Broadband Networks

Prof. Karandikar

Electrical Engineering Department

Indian Institute of Technology, Bombay

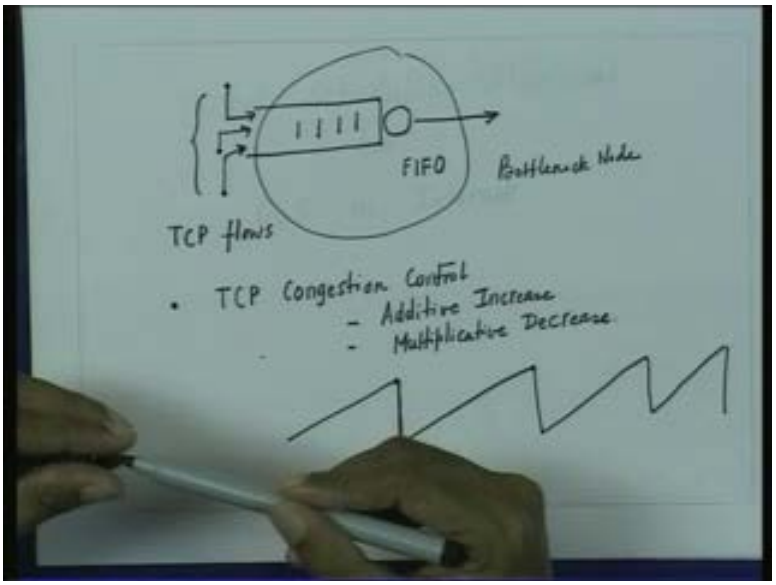
LECTURE – 9

Packet Scheduling Algorithm Introduction

So, we will today discuss about packet scheduling algorithms for providing quality of service guarantees in internet. Till today, we have been considering that various flows when they share the buffer at a particular router or a statistical multiplexer; then there is one common buffer where all the flows are sharing this common buffer and there may be a packet scheduler which schedules these packets in the first in first out manners.

So, let me just see what is the scenario that we have been considering till now. Scenario that we have been considering is that there is a one single buffer and all the internet flows are sharing this buffer and this buffer may be scheduling the packets may be in a first in first out manner.

(Refer Slide Time: 1:20)



Now, for the time being, let us consider that we are not concern about providing specific quality of service guarantees to the flows. But let us say that at least we want that these various flows when they share one common output link; then they get what we call as a fair share of the bandwidth. We will shortly define what is meant by a fair share of the bandwidth and so on. But let us consider that these flows which are sharing this single common buffer are the TCP flows.

So basically, what we are saying is that all of these are let us say are TCP flows. Now, these TCP flows are sharing one common buffer and the scheduler is scheduling the packets in the first in first out manner. So, we would like to know that what are the problems with these scheduling disciplines. So, towards that we should know that the TCP has a specific congestion control techniques. The TCP's congestion control technique is based on what is called as additive increase and multiplicative decrease, additive increase and multiplicative decrease. What does it mean?

It means that the TCP increases its window size in an additive fashion after the visit of an acknowledgement. So, when the TCP sends packets worth its window size and when as if when the TCP the keeps on receiving the acknowledgement, it slides the windows and it increases the window size in an additive fashion.

When the TCP's congestion control mechanism detects the packet loss, then the window drops to half of its previous size or it drops to 1 depending upon various versions of the TCPs that are being used and the various types of congestion control algorithms that are being deployed. But the basic principle is that the increase is in an additive fashion and the decrease is in a multiplicative fashion.

Typically, if you see the window evaluation, then this window evaluation of the TCP will follow a saw tooth pattern. That is it increases in an additive fashion and whenever it detects the packet loss, the window size drops to 1 or may be the window size may be dropping to half depending upon the versions that are used.

Now, note that for the acknowledgement, it takes about 1 round trip time. The packet needs to go from the center to the receiver and then the acknowledgements need to travel from the receiver to the center. Therefore, this increase of the window size or this decrease of the window size, it can happen at an interval of the round trip times.

Now therefore, what happens is that when this multiple TCP flows like this or sharing a single common buffer, it might be quite possible that these sources of these TCP flows have varying round trip times between their destinations. So, these TCP flows are having varying round trip times and let us assume that this node is a congested or what we call as a bottleneck node or a congested node.

So, as a result when these TCP flows are having varying round trip times, then this congestion control mechanisms will be biased towards what it is you know those connections which are having shorter round trip times. The TCP's congestion control mechanism of additive increase and multiplicative decrease is biased towards those flows which have shorter round trip times. Why? Because, if the flow has a shorter round trip times, it rams up to its available capacity very fast and when it detects the packet loss, when the bottleneck node is congested, it drops its window size to 1. But then, it rams up again very fast.

On the other hand, the flows which are having larger round trip times, they respond to the congestion conditions and then ramp up their window size proportionate to its round trip times. If it is large, so the increase is little slow. So therefore, the AIMT method is biased towards connections with the shorter round trip times.

As a result, what happens? When these TCP flows are sharing a single buffer at a node which is scheduling the packets as the first in first out manner; then obviously, the flows with shorter round trip times are likely to have an unfair share of the bottleneck bandwidths. So, this is one aspect of when the various flows are sharing common buffer, then we would like to know whether these, each of these flows are getting a fair share of the output link bandwidth.

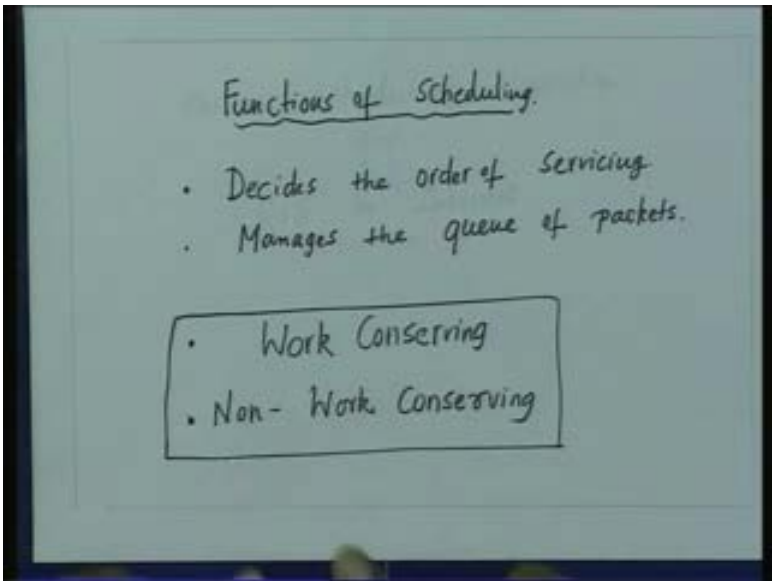
Even if these are not TCP flows, even if these are not TCP flows, some of these flows could be the non TCP flows also and suddenly they may start transmitting a large number of packets. Now, if they transmit the large number of packets and congest the link, then the packets will get dropped and the TCP in response to this packet drop will drop its window size. But on the other hand, the non responsive flows may not drop the packets and keep transmitting the packets. As a result hogging the bandwidth and getting an unfair share of the link bandwidth.

So, the question therefore is that we need to segregate these flows into various queues and then schedule the packets in such a manner that each of these flows gets a fair share of the link bandwidth. Essentially, it means that we need to go to some kind of a non first in first order, non first come first serve kind of a scheduling technique. So, for both you know for protecting the flows from misbehaving users or non responsive users as well as for the user giving a fair share of the link bandwidth; we need some kind of a fair scheduling disciplines and then we will also see how we can provide the specific quality of service guarantees like the delay or the delay jitters to these various flows.

So, today we will see that how we can design scheduling disciplines that meets our objective of providing fairness to various flows which are sharing our outer link as well as protecting this links from the misbehaving or non responsive flows or even non responsive protocols like the UDP or the constant bit rate flows or the CBR traffic, let us say in an ATM switch.

So, we will discuss today about those scheduling disciplines. Now, let me just discuss about this various scheduling disciplines. What are the functions of the scheduling algorithms?

(Refer Slide Time: 9:30)



So, functions of scheduling - what we mean by the scheduling algorithms? Now, it has 2 orthogonal components. One you know, it decides the order of service and second it manages the queue of packets. So, there are 2 orthogonal components of a scheduling algorithm. You decide the order of servicing the packet that in which order you want to service the packet and secondly, you will manage the queues of various packets.

Now, various objectives the scheduling algorithm need to satisfy: one - as we have just spoken is we would like to schedule the packets in such a manner that each of these flows gets a fair share of the link bandwidth as well as they are protected from the misbehaving users. Secondly, we would also like to have some kind of a network level quality of service guarantees in terms of the delay bounds, in terms of the bandwidth, in terms of the delay jitters. So, these kinds of quality of service guarantees, we would like to give through the scheduling algorithms and secondly, we would like to have some kind of hierarchical link sharing.

By hierarchical link sharing, we mean that there may be a link which may be shared by let us say, multiple departments of an organization and in that department… So, basically that output link bandwidth which you may have to a service provider that is being shared between multiple departments in a fair manner and then within each departments, there may be multiple applications where this link bandwidth may be shared. So, as a result we are sharing the link bandwidth in a hierarchical fashion through the scheduling disciplines. So, these are the 3 objective that a scheduling discipline has to meet.

Then the scheduling algorithms can also be a work conserving scheduling algorithms or it can be a non work conserving scheduling algorithms. So, the scheduling algorithms can be work conserving and non work conserving. What is a work conserving scheduling algorithm? In a work conserving scheduling algorithm, a server is never ideal if there are packets in the queue. It always serves the packets if there are non empty queues. On the other hand, in a non work conserving scheduling algorithm, even if there are packets in the queue, a server may be ideal.

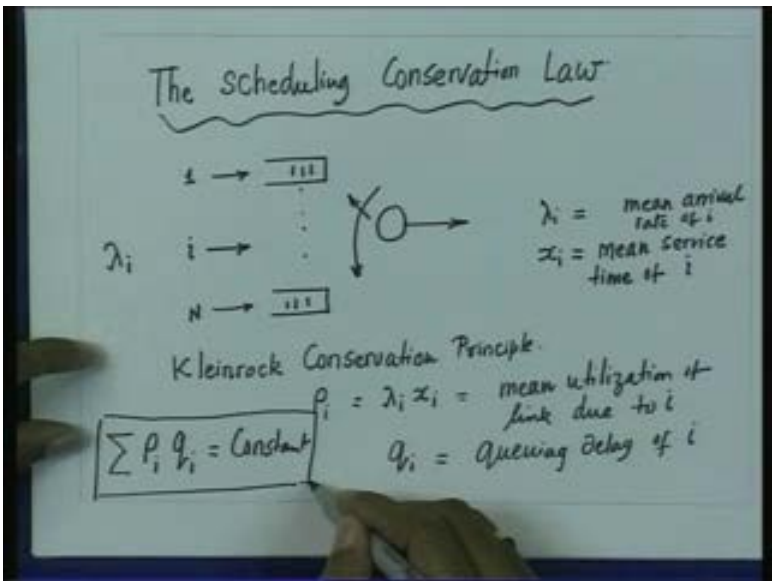
The question of course is that why one would like to have a non work conserving scheduling algorithm? Because, a non work conserving scheduling algorithm; it seems will wait even if there are packets in the queue and therefore wasting the link bandwidth. So, why we would like to have such a scheduling algorithms? We will see later on in the discussions in the on the work, non work conserving scheduling algorithms that sometimes it may be desirable to use non work conserving scheduling algorithms to regulate, let us say the burstiness of the output streams.

You may like to regulate the burstiness of the output stream; otherwise if we schedule these packets, then the downstream nodes may receive busty inputs. So, we would like to regulate this. So, to achieve this objective, we might want to have a non work conserving scheduling algorithms. Non work conserving scheduling algorithms can also be used to control the delay jitter. To achieve these kinds of objectives, sometimes, it may be necessary for the server to remain ideal even if there are packets in the queues and that is why we call such scheduling algorithm as non work conserving.

For the time being, for most of these discussions in this lecture and the subsequent lectures, we will concentrate mostly on the work conserving scheduling disciplines. However, I will take few examples of non work conserving disciplines also and will try to illustrate where they can be used in the actual routers.

So now, let us see the work conserving scheduling algorithms only for the time being. Now, in the work conserving scheduling algorithm, interestingly, there is this scheduling conservation laws.

(Refer Slide Time: 15:07)



So, that is what I would like to illustrate which we call it the scheduling kleinrocks. It is a Kleinrocks scheduling conservation law.

Now, first thing we should know that the first come first serve scheduling algorithms, where we were considering earlier that there is one single buffer and multiple flows are sharing this buffer and the packet are being served in the first in first out manner. Such first come first serve scheduling algorithms cannot differentiate between different connections. So, if we have to give fairness, if you have to give protections, if you have to give quality of service guarantees; then we must resort to non first come first served scheduling algorithms.

So, by the non first come first serve scheduling algorithms we mean that we have different queues for different flows. So, let say 1 cube n flows are sharing these different flows and then there is a scheduler which is trying to schedule the various packets out of these queues and may try to give certain quality of service guaranties. However, if this scheduling algorithm if this scheduler is a work conserving scheduler, then we have a very important kleinrocks conservation law.

So, what is that kleinrocks conservation law? Now, this sates let us say rho $_i$ is equal to lambda $_i$ x $_i$. Now, let us say lambda $_i$ is the arrival rate for the i^{th} connections. So, lambda $_i$ is the arrival rate for the i^{th} connections and let us say that x $_i$ is the mean service time; lambda $_i$ is the mean arrival rate and x $_i$ is the mean service time of i^{th} connections and lambda $_i$ is the is the mean arrival rate or the average arrival rate of i. So, lambda $_i$ x $_i$ gives you the rho $_i$ and the rho $_i$, we call it to be the mean utilization of the link due to i.

Now, let us say that q $_i$ is the queuing time, average queuing delay of i^{th} connections. Then, the kleinrocks conservation law states that the sum of rho $_i$ q $_i$ is going to be a constant. So, what does it say that suppose we have a non first come first serve scheduler were there are multiple connections; each of the connections, let us say is having a queue and these queues are being served by a work conserving scheduler. Now, these queues could be virtual or physical in a practical system. We are not right now bothered, but let us say that there are n such queues.

Let us say that in a i^{th} queue, lambda i is the average arrival rate and x $_i$ is the average service time; then lambda $_i$ into x $_i$ denotes the mean utilizations of the link to i^{th} connections. Now, what the kleinrocks conservation law is stating is the sum of the mean utilization due to the i^{th} connections multiplied by its queuing time and the sum over all such connections is going to remain constant. What is the significance of this?

The significance of this is that if you want to reduce the queuing time or queuing delay of some connection, let us say that of the i^{th} connection, you want to reduce its queuing delay; that is q $_i$ you want to reduce. So here, this q $_i$ you want to reduce and if you want to reduce this q $_i$, it can only happen at the expense of an increase of the delay of the other connections. Because, the sum of mean utilization multiplied by the queuing time is going to remain constant over sum taken over all connections.

We can give lower delays or lower bandwidths to other connections only at the expense of an increase in the delay of other connections. So, we can give lower delays to the other connections at the expenses of an increasing delay of the other connections. Essentially, what does it mean is that utilization times delay, this gets conserved. So, you can only play between the delays of the different packets.

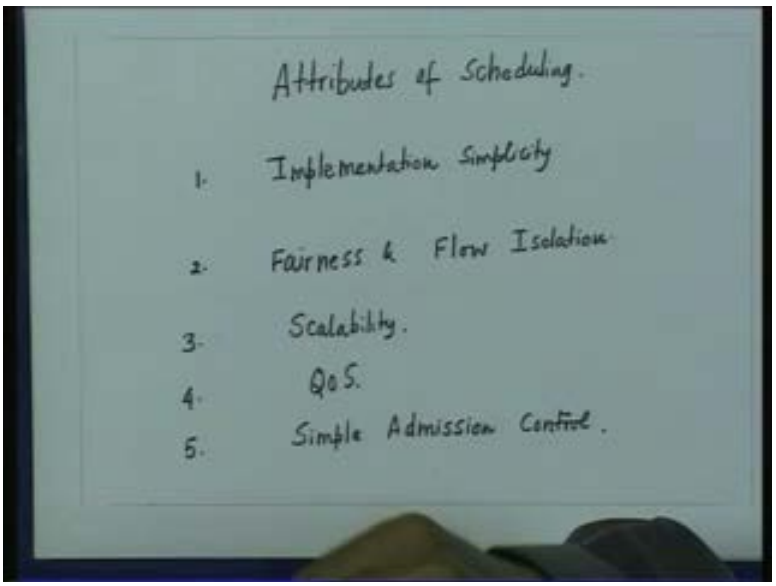
This sum however is going to remain constant over the all connections. Now, this is important because what does it mean that you cannot arbitrarily scheduled connections having arbitrary delay or bandwidth requirements. We need to then see that whether a particular set of delay vectors and the average arrival rate or the utilizations vectors, whether they are schedulable under the work conserving scheduling disciplines or not. Because, we have the kleinrocks conservation law in places.

So, this is very important. What is that we try to say is that we cannot design a work conserving scheduling algorithm to give arbitrary delays and arbitrary bandwidth guarantees to various flows. We can only do that with a non work conserving scheduling algorithms because the non work scheduling algorithms, a server will remain ideal even if there are packets in the queues and as a result, in order to give, it can therefore essentially decouple the delay and the bandwidth guarantees.

In this case, the delay and the bandwidth guarantees in some sense are coupled. However, note that typically in a non work conserving scheduling discipline, the average queuing delay suffered by connection will always be higher than the queuing delays than the average queuing delays in a work conserving scheduling disciplines.

So, what therefore we see now in the work conserving scheduling algorithms is the bandwidth and the delay guarantees are coupled and at the same time, the conservation law holds good. So, this is one important factor in the design of a non FCFS scheduling algorithm. Let us move forward and let us try to see what are the various attributes of a scheduling discipline that we would want to have, what are these various attributes.

(Refer Slide Time: 23:51)



So basically, our objective is to design a work conserving scheduling algorithms and which operates under the principles of this conservation law. But we would like to know what are the other attributes of scheduling algorithms. One is we would like to have implementation

simplicity and just elaborate on that. Second, we would like to have fairness; I mean this is the objective with which we really started this topic of fairness and the flow isolation and thirdly, we would like to have the algorithms which are scalable. Scalability, we need to provide the Qos and we need to have simple admission control algorithm.

Now, let us look at the point of the implementation simplicity. If you look at the implementation simplicity, then the scheduling algorithms must be very simple to implement. Now, in order to give the quality of service guarantees to the various flows, we cannot afford to have the scheduling algorithms very complicated.

Now, to give you an idea if you really want to do, take this scheduling discipline, scheduling decisions in a fast manner; then it really depends upon how much time is available for the scheduler to take a decision depending upon the packet input rates. So, that thing must be kept in mind in devising a scheduling algorithm.

So, note that what we are essentially having is that in a non first come first served scheduler, there are various, going to be various flows and here is going to be a scheduler. Now, this scheduler needs to take a decision which one of these packets it will pickup for the transmission on the output link to satisfy the objective to the fairness and the quality of service guarantee. So, the scheduler has to take a decision which one of these packets, it needs to pickup.

So this decision, a scheduling discipline must take within certain time budget. Now, what is the time budget available? That time budget available will depend upon what is the input arrival rate to these various queues so that the scheduling algorithm takes a decision fast in a line rate manner. So, that is one aspect of the implementation simplicity. The scheduling algorithm must not do lots of computations in order to determine which one of these packets it should pickup for transmissions.

Of course, one would argue that with the present day VLSI implementation, these days you can implement very complex logic of computing you know, which one of these packets to pickup for transmissions. So however, when we say that the implementation simplicity; of course, it depends upon the time budget. But it also depends upon the amount of scheduling state which this algorithm needs to store in order to take a decision. Why because; these scheduling states will need to be stored in a memory and if these scheduling states are large; then of course, it needs to be stored in some memory which needs to be outside the chip and if it is outside the chip, the access time will be large and therefore the time required to compute the decision will also be large.

As a result typically, we would like to store this scheduling state on a chip memory to reduce the access time and therefore the scheduling states, the number of scheduling states in a particular scheduling algorithm should also be small. So, that is one aspect of implementation simplicity. In order that the algorithm can scale also the number of operations that are required should be independent of the number of connections that are there that means it should be independent of this end.

For the implementation simplicity, your scheduling algorithm should be independent of … Typically, we would like to have the scheduling, this scheduling operations. The operations

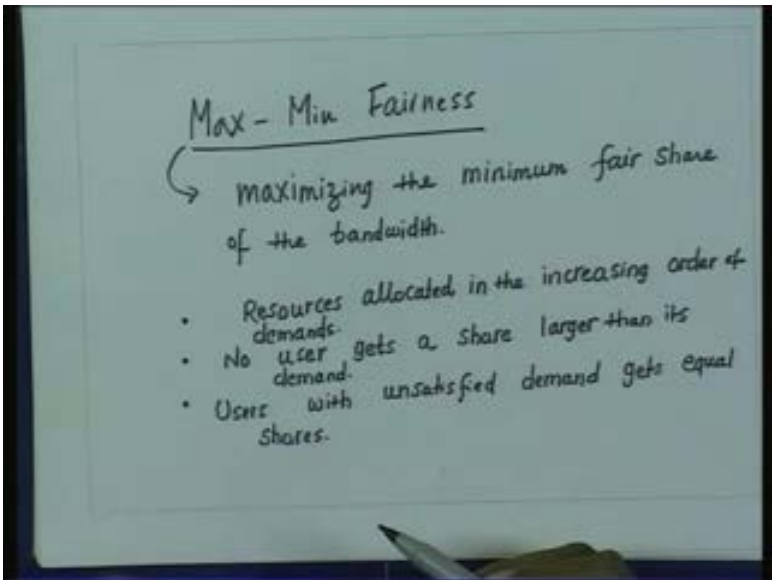
requires in the scheduling algorithms of order 1. We do not want to be order n. If they are order n, then the algorithm will not scale because typically in a core routers, the number of connections can be as large as 1,00,000 or 2,00,000 flows and if there are 1,00,000 and 2,00,000 that kind of flows, then obviously the number of scheduling operations will not scale if the number of scheduling operations are scaling order n.

So therefore, we would like to have the scheduling operations independent of the number of connections that are there. The number of scheduling states which an algorithm has to store should be smaller and the overall number of computations or operations that are required to do should also be small. So, it fits within the time budget. So, by implementation simplicity we mean that scheduling algorithm should not only have less number of operations to compute, but also should require less memory. So, that will make an implementation of a scheduling algorithm simpler – this is one thing.

Second thing is fairness and the flow isolations. Of course the scheduling algorithms should be fair and as we have already seen earlier itself that the TCP flows, the TCPs congestion control algorithm is biased towards connections with shorter round trip times and therefore we need a mechanism at the router which can give each TCP flow a fair share of the output link bandwidth and at the same time, it can product these TCP flows from non responsive flows like UDPs or some misbehaving sources.

So, the scheduling algorithms must provide fairness and must provide some kind of flow isolations. Now of course, this question is there that what you mean by fairness? This is obviously a question. Now, we will talk fairness in our discussion in the context of max min fairness. So, let me just explain what is meant by max min fairness.

(Refer Slide Time 31:32)



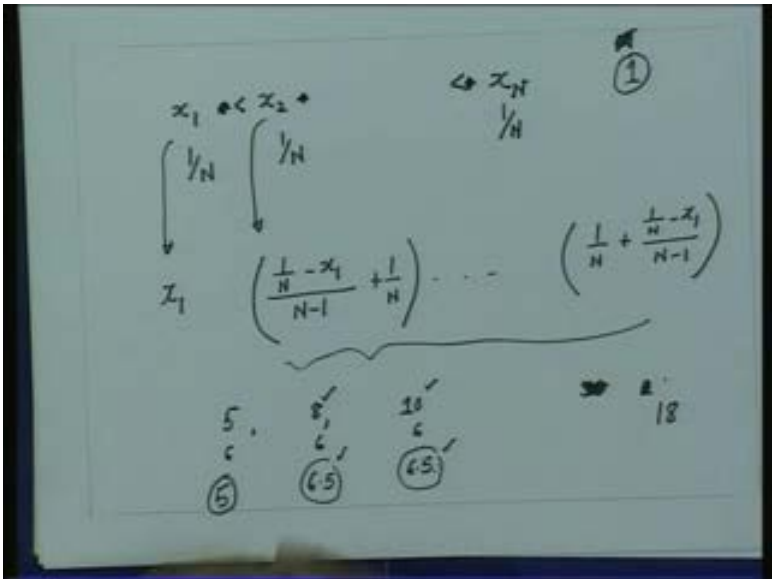
Max min fairness: so it actually means that we are maximizing the minimum fair share of the bandwidth. How does it work? It works that it adopts 3 principals. Resources are allocated in the

increasing order of their demands. So, we will allot the resources in the increasing order of their demands. No user gets a share larger than its demands and users whose demand are unsatisfied gets equal shares.

Of course, you can give waits. So, we are assuming that the users whose demands are not satisfied, they do not have, they are indistinguishable in any other sense. However, if you allocate weights, then the users with unsatisfied demands will get shares in proposition to foot to their weights.

So, we are assuming here that weights of all the users are one. Then the resource are allocated in the increasing order of their demand. No user will get a share larger than its demands and users whose demands are not satisfied, they will get equal shares.

(Refer Slide Time: 34:24)



Just to give you an example, let me just explain this with an example: suppose, there are users like with demands like x_1 x_2 so on till x_n and obviously, x_1 plus x_2 plus x_n; the sum of these demands and now these are let us say the bandwidths which a user wants and let us say that we have a total bandwidth of let say n, the total bandwidth is n; then obviously that there n users and one would say that fair share would be that you give each of these n users 1 by n bandwidth.

So, we can give them a short of 1 by n. Obviously, one thing is there that the sum of these bandwidths may be greater than n because their demands, it is possible that these accents may be greater than 1 upon n. So, we have arranged them in such a manner that this x_1 is less then x_2 is then x_3, so x_n is the highest demand.

Now, we allocate each users, let us say 1 by n. Now, it turns out that x_1 happens to be less than 1 upon n. If x_1 is less happens to be less than 1 upon n; then we allocate to the first user, the resource x_1 only and if we allocate the resource x_1, then we have a bandwidth available which

will be 1 by n minus x_1. This will be the extra bandwidth that gets available; divide by n minus 1 - this share we add it to each of these users. So now, this share gets added to each of these users.

Now again, we take x_2 and try to see whether x_2 is greater than this or not. If it turns out that x_2 is less than this, then obviously x_3 x_n all will be less than this and then the users, all the users will get this.

However, if x_2 is greater than this then the user 2 will get only x_2 and we subtract from this x_2 and then divide that equally among all the users. So finally, we will then achieve a fair share of the link bandwidth to all the users in the sense of max min fairness.

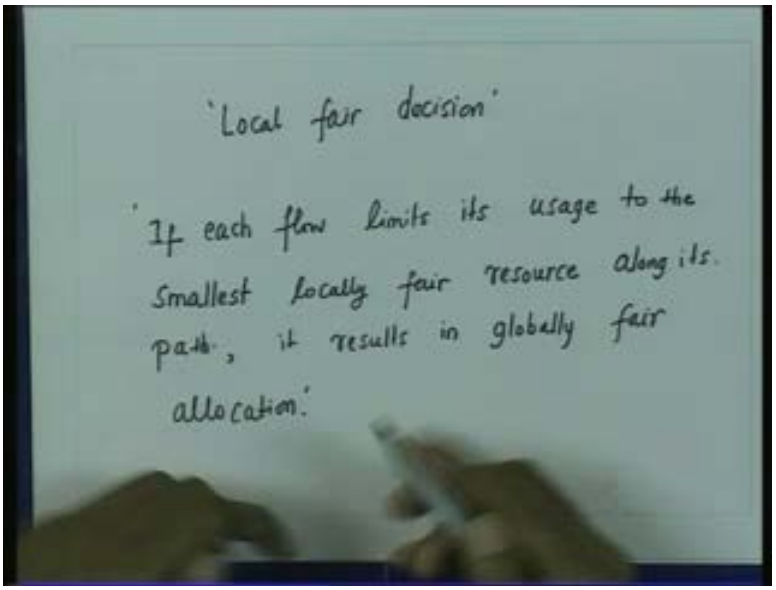
Let me just give you an example. Let us say that there are 3 users and the user 1 has a demand of let us say 5, user 2 has a demand of let us say 8 and user 3 has a demand of let us say 10. And, I wanted to say that we have a total resources of total resources of let us say that we have 20, let us say 18, total resources of 18.

Now, if we divide 18 in 3 users, among the 3 users; here of course I should assume that the total number of users are n and the link bandwidth was actually some kind of unity. Now, if you divide this bandwidth among 3 users, then each user should get 6, 6 and 6 units. But this user has the requirement which is less than 6, this user will get only 5 and I have 1 left which I can divide them into equal parts and make them 6.5 and 6.5

Now, 6.5 turns out to be less than 8 and 6.5 turns out to be less than 10. Now, I cannot do anything more than this. So, these users will be struck with 6.5. So, we satisfied this principles that no user gets share larger than its demand and users with unsatisfied demands get equal shares. So, if you see that these users, users with 8 and 10, their demands are unsatisfied. But they get equal shares of 6.5, 6.5. So, which is actually their fair share of the link bandwidths.

So, this is what is called as that we are allocating the resources in such a manner that we are maximizing the minimum share of the link bandwidth, we are allocating the resources in the sense of max min fairness.

(Refer Slide Time: 40:02)



So, one more thing which I would link to point out in terms of fairness is typically this scheduling algorithm takes only the locally fair decisions. Because, you are there at a network node where there are these non first come first serve scheduler and there are these various queues. So, when a scheduler is scheduling these packets in such a manner to give a fair share of the link bandwidth to these users, it is taking a local decision.

Now, remember that these nodes are connected together in the network nodes and fairness is actually a global objectives, fairness really is a global objectives. Now, it can be shown that if each source users sticks to its smallest locally fair resource, available resource, smallest locally fair resource; then, it will result in a globally fair allocation.

So, what we can show is that if each connection or each flow limits its resource usage to the smallest locally fair resource allocation along its path, it results… Now, we are not going to prove this but I am just trying to point out that in a network of a nodes where the nodes are individually, locally giving the fairness or ensuring the fairness and if a flow is going through several nodes, if this flow restricts its resource allocation, its resource usage to the smallest locally fair available resource; then overall, in network of nodes, it will result in a globally fair allocations.

Now however, the user usage pattern changes at let us say at some nodes. Suddenly, a source decreases its resource allocation. Then it will increase the resource allocation, resource usages of the other users. Now, it will take certain time to propagate these allocations to all the users. So, in order that we maintain these globally fair objectives, if the resource usages patterns are changing rapidly; then it may be difficult to achieve a globally fair allocation. However, the usage patterns are not changing rapidly, then each flow can ensure that it limits its resource usage to the smallest locally fair available resource allocation and then it can result in a globally fair allocations. So, that is what about the fairness aspects which you want.

Typically, if a scheduler is giving a fair allocation of the resources to each of its fit flows, then it is also ensuring protection from the misbehaving source. However, sometimes the converse may not be true; if you are providing protection by using some kind of a traffic regulators or shapers at the ingress to the scheduler, it does not mean then you will also give a fair allocation of the output look link bandwidth to the flows.

So therefore, while a scheduler is trying to be fair, it will automatically also ensure that it is protecting the other it is protecting the flow from the other misbehaving users. Because, that flow will definitely get a fair share of the link bandwidth. But if a node is protecting and trying to ensure the protection by using some kind of rate limiting or some kind of traffic shaping at the ingress point; then it does not mean that it will result in a max min kind of fair allocation to all the flows. So, that is what about the sort of fairness.

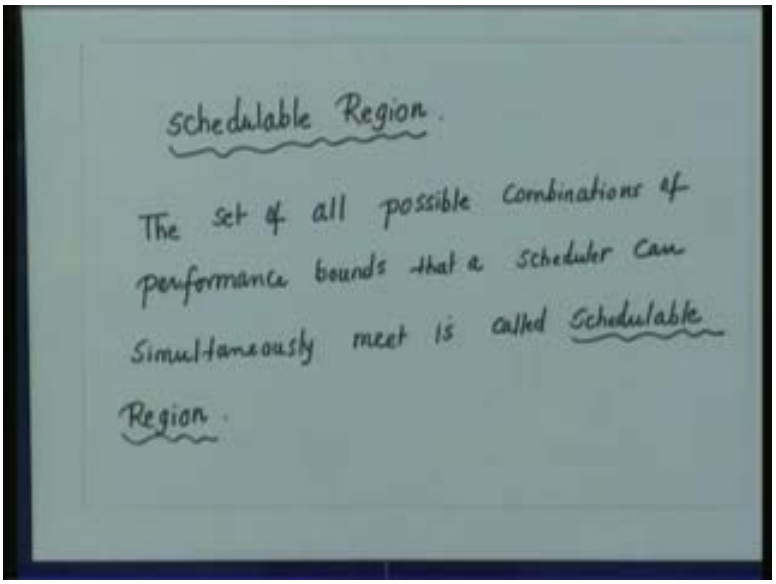
The third point that we were trying to see was of the scalability and that we have already seen that in order that a scheduling algorithms should be scalable. The scheduling computations which are required for the scheduling algorithms must be independent of the number of connections that are sharing the buffer and that will make the scheduling algorithm to be scalable. We want to have the quality of services guarantees. Now, what are the quality of service guarantees?

We want to give the delay bounds, we also want to give the delay jitter. We will see that it may be difficult to give the guarantees on the delay jitter in a work conserving scheduling disciplines. For this we may need to go for the non work conserving scheduling discipline. As for as a delay bounds are concerned, the worst case delay bounds we can give provided, the flows are rho sigma regulated. If the flows are not rho sigma regulated, then it may not be possible to give delay guarantees through a non first come first serve work conserving scheduling algorithms. By delay guarantees I mean, worst case delay amount. It may not be possible to give the worst case delay bound if the traffic is not regulated to rho sigma regulations.

We would also like to have one of the other attributes of the scheduling algorithm that the scheduling algorithms must lead to a very simple admission control decisions. What does it mean? Typically, as we have seen that a traffic source will give or specify its traffic descriptors which in internet are likely to be rho and sigma values and the quality of service attributes like the worst case delay bounds and so on. And then, the network will try to see whether you know it can admit these flows or not.

Now, when the network tries to see is that whether it can admit the flows or not; obviously, the network is trying to see whether these flows by admitting these flows, it will not affect the quality of service guarantee. In other words, the network is trying to see that all the flows are schedulable, they become schedulable under the scheduling discipline which the particular network node is implementing. So, whether these nodes are schedulable or not. So then, we need to define what is the schedulable region. So, let me just give you the definition of the schedulable region.

(Refer Slide Time: 47:58)



The schedulable region: the set of all possible combinations of performance bounds that a scheduler can simultaneously meet is called schedulable region. Now, given a schedulable region, we can have a very simple admission control policy. Essentially, we need to see whether these performance bound, these set of performance bounds of the connections that we want to admit falls within the schedulable region or not. Because, the schedulable region by definition is a set of all possible combinations of performance bounds that a scheduler can simultaneously meet. So, all we need to do is to characterize the schedulable region of a particular scheduling algorithm that we are deploying in the work conserving schedule scheduling discipline.

Now, it is sometimes very difficult to compute the schedulable regions of a scheduling algorithm. But if we can compute that then the network will have very simple admission control policy. We will try to see the schedulable regions of the certain scheduling algorithms that we will discuss in our subsequent discussions.

Now, these are the various attributes of the scheduling algorithms that we have seen. One of them - it should be simple to implement, it should achieve farness, it should give flow isolations, it should give quality of service guarantees, it should be scalable and it should also lead to some kind of a simple admission control mechanisms. It should give farness - we are saying that it should give farness, in the sense of max min farness that means it must maximize the minimum share of the bandwidth which means that resources were allocated in the increasing order of demands, no user will get a resource larger than in demand and users whose demands are not fulfilled or not satisfied, they all will get equal shares.

So, this way we can ensure that the resources have been distributed to different users in a so called fair manner. There could be different definitions of fairness but most scheduling algorithms that we are going to discuss are going to ensure this kind of max min fairness.

Now, you just like to see what are the degrees of freedom for implementing scheduling algorithms. There are various degrees of freedom for implementing a scheduling algorithm. One degree of freedom is that the priority. Now, what is the priority? The scheduling algorithms can divide these queues into the low priority queues and high priority queues and so on. There could be various priority levels also, priority level 1, priority level 2, priority level 3 and so on.

Now, in a work conserving scheduling algorithms if a priority level 1 is higher than all the other priority levels, then the scheduling algorithms will serve the packets from the priority level 1 only if there is a packet in that queue. If there is no packet in the queue, then it will lead to the other priority levels. Obviously, this might lead to starvation of the low priority users if the high priority users is all the time pumping the packets into it.

So therefore, if we are implementing a priority kind of schedulers, then we must have some kind of admission control and traffic shaping which will ensure that the high priority users are not leading to starvations of the low priority users. It requires an admission control mechanisms. The other the other degrees of freedom that scheduling algorithms can have is whether it is a sorted priority based or whether it is some kind of a frame based. Now, a sorted priority base; typically, each packet will be tagged with some kind of service tag.

So, whenever a packet comes into the system, it is tagged with some kind of a service tag. Now, how do you compute the service tag is a different matter. But suppose, these packets are tagged with service tags and then they are sorted in the order of those service tags and a scheduler will serve the packets in according to those service tags. These are sorted priority based. This could be frame based. By frame based we mean that we define some kind of frame length. Fixed, this could be fixed frame or this could variable length frame.

Now, the scheduler ensures that a maximum amount of traffic which can be sent by a flow within this frame is bounded. That is what a scheduler is trying to ensure. So, this is another degree of freedom in terms of implementing a scheduling algorithm.

Then, another third degree of freedoms that a scheduling algorithm can have in terms of degree of aggregations is whether the various flows have been aggregated into 1 super flows and then the scheduler is scheduling the quality of service objectives or farness objectives among these aggregated flows. So, there are various pros and cons of the aggregations that we would like to achieve at a network nodes. We can discuss that later but these are the various degrees of freedoms which are available in implementing a scheduling algorithm.

The next step that we would like to take is that given that these are the attributes of the scheduling algorithms and these are the various degrees of freedom that we have discussed today that a scheduling algorithm has and we would like to have some kind of a work conserving, non first come first serve scheduling algorithms, what are the different scheduling algorithms that are available today, that are implemented in practice today, that achieves these desirable objectives: so, that we will all take up in the subsequent lectures.

(Refer Slide Time: 55:16)