International Journal of Applied Engineering Research, ISSN 0973-4562 Vol. 10 No.56 (2015)

Improving Network Performance and Reducing the Network Traffic using Queue Management Algorithms

Sakthivel V
Department of CSE
Jansons Institute of Technology
Coimbatore, India.
mvsakthi@gmail.com

Jansirani S
Department of CSE
Jansons Institute of Technology
Coimbatore, India.
jansishanmugam@gmail.com

Savitha N
Department of CSE
Jansons Institute of Technology
Coimbatore, India.
nsavi16edu@gmail.com

Kiruba Priyadharshini V
Department of CSE
Jansons Institute of Technology
Coimbatore, India.
kirubampc@gmail.com

***Abstract*—Due to the fast developments of the Internet, and other new technologies congestion control is an important factor to increase the network utilization, limit packet loss and delay. Several methodologies are used to reduce the packet loss during transmission via communication channel. This paper compares the performance of Drop Tail, RED, FQ, SFQ queuing mechanism and the performance is evaluated under various load situations for FTP and HTTP respectively. We use some measures and simulations to evaluate the performance of the Queue Management Algorithms and assess their impact on a set of performance metrics. We find that in total (i) no performance improvements of RED compared to Tail Drop can be observed; (ii)Due to the variability in traffic load fine tuning of RED parameters is not sufficient. In order to limit the escalating packet loss rates caused by an exponential increase in network traffic, Queue Management Algorithms such as Drop Tail, RED, FQ, SFQ are used. Stochastic Fair Queuing (SFQ) ensures fair access to network resources and prevents a busty flow from consuming more than its fair share. Performance parameter of all algorithms is analyzed using NS-2 network simulator.**

Keywords—Queue Management Algorithms (QMA), Congestion Control, Random Early Detection (RED), Fair Queuing (FQ), Stochastic Fair Queuing (SFQ).

I. INTRODUCTION

In the last few years there has been an explosive growth in internet access like IP telephony, teleconferencing, streaming video/audio, interactive games, distance learning etc., in which data has been transmitted between pair of applications in distinct host which are connected to the communication network. During data transmission when a node carrying large volume of data on low bandwidth across the network results in congestion which slows down the packet movement, packet loss and drop in service quality. Hence buffer space or queue is necessary to avoid information loss during data transmission. The primary purpose of a queue is to smooth out bursty arrivals, so that the network utilization can be high. These queues increase network end-to-end delay which significantly affects the quality of real-time applications. Internet best effort traffic is controlled through the interaction of end-to-end congestion control and queue management.

Queue management decides when to start dropping packets and which packets to drop at a congested node output port. Traditional technique for queue management is "Drop Tail", which discarding arriving packets if the buffer of the output port overflows.

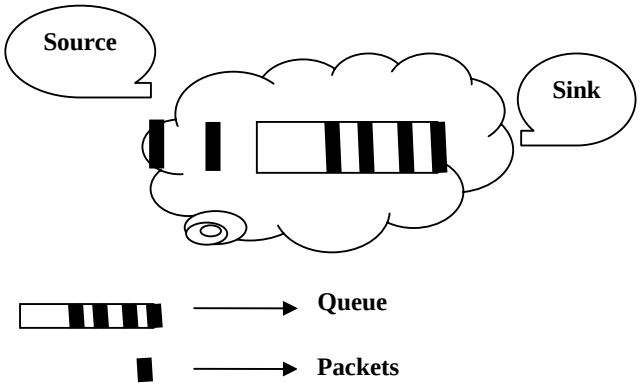

Fig.1. Queue Management and Packets Flow from source to destination

Contrary to Drop Tail, queue management Algorithms (QMA) start dropping packets earlier to enable notification of traffic sources about the *early* stages of congestion. SFQ (Stochastic Fair Queuing) is a class of queue

scheduling disciplines that are designed to allocate a pretty large number of separate FIFO queues. Increasing the number of queues to a large extent helps to achieve fairness. RED queue management aims at alleviating this problem by detecting incipient congestion in advance and communicating the same to the end-hosts, allowing them to trim down their transmission rates before queues begin to overflow and packets start dropping. For this, RED maintains an exponentially weighted moving average of the queue length which it used as a congestion detection mechanism. RED must also ensure that the queue is configured with enough buffer space.

II. QUEUE MANAGEMENT ALGORITHMS

Before transmitting the data, the node senses the medium to check if the medium is free or not. If the medium is free, it sends the data to another node via network channel to avoid the congestion which results in packet loss. To aggravate the network performance drop caused by the increased load, new techniques like Tahoe, Reno, Vegas etc., have been introduced which only provides end-to-end congestion control functionality by using the drop-tail policy for routing. Instead of waiting for the congestion to occur in the nodes, threshold value is fixed in the queue and based on that threshold value the node signaling to the sender to adjust the window size to avoid packet drops. To avoid congestion several Queue Management Algorithms have been developed and every algorithm has its own merits and demerits which are shown below.

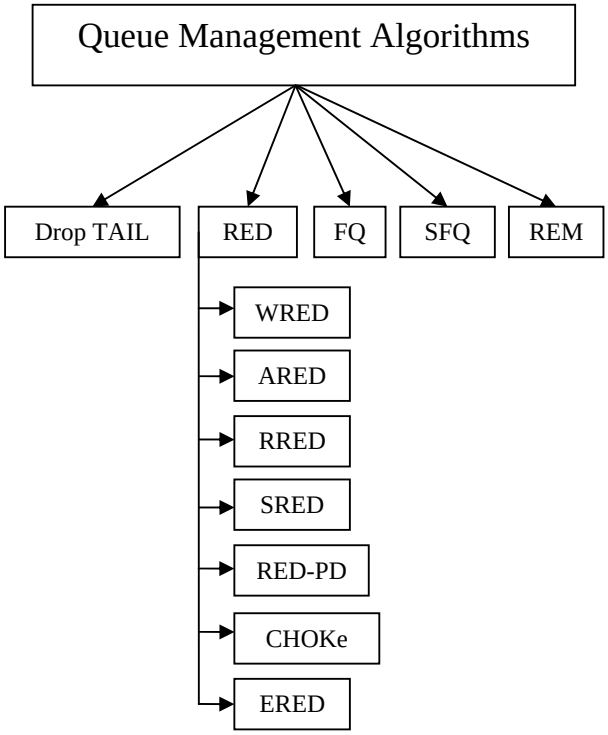

Fig 2. Classification of Queue Management Algorithms

A. DropTAIL

Drop Tail is the simplest queue management algorithm used in many routers; it drops all incoming packets once the queue is full. It considers the traffic identically, in spite of the type of traffic which it belongs to. Packet loss indicates the sender to reduce the number of packets sent before receiving the acknowledgment from the receiver. Automatically the throughput is reduced until the sender node start again to receive acknowledgments and begin rising the size of its congestion window. It is easy to implement since it has minimal computation overhead but it doesn't distribute buffer space reasonably. Hence it results in unsatisfactory performance. If multiple connections exist in the network and a buffer overflow brings global synchronization, which reduce the network throughput and utility significantly. Drop probability used in Drop Tail is 0 and 1. When the number of packets arrived to the queue larger than the buffer size, the probability of packet dropping is 1. Otherwise the dropping probability is 0.

B. RED

RED (Random Early Detection) mechanism works on certain probability in which the minimum and maximum threshold values are assigned to a queue. When the average queue length exceeds the predefined minimum threshold value, then it indicates to the sender to shrink the window size. When the average queue length exceeds the maximum threshold value, the probability of packet dropped becomes 1. When the queue is full all the incoming packets are dropped. RED monitors the average queue length by discarding or Explicit Congestion Notification marking packets based on statistical probability.

A network with RED algorithm detects congestion and measures the traffic load in the network by using the average queue size *avg*. This is calculated using an exponentially weighted moving average filter and expressed as the following equation,

$$avg = (1 - w_q) \bullet avg + w_q \bullet q \quad (1)$$

where w_q is queue weight.

When the average queue size is smaller than the minimum threshold min_{th}, no packets are dropped. Once the average queue size exceeds the minimum threshold, the router considers the network in congestion and randomly drops the arriving packets with a given drop probability. The probability that a packet arriving at the RED queue is dropped depends on the average queue length, the time elapsed since the last packet was dropped, and the maximum drop probability parameter max_p. The drop probability P_a is computed as

$$P_a = \frac{P_b}{1 - count \bullet P_b} \quad (2)$$

where $P_b = \max_p \bullet \frac{avg - \min_{th}}{\max_{th} - \min_{th}}$, max_p is maximum value for P_b and count is a variable that keeps track of the number of packets that have been forwarded since the last drop. If the

average queue size is larger than the maximum threshold max_{th}, all the arriving packets will be dropped.

It is shown in that the average queue length (*avg*) is related to the number of active connection (*N*) in the system as follows,

$$avg = 0.91 N^{2/3} (max_{th} / max_{p})^{1/3} \quad (3)$$

For the RED algorithm, this equation implies that *avg* increases with *N* until max_{th} is reached, and there is always an *N* where max_{th} will be exceeded. Since most existing routers operate with a limited amount of buffer, max_{th} is usually small and can easily be exceeded even with small *N*. Dropping all incoming packets may result in global synchronization, which is usually followed by a sustained period of low-link utilization.

The purpose of RED is to achieve high link utilization, minimize queuing delay and packet loss, to better accommodate bursty sources, and to provide a low-delay environment for interactive services by maintaining a small queue size.

C. Fair Queuing

In fair queuing every flow gets the bandwidth propositional to its demand. The main goal of fair queuing is to allocate resources fairly to keep separate queue for each flow currently flowing via a router. Every queue gets equal bandwidth when the packets are in same size and non-empty queue follows a round robin fashion like FIFO. But if the packets are in different size the flow of large size packets gets more bandwidth than small size packets and these problems are overcome by weighted fair queuing algorithm, etc. Maintaining a separate queue for each flow requires a gateway or router to map from source to destination address pair for the related queue on a per packet basis.

D. Stochastic Fair Queuing

Stochastic Fair Queuing uses a hash algorithm to divide the traffic over a limited number of queues. Due to the hashing in SFQ multiple sessions might end up into the same bucket. SFQ changes its hashing algorithm so that any two colliding sessions will only work for a small number of seconds. Stochastic Fair Queuing algorithm is the best algorithm among all algorithms in case of providing satisfactory bandwidth to the legitimate users (TCP and UDP) in network. It is called Stochastic due to the reason that it does not actually assign a queue for every session; it has an algorithm which divides traffic over a restricted number of queues using a hashing algorithm.SFQ assigns a pretty large number of FIFO queues. Stochastic Fair Queuing (SFQ) ensures fair access to network resources and prevents a busty flow from consuming more than its fair share.SFQ has a minimum average loss ratio and maximum throughput compared to RED. RED has a maximum loss ratio. SFQ has deliver maximum number of packets and it is best against attack intensities for buffer size 60. SFQ gives maximum packet loss for CBR and it is suitable for use in high speed computer networks that cover a wide range of CPU, memory and fairness trade-offs. It offers elegant degradation under overload and sudden failure. The only drawback of SFQ is unfair behavior with the flows colliding with other flows. Thus, as the name reveals, fair is guaranteed as stochastically.

III. EXPERIMENTAL RESULTS AND DISCUSSION

The simulation tool is NS2. The experimental platform is with Lenovo Think Centre PC, the system of Fedora core Linux 19.0, memory of 4GB ,and network card ,hard disk of 200 GB. In order to evaluate the performance of Drop TAIL, RED, FQ and SFQ a simulation experiments were done using ns2 simulator over a small network contains 8 nodes and one gateway. The nodes (no, n1, n2, n3, n5, n6, n7, n8) are represented as circle and the gateway (n4) is represented as square which is shown in Fig.3.

Using this network a TCP full duplex link is established between node0 to node3 (n0-n3), node1 to node6 (n1-n6), node2 to node5 (n2-n5), node3 to node7 (n3-n7), node5 to node8 (n5-n8) respectively. File transfer protocol (FTP) is used to transmit data via gateway. To indicate the TCP data flow various colors are assigned using flow ids. For node0 to node3 green color is assigned, node1 to node6 blue color, node2 to node5 red color, node3 to node7 yellow color and node5 to node8 purple color is assigned. TCP flow is occurred between node0 to node3.Sender (node 0) sends the data at the maximum speed of 15.5 Mbps and the receiver process the data at the maximum speed of 3.5Mbps along the duration of 10ms.The queue size is fixed as 5 for all flows and queue position is fixed as 0.5. The experiment is done by using various Queue Management Algorithms Drop TAIL, RED, FQ and SFQ and the result is shown in Fig.3 to Fig.6.

From the Tcl script, it runs the program that generates three graphs(Fig.7 to Fig.9) showing the behavior of a TCP flow under different background traffic. From the Tcl script, it runs the program that generates trace files(.tr) and executes nam window. While running nam window drop.nam the information is updated to the trace file drop.tr. Trace file contains event information like enqueue(+), dequeue(-), received(r), dropped(d), datalink layer details(source address and nearby network device address) and network layer detail(exact source and destination address) ,port number, flag, protocol details, packet size, packet number, ack details, duration, etc,. The trace file contains useful data from the simulation, using this result one can find the number of packets dropped, the throughput, delay, etc.

To fetch the information from the trace file another scripting language awk is used.awk is specially designed for working with structured files and text patterns. By using this awk file the information is parse out from the trace file drop.tr and stored to the new file drop, The new file contains number of packet loss for each flow i.e between 2 nodes(n0 –n3,etc).

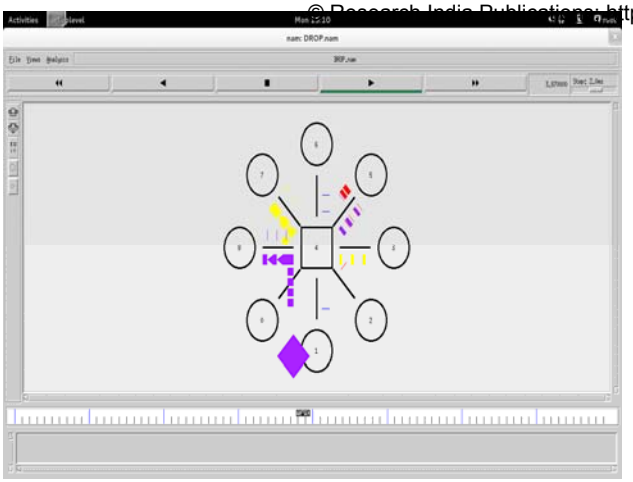

Fig 3. Illustration of Queue Management Algorithms using Drop TAIL

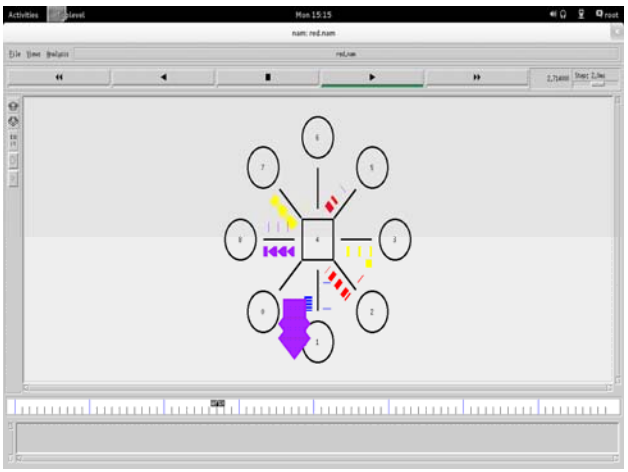

Fig 4. Illustration of Queue Management Algorithms using RED

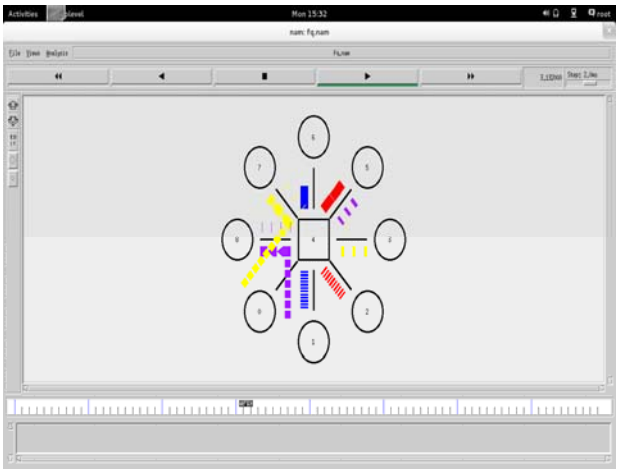

Fig 5. Illustration of Queue Management Algorithms using FQ

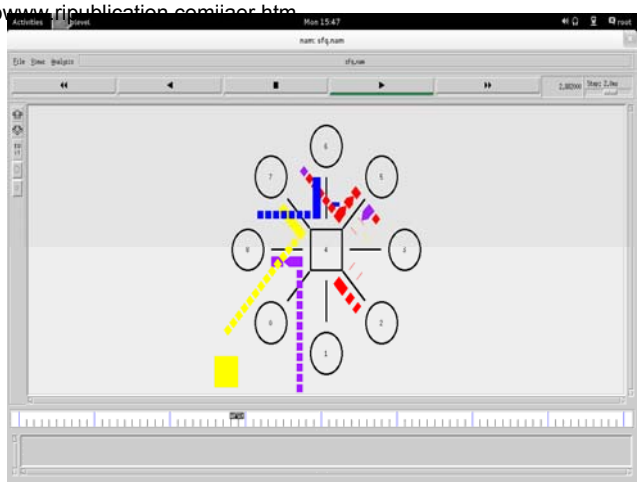

Fig 6. Illustration of Queue Management Algorithms using SFQ

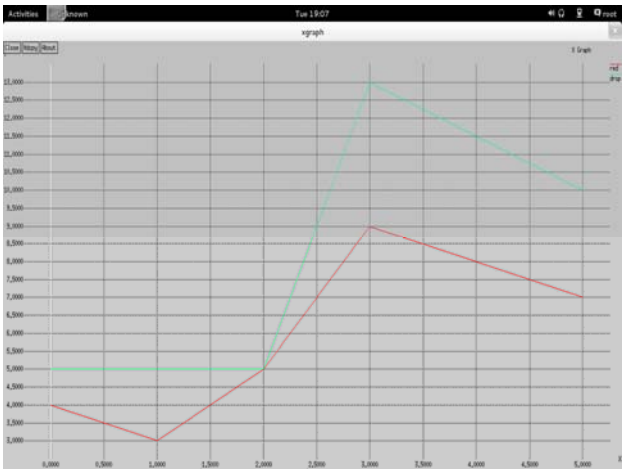

Fig 7. Illustration of Packet loss by Drop TAIL, RED

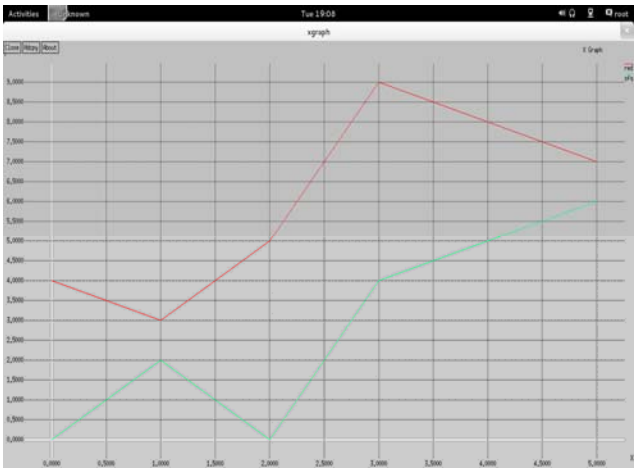

Fig 8. Illustration of Packet loss by RED and SFQ

Fig 9. Illustration of Total Packet loss

S.NO	NODE		PACKETLOSS			
	SOURCE	DESTINATION	DROP TAIL	RED	FQ	SFQ
1	0	3	5	4	0	0
2	1	6	5	3	0	2
3	2	5	5	5	0	0
4	3	7	13	9	0	4
5	5	8	10	7	0	6

Table 1: Illustration of Packet Loss by various QMA

Table 1 illustrates the packet loss information obtained for the various queue management algorithms (DROP TAIL, RED, FQ, SFQ). For the TCP flow between the node 0 to node 3 Drop TAIL lost 5 packets and RED lost 4 packets. But there is no packet loss in FQ and SFQ.

The result shows that Drop TAIL has high packet loss when compare to all other algorithms and RED has less packet loss compare to Drop TAIL and higher packet loss compare to FQ and SFQ. In FQ no packet is dropped during transmission via communication channel, because it creates a separate queue for each flow. But the problem is time consuming.SFQ has less packet loss compared to Drop TAIL and RED.

IV. CONCLUSION

In this paper we compared Drop tail, RED, FQ, SFQ Queue management algorithms under different network traffic. From the simulation result, it clearly shows that Drop TAIL performance is very poor which has high packet loss when compare to all other algorithms. But it is so simple and easy to implement compare to other mechanisms. RED does not perform well in a heavily loaded network and also stabilizing the queue size and parameterize the queue is a difficult task. Fair queuing maintains a data flow good way such that all packets shares the resources equally. In Fair Queuing no packet is dropped during transmission via gateway, because it creates a separate queue for each flow. But the problem is time consuming which will degrade the network performance. SFQ has high intelligence which create a queue only traffic is occurs. Instead of creating the separate queue for each flow SFQ create limited number of queues by dividing the network traffic using hash function. It avoids bursty flow and achieves maximum throughput minimum packet loss.

V. FUTURE ENHANCEMENT

In the Queue Management Algorithms we compared only the four algorithms like Drop TAIL, RED, FQ and SFQ. In this SFQ has high performance and low network overhead. In future we will compare all the queue management algorithms like BLUE, REM, etc. In this paper we used Ns2 for our simulation results and in future we use ns3 for script oriented web and simulation results.

REFERENCES

[1] Ganesh Patil, Sally Mcclean And Gaurav Raina G, Drop Tail And Red Queue Management With Small Buffers: Stability And Hopf Bifurcation**,** Ictact Journal On Communication Technology: Special Issue On Next Generation Wireless Networks And Applications, June 2011, Volume – 2, Issue – 2.

[2] Md Tarmizi, A. Albagul, Othman.O. Khalifa, WahyudiQoS Evaluation of Different TCPs Congestion Control Algorithm Using NS2, 2006 IEEE.

[3] G.Thiruchelvi and J.Raja, A Survey On Active Queue Management Mechanisms, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.12, December 2008.

[4] G.Thiruchelvi and J.Raja, A Survey On Active Queue Management Mechanisms, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.12, December 2008.

[5] Ashish Kumar , Ajay K Sharma , Arun Singh, Dr. B R Ambedkar,Comparison and Analysis of Drop Tail and RED Queuing Methodology in PIM-DM Multicasting, International Journal of Computer Science and Information Technologies, Vol. 3 (2) , 2012,3816 – 3820.

[6] Dr. Neeraj Bhargava, Dr. Ritu Bhargava, Anchal Kumawat, Bharat Kumar, " Performance of TCPThroughput on NS2 by Using Different Simulation Parameters", International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970) Volume-2 Number-4 Issue-6 December-2012.

[7] Mr. Ajay Singh and Dr. Pankaj Dashore, " Comparative Analysis of TCP and UDP by using NS-2", International Journal of Computer Science and Information Security (IJCSIS) Volume (1) : Issue (1) IROCS Published Online June 2013.

[8] http://en.wikipedia.org/wiki/Network_simulation.

[9] D. Lin and R. Morris. *Dynamics of Random Early Detection.* In Proceedings of ACM SIGCOMM' 97, pp. 127-137, Cannes, France, October 1997.

[10] W. Feng, D. Kandlur, D. Saha, K. Shin, *Blue: A New Class of Active Queue Management Algorithms* U. Michigan CSE-TR-387-99, April 1999.

[11] A. Demers, S. Keshav, and S. Shenker. *Analysis and Simulation of a Fair Queueing Algorithm,* In Journal of Internetworking: Research and Experience, 1, pp. 3-26, 1990.

[12] Floyd, S., and Jacobson, V., *Random Early Detection gateways for Congestion Avoidance* V.1 N.4, August 1993, pp. 397-413.

[13] Jain, R., and Ramakrishnan, K.K., *Congestion Avoidance in Computer Networks with a Congestionless Network Layer: Concepts, Goals, and Methodology,* Proc. IEEE Comp. Networking Symp., Washington, D.C., April 1988, pp.134-143.

[14] Floyd, S., *TCP and Explicit Congestion Notification*. ACM Computer Communication Review, V. 24 N. 5, October 1994, pp. 10-23.