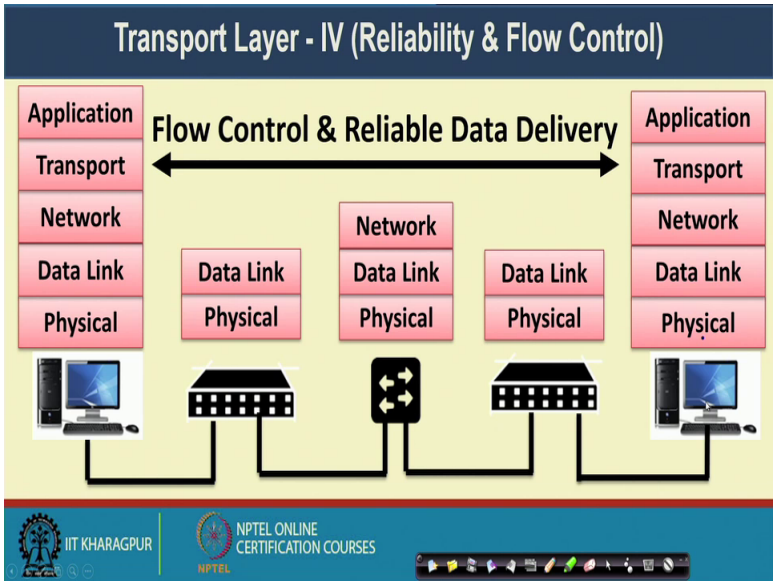
Computer Networks and Internet Protocol
Prof. Sandip Chakraborthy
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 14
Transport Layer – IV (Reliability)

Welcome back to the course on Computer Networks and Internet Protocol. So, in the last classes you have looked various services in transport layer of the protocol stack, and we have looked into in details the connection establishment and the connection termination procedure. So, today we look into the second service which is the flow control and the reliability in transport layer.

(Refer Slide Time: 00:40)



So, the flow control and reliability ensures that whenever you have establish certain end to end connection between the two remote host; so, on top of that connection; now you need to send a sequence of data base.

So, in transport layer in TCP kind of protocol you send it in the form of sequence of bytes. On the other hand in certain version of transport protocols the data is transmitted in the form of sequence of packets or sequence of segments. So, in this end to end connection which is being established on the two end of the devices; the two end devices. Here what objective is the first objective is that the sender should not send more data compare to what the receiver can receive.

So, this particular methodology or this particular philosophy, we call it as a flow control algorithm. So, the flow control algorithm ensures that the sender is always aware about what a receiver can receive or how much data the receiver can receive. And, accordingly the sender adjusted its rate of transmission; such that it does not overshoot the rate at which the receiver can receive. At the same time as we have discussed earlier that the lower layer of the protocol stack is kind of unreliable.

So, whenever the network layer forwards packets or forwards data to an remote host, the network layer does not ensure that there is a kind of guaranty or assurance that your data will be delivered at the other end. So, it makes a best try to deliver your data based on your destination at this and finding out the intermediate hubs through which the packet need to be delivered. But on the other hand this network layer it does not bother about that who much data need to be push to the network or how much data need to be transferred.

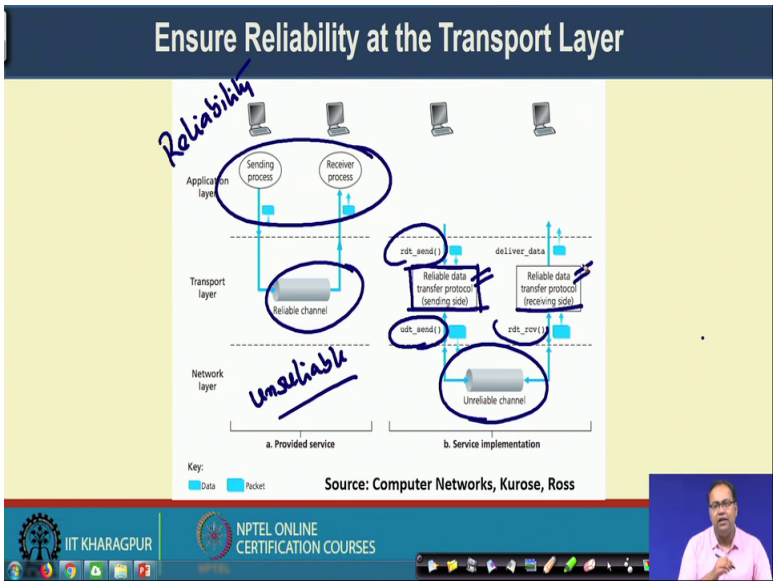
And as a result what happens in a packet switching network that I described in the last class that the intermediate buffers, at the intermediate network devices that at the routers. They may get filled up and you may experience a buffer over flow from those intermediate routers. And because of that whenever you are delivering a data through this network layer using the IP delivery method, there is always a possibility that a considerable number of data frames or data segments; they are getting dropped from the from the network layer.

Now, the task of the transport layer is to consider that to find out or to sends whether certain data has been transferred correctly at the other end or not and if it is not transfer at the other end correctly. Then apply this reliability mechanism to ensure that every message which is send by the application layer that is getting delivered at the other end eventually.

So, the broad idea is that you send the channel, you sends the media to find out whether the data is being transmitted correctly or not, if the data is not being transmitted correctly. That means, the other end is not able to receive the data if you are able to find it out, then retransmit the data to make sure that eventually the data is received by the other end receiver.

Now, in a typical transport layer this flow control and reliability are implemented together. So, we look into the different mechanism through which we implement the flow control and the reliability algorithms over the transport layer of a TCP IP protocol stack. So, let us proceed with the detail mechanism of that one.

(Refer Slide Time: 04:25)



So, this is the broad idea of ensuring reliability. So, as I have mentioned that your network layer provides you an unreliable channel. So, the network layer does not guaranty that the packet that it is trying to deliver at the other end it will be eventually transmitted. So, there is always a possibility of buffer overflow from the intermediate routers because of which there is a possibility of data loss.

Now, we are we are having some hypothetical function or hypothetical method which is through which we are making a interaction among different layers to make you understand about the entire procedure. Now if you just think about from the application perspective, the application always wants or the application always try to have an methodology through which you will be able to ensure reliability.

So, that means, the application always wants that if a sender is or a sending process is sending certain message or certain data that data will be eventually received by the receiver. So, the receiver will receive all the messages. So, you can just think of an application like a file transfer. So, the entire large file is divided into multiple chains and then the sender process sending the data bits for that file.

Now, at the receiver end you need to ensure that all the data that is send by the sender that is received eventually otherwise you will not be able to reconstruct the entire file. So, that is why from the application perspective reliability is at most importance for many of the application. But as I have mentioned earlier that they are exist a certain group of applications where reliability are is not that much important; rather delivering the data or delivering the messages more important within a pre defined time or duration. And in that cases we use UTP type of protocols where reliability is not a concern or reliability is not used. But for the application for the set of applications when reliability is a concern during that time the application always expects that the data or the message that we transferred from the application. They will be eventually received at the receiver side.

Now, the application here expects a reliable channel. So, the question comes that your network layer is unreliable. So, on these unreliable channels, how can you express or how can you write a methodology to ensure reliability. So, the idea is there that at the transport layer, you have this reliable data transfer protocol at the sender side and the receiver side.

So, this sending process at the network layer it is unreliable send udt send that express a unreliable way of sending the data. On top of that you are implementing this reliability mechanism which is ensuring that reliable data send on top of the transport layer.

So, whenever you are making a interfacing between the network layer at the transport layer, there you have this unreliable data send and at the interface of transport layer and the application layer, you have the interface of reliable data send.

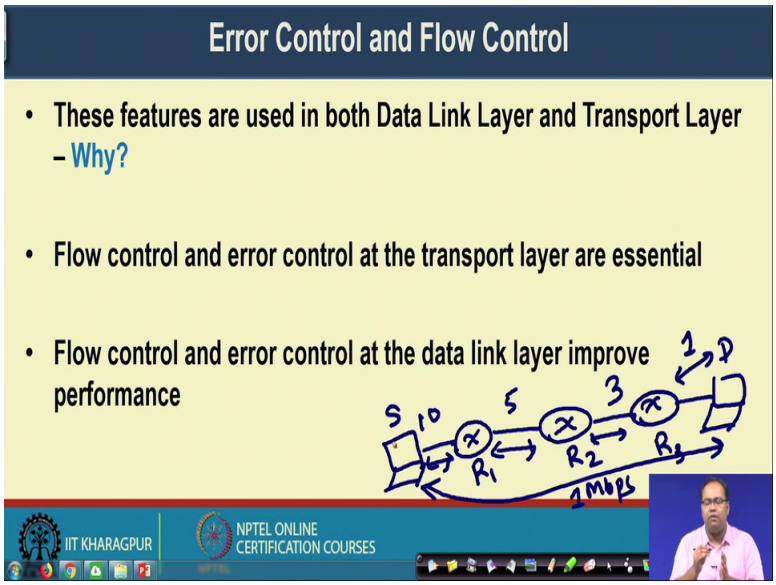
Now, at the other end it receives the data in a reliable way because of this reliable data send mechanism which is there and this reliable data transfer protocol at the receiver side; eventually it will receive the message and it will deliver the data to the application layer.

So, that way the application layer will always expect a reliable delivery of the messages. And here we will see that in the transport layer, how can you implement this two mechanism at the sender side, and at the receiver side which will help you to ensure reliability in the system.

So, let us look into this process in details. So, as I discussed earlier during the discussion of a different layer of the TCP IP protocol stack that certain services are implemented in multiple layers of the protocol stack. So, this flow control and error control these are the two mechanism which are implemented both at the transport layer and the data link layer. So, the questions come that; why do we need to implement flow control at the transport layer as well as the data link layer.

So, if you if you just ask the question in this way that let us assume that I have my flow control algorithm in the data link layer which being implemented, do I need to have the flow control algorithm at the transport layer itself?

(Refer Slide Time: 09:05)



So, let us look into one example where you have this flow control algorithm at the data link layer. So, the data link layer ensures this hop by hop flow control algorithm. So, as we have leaned already that your data link layer protocol. It ensures the hop by hop principles the hop by hop transmission of data where as the transport it ensures the complete end to end delivery of the data. Now whenever we are saying that the flow control algorithm is implemented at the data link layer, it is like that this hop by hop flow controls are implemented.

So, if you think about this as the intermediate routers R1, R2, R3 and this is the source and this is the final destination. So, these flow control algorithm at the data link layer; it

ensures that you have flow control mechanism between S and R R1 between R1 and R2 between R2 and R3 and between R3 and the D.

Now so, this hop by hop flow control algorithms are there. Now the questions comes if I have this hop by hop flow control algorithm, do I still need to have a flow control algorithm at the transport layer? Now just think of a scenario that well this link from S to R1, it is 10 mbps; the link from R1 to R2, it is 5 mbps; the links from R2 to R 3, it is 3 mbps and the link from R3 to this D this is 1 mbps.

Now, what happens here that whenever you are implementing this hop by hop flow control at the data link layer, then from S to R1, the S finds out that well I can send the data at a rate of 10 mbps. So, it sends the data at a rate of 10 mbps. But then R1 finds out that it will not be able to send the data to R2 at a rate of 10 mbps, although it is receiving the data at a rate of 10 mbps, but it requires 3 mbps of transmission. Similarly R2 it finds out that it will not be able to send data at a rate of 5 mbps; rather it need to sends the data at 3 mbps and finally, from R3 to d it can only send the data at a rate of 1 mbps.

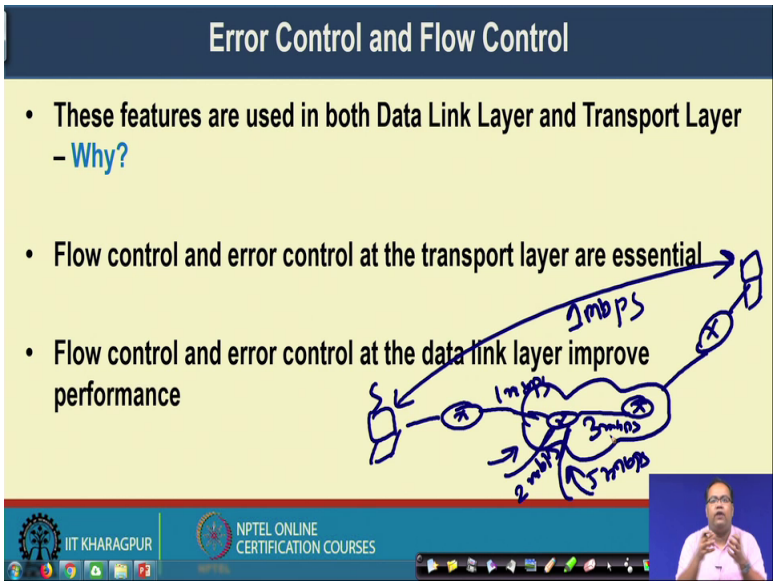
Now, if S does not know that this entire so, if you look into the effective rate of this entire end to end path. So, this effective rate of this entire end to end path will be equal to 1 mbps. Now if S does not get this information, it will try to push the data at a rate of 10 mbps and what will happen that R1 will not be able to deliver the data to R2 and so on.

So, that way that additional data which is coming to R1; so, R1 is receiving the data at a rate of 10 mbps, but it is only able to deliver the data at a rate of 5 mbps. So, that as a result this additional data that it is receiving it will get on filling up the buffer space which is available at R1 and eventually what will happen that eventually we will experience the data loss from the buffer due to buffer over flow.

So, there will be a huge amount of data loss because source is transmitting data at a rate of 10 mbps, but the receiver the other end, receiver is only able to receive data at a rate of 1 mbps. And as a result this 9 mbps of data that will get accumulated over the different layers of buffers at different intermediate routers; and after some time it will experience a drop of data from those intermediate buffers, and this the reason that we are not able to implement. If we implement this flow control algorithm only at the data link layer for hop by hop flow control, it is not sufficient we have to implement it at the transport layer.

Now, let us look into the other way around like you have this end to end flow control algorithm and in that case, still we still we require that this flow control algorithm at data link layer or not. Now in transport layer; what happens that you are only ensuring the end to end data delivery or end to end flow control algorithm?

(Refer Slide Time: 13:03)



So, you have one router with which your source is getting connected. Then you have this intermediate network, then another router and then your destination which is there. And you are only ensuring the flow control among these two end host and just think about the earlier example that you are sending data at a rate of 1 mbps.

Now, you just think of two intermediate routers here in the network. Now this router has multiple incoming ports. So, it is just like a road network that you are getting data from multiple parts all together. So, you are getting data from this link, you are getting data from this link, you are getting data from this link and so on.

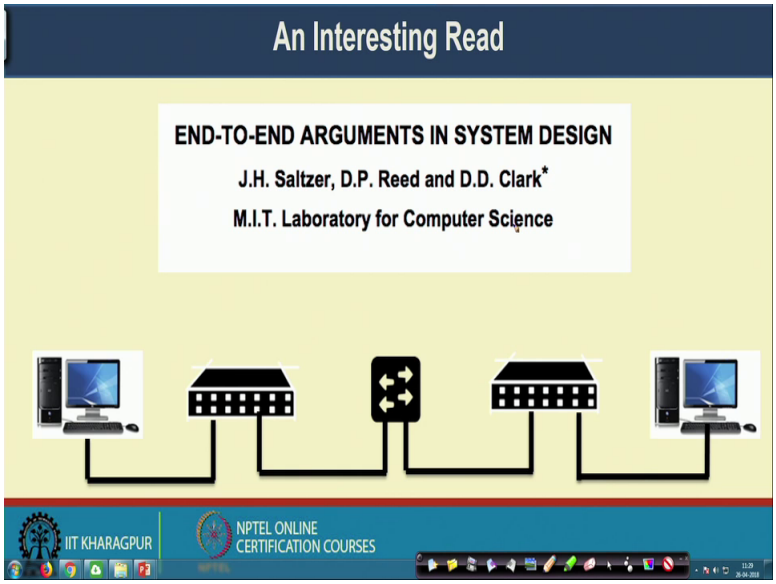
So, that way it may happen that which link is pushing data at a rate of 1 mbps, but this individual links may push data at a rate of 2 mbps and say this link is pushing data at a rate of 5 mbps. But this out going link it only has a speed of say 3 mbps. In that case because this multiple incoming link are getting converged in a intermediate router, it may happen that the total incoming rate which is being there from multiple others incoming links that is exiting the total out going capacity that the router has. And because of that this end to end flow control algorithm, it may perform poorly in this kind of scenario. So,

that is why to make it control to have a control, you need hop by hop flow control mechanism in the network. But here you will see by applying this hop by hop flow control algorithm in the network. So, what your task would be your task would be to reduce the incoming rate at every individuate hop such that intermediate routers is experience less congestion less amount of congestion

So, by doing that you are effectively improving the performance of the system, but remember that by even after implementing the hop by hop flow control, it may not be possible to possible to ensure complete reliability that a complete elimination of data loss. Because you are receiving data from multiple hops and every, where there is a kind of estimation going on that what should be the ideal rate at which the one ensure send the data at the other end, and this is the estimation takes some time. So, before the systems moves to the convergence there is always a possibility of having a significant amount of data loss.

So, you will not be able to completely eliminate the data loss by applying this hop by hop flow control mechanism, but certainly you can improve the performance. And that is the reason we say that flow control mechanism, the error control we will discuss later on.

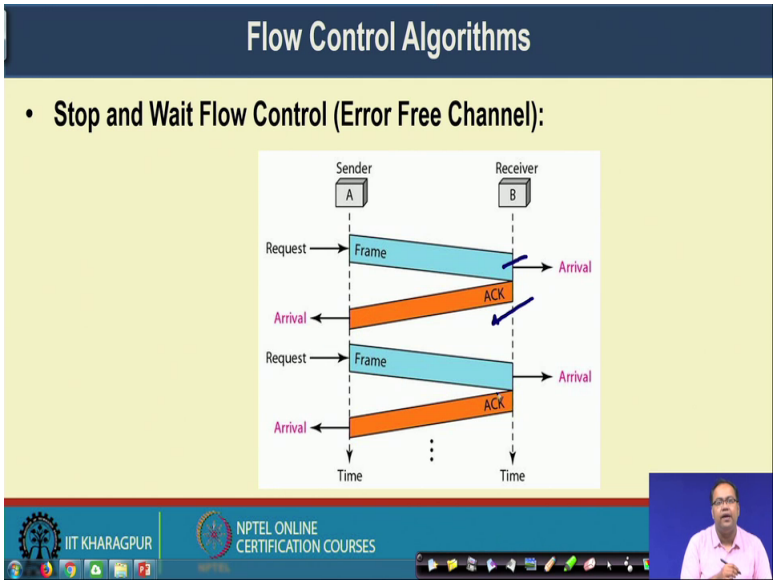
(Refer Slide Time: 16:03)



The flow control and this error control mechanism at the transport layer; they are essential whereas the flow control and the error control mechanism and the data link layer they improve the performance of your protocol.

Now, let us look into different type of flow control algorithms and before going to that, why do we require this different kind of end to end protocols in the network. So, for that I suggest all of you to read through this particular paper which was which was published by Saltzer Reed and Clark. So, it is a fundamental paper in Computer Networking that talks about that, why do you require this kind of end to end protocols in the internet when there is this hop by hop protocols already existing. So, I suggest all of you to read this particular paper to get more details about the principles which you are adopted in the TCP IP protocol stack to implement this kind of end to end protocols over the internet.

(Refer Slide Time: 17:02)



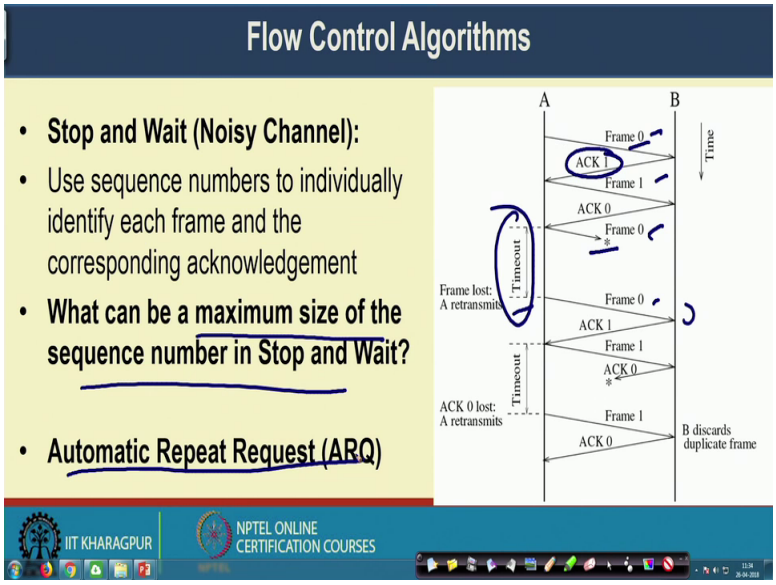
Now, let us look into this flow control algorithm. So, the simplest flow control algorithm that we have, we call it as a stop and wait flow control algorithms. So, the stop and wait flow control algorithm in a error three channel works as follows. The protocol is pretty simple that you send a frame or you send a packet and then, you wait for its acknowledgement.

So, once the receiver receives this frame, it sends back an acknowledgement and the sender it waits for the acknowledgement before sending the next frame or the next packet. So, once you are receiving this acknowledgement, then you only send the next frame. So, that way every frame has an acknowledgement associated with it and only when you receive the acknowledgement you transmit the next frame.

Now, if it is a error free channel it is always guaranteed that eventually the receiver will receive the frame and it will be able to send you back the acknowledgement and the sender will eventually receive the acknowledgement, because it is an error free channel and there is no loss. So, once you are receiving the acknowledgement, you are sure that the receiver has received this particular frame and so, you transmit the next frame.

So, that is the broad idea of this; stop and wait the protocols. So, you stop after sending transmission, start sending the next frame, after sending one frame then, wait for the acknowledgement; one you have received the wait, then you send the next frame.

(Refer Slide Time: 18:21)



Now, let us look into this flow control algorithm in a noisy channel the same Stop and Wait protocol. So, here we use the concept of sequence numbers to individually identify each frame and the corresponding acknowledgement. So, every frame is associated with one sequence number. So, if you look into this example this frame 0 is associated with a sequence number. So, this 0 is the sequence number for this frame and then, we are getting an acknowledgment. So, this acknowledgement mechanism it is sending acknowledgement 1; it is sending acknowledgement 1 means that B has correctly received frame 0 and then it is expecting for frame 1.

So, then you send the frame 1. So, once you have B has receive frame 1, then it sends acknowledge 0; that means, B has received frame 1 and it is waiting for the acknowledgement 0. So, that way in a noisy channel, the first reason is the first principle

is that you separate out every frame with by using the a corresponding sequence number which will be uniquely identify every frame in the channel.

Now, because it is a noisy channel, there is a possibility of having a frame loss because this frame is being lost from the network. So, if there is a frame loss, so we will not send back any acknowledgement. So, you wait for a timeout value. So, A will wait for a timeout value and once this timeout expires, then a will returns with the frame.

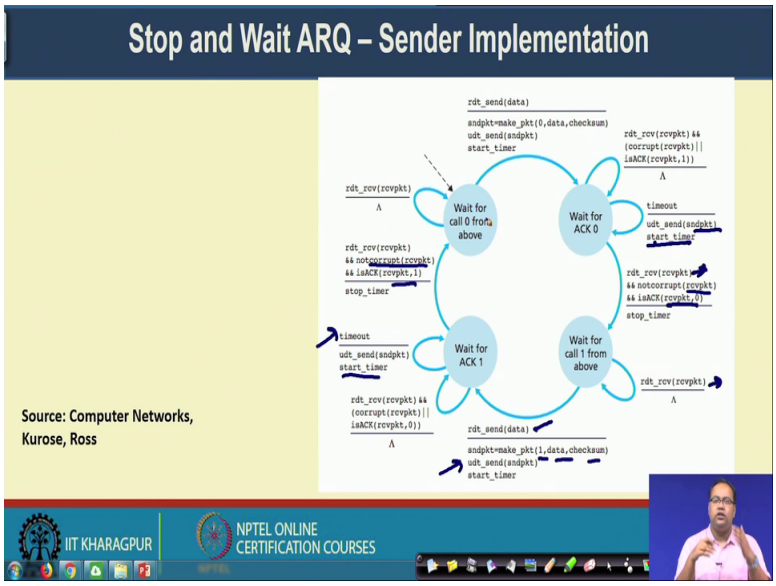
Now, one interesting question here is that what can be the maximum size of the sequence number in stop and wait. So, in stop and wait you can see that at one instance of time, only one frame can be outstanding in the network. So, whenever you have send frame 0, unless you are getting the acknowledgement from that frame; you will not send the next frame. So, that is why 2 bit sequence number will be sufficient and because of this reason you can see from this diagram that, every frame is associated with one sequence number and the sequence numbers are 0's and 1's. So, repeated 0's and 1's so; that means, whenever you have send frame 0, unless you are getting this acknowledgement with the expected frame 1; you will not send frame 1.

So, you can always be sure by looking at the acknowledgement that which for which frame this acknowledgement corresponds to. If you are getting an acknowledgement 1; that means, you have correctly received frame 0 and you are expecting from for frame 1. So, that is the reason that we use to bit sequence number for a this flow control algorithm using stop and wait and this kind of algorithm where we are utilizing the concept of sequence number. And applying flow control algorithm in case of a noisy channel and you can see that by applying this flow control algorithm in a noisy channel. We are also ensuring reliability in the system.

So, we are also ensuring that the receiver receives all the frames correctly. So, if there are if a timeout occurs then you retransmit that frame again. So, that this frame 0 eventually receives by the receiver B. So, that way you are also ensuring reliability in the system and this flow control and reliability algorithm all together, we call it as a automatic repeat request or ARQ algorithms.

So, now onwards we look into these ARQ algorithms in details; the different versions or the different variance of ARQ algorithms.

(Refer Slide Time: 21:41)



Well, so this is a kind of sender side implementation of this stop and wait ARQ algorithm in the form of a straight transition diagram. So, here the thing is that this you can think of the initial state.

So, the initial state talks about that you wait for call 0 from above; that means, you are waiting for frame 0 initially. So, the event which is there it is in the sender side. So, you are from the application layer you are sending a reliable data delivery. Now once this things happens. So, at the transport layer side, you are making a call to reliable data delivery to send with this data.

Now, how it ensures the reliable data delivery? So, you need to map this rdt to the corresponding udt call because if you remember that earlier, we have seen that at the network layer the calls are unreliable calls like this udt calls. So, for that what you are doing that you are appending one sequence number with this packet you are providing the data and some checks some checksum, we will discuss later on to ensure the error free transmission of the data. Then you are sending the data over unreliable channel and starting the timer

Now, whenever you are waiting for the acknowledgement 0, you are moving to the state that you have sent the packet 0. So, you are waiting for the corresponding acknowledgement. So, here once you are receiving a packet and if you are finding out that the packet is corrupted and in that case, you are in the same loop you again wait for

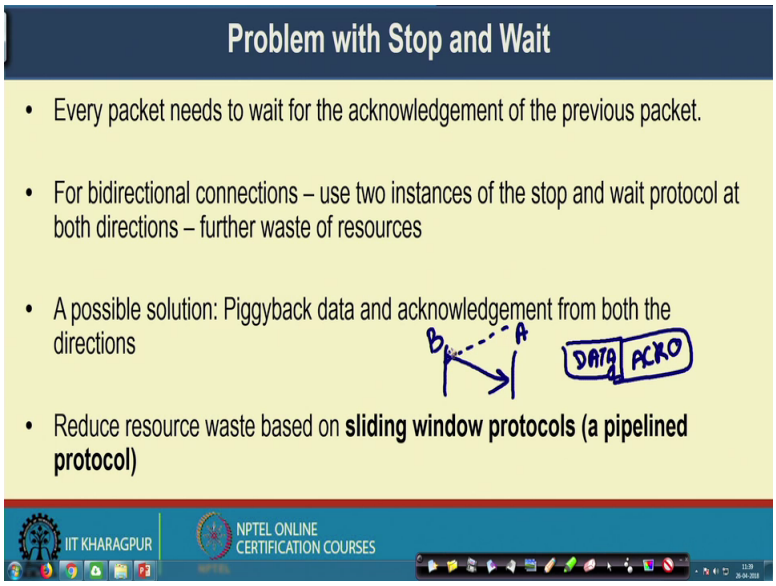
an acknowledgement. If a timeout occurs, then again you send the packet you retransmit the packet through this udt send mechanism. You retransmit the packet and start the timer again. And then once you are receiving the acknowledgement, then you move to this state that wait for call one because you have received that acknowledgement. Now you want to send the next frame that is frame 1.

So, in that case you have receive the packet from the upper layer and once you have receive that packet and that packet is not a corrupted packet and it is the acknowledgement corresponds to frame 0. So, you are moving to this state. Once you are in this state in that case, you are waiting for receiving the packet from the upper layer.

So, once you have received the packet from the upper layer, then you make this send call with the data with this frame 0 and the same process gets repeated that you append this sequence number; sequence number 1 along with the data and the corresponding checksum and then send the packet through the unreliable channel. And you move for waiting for that corresponding acknowledgement. And the same process gets repeated that if you are receiving the data and the packet is corrupted, you are in this loop. If a timeout occurs, then you transmit the packet again and start the timer again and then whenever you are receiving the acknowledgement; that means, you are receiving a non corrupted packet and you have received the acknowledgement corresponds to frame 1; then you stop the timer and wait for this frame 0.

So, that way you send this frames one after another frame 0's and frame 1's, one after another and you move through this state transition diagram to send the packets one after another. And ensure proper flow control along with the reliability in the system.

(Refer Slide Time: 25:19)



Well so, let us look into the problems which are associated with stop and wait type of flow control or ARQ algorithm. In stop and wait ARQ, first of all every packet needs to wait for the acknowledgement of the previous packet. So, until you are receiving the acknowledgement for the previous packet, you will not be able to send the next packet.

Now, for if you think about bidirectional connections; for bidirectional connection, you can use two instances of stop and wait. One instances of stop and wait will ensure one instances of stop and wait will ensure transferring of data from A to B and another instances of stop and wait will ensure transferring data from B to A, but this will again result in a significant ways of network resources that for both the side you have to wait for the acknowledgement.

So, one possible solution to solve this particular problem is that you can piggyback data and acknowledgement from both the direction. But even if you are piggybacking data and acknowledgement for the both the direction, so piggybacking here means that whenever you are sending a data frame along with the data frame. You also adds up the acknowledgement.

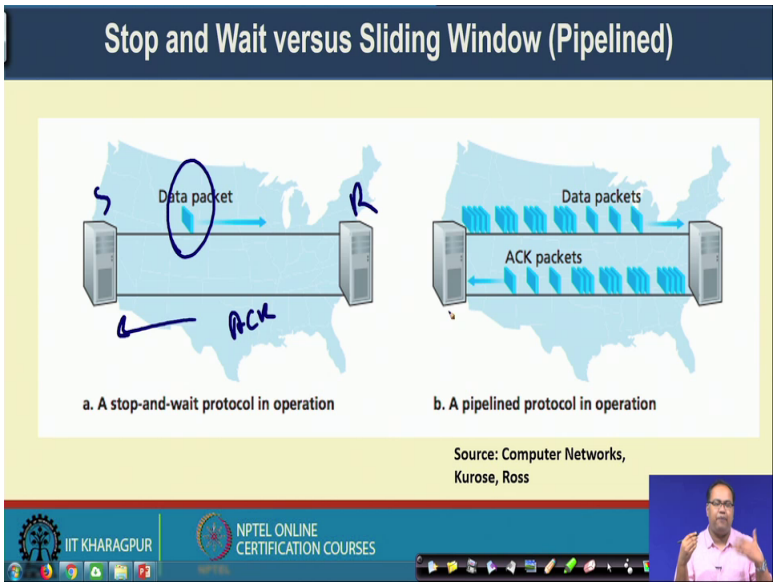
So, this data is say this is going for sequence one and along with the data with sequence one you send the acknowledgement for the previous one which is coming from B to A. So, it is like that say this is B and this is A. From B to A you are sending one data packet; for B to A whenever you are sending a data packet say A has earlier send a packet to B.

You are sending the acknowledgement along with whenever you are sending the data packet to A.

So, although this piggybacking mechanism is more efficient compare to compare to this using these two instances of stop and wait, but still it is wasting a huge amount of resource because for all packets you need to wait for the acknowledgement.

So, you will not be able to parallelly transmit the packets in the network. So, to solve this problem we use a class of flow control algorithms which we call as the sliding window protocol. So, sliding window protocols are a pipe lined protocol where you can send multiple frames or multiple packets altogether without waiting for the corresponding acknowledgement. So, you can send multiple frames and all the frames can go in a pipelined way.

(Refer Slide Time: 27:49)



So, a broad idea of this sliding window protocol is something like this. If you look into the stop and wait protocol, so the stop and wait protocol you are sending only one data packet. So, only one data packet can be outstanding in the network.

So, once this data packet is received by this receiver, then this receivers send back the ACK and once you are receiving that ACK at the sender, then only you will be able to send the next data packet. In case of my pipeline protocol or the sliding window

protocol, what we do that we can send a sequence of packets and parallelly we can receive the sequence of acknowledgement.

So, that way we will be able to use this pipeline concept over the network where you could be able to send a sequence of packets all together and parallelly you can receive a sequence of acknowledgement. So, that way, you will be able to utilize the network resource more efficiently because nowadays. Now with this particular approach of sliding window protocol; you will be ensuring that more number of packets can be push to the network if the network has that much of capacity. And parallelly we will be able to receive the acknowledgement and you will be able to adjust your transmission rate accordingly so that you can receive the packets correctly at the other end.

So, this is the broad idea of the sliding window protocol. And in the next class we look into this sliding window protocols in details.

Thank you all for attending the class.