# Broadband Networks

**Prof. Karandikar**

**Electrical Engineering Department**

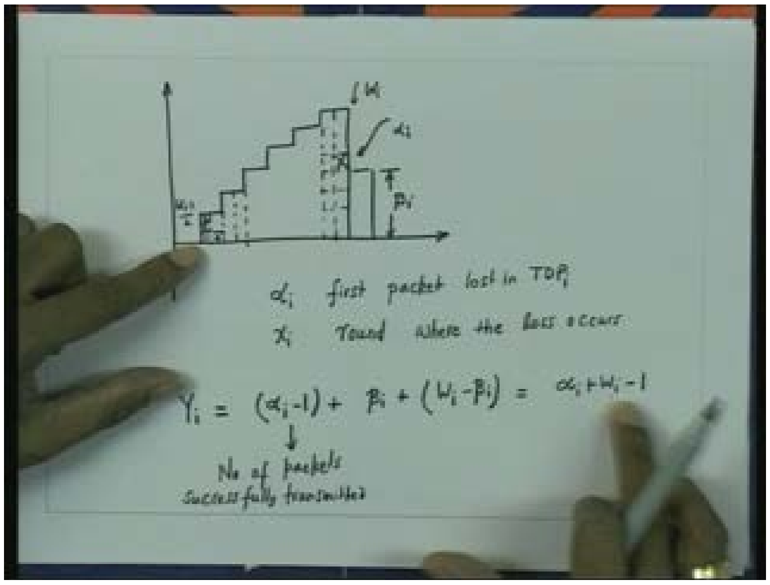
**Indian Institute of Technology, Bombay**

**Lecture - 20**

**Buffer Management**

So, in the previous lectures we were trying to determine the number of packets that were transmitted in a triple duplicate period and the length of the triple duplicate period. So, let us continue with determining an expression for the number of packets that have been transmitted in a triple duplicate period and also the length of the triple duplicate period.

(Refer Slide Time: 1:46)



$\alpha_i$ first packet lost in $TDP_i$

$X_i$ round where the loss occurs

$Y_i = (\alpha_i - 1) + \beta_i + (W_i - \beta_i) = \alpha_i + W_i - 1$

No of packets successfully transmitted

So, we had seen here that let us say that we had assumed that alpha $_i$ is the first packet which is lost in the triple duplicate period i and X $_i$ happens to be that round where the packet loss occurs. Now, note that even though this is alpha $_i$ is the packet which has been lost, the total packets that were sent in this window which were of the size W $_i$ because W $_i$ was the window size at this end and this loss will be detected only after a round trip time and therefore in the next round, the source would have transmitted some beta $_i$ packets also.

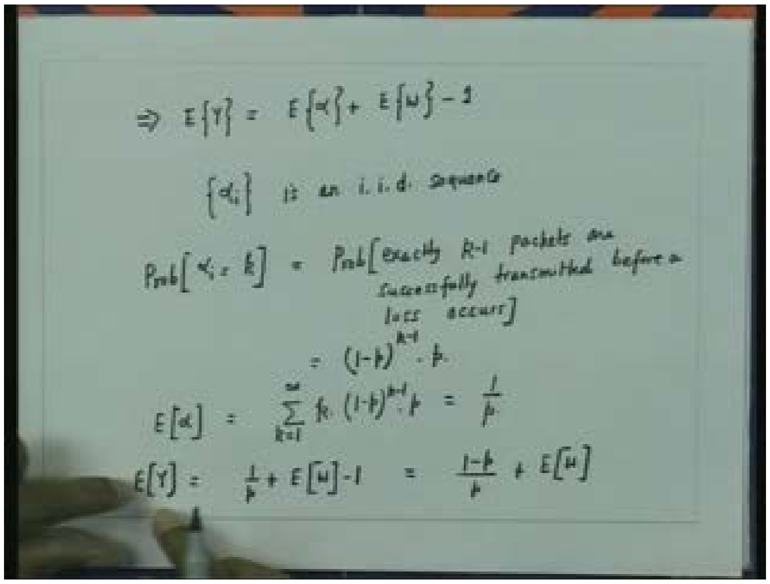
So therefore, if you try to determine how much is the number of packets which has been transmitted in the triple duplicate period i that would be given by alpha $_i$ minus 1. Now, this is the number of packets which have been successfully transmitted number of packets successfully transmitted because the loss was detected at the alpha $_i$ packet. So, alpha $_i$ minus 1 is the number of packets which have been successfully transmitted plus beta $_i$, this is beta $_i$ is the packet which will be transmitted in the round X $_i$ plus 1. So, beta $_i$ packet will be transmitted in the next round plus W $_i$ minus beta $_i$, these many number of packets which is W $_i$ is this, beta; these many number of packets would be further transmitted in this round and this would be then given by alpha $_i$ plus W $_i$ minus 1.

So, the total number of packets which were transmitted in a triple duplicate period of i is given by alpha $_i$ plus W $_i$ minus 1. Again, I would like to repeat; so, what we are saying is that many numbers of packets will be transmitted. Let us say packet number 1 is transmitted here, packet number 2, packet number 3, packet number 4, 5, 6; so such packets will be transmitted. The loss is detected at the alpha i'th packet.

So, alpha $_i$ minus 1 is the number of packets which have been successfully transmitted in this round. So, 1, 2, 3, 4, 5, 6, 7 and let us say alpha $_i$ could be 15. So, about 15, that is 14 packets have been successfully transmitted. Furthermore, these many number of packets also have been transmitted in this W $_i$ and how many of these packets are there? These packets are W $_i$ minus beta $_i$. What is beta $_i$? Beta $_i$ is the number of packets which have been successfully transmitted in this window only.

So therefore, the total number of packets which have been transmitted in a triple duplicate period of triple duplicate period i is given by alpha $_i$ plus W $_i$ minus 1. And hence, if you take the expectation this means that expected value of Y is given by expected value of alpha plus expected value of W minus 1.

(Refer Slide Time: 5:08)



$$\Rightarrow E\{Y\} = E\{\alpha\} + E\{W\} - 1$$

$\{\alpha_i\}$ is an i.i.d. sequence

$$\text{Prob}[\alpha_i = k] = \text{Prob}[\text{exactly } k-1 \text{ packets are successfully transmitted before a loss occurs}]$$

$$= (1-p)^{k-1} \cdot p$$

$$E[\alpha] = \sum_{k=1}^{\infty} k\,(1-p)^{k-1} \cdot p = \frac{1}{p}$$

$$E[Y] = \frac{1}{p} + E[W] - 1 = \frac{1-p}{p} + E[W]$$

Now, since the packet is lost in a round independent of the loss that occurred in the previous triple duplicate periods and also if this packet is lost; all the subsequent packets will also be lost. So, this sequence alpha $_i$ is essentially is an i.i.d sequence. So, alpha $_i$ which is the number of packets successfully transmitted or rather the number of packets successfully transmitted, alpha $_i$ minus 1, this is actually an i.i.d sequence - independent and identically distributed sequence.

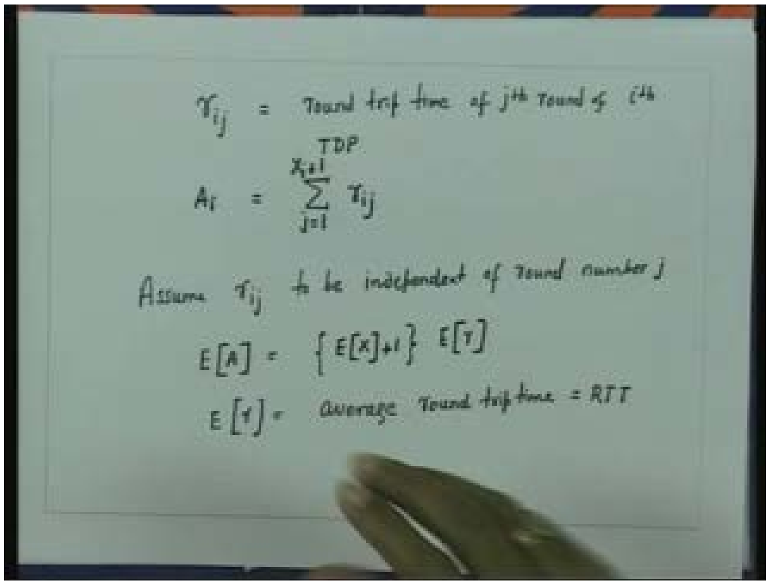
And, probability that alpha $_i$ is equal to k that means the loss is detected at the K’th packet is the same is the probability that exactly K minus 1 packets are successfully transmitted before a loss occurs. So, this is the same as the probability that exactly K minus 1 packets are successfully transmitted before a loss occurs.

Now, the probability of packet loss, we have assumed it to be p and we have also assumed that once the packet is lost, the probability of a packet loss is p and we have assumed that once a packet is lost, all the subsequent packets will also be lost. Now therefore, the probability that exactly K minus 1 packets are successfully transmitted will be given by 1 minus p raised to power K minus 1 into p; this will be the probability that exactly K minus 1 packets are successfully transmitted before a loss occurs.

Therefore, expected value of alpha is given by K is equal to 1 to infinity K (1 minus p) raised to power K minus 1 into p which is actually given by 1 by p. So, that means the expected value of alpha is inverse of the probability of packet loss which we have assumed it to be p. So, as a result if you substitute this expression here, then we get expected value of Y is given by 1 by p plus expected value of W minus 1 which is also equal to 1 minus p upon p plus expected value of W.

So, now in order to determine Y, we now have to determine what is the expected value of W that is what is the expected value of the window size and that is what we need to determine. So, let us see how we can determine the expected value of W.

(Refer Slide Time: 8:54)



$r_{ij}$ = round trip time of $j^{th}$ round of $i^{th}$ TDP

$$A_i = \sum_{j=1}^{X_i+1} r_{ij}$$

Assume $r_{ij}$ to be independent of round number $j$

$$E[A] = \{E[X]+1\}\, E[r]$$

$E[r]$ = average round trip time = RTT

Now, let us define r $_i$ r $_{i\,j}$ to be the round trip time of j’th round of i’th triple duplicate period. So, this is our i’th this is our i’th triple duplicate period is what we have assumed. So, we are saying that r $_{i\,j}$ is the round trip time of the j’th round. Note that the loss has occurred in the X i’th round. So therefore, A $_i$ which happens to be the duration of this triple duplicate period, A $_i$ A $_i$ would be then given by summation of j is equal to 1 to X $_i$ plus 1 r $_{i\,j}$. Why X $_i$ plus 1? Because the loss occurs in the X $_i$’th round and then there is one more round following that.

Now, we assume that this round trip time $r_{ij}$, they are independent of the round number j. We assumed it to be independent of round number j, so if that is independent; then it is clear that expected value of A will be given by expected value of X plus 1 into expected value of r. Now, expected value of r is something the average round trip time because r happens to be the round trip time. So, we call expected value of r to be the average round trip time and this we call it to be the round trip time RTT.

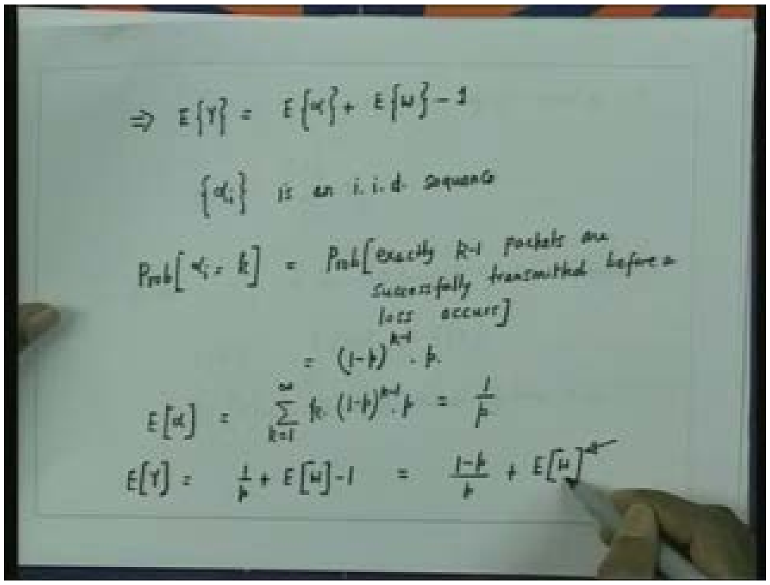
So now, basically, as you have seen, again let me repeat the steps. What we are actually trying to do is that we are trying to determine an expression for the TCP throughput in the presence of a random packet loss and we are assuming that the packet loss is detected due to the triple duplicate acknowledgment. The analysis can also be extended to the case when the packet loss is detected due to the receiver time out. But in this analysis we are assuming that the packet loss is detected due to the triple duplicate acknowledgment.

Now, the window evolution takes place something like this that every time an acknowledgment is received, the window size is increased by W by b where b is that the acknowledgment is cumulative and for every b packets, an acknowledgment is received for every b packet. And, in this simple case we have assumed it b to be equal to 2; so for every 2 packets, an acknowledgment comes. So, when all the packets belonging to a window have been acknowledged, the window size will increase by W by b into 1 by W. That is the window size will increase by 1 by b. So, in b rounds in b rounds the window size will increase by 1. So, that is how you have a staircase evolution of the window size.

Now, when the packet loss is detected due to the triple duplicate acknowledgment, the window size drops to half and again the new cycle starts. So, this period, we call it to be a triple duplicate period. Now, what we are trying to determine is that what is the period of this triple duplicate period and how many number of packets have been transmitted in this triple duplicate period. Now, it can be shown that this window evolutions $W_i$ happens to be a Markov regenerative process and therefore the TCP throughput can be given by expected value of number of packets transmitted in this triple duplicate period divide by the expected duration of the triple duplicate period.

So, right now what we are trying to do is that we are trying to determine how much is the expected number of packets successfully transmitted or number of packets transmitted during the triple duplicate period and how much is the duration of the triple duplicate period. So, that is how we are trying to determine and we have determined an expression that the duration of the triple duplicate period is given by this.

(Refer Slide Time: 14:09)



$$\Rightarrow E\{Y\} = E\{\alpha\} + E\{W\} - 1$$

$\{\alpha_i\}$ is an i.i.d. sequence

$$\text{Prob}[\alpha_i = k] = \text{Prob}[\text{exactly } k-1 \text{ packets are successfully transmitted before a loss occurs}]$$

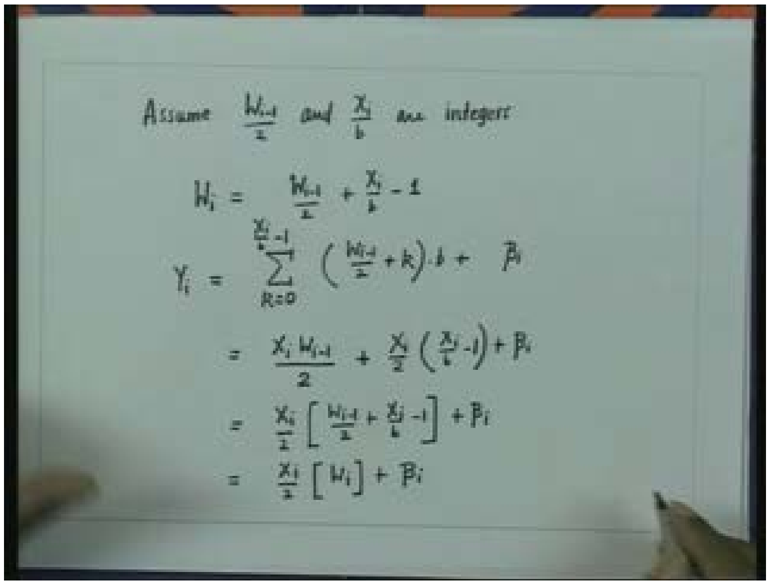
$$= (1-p)^{k-1} \cdot p$$

$$E[\alpha] = \sum_{k=1}^{\infty} k \cdot (1-p)^{k-1} \cdot p = \frac{1}{p}$$

$$E[Y] = \frac{1}{p} + E[W] - 1 = \frac{1-p}{p} + E[W]$$

Now, previously we had determined that expected value of Y is given by 1 minus p upon p plus expected value of w. So, we still need to determine this expression that is the expected value of W.

(Refer Slide Time: 14:27)



Assume $\frac{W_{i-1}}{2}$ and $\frac{X_i}{b}$ are integers

$$W_i = \frac{W_{i-1}}{2} + \frac{X_i}{b} - 1$$

$$Y_i = \sum_{k=0}^{\frac{X_i}{b}-1} \left(\frac{W_{i-1}}{2} + k\right) \cdot b + \beta_i$$

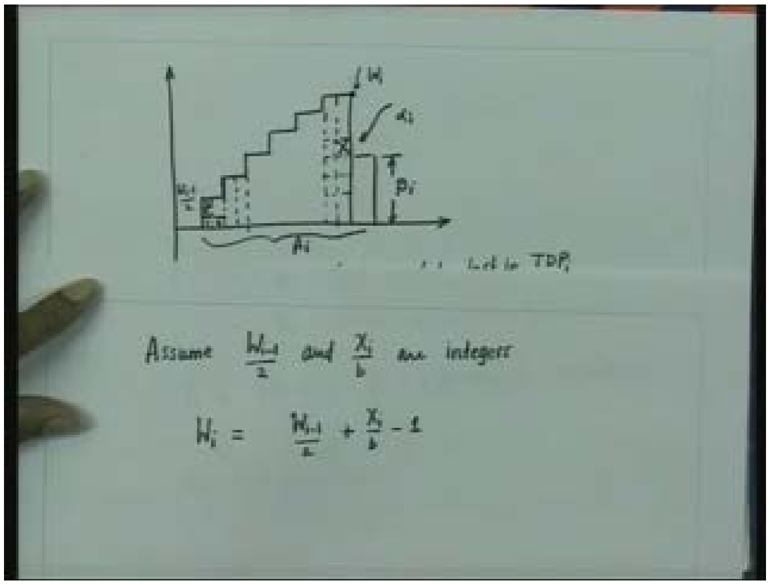
$$= \frac{X_i W_{i-1}}{2} + \frac{X_i}{2}\left(\frac{X_i}{b} - 1\right) + \beta_i$$

$$= \frac{X_i}{2}\left[\frac{W_{i-1}}{2} + \frac{X_i}{b} - 1\right] + \beta_i$$

$$= \frac{X_i}{2}\left[W_i\right] + \beta_i$$

Now, in order to determine that, make an assumption that assume that W $_i$ minus 1 by 2 and X $_i$ by b, they are integers, assume that they are integers; then it is very easy to see that W $_i$ is given by W $_i$ minus 1 by 2 plus X $_i$ by b minus 1. So, this will be clear from this expression.

(Refer Slide Time: 15:04)



Assume $\frac{W_{i-1}}{2}$ and $\frac{X_i}{b}$ are integers

$$W_i = \frac{W_{i-1}}{2} + \frac{X_i}{b} - 1$$

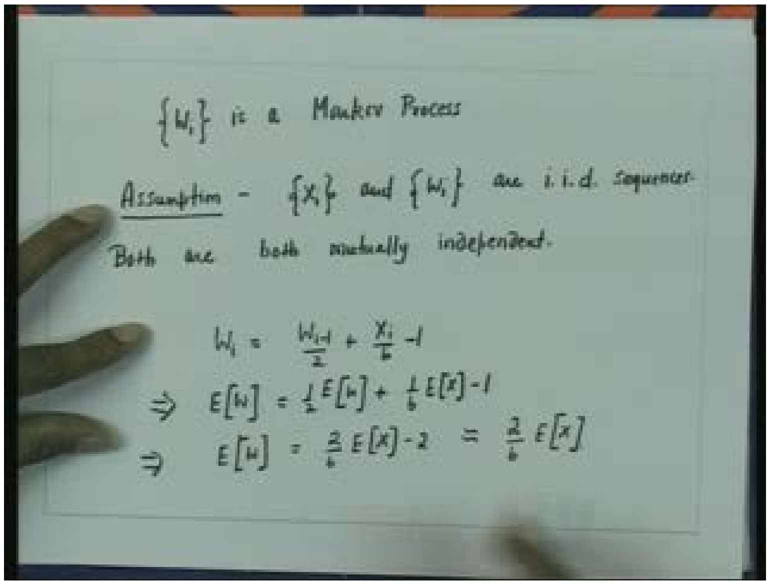
As you can see here, the initial window size was W $_i$ minus 1 by 2 W $_i$ minus 1 by 2 and so though, in this diagram b is equal to 2; so therefore, for every 2 rounds the window size increases by 1 and this is the X $_i$'th round. So therefore, the window size would have increased by W $_i$ minus 1 by 2 plus X $_i$ by b. Actually, the window size would have so, window size actually increased at the end of the round; so at the end of this round, window size would have increased by 1 here. But since the loss was detected in the X $_i$'th round, we need to only consider up to this. So therefore, we have to subtract minus 1 here.

And, how much is the number of packet that has been transmitted? Y $_i$ would be given by K is equal to 0 to X $_i$ by b summation W $_i$ minus 1 by 2 plus K into b plus beta $_i$. So, if you see here, again for this expression for k is equal to 0, total number packets which have been transmitted is W i minus 1 by 2 into b; this is what is the total number of packets which would have been transmitted. Now, the window size increases by 1. So, the total number of packets transmitted during this duration will be given by W $_i$ minus 1 by 2 plus K into b and so on upto Xi by b minus 1 plus beta $_i$. The beta $_i$ denotes the number of packets which were sent in the last round.

Now, this is given by X $_i$ into W $_i$ minus 1 by 2 plus X $_i$ by 2 X $_i$ by b minus 1 plus beta $_i$ by simplifying this expression which is also equal to X $_i$ by 2 W $_i$ minus 1 by 2 plus X $_i$ b minus 1 plus beta $_i$ which is actually equal to X $_i$ by 2 W $_i$ plus beta $_i$. Now, actually the W $_i$ that is the window evolution is a Markov process because the window size at the end of the next triple duplicate period actually depends upon the window size which was there in the previous triple duplicate acknowledgment period.

So, strictly speaking W $_i$ is a Markov process but for simplicity of analysis we assume that W $_i$ is the sequence of W $_i$ is an independent and identically distributed random variables.

(Refer Slide Time: 18:33)



$\{W_i\}$ is a Markov Process

<u>Assumption</u> - $\{X_i\}$ and $\{W_i\}$ are i.i.d. sequences.

Both are both mutually independent.

$$W_i = \frac{W_{i-1}}{2} + \frac{X_i}{b} - 1$$

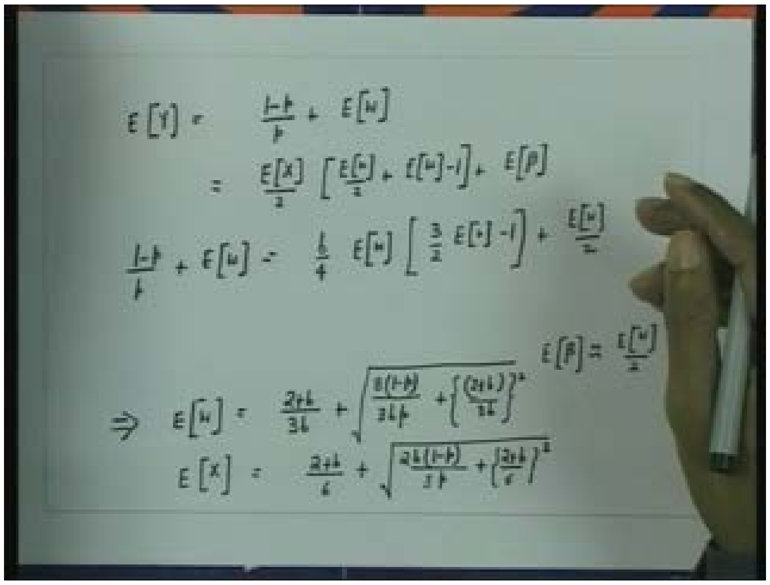
$$\Rightarrow E[W] = \frac{1}{2}E[W] + \frac{1}{b}E[X] - 1$$

$$\Rightarrow E[W] = \frac{2}{b}E[X] - 2 \approx \frac{2}{b}E[X]$$

So, strictly speaking W $_i$ is a Markov process but we make an assumption that both X $_i$ sequence X $_i$ and W $_i$ are independent identically distributed i.i.d sequences and both are mutually independent. So, we make this assumption that both are mutually independent. Now, note that we had assumed that W $_i$ is actually equal to W $_i$ minus 1 by 2 plus X $_i$ by b minus 1 which will mean that expected value of W is actually equal to 1 by 2, expected value of W plus 1 by b, expected value of X minus 1 and this can be given by expected value of W will be equal to 2 by b expected value of X minus 2 which we will approximate it to 2 by b expected value of X.

Now, we have already determine that expected value of Y, we had already determine that expected value of Y is 1 minus p upon p plus expected value of W.

(Refer Slide Time: 20:32)



$$E[Y] = \frac{1-p}{p} + E[W]$$

$$= \frac{E[X]}{2}\left[\frac{E[W]}{2} + E[W] - 1\right] + E[\beta]$$

$$\frac{1-p}{p} + E[W] = \frac{b}{4} E[W]\left[\frac{3}{2}E[W] - 1\right] + \frac{E[W]}{2}$$

$$E[\beta] = \frac{E[W]}{2}$$

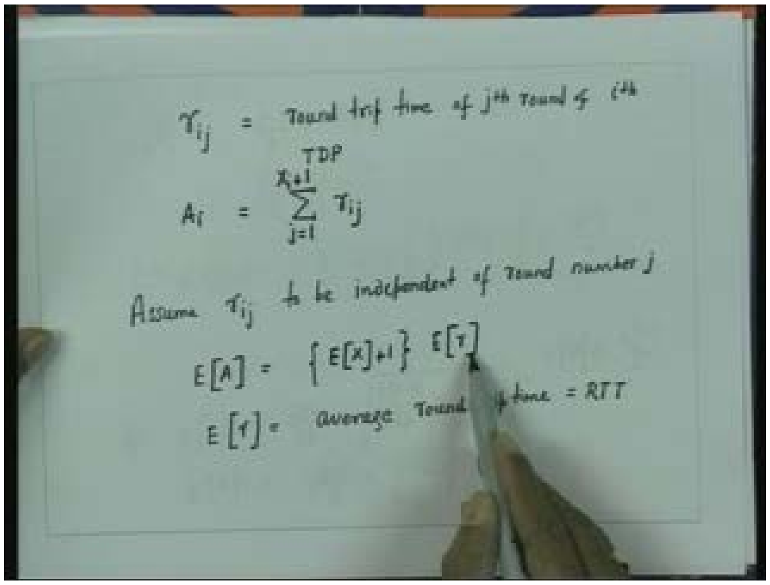
$$\Rightarrow E[W] = \frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2}$$

$$E[X] = \frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2}$$

So, we will write this - expected value of Y is given by 1 minus p upon p plus expected value of W which is also given by expected value of X by 2 into expected value of W by 2 plus expected value of W minus 1 plus expected value of beta and this comes from the previous expression, so therefore 1 minus p up on p plus expected value of W is given by… now, in place of expected value of expected value of X by 2, we write this expression here.

So, this will become therefore as b by 4 expected value of W into 3 by 2 expected value of W minus 1 plus; now this beta we assumed it to be uniformly distributed between 1 and W, so we will get an expression as expected value of W by 2, this is an approximate one. So, we assume that expected value of W is approximately equal to expected value of W by 2 and if you solve this equation, then it turns out that expected value of W can be given by 2 plus b upon 3 b plus 8 into 1 minus p upon 3 bp plus 2 plus b upon 3 b square and from this therefore, the expected value of X will be given by 2 plus b upon 6 plus 2 b upon 1 minus p upon 3 p plus 2 plus b upon 6 square. Just by this expression that expected value of W is given by 2 by b expected value of X and from this expected value of A was given by expected value of X plus 1 into round trip time.

(Refer Slide Time: 23:30)



$r_{ij}$ = round trip time of $j^{th}$ round of $i^{th}$ TDP

$$A_i = \sum_{j=1}^{X_i+1} r_{ij}$$

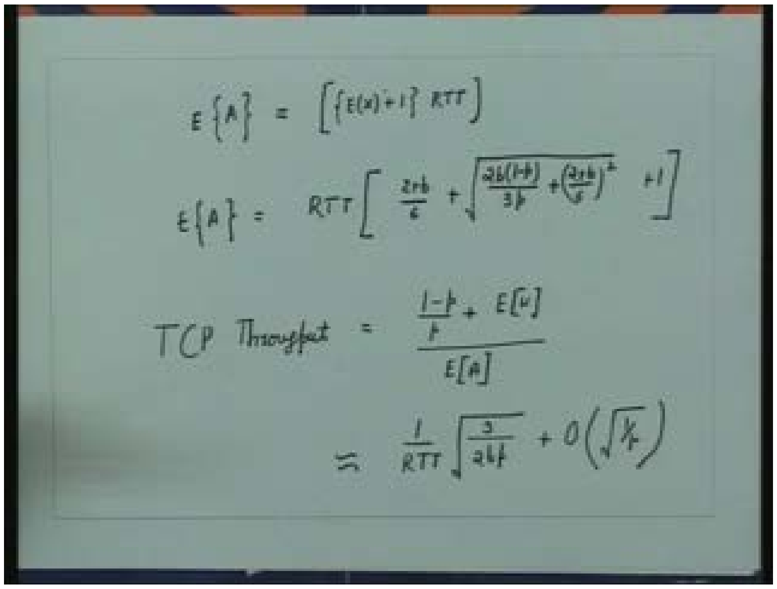
Assume $r_{ij}$ to be independent of round number $j$

$$E[A] = \{E[X]+1\} E[r]$$

$E[r]$ = average round trip time = RTT

Therefore from this, the expected value of A can be computed and expected value of A is as you know expected value of X plus 1 into the round trip time.

(Refer Slide Time: 23:40)



$$E\{A\} = [\{E(X)+1\} RTT]$$

$$E\{A\} = RTT\left[\frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2} + 1\right]$$

$$\text{TCP Throughput} = \frac{\frac{1-p}{p} + E[W]}{E[A]}$$

$$\approx \frac{1}{RTT}\sqrt{\frac{3}{2bp}} + O\left(\sqrt{1/p}\right)$$

So, this means that expected value of A is given by round trip times 2 plus b upon 6 plus square root of 2 b into 1 minus p upon 3 p plus 2 plus b upon 6 square plus 1. So therefore, the TCP throughput, an expression for TCP throughput will be given by 1 minus p upon p plus expected value of W divide by expected value of A. This is the expected number of packets which have been transmitted during the triple duplicate period divide by the expected value of the duration of the triple duplicate period. So, if you divide these 2 quantities, we get an expression for the TCP throughput.

Now, we have already determined an expression for expected value of W and expected value of A; if you put this expression, we can actually determine an exact expression for the TCP throughput. Now, an approximation one can be done and one can show that this can be approximated as 1 by round trip time into 3 upon 2 bp plus turns which are having order of 1 by root p turns. So, one can see from an expression that the TCP throughput is inversely proportional to the round trip time and also inversely proportional to the square root of the packet loss probability.

Now, this expression we had already derived under the steady state and by using a very simplified analysis in our previous lectures. But this analysis gives a more a rigorous analysis in terms of considering the triple duplicate periods to be a Markov regenerative process and then determine an expression for the number of packets which have been transmitted in a triple duplicate period and also the expected duration of the triple duplicate period.

So, the in conclusions what we have really seen is that the TCP throughput is inversely proportional to the round trip time and also inversely proportional to the square root of the packet loss probability. So, this brings us in conclusion our discussion on the resource allocation problem for the best effort networks.

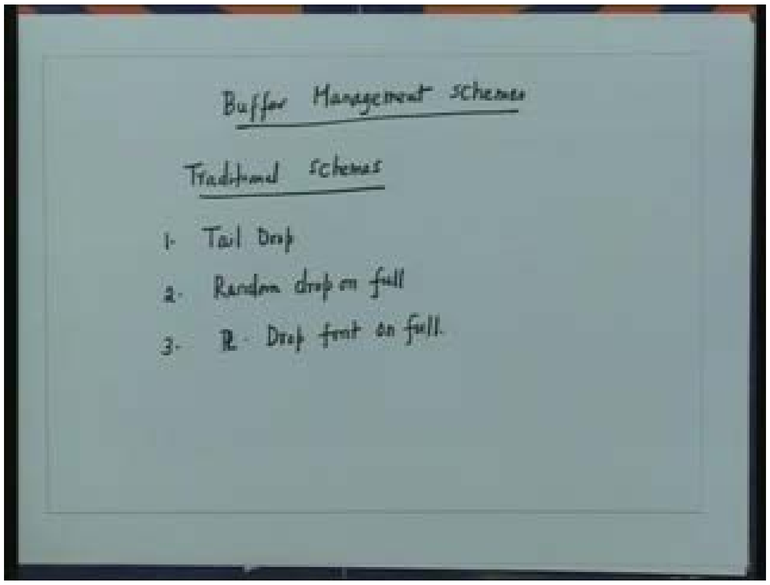
So, what we have really seen is that while in a network which provides quality of service guaranties, a traffic will normally be shaped by a rho sigma regulator and then the admission control and the scheduling algorithms guaranties the delay bounds and the minimum bandwidth to a rho sigma regulated traffic. For the best effort traffic, the resource, for the best effort internet, the resource allocation is done by a combination of the congestion control mechanisms plus a fair scheduling algorithm which are implemented at the core routers. So, this is done by a combination of these 2 things.

There is a one more thing which can be used to provide some amount of quality of service guaranties and that is the buffer management. There is a buffer management schemes which can be deployed at the routers. So, we will shortly discuss those buffer management schemes which are deployed at the routers. So, what we have however seen is that the TCP throughput is inversely proportional to the square root of the round trip time; so therefore, there is some amount of unfairness in terms of the TCP throughputs that if a particular router, the flows which are having a shorter round trip times, they will ramp up their windows faster and therefore will likely to have a higher steady state TCP throughput than the flows which have a longer round trip times.

So as a result, some kind of unfairness gets introduced at routers. So, if you want to provide fairness to the TCP flows, then it is necessary that we implement some kind of fair scheduling algorithms at the routers. So, this is one aspect of the congestion control in the internet. However, as you have already seen that if there are flows which have the same round trip time, then the additive increase and multiplicative decrease leads to proportionally fair proportionally fairness in a typical packet switched networks.

So, in a best effort networks, we are trying to provide the quality of service guaranties by a good congestion control schemes, fair scheduling algorithms and then thirdly, another important thing is the buffer management scheme. So, let us now look at what kind of buffer management schemes can be used in a core internet router to provide the quality of service guaranties. So, now 3 kinds of buffer management schemes are possible to implement in a router.

(Refer Slide Time: 29:26)



Buffer Management schemes

Traditional schemes

1. Tail Drop
2. Random drop on full
3. ~~R~~ Drop front on full.

So, buffer management schemes: so, what are the traditional schemes? So, there are one traditional scheme and which is common and is like tail drop. Now, in a tail drop, you drop the packet which has newly arrived when the buffer is full. So, this is very simple and most intuitive also that when the buffer is full, then you drop the arriving packet. Unfortunately, however, it turned out that this simple buffer management scheme is not sufficient or not suitable for effective resource allocation and congestion management in an internet and we will shortly see why it is so.

Now, the other scheme that could be there, other possible scheme could be there is that random drop on full. Now, in a random drop on full; when the queue is full, then you pick up a packet randomly from the queue and then drop it. What is the rational for doing so? The rational for doing so is that if you pick up a packet randomly from the queue, then the chances are very high that the packet from that flow will be picked up who has sent more number of packets in the queue and therefore the user who has transmitted more number of packets will be punished proportionately more that the users who are transmitting less packets. So, this is the logic or this is the rationale behind dropping a packet randomly while the buffer is full.

The third technique that could be there is drop from front drop from front on full. So, the drop from front on full, drops the packet from the front of the queue than instead of from the tail of the queue. It drops the packet from the front of the queue, the rational being that that if this packet has waited in the queue for a long enough time and has wasted the resources so much, then there is no point in transmitting this packets further down the downstream routers and therefore it may be better to drop the packets from the front of queue to make way for the recently arrived or newly arrived packets which are fresh. So, that is the reason or that is a rational behind dropping the packets from the front of the queue.

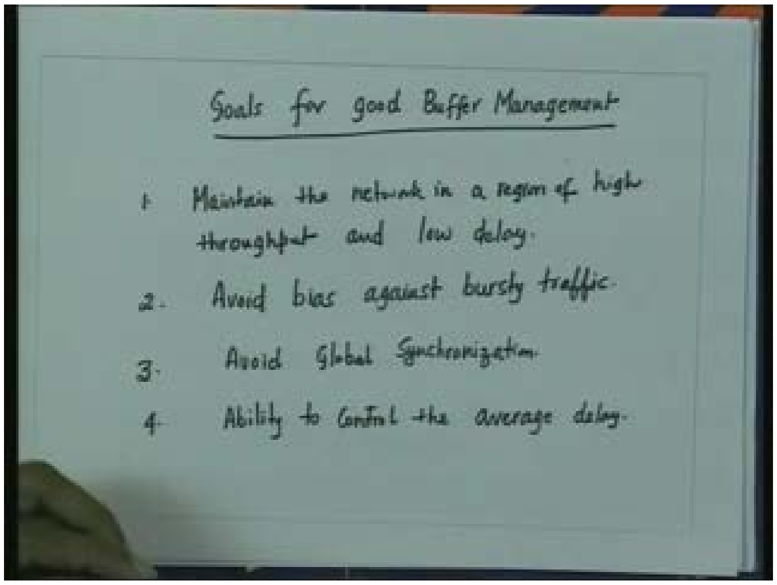
Now, most routers traditionally have implemented tail drop schemes. Tail drop is that when the buffer is full, you drop the arriving packet. There are several serious disadvantages of this tail drop scheme. One of the 2 important disadvantages of this tail drop scheme is that it leads to lockout and full queues.

By lockout means that it allows a single connection, it can allow a single connection to monopolize the queue space because what may happen is that irresponsible an irresponsible flow may transmit large number of packets into the queue. As a result, the queue may become full and if the queue is bit if queue is full, then if a responsible flow submits its packet and its find a queue to be full, then that packet is dropped.

As a result, nonresponsive flows may try to monopolize a queue space by transmitting large number of packets into the network. So therefore, it leads to what is called as lockout. Also the full queues, the queue can remain full for a long interval of time and as a result the average delay of the flow may increase. So, these 2 are the serious disadvantages of the schemes which are tail drop scheme. That is they drop the packets from the tail of the queue when the buffer is full.

Now, the random drop from the queue and the random drop on front, although they are difficult to implement but have certain advantages in the sense that they try to ensure some kind of fairness while dropping the packet. So, we will see that some variants of these schemes will be implemented as a good buffer management schemes. Now, towards designing of a good buffer management schemes, we need to then think of what are the goals of a good queue management or a buffer management policy. So, we will look at what are the goals of a good queue management policy.

(Refer Slide Time: 34:57)



Goals for good Buffer Management

1. Maintain the network in a region of high throughput and low delay.
2. Avoid bias against bursty traffic.
3. Avoid Global Synchronization.
4. Ability to Control the average delay.

So, the goals for good buffer management is that one thing is that we would like to maintain the network in a region of low delay and high throughput. So now, how can you maintain low delay? If the queue if the queue is having even the average queue size, if the average queue size is maintained low, then we can maintain low delay. But if you maintain the average queue size to be low by dropping large number of packets, then this will lead to lot of packet loss and also a low throughput.

So, our buffer management policy should be such that it should maintain the network in a region of high throughput that is high link utilization. Buffer should not become empty and at a same time, on an average the queue size should be kept low. Occasionally, the queue size may become high but on an average, the queue size should be kept low. So, average throughput should be high and average delay should be low. We have to define our buffer management policy in such a manner.

So, our first goal should be that it should maintain the network in a region of high throughput and low delay. Second is a buffer management policy should avoid bias against bursty traffic. Now, let me just tell you how the tail drops as a bias against bursty traffic. Suppose that there is a flow which has been transmitting consistently large number of packets and therefore the queue is likely to remain full for most of the time. Now, here is a flow which transmits occasional burst which it has not transmitted, suppose the flow has not transmitted any packet for a long enough time and suddenly now it transmit burst; now when you transmit burst, it may find that a queue is full and the tail drop scheme will drop all the packets from the burst when it arrives.

Now, this way the flow is likely to complain that it has not transmitted any packets for a long enough time and now it is transmitting a burst and it finds that the queue is essentially full. On other hand, while designing the theme of the statistical multiplexing bias packet switched networks, we are argued that such a multiplexing and switching scheme is best suited for the bursty traffic. Now, if that is so; then why it is happening that when you transmit a bursty traffic, it finds a queue to be full?

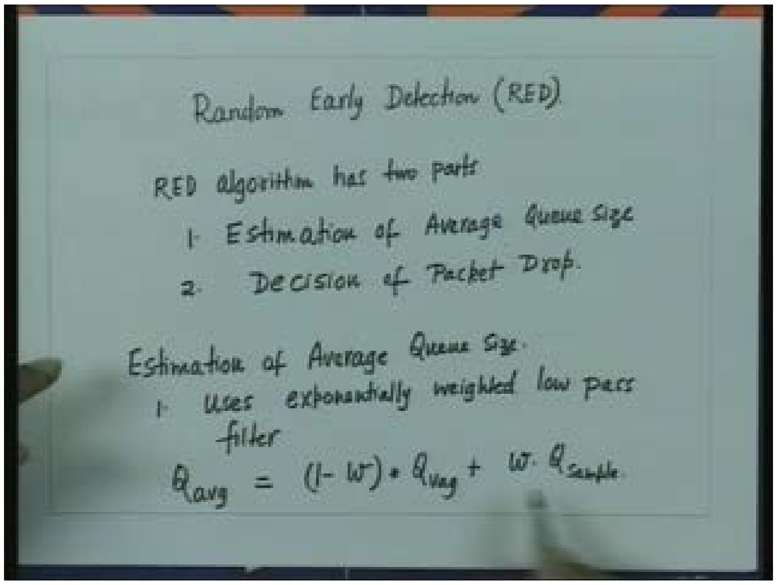
So obviously, in order to really satisfy our theme of a packet switching statistical multiplexing to be suitable for the bursty traffic, we have to design a buffer management schemes in such a way that it avoids bias against bursty traffic. Third thing that we should have is that we should avoid global synchronization. Tail drop actually leads to global synchronization. How it leads to global synchronization? Remember that there will be packets, there will be TCP flows which are sharing the queue which might have transmitted their flows, which might have started their flows at different times.

Now, when the packet loss occurs, all the flows will simultaneously reduce their window size to half of its previous window size and then will start increasing the flows and somehow they will come into now synchronization. So, we should avoid this global synchronization. So, our buffer management scheme should avoid global synchronization. We should also have an ability the buffer management scheme should also have an ability to control the average delay or the average queue size.

So, the average delay should be a tunable parameter, the average queue size should be a tunable parameters and it should be under the control of a network operator to control the queue size. So, your queue management scheme or your buffer management schemes should have an ability to control the average delay. So, this should be should be an ability to control.

So, what does it mean is that a buffer management scheme while it looked early very simple that drop the packet when the buffer is full, it is not so simple if you want to really achieve our goal of an effective resource allocation and a fair resource allocation in a best effort internet. So now, while looking at these goals, we can classify the buffer management schemes into several ways. So, let us see the classification of the buffer management schemes which can be done; the classification of buffer management schemes.

(Refer Slide Time: 40:29)



Random Early Detection (RED)

RED algorithm has two parts

1. Estimation of Average Queue size
2. Decision of Packet Drop.

Estimation of Average Queue Size.

1. Uses exponentially weighted low pass filter

$Q_{avg} = (1-W) \cdot Q_{avg} + W \cdot Q_{sample}$

Now, the classification, one classification can be done based on degree of aggregation. It is important whether all the flows have been aggregated into one single flow and they are sharing one queue and then you are employing a buffer management scheme or we have done the classification of the flows and divided the flows into several flows and then there are either separate queues or there are virtual queues to be handled. Obviously, if we have to do a flow based buffer management scheme, we will have to maintain large number of states and the complexity of buffer management will increase but we can ensure fairness in the buffer management schemes.

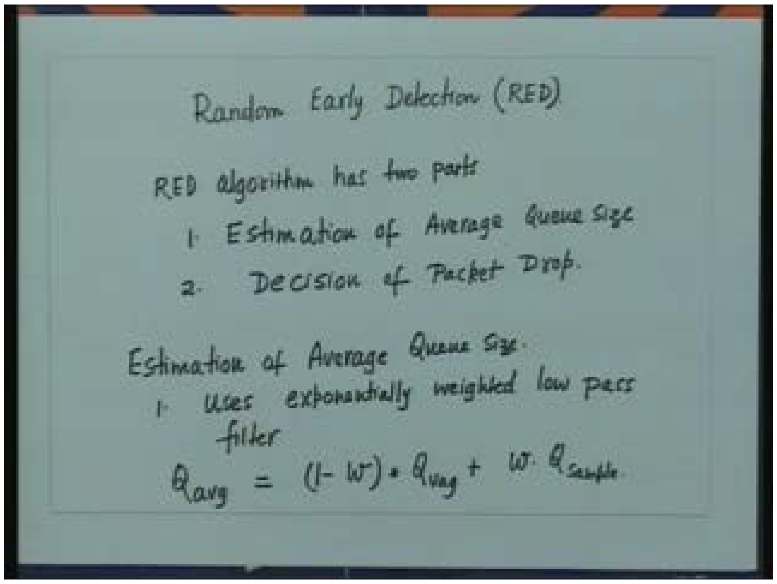
So, the degree so the buffer management, so the buffer management scheme will depend upon whether it is a flow specific flow based or whether all the flows have been aggregated into one and there is only stateless buffer management schemes. Then the dropping priorities: buffer management scheme may implement the priorities in terms of dropping. Some packets might be marked and which have a higher priority of dropping in the event of congestions and in the event of congestions, of some packets which have been marked specifically; they may be dropped first, then the packets which are not marked and so on.

So, while implementing the buffer management scheme, you may also implement the dropping priorities. The third is the buffer management schemes could be early drop also. Early drop means that even though the buffer is not full but we can proactively drop certain packets into the flow so that the sources will be informed of the incipient congestions that is going to build up and therefore the source will reduce their rate of transmissions. So, the buffer management schemes can also be early drop or they start dropping the packets only when the buffer is full.

So, depending upon that you have a classification of the buffer management schemes and the last is in terms of dropping position. So, dropping position is whether you are dropping it from the tail or whether you are dropping it from the front or whether you are dropping randomly. So, depending upon that also you can have a various buffer management schemes. Now, considering the fact that we are looking for a buffer management schemes; we should avoid global synchronizations, we should avoid bias against bursty traffic, we should maintain the network in a region of high throughput and low delay and not only that we should also have an ability to control the average queuing delay. There are network operator should have an ability to control the average queuing delay.

Now, taking to consideration all these factors, Floyd and Jacobson's has proposed a buffer management schemes that is most commonly used these days in a internet router which is called as the random early detection. So, let me just explain the basic principle behind the random early detection scheme.

(Refer Slide Time: 44:09)



## Random Early Detection (RED)

RED algorithm has two parts

1. Estimation of Average Queue Size
2. Decision of Packet Drop.

Estimation of Average Queue Size.

1. Uses exponentially weighted low pass filter

$$Q_{avg} = (1-W) * Q_{avg} + W \cdot Q_{sample}$$

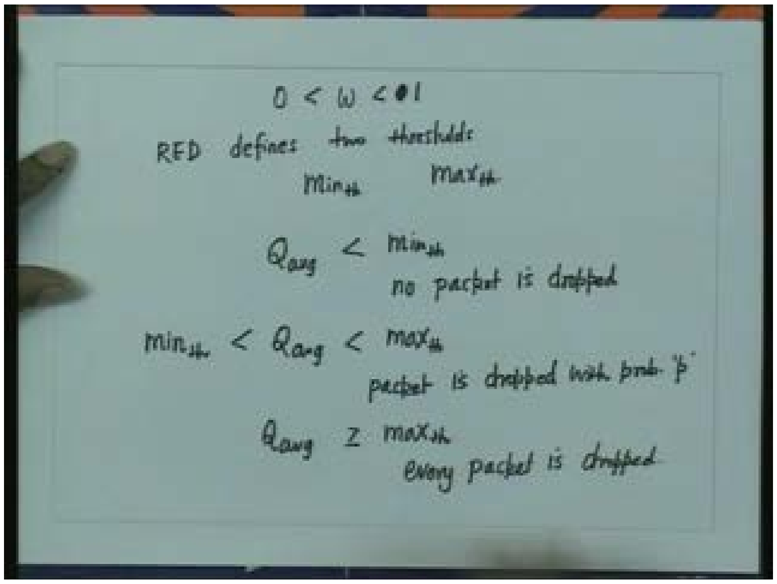
Random early detection: or as it is called RED, RED. Now, this algorithm has 2 parts: 1 is it estimates, an estimation of the average queue size which is actually an indication of the incipient congestions and second one is the decision of packet dropping. So, the RED algorithm has 2 parts; one is that estimation of the average queue size and second is the decision of packet drop.

Now, till now, the buffer management schemes which we were considering, we were saying that we will take a decision to drop the packet by looking at the instantaneous queue size. So, when the instantaneous queue size is full; then you drop a packet, may be from the tail of the queue or may be by choosing a packet randomly from the queue or may be from the front. But now, the RED advances the hypothesis that average queue size will be a better indication of the incipient congestions that is going to build up in the network. So, the decision to drop a packet is not based upon the instantaneous queue size but it is based upon the average queue size.

So, if the average queue size is greater than certain threshold, then only we will take a decision to drop the packet; otherwise, we will not take a decision to drop the packet. So now, this queue size is estimated using an exponentially weighted moving average of the queue size is determined. So, the queue size is estimated as estimation of average queue size. So, it uses an exponentially weighted low pass filter exponentially weighted low pass filter. So, the queue size is determined as Q average is equal to 1 minus which is the weight W multiplied by Q average plus weight into Q sample length.

So, what you do is that on every packet arrival or departure, whenever this queue sample length changes; you multiply this by weight plus 1 minus W into Q average and you add it to a queue average. So, this way you are actually determining a moving average of the queue size.

(Refer Slide Time: 47:50)



$0 < W < 1$

RED defines two thresholds

$min_{th}$ $max_{th}$

$Q_{avg} < min_{th}$

no packet is dropped

$min_{th} < Q_{avg} < max_{th}$

packet is dropped with prob. 'p'

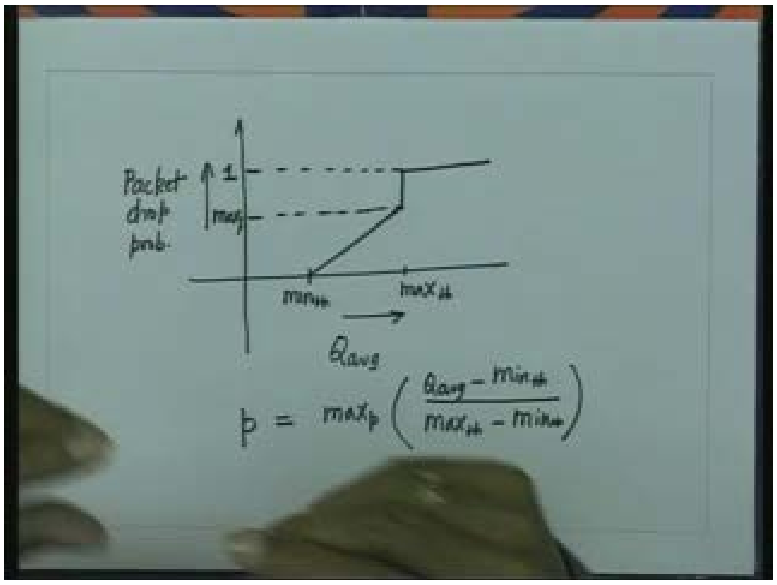
$Q_{avg} \geq max_{th}$

every packet is dropped

Now, the parameter W if the parameter W is chosen, so the parameter W lies between 0 and 1; if this W is chosen to be closer to 1, say 0.9 or so, then as you can see that the W is closer to 1, when the average queue size is more an indication from the instantaneous queue size. So, we will keep W to be very less, closer to 0.001 or 0.02 and so on. In that case, the average queue size will smoothen out any temporary fluctuations in the instantaneous queue size that may occur due to some bursts coming or so on.

So therefore, if you keep W to be closer to 0, then the average queue size will give us an indication of the congestion that is building up in the network. Now, the decision to packet drop is then taken, so the RED defines 2 thresholds. One is min thresh and another one is max thresh. So, if the Q average is less than min thresh, then no packet is dropped. If Q average is less than the max thresh but greater than the min thresh, then the packet is dropped with a probability p with probability p where p is computed and then and if Q average is greater than or equal to max thresh, then every packet is dropped. So, the packet dropping decision behaves something like this.

(Refer Slide Time: 50:22)



So, this is the Q average and this is the packet drop probability. So, as long as so this is min thresh and max thresh; so as long as it is less than min thresh, the packet is dropped at the probability p. As soon as it reaches the max thresh, the packet is dropped with 1. So, this p, the packet drop probability p is computed to be and this is max $_p$, max $_p$ into Q average minus min thresh upon max thresh minus min thresh. So, this is how you will compute the packet drop probability.

So, the basic principle behind RED is that it computes the average queue size which it considers to be an indication of the incipient congestions and then takes a decision on the packet drop probability depending upon whether the average queue size is between min and max threshold, whether the average queue size is less than the min threshold and whether the average queue size is greater than the max threshold.

It has been found through experimentations and simulations that the RED is able to achieve most of the desirable objectives of a good buffer management schemes. So, in conclusions therefore for TCP flows; a fair scheduling algorithms like weighted fair queuing in combination with a good buffer management schemes like RED, we can achieve the objectives of effective resource allocations and that is to say some kind of quality of service guaranties in a best effort internet.