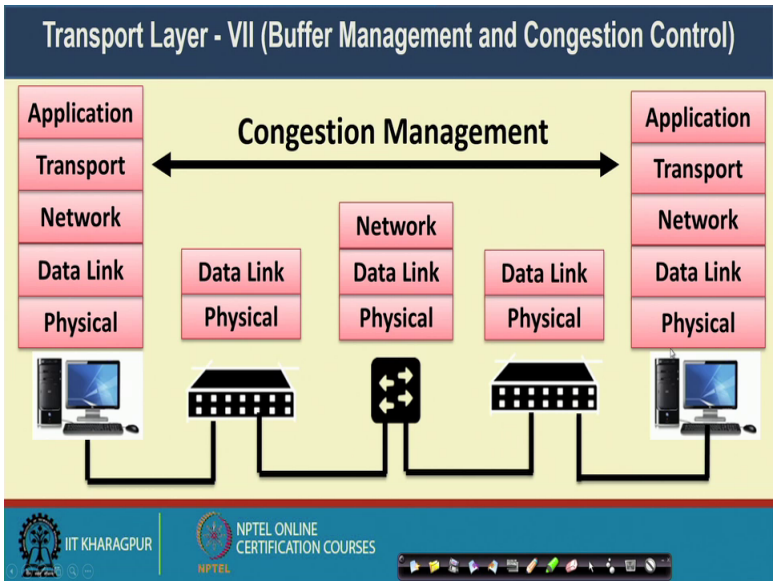
Computer Networks and Internet Protocol
Prof. Sandip Chakraborty
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 17
Buffer Management and Congestion Control

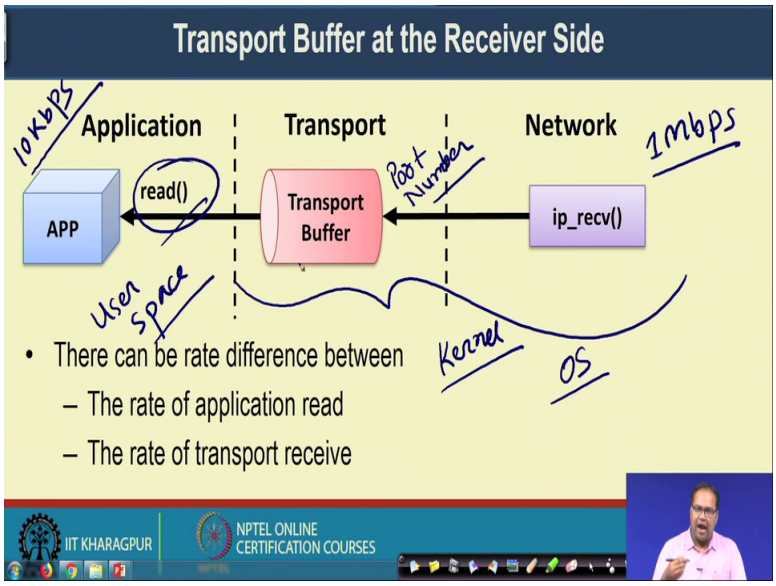
Welcome back to the course on Computer Networks and Internet Protocols. So, we are discussing about the transport layer and various services under the transport layer.

(Refer Slide Time: 00:29)



So, in the last class we have discussed about the basic performance modules in the transport layer along with a hypothetical transport layer protocol that how will you interface the transport layer with the application layer . So, continuing from that point, in this lecture, we will discuss about the buffer management mechanism in the transport layer and then we will look into the details of the congestion control algorithms.

(Refer Slide Time: 00:54)



So, coming to the buffer management; so, as you have looked into the last class that at the sender side as well as at the receiver side, we maintain a transport layer buffer which is a software buffer maintain as a queue and in that queue, at the sender side, whenever you are sending some data, you put the data in that queue and then based on your net control algorithm, the transport layer will fetch the data from the queue and send it to the unreliable ip layer of the network layer.

On the other hand that the receiver side, you have a queue where the data from the network layer is put into the queue and then the application will fetch the data from that queue. Now let us look into this diagram. So, here at the network layer, this ip receive is again a hypothetical function that receive the data from the ip layer on the network layer, and it put the data into the queue and this queue is the buffer at the transport layer corresponds to an application, and we use port number to identify that in which buffer you need to put the data.

So, once this network layer put the data at the transport layer, then the application; from the application, you have to use the read system call or there are other system calls like the write system, the write system call to which you will receive the data from the transport layer. So, this read system call which is the part of the socket programming that we will discuss later on in details.

So, the read; using the read system call, you will receive the data from the transport layer. So, if you are using this write call at the sender side, then you can use read call at the receiver side at the application to read the data and regarding this transport layer implementation, as we have seen that this entire part of the implementation of the protocol stack that is implemented inside the Kernel, if you consider an a Unix type of operating system or broadly, you can say that this protocol stack is the part of the operating system where as this application is written in the user space.

So, the frequency of the read call that is actually managed by the application which you are writing and the application is using this socket programming that we will discuss later in details s; it will use the socket programming to read the data from this transport layer buffer. Now it may happen that well application is reading the data, data at one speed where as the network, it is receiving the data at another speed.

Many of the times, it may happen that will the network is sending the data at a more higher speed compare to what the application is reading see may be the network is receiving data at a rate of some 1 Mbps and the application is reading the data at the rate 10 Kbps.

So, the application is reading the data at a rate of 10 Kbps means at that range, you are executing the read system call. So, you are may be coming at every 1 second and you are making a read system call with the buffer size of 1 Kb 10 Kb. So, you will receive the data at a rate of 10 kbps because of this difference, we face certain problems because the data that will get buffered inside this transport buffer.

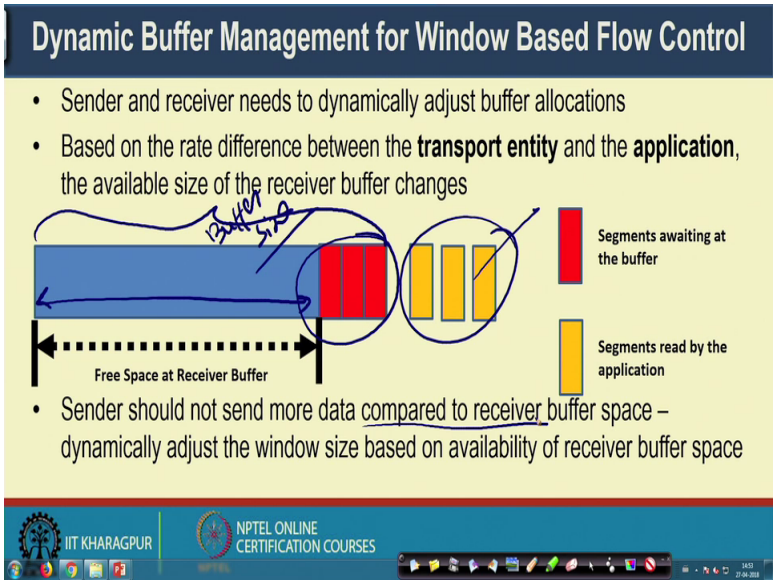
So, because you are receiving data at a higher rate so that particular data will get buffer inside the transport layer, buffer inside this particular queue and after sometime, it may happen that the queue becomes full if, then queue becomes full that will be a problem because in that case, if the senders sends more data to the network, then that particular data will come here at the transport layer, but because this buffer has become full you will experience a packet drop from this particular layer.

Now, you want to avoid that. Now to avoid that; what you have to do? You have to tune the flow control algorithm. So, how will you tune the flow control algorithm? You have to something called a dynamic buffer management where this receivers buffer side, it is changing dynamically, it is changing dynamically because of this rate difference between

the application layer and the transport layer at rate at the rate at which it is receiving the data from the network layer because of this rate difference you may face a problem you may have dynamically changing buffer size.

Now, to handle that; what you have to do like you have to send that information to the sender side so that sender can stop sending data, unless there is sufficient space in the receiver buffer. So, this particular concept, we call as the dynamic buffer management at the receiver side. So, let us look in to this concept of dynamic buffer management in little details.

(Refer Slide Time: 05:53)



So, in case of dynamic buffer for window based flow control for sliding window based flow control, what you have to do that the sender and the receiver needs to dynamically adjust their buffer allocation.

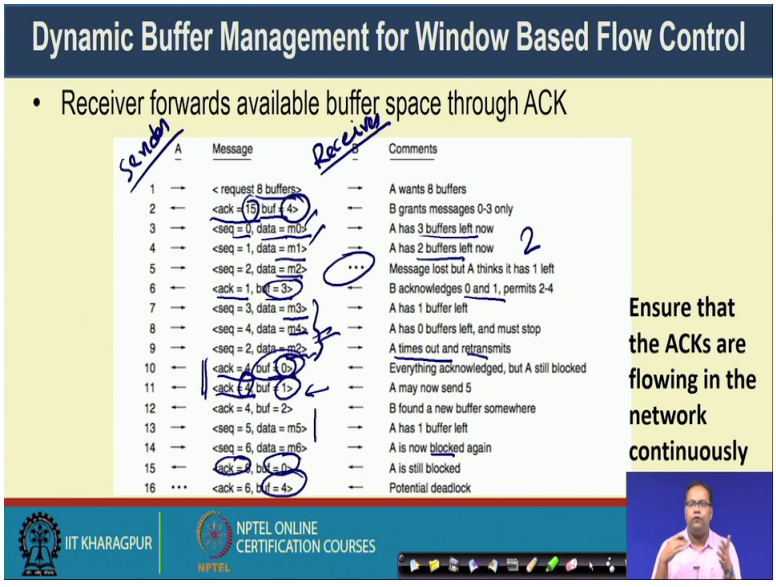
So, that is based on the rate difference between the transport entity and the application, the available size of the receiver buffer may change; so, in this particular diagram. So, these are the set of segments that the application has already read out. So, that has went from the application buffer. Now this is your complete buffer size well. So, this is your complete buffer size. Now out of that there are 3 segments, which are already they are inside the buffer.

. So, these segments are waiting inside the buffer for the application to read them. Now here the free space that you have in the receiver buffer that is this amount. So, you need to advertise this amount to the sender. So, that the sender does not send more data compare to the what the receiver buffer can hold. So, the sender should not send more data compared to the receiver buffer space. So, you need to dynamically adjust the window size; the sender window size based on the availability of the receiver buffer space.

So, what we have looked into the window based flow control algorithm that you can dynamically tune the sender window size and the sender window size basically depicts that how much data you can send to the receiver without waiting for the acknowledgement. Now if you send the feedback from the receiver side to the sender that when the receiver has the, this much of buffer space available, then the sender can set its window size to maximum that value. So, that it will never send data to the receiver more than that particular size.

Now, once the receiver will receive that data, after receiving the data, the receiver can again send an acknowledgement. Once this data has been read by the application and when it is sending that the acknowledgement with the acknowledgement, it can announce the available buffer size. So, let us look how this particular procedure works.

(Refer Slide Time: 07:55)



So, here is an example. There are 2 nodes A and B. They are communicating with each other.

So, the fast message is that a request for the 8 buffers. So, here we are representing buffer at the number of segments that you want to transfer and we are assuming that every segment is a fixe size although that does not hold to for TCP, but here this is just for simplification, we are making an assumption that every segment has of same size and the receiver at the sender; sender A.

So, A is my sender who will send the data and B is my receiver who will receive the data. So, the sender first request for 8 buffer. So, A wants 8 buffer space from B, but B finds that well only 4 buffer space are available buffer space for 4 segments are available.

So, A sends back an acknowledgement with an acknowledgement number along with this buffer space value. So, the buffer space value is mentioned as 4 that determines that A will only grant 3-4 messages from message 0 to message 4 or for segment 0 to segment 4. Now A sends 1 message; so, ones A sends this message, this data with sequence number 0. Now at this time, A has sent 1. So, A has 3 buffers left, then A sends another message A m 1. Now A has 2 buffers left, then A has sent another message and assume that this message has lost.

Now, this message has lost . So, although at the receiver side, you have 2 buffer space left, but A thinks that there are only one buffer space left because A has already sent 3 segments. So, at this stage B acknowledges 0 and 1. So, once this acknowledge is 0 and 1; A send B sends an acknowledgement 1. So, here this acknowledgement is a humility acknowledgement. So, once you are sending back acknowledgement 1; that means, you are acknowledge acknowledging both message 0 and message 1 along with that it is advertising that the buffer space is 3.

So, A get an update that well this message 0 and message 1 has been successfully received by the receiver and it has a available buffer space of 3. So, A again sends message m 3 because it has sent message m 2 already, it has sent message m 2 already. So, it does not know that whether the message has been received or not then it again sends m 4 and finally, sends m 2.

So, after sending this 3 it the advertised buffer space was 3. So, it has sent 3 message. So, once it is it has sent 3 message, during that time, A cannot send anymore data because a sending window was set to 3. So, it has already transmitted 3 3 messages. Now at this stage, this B; it sends an acknowledgement saying that acknowledgement number equal to 4. So, when this acknowledgement number is equal to 4 at this stage A finds out that well, all the 4 messages starting from m 1, starting from m 2, m 3 and m 4, they got acknowledged because 4 is again a humility back acknowledgement.

So, at this stage, there was a timeout for there was a timeout for message m 2 for which it has not received the acknowledgement and it transmits that message again. So, B has received m 2, m 3 and m 4. So, B has sent an acknowledgement 4 with buffer space 4. So, with this buffer space 0, what A is acknowledging? The A is acknowledging that that this particular message all the message has been received by B, but it does not have any buffer space available; that means, the application has not read that data. Now once the application has read that data, A sends another acknowledgement saying that the acknowledgement number the same acknowledgement number 4, but announcing the buffer space as 1.

So, at this stage, A makes one buffer space available, B makes 1 buffer space available to A saying that it can send one more message; one more segment. So, here you can see that once you are advertising buffer space 0, after that; once that buffer space becomes available, you need to send another acknowledgement, otherwise, the sender will get blocked at this stage because the sender; once it gets that the buffer space is 0, it will not send anymore data.

So, it is it will get blocked here. So, that way the things get continues continued. So, here in this case, A A sends the data and gets block and then it gets an acknowledgement number with buffer space 0, here A is still blocked, then A A can send get another message with the available buffer space.

So, here you can see that well, it may it may sometime happen that that because of you are sending this advertisement that that you have do not you do not have any sufficient buffer space, there is a possible dead lock at the sender side because the sender can find out or sender can thinks out that no more space is available at the receiver side.

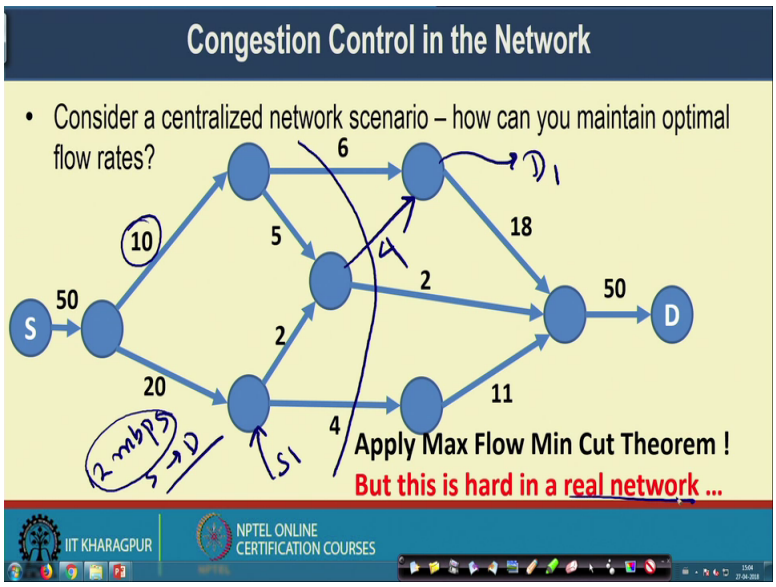
Now, to avoid this particular thing; what you have to ensure? You have to ensure that the acknowledgements are flowing in the network continuously. So, in this particular example, if it happens that well initially, you have advertised that the buffer space is 0, then B sends another acknowledgement saying that the buffer space is available, but somehow this acknowledgement got lost.

So, because this acknowledgement got lost, the system can lead to a dead lock unless B sends another acknowledgement of that read gets a timeout. So, it need to explicitly tell to A that now sufficient buffer space is available. So, a will be able to send more data. So, in that particular case, you have to ensure that after every timeout, B should if B is not receiving any more data from A and the connection is still open.

So, B should send the duplicate acknowledgement announcing that it has sufficient buffer space to receive further data, otherwise, there is a possibility of having a deadlock in case the acknowledgement get lost.

Well so, this is the concept of dynamic buffer management at the transport layer.

(Refer Slide Time: 14:41)



Now, we will see another important aspect of the transport layer which we call as the congestion control. So, what is that congestion control? So, you just initially think of a centralized network scenario. So, each node has an edge. There is an edge between 2 nodes and we have an edge wait.

So, this edge wait signifies that what is the capacity of that particular link, say if you want send some data S to D, at that case, if you want to find out that what would be the capacity of that flow what would be the maximum capacity of that flow.

So, you can apply this max flow min cut theorem which is being covered in the algorithm course. So, you can apply the max flow in cut theorem and from the max flow min cut theorem you can find out; what is the minimum cut here. So, just by looking into the scenario, this seems to be the minimum cut because this is the minimum cut. . So, you can send a maximum flow at the rate of 6 plus 4 plus 10 plus 2; 12.

So, you can send a data at the rate of if you send a think as the unit as Mbps. So, you can send the data at a rate of 12 mbps from S to D. Now if you have this kind of centralized scenario, you can apply this kind of algorithm, this kind of mechanism to restrict your flow rate to 12 Mbps, but if it is not there, if it is not there, then how will you be able to find it?

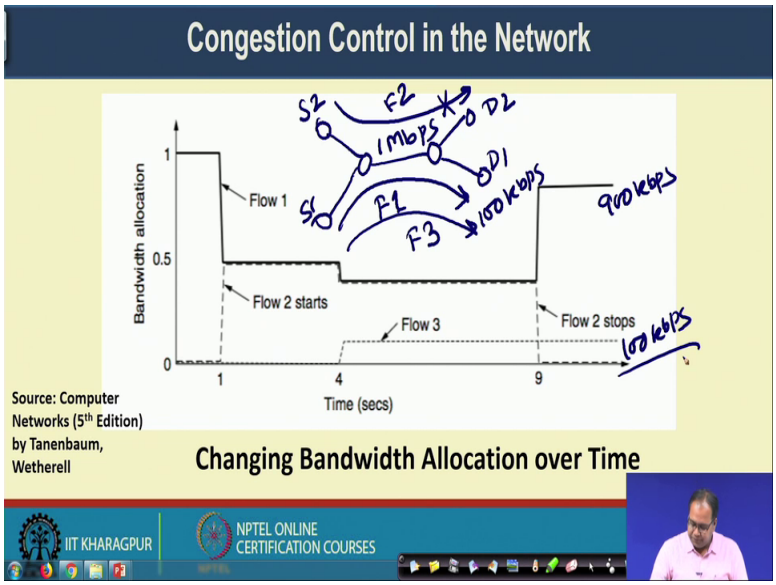
Now, your flow control algorithm will not be able to guaranty that your transmission is always restricted to 12 mbps because you are getting the rate from multiple paths and the thing is restricted to this maximum segment size that is an approximate calculation that we have looked earlier . So, because of that it may happen that in a distributed scenario; the sender may push more data compare to this 12 Mbps button click capacity which is there in this particular network. So, this capacity is the button click capacity. So, if you want to send some data from S to D even if there are no other flows in the network, you will never be able to achieve more than 12 mbps.

Now, the scenario get more complicated if you have multiple flows, if you think of that there is another flow from this S 1 to D 1, that will also use these links this individual links in the network, you can think of that there is a link from here to here with the capacity of 4. Now it will use this particular link. Now this flow may go through any of this link and there would be this kind of overlapping among multiple end to end flows and they will share the capacity in this bottle neck links.

So, this entire thing is difficult to implement in a real network because in a real network you have to implement it in a distributed way. So, in that particular concept, the congestion may occur in the network where this bottle neck links in the bottleneck link, you are trying push more than the capacity of the particular button linking. Now if you

want to push more data compare to the button click link capacity of the bottleneck, what will happen that the intermediate buffer at the nodes, they will get filled up you will experience packet loss, which will reduce the end to end transmission delay or which would fill increase the end to end transmission delay significantly.

(Refer Slide Time: 18:18)



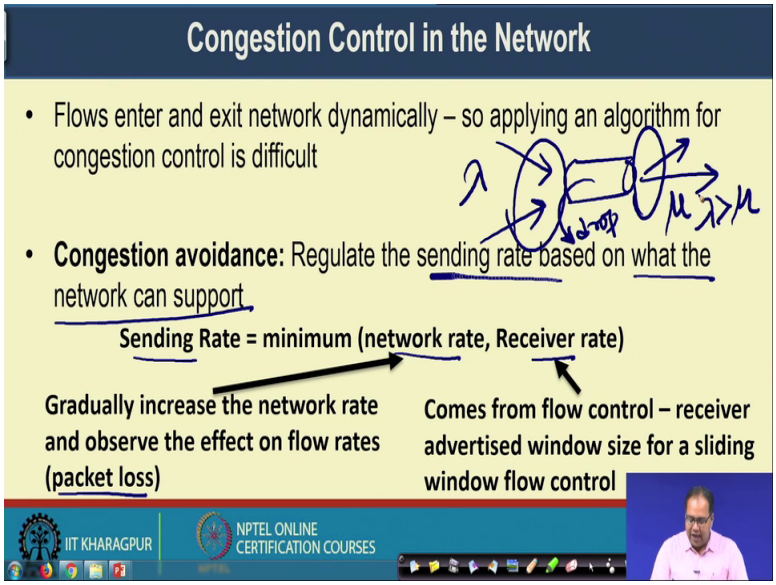
So, from here let us look into that how your bandwidth changes when you when you allocates the more flows between the same source destination pair. So, initially say your bandwidth allocation we are normalizing it in 1 mbps. So, initially if you have a single flow. So, we just think of a network like this. So, you have see this network and you are you are sending a single flow from this source to S 1 to destination D 1 and assume that this bottleneck link capacity is 1 Mbps.

Now, if that is the case, then once you are starting flow 1, then flow 1 will be able to use this entire 1 mbps bandwidth which it is there in this bottleneck link now see after some time at once again you start another flow from says this S 2 to D 2 this is layer says flow 2. Now if you start that then this link capacity is 1 mbps. So, this link is being shared by both F 1 and F 2. So, ideally what should happen that well whenever you are starting this particular flow, it will it will the entire bandwidth the bottom link bandwidth will be divided between F 1 and F 2. So, everyone will get approximately both flow 1 and flow 2 will get approximately 0.5 mbps of speed if their sending rate is more than 0.5 mbps.

So, in that case, it is entire button link capacity is divided between flow 1 and flow 2 and after sometime say you have started another flow, flow 3 which has required which whose required bandwidth is little less see its required bandwidth is something close to see 100 kbps if that is the case, it will drag this hundred kbps bandwidth from here and then the remaining bandwidth will shared between flow 1 and flow 2 and flow 1 and flow 2 are utilizing both this bottleneck bandwidth.

Now, after some time if flow 2 stops, then flow one say flow 2 gets stop flow 2 flow 2 finishes at that time flow one will get the bandwidth which is close to 900 m, 900 kbps and flow 1 is utilizes some 100 kbps that way this entire bandwidth allocation among multiple buffer multiple flows gets changed over time.

(Refer Slide Time: 20:59)



So, in this context, the congestion we discussed the congestion controlled algorithm in the network . So, this congestion control algorithm, it is required because this flows they enter an exit network dynamically, the example that we have seen and because of this reason apply an algorithm for congestion control in the network is difficult because you do not have this centralized network information like the first example that I have shown you where you can apply this min cut theorem to find out the maximum flow between one source and one destination.

The scenario is much more difficult here because every individual router do not have this entire network information, even the end host do not have that entire network

information and a distributed or in a decentralized way you have to allocate the flows among flow rates among different end to end flows. So, we apply something called a congestion avoidance algorithm rate that at an a congestion control because a priori estimation of congestion is difficult. So, rather than going for congestion control we go for congestion avoidance.

So, the congestion avoidance is that whenever there is a congestion, you detect that congestion and then try to come out of that congestion. So, how will you do that? So, you regulate the sending rate based on based on what the network can support. So, your sending rate now is the minimum of something called the network rate and the receiver rate. So, earlier your sending rate was equal to the receiver rate.

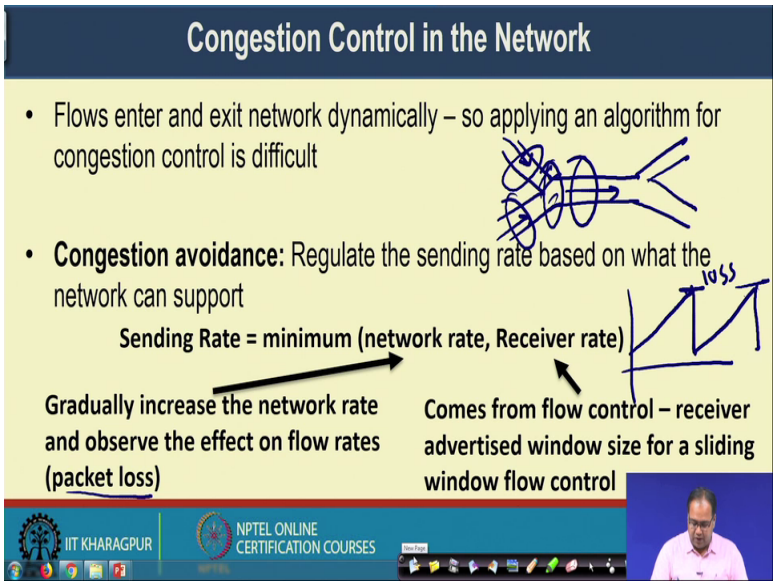
So, based on the buffer advertisement that was given by the receiver you are controlling you window sides and you are sending the data at that particular rate. Now you are sending rate will be the minimum of your network rate, what the network can support and what the receiver can support. So, this network rate receiver rate it comes from the flow control algorithm. So, the it comes from the receiver advertise window size for a sliding window based flow control.

So, the receiver is advertising that particular window information and this network rate you do not have any control over the network rate or rather to say you do not have any estimation mechanism over the network rate. So, what you can do that you can gradually increase this network rate component and observe the effect on the flow rate. So, ideally what can happen in case of wire network, if you assume that the loss due to channel error is very less; that means, if there is a loss from the network that loss is coming due to the buffer overflow at the intermediate routers. Now if buffer overflow happens that gives an indication that, will the total incoming rate to the buffer exceeds the total outgoing rate from that buffer.

So, as an example, if you just think of an intermediate buffer intermediate buffer queue it is receiving data from multiple flows and there is some outgoing rate the outgoing rate can also be multiple. So, assume that total incoming rate is lambda and total outgoing rate is mu. Now if your lambda is more than mu, this indicates that after some time the buffer will get filled up and once the buffer will get filled up there will be packet drop from the buffer and you will experience a loss.

Now, if you observe a packet loss here that indicates that the total incoming rate to the intermediate buffer is exceeding from the total outgoing rate; that means, lambda is more than mu. So, this gives an signature of the congestion; that means, if just think of a rode network see assume that you have rode network something like this.

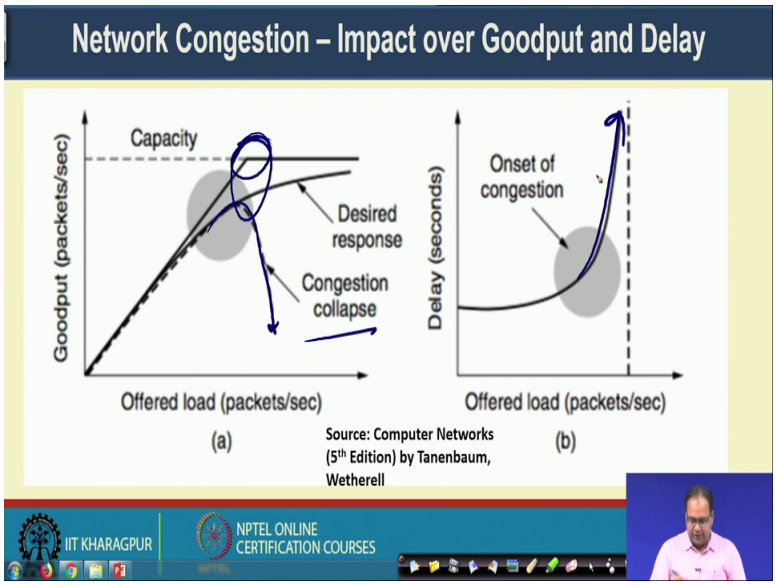
(Refer Slide Time: 24:32)



So, this is an example rode network. So, the cars are coming to this road. So, this is the bottle neck here. So, see here the total capacity that it can support if the total incoming capacity that from this 2 roads are exceeding this capacity. So, you will experience the congestion here. So, the same things happen in case of network and we are sensing a congestion from this packet loss because packet loss gives an indication that the buffer that you have that buffer is getting exceeded.

So, you identify it as a congestion and you again drop the network rate. So, the broad idea is that you gradually increase the network rate at some time, you will experience, the packet loss the moment you are experience the loss, then you drop the rate and again increase the rate and again whenever you get a loss you will drop the rate. So, that way we apply the congestion control algorithm in the network.

(Refer Slide Time: 25:36)

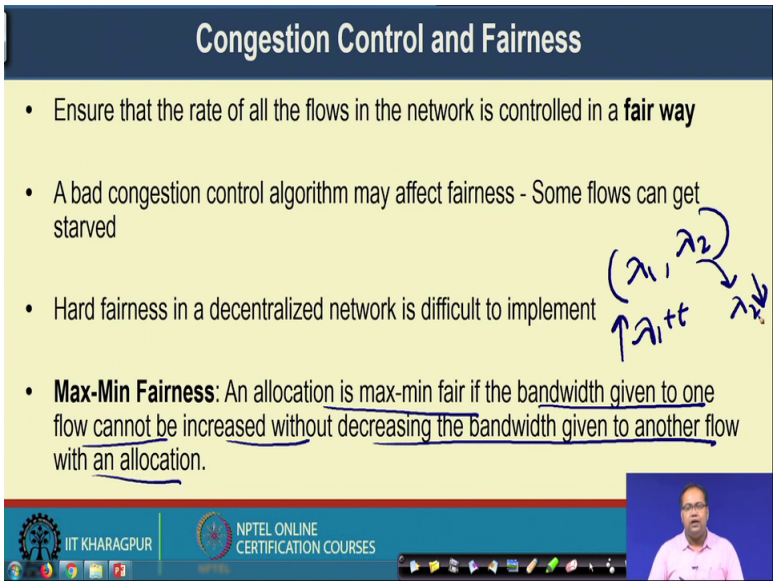


Now, the question comes here that how will you increase the rate. So, we see the first we want to see the impact of network congestion over good put and delay. So, what we see that if you look into the network good put that the number of packets per second that you are receiving at the transport layer and with the offered load. So, you have a maximum capacity. So, normally what happens that will the rate gets increased up to the maximum capacity.

But at the moment here is this congestion you see a certain drop in the good put because your packets are getting dropped and if packets are getting dropped. The flow control algorithm will try to retransmit the packets. So, you will get a certain drop here, we call that the congestion collapse. Now when the congestion collapse happens if the you experience significant increase in the dealing.

Because packets are getting dropped the flow control is trying retransmit, the packets if that that time the link is not able to come out of the congestion again that retransmitted packet will fall in the congestion and there is a high probability that the buffer is still full and the packet may get dropped. So, because of that the total end to end delivery of the successful packet that may get increased.

(Refer Slide Time: 26:49)

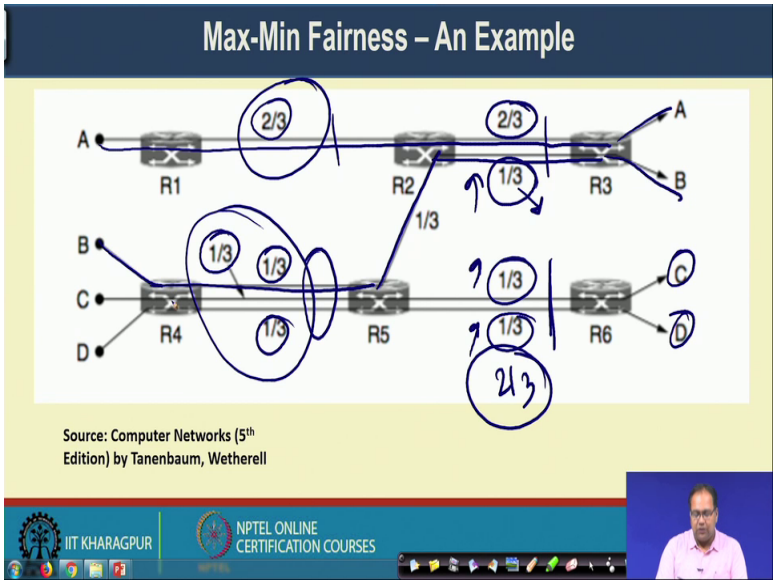


So, to ensure congestion control over the network we need to ensure another thing which we call as the fairness. So, what is fairness the fairness ensures that the rate of all the flows in the network is controlled in a fair way what do what does mean that now bad congestion control algorithm may affect fairness; that means, some flows in the network may get starved. So, because it is flowing n the congestion, you can just think of a scenario in a road network, if you are following in a congestion, if a car is following in a congestion. So, the car can have a very bad lack or have a huge delay to reach at the destination. So, the similar thing may happen in the network that some flows may get starved.

Now in a decentralized network ensuring hard fairness is very difficult because you again you require the entire networking information and want do some mathematical calculation to find out this min cut from that min cut theorem what would be the available capacity and restrict the bandwidth to that particular capacity. So, doing all this calculation of a on a central network is very hard. So, rather than providing this kind of hard fairness what we try to do we try to allocate what is called as a max min fairness.

So, what is a max-min fairness. So, an allocation is max min fair if the bandwidth given to one flow cannot be increased without decreasing the bandwidth given to another flow with an allocation so; that means, in an allocation see you have 2 flows say lambda one lambda 2, we can say this lambda 1 lambda 2 is max min fair if you make lambda increase lambda one to some value epsilon, then you have to decrease lambda 2. So, lambda 2 needs to be decreased. So, you cannot increase the value of lambda one without decreasing the value of lambda 2 this particular allocation, we call it as a max min fair allocation.

(Refer Slide Time: 28:46)



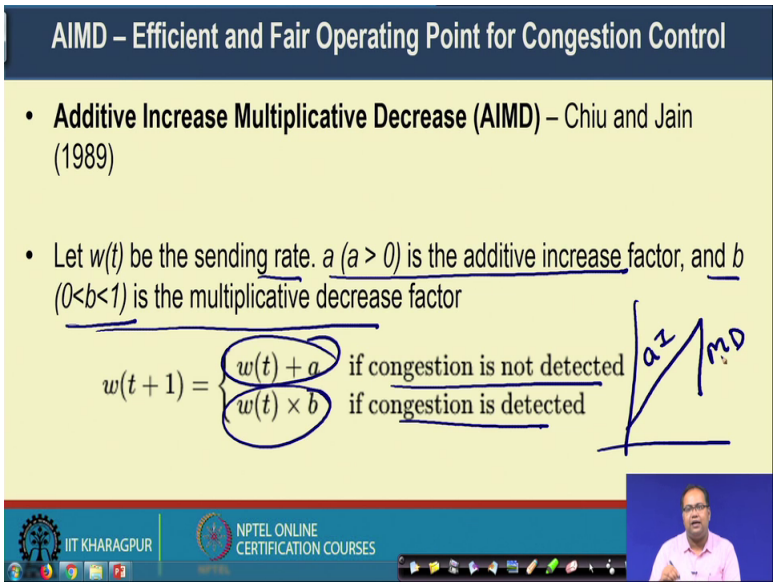
So, let us look into an example of max min fair allocation. So, in this particular example we have multiple flows here. So, you can see that this is the bottleneck capacity where 3 flows are sharing the link. So, 3 flows are sharing the link means each of them will get one third of the bandwidth. So, this particular link is shared by maximum number of flows.

So, this is shared by 3 flows this is shared by 2 flows, this is shared by 2 flows and this is shared by one flows. So, here is the bottleneck. So, each of them will get one third of the bandwidth if they are getting one third of the bandwidth the flow which is this flow number D, it will use one third of the bandwidth this flow number is C and it will use the one third of the bandwidth.

Then flow B which is moving from here because of this bottleneck capacity, it is using one third of bandwidth. So, it will utilize one third of bandwidth in this link. So, in this link the remaining capacity is 2 third. So, this flow A, it can use the 2 third of the bandwidth. So, this is the max min fair algorithm because if you want to increase the bandwidth of this from one third see for flow B, if you want to want to increase it from one third you have to decrease the band width of flow rate.

Similarly, if you want to increase the bandwidth for flow C or flow D because you can see that in this link the total capacity that is being utilized two third it is not utilizing the full capacity here also, it is not utilizing the full capacity and just by taking that if you want to increase the capacity of any of this link because of this bottleneck capacity distribution you have to decrease the capacity of say flow B. So, this particular allocation is a max min fair allocation.

(Refer Slide Time: 30:36)

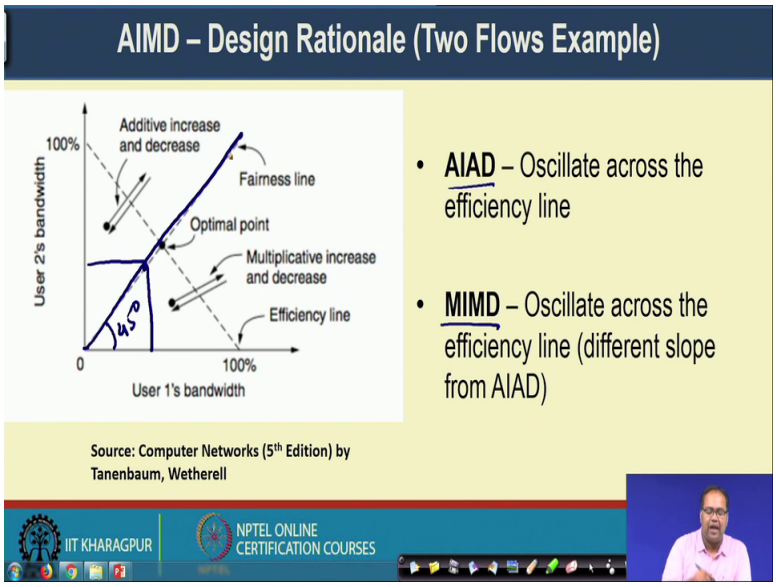


Now, in a distributed way we can we can ensure max min fair allocation by applying this AIMD algorithm. So, we call it the algorithm as additive increase multiplicative decrease which was proposed by Chiu and Jain in 1989. So, the algorithm is something like this. So, you increase the flow rate additively, but drop the flow rate multiplicatively what is mean by that.

So, let w t be the sending rate and a is an additive increase factor and b is the multiplicative decease software. So, whenever you are increasing the rate that mean the congestion is not detected you provide an additive component, but when congestion is decremented you provide the multiplicative components. So, the value of b is from 0 t 1. So, if you are multiplying by b; that means, you are dropping the rate.

So, the example of a additive increase multiplicative decrease is that see you increase it linearly in additive way adding a fixed component whenever you are increasing, but whenever you are dropping you make a significant drop now let us see that how this additive increase multiplicative decrease. So, this part is the additive increase and this part is a multiplicative decrease. So, how these additives increase multiplicative decrease can help in ensuring fairness.

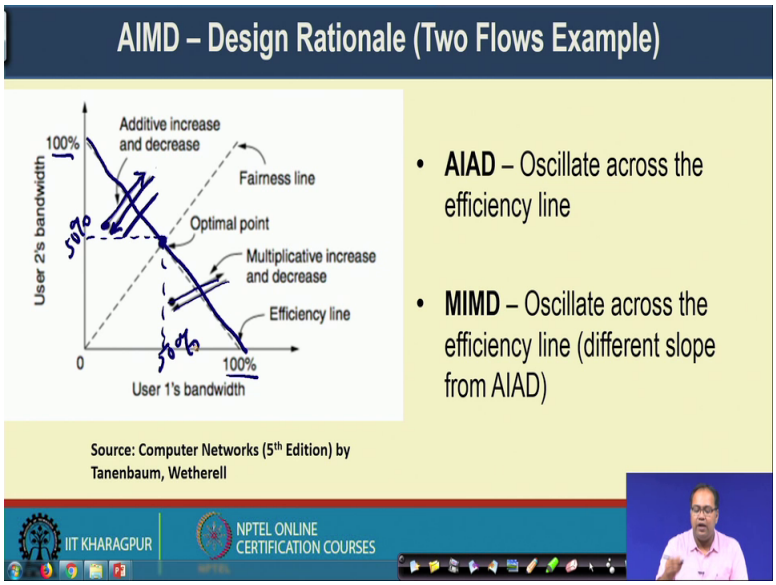
(Refer Slide Time: 31:54)



Now, let us look into another variant which we call as the AIAD or MIMD. So, AIAD is the additive increase additive decrease you increase additively as well as decrease additively MIMD is you increase multiplicatively as well as you decrease multiplicatively. So, we are taking an example of 2 users who are sharing the bottleneck link.

Now, if 2 users are sharing the bottleneck link. So, this line gives you the fairness because this line with the 45 degree slope, its assumed that will if you take one point here that both user a and user b gets almost equal amount of bandwidth or in hard fairness, they get equal amount of bandwidth. So, this line gives me the fairness line.

(Refer Slide Time: 32:38)

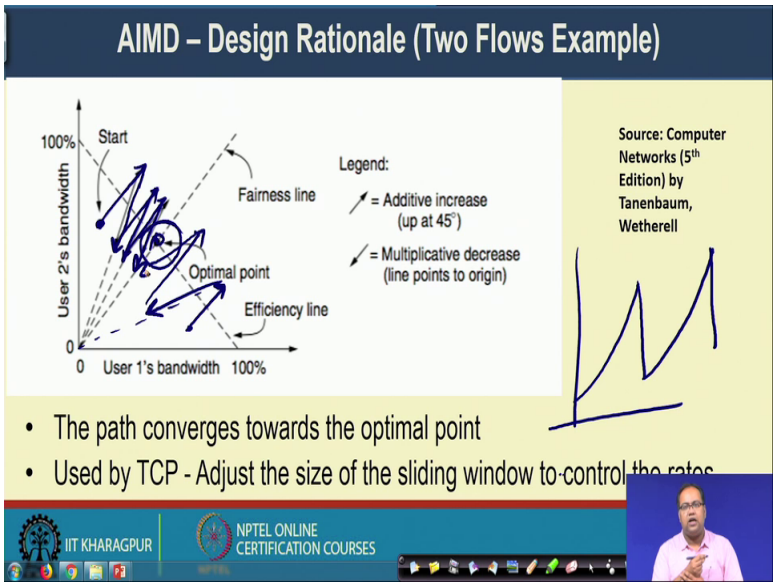


Similarly, this line gives you the efficiency line because here the total allocation has reached to 100 percent. Now if you start from this point and with this additive increase if you increase the bandwidth once you exceed this line; that means, you are putting more data than the capacity of the link this hundred percent bandwidth link. So, you will experience a packet loss; that means the motion of the congestion.

Now at this case if you want to decrease the bandwidth in a additive way you just decrease it parallelly. So, you can see that if you are doing a additive increase and additive decrease you will just osculate around this efficiency line, but that is not the optimal point we want something here this is the optimal point where both the flows will get 50 percent of the 50 percent of the available bandwidth and together they will on the efficiency line that measure the 100 percent.

So, we want to reach at this point. So, in additive increase additive decrease, you will just osculate on the efficiency line. Similarly on the multiplicative increase and the multiplicative decrease, you will osculate on the this efficiency line because you will increase in the same rate and drop in the same rate. But if you are going for additive increase multiplicative decrease the scenario becomes something like this.

(Refer Slide Time: 34:04)



So, the scenario become something like this. So, you are you are starting from this point you are making a additive increase, then you make a multiplicative decrease then again you make a additive increase you make a multiplicative decrease you make a additive increase and the multiplicative decrease. So, gradually you will move towards the optimal point.

So, if you are making as an additive increase followed by a multiplicative decrease gain a additive increase followed by a multiplicative decrease and both the users are following this principle, gradually, they will come to the optimal point similarly if you start from here you make a additive increase and then towards this center point make a multiplicative decrease again make a additive increase make a multiplicative decrease gradually you will move towards this optimal point.

So, that way this AIMD algorithm additive increase up at 45 percent and multiplicative decrease towards the line points to origin, if you apply this particular algorithm, you can converge towards a optimal point. So, this particular algorithm is used by TCP to adjust the size of the sliding window to control the rates that will look into the details sometime later well . So, in this particular lecture you got the broad overview about the congestion control algorithm and with this lecture, we have covered basic services which are being offered at the transport layer.

So, in the next set of lectures, we will look into more details of the transport layer from the protocol perspective, and also we look into that how this difference service can be combined together to build up the final servicing so.

Thank you all for attending, hope we will meet in the next class.