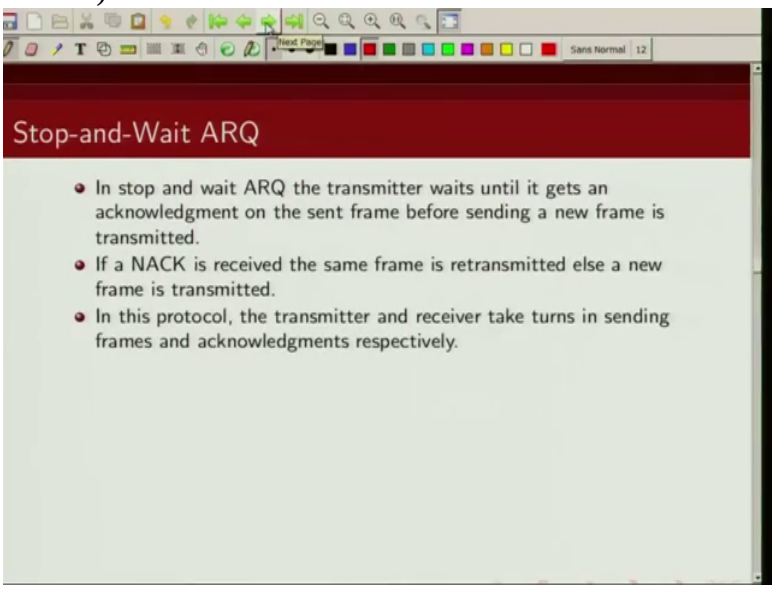
So in Stop and Wait A R Q Protocol we are going to send the packet, the receiver is going to receive it, it is going to detect, depending on whether the packet is received correctly or not, it is going to send back to transmitter an acknowledgment or negative acknowledgment and based on that the transmitter will now send either a new frame or a packet or the same packet if it is not received correctly.

(Refer Slide Time 18:56)



So if NACK is received, the same frame is re-transmitted else a new frame is

(Refer Slide Time 19:03)

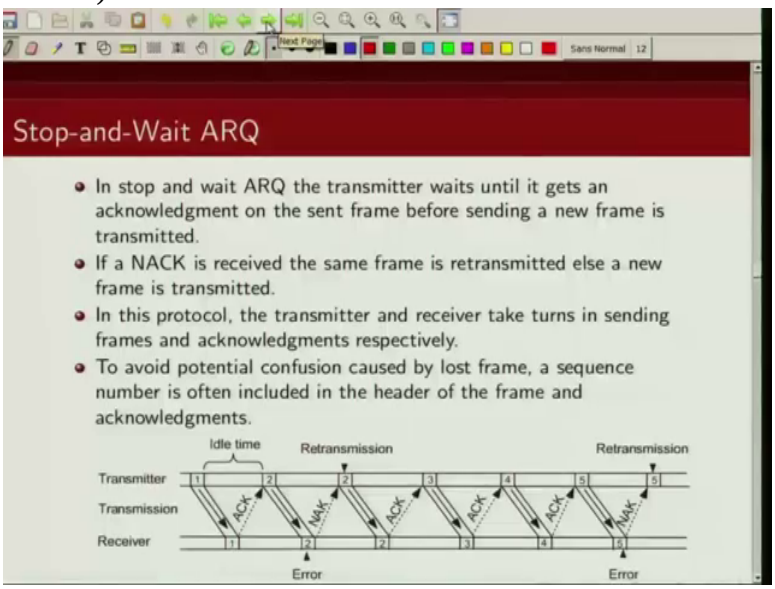


transmitted. So in this protocol the transmitter and the receiver take turns in sending frames and acknowledgment.

(Refer Slide Time 19:13)

So transmitter is sending a packet, is sending a frame to the receiver. The receiver is sending an ACK or a NACK to the transmitter back

(Refer Slide Time 19:25)



and to avoid potential confusion caused by lost frame the sequence number is often included in the header of the frame that we are sending, similarly there is a sequence number embedded in the negative or positive

(Refer Slide Time 19:40)

acknowledgment so that the transmitter knows which packet has been received correctly or not. So in this

(Refer Slide Time 19:48)

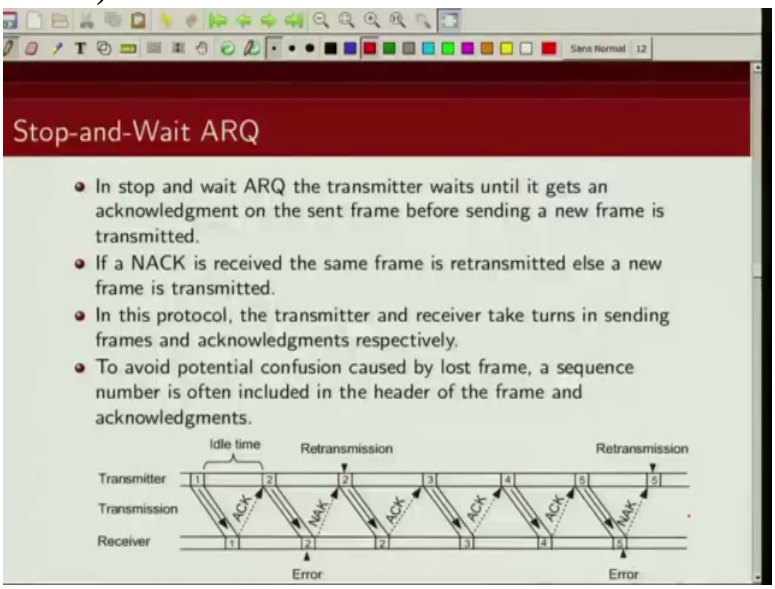
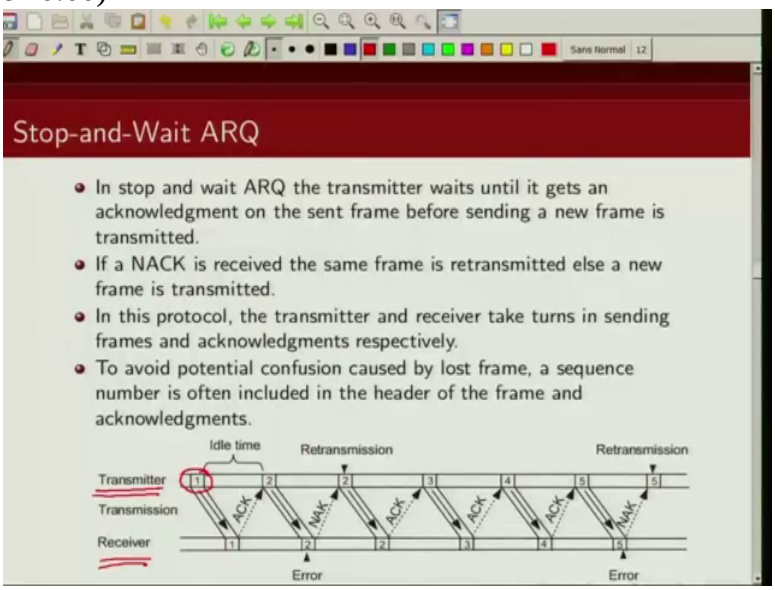


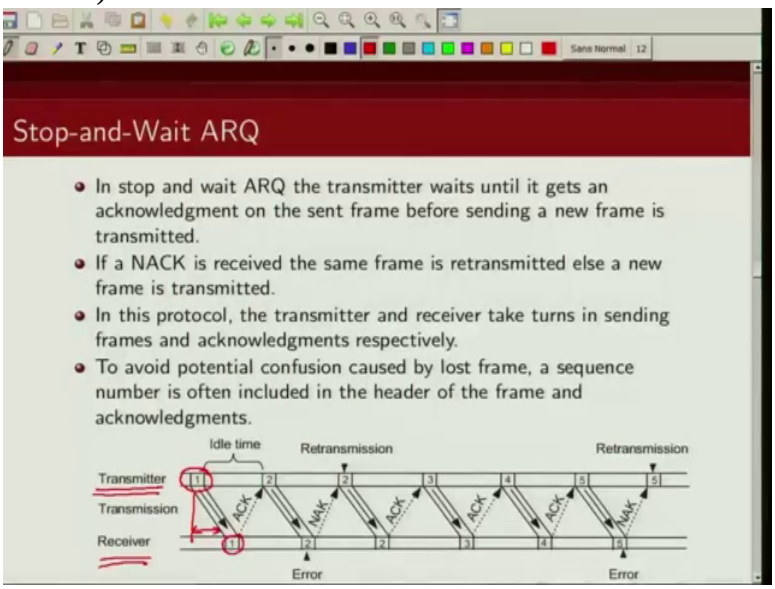
diagram I have shown how this Stop and Wait A R Q works. So this is my transmitter, this is my receiver. So let's say it

(Refer Slide Time 20:00)



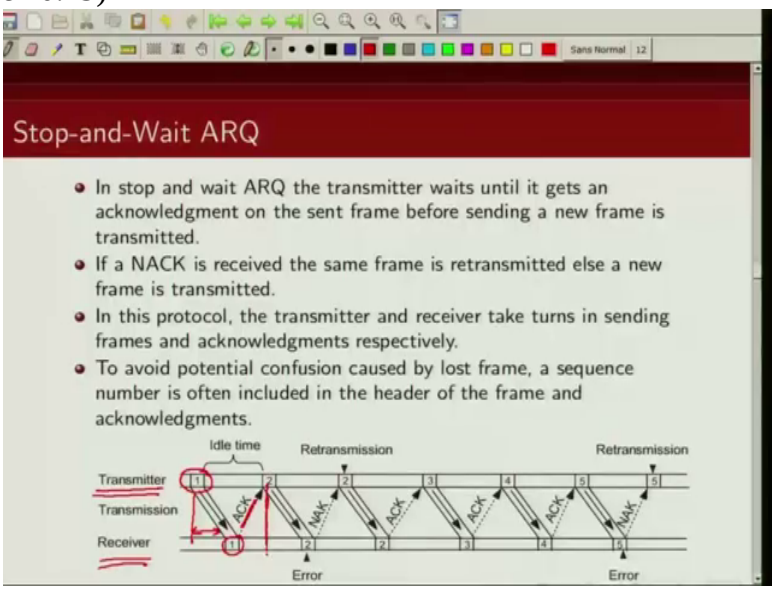
transmits frame number 1, Ok. Now there is some propagation and the transmission delay so basically after some time, the receiver receives this packet 1.

(Refer Slide Time 20:12)



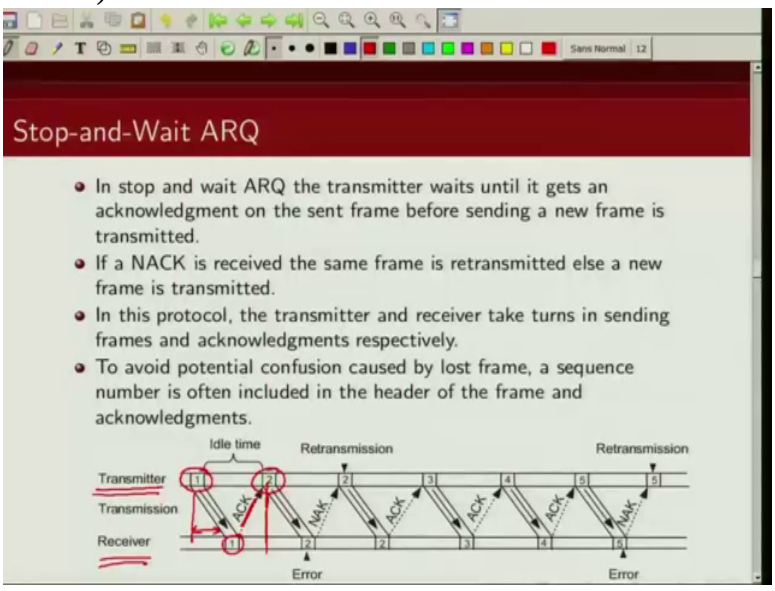
And using the C R C code it detects whether the packet is in error or not. So in this case, the packet is not in error so it sends a positive acknowledgement. So it sends an acknowledgement to the, back to transmitter. At this time

(Refer Slide Time 20:29)



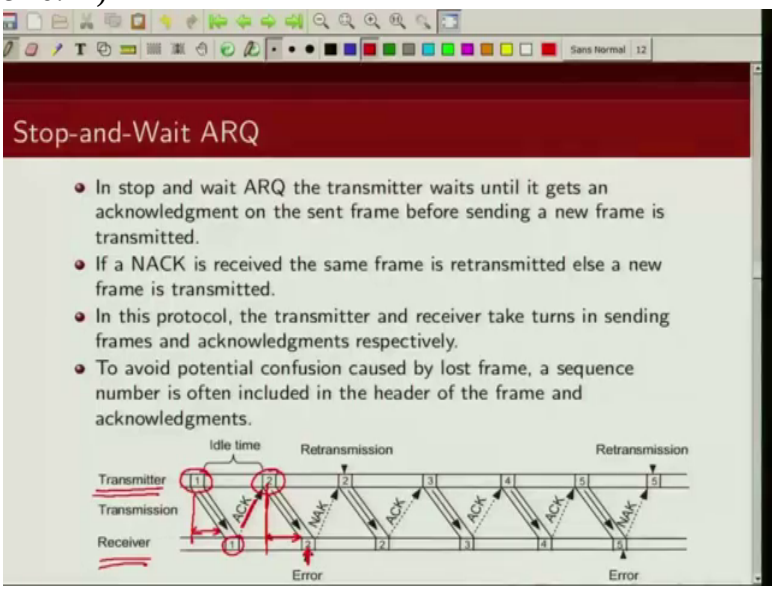
instance the transmitter has received an acknowledgement of frame number 1. So now it is going to send frame

(Refer Slide Time 20:37)



number 2. Again after some delays the frame number 2 is received at this

(Refer Slide Time 20:44)

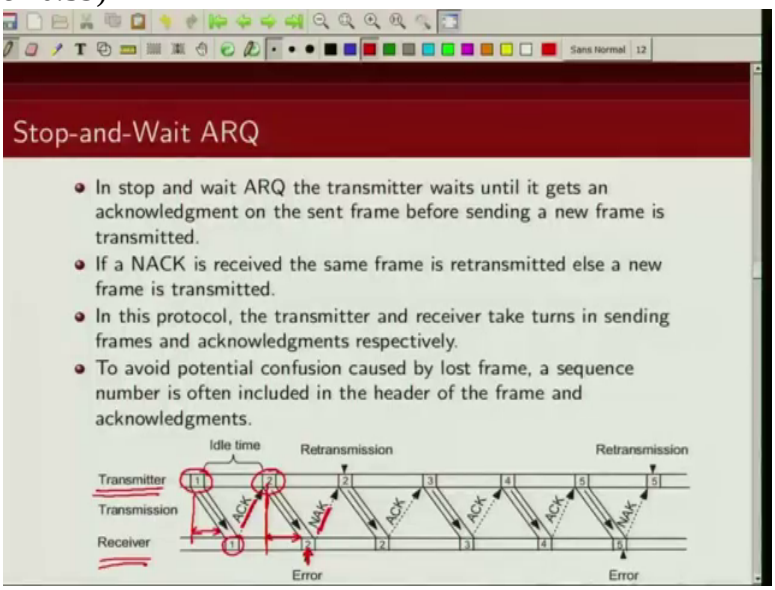


receiver. The receiver will employ C R C check to find out whether there is

(Refer Slide Time 20:51)

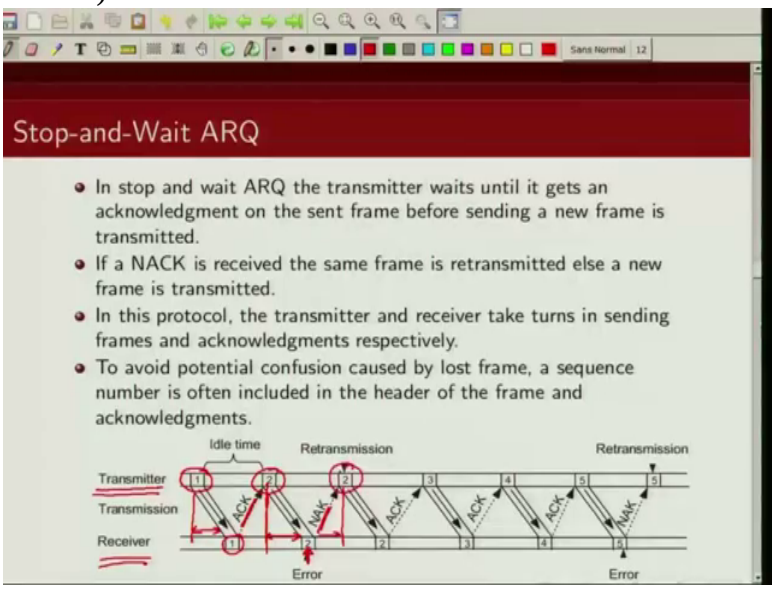
any parity violation. In this particular example there is a parity violation so the receiver sends a negative

(Refer Slide Time 20:59)



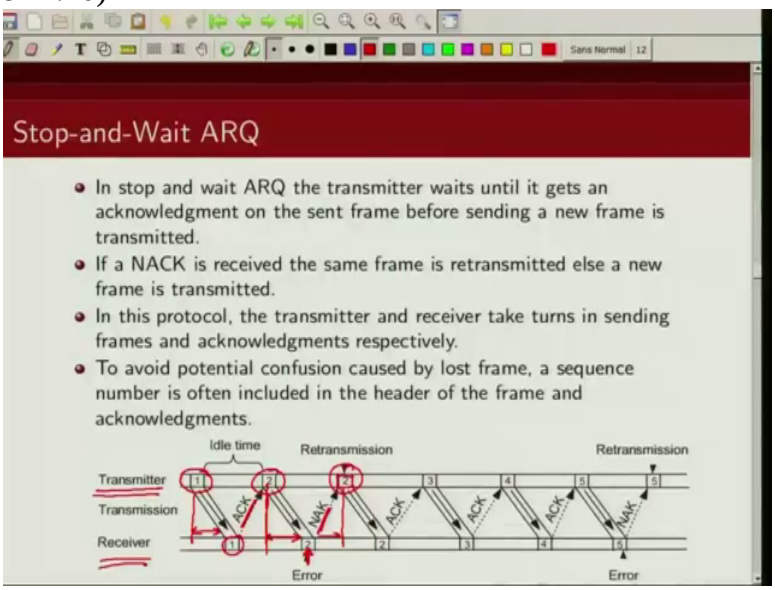
acknowledgement to the transmitter. Now after this time delay the

(Refer Slide Time 21:05)



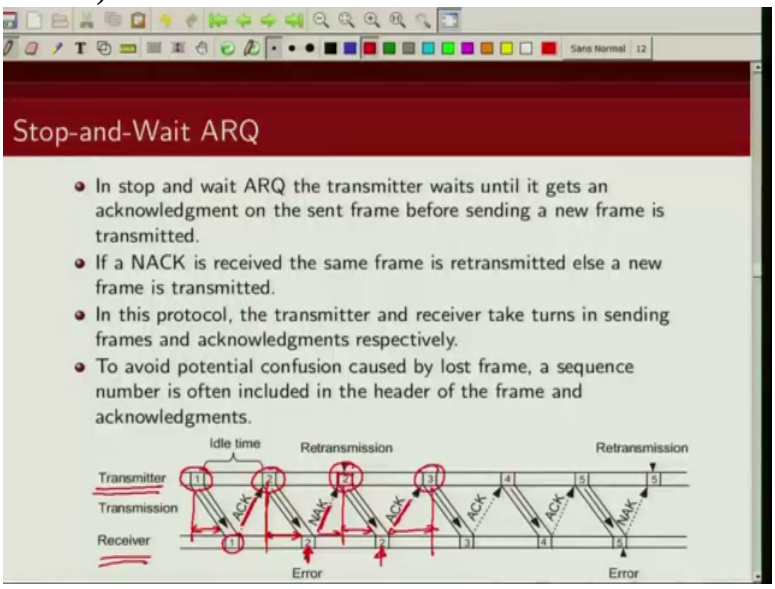
transmitter receives at this time instance a negative acknowledgement that is the packet 2 or frame 2 has not been received correctly. So what does the transmitter do? It retransmits packet 2.

(Refer Slide Time 21:20)



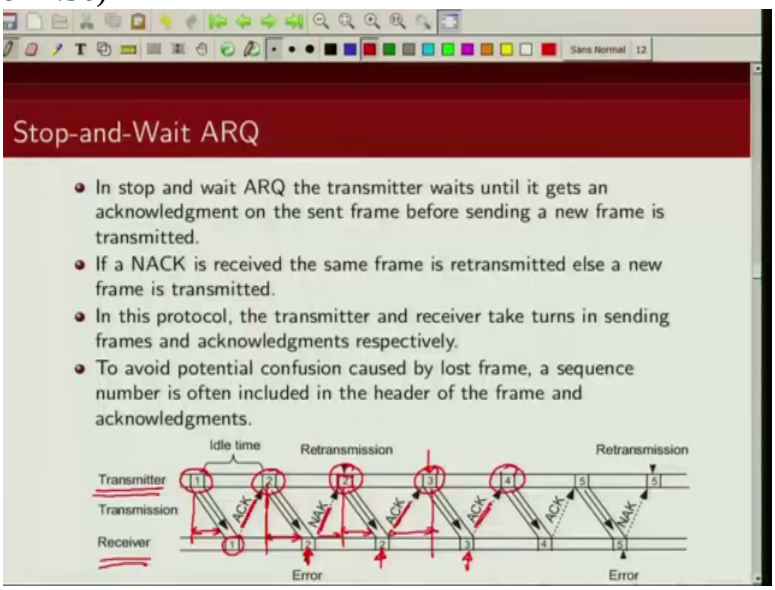
And again after some delay this packet is received at the receiver. This time this is received correctly. So receiver sends an acknowledgement after some time, propagation delay, this acknowledgement

(Refer Slide Time 21:36)



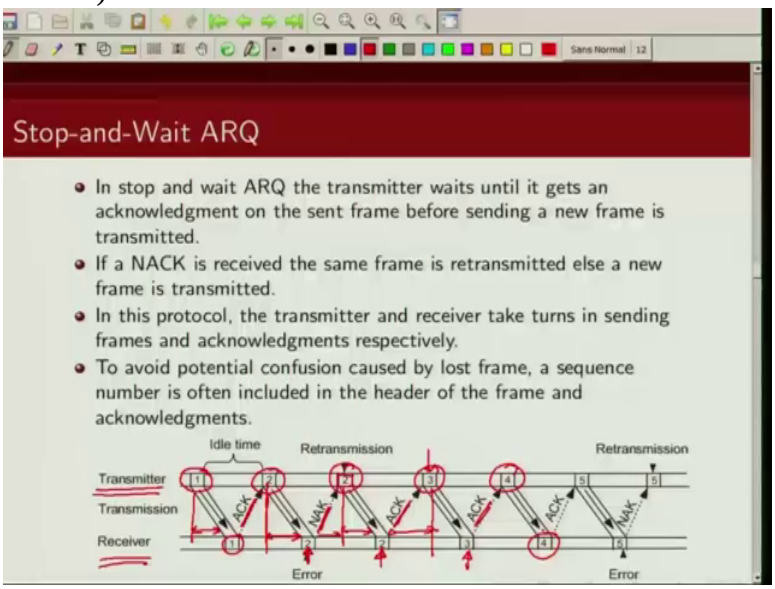
is received at the transmitter, since it is a positive acknowledgement, the transmitter will decide to send a new frame, in this case frame number 3. So this process continues. This frame is received here after some time, delay, this is received correctly so it sends an acknowledgement. Next the transmitter is going

(Refer Slide Time 21:56)



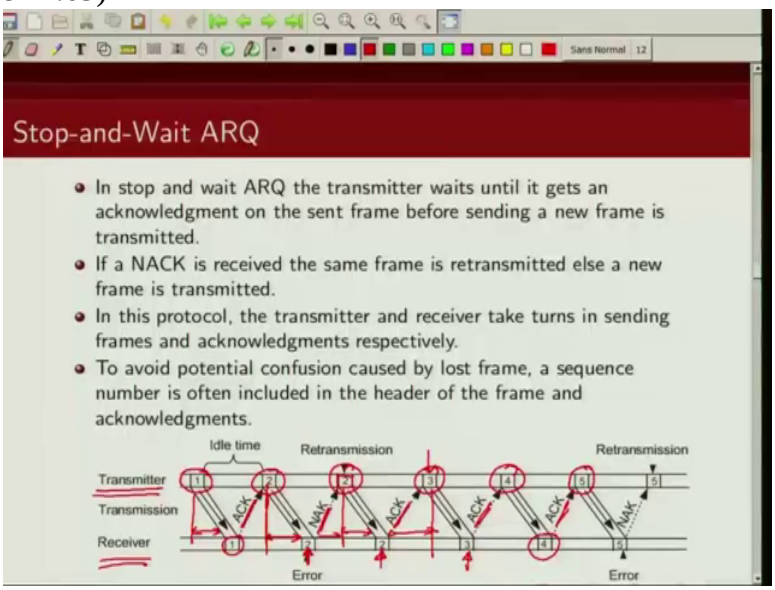
to send a packet number 4. This packet is received here.

(Refer Slide Time 22:01)



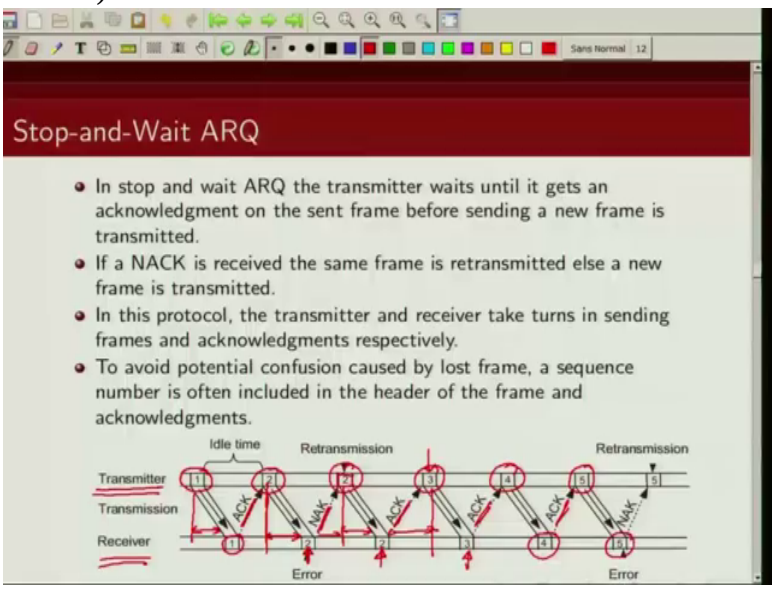
Again this packet is received correctly so the receiver will send an acknowledgement which is received at transmitter at this

(Refer Slide Time 22:09)



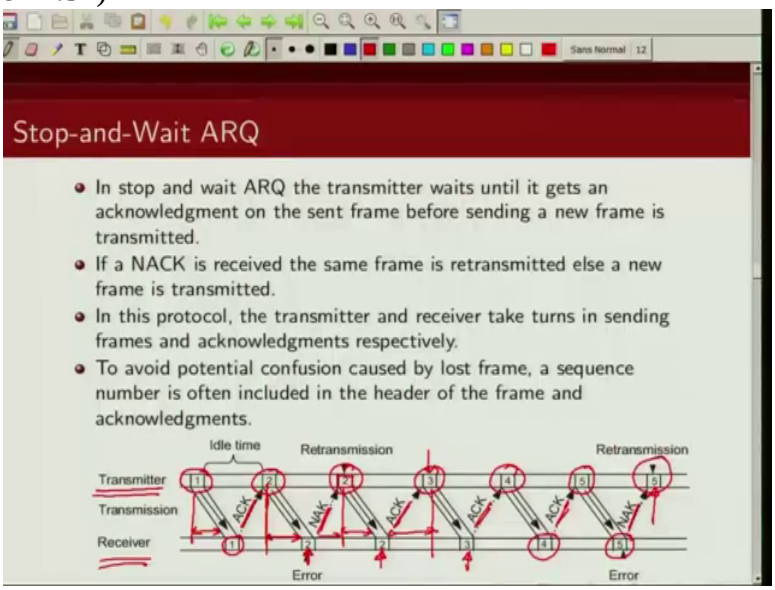
time instance. So the transmitter decides to send packet number 5 which is received by the receiver

(Refer Slide Time 22:16)



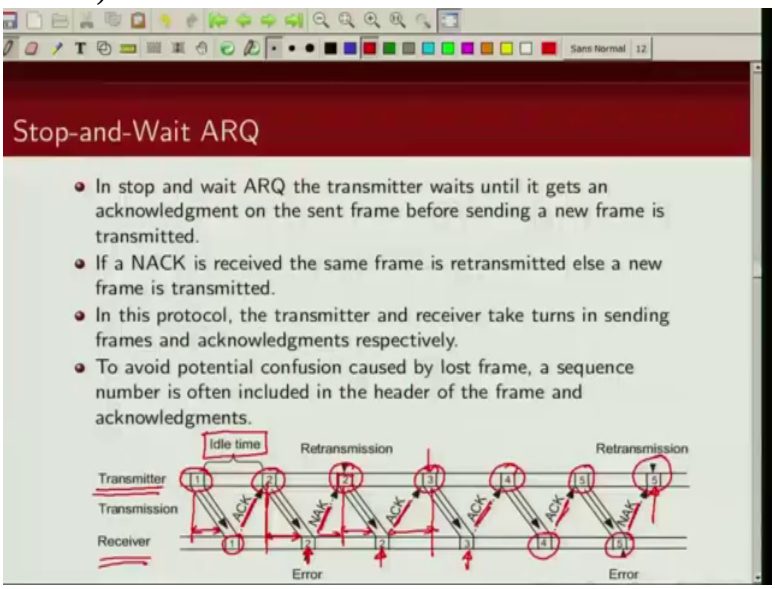
after some delay. Now this packet is not received correctly so the receiver is going to send a negative acknowledgement to the transmitter and after some time delay this negative acknowledgement is received at the transmitter.

(Refer Slide Time 22:34)



Since it is a negative acknowledgement, the transmitter will decide to send the same packet again. So you can see there is an idle time involved here

(Refer Slide Time 22:46)

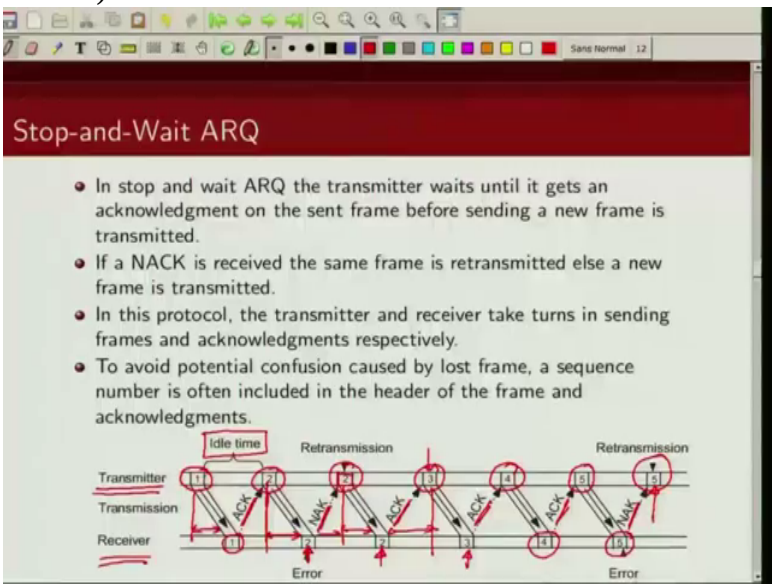


for the transmitter. it sends the packet and waits for the round trip delay

(Refer Slide Time 22:51)

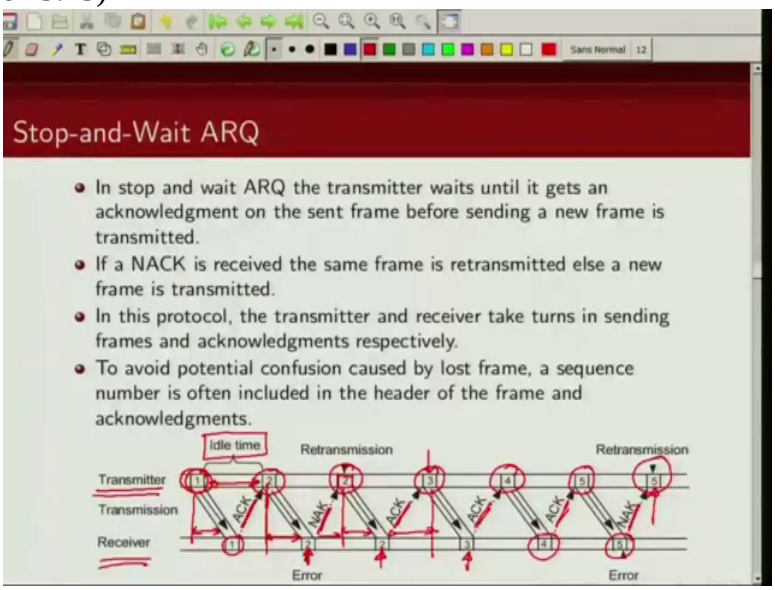
until it gets back acknowledgement or negative acknowledgement from the receiver. So clearly from the point of view of efficiency, this does not seem to be very good scheme. Why? Because

(Refer Slide Time 23:07)



you are transmitting for a very small time but then you are for this idle time you are not sending any packets. But from the

(Refer Slide Time 23:13)

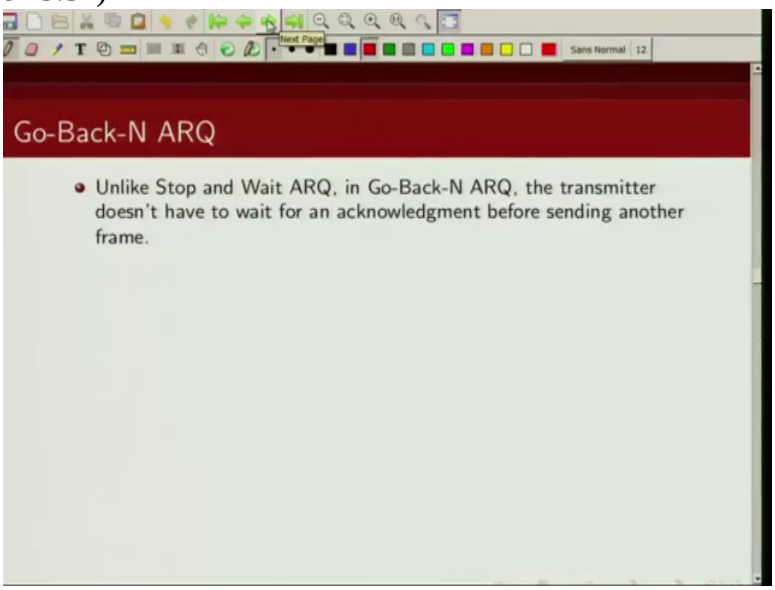


implementation point of view this is pretty

(Refer Slide Time 23:16)

straightforward and simple because we are sending a new packet only when you are sure that the packet you sent earlier has been received correctly. So this is your Stop and Wait A R Q.

(Refer Slide Time 23:31)

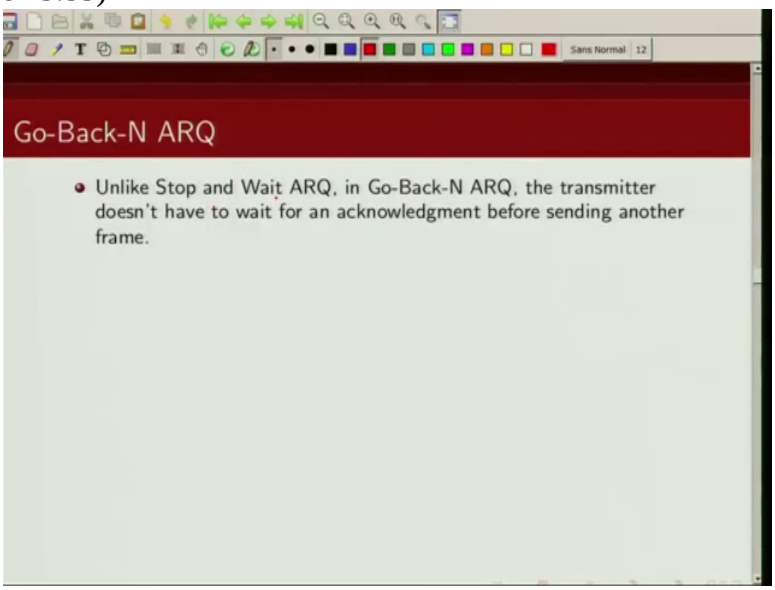


Now as I said the problem with Stop and Wait A R Q is you send a packet you wait for to receive its acknowledgement

(Refer Slide Time 23:40)

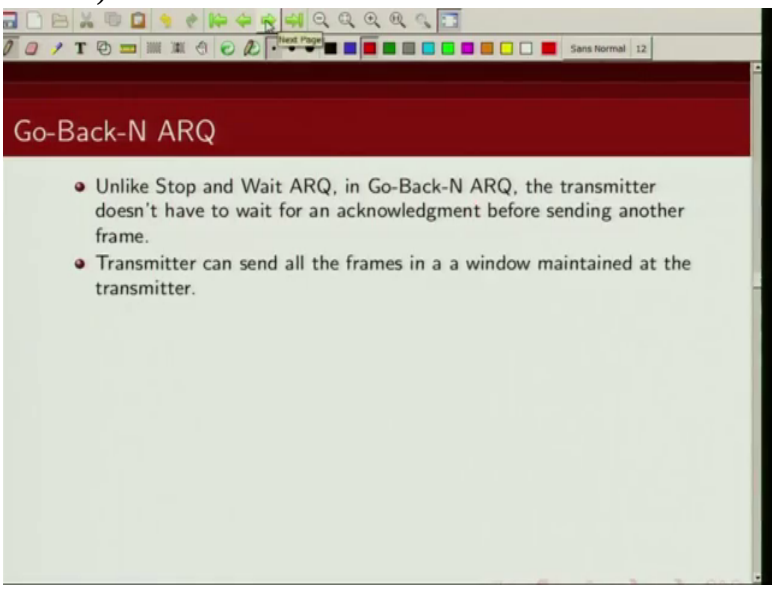
before you send the next packet. So you have this large idle time, possibly large idle time. So this problem has been solved in Go Back N A R Q protocol. So unlike

(Refer Slide Time 23:53)



in Stop and Wait A R Q, in Go Back N A R Q, the transmitter does not wait for an acknowledgement before sending another frame, Ok. So

(Refer Slide Time 24:05)

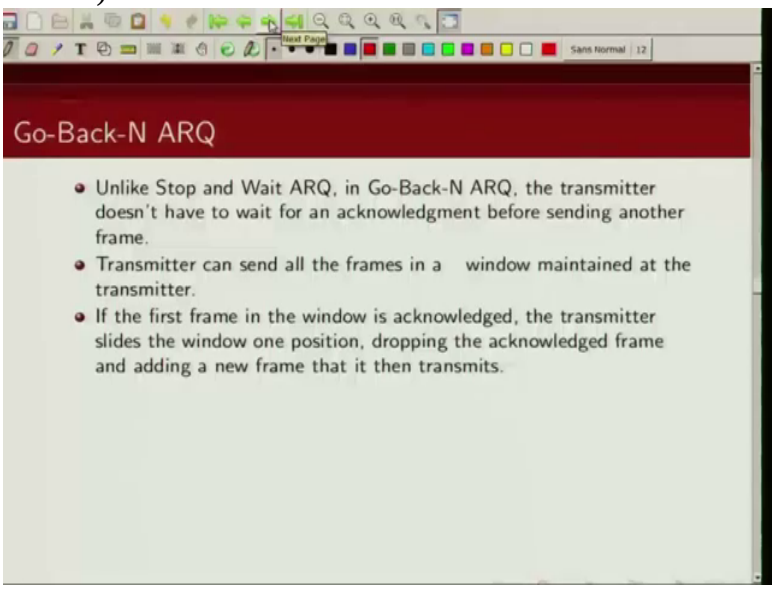


it's like a sliding window kind of protocol, so you can send packet number

(Refer Slide Time 24:09)

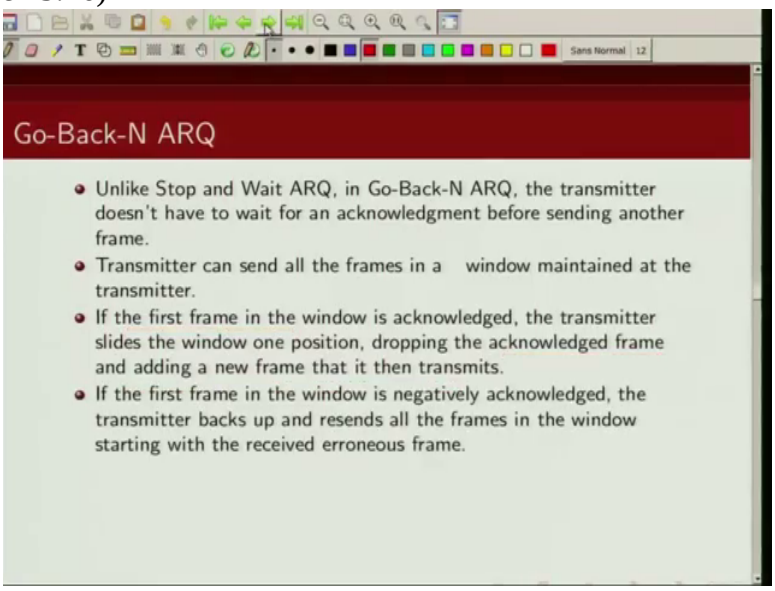
1, 2 and let's see, Ok and in the mean time you send packet 1, 2, 3, 4, 5 up to n. Now you send packet number n plus 1 only when you would have received the acknowledgement for packet number 1. You are going to send packet number n plus 2 only when you get acknowledgement for packet number 2. So you send a block of frames or packets, Ok in this, this particular scheme we do not wait for each individual packet acknowledgement or negative acknowledgement to reach the receiver. We are sending a block of

(Refer Slide Time 24:54)



packets, you can say. Now if the first packet or first frame in this window is acknowledged to be received correctly, what the transmitter does is, transmitter slides this window one position dropping the acknowledged packet or the frame and then adding a new frame that is then transmitted.

(Refer Slide Time 25:16)

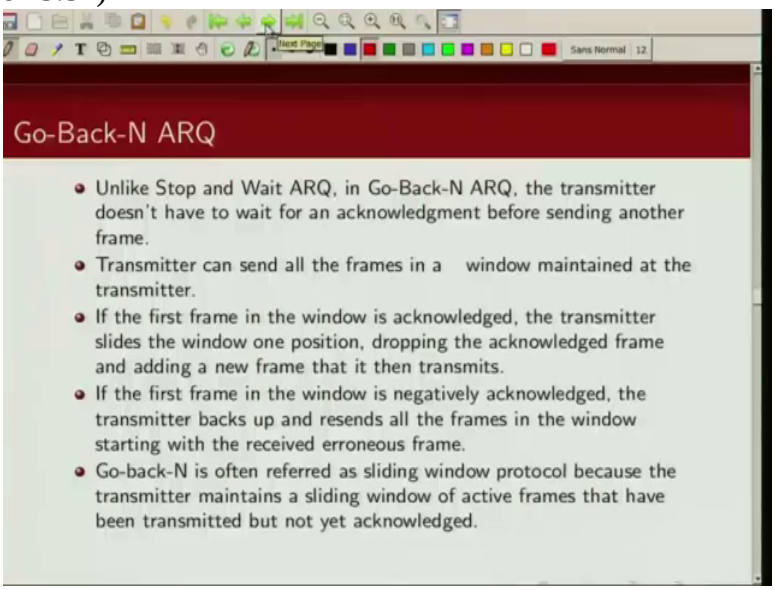


Now what happens if the first frame is, in that window is negatively acknowledgement, that is it is not received correctly? Then the transmitter backs up and resends all the frames in the window starting from

(Refer Slide Time 25:31)

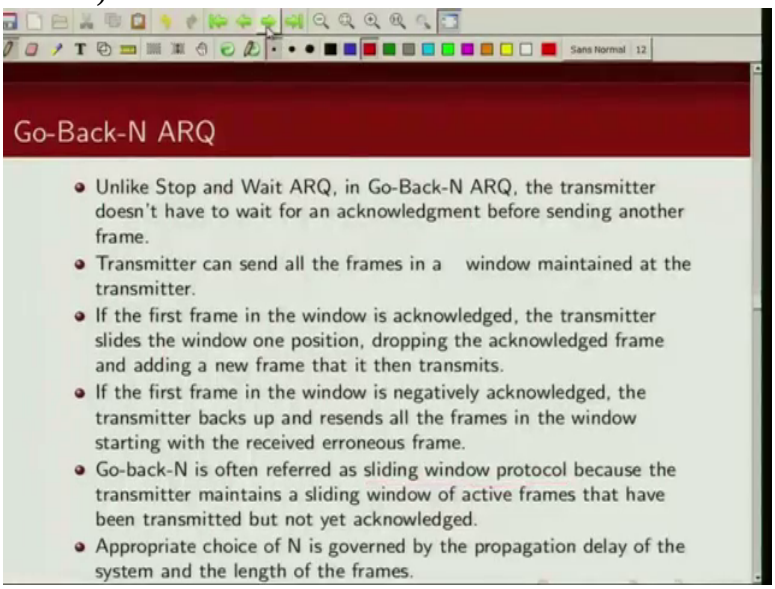
the erroneous received frame. So for example if I am sending packet 1 to n and my fifth packet is in error, I get a NACK corresponding to fifth packet, then fifth onward I am going to start sending those packets again.

(Refer Slide Time 25:51)



So this Go Back N is often referred to as Sliding Window Protocol because transmitter maintains a sliding window active frames that has been transmitted and for whose acknowledgement it is basically waiting to get the acknowledgement for those frames or packets.

(Refer Slide Time 26:08)

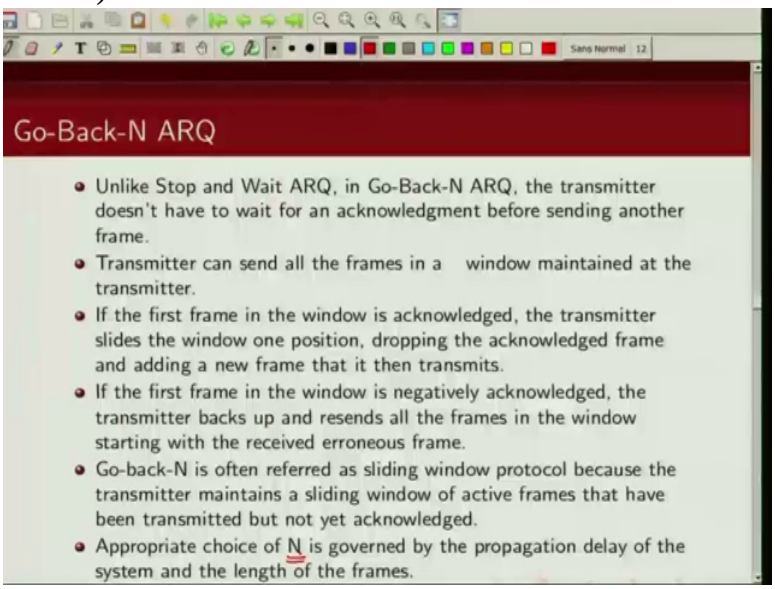


Now as you can see here, this window size or sliding window size n is crucial. Because we don't want

(Refer Slide Time 26:18)

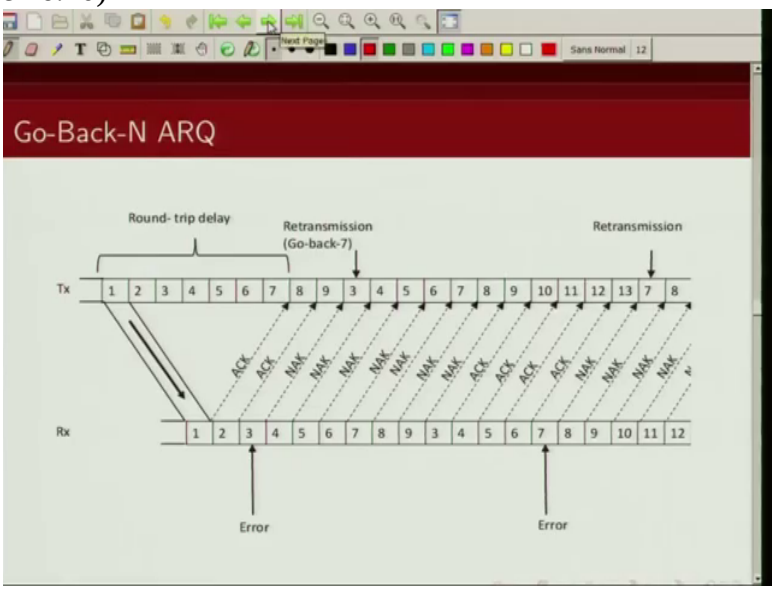
a very special case of n equal to 1 is Stop and Wait. And the other extreme is if n is too large then of course that is also not good. So we need to choose appropriately what n is

(Refer Slide Time 26:32)



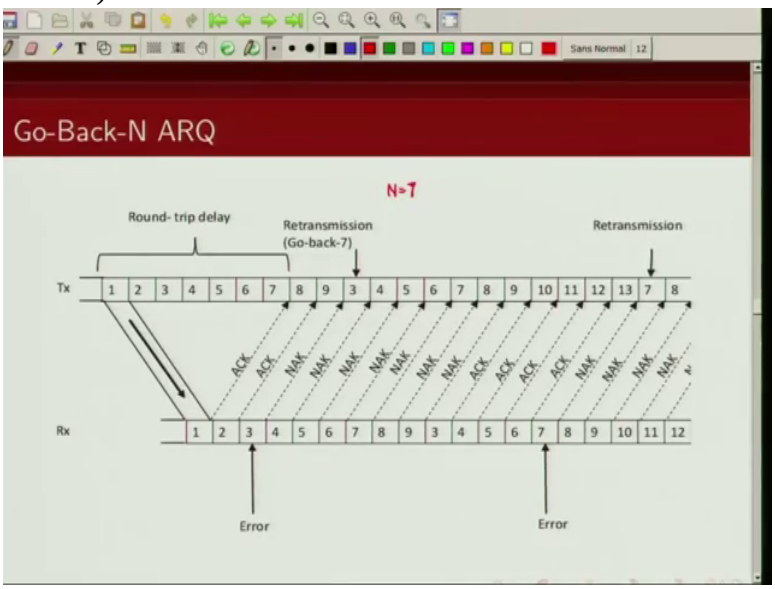
depending on what is the propagation delay of the system or what's the length of the frame that we are sending.

(Refer Slide Time 26:40)



Now I am illustrating Go Back N Protocol using this diagram. So in this case, I am taking n to be 7. So

(Refer Slide Time 26:51)

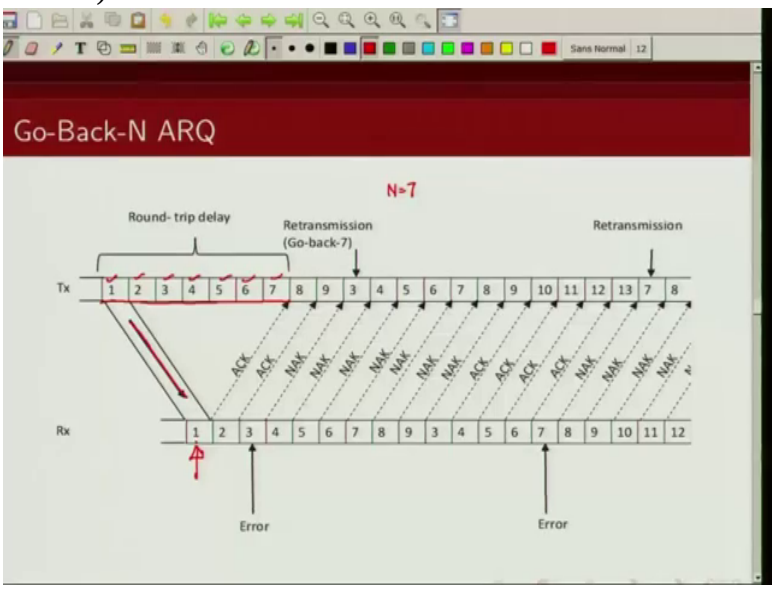


you can see here, I am sending the frame of 7 bits; 1, 2, 3, 4, 5, 6, 7. I am sending these frames. Now what happens is, frame number 1, after some delay is received at this time instance. Now the receiver again will apply C R C check and will try to find out whether is any parity

(Refer Slide Time 27:21)

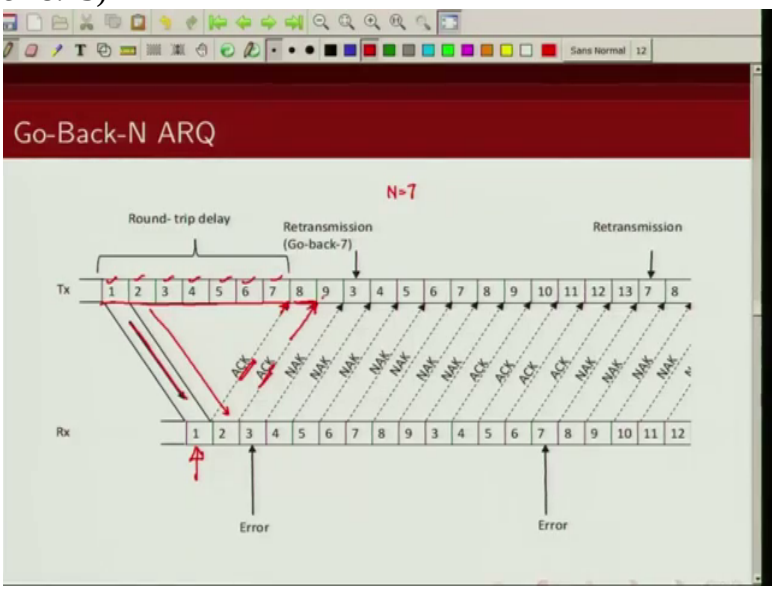
violation or not. If the, if C R C code detects any error, it will send a negative acknowledgement, otherwise it will send a positive acknowledgement. So in this case,

(Refer Slide Time 27:32)



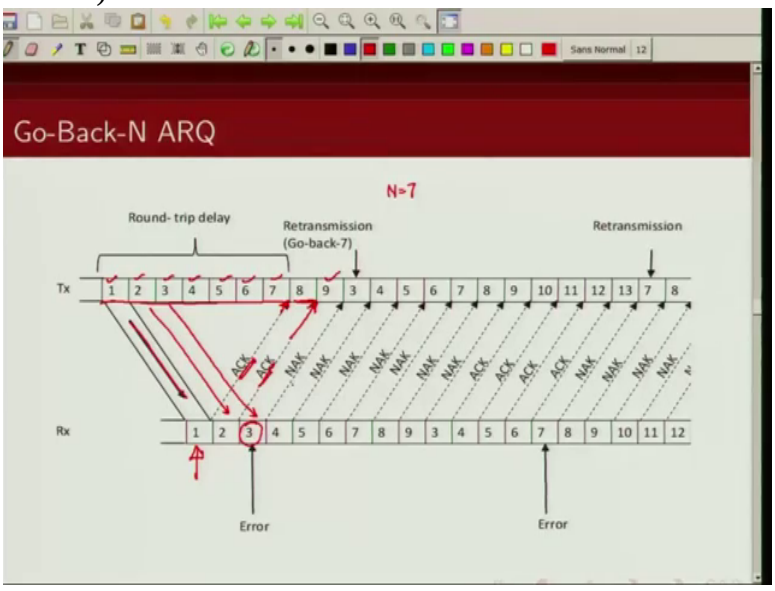
this packet 1 was received correctly. So positive acknowledgement was sent. Now there is a delay in receiving this positive acknowledgement so at this particular time instance the transmitter receives a positive acknowledgement about frame number 1. Remember I said here we are sending a block of n bits. Now we are going to send n plus 1 frame only when we get the acknowledgement about frame number 1. So at this time instance we are getting an acknowledgement of the frame number 1, so this is the acknowledgement, so we are going to send packet number 8. Now in the mean time the second packet was sent and this was received correctly. So at this instance we get,

(Refer Slide Time 28:23)



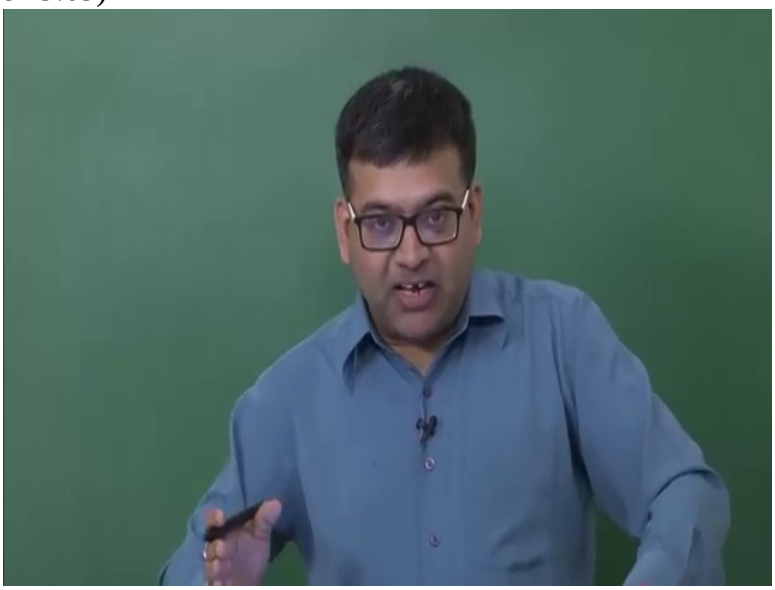
the transmitter gets an acknowledgement that the second packet has been received correctly. So it transmits a new packet which is a packet for frame number 9. Next this 3 has been received. Now this 3 has been received in error.

(Refer Slide Time 28:43)



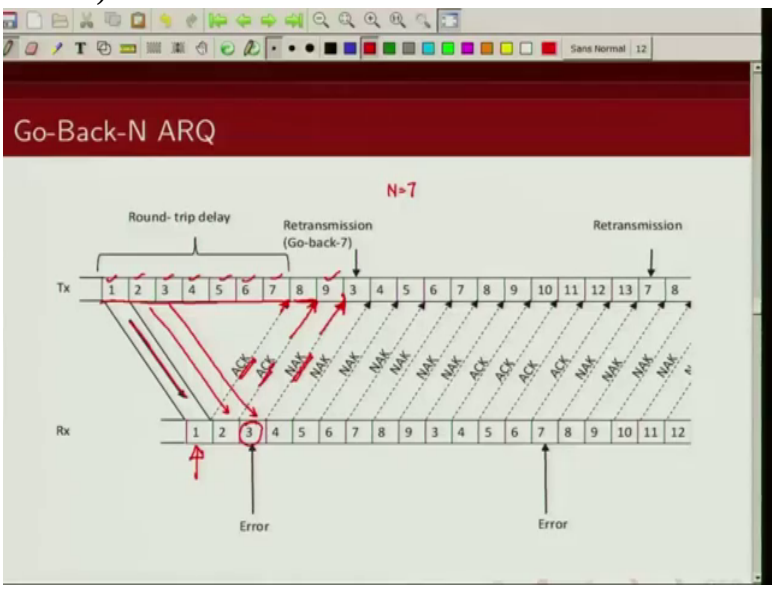
So this receiver is going to send a negative acknowledgement. So receiver sends a negative acknowledgement which is received at the transmitter at this time instance. Now since this is a negative acknowledgement, as I said if you get intimation about an incorrect frame received we are going to,

(Refer Slide Time 29:09)



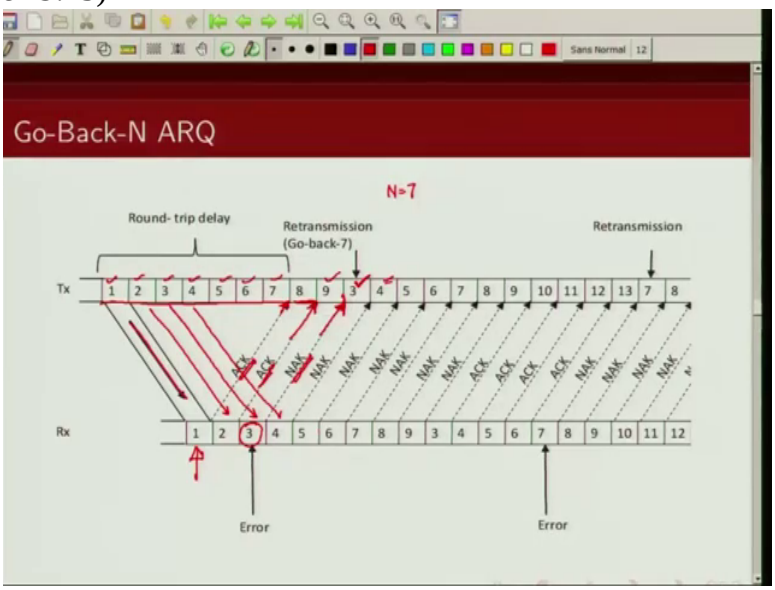
the transmitter is going to back off and start sending again from the packet which has been erroneously received. In this case

(Refer Slide Time 29:19)



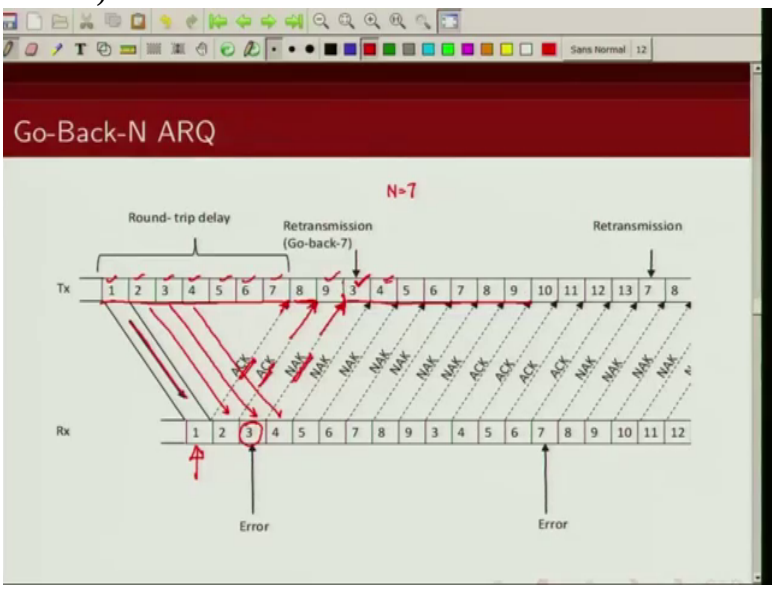
the third packet was not received correctly. So we will start sending again from packet number 3. So at this time instance I am going to send packet, frame number 3, Ok now similarly you get a 4 here, you have received correctly so you send, now the next packet after 3 is 4, so like that basically we are going to send

(Refer Slide Time 29:45)



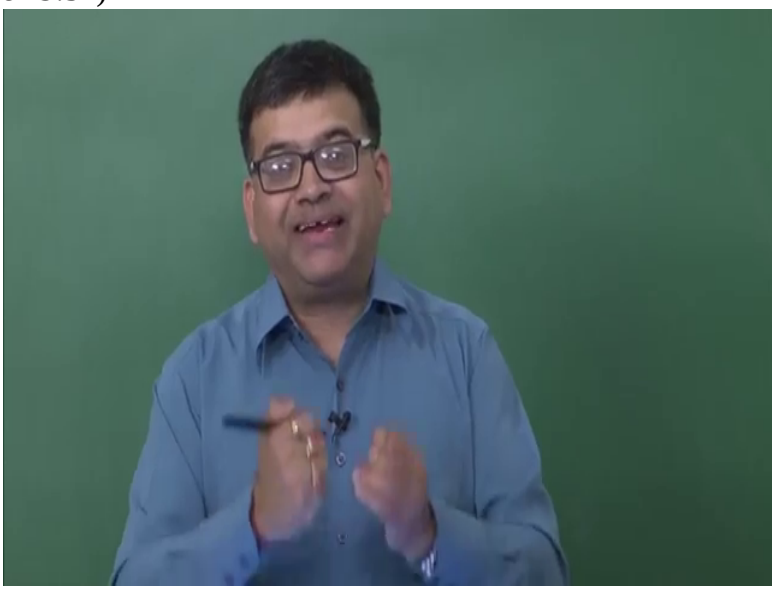
next 7; 3,4, 5,6, 7,8,9 we are going to send. So in

(Refer Slide Time 29:52)



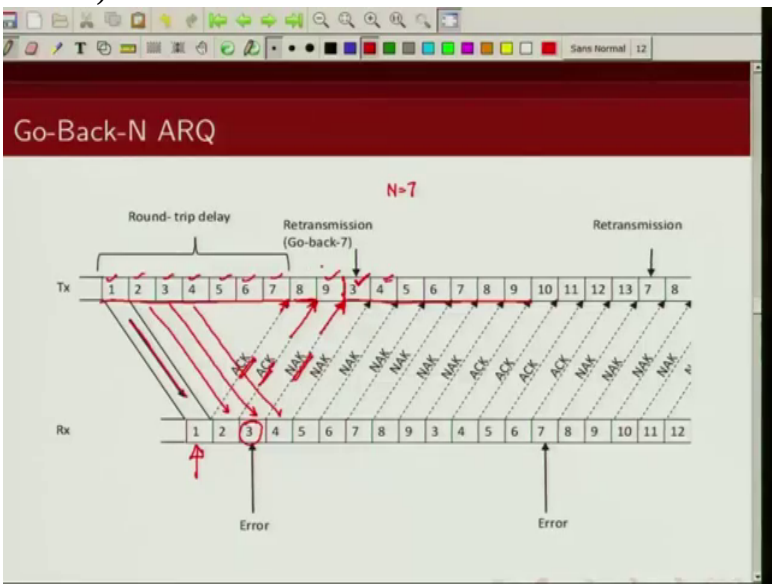
Go Back N what we do is whenever we receive a packet

(Refer Slide Time 29:57)



in error, we start retransmitting from this packet. So in this case, we receive packet number 3 in error. We start retransmitting from packet number 3, 4, 5. So it may so happen that earlier we sent packet

(Refer Slide Time 30:14)

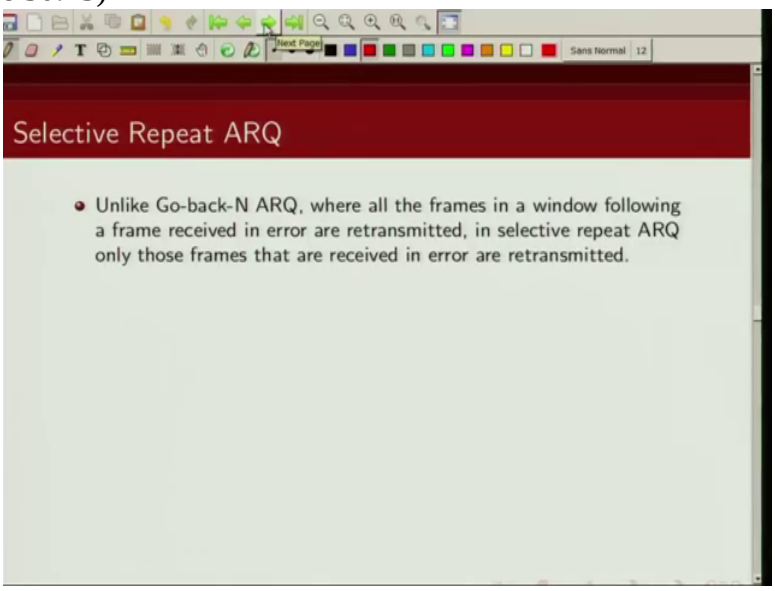


number 1 to 7; 4, 5, 6 might have been received correctly but we ignore them.

(Refer Slide Time 30:19)

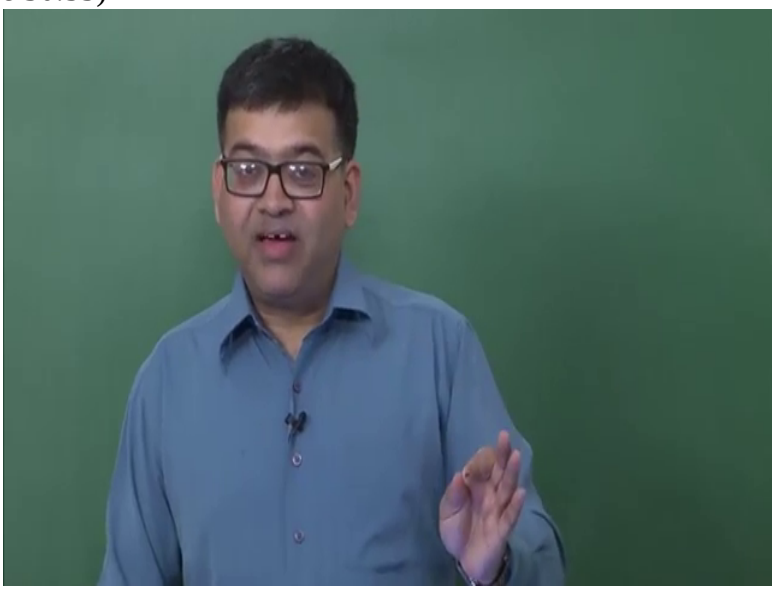
We again retransmit the packet. So in terms of efficiency this is still not very good because we are wasting, we are retransmitting some packets which we might have received correctly. So the other extreme is what is known as Selective Repeat A R Q.

(Refer Slide Time 30:43)



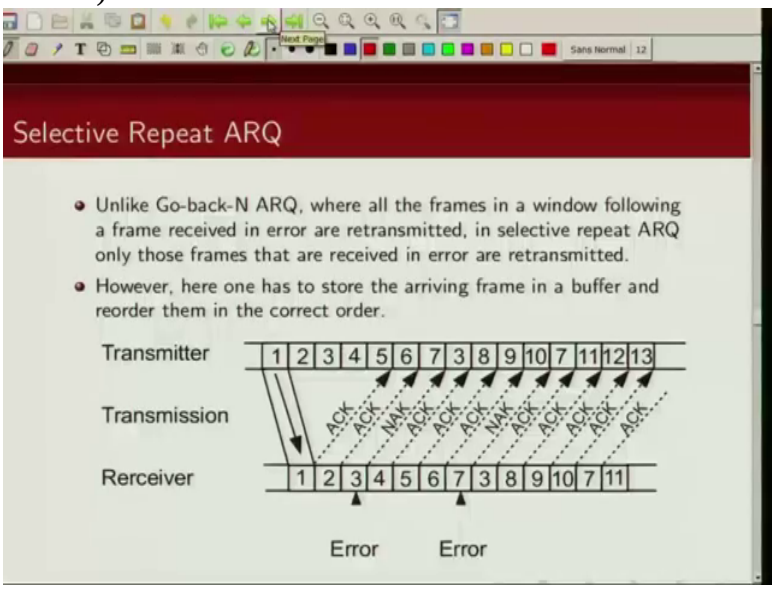
So in this case, when a frame is received in error, we are only going to send those frames which have been received in error. So unlike

(Refer Slide Time 30:55)



Go Back N, in Go Back N what were we doing? If a packet number i was received in error, we start retransmitting again from packet i; i, packet number i plus 1, i plus 2, i plus 3. But in Selective Repeat if a packet i is not received correctly we are only going to send, retransmit packet i. We are not going to retransmit packet i plus 1 or i plus 2 if they were received correctly, Ok. So we are selectively only retransmitting those packets which are not received correctly by the receiver.

(Refer Slide Time 31:34)

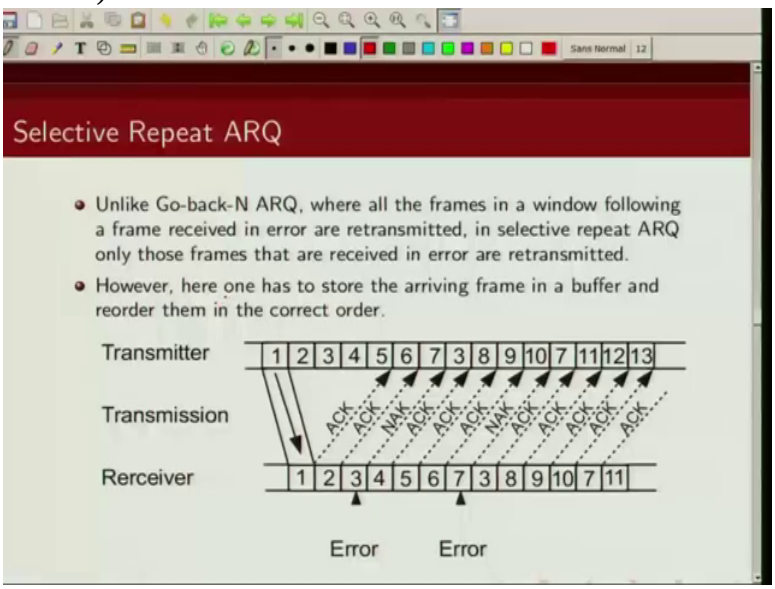


What is the drawback of the system? Now since we are only selectively asking for retransmission of only those packets which are in

(Refer Slide Time 31:45)

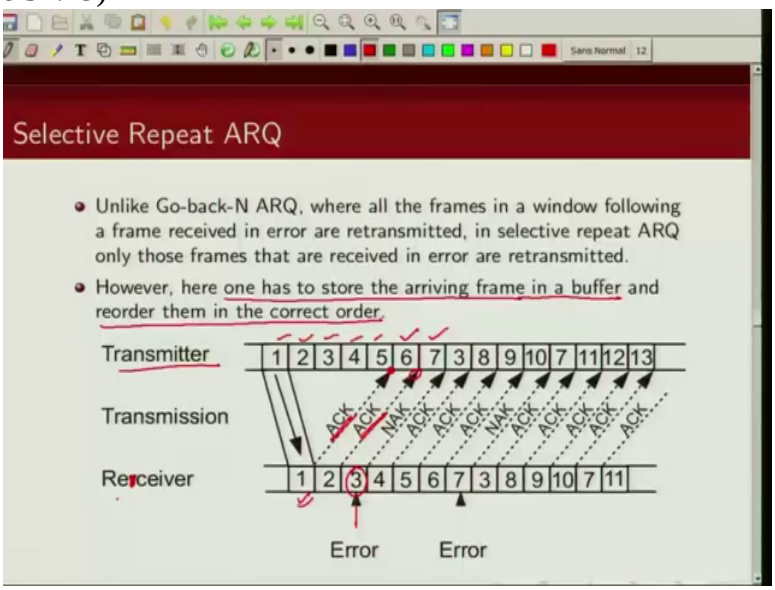
error, we need to maintain some buffer so that once we receive those erroneous packets we can put them back in the right order and deliver it to the intended receiver, Ok. So

(Refer Slide Time 32:00)



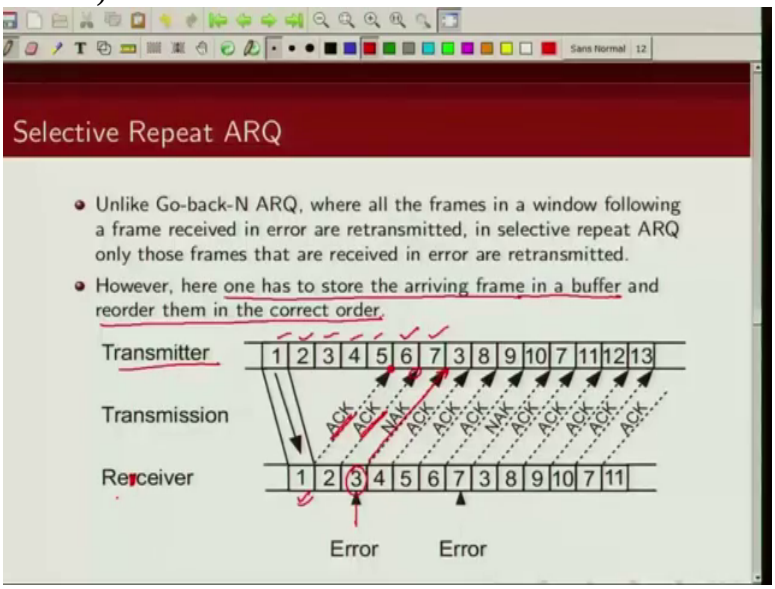
the drawback of selective repeat system is one has to store these arriving frame in a buffer and reorder them in the correct order so that they are delivered to the recipient in the correct order. So you can see here, so you have, this is my transmitter, this is my receiver Ok, so you send packet 1, packet 1 is received correctly so you send an acknowledgement, so you are sending packet 1,2, 3, 4, 5 now here you receive an acknowledgement for packet number 1, so you send next packet, packet number 6, here you are receiving acknowledgement for packet number 2, you send the next packet 7, in this case packet number 3 was received in error.

(Refer Slide Time 32:49)



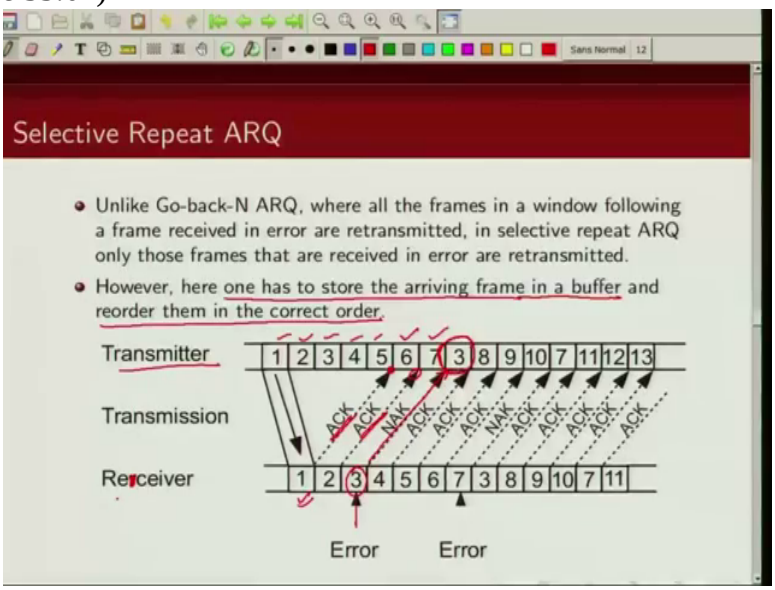
So you, at this time instance you are getting a negative

(Refer Slide Time 32:54)



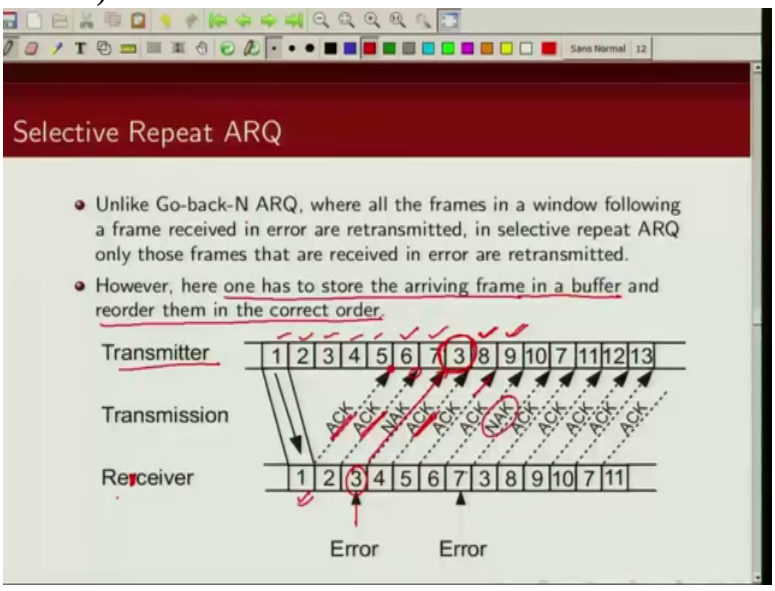
acknowledgement of packet number 3. So what this transmitter will do is it will retransmit packet

(Refer Slide Time 33:02)



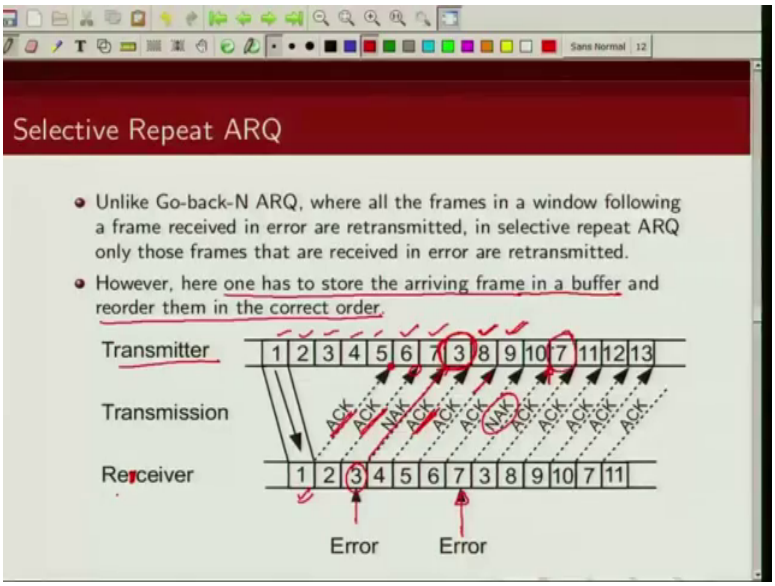
number 3. Now note the difference from Go Back N. In Go Back N what we would have done was we would have started retransmitting packet 4, 5, 6. Here since packet 4 is received correctly we don't retransmit it. We send the next packet. We have already sent up to packet number 7, so now we are going to send packet 8. So this is the difference from Go Back N. In Go Back N we start retransmitting again from the incorrectly received packet. Now here you get acknowledgement for packet number 5, so you send the next packet which is packet number 10. Now here you see acknowledgement for sixth, look at this example again.

(Refer Slide Time 33:45)



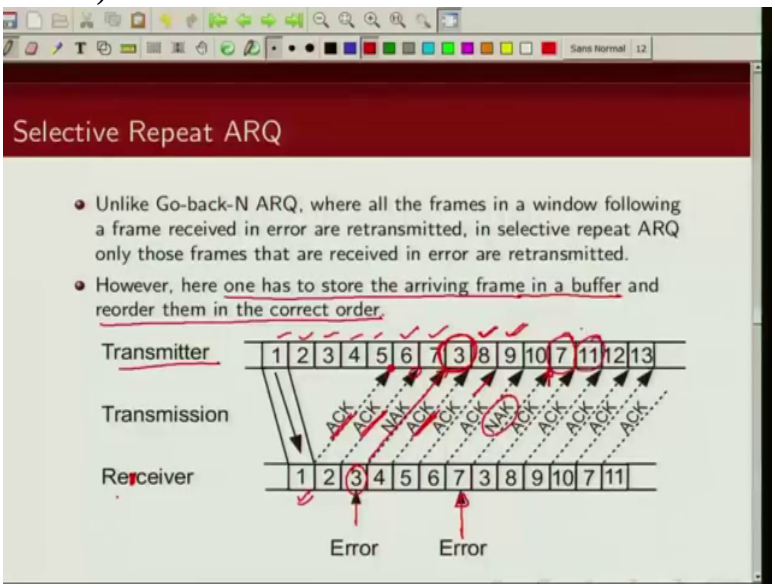
So the seventh bit, seventh frame was not received correctly so you send a negative acknowledgement for this instance, the transmitter receives a negative acknowledgement. So what you are going to do is you are going to insert this

(Refer Slide Time 33:58)



frame number 7 which was incorrectly received here and continue. Subsequently you received an acknowledgement about the packet that you sent so

(Refer Slide Time 34:13)

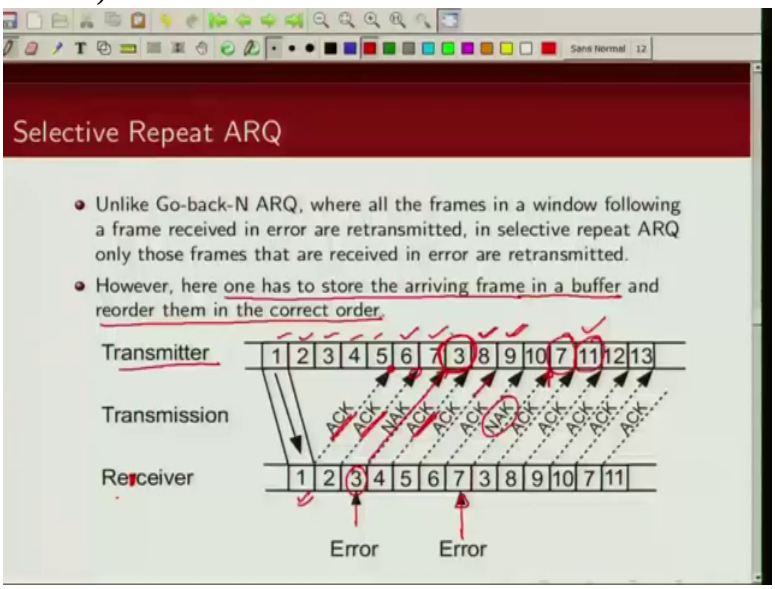


you send the new packet, packet number 11. So you can see in Selective Repeat

(Refer Slide Time 34:20)

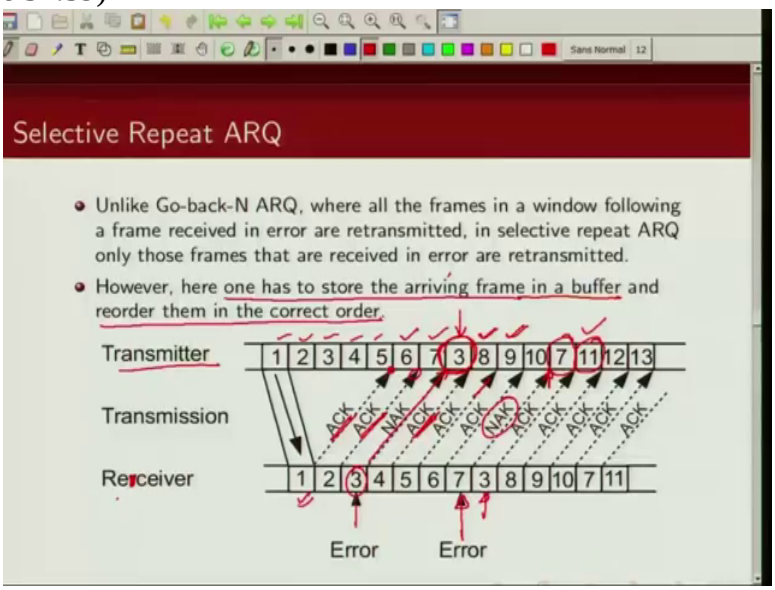
A R Q, we are selectively retransmitting only those packets which are not received correctly. So you can see here I am

(Refer Slide Time 34:32)



receiving packet number 3 here correctly because this was not, I mean, actually received packet number 3 correctly at the receiver because

(Refer Slide Time 34:39)

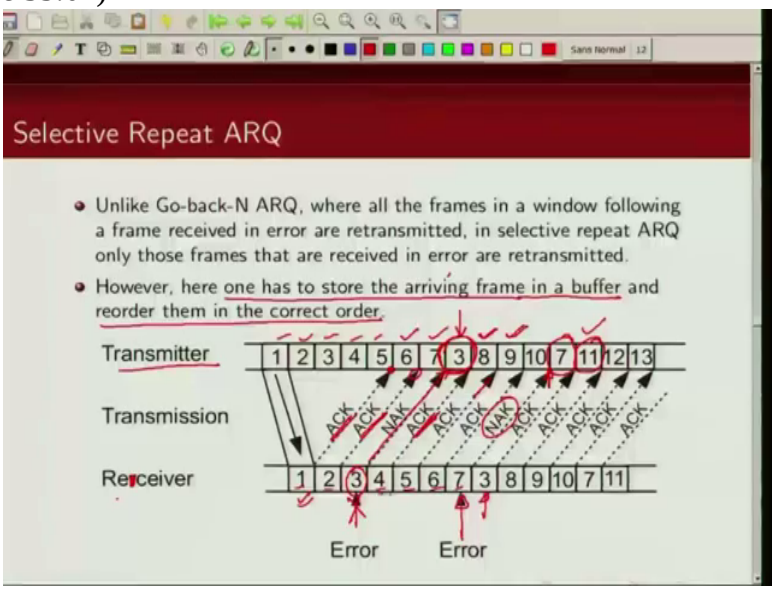


this was not incorrect. In the mean time, I have already received correctly packet number 1, 2, 4, 5, 6, 7. So once I get packet number 3 or frame number 3, I need to put it back in the order after

(Refer Slide Time 34:54)

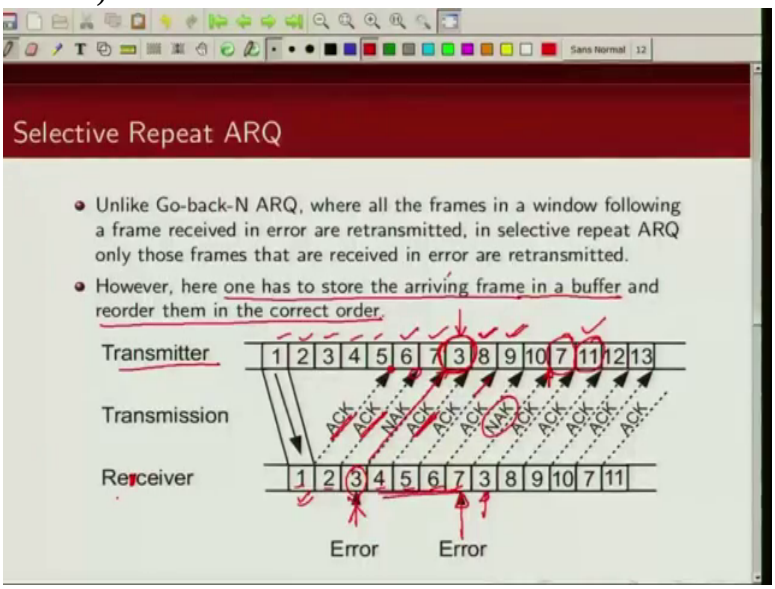
packet number 2 and before packet number 4. So that's what I meant; at the receiver you need to keep buffer to save

(Refer Slide Time 35:01)



or store

(Refer Slide Time 35:02)

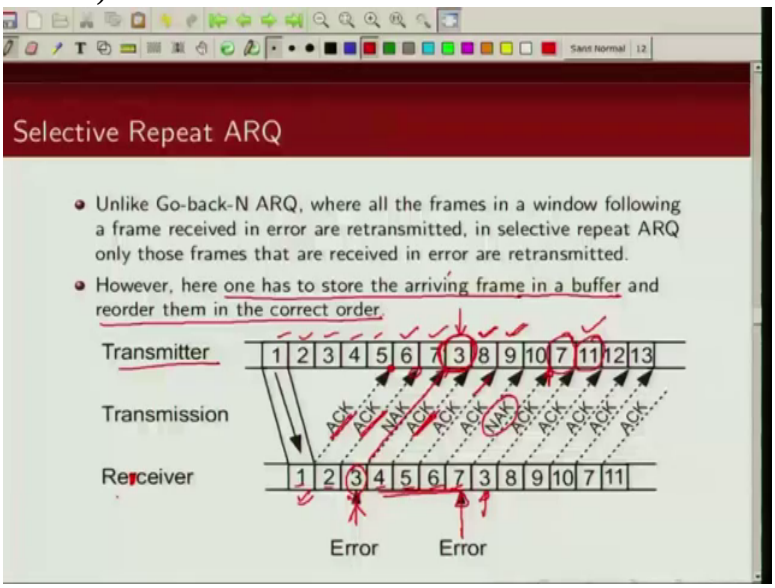


all the packets that you have received before the, this packet number 3 you have received, so 4, 5,6 so you need to store them in the buffer and once you get packet number 3, you put it back in the order after packet number 2. So these are the three protocols you can see, Stop and Wait very inefficient but very simple

(Refer Slide Time 35:22)

protocol, Selective Repeat from efficiency point of view very good but from the complexity point of view the most complex and Go Back N is a in-between, I mean efficiency more than [Selective Repeat/Stop and Wait] Stop and Wait but then

(Refer Slide Time 35:38)



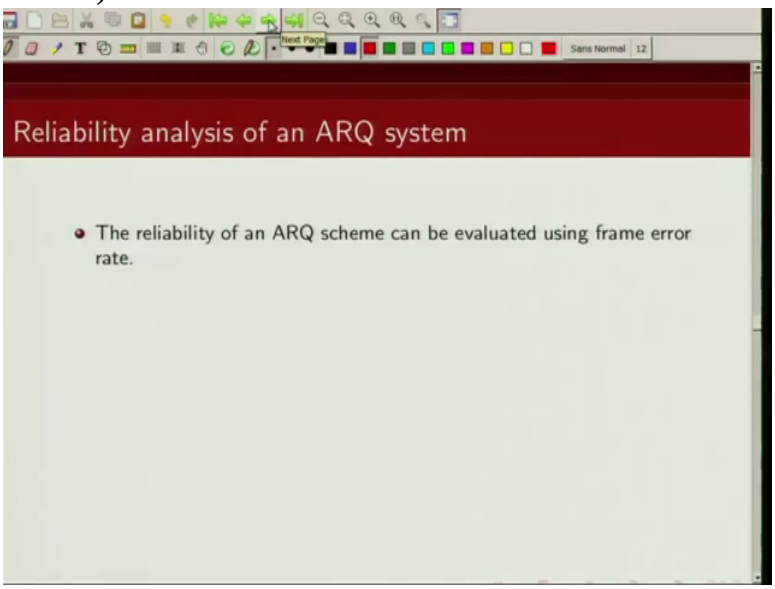
it is easier than Selective Repeat Request scheme.

So now we are

(Refer Slide Time 35:44)

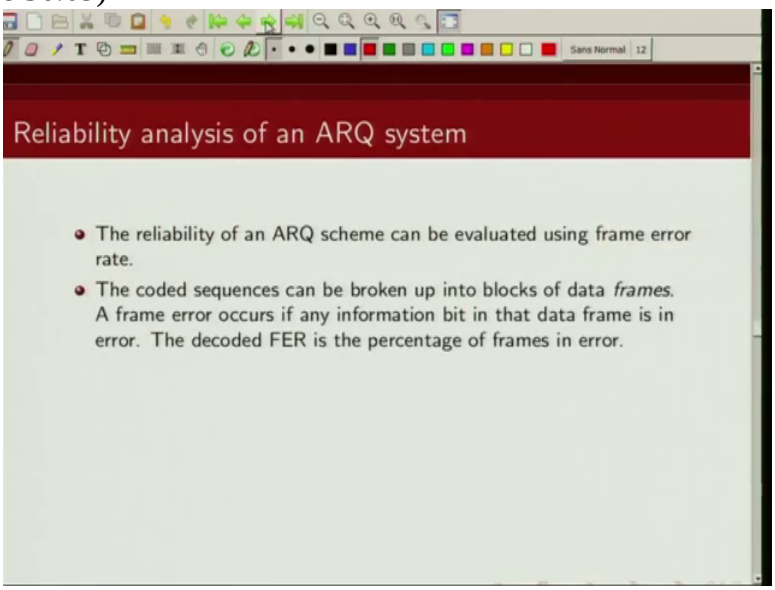
going to talk about performance evaluation of these A R Q protocols. So the first thing we are going to talk about is reliability of

(Refer Slide Time 35:53)



these A R Q schemes. So how can we evaluate the reliability of these A R Q schemes? We can evaluate their reliability using frame error rate. So when you send a packet and

(Refer Slide Time 36:09)

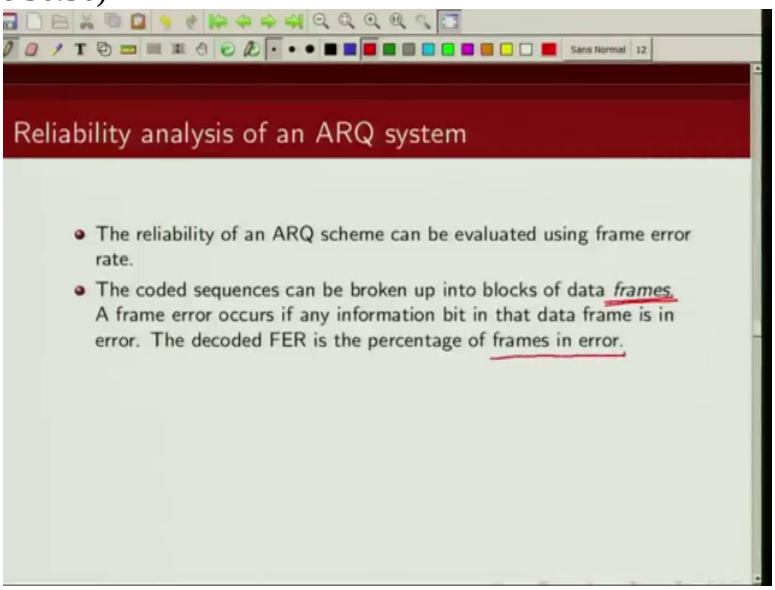


the packet is received in error, we say the frame or the packet is in error, right. So what we are doing is we are sending this coded sequence into small, small blocks of data which are known as frames or packets and the frame error happens if any of the information bit in that frame is

(Refer Slide Time 36:30)

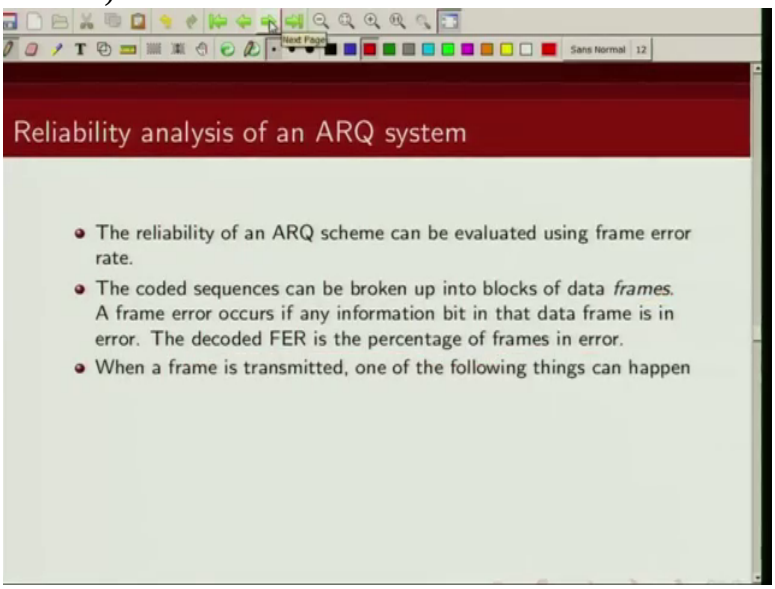
in error. And the percentage of frames which are in error is known as

(Refer Slide Time 36:36)



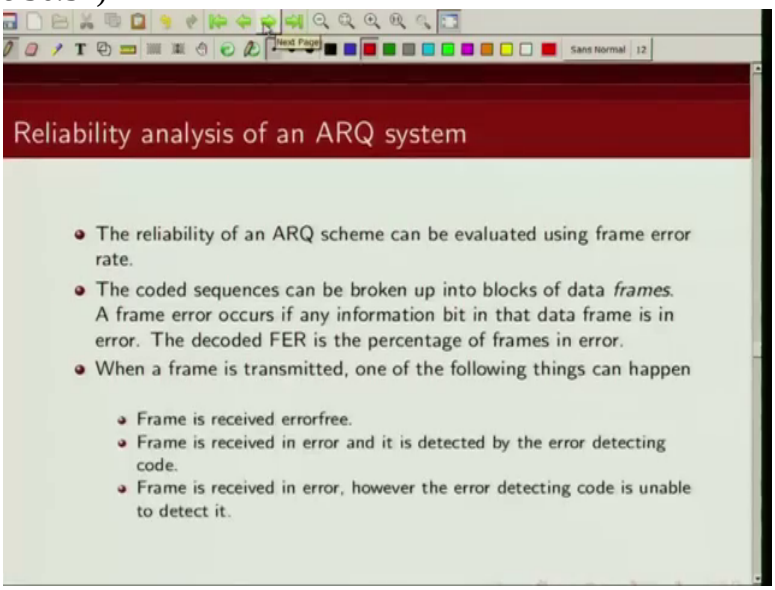
frame error rate. So that's the decoding frame error rate, it is percentage of frames in error.

(Refer Slide Time 36:44)



Now what happens when we are sending a frame? What happens when a transmitter sends a frame?

(Refer Slide Time 36:51)

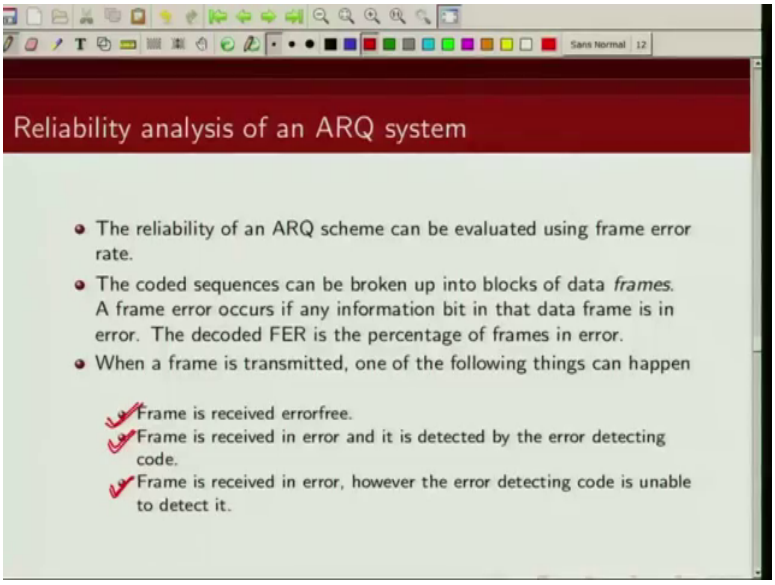


So there are 3 possibilities. The first possibility is the frame is received correctly. So there has been no transmission errors. Whatever is sent exactly the same thing is received. So the frame is received error free. The second possibility is frame is received in error and the error detecting code that we are using is able to detect that there is an error. So that's option number 2. And the third option is frame is received in error however the error detecting code

(Refer Slide Time 37:35)

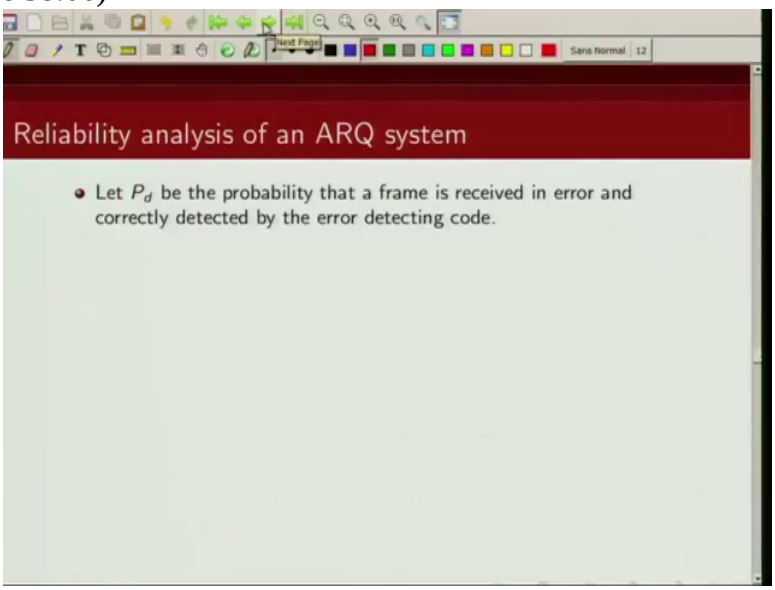
that we use is unable to detect that error. So when does that situation happen? We call; we say an undetected error has happened. An undetected error happens if the transmitted codeword gets transformed into some other codeword. So the error pattern is such that it transforms one codeword to the, some other codeword and then we would not be able to detect the error.

(Refer Slide Time 38:04)



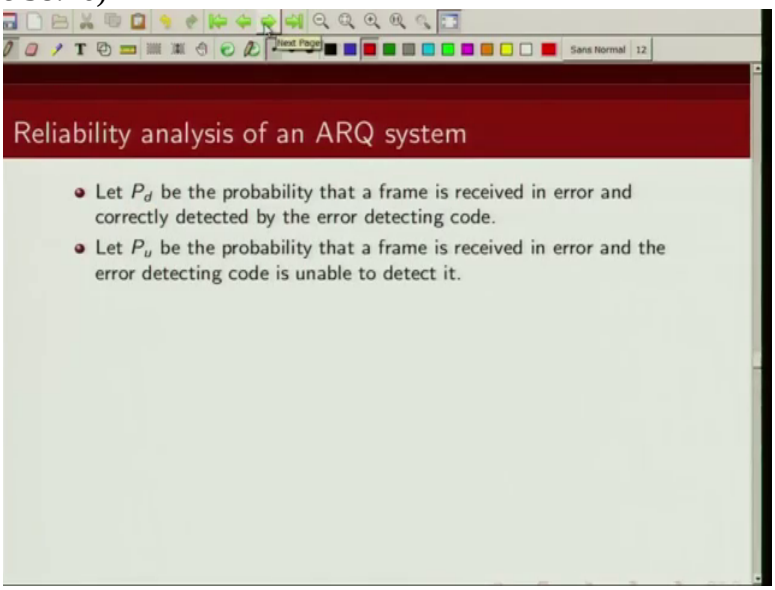
So that's the third possibility. So

(Refer Slide Time 38:06)



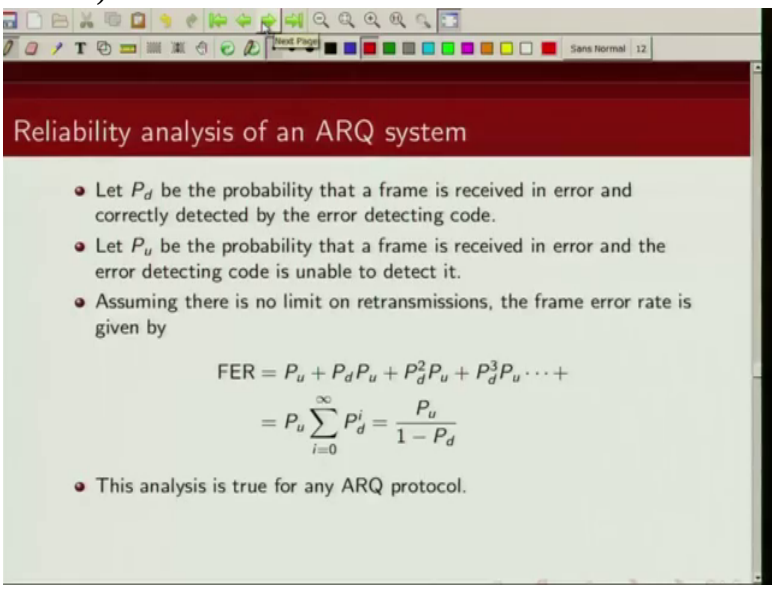
let P d be the probability that the frame is received in error and it is correctly detected by the error detecting code

(Refer Slide Time 38:16)



and let P u be the probability that the frame is received in error but the error detecting code is unable to detect it.

(Refer Slide Time 38:29)



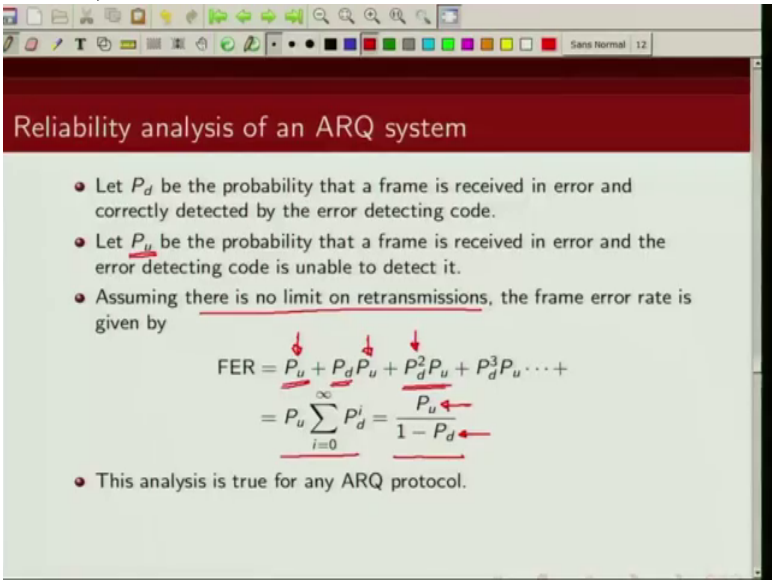
Now we are trying to find out an expression for frame error rate for this A R Q protocol. So let's assume that there is no limit on retransmission. So whenever we get a negative acknowledgement we will ask for retransmission. So when does the frame error occur? A frame error occurs if we accept a packet which is in error but the error detecting code is not able to detect those error and that probability is given by P u, Ok. So it can happen that the first time that we send the packet the error detecting code is not able to detect the error and it accepts the packet. So this is given by this probability. Or it may happen that first time instance it was able to detect the packet but when we retransmitted it for the first time, the

error detecting code was not able to detect the error. That probability is given by P d times P u or it may so happen that two times, you were able correctly detect the error but the third time, you were not able to detect the error. So that probability is given by this. Similarly we can write P d cube, P u plus P d 4 P u like that. So this can be written in this form which is, can be written as geometric series, so which can be written as P u divided by 1 minus P d. Now this is true for any A R Q protocol. Now so the frame error rate depends on P u and P d. So P d is the probability that it is able to

(Refer Slide Time 40:26)

correctly detect that the frame is in error. So that depends on the error detecting capability of the error detecting code that we are using. Similarly this

(Refer Slide Time 40:39)



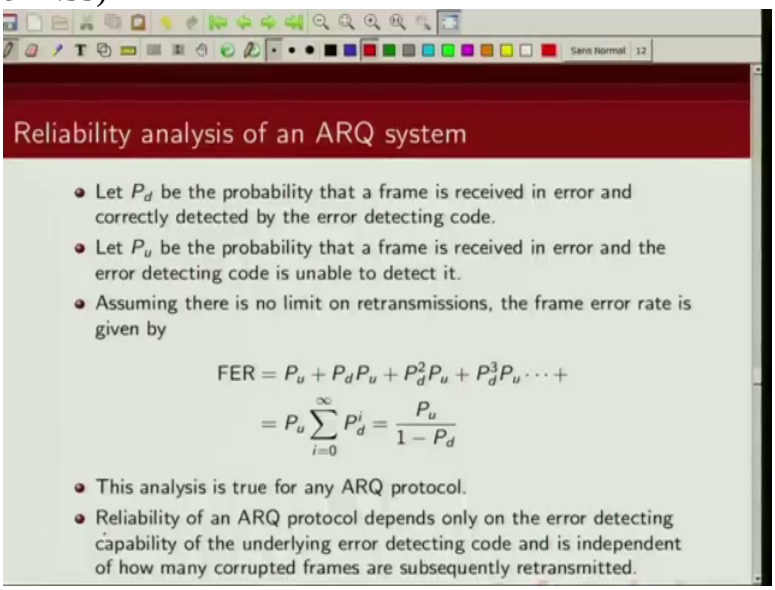
undetected error probability is also related to the weight distribution of the error detecting code that we are using. In fact we have, when we talked about linear block code, we talked about what's the probability of undetected error because we are dealing with linear block code. So if you are sending all zero codewords and what we receive

(Refer Slide Time 41:02)

a nonzero codeword then there is an error. So we can use the weight distribution of a linear block code to find out the probability of undetected error. So you can see that frame error rate performance of this A R Q protocol does not depend on what we are transmitting or how we are transmitting. It only depends on the error detecting code that we are choosing.

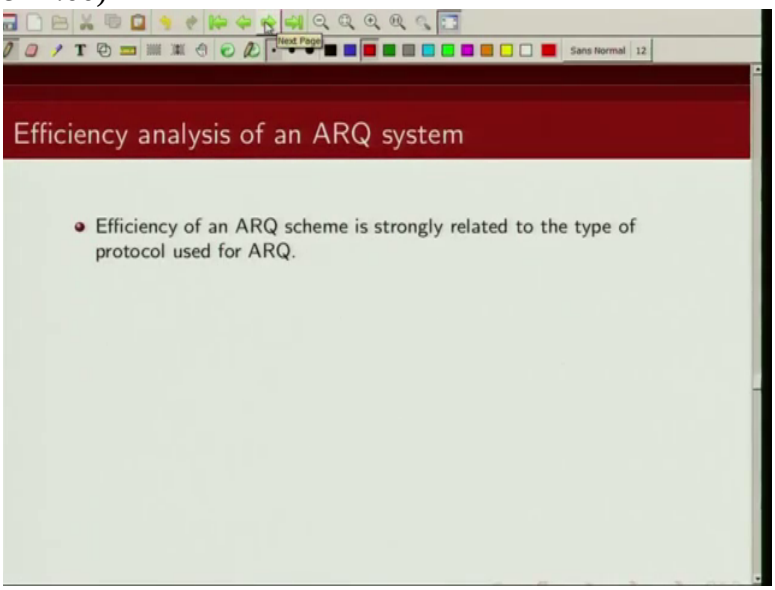
So

(Refer Slide Time 41:35)



to summarize the reliability of A R Q protocol which we are determining on the basis of frame error rate depends only on the error detecting capability of the underlying error detecting code and it is independent of how these corrupted frames or packets are subsequently retransmitted.

(Refer Slide Time 42:00)

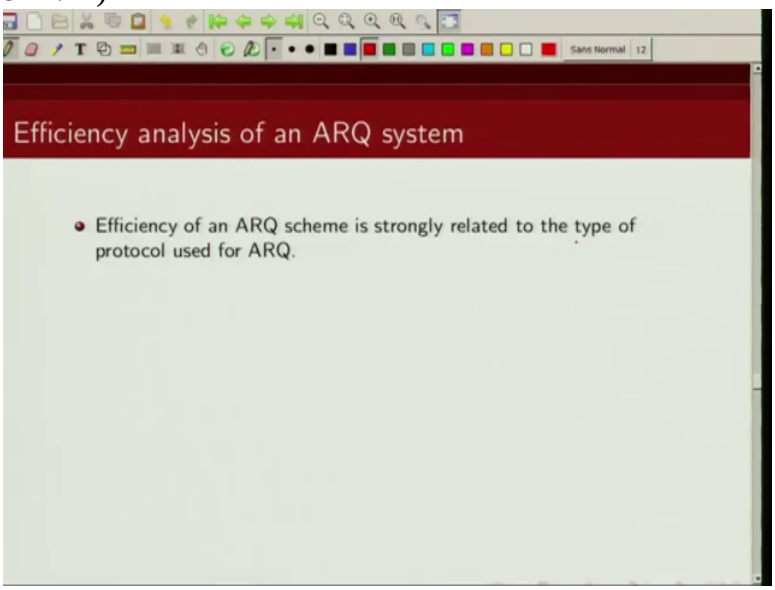


Next we are going to move into efficiency analysis of these A R Q protocols. So how are we going to measure efficiency

(Refer Slide Time 42:10)

of this A R Q scheme? So we are going to do that on the basis of throughput that we can achieve when we are using these A R Q Protocols. And this

(Refer Slide Time 42:22)

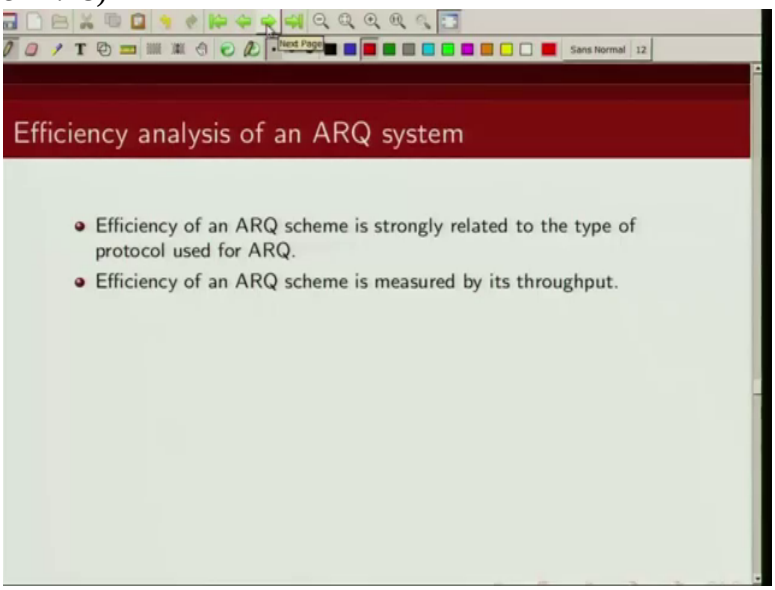


efficiency is very closely related to what sort of protocols that you are using. As you can see we saw in Stop and Wait, for a long time, idle time

(Refer Slide Time 42:33)

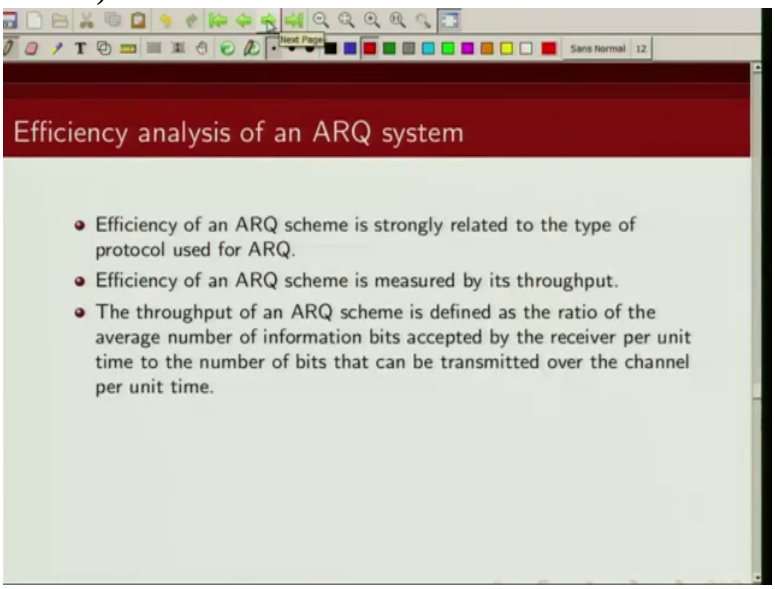
we are just waiting. We are not transmitting anything, so efficiency is very low. On the other stream, Selective Repeat has the best efficiency because we are only retransmitting those packets which are received incorrectly.

(Refer Slide Time 42:48)



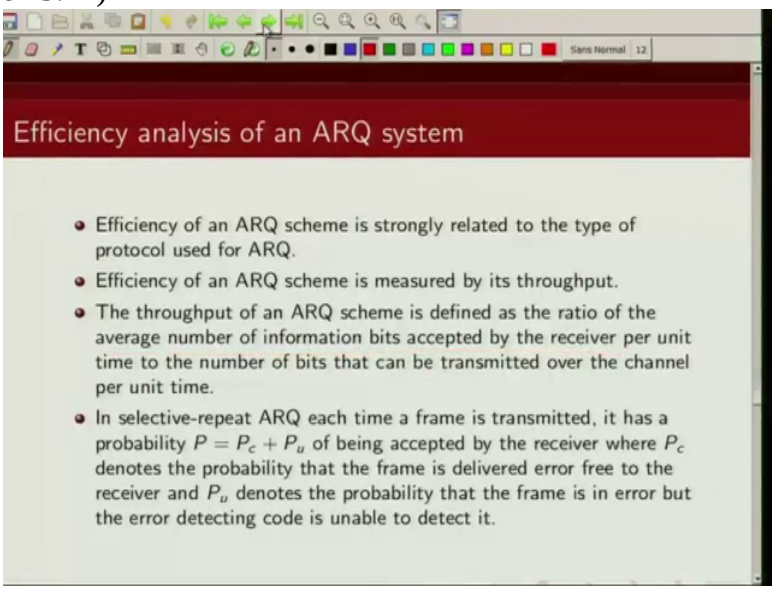
So we are measuring or we are quantifying the efficiency of an A R Q scheme using its throughput.

(Refer Slide Time 42:58)



So the throughput of an A R Q scheme is defined as the ratio of the average number of information bits accepted by the receiver per unit time to the number of bits that can be transmitted over a channel per unit time.

(Refer Slide Time 43:17)

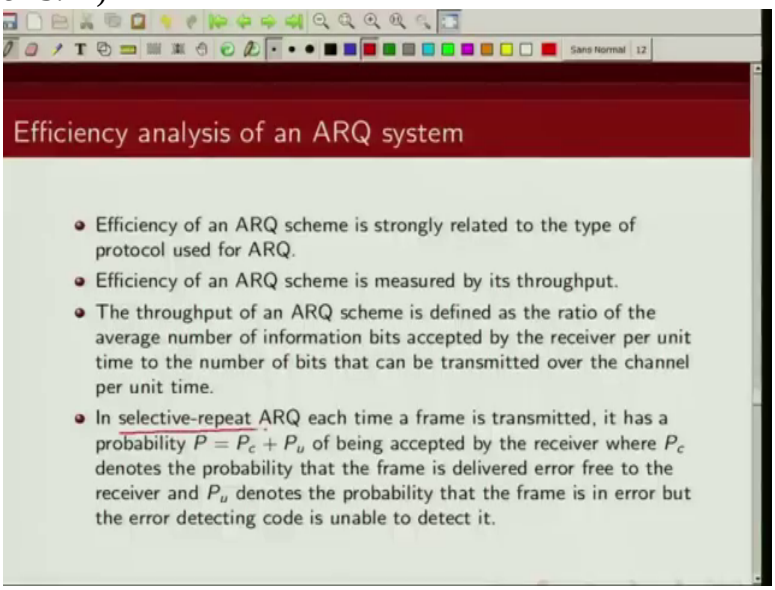


So let us compute this throughput for each of these 3 A R Q

(Refer Slide Time 43:23)

protocols that we talked about. So first scheme that we are going to talk about

(Refer Slide Time 43:27)

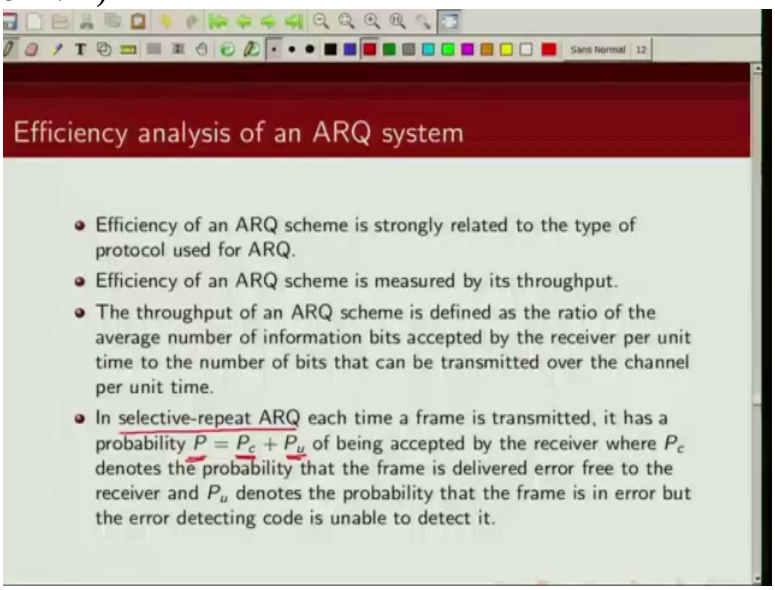


is Selective Repeat A R Q. Now in the Selective Repeat A R Q, the probability of receiving or accepting a packet is given by P where P is the probability that P is sum of two probability. This probability P sub c is basically the probability that the frame is delivered error free and the probability P u is the probability that the error detecting code failed to detect

(Refer Slide Time 43:59)

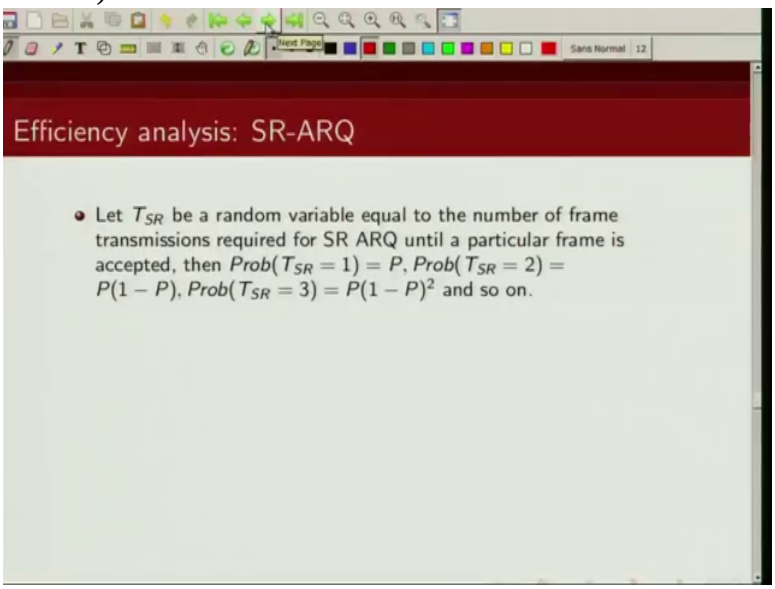
an error. So we are going to accept a packet either when it is received error free or the error detecting code fails to detect the error and thinks that it has received the packet correctly. So

(Refer Slide Time 44:14)



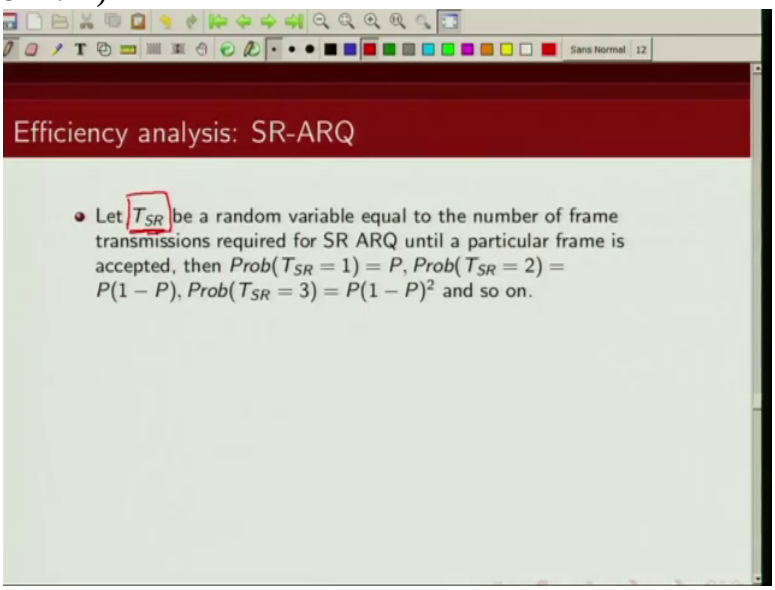
probability P is the sum of these two probability, with this probability we are going to accept, the receiver is going to accept a particular packet in S R A R Q scheme. Now

(Refer Slide Time 44:27)



T sub s r is a random variable that we are defining is equal to the number of frame transmission required for a selective repeat A R Q scheme until a particular frame is accepted. So this is basically a random error because

(Refer Slide Time 44:44)

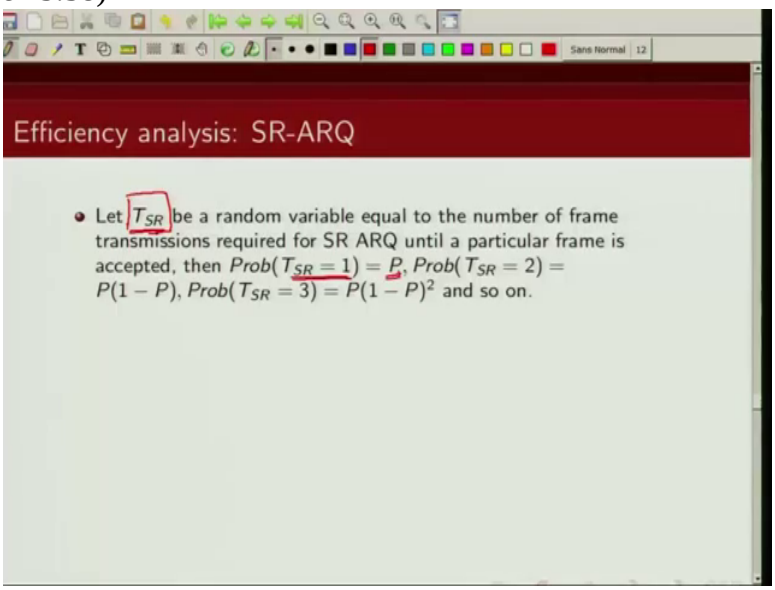


it can happen that at first instance the packet is accepted or it can happen that we need to transmit the packet twice to get the packet accepted or we could have to have transmit thrice or more, so T sub s r is a random variable which denotes the number of frame transmission required for Selective Repeat A R Q until a particular frame is accepted. Now let us compute the probability that T s r is 1. What is the probability that at first transmission itself you are going to accept the packet? Now we just saw that that probability is given by P which is sum of P sub c

(Refer Slide Time 45:35)

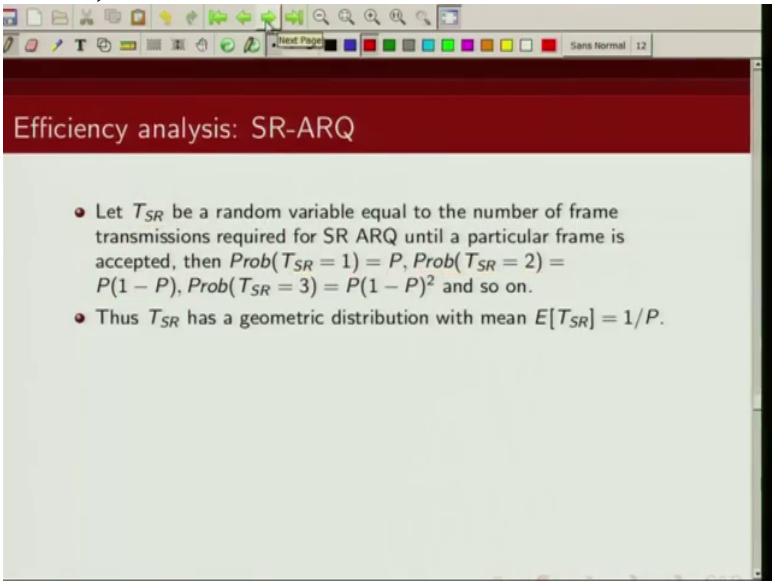
plus P sub u. So with probability P

(Refer Slide Time 45:38)



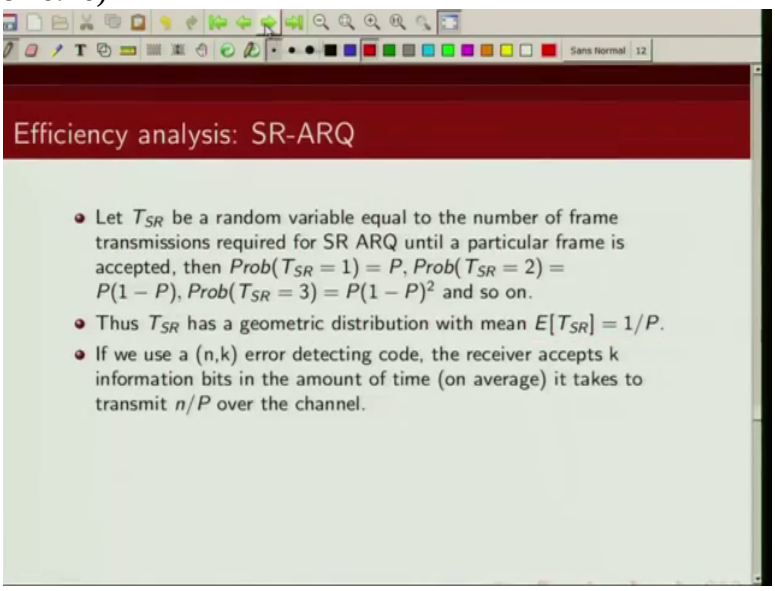
we are going to accept the packet the first transmission. Now what is the probability that the we need to transmit the packet twice? So that would be given by 1 minus P because in the first instance we did not accept the packet, that probability is given by 1 minus P and the second instance we accepted the packet. So the probability is given by P times 1 minus P. Similarly what's the probability that we have to make 3 transmissions before a packet is accepted. Now in this case, it means that the first two transmissions we did not accept the packet, that probability is 1 minus P square. And the third transmission we accepted it so that is P 1 minus P square. So

(Refer Slide Time 46:34)



we can see that this random variable has a geometric distribution and the expected value of this is given by 1 divided by P. Now

(Refer Slide Time 46:46)

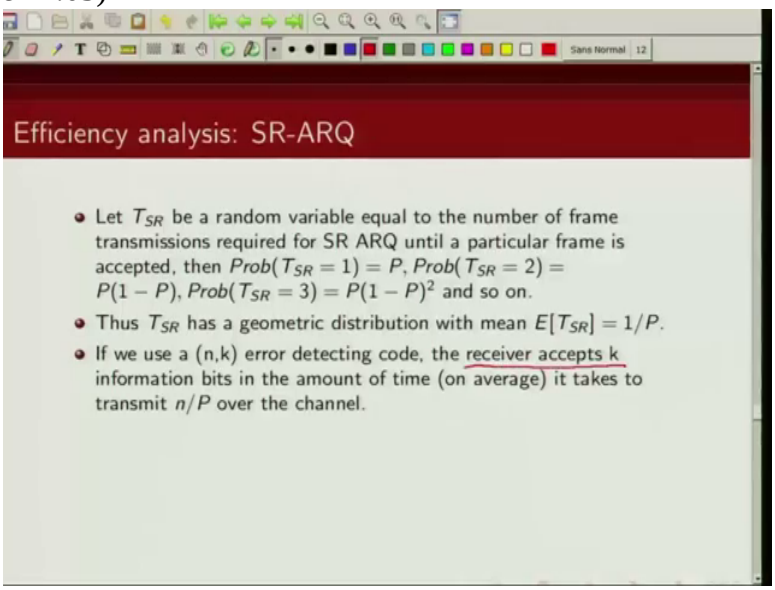


remember we are using

(Refer Slide Time 46:49)

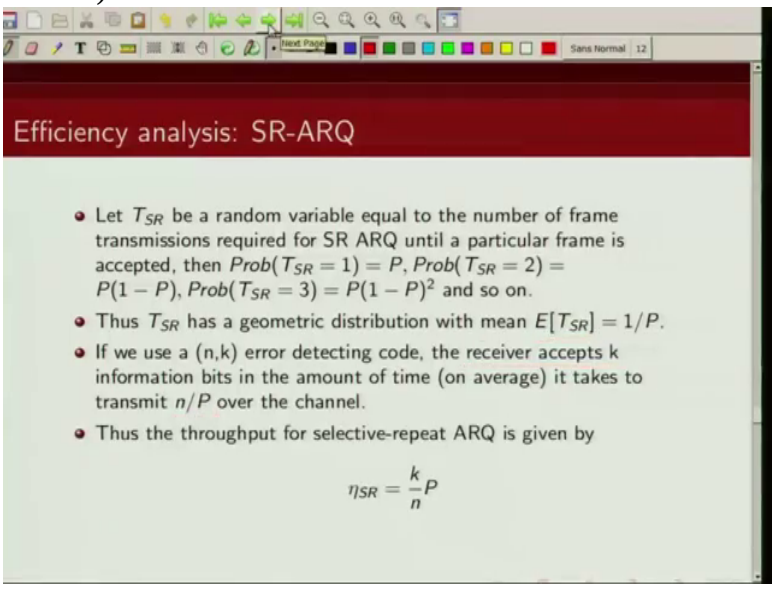
n k linear block code, we have k information bits and then n minus k C R C check bits and overall block length is n. So the receiver accepts k

(Refer Slide Time 47:03)



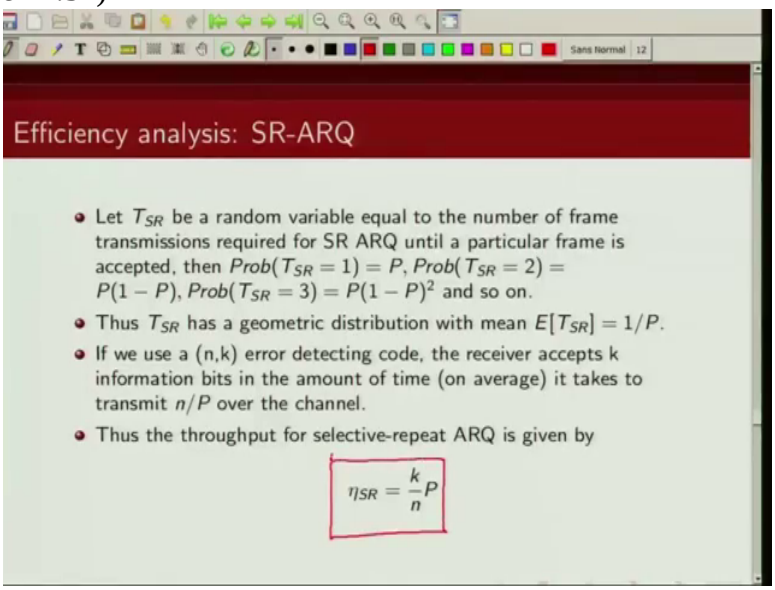
information bits in the amount of time it takes to transmit n into expected value of this. So receiver is accepting k information bits over the time it takes to transmit n divided by P over the communication channel. So the throughput is given by

(Refer Slide Time 47:27)



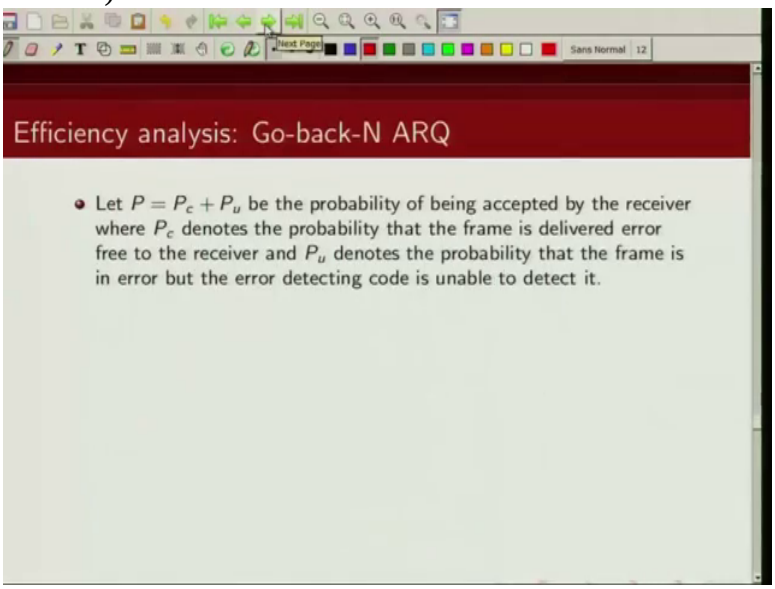
k divided by n by P which is this.

(Refer Slide Time 47:32)



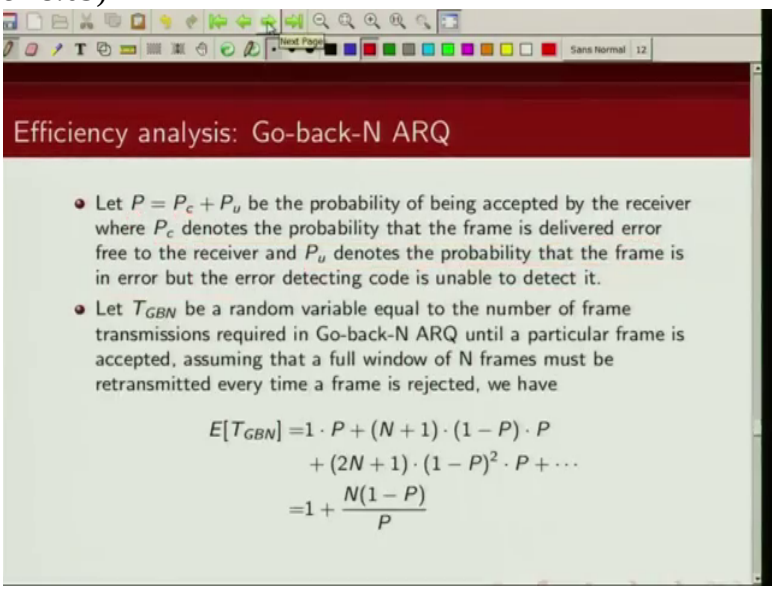
So for a Selective Repeat A R Q scheme, the throughput is given by k by n into P.

(Refer Slide Time 47:41)



Now let us compute the throughput for Go Back N. Again here, P sub c denotes the probability that the frame is received error free at the receiver and P u denotes the probability that the frame is in error but the receiver is unable to detect it. So a packet will be accepted with probability P.

(Refer Slide Time 48:09)

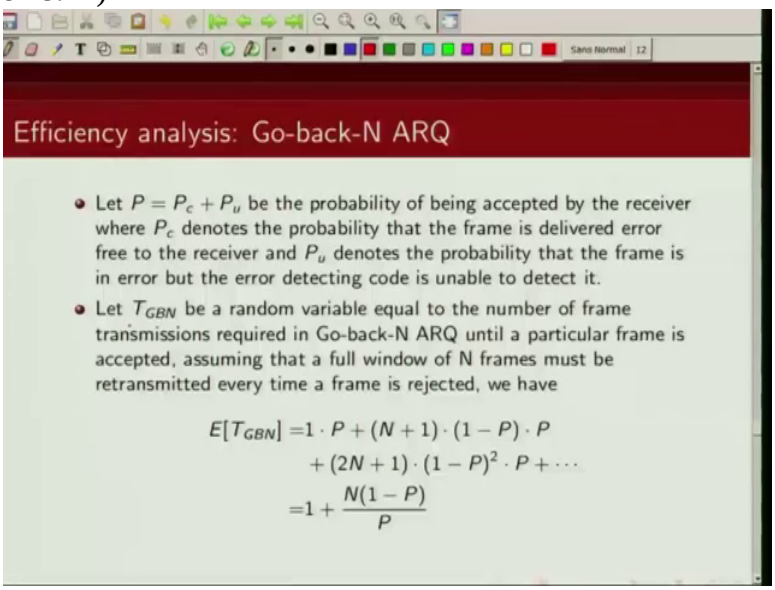


Now case of analysis, we are

(Refer Slide Time 48:12)

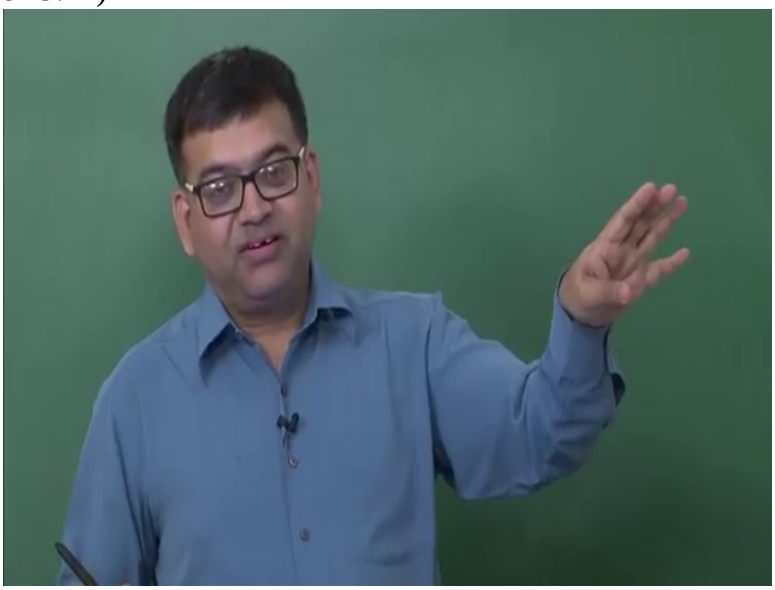
considering the worst case scenario and what is the worst case scenario? So we are assuming that the error happens in the beginning of the block. If the error happens in the beginning of the block then we need to, remember in Go Back N we backtrack and start sending, retransmit the packet from the sequence which has been received incorrectly. So the worst case scenario would be if at the start of the window we receive the first packet itself incorrectly then we would retransmit the next n bits. So

(Refer Slide Time 48:44)



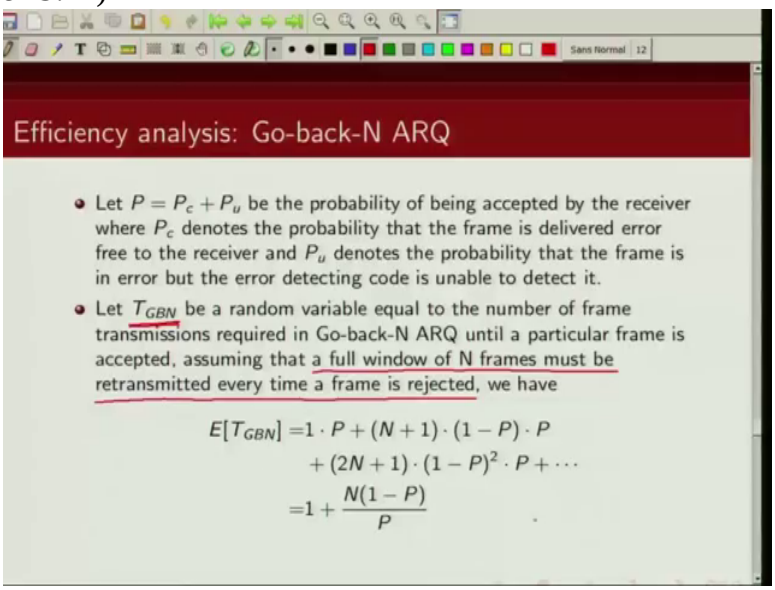
T G B N denotes a random variable which is equal to number of transmissions required in Go Back N until a particular frame is accepted. Now we are considering a worst case scenario, and what is that worst case scenario, we are assuming that a full window of n frames must be retransmitted every time a frame is rejected. So we are assuming the worst case scenario, the first

(Refer Slide Time 49:12)



frame in this window itself is received incorrectly. Therefore we are retransmitting all the packets. Now you can see if the packet is received

(Refer Slide Time 49:24)

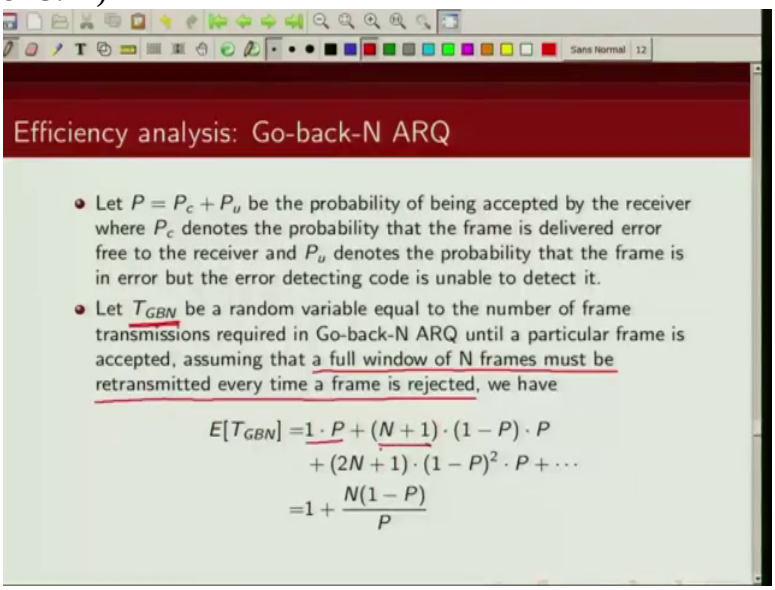


correctly the first instance that probability is 1 into P. Now if the packet is not received correctly first instance we are retransmitting

(Refer Slide Time 49:38)

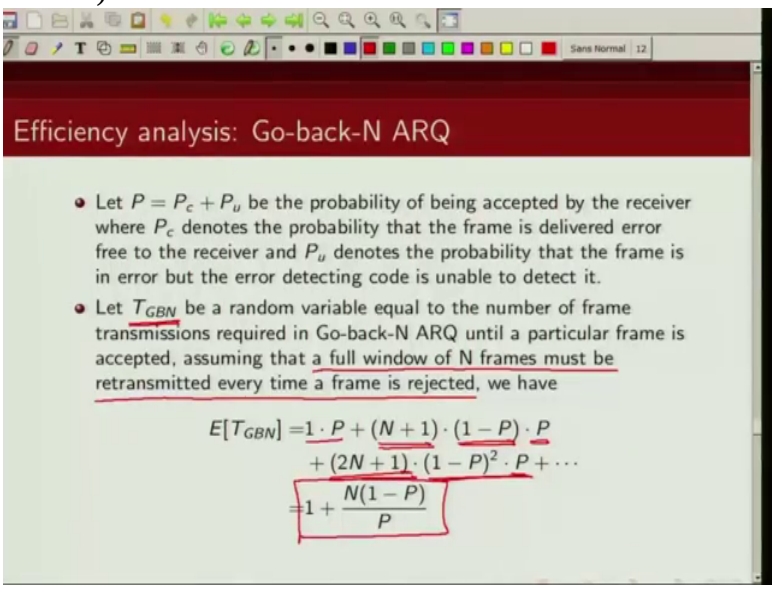
because we are considering the worst case scenario, we are re-transmitting these n packets so the number so we,

(Refer Slide Time 49:44)



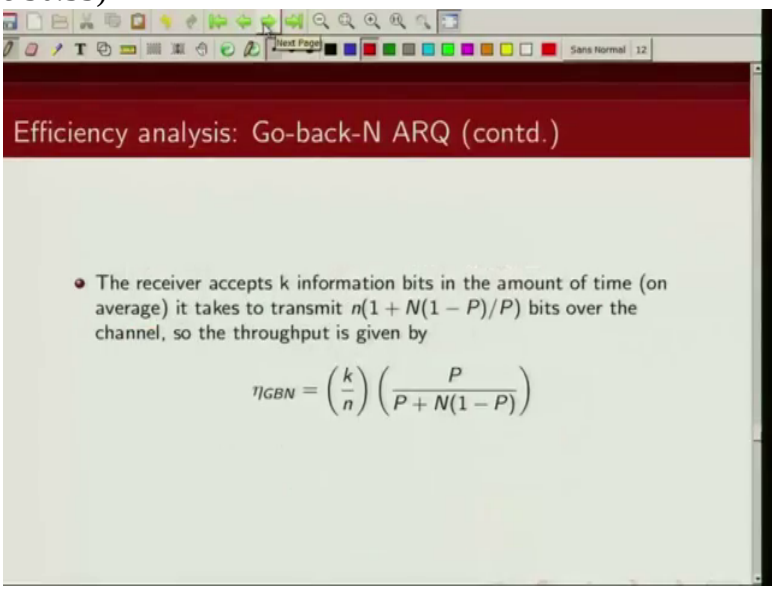
second time it will be received correctly n plus 1 into this is the probability that it was not received correctly in the first instance and this is the probability that the second instance it was received correctly. Similarly what is the probability that T, G B N is 3, so the first 2 transmissions, we did not receive the packets correctly but the third transmission we received correctly. That probability is given by this and number of frames that we transmitted is 2 raised to power 2 into N plus 1. So this, like this we can compute and this expected value comes out to be this quantity

(Refer Slide Time 50:27)



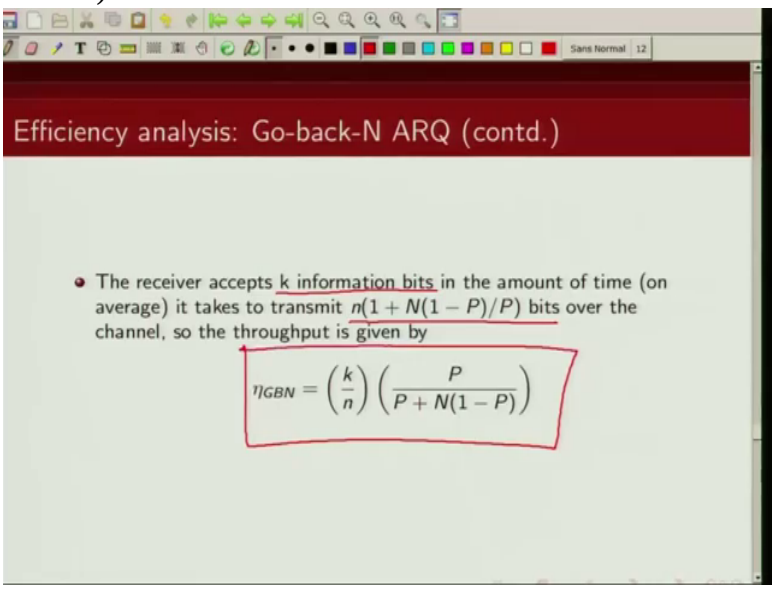
which is 1 plus N into 1 minus P divided by P. Now remember

(Refer Slide Time 50:35)



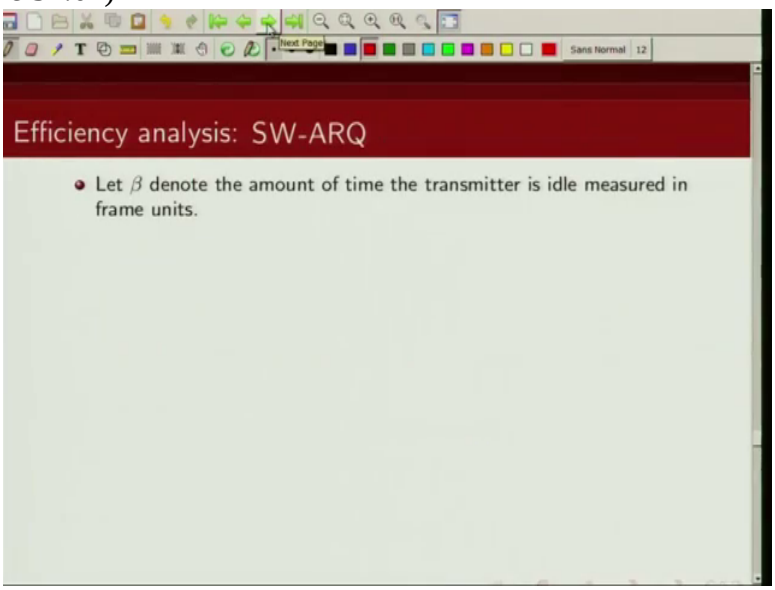
the receiver is accepting k information bits in this amount of time where it takes to transmit this many bits over the channel. So the throughput of Go Back N is given

(Refer Slide Time 50:51)



by this expression. Now this is, this quantity is less than what we can achieve from Selective Repeat scheme

(Refer Slide Time 51:02)

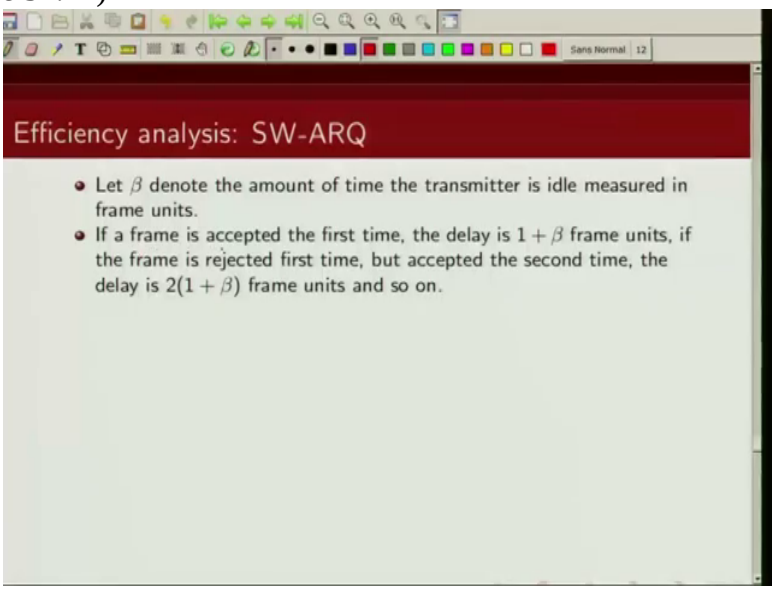


And finally we do this analysis for Stop and Wait A R Q. So let beta denotes amount of time the transmitter is idle and this idle time we are

(Refer Slide Time 51:17)

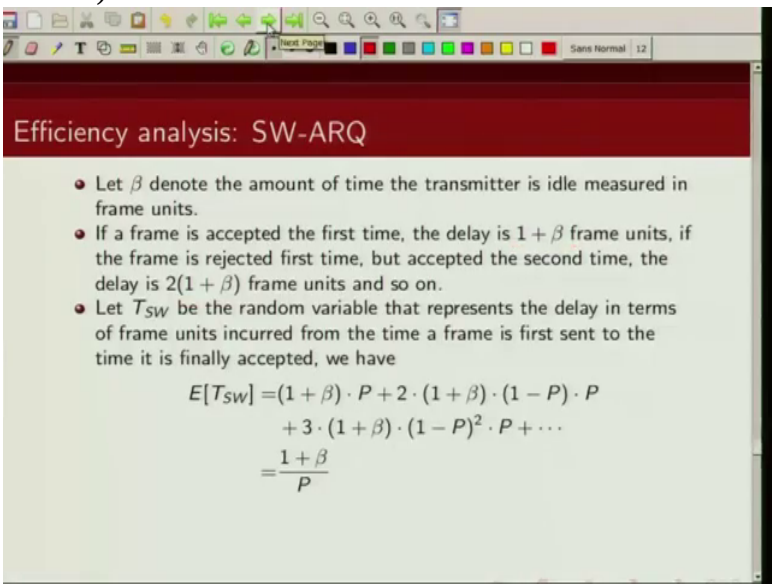
measuring in terms of this frame unit or frame length. So if a frame

(Refer Slide Time 51:24)



is accepted for the first time the total delay is 1 plus beta frame unit. If the frame is rejected and accepted second time, then the delay is two times 1 plus beta. If the frame is rejected second time and accepted third time the delay is three times 1 plus beta.

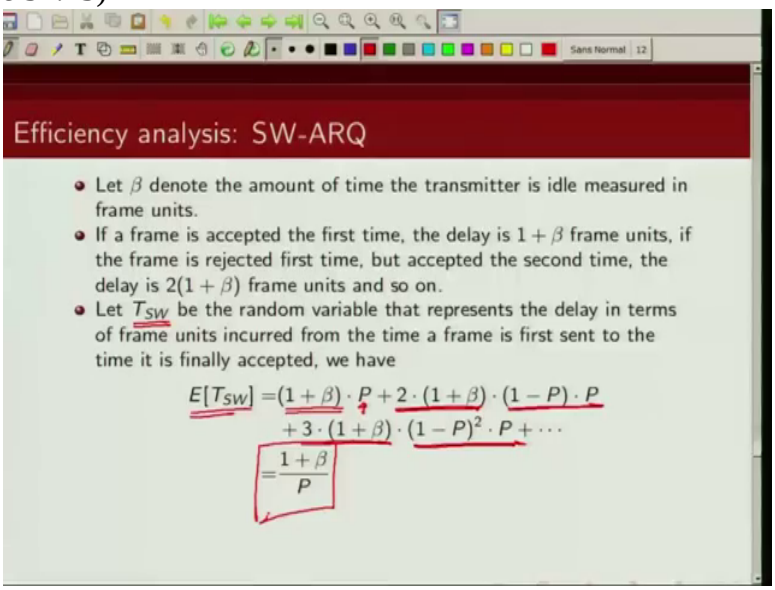
(Refer Slide Time 51:46)



So if we use this random variable T sub S W to represent the delay in terms of the frame unit, then expected value of this is given by, so this is the delay involved when the packet is accepted for the first time and this is the probability with which the packet will be accepted, the frame will be accepted for the first time. If it is rejected the first time and second time it is accepted this is the delay and this is the probability of the frame being rejected first time but accepted second time. Similarly this is the probability that the frame has been rejected twice

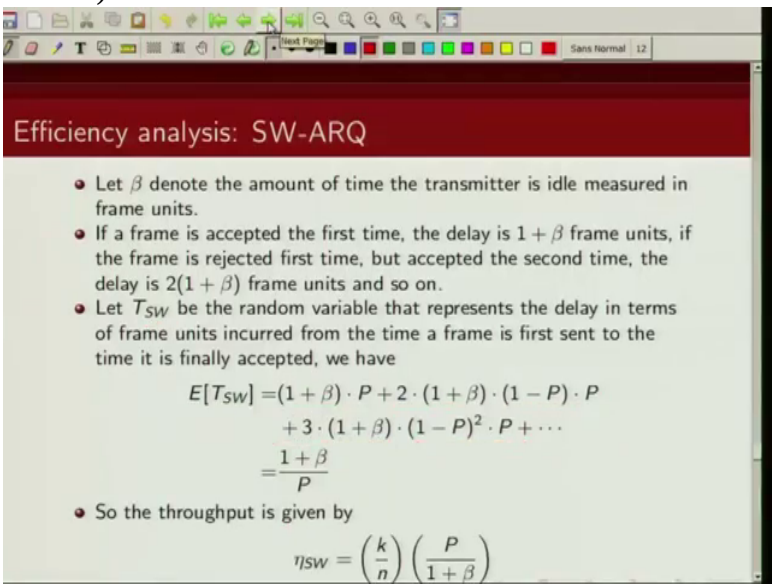
but the third instance it has been accepted and there is the corresponding delay. So like that if you write it, expected value of this T S W comes out to be 1 plus beta

(Refer Slide Time 52:43)



divided by P.

(Refer Slide Time 52:47)

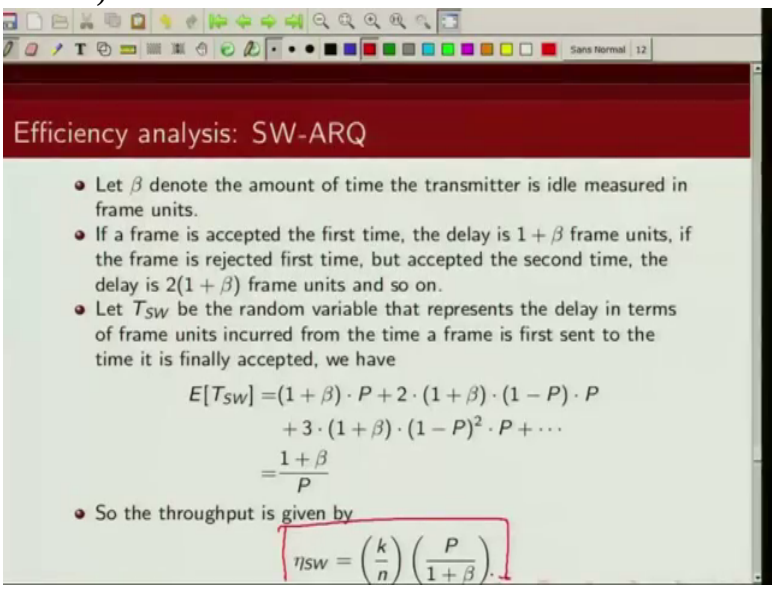


And again so the receiver is receiving k bits

(Refer Slide Time 52:53)

over the time N times 1 plus beta by P so the throughput will be given by

(Refer Slide Time 53:01)

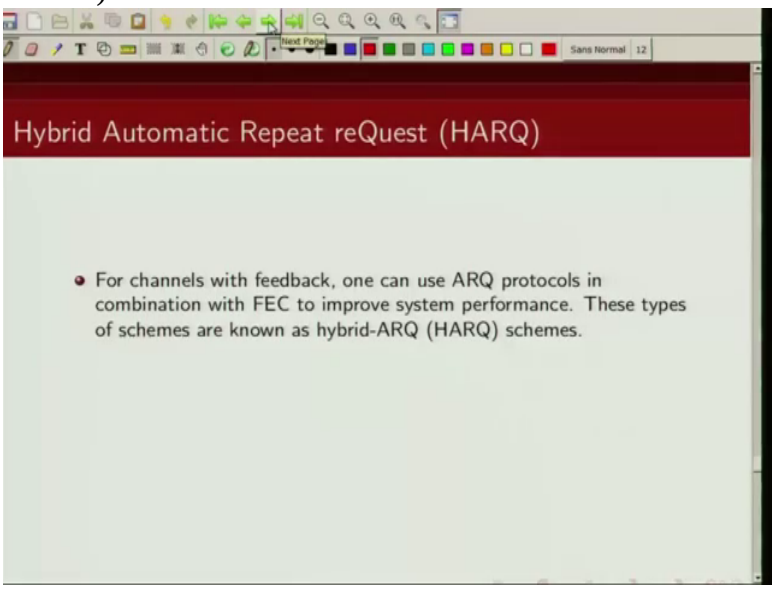


this expression. Now if we compare the efficiency of Stop and Wait, Go Back N

(Refer Slide Time 53:09)

and Selective Repeat A R Q, Selective Repeat A R Q will give us the best throughput followed by Go Back N and then Stop and Wait. However from the complexity point of view, Stop and Wait is the simplest, then Go Back N and then the most complex scheme is Selective Repeat A R Q.

(Refer Slide Time 53:32)

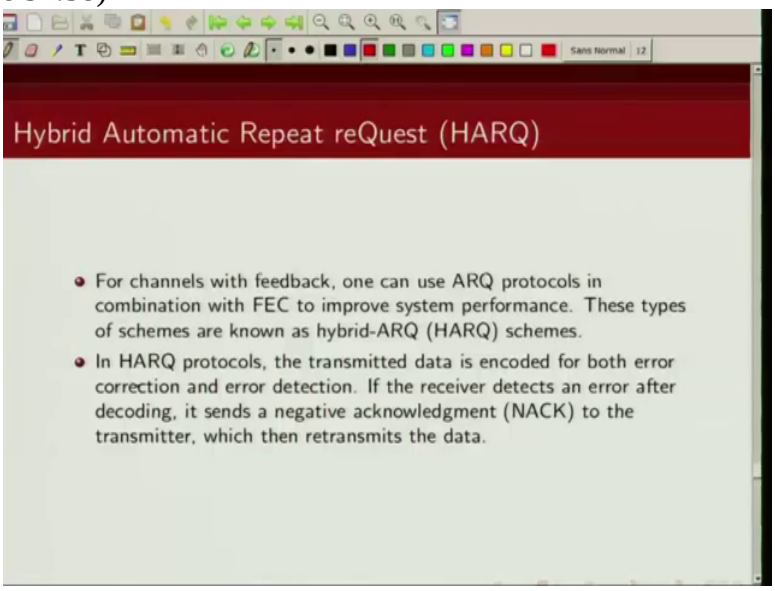


So next we are going to talk about hybrid A R Q which combines forward error correction with A R Q. So as you know that

(Refer Slide Time 53:46)

channels, one way channels use forward error correction and in two way channels we can use A R Q. In the pure A R Q protocols we send uncoded packets along with some parity bits which we use for error detection. Now in hybrid A R Q, we are using f e c coded packets for transmission, so at the receiver we will first try to correct the errors and then we will detect for errors. Now if the packets are detected to be in error, there receiver will send a NACK and another packet will be re-transmitted. So the difference between A R Q and hybrid A R Q is in pure A R Q protocols we send uncoded packets along with some parity bits which we use for error detection. So typically in pure A R Q protocol the information bits are coded using an error detecting code. Here we are encoding the packet using an error detecting as well as error correcting code.

(Refer Slide Time 54:58)

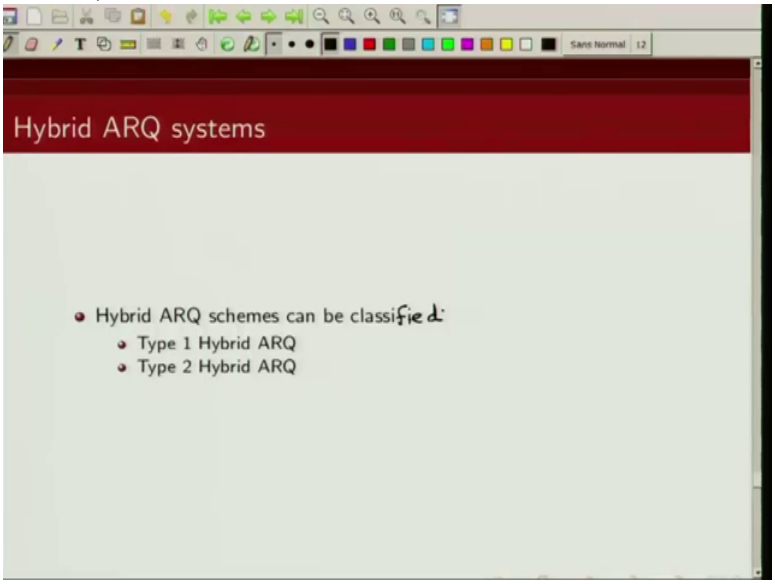


So in hybrid A R Q protocols we encode the transmitted data for both error correction as well as error detection. The receiver detects the error; after decoding it will send a negative acknowledgement and then the data will be again transmitted.

(Refer Slide Time 55:20)

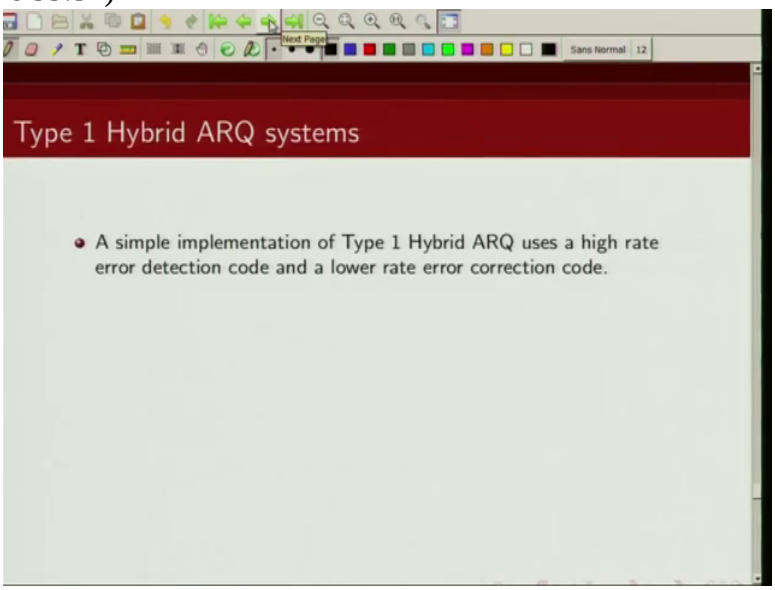
So hybrid A R Q schemes can be classified into 2 categories,

(Refer Slide Time 55:25)



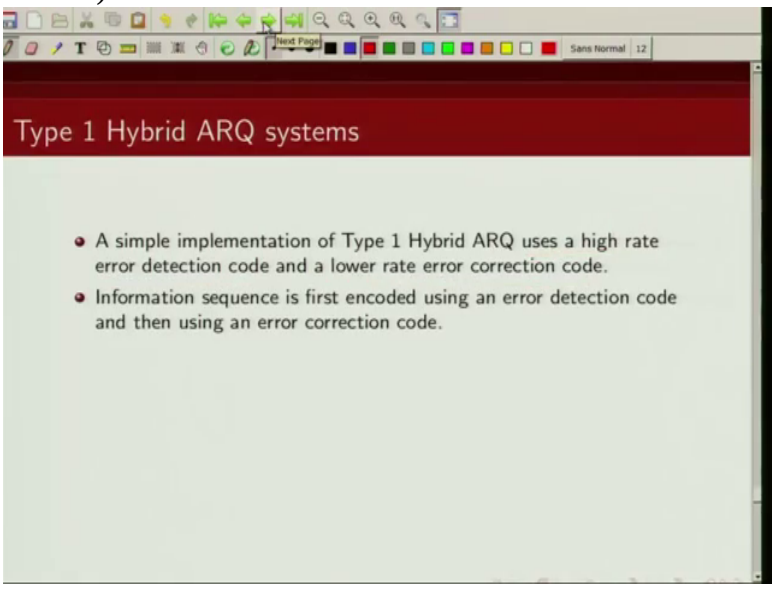
first one which is known as Type 1 hybrid A R Q scheme and second one which is known as Type 2 hybrid A R Q schemes.

(Refer Slide Time 55:34)



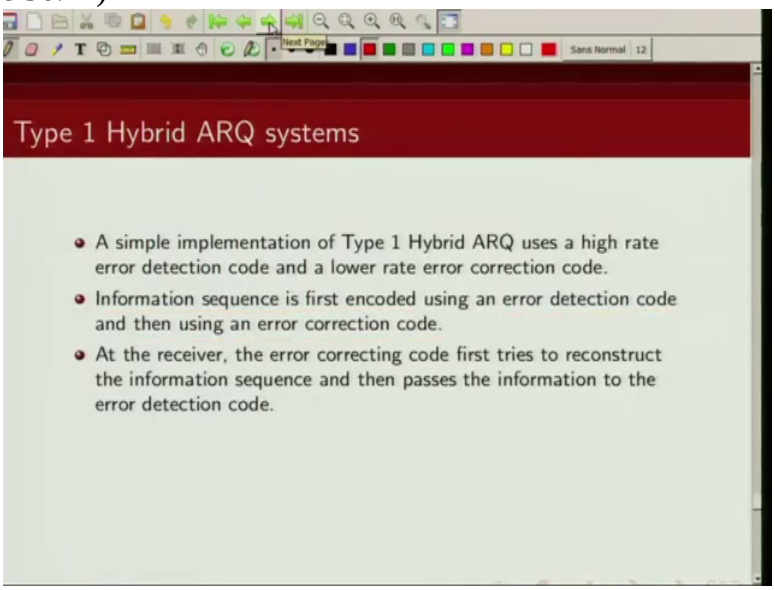
So we will first describe Type 1 hybrid A R Q scheme. So simple implementation of a Type 1 hybrid A R Q scheme is as follows. We use a very high rate error detecting code and a lower rate error correcting code. Now this high rate error detecting code could be like for example, C R C which we are going to use to detect error and this lower rate error correcting code could be convolutional code, turbo code, L D P C code

(Refer Slide Time 56:09)



so the information sequence is first encoded using an error detecting code and then using a error correcting code. At the receiver

(Refer Slide Time 56:21)

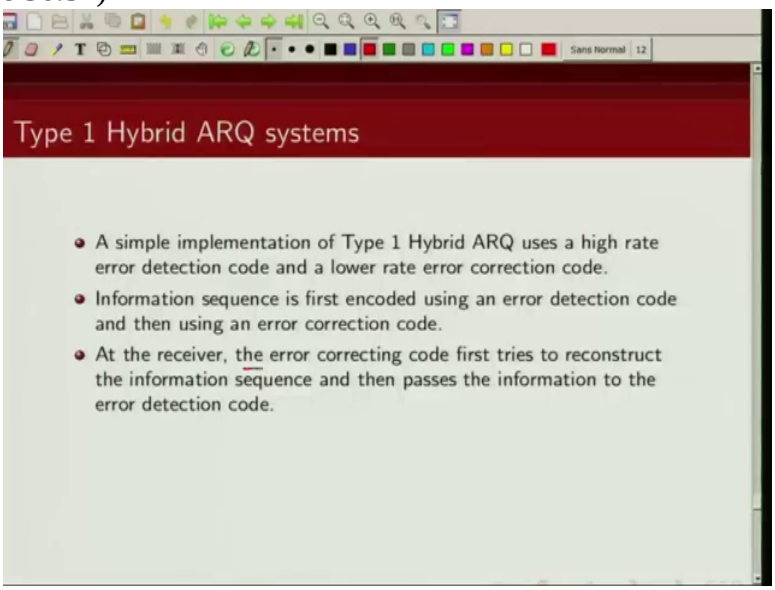


when you receive this coded packet you first

(Refer Slide Time 56:25)

try to decode and try to correct errors. So at the receiver

(Refer Slide Time 56:31)

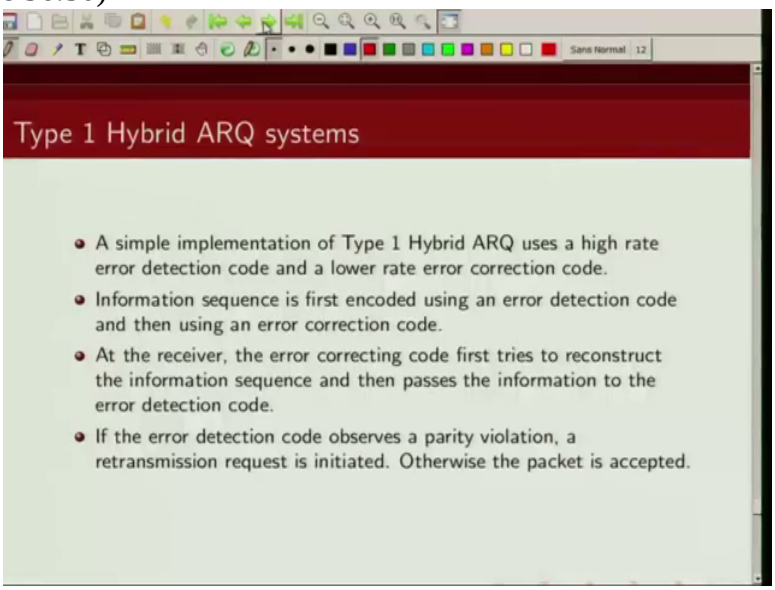


the error correcting code first tries to reconstruct the information sequence or estimate the information sequence and then it passes

(Refer Slide Time 56:40)

the information to the error detecting code which is going to check whether estimated information sequence is in error or not.

(Refer Slide Time 56:50)

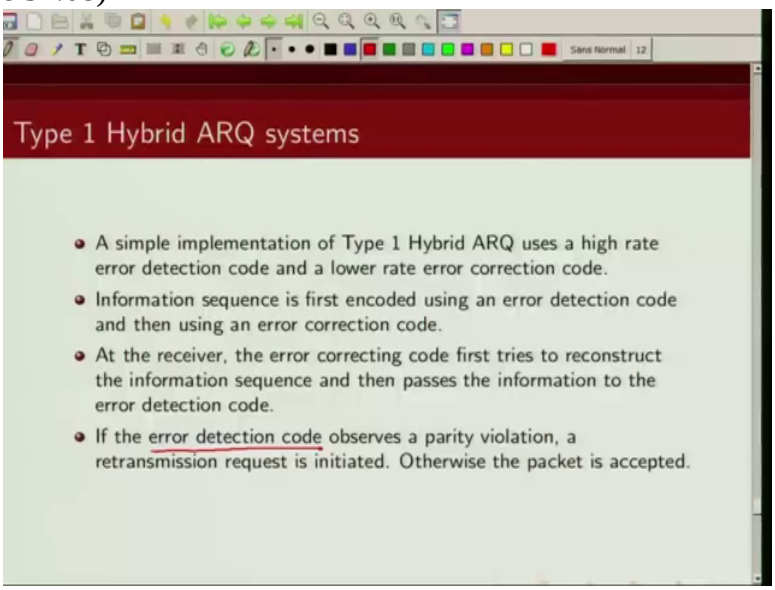


If the error detection code observes that there is a parity violation, for example

(Refer Slide Time 56:57)

if you are example if you are using a C R C we know that the received polynomial should be divisible by the generator polynomial of the C R C code. So if the

(Refer Slide Time 57:08)

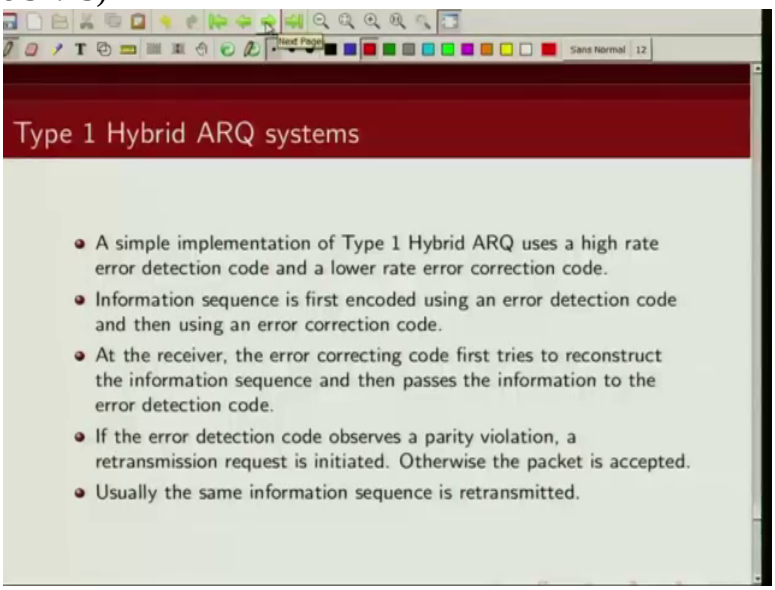


error detecting code observes the parity violation then it will ask for re-transmission. Otherwise the packet would be assumed to be

(Refer Slide Time 57:19)

received correctly and we will accept the packet.

(Refer Slide Time 57:25)

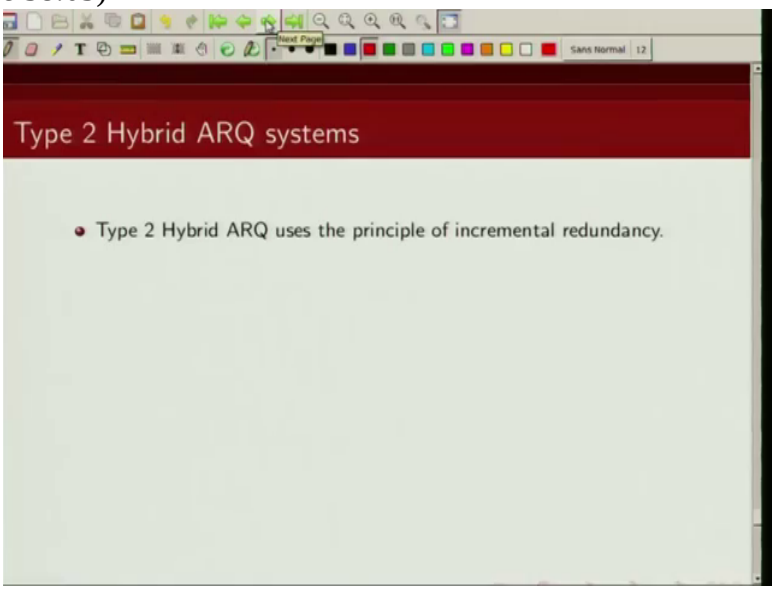


And usually in this Type 1 A R Q scheme when we retransmit we are again sending the same coded packet. So

(Refer Slide Time 57:35)

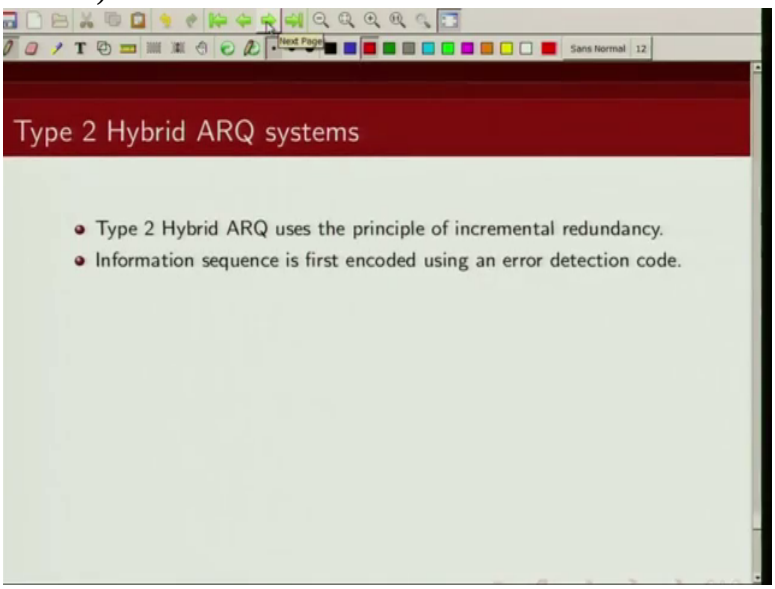
whenever the receiver is asking for retransmission, we are sending the same coded packet again. The receiver tries to correct the errors and then the information, extrinsic information sequence is passed to a error detecting code which will try to detect whether there is any error in the decoded sequence. If there is error again, receiver will set a NACK and the same coded sequence would be sent again.

(Refer Slide Time 58:05)



Now type 2 hybrid A R Q scheme works on the principle of incremental redundancy. So what happens here is

(Refer Slide Time 58:16)

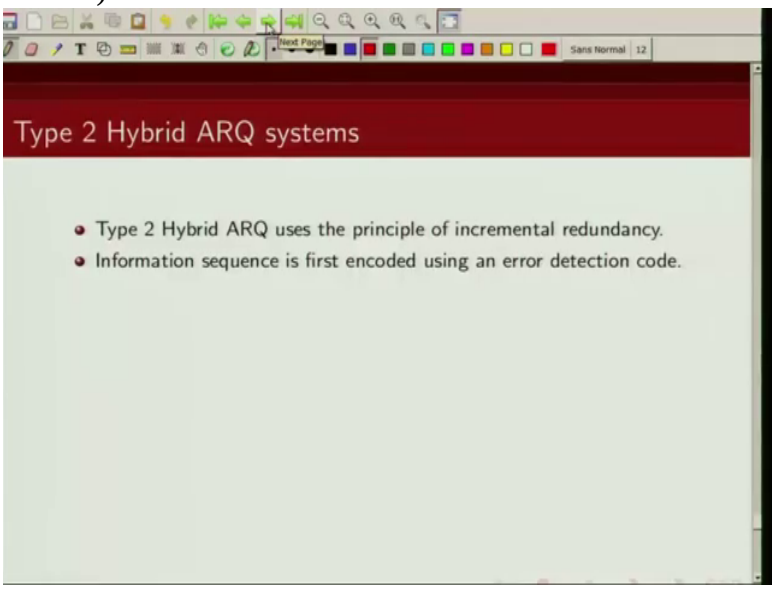


you first send information sequence coded using an error

(Refer Slide Time 58:21)

detecting code, alright. Now

(Refer Slide Time 58:24)

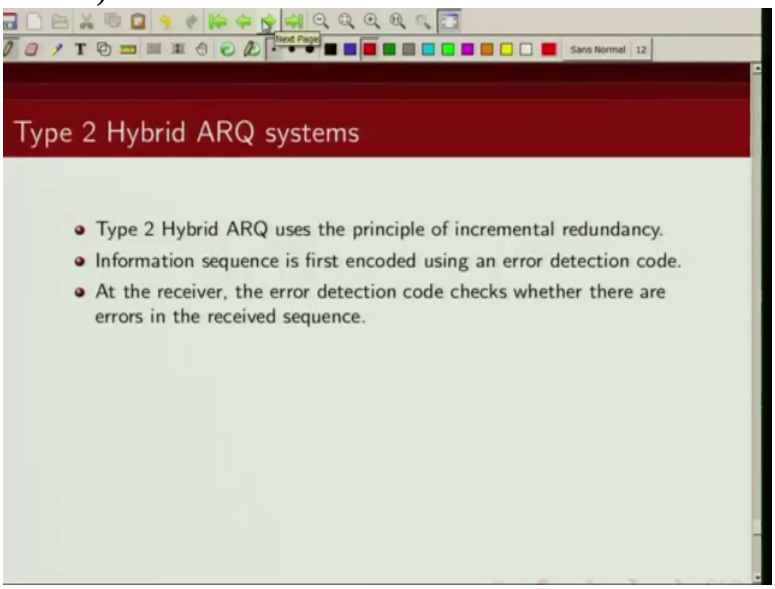


once you receive this packet the error detecting code will try to see if the

(Refer Slide Time 58:31)

packet has been received correctly or not. Now if the

(Refer Slide Time 58:36)

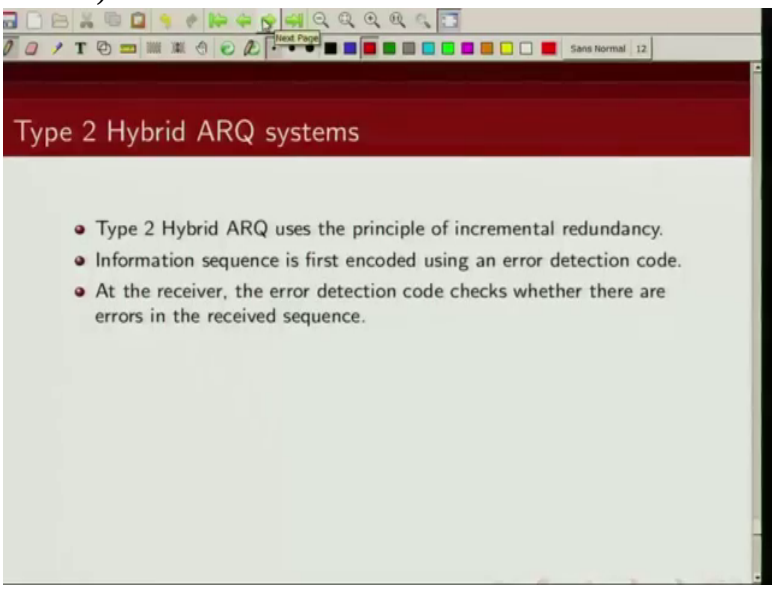


error detecting code does not find any error or it passed the parity constraint checks then the packet will be accepted

(Refer Slide Time 58:43)

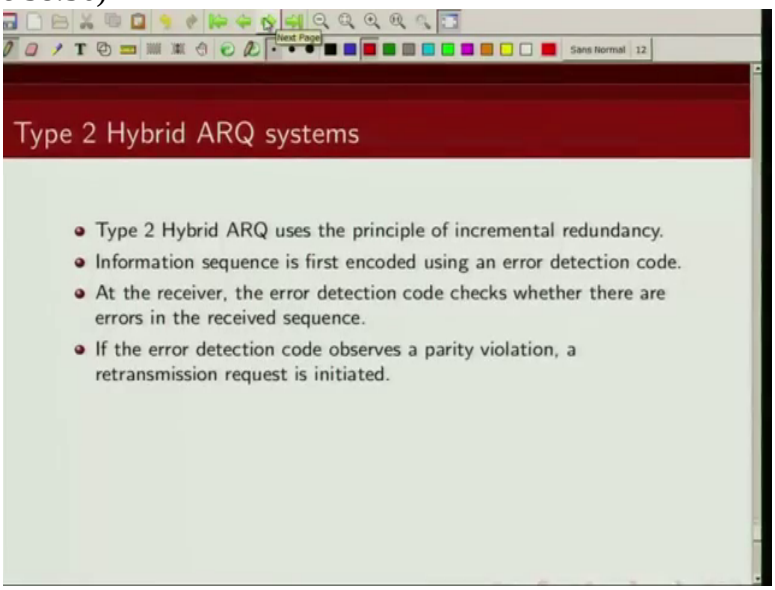
otherwise you will, the receiver is going to send a negative acknowledgement and then the transmitter is going to send a new packet. And what is there in this new packet?

(Refer Slide Time 58:55)



That is where

(Refer Slide Time 58:56)

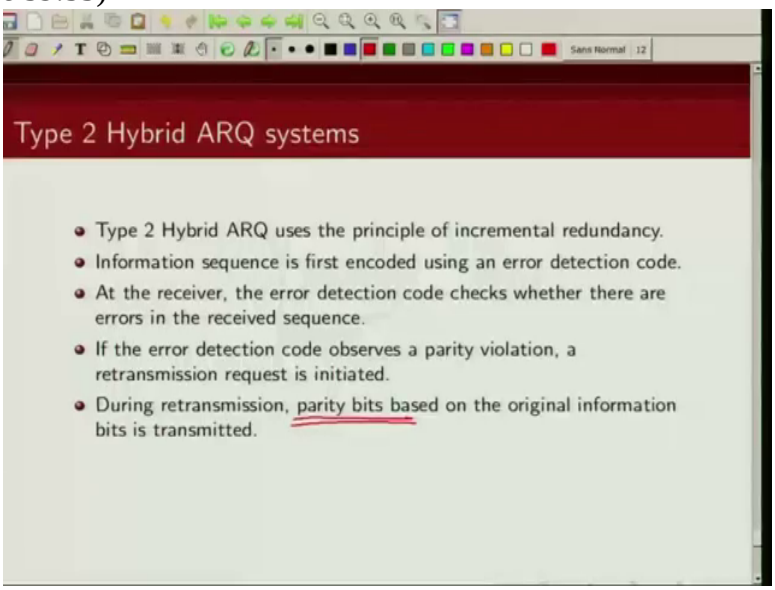


the difference lies. So the new packet that we will send will have send parity bits based on the information sequence. If you remember, during initial transmission we are sending only the information sequence coding

(Refer Slide Time 59:14)

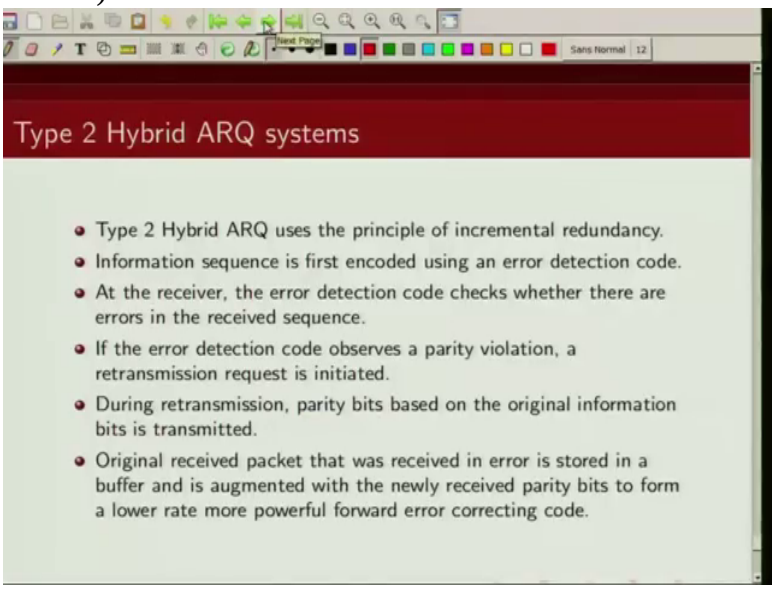
using error detecting code. Now if that is not received correctly in the first retransmission we are going to send parity bits corresponding those information bits and we are encode them using error detecting code and then transmit over the communication channel. Now at the receiver you already know the information bits that you received in the first transmission. Now we are receiving a parity bit. So the receiver you can make a low, a more powerful lower rate code and decoder can now try to correct the errors.

(Refer Slide Time 59:53)



So

(Refer Slide Time 59:54)

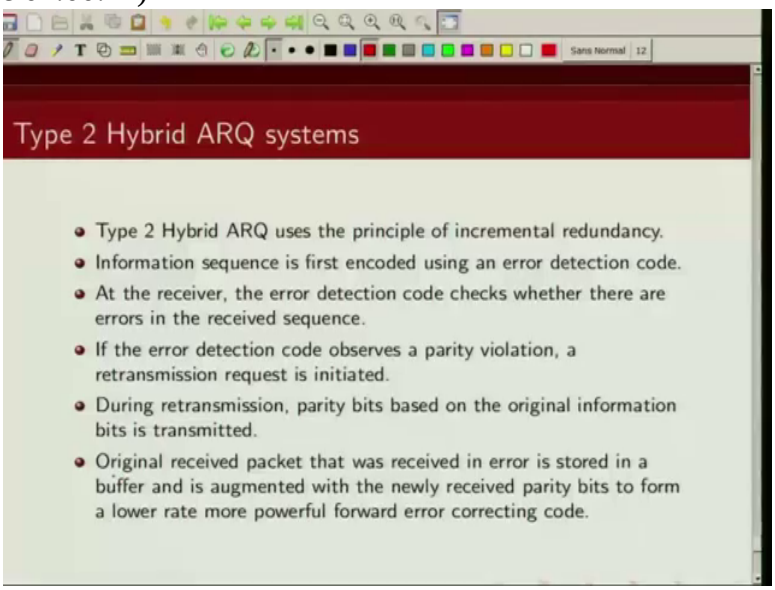


the difference here is, in Type 2 hybrid A R Q scheme each time when we retransmit, we send new set of parity bits and at the receiver we are adding these new set of parity bits to our original received sequences

(Refer Slide Time 01:00:10)

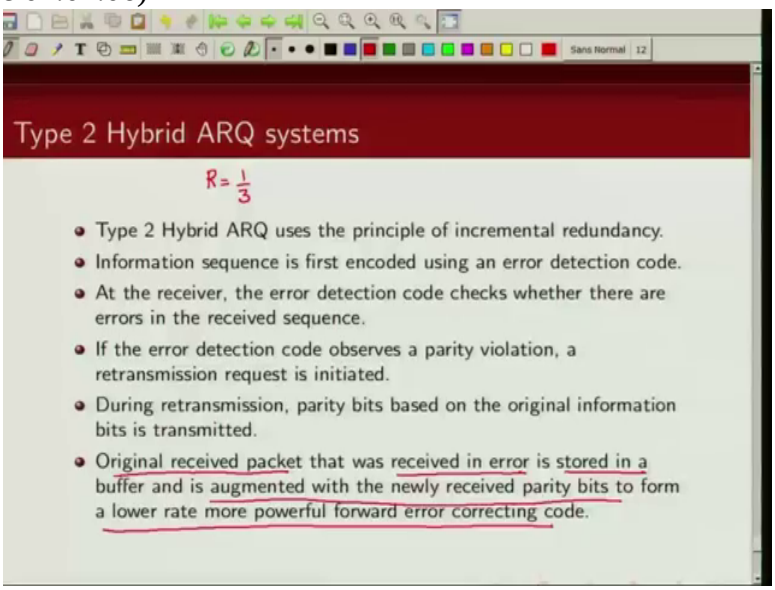
to create a more powerful lower rate code. In type 1 kind of schemes we were sending coded packets but then when we are retransmitting we are sending the same coded packet. There also if you want, you could combine those packets together to get a better performance. But the difference here is when we are retransmitting in type 2 hybrid A R Q scheme, each time instance we are sending a new set of parity bits. So

(Refer Slide Time 01:00:42)



the original received packet which was received in error is now stored in a buffer and it is augmented with newly received parity bits to form a more powerful lower rate forward error correcting code. So if you take an example, let’s say I have a rate one third code, Ok. So let's say

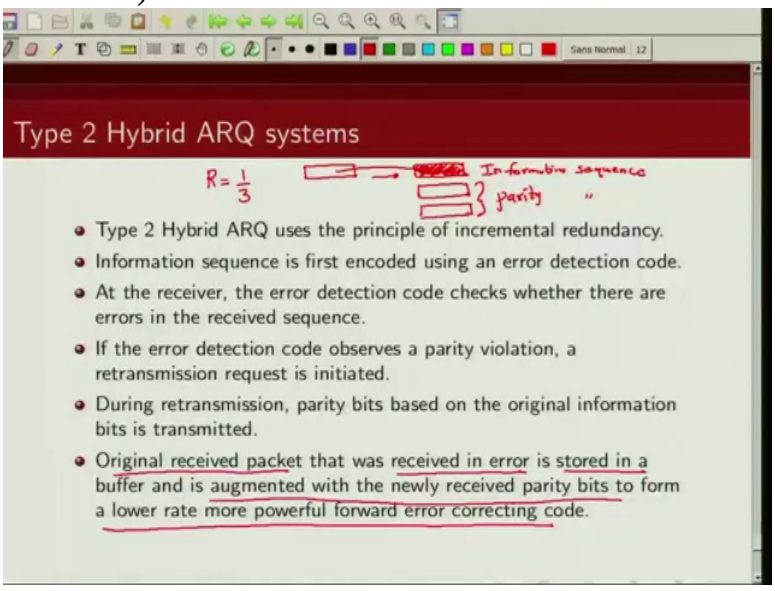
(Refer Slide Time 01:01:06)



this is my block of information bits. Now if pass this block of information bits through a rate one third encoder, what you will get I will get 3 sets

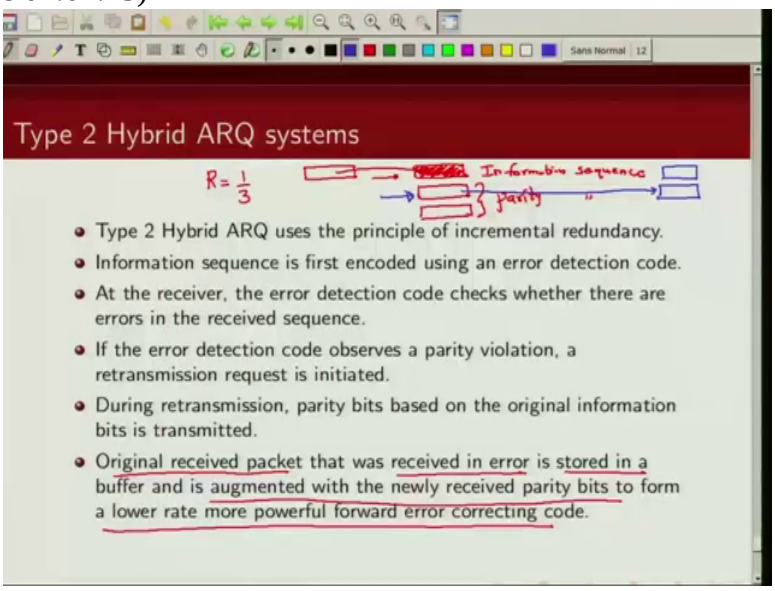
Let's assume this is a systematic code. So this is nothing but this and this are same, these are information sequence, information sequence and these are my parity sequence. Now in Type 2 hybrid A R Q scheme you are sending these information sequence first. Now if the error detecting code finds out

(Refer Slide Time 01:01:50)



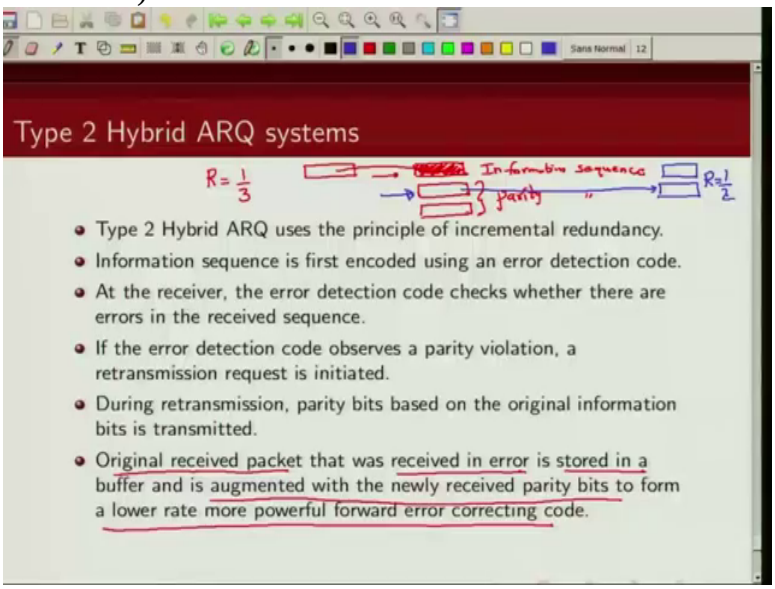
that these information bits are in error, in the first retransmission you are going to send these parity bits. At the receiver what you are going to do is you already have these information bits that your received at the first transmission. Now to that you add this parity bit. So now

(Refer Slide Time 01:02:13)



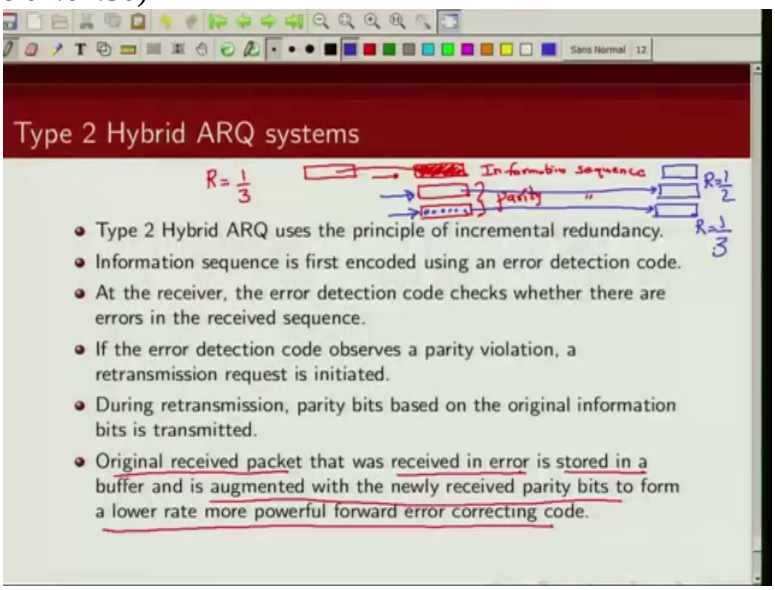
what's going to happen is you are now effectively creating a rate half

(Refer Slide Time 01:02:18)



code at the receiver. And now this rate half decoder will try to correct error. Now if this fails again you will ask for, the receiver will ask for retransmission. Then the transmitter is going to send you another set of parity sequence which is different from what was sent earlier. So this new set of priority sequence is now sent here. Now your code has become a rate one third code. So

(Refer Slide Time 01:02:50)

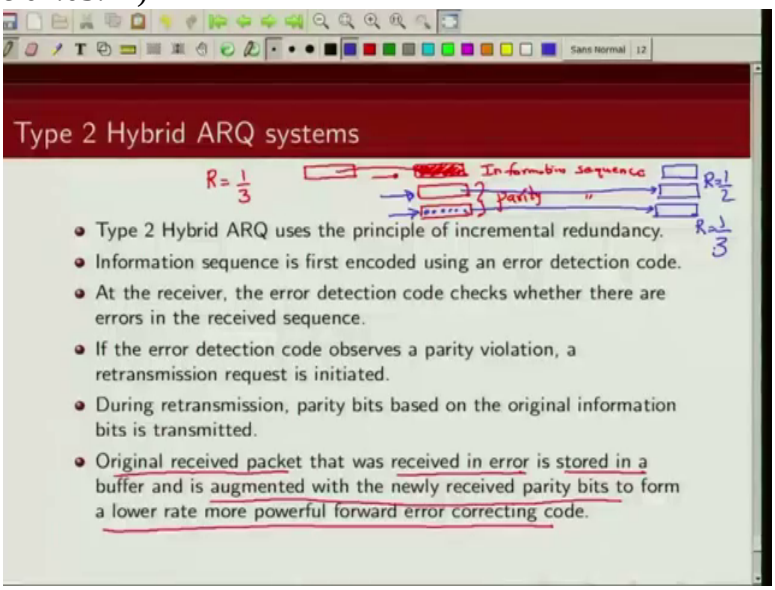


in Type 2 hybrid A R Q schemes what you are doing is at each retransmission

(Refer Slide Time 01:02:56)

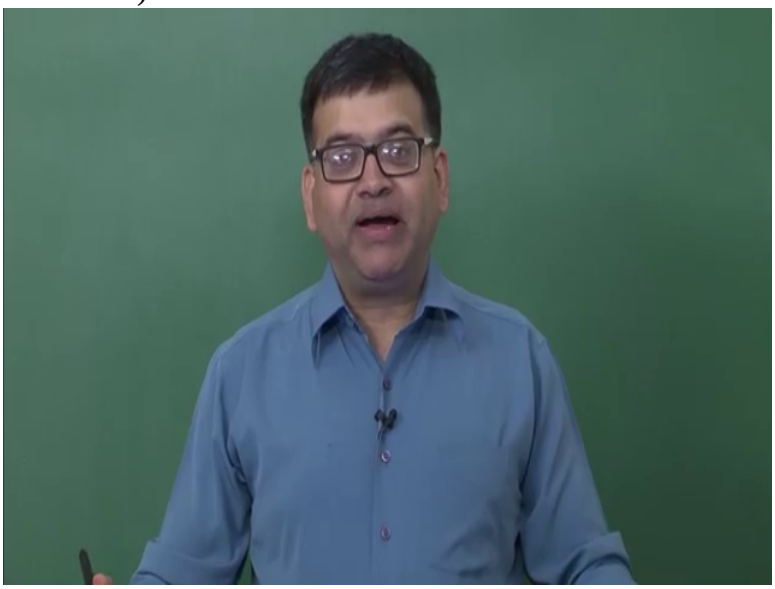
you are sending new parity bits and at the receiver you are adding them to already received information sequence and parity sequence and making it a more powerful lower rate code. So this

(Refer Slide Time 01:03:12)



is how a type 2 A R Q schemes work, Ok. So with this I am going to conclude my

(Refer Slide Time 01:03:19)



discussion on automatic repeat request schemes, thank you