Performance Evaluation with the help of RED Queue in the Buffer Management Mechanism

Rakesh kumar
REC, Ambedkar Nagar
Email Id-rakesh.knit1@gmail.com

Dr.Manisha Manjul
G.B.Pant Govt. Engineering College,New Delhi
Email Id-manishamanjul@gmail.com

Abstract— we introduce the buffer management mechanism in the computer network which is important to prevent the loss of packet and delay in the network for efficient and reliable delivery of packet. So the proposed scheme is based on the observation that when congestion control is implemented at the source, most of the loss occurs at the source. We evaluate the performance of buffer management on the basis of network parameters like bandwidth, queue size, packet size and transmission delay. We take the different type of scenario for finding the optimal solution of the problem which comes in the networks.

Index Terms— Buffer management mechanism, red queue, delay, bandwidth, queue management, Cogetion control, scheduling in queue, drop tail.

1 Introduction

Video enable applications are mostly used in our life without any delay, which also improve the quality of video. The needs for a central buffer management to achieves better memory utilization by enabling video stream sharing across components and to all network condition.
We will implement a queue management [1] scheme to manage the buffer at destination for video enable services which carries huge amount of data through network channel. Video data is generated at source which it reached to destination through various nodes and links. So, there may be delay, packet loss and jitter. To provide the better service at destination, we require a less delay, less amount of packet loss and less jitter. We are implementing a buffer management mechanism which care about packet loss and jitter.

1. Buffer Management

It is a technique which is used to improve the delivery of the packet in the network .We can improve these with the help of implementing different queue in the network. Congestion in a network may occur if the load on the network is more than the carrying capacity of the network [6, 7]. Congestion in a network or internetwork occurs because routers and switches have queues- buffers that hold the packets before and after processing. It degrades quality of service and also can lead to delays, lost data. Congestion can be brought on by several factors.
If all of a sudden, streams of packets begin arriving on three or four input lines and all need the same output line, a queue will build up. If there is insufficient memory to hold all of them, packet will be lost. This problem cannot be solved by increasing memory, because Nagle discovered that if routers have an infinite memory, congestion gets worse, not better. Slow processor can also cause congestion. If routers CPU's are slow at performing the tasks required, queues can build up, even though there is excess line capacity. Similarly, low bandwidth lines can also cause congestion. Network is greater than the capacity of the network-the number of packets a network can handle.

2. Queue Management

As we have seen, traffic phase effects occur when different flows of packet from different source and we can see different performances of the network. So we can solve this problem by implementing queue and solved by simply increasing the buffer size in the router. It seems that these effects would not occur or could at least be significantly diminished by increasing the maximum queue length. Since a queue is only meant to compensate for sudden traffic bursts, one may wonder what would happen if the queue length was endless. Of course, there is no such thing as an endless buffer, but it could be quite long [6, 7].

2.1 Need for Buffer

Congestion occurs when resource demands exceed the capacity [2, 3, and 4]. As users come and go, so do the packets they send; Internet performance is therefore largely governed

*Rakesh Kumar, Department of Information Technology, Rajkiya Engineering College, Ambedkar Nagar (U.P) – 224 122, India (e-mail: rakesh.knit1@gmail.com
*Manisha Manjul,Department of Computer Science and Engineering,GovindBallabh Pant Govt engineering college,New Delhi,(Email-id-manishamanjul@gmail.com)

by these inevitable natural fluctuations. Would it make sense to connect their Internet gateway option for now because this link is cheaper and suffices most of the time.
In this case, the gateway would see occasional traffic spikes that go beyond the capacity limit as a certain number of customers use their maximum rate at the same time. Since these excess packets cannot be transferred across the link, there are only two things that this device can do- buffer the packets or drop them.
Network Congestion Control: Managing Internet Traffic limited in time, standard Internet routers usually place excess packets in a buffer, which roughly works like a basic FIFO ('First In, First Out') queue and only drop packets if the queue is full [3]. The underlying assumption of this design is that a subsequent traffic reduction would eventually drain the queue, thus making it an ample device to compensate for short traffic bursts. Also, it would seem that reserving enough buffers for a long queue is a good choice because it increases the chance of accommodating traffic spikes.
There are however two basic problems with this:
1. Storing packets in a queue adds significant delay, depending on the length of the queue.
2. The consequence of the first problem is that packet loss can occur no matter how long the maximum queue, because of the second problem, queues should generally be kept short, which makes it clear that not even defining the upper limit is a trivial task.

2.2 Packet Scheduling In Queue

Queues represent locations where packets may be held (or dropped). Packet scheduling refers to the decision process used to choose which packets should be serviced or dropped. Buffer management refers to any particular discipline used to regulate the occupancy of a particular queue. At present, support is included for drop-tail (FIFO) queuing, RED buffer management, CBQ (including a priority and round-robin scheduler), and variants of Fair Queuing including, Fair Queuing (FQ), Stochastic Fair Queuing (SFQ), and Deficit Round-Robin (DRR). In the common case where a delay element is downstream from a queue, the queue may be blocked until it is re-enabled by its downstream neighbour. This is the mechanism by which transmission delay is simulated. In addition, queues may be forcibly blocked or unblocked at arbitrary times by their neighbours (which is used to implement multi-queue aggregate queues with inter-queue flow control). Packet drops are implemented in such a way that queues contain a "drop destination"; that is, an object that receives all packets dropped by a queue.

2.3 Factor Responsible for Occurrence of Congestion

Factor responsible for occurrence of congestion for that we need buffer Limited memory space, channel bandwidth, router capacity load of network, link failure, heterogeneous channel bandwidths the same key forcing the network to stuck into congestion. Detailed discussions of these factors are given below.

2.3.1 Effect of Buffer Space

The amount of buffer space given at a node is limited, the amount of information that can be stored at the node. For the case sufficient buffer is available more and more packets received at the node may accumulate and get transmitted later on. Here, the packets may suffer very large delay and subsequently leads to congestion. However for lower buffer space the packet may drop very frequently when load is increased and have a lower throughput. This is because packets have less room to wait for their chance. Thus neither using very large amount of buffer at node nor very less amount buffer is able to reduce congestion for all the case for the application have random load higher buffer size may be beneficial whereas medium range buffer is preferred for constant load the effect of channel bandwidth.

2.3.2 Effect of Channel Bandwidths

How Buffer (queue) control the packet in the network .How one design a mechanism could that automatically and ideally tunes the rate of the flow from sender to receiver .In order to answer this question, we should take a closer look at the elements involved.
• Traffic originates from a sender; this is where the first decisions are made (when to send how many packets).For simplicity, we assume that there is only a single sender at this point.
• Depending on the septic network scenario, each packet usually traverses a certain number of intermediate nodes. These nodes typically have a queue that grows in the presence of congestion; packets are dropped when it exceeds a limit.
• Eventually, traffic reaches a receiver. This is where the final (and most relevant) Performance is seen – the ultimate goal of almost any network communication code is to maximize the satisfaction of a user at this network node. Once again, we assume that there is only one receiver at this point, in order to keep things simple. Traffic can be controlled at the sender and at the intermediate nodes; performance measurements can be taken by intermediate nodes and by the receiver.

2.3.3 Assumption of queue length

Choosing the right queue length is essential for the performance of any computer network.
There are two reasons for this.
First, the source behaviour that we have so far taken into consideration relies on packet loss as a congestion indicator – thus, the rate of sources will keep increasing until the queue length grows beyond its limit, no matter how high that limit is.
Second, a queue can always overflow because of the very nature of network traffic, which usually shows at least some degree of self-similarity.
There is another reason why just picking a very large number for the maximum queue length is not a good idea: queuing

delay is a significant portion in the overall end- to-end delay, which should be as small as possible for obvious reasons (just consider telephony – delay is quite bothersome to users in this application). Remember what I said Queues should generally be kept short. The added delay from queues also negatively influences a congestion control algorithm, which should obtain feedback that reflects the current state in the network and should not lag behind in time.
After this discussion, we still do not know what the ideal maximum queue length is; it turns out that the proper tuning of this parameter is indeed a tricky issue.
Let us look at a single flow and a single link for a moment. In order to perfectly saturate the link, it must have c × d bits in transit, where c is the capacity (in bits per second) and d is the delay of the link (in seconds). Thus, from an end-system performance perspective, links are best characterized by their bandwidth × delay product.
On the basis of this fact and the nature of congestion control algorithms deployed in the Internet, a common rule of thumb says that the queue limit of a router should be set to the bandwidth × delay product, where 'bandwidth' is the link capacity and 'delay' is the average RTT of flows that traverse it.

3. Buffer Management Mechanism

3.1 Source Mechanism

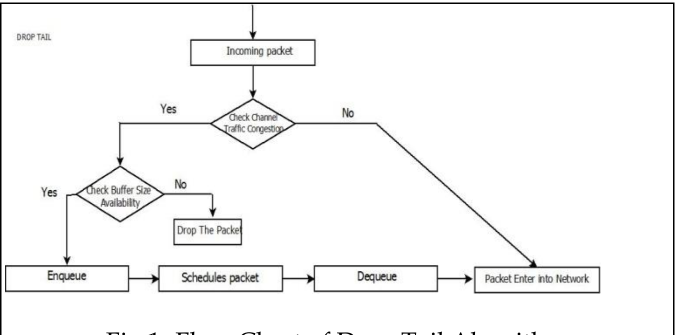
Buffer management mechanism is used to maintain the flow of packet in the network. In this, the queue mechanism is implemented at the source which has limited size to maintain the delay between source and destination. The packets arrive in the network and get enqueue in the queue and dequeue. If the buffer size is full the packet will discarded. So for extensive simulations we infer that for multimedia transmission into a TCP based network, most loss occurs at the point of transmission i.e. the source, and not at the nodes inside the network [7].
This is contrary to the belief that the packet loss in the network due to congestion is the major contributor to the total loss a TCP flow suffers. Our simulations show that in response to congestion the transmission queues at the sources increase which finally leads to packet drops at the source and it is this dropping at the source that is the major contributor to the aggregate loss of the flow.
In this section we propose a simple buffer management strategy for evaluating the buffer management performance. Packets produced by the source are stored in the buffer prior to transmission.

3.2 Mechanism

In this paper we have discuss many algorithm which is used to prevent the packet loss the packet, some of them are RED and Drop Tail [2, 11]. And we also discuss the flowchart and working of the algorithm which are given below.

Drop Tail

Tail Drop or Drop Tail, is a simple queue management [1] algorithm used by Internet routers to decide when to drop packets. In contrast to the more complex algorithms like RED and WRED, in Tail Drop all the traffic is not differentiated. Each packet is treated identically. With tail drop, when the queue is filled to its maximum capacity, the newly arriving packets are dropped until the queue has enough room to accept incoming traffic as we can see in figure1.


DROP TAIL
Incoming packet
Yes
Check Channel Traffic Congestion
No
Yes
Check Buffer Size Availability
No
Drop The Packet
Enqueue
Schedules packet
Dequeue
Packet Enter into Network
Fig.1- Flow Chart of Drop Tail Algorithm

The name arises from the effect of the policy on incoming data grams. Once a queue has been filled, the router begins discarding all additional data grams, thus dropping the tail of the sequence of data grams. The loss of data grams causes the TCP sender to enter slow-start, which reduces throughput in that TCP session until the sender begins to receive acknowledgements again and increases its congestion window. A more severe problem occurs when from multiple TCP connections are dropped, causing global synchronization, i.e., all of the involved TCP senders enter slow-start. This happens because, instead of discarding many segments from one connection, the router would tend to discard one segment from each connection.

Drop Tail: The working of drop tail algorithm with the help of flow chart we can see step by step:

```
1. If (No Of Packet Incoming>Channel Bandwidth){
2. Then Check Buffer Size}
3. else if( buffer size >= size of packet){
4. Then enqueue the packet
5. Else drop the packet}
6. If (noof incoming input <channel bandwidth ){
7. Then packet enter the channel}
```

Flow Random Early Drop
If it is acceptable to maintain per-flow state because the number of flows is boundedand a large amount of memory is available, fairness can be enforced by monitoring all individual flows in the queue and the result can be used to make appropriate decisions.
This is the approach taken by flow random early drop this mechanism, which is another incremental RED enhancement, always accepts flows that have less than a minimum threshold minq packets buffered as long as the average queue size is smaller than maxth. As with standard RED, random dropping comes into playonly when the average queue length is above

minth, but with FRED, it only affects flows that have more than minq packets in the queue. Note that this type of check requires the mechanism to store per-flow state for only the flows that have packets in the queue and not for all flows that ever traversed the link, and thus, the required memory is bounded by the maximum queue length.

Random Early Detection (RED)

Red algorithm provides the mechanism which mange the transfer of packet from source to destination through multiple nodes and the mechanism called buffering of the packet into the queue. As in the case of the Internet, the chosen entity was the router describes a mechanism called Random Early Detection (RED), which is now widely deployed and makes a decision to drop a packet on the basis of the average queue length and a random function as well as some parameters called probability of dropping the packet. RED is a popular example of a class of so-called active queue management (AQM) [2].

This algorithm calculates the average queue length, if it is greater than the maximum then packet drop. Randomness comes into play only when the average queue length is between minth and maxth – then, the probability of dropping a packet will be between zero and the maximum marking probability maxp, and it will directly be proportional to the average queue length.

In other words, when the average queue length grows beyond minth, the marking probability rises linearly from zero to maxp, which is when the average queue length will grow beyond maxth and all packets will be marked. Where avg is the average queue length estimate, q is the instantaneous queue length and wq is a weighting factor that controls how fast the moving average adapts to fluctuations. Then, it is compared to two thresholds called minth and maxth. If the average queues size is less than the minth, maxth.

These values depend on the desired average queue size. In other words, setting this parameter to a small value will lead to a small queue (and thus short delay). On the other hand, the parameter minth depends on the burstiness of traffic – if fairlybursty traffic should be accommodated, it must be set to a rather large value – and at the same time, (maxth − minth) should not be too small to allow for the randomness to take effectmaxp: This parameter controls how likely it is for a packet to be discarded when the average queue length is between minth and maxth. Complete mechanism of RED given as a flow chart in figure2.

RED queue: The algorithms of red queue in flow chart figure2 given in step by step.

1 calculate average length queue

2. If (average length >=threshold) or

Probability to drop the packet is high

3. ElseIf (Average length <=threshold) or

Probability value is low then

4. Enqueue packet

5. Packet sent to network

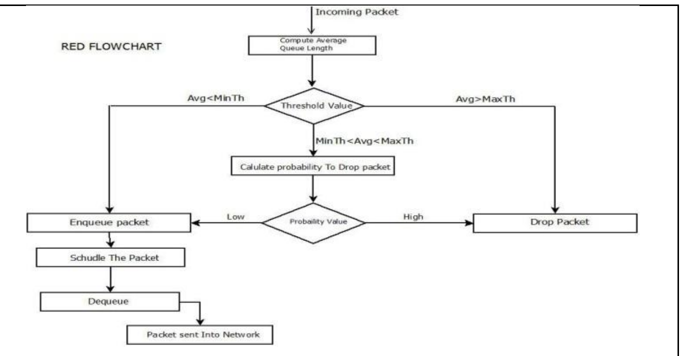


Fig.2-Flow Chart of Red Algorithm

Adaptive RED

This is the underlying idea of Adaptive RED [2], which was originally described on the basis of the dynamics of the queue length, the maxp parameter is varied. This makes the delay somewhat more predictable because the average queue length is under the control of this parameter. When the network is generally lightly loaded and maxp is high, the average queue length is close to minth, and when the network is heavily congested and maxp is low, the average queue length is close to maxth.

Dynamic-RED

Dynamic-RED (DRED) is a mechanism that stabilizes the queue of routers; by maintaining the average queue length close to a fixed threshold, it manages to offer predictable performance while allowing transient traffic bursts without unnecessary packet drops. The design of DRED is described in it follows a strictly control-theoretical approach. The chosen controller monitors the queue length and calculates the packet drop probability using an integral control technique, which will always work against an error in a way that is proportional to the time integral of the error, thereby ensuring that the steady-state error becomes zero. The error signal that issued to drive the controller is filtered with a EWMA process, which has the same effect as filtering (averaging) the queue length – just like RED, this allows DRED to accommodate short traffic bursts.

Stabilized RED

Stabilized RED (SRED) also aims at stabilizing the queue length, but the approach is quite different from DRED: since the queue oscillations of RED are known to often depend on the number of flows, SRED estimates this number in order to eliminate this dependence. This is a achieved without storing any per-flow information, and it works as follows: whenever new packet arrives, it is compared with a randomly chosen one that was received before. If the two packets belong to the same flow, a 'hit' is declared, and the number of 'hits' issued to derive the estimate. Since the queue size should not limit the chance of noticing packets that belong together, this function is not achieved by choosing a random packet from the buffer – instead, a 'zombie list' is kept [9, 10].

International Journal of Scientific & Engineering Research Volume 7, Issue ƕlǾ#ÌÊÌÔÉÌÙıl˩ƕłⅢ
ISSN 2229-5518

4. Implementation of Queue Management

NS-2 is an object oriented simulator, written in C++, with an OTcl interpreter as a frontend. The simulator supports a class hierarchy in C++ (also called the compiled hierarchy in this document), and a similar class hierarchy within the OTcl interpreter. NS-2 uses two languages because simulator has two different kinds of things it needs to do. On one hand, detailed simulations of protocols require a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important. NS-2 meets both of these needs with two languages, C++ and OTcl. C++ is fast to run but slower to change, making it suitable for detailed protocol implementation. OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration. ns (via tclcl) provides glue to make objects and variables appear on both languages simulator: The overall simulator is described by a Tcl class Simulator. It provides a set of interfaces for configuring a simulation and for choosing the type of event scheduler used to drive the simulation.
A simulation script generally begins by creating an instance of this class and calling various methods to create nodes, topologies, and configure other aspects of the simulation.

4. SIMULATION Environment

Parameter Used

The initial parameters are used in the following tabel:

Table 1: Parameters Used

S.No.	Parameters	Quantity
1	Bandwidth(Mb)	20
2	Queue size	19
3	Link Delay	30
4	Packet size	5000
5	Source Rate	1.5Mb
6	Simulation Time	5 minute

So, all initial parameters shown in the above table 1 which is taken for our simulation. The bandwidth between link node 0 to node 3, node 1 to node 3 and node 2 to node 3 is 20Mb, node 3 to node 4 is10Mb and node 4 to node5, node 4 to node 6 is 20 Mb and delay in all links is 30ms. The initial queue size assumed is 4 packets. But the bandwidth, queue size and transmission delay varies from scenario to scenario.

Network simulator-2 (NS-2)

So for creating the environment for buffer management in Queue exist in NS-2 as given in figure3.
In this we go to the NS-allinone, ns-2.34 then in queue in which drop tail and read queue exist.

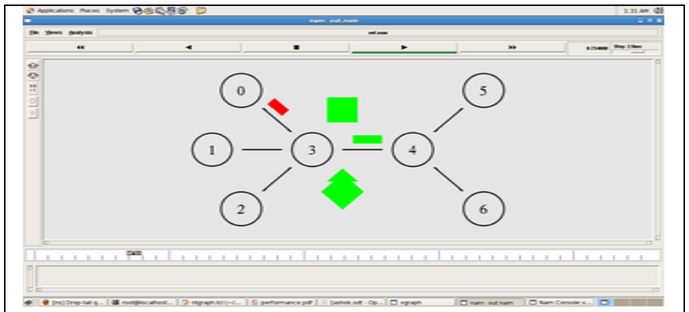


Fig.3-Simulation topology

5. Result and Discussion

In first scenario we consider variation in bandwidth only in link n3-n4 because we are implemented RED queue on this link. In which we keep changing the bandwidth i.e.10Mb, 20Mb, 24Mb. And all the parameter remains fixed i.e. queue size is 4 packets and the transmission delay is 30ms in this, we consider the queue size minimum. Networks have congestion because we provide the insufficient bandwidth to the link and not enough queue size.
So from this we have analyzed that the number of packet loss is more and less number of packet is received. We have analyze the other variation of parameter in next scenario i.e. queue size and transmission delay.

Table 2: Variation in Bandwidth.

S.No.	Link	Bandwidth (Mb)	Queue size	Transmission Delay	Packet Loss	Packet Receive	Queued Packet	Avg Queued Packet
1	N3-N4	10	4	30	206	416	2000	205
2	N3-N4	20	4	30	256	372	2013	92
3	N3-N4	24	4	30	328	336	1970	64

We only show the graph of serial number 3 in which link n3-n4 have bandwidth is 24Mb.

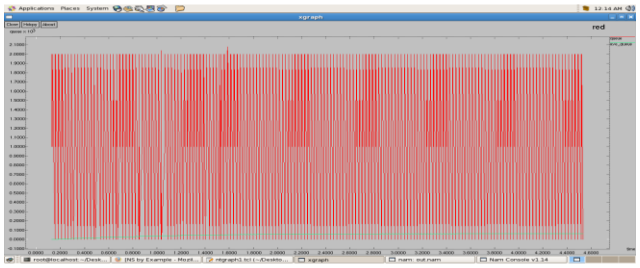


Fig.4- Time Vs Quaeue Size at Different Bandwidth

In second scenario we consider the variation in queue size only in link n3-n4 because we are implemented RED queue on this link. In which we keep changing the queue size i.e.10, 15, and 19 packets respectively. And all the parameter remain fixed i.e. bandwidth is 24 Mb and the transmission delay is 30ms.In this we varies the queue size. Networks have congestion because we provide the traffic on to the link.

So from this we have analyzed that the number of packet loss is less from previous case number of packet is received is more we see it in serial no3 link n3- n4. We have analyze the other variation of parameter in next scenario i.e. transmission delay.

Table3: Variation between Queue sizes from initial parameter.

S.no.	Link	Bandwidth(Mb)	Queue size	Transmission delay	Packet Loss	Packet Receive	Queued Packet	Avg Queued packet
1	N3-N4	24	10	30	394	303	3980	105
2	N3-N4	24	15	30	360	320	3960	100
3	N3-N4	24	19	30	80	829	3970	107

We only show the graph of serial number 3 in which link n 3-n 4 have bandwidth is 24 Mb and queue size is 19 packets.

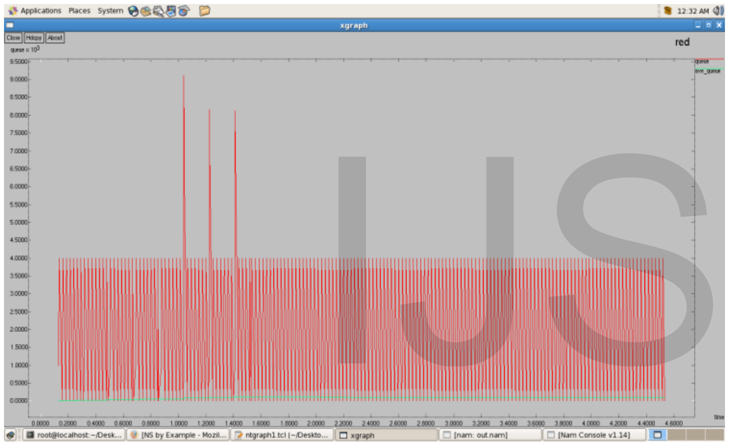


Fig.5-Time Vs Queue Size at Different Queue Size

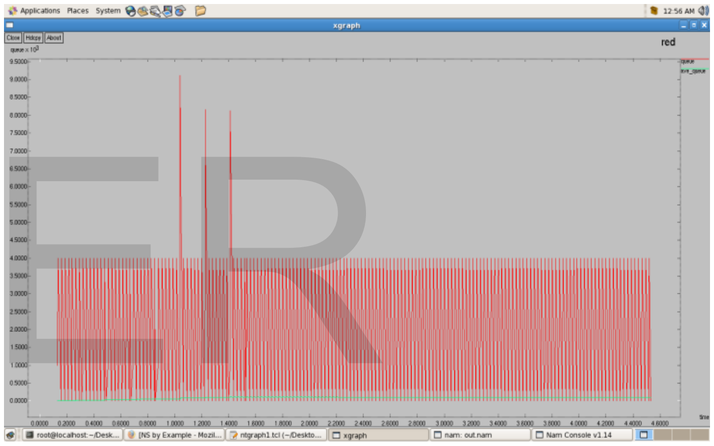


Fig.6-Time Vs Queue Size at Different Transmission Delay

S.no.	Link	Bandwidth(Mb)	Queue size	Transmission delay	Packet Loss	Packet Receive	Queued Packet	Avg Queued packet
1	N3-N4	24	19	10	No	829	3950	104
2	N3-N4	24	19	20	No	829	3985	102
3	N3-N4	24	19	30	No	829	4000	102

In third scenario we consider the variation in transmission delay only in link n 3-n 4 because we are implemented RED queue on this link. And every packet provided with transmission time. In which we keep changing the transmission delay
i.e.10ms, 20ms, 30ms. And all the parameter remains fixed i.e. Bandwidth is 24Mb and queue size is 19 packets. In this we varies the transmission time to analyze the result of packet loss in the network.

Table4: Variation between Transmission Delay

So from this we have analyzed that no packet loss occurs in the network and the number packet received to the destination side is maximum. So we have analyzed that when we are taking the bandwidth 24Mb, queue size 19packet and transmission delay is 30ms, so no packet loss occur in the network. Number of packet queued is 4000 and average

queued is 102 packet.
We only show the graph of serial number 3 in which link n3-n 4 has bandwidth is 24Mb and queue size is 19 packets and transmission delay is 30ms. We can analyze it from the graph.
So from the entire scenario we make the Comparison between number Packet Loss and number of Packet Received. This comparison provides the solution of the problem which occurs in the network that is packet loss is minimized as in the previous scenario.
So the parameter which provides the solution is considered are bandwidths is 24Mb, queue size is 19 Packet and transmission delay is 30ms. In this simulation no packet losses occur in the network which is optimum solution of this problem.
So the parameter which provides the solution is considered are bandwidths is 24Mb, queue size is 19 Packet and transmission delay is 30ms. In this simulation no packet losses occur in the network which is optimum solution of this problem.

4. CONCLUSION

We evaluated the performance of buffer management on the basis of network parameters like bandwidth, queue size, and transmission delay. We take the different type of scenario for finding the optimal solution of the problem which comes in the networks .The problem is in the network like packet loss, congestion and delay.
We discuss the algorithm Red and Drop tail and by applying this algorithm we maintain the loss of the packet in the network. We finally reach on the result in the different-different scenarios that provide efficient way for transmission in network using buffer management.

Table5: Comparison between packet loss and packet received

S.No.	Parameter change	Packet Received	Packet Loss
1	Bandwidth	416	206
2	Bandwidth	372	256
3	Bandwidth	336	328
4	Queue size	303	394
5	Queue size	320	360
6	Queue size	829	80
7	Transmission delay	829	No
8	Transmission delay	829	No
9	Transmission delay	829	No

REFERENCES

[1] Arash Dana and Ahmad Malekloo "Performance Comparison between Active and Passive Queue Management" IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 5, May 2010.

[2] Hussein Abdel-jaber "Performance study of Active Queue Management methods: Adaptive GRED, REDD, and GRED-Linear analytical model" Journal of King Saud University –

[3] Computer and Information Sciences Volume 27, Issue 4, October 2015, Pages 416–42.

[4] Shensheng Tang and Wei Li, "QoS Provisioning and Queue Management in Mobile Ad hocNetworks"in Wireless Communications and Networking Conference, pp.400-405, April 2006. [4]Andrew S. Tanenbaum fourth edition "Computer Network" Prentice Hall, May 6, 2011

[5] Behrouz Forouzan's fifth edition"Data Communication and Network" Science Engineering & Math -July 1, 2012.

[6] S. Rajeswari, Dr.Y.Venkataramani "Congestion Control and QOS Improvement for AEERG protocol in MANET

[7] " International Journal on AdHoc Networking Systems (IJANS) Vol.2, No.1, January 2012.

[8] Karan Singh, Rama Shankar Yadav and Grish Kumar Gupta "Effective Queue Management for Layered Multicast" Accepted for publication in IEEE Bangalore Section -18th Anual Symposium on Emerging Needs in Computing, Communication, Signals and Power ENC2SP 2009, Bangalore 29 August 2009.

[9] Ivan V. Baji´c, Omesh Tickoo, Anand Balan, Shivkumar Kalyanaraman" Integrated end-end buffer management and congestion control for scalable video communications:" in December 12, 2002

[10] Shalini batra ,Yan bai" Packet buffer management for high speed network Interface Card "

[11] Z. Miao and A. Ortega, "Optimal scheduling for streaming of scalable media," Proc. Asilomar Conf. on Signals, Systems, and Computers, Pacific Grove, CA, November 2000.

[12] Eitan Altman and Tania Jimene"NS Beginner" Lecture notes,2003-2004,university de Los Andes,Merida venazuela and ESSI December4,2003.