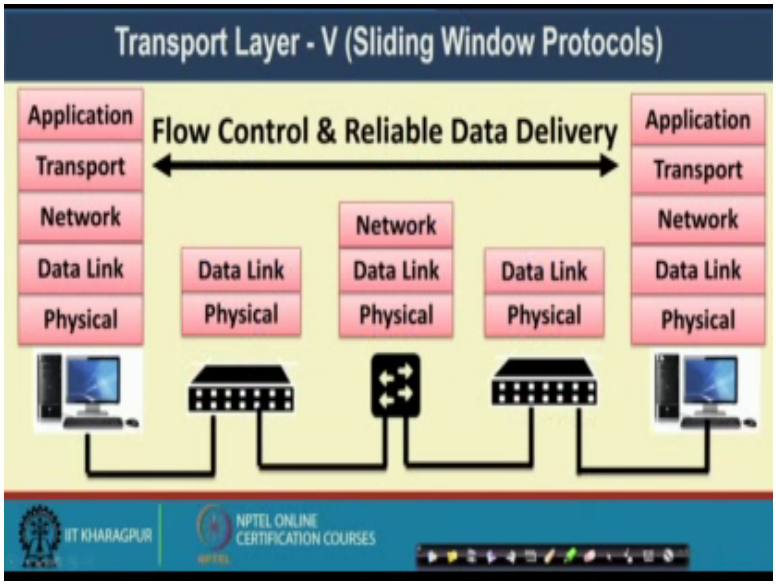
Computer Networks and Internet Protocol
Prof. Sandip Chakraborty
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 15
Transport Layer – V
(Sliding Window Protocols)

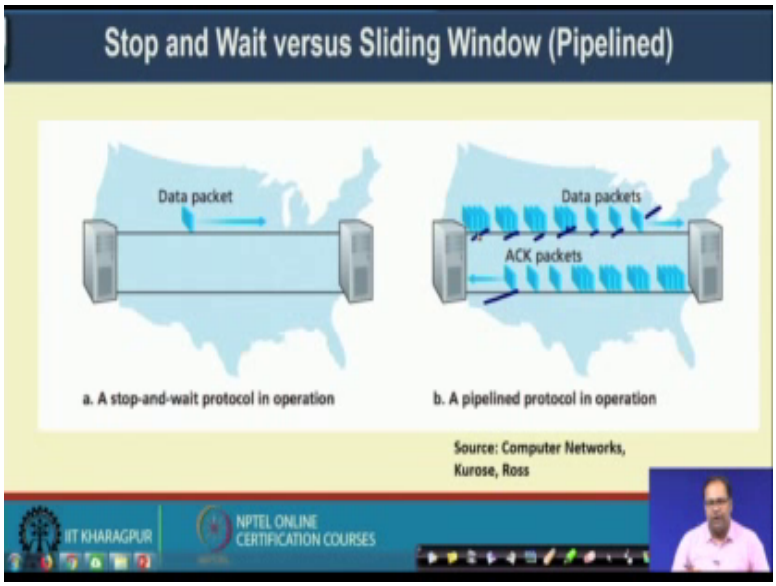
Welcome back to the course on Computer Network and Internet Protocols.

(Refer Slide Time: 00:24)



So, in the last class we have discussed about this flow control and reliable data delivery protocols over the transport layer and we have looked into the details of the stop and wait flow control and reliable protocol; which we call as the stop and wait to ARQ.

(Refer Slide Time: 00:44)

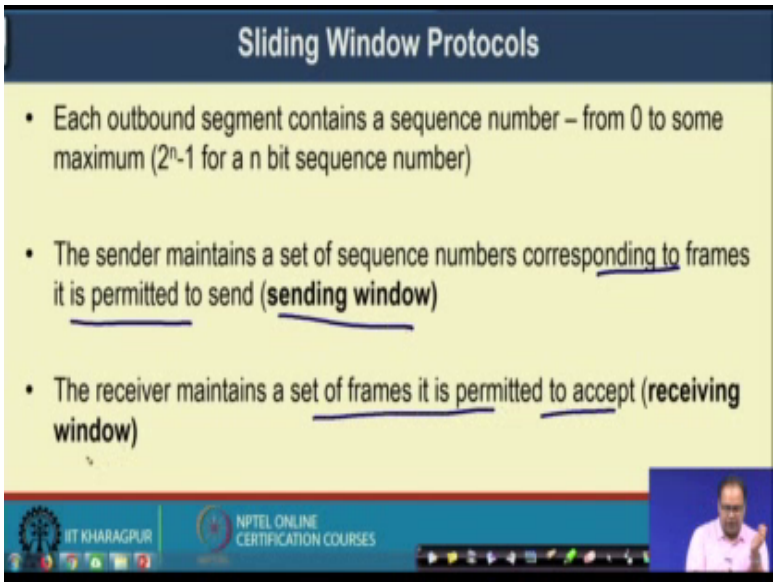


And there we have see that this stop and wait protocol in case of stop and wait protocol one major disadvantage is that, you can have only 1 packet outstanding in the network and that is way you are not able to utilize the full capacity of the link. So, for every out going packet you have wait for the acknowledgement unless you are receiving the acknowledgement you will not be able to send the next packet.

In case of the sliding window protocol we want to ensure a pipeline version of the protocol where you can send multiple packets all together in a pipeline fashion. So, we will look into the details of the sliding window protocol about how we can achieve this pipelining and at the same time you can receive the acknowledgement parallelly and accordingly control your transmission rate.

So, the broad idea is something like this if you look into this diagram you can see that initially I started sending 1 packets here I am I have started sending 1 packets and once I receive the acknowledgement for that 1 packet I increase my window size to send more packets in parallel. So, here we are sending 3 packets in parallel, then again we have increased it to 4 packets and that way you can push the packets parallelly in the network such that by following a mechanism, such that it does not exceed the capacity of that particularly link. So, let us look into different sliding window protocols in details.

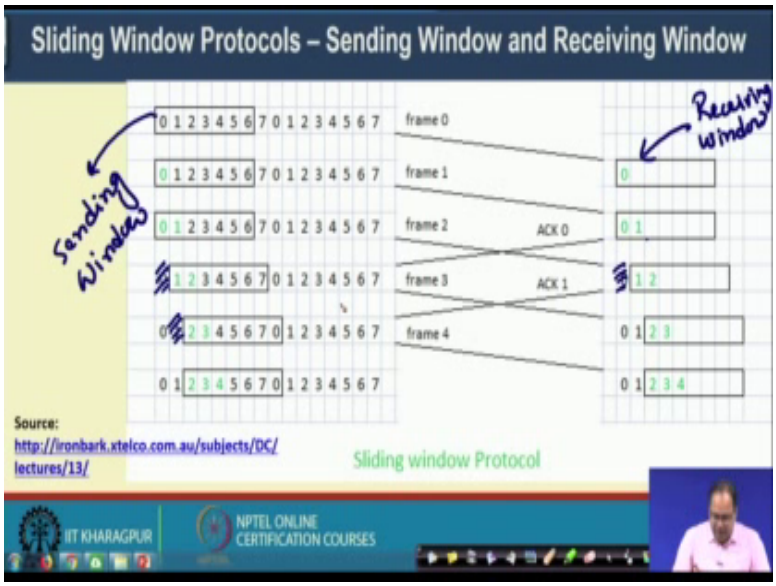
(Refer Slide Time: 02:13)



So, the broad idea of a sliding window protocol is as follows that each outbound segment in the network, it contains a sequence number. So, the sequence number field is from starting from 0 to some maximum sequence number say for example, if you are using a n bit sequence number, then the sequence number space can go from 0 to 2 the power n minus 1.

Now, the sender it maintains a set of sequence numbers corresponding to the frames it is permitted to send. So, this particular set of sequence number we call it as a sending window, similarly, the receiver it maintains a set of frames which it is permitted to accept we call it as the receiver window or receiving window. So, let us look into an example in details to clear this concept.

(Refer Slide Time: 03:01)



So, here in the sender side so, this bounding box is the sending window. So, what does it mean? So, this is the sending window here and this is the receiver window or the receiving window. Now, the sender window indicates that well at a time you can send frames or packets from 0 to 6 without waiting for the corresponding acknowledgement so; that means, you can send this frames in parallel. So, you can send a frame 0, then immediately you can send frame 1, then you can send frame 2 without waiting for the corresponding acknowledgement.

And the receiver window say that here the receiver window side is 1 2 3 4 5 6 7 so, receiver window says that well; you can receive 7 frames all together and once you have received 7 frames you will not be able to receive any further frame without sending back an acknowledgement. So, what happens here that well the sender keep on sending the frame and once the sender gets back an acknowledgement so, here say in this example the sender has transmitted frame 0 followed by frame 1 followed by frame 2 it is sending the frames in parallel without waiting for the acknowledgement, but whenever the receiver has received frames 0's and 1's it sends the acknowledgement for frame 0.

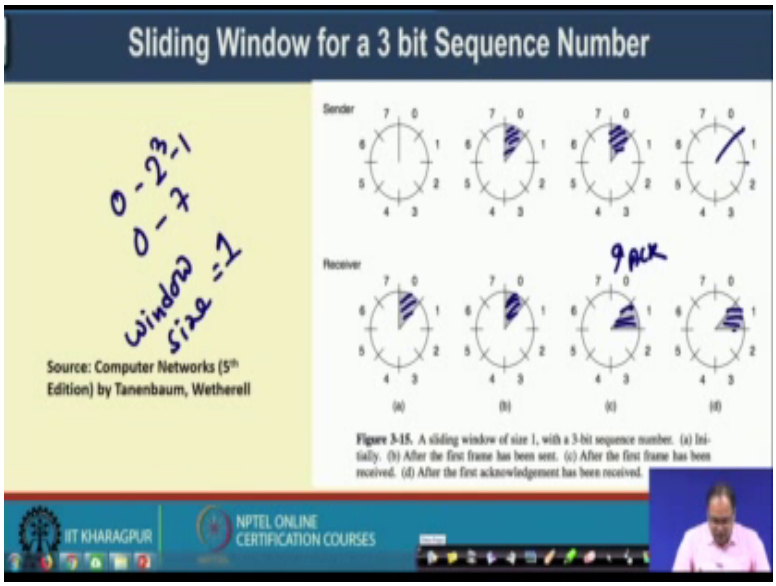
Whenever, the receiver is sending the acknowledgement for frame 0, so, the sender what it does that it shifts the sending window for 1 unit. So, it has already received the acknowledgement for this frame so, it does not bother about this frame anymore because this frame has correctly being transmitted. So, my sending window becomes from 1 to 7,

once you have received acknowledgement for 0. Similarly, the receiver window also get shifted because you have already send a acknowledgement for frame 0; that means, this frame 0 has correctly received and you do not bother about this frame anymore.

Then you send the receiver sends the acknowledgement 1, so, once the sender receives a acknowledge 1, so, frame 1 has also been correctly transmitted it is able to know that information. So, it shifts the sending window for 1 again, now my current sending window becomes 2 2 2 3 4 5 6 7 and again starts from 0. So, this is the repeated sequence number so, we will look into the relation between these windows size and the sequence numbers in the subsequent discussion.

So, that is the idea of this sending window and the receiving window. So, the sending window basically says that you can send that many number of frames without waiting for an acknowledgement. So, if you are if you have sent all the frames of your sending window and you have not received an acknowledgement, you will not be able to send any more frames any further until you are receiving an acknowledgement and you are shifting the frames to the right side. So, that is the broad idea about the sliding window protocol.

(Refer Slide Time: 06:04)



So, here is an example of a sliding window protocol with a 3 bit sequence number. So, if you have a 3 bit sequence number; that means, your sequence number space can be from 0-2 to the power 3 minus 1; that means, from 0-7. So, here is the sequence number from

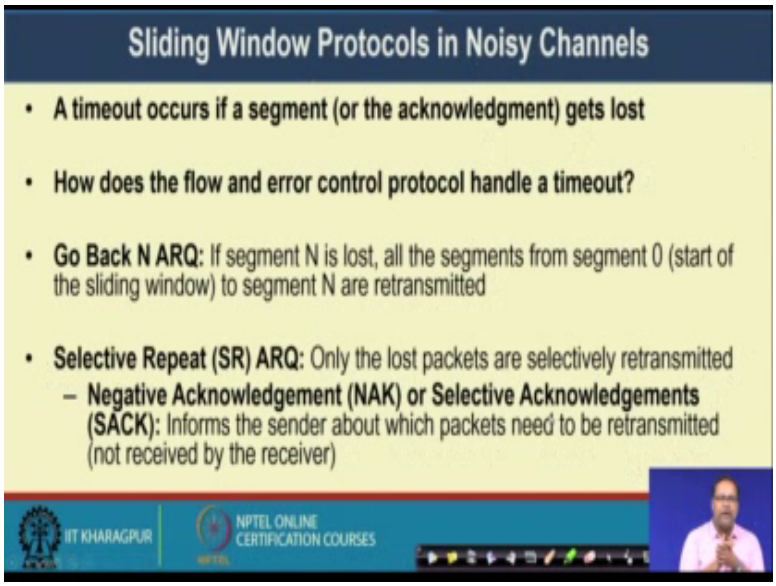
0-7 so, say at the sender side you have sent one frame. So, this is a example where we are using this sequence number field in a circular queue fashion. So, after sequence number 7 again the next frame will be marked as sequence number 0 and we are considering as window size of 1.

So, my window size is equal to 1 so, window size of 1. So, the sender here it has transmitted say 1 frame so, once this sender has transmitter 1 frame the sender is blocked at that position. So, the receiver is expecting for receiving this frame 0, so, the initially the sender has not transmitted any frame, but the receiver is expecting frame 0 receiver is expecting frame 0. Now, sender has transmitted say frame 0 and the sender is blocked here because my window size is 1 and sender is transmitting the frame the frame is on the channel receiver is still expecting frame 0, now say receiver has received frame 0.

So, once receiver has received frame 0 receiver is expecting from frame 1 and receiver has say send back the corresponding acknowledgement. So, the acknowledgement is there in the channel the sender is again blocked with the frame 0, because it has transmitted that. Once the sender receives that acknowledgement so, it has now at this position so, the sender is ready to send frame 1 and the receiver is at this point expecting from frame 1. So, a sliding window protocol window size 1 is somehow synonymous to a stop and wait flow control algorithm, because for every individual frame you are waiting for the acknowledgement, but if you increase the window size gradually you will get the fill of parallelism.

So, if you make the window size 2 from 1; that means, you can send 2 frames in parallel without waiting for the acknowledgement. So, once you are you have sent 2 frames then you wait for the acknowledgement and in between if you receive an acknowledgement you can slide the window. So, that out from there the name sliding window comes, you can slide the window and you can send frame 3 so, that way it is entire sliding window protocol works.

(Refer Slide Time: 08:43)



So, let us look into that how the sliding window protocol works in a noisy channel. So, in case of a noisy channel similar to the stop and wait protocol we also implement similarly a timeout mechanism. So, if you have transmitted data segment if that segment is getting lost or sometime the acknowledgement can also get lost. If the segment or the corresponding acknowledgement get lost and you have already sent all the frames or all the segments in your say sending window and you are waiting for the acknowledgement corresponds to that then, a timeout may occur.

And if a timeout occurs then the question comes that which particular frame you will retransmit. So, the question comes that: how does this ARQ protocol will handle the timeout. So, timeout occur means the receiver was not able to receive the frame correctly and in case of a sliding window protocol you have send the set of frames and for those set of frames the receiver has not received the frames, either the frame has been lost or the corresponding acknowledgement has been lost. So, the question comes that how does sender will react to it this particular loss when a timeout has occurred in the network.

So, there are two different mechanism to handling this timeout, one mechanism is called as go back n ARQ and the second mechanism is called a selective repeat ARQ. Now in case of an go back N ARQ if a particular segment say segment N is lost then, all the segment starting from segment 0, so, here I am assuming that segment 0 is the start of the sliding window to segment N are retransmitted. So, in a broad way that all the frames,

which are there in the current sliding window all those frames are retransmitted if there is a timeout.

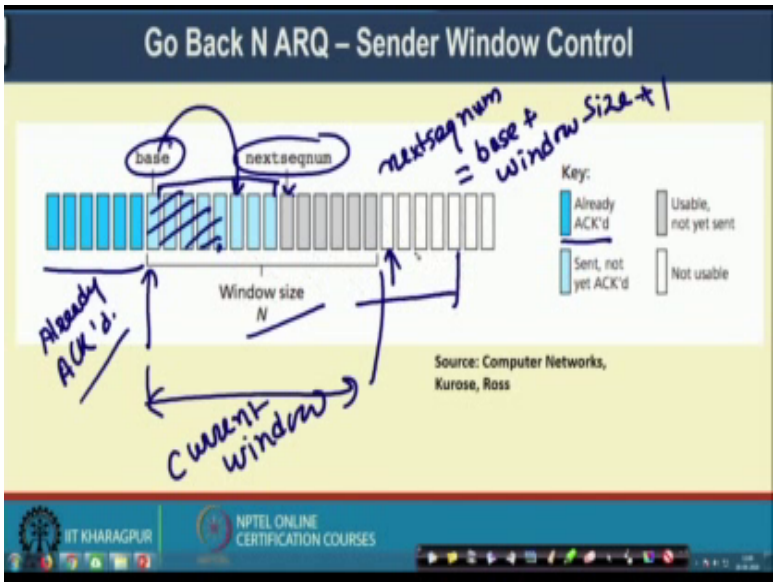
In case of the second methodology which is called as the selective repeat ARQ in case of selective repeat ARQ, you only send the lost packet or you selectively transmit retransmit the packet which has been lost in the channel. Now, whenever we say that you need to selectively transmit the lost packet or which has been not received by the receiver or the corresponding acknowledgement has been lost and the sender has not received that acknowledgement.

Then; obviously, there should be some mechanism to identify those packets because, the first mechanism there go back N ARQ is simple enough. If you are having a timeout; that means, you have not received an acknowledgement and you are not receiving an acknowledgement means you retransmit all the frames, which are there in your window your current window. But, if you are going to selectively retransmitting the frame then you have to identify that which particular frame has been lost.

So, for that we have one special type of acknowledgement which is there in selective repeat ARQ, we call them as the negative acknowledgement NAK or some time in TCP it calls selective acknowledgement or SACK. So, this negative acknowledgement NAK or SACK, they informs the sender about which packets need to be retransmitted; that means, the receiver has not received those packets and it is expecting those packets.

So, those information is passed to the sender with the help of the NAK packet or with the help of this sack packets. So, this NAK and the sack packets the negative acknowledgement and the selective acknowledgement packets, it helps you to find out that which particular packet is expected by the receiver and like that receiver has not received those packets and it returns with only those packets. So, let us look into this two protocols in details the go back N ARQ and the selective repeat ARQ.

(Refer Slide Time: 12:36)



Well first let us look into this go back N ARQ so, this go back N ARQ the sender window implementation is something like this. So, here we maintain two different pointer so, one is called the base pointer so, this base pointer is the pointer from where your current window start. So, your current window starts from here so, this is pointing by the base pointer so, base pointer. So, this indicates that all the frames before this base pointer has been acknowledged.

So, this frames has been already acknowledged. Well so, you have already received acknowledgement for this frames, now these are the frames in your current windows. So, this is the this is your current window so, the base pointer points to the start of the window and in this current window you can send multiple frames without waiting for the acknowledgement and assumed that you have sent up to this frames. So, this setup frames sorry so, not so, this set of frames has been transmitted.

So, this next sequence number this is another pointer so, this next sequence number points to the frame which you can send without waiting for the acknowledgement. So, up to this frame you have already sent and you are waiting for the acknowledgement for these frames and then you can send this particular frame further without waiting for anymore acknowledgement.

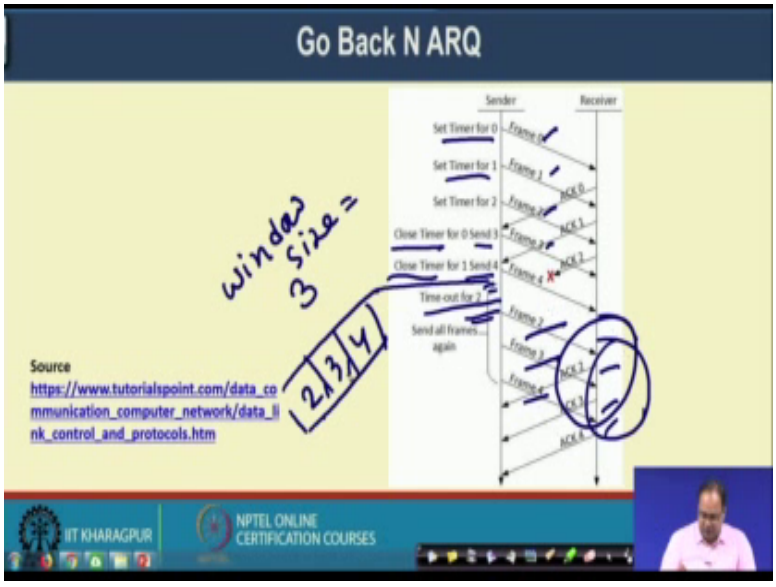
So, that way with the help of these three parameters the base pointer which points to the starting of the current window, the next sequence number the next sequence number

points to the frame which you can transmit without waiting for the acknowledgement and the window size parameter the window size parameter indicates that what is your maximum window size; you can maintain the sliding window at the sender sides sender side for go back ARQ.

Now, here if your next sequence number if your next sequence number becomes equal to your base plus window size; that means, you have transmitted all the frame. So, here actually in this diagram base plus window size plus 1 if it is like this so, if you the next sequence number is pointing to this white frame; that means, it is out of your window current window. So, you cannot transmit this frame unless you are receiving the acknowledgement for this frames which you have already transmitted.

So, once you have received these frames so, you can shift the base pointer. So, if you received the acknowledgement for these 5 frames up to this frame then, you can shift the base pointer from here to here and accordingly the window size becomes here to 5 more frames up to here. So, you will be able to utilize this next sequence number to send more frames in parallel.

(Refer Slide Time: 15:52)



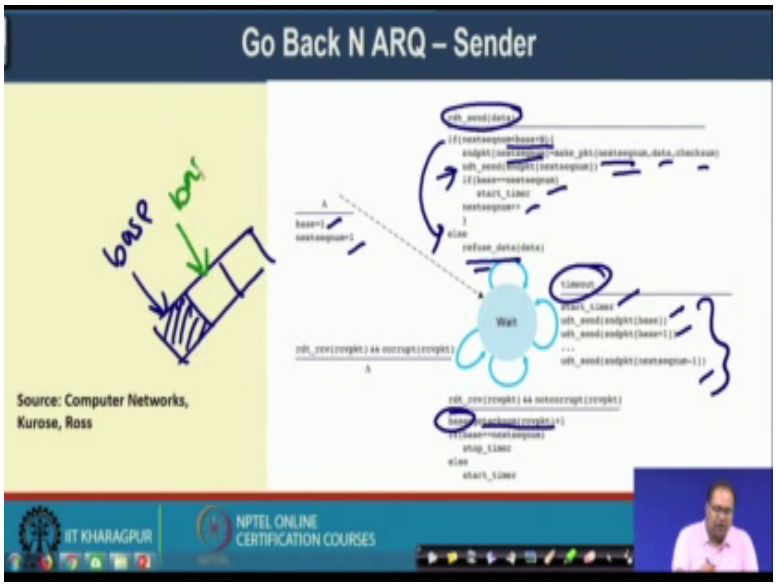
Now the go back N ARQ mechanism in a noisy channel works in this way. So, you have transmitted say frame 0, frame 1 and at this pointer the receiver has sent acknowledgement 0. So, you have transmitted frame 2 then, frame 3 when once you have received the acknowledgement 0 then; that means, you can reset the timer for 0.

And you can send 3 because, it belongs to your current window size you transmit the frame 3 and then you received the acknowledgement number for 1 once you are receiving the acknowledgement for 1 you can reset the timer for 1 and you can send frame 4 because you have you have more provision to send more frames in the current window and you transmit frame 4.

Now, at this point say assume this acknowledgement from receiver gets lost. So, once the acknowledgement from receiver gets from this receiver gets lost so, you are keep on waiting for getting this acknowledgement because, at this point you are full with your current sender window so, we are here assuming that my window size is 3. So, because my window size is 3 so, window size is equal to 3 because my window size is equal to 3. So, you have already received acknowledgement for 0 and 1ah, but now you are you have transmitted frame say transmitted frame 3 3 and 4 and you are waiting for the corresponding acknowledgement. And here this acknowledgement for two got lost so; this timeout for 2 is running.

So, after some time this timeout will for 2 will occur. So, once this timeout for 2 will occur you retransmit all the frames here so, in case of go back N ARQ. So, in your current window you had the frame 2 3 and 4. So, you again retransmit frame 2 frame 2 and frame 4 and again then you can get the acknowledgement immediately for acknowledgement 2, acknowledgement 3 and acknowledgement 4 once, you are getting this 3 acknowledgement 2 3 and 4 then you shift the window further from 2 3 4 to 5 6 7 so, by transmitting those frames. So, that way this entire go back N ARQ protocol once and the abroad idea is here that once this time out for 1 frame occurs then you retransmit all the frames which were there in your current window.

(Refer Slide Time: 18:26)



Well so, this is the implementation of the go back N ARQ so; you are in the waiting state initially. So, you start with the base as 1 and the next sequence number 1, now you are going to getting a call for reliable data send from the application layer. So, you check whether your next sequence number is less than base plus, if your next sequence number is less than base plus N; that means, you are able to send more data. So, you send the packet with that particular sequence number you construct the packet by appending that next sequence number along with the data and the checksum.

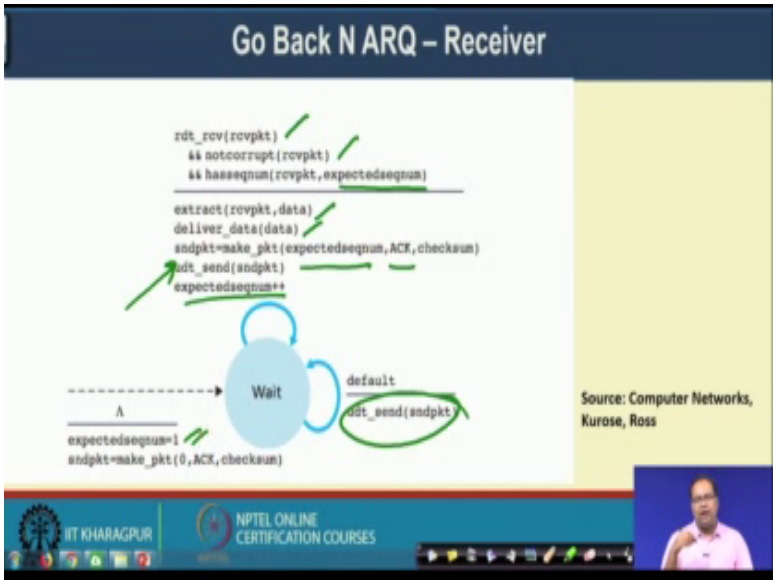
So, that way you are constructing the packet and with this sequence number next sequence number and then you are utilizing this on reliable channel at the network layer to transfer this protocol. Now, if your basis becomes equal to the next sequence number you start the timer and increment the next sequence number ok. So, that is the case and otherwise you refuse the data so, otherwise you refuse the data means from this else is coming. So, if your next sequence number is greater than this plus; that means, you are trying to transfer a frame you are receiving a frame which is outside your current window. So, if it is outside your current window you do not transmit that frame.

Now, if a time out occurs you again start the timer and sent all the packets from base to next sequence number minus 1. So, you transmit all the frames which are there in your current window. Now whenever you are getting a packet and that packet is not corrupted, in that case you are checking that what is the corresponding acknowledgement number

and you are updating the base pointer accordingly. So, you are updating the base pointer according to the acknowledgement number that you have received plus 1 so; that means, if this is your window size and this was the base pointer now you have receive the acknowledgement for this one.

So, once you have received acknowledgement for base one this one, then you move the base pointer to your next frame. And if the packet is corrupted then you do not bother about that you are again in the wait loop. So, that way you can implement this go back N ARQ mechanism.

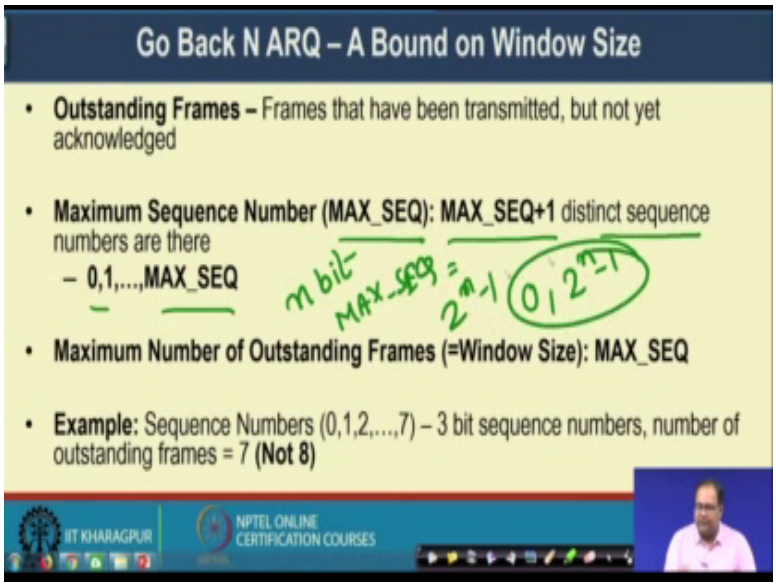
(Refer Slide Time: 20:46)



Now, similarly at the receiver side the receiver side whenever it is receiving a packet the packet is not corrupted and it has the sequence number which is equal to the expected sequence number. So, you extract the packet deliver the data to the application, construct an acknowledgement with the expected sequence number and use the unreliable channel at the network layer to transfer the packets and implement your expected sequence number.

And if the default case is you transmit that acknowledgement and otherwise you just wait it is expected sequence number you initiate the system with expected sequence number 1. So, that is the way we implement the receiver at the receiver side for go back N ARQ.

(Refer Slide Time: 21:41)

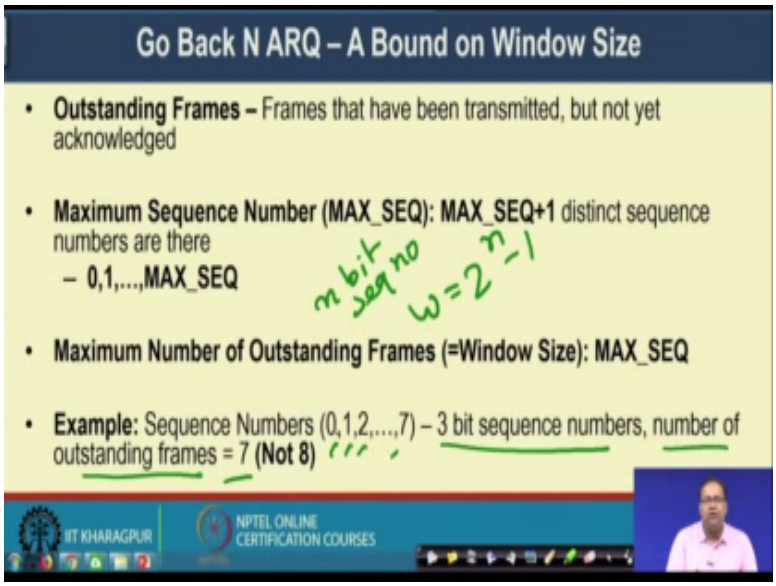


Now, let us look into that what is the relation between the window size at the sequence number in case of go back N ARQ. Now, in case of go back N ARQ the frames that have been transmitted, but not yet acknowledgement acknowledged those frames we call as the outstanding frames. Now assume that MAX sequence MAX SEQ MAX sequence this your maximum sequence number.

So, if MAX sequence is your maximum sequence numbers then you have MAX sequence plus 1 distinct sequence numbers from 0 to up to MAX sequence. So, this is the available sequence number space which is there with you. So, if you are using say n bit sequence number then this MAX sequence is 2 to the power n minus 1. Well so, you have from 0 to 2 to the power n minus 1 is the your total sequence number space.

Now, this maximum number of outstanding frames; assume that it is equal to the window size so; that means, your windows size is equal to max sequence. So, that as an example in case of our go back n ARQ protocol what we try to ensure, we try to ensure that your windows size is equal to max sequence.

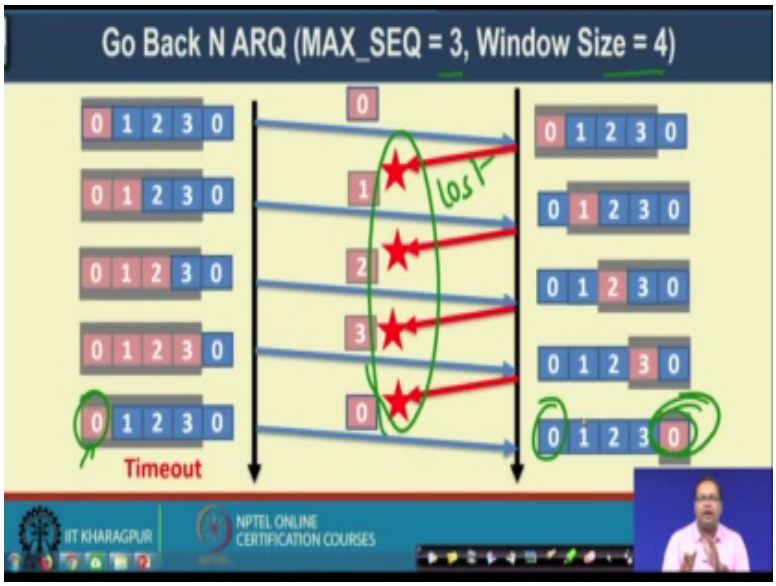
(Refer Slide Time: 23:06)



That means if you are using n bit sequence number then, your window size w will be equal to 2 to the power n minus 1. So, you can see that it is always 1 less than the total number of distinct sequence numbers that you have. So, for example, if your sequence numbers are some 0 to 7; that means, you are using a 3 bit sequence numbers.

So, you can have maximum number of outstanding frame equal to 7. So, your window size can be equal to 7 and it is not 8 so, you have 8 distinct sequence numbers here from 0 to 7, but we are not making window size equal to 8 rather we are making window size equal to 7. So, let us see why so, let us see that why my window size is equal to max sequence and not equal to max sequence plus 1 all do I have max sequence plus 1distinct sequence numbers.

(Refer Slide Time: 23:59)



So, here is an example here I am taking an example, where your max sequence is equal to 3 and window size is equal to 4. So, max sequence equal to 3 means you have 4 different sequence number 0 1 2 and 3 and I am making window size equal to that. So, if that is the case now think of a scenario when you have you this is your current window size you have transferred frame 0. So, it has received frame 0 send back the acknowledgement, but the acknowledgement is lost then, the sender has transmitted frame 1.

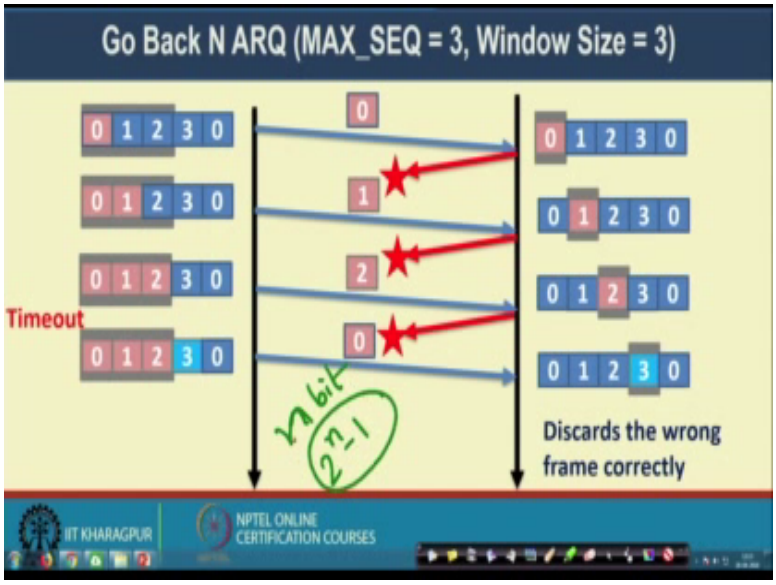
So, it has already send the acknowledgements so, the receiver has shifted its window. So, receiver has shifted it is window and receiver has send the acknowledgement 1, again this acknowledgement got lost. The sender has transmitted frame 2 receiver has receive this frame to shifted it is receiving window size and it has transmitted the acknowledgement that acknowledgement got lost, the sender has finally, sent 3. So, the 3 has being received by the receiver so, receive has received that frame it has shifted the window and it has send the acknowledgement again acknowledgement got lost.

Now, here is a typical example, where the receiver has correctly received all the frames, but it was not able to send the acknowledgement, it has send the acknowledgement, but all those acknowledgement got lost in the channel so, all this acknowledgement got lost. Now in this case, now the sender will experience a timeout because the sender has not received any acknowledgement from the receiver side. So, sender will get a timeout and

sender will send this frame 0 so, when the sender is sending this frame 0 unfortunately here in the receiver side you can see the receiver is also expecting frame 0, but that is the next group of frame.

So, this is the next group of frames not the intended frames. So, this was the actual intended frame, but you have sent a frame with sequence number 0, you have send this frame, but a receiver will correctly think this frame has this expected frame 0 which is not the correct frame. So, we see that there is a problem here when all the acknowledgement got lost in the channel.

(Refer Slide Time: 26:16)



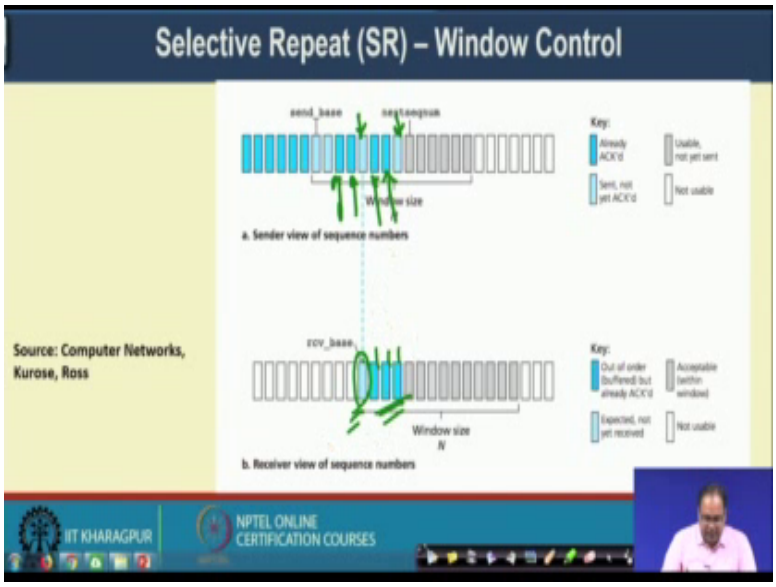
Now, let us see the an example that when your max sequence is 3, but the window sizes also 3 and the window size is not 4. In that case, the sender has send 0 1 2 3 1 and 2 because my window size is 3 you can send maximum up to 2 and then you have to wait for an acknowledgement. Now in this case, if all the acknowledgement gets drop so, you can see that the receiver is now expecting from frame 3 and the sender will retransmit frame 0 or better to say from 0 to frame 2. Now whenever the receiver is sending the frame 0 to frame 2 the receiver will be able to correctly decode that these are not the expected frame.

So, it will be able to discard the wrong frames correctly and send an acknowledgement to the sender saying that this frames I have received. So, it will be able to find out that will the centre has possibly not received the acknowledgement it will retransmit those

acknowledgement. If the sender gets back those acknowledgement then the sender can shift the window further and start transmitting the data from frame 3. So, you can see here in this example that with maximum sequence as 3 and window size has 3 will be able to correctly find out that whether retransmission is a duplicate 1 is a delayed duplicate or a new 1.

So, here in this case the receiver is expecting frame 3 because, retransmission you are retransmitting frame 0 1 2 the receiver will be able to correctly decode that which was not possible in the earlier case that we have looked into. So, because of this we keep window size 1 less than the maximum sequence number space that you have. So, if you have n bit if you have n bit window size then we have; sorry, if you have n bit sequence number then, you have 2 to the power n minus 1 as your maximum window size.

(Refer Slide Time: 28:22)



Now let us look into the selective repeat protocol. In case of selective repeat protocol what you can do that you can acknowledge intermediate frames and some frames may remain unacknowledged. So, what we have seen that in this particular case in case of selective repeat.

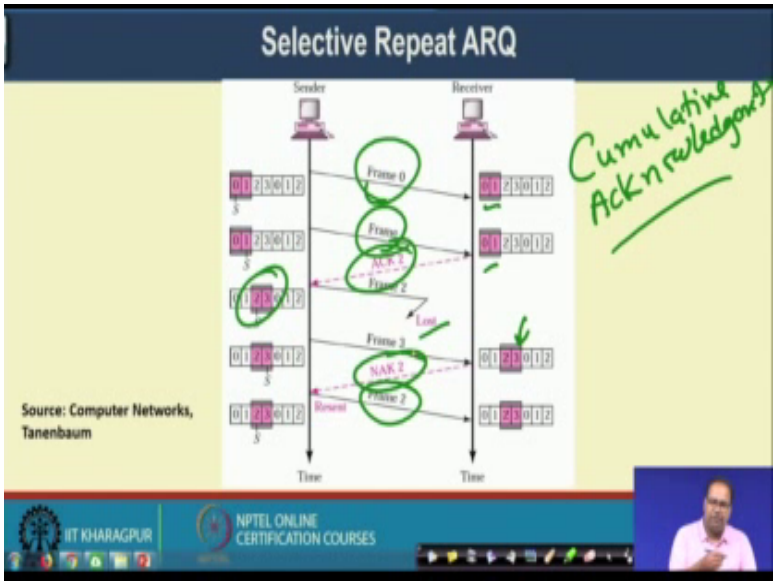
You do not need to need to retransmit all the frames once there is a timeout rather you selectively retransmit the frames. So, here in the sender window size; it may happen that well there are some intermediate frames here this had been same, but not yet

acknowledged, but the frames here this frames they got acknowledged so, this intermediate acknowledgements are there.

Now in the receiver side the this is the receiver view of the window size so, the receiver it also has a base pointer, this base pointer points that well the receiver is expecting to receive this particular frame. And, it has not received this frame and it has received all other frames which is received out of order because this particular frame has not been received yet rather you have received this frames. So, this has receipt out of order so, you have put those frames in the buffer and send the acknowledgement for them.

So, that is the view of window for the sender and the receiver where certain intermediate packets got acknowledged and some of the packets that are not got acknowledged and the packets which are not got acknowledge that need to be retransmitted. So, this is the idea of the selective repeat ARQ.

(Refer Slide Time: 30:04)

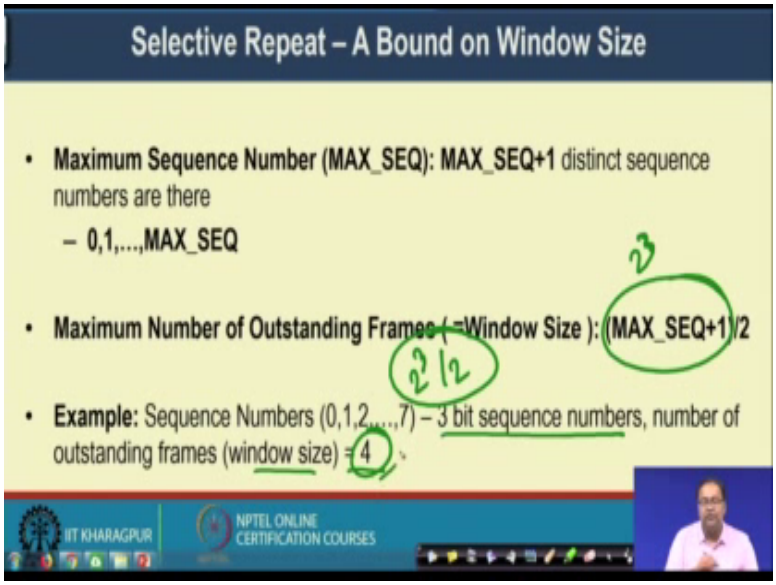


So, you are transmitting frame 0, then you are transmitting frame 1 at this time you are you have transmitted both frame 0's and 1's. So, here we use something called a cumulative acknowledgement so, this cumulative acknowledgement says that well; once you are receiving an acknowledgement 2; that means, frame 0's and 1's have been received correctly and you are expecting for frame 2.

Now, assume that you have transmitted frame 2 once you have got this acknowledgement the sender shifts the window, once the sender shift the window and it has the transmitted frame 2 assume that that frame 2 got lost and then it is able to send frame 3. So, it has transmitted frame 3 so, the receiver has received frame 3 once the receiver has received frame 3 then it has received this frame out of order because it has not received frame 2, but it has received frame 3.

So, it put frame 3 in the buffer and sends a negative acknowledgement. So, once the sender receives this negative acknowledgement it retransmits frame 2. So, that is the idea of the selective repeat ARQ where you can send this negative acknowledgement and keep the out of order frames in the buffer. And from this negative acknowledgment the sender will be able to understand that which frames needs to be retransmitted and accordingly those frames are being re transmitted.

(Refer Slide Time: 31:32)

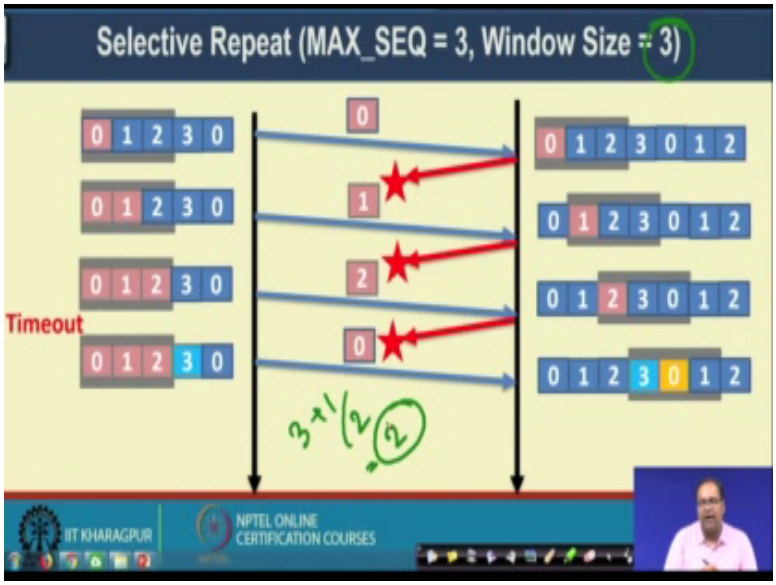


Well, similarly we look into a bound on the window size for selective repeat similar to the earlier case. So, we have MAX sequence plus 1 distinct sequence numbers from 0 to MAX sequence, but in this case in selective repeat my window size will be max sequence plus 1 divided by 2. So, an example is that if you have 3 bit sequence numbers from 0 to 7.

So, you have 8 different sequence numbers so, your number of outstanding frames; that means, your maximum window size will be equal to 2 to the power 3 divided by 2 which

is equal to 4. So, 2 to the power 3 divided by 2; that means, MAX sequence plus 1's becomes 2 to the power 3 2 to the power 3 divided by 2 which is becomes equal to 4.

(Refer Slide Time: 32:24)



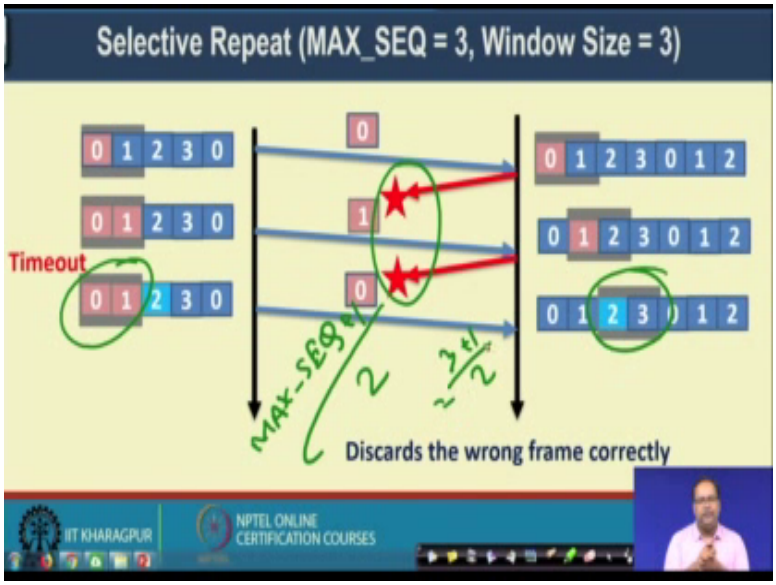
So, let us see an example that why this holds true so, similar to the earlier case here you need to remember that whenever you are sending certain frames the sender can send acknowledgement in between. So, negative acknowledgement in between. So, here the sender has sent so, I have taken an example where my MAX sequence is 3. So, if my MAX sequence is 3; that means, my earlier ideal window size should be equal to 2 to the power if my max sequence is 3 then my window size should be equal to 3 plus 1 divided by 2; that means, equal to 2, but here we are using a window size 3 1 more than the actual it is window size that we should use.

So, here let us see that what is the problem if my window size is 3, then the sender sends all the frames 0 1 and 2 and then similar to the earlier case that all the acknowledgements are lost. Now, if all the acknowledgement are lost the receiver has received frame 0 1 and 2 so, now, the receiver expecting frame 3 0 and 1 and sender gets a time out once the sender gets the time out sender starts sending frame 0. So, once the sender frame 0, now in this case remember that the receiver can receive frames out of order which were not possible in the case of go back an ARQ.

But because the receiver can receive frames out of order receiver will think this 0, as this 0 which it was expecting, but this was not the 0 it 0 that was being transmitted this 0 was

being transmitted so, there would be a confusion here. So, let us see that by utilizing window size as 3 plus 1 by 2 equal to 2 how can we solve the problem.

(Refer Slide Time: 34:12)



So, here we keep the window size as 2 so, if I am keeping window size as 2 then the sender sends 0 and 1 and waiting for the acknowledgement similarly all the acknowledgement got lost. So, it has send frame 0 and 1 and so, it is expecting frame 2 and 3, but if there is a time out here the sender retransmits frame 0. When the sender is re transmitting frame 0 it can correctly find out that it is expecting frame 2 and 3 not frame 0 and 1 so, it can discard the frame correctly. So, we can see that particular confusion which was there that can get resolved if I using window size as max sequence plus 1 by 2.

So, here it is equal to 2 3 plus 1 by 2 equal to so, with this particular window size we are able to resolve this particular confusion. So, that is all about the sliding window based flow control algorithms the go back N ARQ and selective repeat ARQ mechanism which helps you to send the packets in a pipeline fashion and at the same time helps you to resolve for loss. So, in the next class we will look into some other aspects performance aspects of transport layer protocol and during the discussion of TCP we will see that how this flow control algorithms are actually implemented in TCP type of protocol.

So, thank you all for attending the class.