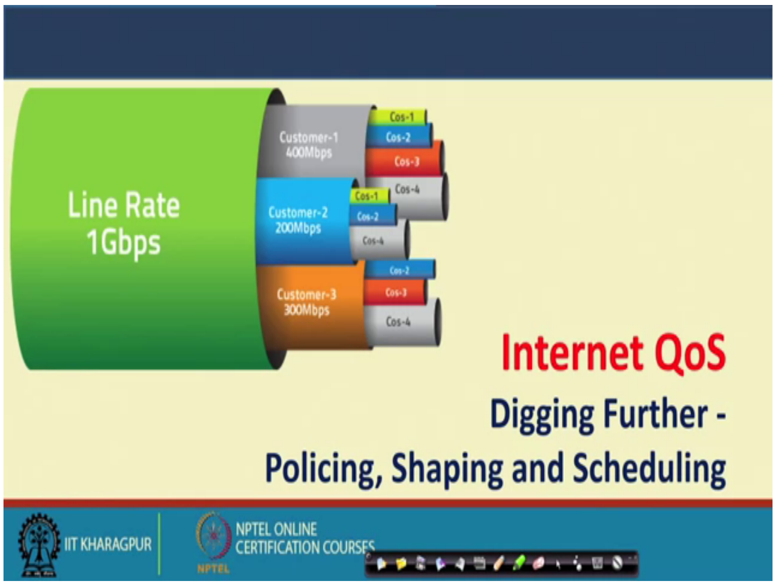
Computer Networks and Internet Protocol

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

Lecture - 33
Internet QoS - III (Traffic Policing and Traffic Shaping)

Welcome back to the course on Computer Network and Internet Protocol. So, we are discussing about internet quality of service in details.

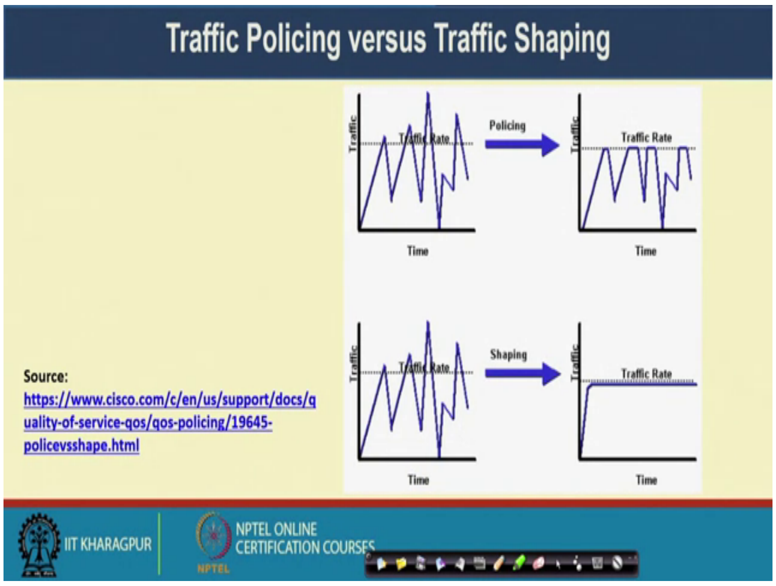
(Refer Slide Time: 00:26)



Internet QoS
Digging Further - Policing, Shaping and Scheduling

So, in the last class, we have looked into the broad internet QoS architecture. Today we will look further into 3 important aspects of internet quality of service traffic policing, traffic shaping and traffic scheduling.

(Refer Slide Time: 00:42)



Traffic Policing versus Traffic Shaping

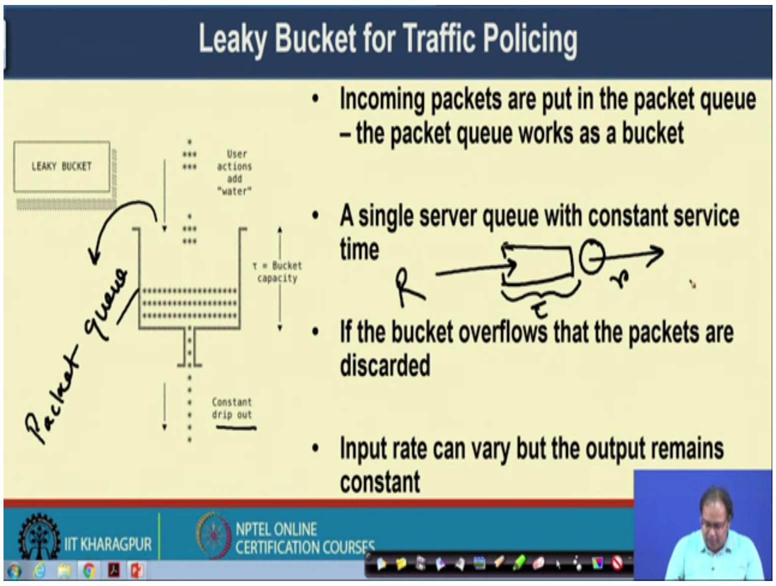
Source: https://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-policing/19645-policevsshape.html

So, let us start with traffic policing and traffic shaping. So, in the last class we were discussing about the difference between traffic policing and the traffic shaping. So, what we have seen in case of traffic policing, you just look into the peaks which are violating the required quality of service and if they are violating you simply cut them out and drop those packets. Whereas, in case of traffic shaping what we are doing that to apply traffic shaping, you actually smooth out this entire this entire rate which is there.

So, in general in a network whatever mechanism we are apply for ensuring traffic shaping or traffic policing we actually apply similar kind of mechanism. So, in a typical network having perfect guarantee of a constant bit rate is very difficult. Because the packets are going via multiple routers one after another and that is why we do certain level of approximation and that approximation actually combines both traffic policing and traffic shaping all together.

So, the idea is that what you do that you design a mechanism such that your output rate will get regulated and at the same time the additional packets which are there which will not be able to confirm to the quality of service policies that will get violated. Now to solve this let us look into the mechanism that we can apply here.

(Refer Slide Time: 02:23)



Leaky Bucket for Traffic Policing

- Incoming packets are put in the packet queue – the packet queue works as a bucket
- A single server queue with constant service time
- If the bucket overflows that the packets are discarded
- Input rate can vary but the output remains constant

So, the first mechanism that we are going to talk about is leaky bucket algorithm. So, although in the slides I have mentioned that leaky bucket for traffic policing, but as I have mentioned the ultimate mechanism is a combination of traffic policing and traffic shaping. So, this leaky bucket mechanism by idea it is very simple, the idea is that you have a bucket just what you are seeing in the example.

So, you have a bucket and you have a hole here. Now if you just think of like pouring water in that bucket what will happen like from that hole based on the size of the hole you will get the output at a constant rate. So, you will get a constant drip out and if you put more water rather than the capacity of the bucket then the additional water we will get overflowed from the bucket.

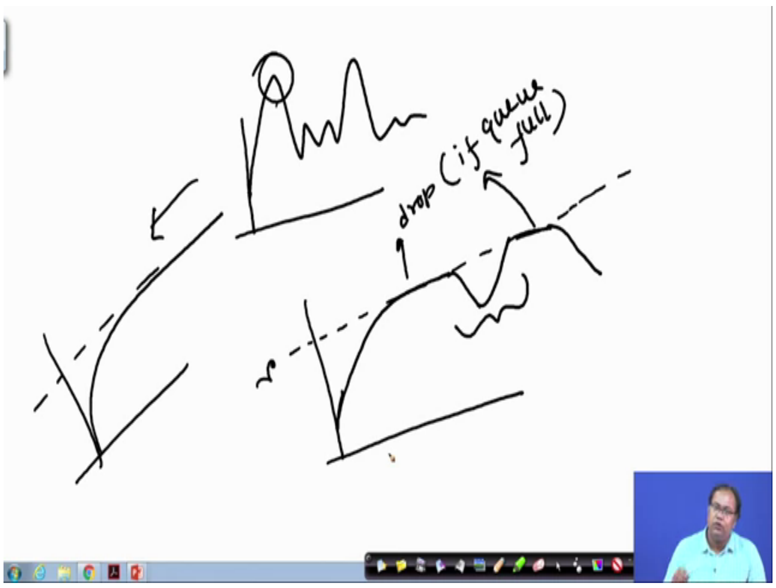
So, that way the capacity of the bucket is actually confirming the requirement for traffic policer, like if you are exceeding this much of capacity at a time then you drop out or you delete drop the additional packets additional data packets which are there. And this size of this hole it will actually trigger that at what constant rate you will send a packet to the output queue. So, that is the idea of leaky bucket algorithm. So, incoming packets are put in the packet queue. So, this particular queue we call it as a packet queue.

So, incoming packets are put at the packet queue. So, this packet queue works like a bucket and a single server queue with constant service time that particular server serves the queue. So, here in this queue so, I am just drawing it in the queuing notation. So, you have a single server queue and this single server queue always output at a constant rate r.

So; that means, whatever be the packet in that particular queue, then you will take packet out of the queue at a rate of r packets per second, say I am assuming that all packets are same. Otherwise you can go for bit per second that how many bits are coming in the queue and how many bits you are taking out of the queue. So, you are getting data at some rate say capital R and packets are getting input in a queue and we are using a constant size queue, say the capacity of the queue as we have mentioned here is tau. So, you can hold as much as many that much of packet in the queue and it is serve at a rate of r.

So, at the output you will always get a constant rate service at r packets per set or I will better say it is the maximum thing like you will get maximum at r packets per second if there is no input data in the queue; obviously, you will not get anything at the output. So, it sometime drop from r otherwise you will get at a peak rate of r packet per second and if your input gets more than tau if you are trying to push more than tau packets then it will get dropped. So, this part where we are taking it out at a constant rate it is applying traffic shaping and this size it is applying traffic policing.

(Refer Slide Time: 06:11)

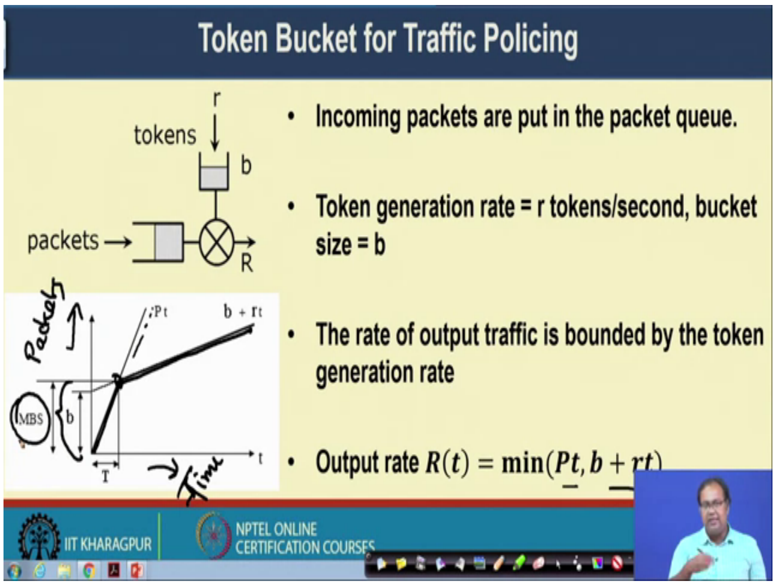


So, the diagram would be something like this that you have an input of something at say some peak rate and whenever you are getting the output the output is having at say this is my rate r, if this is my rate r then you are getting it at a rate if there is some additional packet here due to this peak those additional packet gets dropped if the queue becomes full.

So, if queue full, then those additional packet gets dropped otherwise those packets are transmitted at a rate and whenever there is a drop in the rate well this additional bandwidth will something get consumed here. But then you can see a little grip here due to this drop again whenever it is increasing it will increase and, but it will always maintain the maximum of this constant rate r. And then if whenever there is a drop it will again drop and here from here also if the queue becomes full you can experience certain packet drops.

Now, this diagram you can see that it is little violating from the diagram of traffic shaping that we have shown earlier the ideal diagram of traffic shaping was something like this that it will always maintain a constant rate, but ensuring that in a typical network is difficult because your output rate also depends on your incoming rate, like the rate at which the application is generating packet. Obviously, if the application is not generating any more packets, we will not be able to serve those packets and this dips in the rate are actually because the application is not generating any further data. So, that is why you are observing or drop in the rate, but if otherwise the application is generating rate you will get at that particular rate. So, and that is the case that is the output case. So, that way we are actually applying traffic shaping and traffic policing together with the help of this leaky bucket algorithm.

(Refer Slide Time: 08:22)



Token Bucket for Traffic Policing

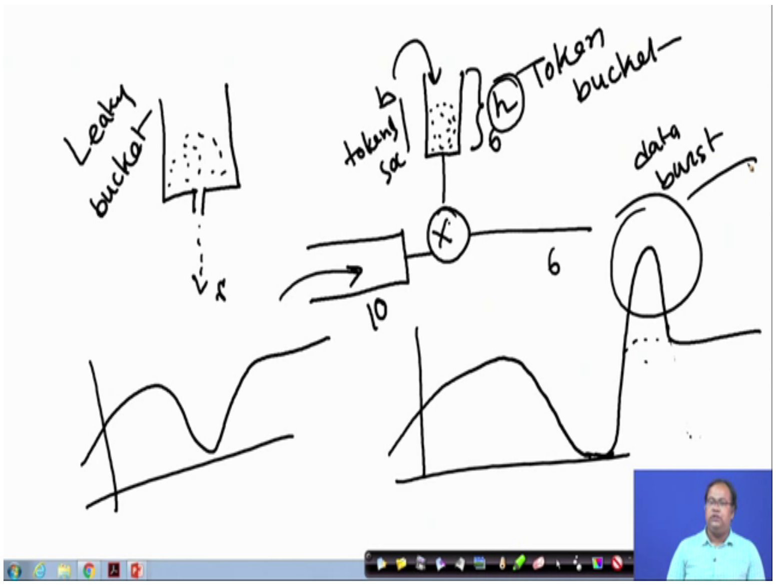
- Incoming packets are put in the packet queue.
- Token generation rate = r tokens/second, bucket size = b
- The rate of output traffic is bounded by the token generation rate
- Output rate $R(t) = \min(Pt, b + rt)$

We will go for another algorithm for traffic policing and shaping it is called a token bucket algorithm. So, the token bucket algorithm the idea is little interesting and it support something called traffic burst.

So, the idea of this token bucket algorithm is something like this you have a bucket a token bucket inside the token bucket. you are inserting the tokens at a rate of r and this token bucket also have a size of b. Now apart from this token bucket you have another packet queue in the packet queue whenever whatever packet you are receiving you are putting that packet in the packet queue. Now you have the scheduler, what the scheduler will do?

If there is a token in the bucket it will take out the token take a packet and send them out. Now these tokens are kind of logical token it is not a kind of physical entity. So, the idea of the token bucket is that you regulate the traffic in such a way; so, that if there is available tokens in the token bucket, then only you will be able to send a packet at the output. Now what is the difference between token bucket and leaky bucket, let us try to visualize that.

(Refer Slide Time: 09:38)



So, in case of leaky bucket what was happening, you had this leaky bucket and a small hole in this leaky bucket. So, whatever packet you are getting here you are taking it output at a constant rate. Now in that case if there is a no at a constant rate r in that case if there is no packet then there would be no output.

In case of leaky bucket what is happening, you have that is in leaky bucket in case of this is for leaky bucket, in case of token bucket you have this token bucket you have inserting tokens at a say at a rate token generation rate of say b I am just putting some arbitrary parameter at a token generation rate b and it has some size say h. And you have a packet buffer where the incoming packets are weighted keeping weighted and then this server it actually multiplex the token bucket and the leaky bucket all together. The idea here is that if there is no packet here if there is no incoming packet here, then the tokens are getting added right.

So, if there is no incoming packet then the token is getting added. So, the moment you are getting a packet now you say at instants of time you are getting 10 packets and during that time, you had some 6 tokens in the token bucket. In that case what can you do? You can immediately transfer these 6 packets in the output rather than sending it one by one. So, that is the difference from the leaky bucket and the token bucket, in case of leaky bucket you have to always send the packets one by one ok.

So, there is no such concept of traffic bars that even if you have not utilized a bandwidth previously, you can utilize it right now and send the bar stops traffic at a moment. Here in case of token bucket it is supporting that in case of token bucket there is a maximum burst length. So, this is the maximum burst length so the maximum amount of tokens that can be generated in that token bucket. So, you can get a kind of burst traffic.

So, burst traffic means say you are making a say regulation of traffic then say, the packets are not there. So, you are observing and drop and during that time the tokens are getting inserted the logical tokens are getting inserted the moment you got some new packet immediately. You can send a burst of packets altogether and then again can say after sending these burst of packets you can start doing the regulation based on the token generation rate. So, you are generating that token at b tokens per second. So, that is the difference between token bucket and leaky bucket.

In case of leaky bucket if I draw the same graph it will look something like this. So, this burst will not come here. So, this kind of peaks we are allowing in case of token bucket algorithm which we call as the data burst. So, that is the difference between token bucket and leaky bucket. So, the idea of token bucket came into practice because sometime it happens that you are not utilizing the bandwidth.

If you are not utilizing the bandwidth then whenever you are getting some additional data that may be of higher priority to you. So, rather than trying to shape it immediately send the entire data to the receiver side. So, for say kind of buffered video streaming, this kind of architecture is sometimes very useful because you will send more data to the play buffer, if you send more data to the play buffer it will get sufficient data and a video player can render for the data with the available data in the buffer.

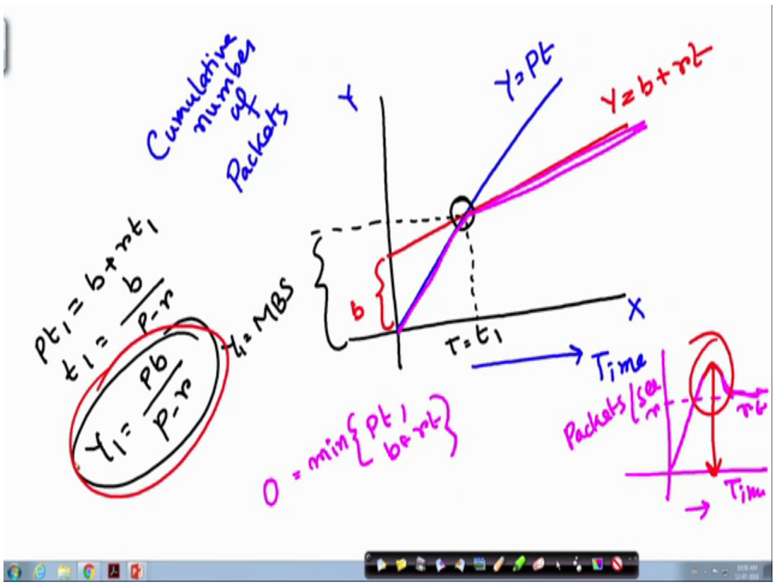
So, coming back to this concept of token bucket algorithm so, as I am mentioning that incoming packets are put in the packet queue; say I assume that the token generation rate is r tokens per second at that rate the tokens are getting generated remember that is token set of actual token. So, we are just you can think of in the implementation side in the program you have implemented the token bucket and you are observing that whether there is a token there or the not. If there is a token then you sending the packet to the output queue.

So, the token generation rate is are tokens per second and the bucket size is b. Now the rate of the output traffic it is bounded by the token generation rate you will not be in generally we will not be able to surpass that. But if you have more data immediately you can send it further. So, the output rate you can characterize it in this way say, Pt it denotes your incoming packet rate. So, this line Pt it denotes your incoming packet rate and this b plus rt it denotes your token generation rate ok.

So, if that is the case then your output rate will be minimum of Pt and b plus rt. So, your output rate will be when your incoming packet rate is less than this b plus rt, you can use that whenever it becomes at this cross section it overshoots b plus rt then you send the data at a rate of b plus rt. So, that actually gives you the output rate.

Now here this gives you this b is the initial token bucket size and this cross point and at this packet side. So, this side I am showing we are showing packets and this side we are showing time. So, at the packet side this particular length will give you the maximum burst sizes.

(Refer Slide Time: 16:35)



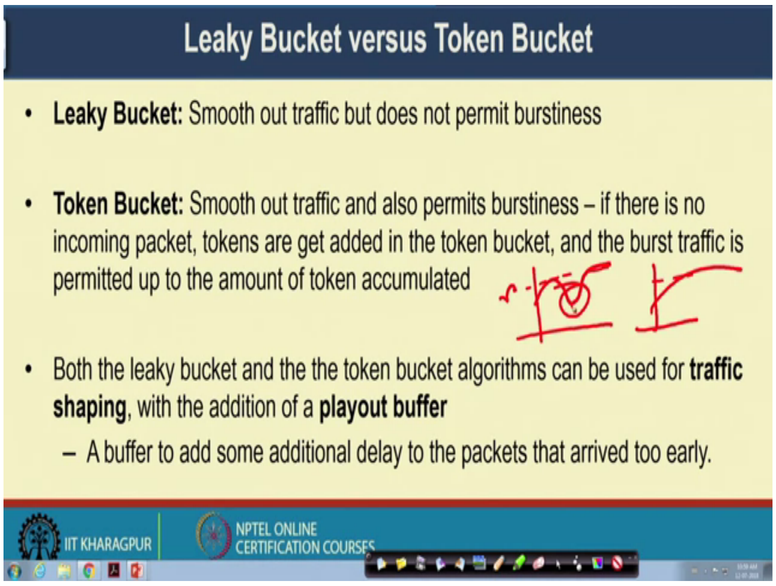
Now let us try to estimate the maximum burst size. So, we had Pt as the incoming packet rate. So, we can just write it as Y equal to Pt at the Y axis. So, the Y axis is denoting packets that have been received and X axis which is denoting time and this is that cumulative number of packets that you are receiving with respect to time. Let us write it cumulative number of packets you are receiving with respect to time. So, this is Y equal to Pt. Now with this let us have the rate for this broken bucket. So, this red line gives you the token bucket. So, this is the bucket size b and then and then so, the notation that we used earlier that b plus rt.

So, Y equal to b plus rt. So, you are inserting token at a rate of r. So, your cumulative number of token will be rt. So, initially you have be number of token initially we are assuming that the token bucket is full from that point we are starting after that you are inserting token at a rate of rt. So, you are getting it at a rate of the cumulative rate is Y equal to b plus rt.

Now this particular point the cross section at time T it will give you. So, this is equal to say T equal to t 1 this will give you the maximum burst size. Now let us try to estimate this maximum burst size, why this will give you maximum burst size at this point you can get the maximum output data. So, to find it out what we can do we can find out this cross section.

So, just at point t 1, we can write down Pt 1 equal to b plus rt 1 at this cross section point from there you can find out t 1 equal to b by P minus r. Now putting this value of t at say Y, you can put it in any of this equation you can find out say if I write it as Y 1 equal to MBS Y 1 will be Pb by P minus r this particular quantity. So, this will give you the maximum burst size that can happen in case of a token bucket algorithm.

So, with this estimation, let us go back to our discussion. So, that way this kind of token bucket algorithm in contrast to leaky bucket it is supporting burstiness well. (Refer Slide Time: 20:07)



Leaky Bucket versus Token Bucket

- **Leaky Bucket:** Smooth out traffic but does not permit burstiness
- **Token Bucket:** Smooth out traffic and also permits burstiness – if there is no incoming packet, tokens are get added in the token bucket, and the burst traffic is permitted up to the amount of token accumulated
- Both the leaky bucket and the the token bucket algorithms can be used for **traffic shaping**, with the addition of a **playout buffer**
  - A buffer to add some additional delay to the packets that arrived too early.

So, here is the difference between the leaky bucket for this is token bucket algorithm. So, the leaky bucket algorithm it smooth out traffic, but it does not does not permit burstiness. Whereas, token bucket it smooth out traffic and also permit burstiness if there is no incoming packet tokens are get added to the in the token bucket and the bus traffic is permitted up to the amount of token that has been accumulated.

So, that particular calculation we did earlier. So, we actually calculated the amount of token that can get accumulated up to this point because after that it will your output rate as we have seen. So, if we draw the output rate your output rate will follow this line. So, whenever your Pt is less than b plus rt. So, your output O it will be equal to minimum of Pt and b plus rt.

So, during that time you did not have sufficient amount of packet to take it out. So, that is why those at that time those tokens are getting accumulated here accumulated in the token bucket. And at this point you have reached at the peak of the burst and then you have transfer the data at the constant data.

So, if I just put it in terms of amount of so, here we have shown the cumulative number of packets, if in the y axis if I show packets per second and in this type time. So, the graph will be something like this, whenever you are getting the incoming packet you will have it then there is this burst after the burst you are you will send it at a rate of this rt. So, here you will send at a rate of rt this was your token generation rate r you will send it at a rate of rt and here you got this burst. So, this is my burst size that we have estimated here.

Now, as you are mentioning that both the leaky bucket and the token bucket algorithms can be used for traffic shaping. And so, the difference that I have mentioned earlier that with this leaky bucket and token bucket what happens that, sometime your rate can go deep although you are expecting the average rate r. But if the application is generating less amount of data, it can go at the lower side in that case for this data what you can do possibly if you want a complete smooth at rate of the original figure that we shown initially. If you want data at that particular depth what you can do that you can add an additional playout buffer.

So, playout buffer is an additional buffer, which can be added in front of your traffic shaper. So, the idea of the playout buffer is to introduce additional delay to the packets which come first. So, here you had this deep because you had certain packets which came faster.

So, you can introduce some delay to additional delay to those packets and to this additional delay you can experience something more similar to that. For some very strict application, we sometime apply this kind of layout buffer to introduce the additional delay.

So, this is all about leaky bucket and token bucket algorithm in details for ensuring traffic shaping and traffic policing. In the next class we will look into the different kind of traffic scheduling algorithms which are there by applying different type of queuing mechanisms, like priority queuing, weighted fair queuing, custom queuing and so on so.

Thank you all for attending this class.