

## **Broadband Networks**

**Prof. Karandikar**

**Electrical Engineering Dept.**

**Indian Institute of Technology, Bombay**

**Lecture - 27**

### **Integrated Services Internet (IntServ) & RSVP**

The traditional internet model is a best effort internet model in which the network makes every attempt to transmit the packet but does not offer any explicit quality of service guarantees in terms of delay bandwidth or delay jitter. Now, this model was considered quite adequate for most non real time applications which are prevalent on the internet; like for example telnet or file transfer or web browsing email, these kinds of applications.

Now, these kinds of applications truly speaking do not require any hard guarantees in terms of delay or delay jitter. They can tolerate some amount of delays. But when broadband multimedia real time applications started their way into the internet in terms of either video conferencing or internet telephony and voice over IP applications, it was felt that these kinds of applications require explicit quality of service guarantees in terms of either minimum bandwidth or delay and delay jitter.

So therefore, the internet engineering task force IETF started working on a new group to develop a framework for defining the services and service model and at the same time, an architecture for an internet which can give quality of service guarantee. **Now**, that model was called integrated services internet model. We will see today in this lecture, the salient architectural features and motivations for this new integrated services internet model.

Unfortunately, the paradigm of the integrated services internet model was to give quality of service guarantees at the individual applications flow level and later on after the architecture and the service model was formulated, it was discovered in the real life or in practice that it is not possible to give explicit end to end quality of service guarantees to individual applications and flows, primarily because large number of states corresponding to each application flow will have to be maintained in the intermediate nodes and routers and clearly it is not possible to maintain such large number of states for such large number of applications.

So, to address that problem therefore, new architectural framework to provide quality of service guarantees at a coarser granularity was proposed and that was called differentiated services framework. We will also study later on these differentiated services framework. So, on the one spectrum of quality of service guarantees, we have best effort internet model which does not offer any quality of service guarantees and on the other spectrum of the QOS, we have integrated services internet model which gives quality of service guarantees at a very fine granularity. In between, the internet has come out with a new paradigm or a new framework which as I just told

that is called as differentiated services framework. It is the differentiated services framework which is likely to be more popular in providing quality of service guarantees and bandwidth provisioning in the present day internet.

We will see the features of differentiated services later. But first let us look at the model which was proposed to provide explicit quality of service guarantees in the internet and the individual application flow level which is called as the integrated services internet model.

(Refer Slide Time: 4:36)



So, this is an integrated services internet model. Now, this internet service model has been specified in the IETFs RFC which is 1633. Now, the key assumptions in this internet service, popularly called internet service; the key assumptions are that resources have to be explicitly managed, the resources must be explicitly managed and these resources must be explicitly managed through a combination of resource reservation and admission control.

So, what must happen is that each individual application flow must specify its resource requirements and then the network must determine through an admission control whether the call can be accepted or not, whether the required or requested quality of service guarantees can be met or not. If it can be met, then the flow will be admitted and then later on a flow specific state is required to be maintained in the router. So, these are the key assumptions of this service model.

Now, as we just said that since the integrated services internet model requires resource reservation and admission control; obviously, each flow must specify its service requirements and after the source has specified its service requirements and its traffic characteristics, the network will make certain service commitments. So, the network will then have to make service commitment.

Now, these requirements and commitments can be met either at the individual flow levels, at the application level and therefore the service requirements needs to be specified in terms of application level performance or they can also be made at the granularities of classes of flows or entities and which is actually the performance has to be judged in terms of the resourced sharing. So, the int serv model will allow both - at the individual application flow level or at the entities level.

Note, as I was just mentioning that the courser granularity is available more at the differentiated services framework but as we will see later on that the entire paradigm of providing quality of service guarantees is completely different in the differentiated services framework. So, when we say that the resource management at the entities level will take place in the int serv, the int serv has 2 basic features.

One, it operates on an end to end level. So therefore, all the intermediate nodes in the network must provide quality of service guarantee because it operates on an end to end level and secondly, the int serv model gives end to end quality of service, so it defines actually an end to end service. As opposed to that we will see in the differentiated services framework, it does not offer any end to end quality of service guarantees. It may give only in certain domains in between. So, that is that is one thing. Second thing is that we will see later on that the differentiated services network does not define any service model. It only defines forwarding treatment.

So, they are the 2 fundamental differences between the int serv model and the differentiated services model. Although, popularly it is told that in the integrated services internet model, you have to give quality of service guarantees at the individual flow level and at the differentiated services framework, you have to give quality of service guarantees on the aggregates of flows. So, **while** that is true. But at the same time you must keep in mind that in the int serv model, you can also aggregate some application flows into one bigger flow between two customers and can give end to end quality of service guarantees by maintaining this aggregate flow specific states in each individual router along the path.

Now, even though you are not giving quality of service guarantees at the application flow level, you are giving at aggregate flow level; it would still be defined within the framework of the int serv.

(Refer Slide Time: 9:10)



Now, let us look at what are the service requirements that occur for the individual flows. Now obviously, the service requirements for the individual flows will be in terms of per packet delay. The packet delay is one of the most important criteria for real time multimedia application flow which is the primary motivating factor for defining the int serv architecture.

Now, this per packet delay usually, you would require a bound on the maximum and the minimum delay. Typically, you will not require guarantees in terms of average packet delay. You will require guarantees in terms of worst case delays because essentially these are real time applications. Then you may require guarantees in terms of what is the minimum bandwidth that can be given and what is the sustained bandwidth that is available.

The third service requirements for an individual flow can be in terms of the packet loss probability. Though, the real time application flows are somewhat insensitive to packet loss in sense that they can tolerate some 10 to 20% of the packet loss depending upon the particular applications; but even then, different application flows depending upon their applications requirements may have different requirements in terms of the packet losses.

Now, all the flows in the int serv model can be defined in therefore 2 types: one is the real time applications and another one is the elastic applications. Now, int serv model was really architected to take care of these real time applications, the elastic applications were actually the applications running within the best effort service model. They are not so much sensitive to packet delays and the bandwidths.

Although, they may be sensitive to the packet loss probabilities, so therefore they have an in built mechanism for the retransmission of the packets if some packets are lost. So, even if the packet arrives late, the applications can still make use of it. So **though**, even though they maybe sensitive to somewhat packet loss but that is addressed by having an inbuilt retransmission mechanism.

So, now let us look first at the real time applications, what kind of real time applications we are talking.

(Refer Slide Time: 11:37)



**Broadband Networks**

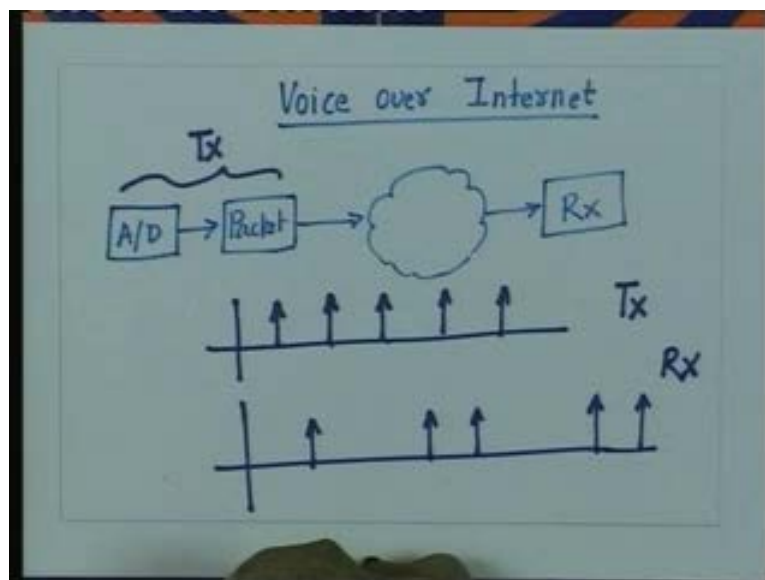
### Real Time Application

We consider only playback real time Applications

- Performance of playback applications is determined by
  - Latency
    - Applications like phone call are sensitive to latency
  - Fidelity
    - Applications exhibit wide range of sensitivity to fidelity loss
- Real Time Applications
  - Tolerant ( Can tolerate loss of fidelity )
  - Intolerant ( Cannot tolerate loss of fidelity )

Now, in the int serv model, we consider only a playback kind of real time applications. To understand ah what is the playback real time applications, let me just explain what is meant by a playback real time applications.

(Refer Slide Time: 11:59)



Now typically, what happens? Let us say that this is a voice over internet kind of applications. So, in the voice over internet, let us say that you digitize the voice. So, you have an analog to digital conversion and then you packetize the voice, packetize it and send it over the internet. Now, this is your receiver where you will play the packets.

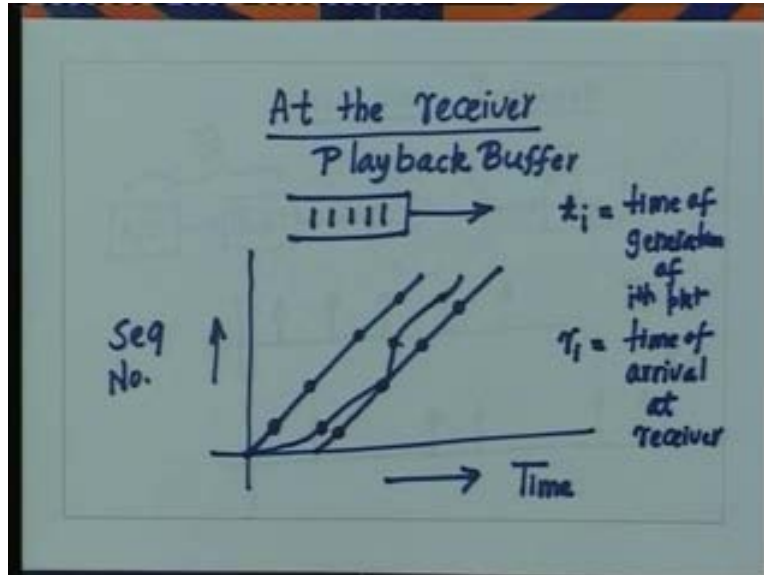
Now, what happens at the source? **The packets**, there may be certain periodicity with which the packets might have been generated. So, let us say that these are the samples which have been generated at the transmitter. So, this is at the transmitter, so at this end. Now, in the internet, each packet may suffer certain delays. But what is more important is that different packets may suffer different delay. So therefore, this packet may arrive here and this packet may arrive here and this packet may arrive only here and then this packet may arrive here and the fourth packet may arrive here.

Now, as you can see here that the periodicity with which the packets were generated at the transmitter, that periodicity has been completely lost at the receiver. So, if these packets are given to the receiver for play out, obviously there will be a distortion in the speech. **What is**. Why this is happening? This is happening because different packets are experiencing different delays. While it is true that depending upon the applications of voice, these voice applications can tolerate some amount of delays but what is more significant here is that different packets are encountering a different delay. That is what is called as delay jitter.

Now typically, depending upon the applications, the amount of delays that can be tolerated can vary from 300 milliseconds to 150 milliseconds. Actually, the human ear cannot perceive delays less than 150 milliseconds. Any amount of delays from 150 milliseconds to 300 milliseconds may be tolerable depending upon applications. Let us say for example, you are having a streaming audio over the internet from an internet, from an audio server; then some amount of delays can be tolerated maybe upto millisecond. But if you are having a two way interactive telephone conversation over the internet; then obviously, the amount of delay that can be tolerated will have to be less.

So, it depends really upon the application, what is the maximum amount of delay that can be tolerated. But even if you assume that the maximum delay that can be tolerated is less than the permissible limit, what is really happening here as we have seen that because different packets are encountering different delays, there may still be a perceptible degradation in the speech quality. So, what is the solution?

(Refer Slide Time: 15:46)



The solution is that at the receiver, we use a playback buffer. So, at the receiver, we use a playback buffer. So, we buffer the packets and then we play out the packets at an appropriate time interval such that the periodicity with which the packets like here, the periodicity with which the packets were generated at the transmitter, that periodicity is maintained. To again explain this fact, something like this that let us say, the packets were generated at this. So, this is like a time and this is like a sequence number of packet. So, this is one packet has been generated, second packet has been generated, third packet has been generated and so on.

However, these packets arrive differently. So, this packet arrives here, this packet arrives here, this packet arrives here and this packet arrives here. So, as a result, the arrival times are ... now. What you do is that you start playing the packets from at this point. So, the first packet is played here, second packet is played here, third packet is played here and the fourth packet is played here.

So, as you can see that this packet which arrived here had to wait in the buffer for this much amount of time and then it was played out here. This packet which arrived here; since it suffered the maximum delay, did not have to wait in this playback buffer. It is played out immediately. This packet which suffered somewhat less delay, it waits in the buffer for some amount of time and then it is played out here. Similarly, this packet which suffers this amount of delay, it waits for certain amount of time.

Now, what you can do is that if you can adjust the playback time in such a manner that even those packets which suffers the maximum delay, they can also be accounted for; then, we are done. We can then appropriately delay the packets in the playback buffers and then play them out. As a result, there will not be any packet loss. But there will be some latencies in playing out the first packet. Now, this latency will depend upon what is the maximum delay that is there in the network. Now, if the network can bound this maximum delay, then we can reduce that latency.

(Refer Slide Time: 18:52)

$$p_i = t_i + d_{max} \quad p_i = t_i + \hat{d}_{avg}$$
$$d_i = r_i - t_i$$
$$d_{buffer} = d_{max} - d_i$$

Just to give you a complete flavor, let us say that  $t_i$  is the time of generation of the packet,  $i$ 'th packet, so the time of generation of  $i$ 'th packet and  $r_i$  is the time of arrival at the receiver. The time at which the packet will be played out that is  $p_i$ , the time at which this  $i$ 'th packet will be played out  $p_i$  will clearly be equal to  $t_i$  that is the time at which it was generated plus  $d_{max}$ , where  $d_{max}$  happens to be the maximum delay.

Note that this packet delay, actual delay has been  $d_i$  has been equal to  $r_i$ .  $r_i$  is the term at which it was received and  $t_i$  is the time at which it was transmitted. So,  $r_i$  minus  $t_i$  is actually the delay. So, the packet has to wait in the playback buffer for an amount of time. So, this is  $d_{buffer}$ , the delay in the buffer is actually equal to  $d_{max}$  minus  $d_i$ . So, for this much amount of time the packet has to actually wait in the buffer.

**So, this would be** so if the minimum packet delay is some  $d_{min}$ , then the amount of time for which the packet has to wait in the buffer is equal to  $d_{max}$  minus  $d_{min}$  that is the maximum amount of time. So therefore each packet has to wait in the buffer equal to the delay jitter that is the difference between the maximum and the minimum delay. If we can reduce this delay jitter that means if we can bound the  $d_{max}$ , then we can also reduce the latency. Such applications are called the playback applications. In the int serv paradigm, only the playback real time applications have been considered for defining the framework or the architectural constraints.



(Refer Slide Time: 21:09)



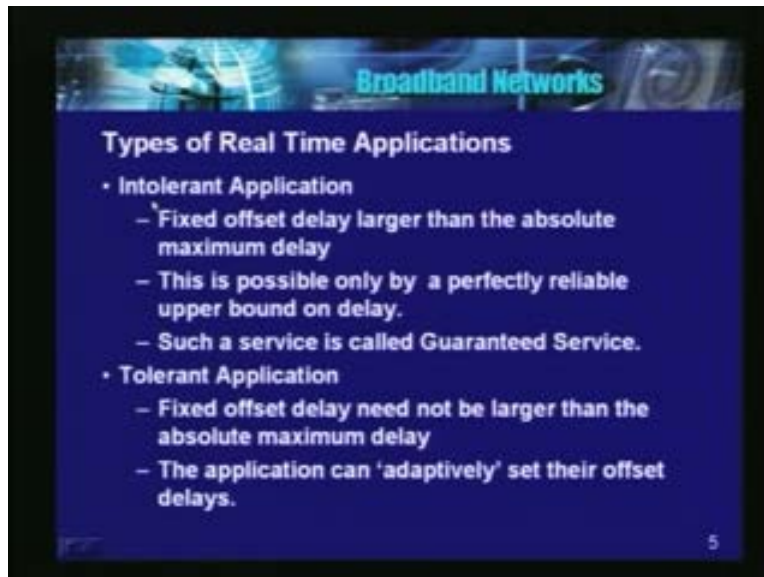
Now, we now go back to the real time applications, our discussion here, we consider only as we have just discussing playback real time applications. Now, as we just saw performance of playback applications is determined by 2 factors; one is the latency, so as we have just talked that applications like two way interactive phone call, they are actually more sensitive to the latency.

Now, if we play out some packets, **if we reduce the play out**, if we reduce the time for which a packet has to wait in the buffer, then it may and if you do not really worry about that  $d_{max}$  that is we do not compute the playback time for each packet to be  $t_i$  plus  $d_{max}$  but some other time that is  $p_i$  is equal to  $t_i$  but let us say some  $d$  average delay; then it may so happen that some packets whose delay may be greater than the  $t$ , the average, they may arrive after their playback time has passed. And as a result, those packets cannot be played out and will be considered as lost. Because of these packet losses which will happen at the receiver, there will be certain loss in the speech quality or fidelity.

Now, real time applications, they really exhibit wide range of sensitivity to the fidelity. Some applications can tolerate more packet losses; some applications can tolerate less packet losses. So, depending upon that we can define the real time applications into 2 types; either they can be a tolerant real time application that means they can tolerate some loss of fidelity or they can be intolerant real time applications **real time applications** which cannot tolerate any packet loss or loss of fidelity.

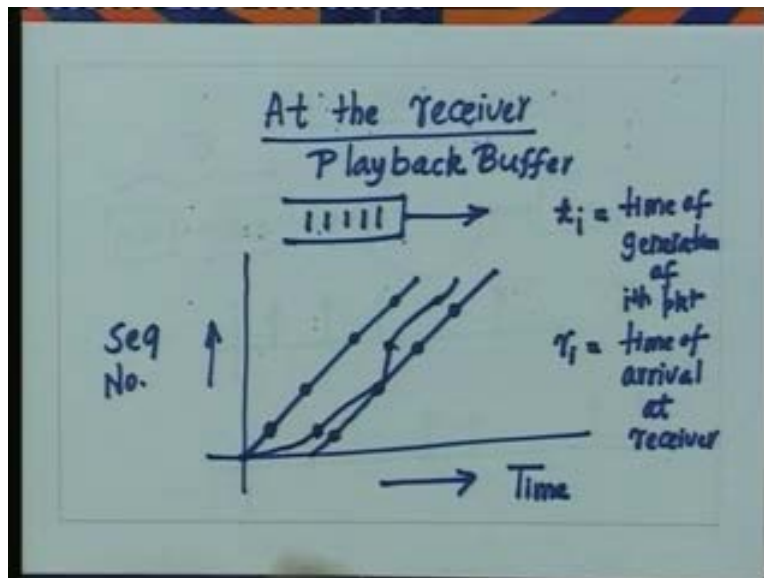
However, remember that both these applications that is tolerant and intolerant applications are the playback applications. That means their packets can be buffered in the playback buffer and then can be played out. If any packet arrives after its playback time has expired, then that packet will be considered as lost in both - tolerant and intolerant applications. But tolerant applications can tolerate somewhat more packet loss and intolerant applications can tolerate very less packet loss. So, as you can see therefore that we need to construct different service models to take care of tolerant and intolerant playback real time applications.

(Refer Slide Time: 23:56)



Now, as you can see, in the intolerant applications that means if you cannot tolerate any packet loss in your playback delay.

(Refer Slide Time: 24:10)



So, if your situation is as I was just discussing here, if you cannot tolerate any packet loss here, then it is necessary that you need to put your playback time in such a manner that it takes care of the maximum delay. That is **you have to** every packet you have to make it  $t_i$  plus  $d_{max}$ . So, we can achieve here as by saying like this that in intolerant application, fixed offset delay can be larger than the absolute maximum delay. Now, if you do this and if you have to reduce the

latency, then in order to reduce the latency you need a perfectly reliable upper bound on the delay.

So, **in order to reduce the latency** sorry in order to have no packet loss and in order to fix an offset larger than the absolute maximum delay, we need to know perfectly what is this absolute maximum delay and therefore we need a perfectly reliable upper bound on the delay and also of course, to reduce the latency this bound has to be less.

Now, a framework, a service framework which provides a upper bound on the delay is called guaranteed service within the int serv framework model. Now, tolerant application, in case of tolerant application; now note that tolerant applications are those applications which can tolerate some packet losses. Now, in this case, the fixed offset delay need not be larger than the absolute maximum delay. What does it mean?

(Refer Slide Time: 25:56)

$$p_i = t_i + d_{max} \quad p_i = t_i + \hat{d}$$

$$d_i = r_i - t_i$$

$$d_{buffer} = d_{max} - d_i$$

$$p_i = t_i + \hat{d} \quad r_i > p_i$$

**You do not have to**, you do not have to fix, you do not have to fix the playback delay as  $t_i$  plus  $d_{max}$  every time. You can adjust the playback delay for every applications  $t_i$  by some  $\hat{d}$ , where this  $\hat{d}$  will be computed separately. It may so happen when some packet  $r_i$ , some  $i$ 'th packet may arrive at time  $r_i$  which turns out to be greater than  $p_i$  so computed. So, if the packet arrives later than its playback time, then that packet will be considered as lost and therefore there will be some fidelity loss.

However, since these are tolerant applications and they can tolerate some packet losses; what is now more important therefore is to adjust this  $\hat{d}$ , this  $\hat{d}$  here in such a manner that large a proportion of the packets are able to arrive before their playback time has expired and therefore they can be played out to absorb the delay jitter.

So, the fact is therefore that we do not require a perfect reliable upper bound on the maximum delay that is possible in the network. We require only somewhat loose guarantees. That model we

call it to be a controlled load service model which can be applied for tolerant applications. So, actually speaking, in integrated services, when this integrated services internet framework was architected, several service models were debated and considered and finally the int serv came to conclusion of defining paradigm or frameworks for two service models. One is the guaranteed service model and another one is the controlled load service model.

So, guaranteed service model is meant for intolerant applications which requires a perfectly reliable upper bound on the maximum delay and controlled load service model is meant for tolerant applications which exhibits less sensitivity to the packet losses. They can tolerate some packet loss and therefore they do not require a hard, perfectly hard guarantee or perfectly reliable upper bound on the worst case. So, let us see what is the guaranteed service model meant for intolerant applications.

(Refer Slide Time: 28:18)



Since you require a perfectly reliable upper bound on the delay; obviously, it is based on worst case assumption of the flows. So, here you need to assume, since **you do not** you do not want any packet loss and since you want that a guarantee on the delay; you have to assume the worst case scenarios for the flows.

On the other hand, in the controlled load model which is actually meant for tolerant applications; the bound is not based on the worst case assumption of the flows. Controlled load model actually emulate some kind of a lightly loaded network. Typically, this model can be implemented by using some kind of a weighted fair queuing scheduling algorithm that we have discussed in combination with an admission control mechanism.

So, what we can do is that **the services or the applications** the flows or the applications which have subscribed for the control load model, they can be given a little bit higher weight in the weighted fair queuing model and therefore they will get certain priorities over the other users and to limit the amount of such flows, you can employ admission control so that you do not admit

large number of such flows. Otherwise, even though they have been given higher weights in the weighted fair queuing, individual application flows will end up seeing a congested network.

So, what you do really is that you give them higher weights but you control the amount of traffic that they are injecting through some kind of traffic regulation or admission control, so **by** through admission control. So, if you do that then these applications, for these applications it will appear as if the network is lightly loaded.

Now, tolerant applications therefore, the thesis, the premises is that the premises or thesis says that most of these applications will behave properly and very well if the network is lightly loaded. So therefore, what we are trying to do is that we are trying to emulate a lightly loaded network. So, the calculations are therefore not based on the worst case scenarios of the flows.

(Refer Slide Time: 30:48)



As a matter of fact, the model is motivated by the conjecture that the performance penalty for applications will be small compared to the gain that you will obtain in network utilizations through statistical multiplexing. Because what will happen clearly is that in guaranteed service model, we want hard guarantees in terms of the worst case delays and therefore your admission control is based upon the worst case assumption of the flows. Your statistical multiplexing gain is clearly very low in the guaranteed service model. Maybe, you have to admit flows based on peak rates. But in the control load model **in the control load model**; since you are not admitting based on worst case, your statistical multiplexing gain maybe little bit higher.

(Refer Slide Time: 31:45)

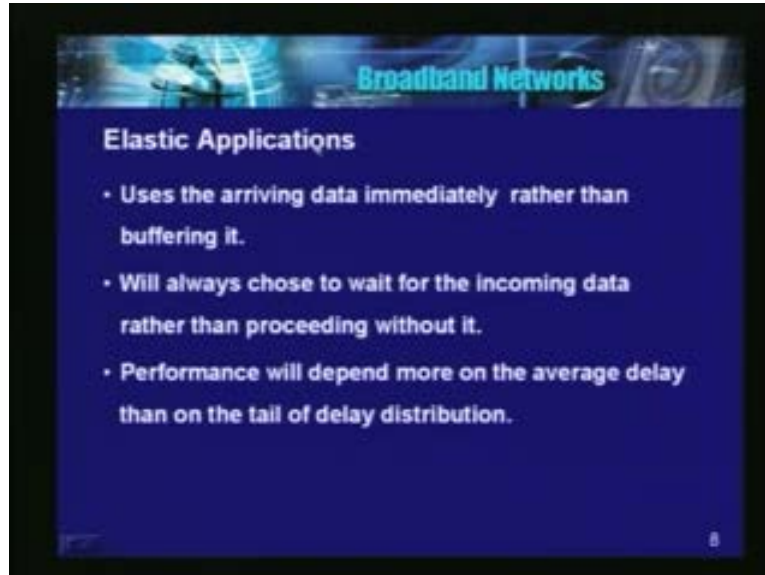


You can of course augment this service model. It has been actually found that many audio and video applications can actually adjust the data rate. So, they can be rate adaptive applications also. For example, they can use some kind of rate control schemes in MPEG codings. So, depending upon the congestion conditions in the networks, the network can actually notify to change their traffic characteristics and the rate adaptive applications can change not only their playback point but also the traffic pattern. **So, this is like a**, so what we are saying is that you have two types of applications: tolerant and intolerant.

Now, the tolerant applications are actually delay adaptive. A delay adaptive is that you can adjust the playback point depending upon the network delays that you are observing. So, you can periodically adjust the playback point. Of course, by periodically adjusting the playback point, there is a possibility that some packet losses will occur because some packets will arrive after their playback point has expired. But your applications are not so sensitive to a small packet loss. So, these are delay adaptive applications.

You can also have, apart from delay adaptive; you can also have a rate adaptive application where say by changing the quantization level in your MPEG encoders, you can actually change your data rate. So, if you observe the congestion conditions in the networks, you can reduce your data rate by making the quantization a little coarser. **So, these are** so therefore, there can be another service model which also takes care of the delay adaptive applications.

(Refer Slide Time: 33:42)

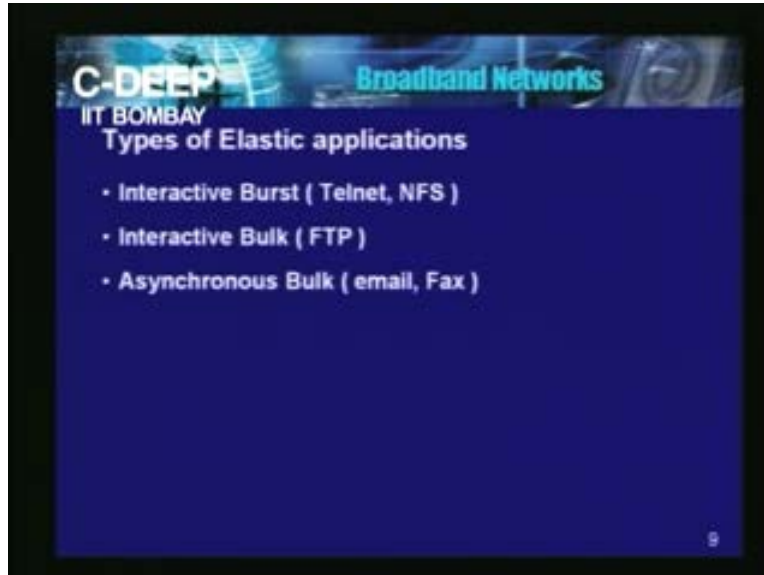


The second application that we were talking is the best effort applications, also called as elastic applications. Here the thesis is that it uses the arriving data immediately rather than buffering it in the playback buffer and **it will always wait to**, it will always choose to wait for the incoming data. So, if some packets have been lost, then it can wait for the retransmitted packet to arrive. There is no hurry to play that packet out and more importantly while in the case of real time application, the performance depends on the worst case bound or delay. Here, the performance depends more on the average delay than on the tail of delay distributions.

So really speaking, the kind of guarantees which elastic applications will require is that an average delay should be lower than the permissible limits. So, as long as the queues in the network or queues in the routers are less than certain or always maintained less than certain permissible limits, your average delay will be lower and therefore the elastic applications will perceive a good performance.

So, the whole idea is really to reduce the average delays of the packets for the elastic applications and that you can do by reducing the average queuing delays and that can be done by actually reducing the congestion conditions in the network.

(Refer Slide Time: 35:20)



Now, even elastic applications actually differ in their requirements for the average delay **queue**. If you really see, there are 3 kinds of elastic applications that we can talk. One is the interactive bursts which are like, applications like telnet or NFS kind of thing, then interactive bulk which are like Ftp applications and then asynchronous bulk which are email fax over IP, those kinds of applications.

So if you really see, among these three kinds of elastic applications, it is the interactive burst really speaking which requires the least average delays and then interactive bulk may tolerate more average delay than the interactive burst and a asynchronous bulk can actually tolerate much higher amount of average delay.

So really, in terms of their average delay requirements also, these elastic applications differ and one can give bandwidth guarantees or delay guarantees for these applications by using some kind of packet scheduling algorithms and put applications like telnet and NFS kind of applications into different queues with higher weights and put applications like asynchronous bulk with their lower weights.

So, by using appropriate packet scheduling algorithms at the routers, it is possible to guarantee these kinds of average packet delay. Though, for elastic applications, no hard guarantees are required, only qualitative guarantees are really required in the int serv model. In the int serv model really speaking, the hard measurable quality of service guarantees is given only to the guaranteed service traffic.



(Refer Slide Time: 37:09)



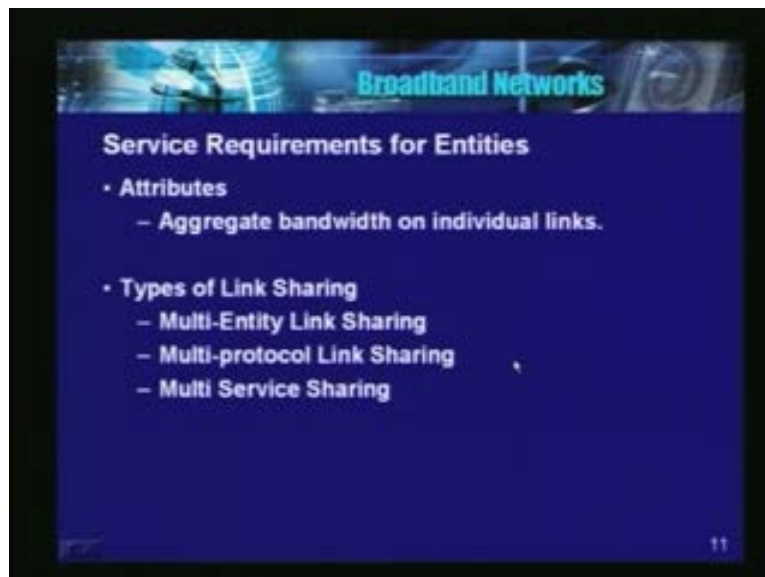
The slide is titled "Broadband Networks" and "Service Model For Elastic Applications". It lists the following:

- ASAP Service Model
- Delay Classes
  - Interactive Burst < Interactive Bulk,
  - Interactive Bulk < Asynchronous bulk

The slide number 10 is in the bottom right corner.

So, this service model for elastic applications is really what is called as ASAP - as soon as possible service model. And as we have just discussing, there are three delay classes: interactive burst requires lower delay than the interactive bulk and the interactive bulk requires lower delay than asynchronous bulk, asynchronous bulk can tolerate much higher amount of average delays and these are the applications like email and so on.

(Refer Slide Time: 37:35)



The slide is titled "Broadband Networks" and "Service Requirements for Entities". It lists the following:

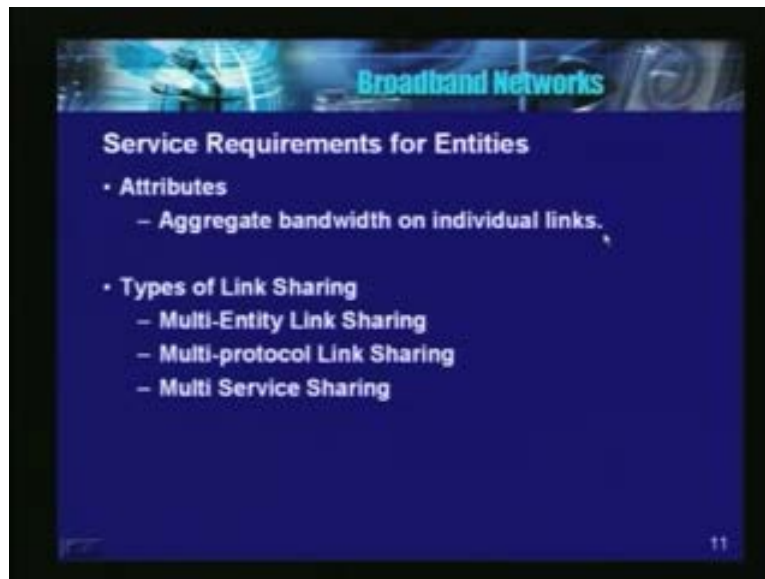
- Attributes
  - Aggregate bandwidth on individual links.
- Types of Link Sharing
  - Multi-Entity Link Sharing
  - Multi-protocol Link Sharing
  - Multi Service Sharing

The slide number 11 is in the bottom right corner.

Now, this is we were talking at the individual applications flow level. **It is also possible,** so as far as the individual applications flows are concerned, their quality of service attributes are specified in terms of per packet delay or bound on the maximum and the minimum delays. The delay is

one of the most important quantities there and most of these applications are playback applications. Now, we can talk of when we talk of giving end to end quality of service guarantees under the int serv paradigm for the classes of flows or the aggregates or at the entity level, the service commitments and service requirements are of different types.

(Refer Slide Time: 38:25)



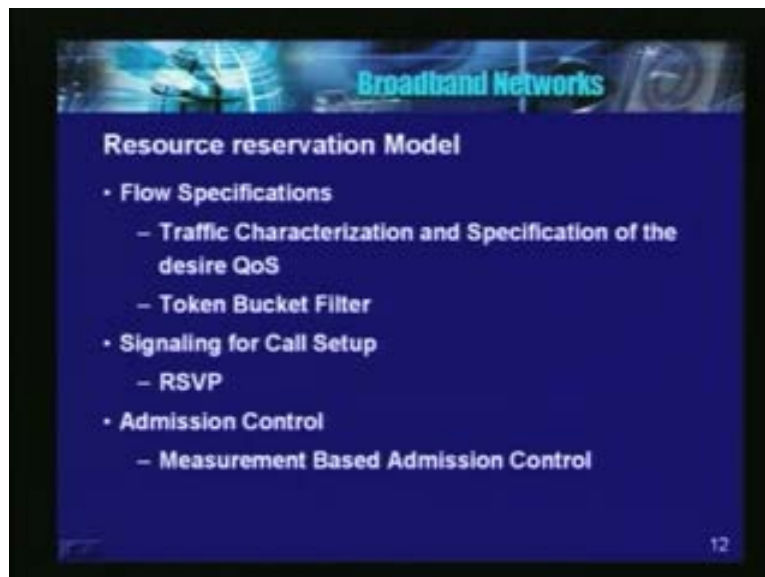
So, the service requirements for entities will be more in terms of aggregate bandwidth on the individual links that is what they will be talking and these links, so different entities could be there; for example, these links, the bandwidth maybe shared among multiple entities or multiple organizations or you may allocate bandwidth depending upon various protocols like for example the IP V 4 traffic or IP V 6 traffic and so on or you may allocate it depending upon different services. So, you can have a multi service sharing as well.

(Refer Slide Time: 39:04)



So, this brings so, what we have specified right now is the requirements of the services both from individual application flow point of view and from entities point of view. Now, as you are saying that the paradigm or the framework of the integrated services internet model is that the resources have to be explicitly managed; whether they are individual application flows or whether they are the entities.

(Refer Slide Time: 39:39)



So, it requires therefore a resource reservation model and the resource reservation model which has been specified in RFC 1633 is that each individual flow will specify its traffic characteristics

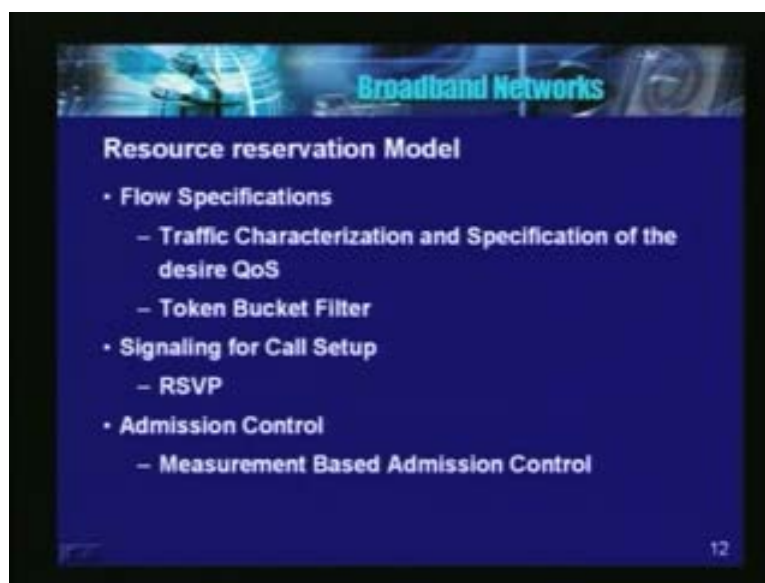
and the desired quality of service attributes. The traffic characteristics will be specified in the form of a token bucket filter. They have already studied the token bucket filter.

The token bucket filter is specified by two parameters: one is the token generation rate  $r$  and second the bucket depth  $b$ . So, each flow has to specify its characteristics through  $r$  and  $b$  which is what is called as  $t$  specs or the traffic specifications and it has to specify its quality of service characteristics or reservation characteristics through reservation specs which is called as  $r$  specs.

Now, like in the telecom networks, before you can actually start transmitting the data, you have to establish a call; in the int serv model also, before you can start transmitting the data, you have to undergo a signaling procedure. Note that in the best effort internet model, there was no need for signaling, there was no signaling. The model, the delivery model was actually a connectionless datagram delivery model.

Now, here however you have to undergo a signaling procedure just like in the broadband ATM network, here also you will have to undergo a signaling procedure and that signaling in the int serv model is called resource reservation protocol or RSVP. We will see shortly what are the attributes or features of the signaling protocol and how this signaling protocol is distinct or different from the QOS signaling that is done in networks like ATM or packets or circuit switch networks.

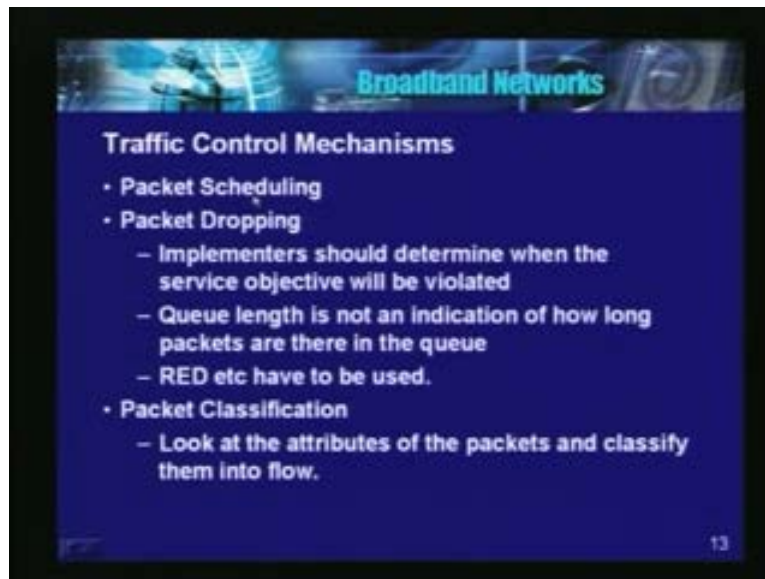
(Refer Slide Time: 41:35)



And then, once the  $t$  specs that is the traffic specifications and the reservation specifications are specified through RSVP, the network undergoes an admission control algorithm which could be measurement based and determines whether the flow can be admitted or not and once the flow is admitted a virtual circuit is setup between the source and the destinations on an end to end basis and then the flow can start transmitting the packets.

Since the int serv defines an end to end service model, a virtual circuit needs to be established for each individual application flow on an end to end basis and that is the primary reason why a flow specific state is required to be maintained in each individual routers leading to a leading to maintenance of large number of states in the routers.

(Refer Slide Time: 42:47)



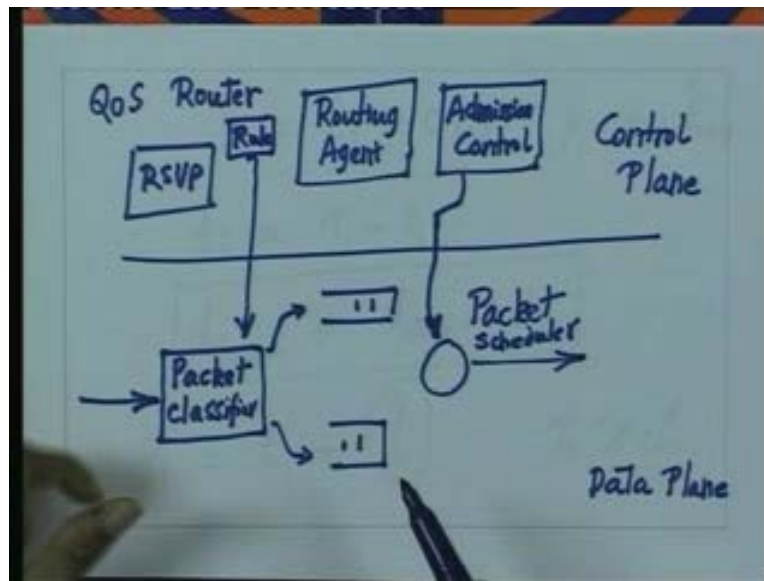
So, the traffic control mechanisms that are used, so up so one is as we have discussed, we require a signaling protocol an admission control mechanism. Then we also require individual traffic control mechanisms in the int serv routers. One of the important mechanisms is packet scheduling algorithms. We have already discussed these packet scheduling algorithms. Some of the popular packet scheduling algorithms which are used to provide quality of service guarantees is weighted fair queuing algorithm. We know that a bound on the worst case delays in the case of weighted fair queuing algorithms can be given if the traffic is token bucket regulated or rho sigma regulated. We can give guarantees by using weighted fair queuing in terms of worst case delay bound.

Now therefore, it is these weighted fair queuing kind of algorithms have been the primary implementation mechanisms for the QOS routers. We have to also take into account, how we will drop the packets. We have to determine, for example that by dropping the packets the service objectives are not violated. So therefore, the applications like real time applications which exhibit lot of sensitivity to packet loss or fidelity loss, we have to sort of take care that their packets are not lost.

Unfortunately, the length of the queue is not an indication of how long the packets are there in the queue. It is not an indication of the congestion conditions in the networks; you have to take decision on dropping the packets based on some long term average of the queues. So therefore, algorithms like weighted RED or RED random early detections, they have to be sort of used. We had already discussed both, the packet dropping algorithms and packet scheduling algorithms in our previous discussions.

And, our third important component of a QoS router is packet classification. Packet classification will look at the attributes of the packets and then will classify the packets appropriately into individual flows which can then be queued. So, if you really see the model of a QoS router, it will look something like this.

(Refer Slide Time: 45:08)



So, there is a control plane and there is a data plane. So, in control plane, you have routing agent and you have admission control and admission control and signaling like RSVP. These are all elements of the control planes. Now, when the packet comes at the, if the packet goes through a classifier and classifier then puts the packets into individual application flows and then you go for packet scheduler.

So, the admission control will configure the packet scheduler and also rule base will configure the packet scheduler, the classification rule engine where you need to define what is the definition of your flow. Your flow could be based upon a pure applications or the flow could be an entity. If the flow is an entity, then the packet classifier will look at the source IP address only and all the packets which bears the source IP address of this particular entity, they will be put into one flow. If you are looking at an individual TCP application; then you will look at the source IP address, destination IP address, source port number, destination port number. This will identify a particular TCP flow and you will classify the packet into a particular TCP flow. So, you will put them into different queues.

So actually, you will look at packet attributes and depending upon how your packet classifier has been configured through a rule engine, you will classify the packets into individual flows. Before actually, you do this in the data plane as we have already discussed that before any source can start transmitting its packet, it has to send the signaling RSVP signaling messages which contain the traffic specifications and the reservation specifications.

The admission control through an admission control algorithm will determine whether the flow can be accepted or not and if the flow can be accepted, then the admission control algorithm will configure the scheduler accordingly. Set up the virtual circuit path and then allow the flow to start transmitting the packets. When the packets come, now the packets pass through the data plane, these are the data packets; therefore, they will undergo the packet classification, scheduling and switching. So, these are the 3 primary engines in the data plane of any particular router.

Now, what we will do now in the next is to understand how the signaling for the call set up takes place in the int serv paradigm which is what is called as the resource reservation protocol or RSVP. As far as this individual implementation mechanism like packet scheduling or classifications or various admission control algorithms or token bucket regulations, we have already discussed these individual components earlier in our previous lectures.

We will now discuss the resource reservation protocol and that will complete our framework, our description of the framework for the int serv model; we will then see what are the difficulties or challenges that were there in with this int serv model for their practical implementations and therefore another model to provide quality of service guarantees at a coarse granularity level that is the differentiated services framework must propose. So, with this we will conclude today's lecture.

(Refer Slide Time: 49:55)

