Review On Red Algorithm Along With Active Queue Management

[1]Karanpreet Singh Virk, [1]Jatinder Singh, [1]Lovepreet Kaur, [1]Abhinav Bhandari
[1]*University College of Engineering, Department of Computer Engineering, Punjabi University, Patiala, Punjab, India*

Abstract-This review paper presents Random Early Detection (RED) gateways for congestion avoidance in packet-switched networks. The gateway detects incipient congestion by computing the average queue size. The gateway could notify connections of congestion either by dropping packets arriving at the gateway or by setting a bit in packet headers. When the average queue size exceeds a preset threshold, the gateway drops or *marks* each arriving packet with a certain probability, where the exact probability is a function of the average queue size. RED gateways keep the average queue size low while allowing occasional bursts of packets in the queue. During congestion, the probability that the gateway notifies a particular connection to reduce its window is roughly proportional to that connection's share of the bandwidth through the gateway. RED gateways are designed to accompany a transport-layer congestion control protocol such as TCP. The RED gateway has no bias against bursty traffic and avoids the global synchronization of many connections decreasing their window at the same time. Simulations of a TCP/IP network are used to illustrate the performance of RED gateways.

Keywords- *AQM, DDoS, RED Queue Modeling, TCP Flow, RED Parameter tuning.*

I. INTRODUCTION

The traditional role of Active Queue Management (AQM) in IP networks was to complement the work of end-system protocols such as the Transmission Control Protocol (TCP) in congestion control so as to increase network utilization, and limit packet loss and delay. During the earlier days of IP networks, the network traffic consisted mainly of bulk data transfers. The volume of web traffic was gradually increasing. The first formal and full proposal of an AQM scheme was Random Early Detection (RED), introduced by[1] in 1993. What followed was a plethora of AQM schemes proposed in the research literature, many of which sort to improve upon the RED algorithm itself in one aspect or another. There were, however, AQM schemes that were completely new. Additionally, there was also work that consisted primarily of a rigorous analysis of RED and which consequently highlighted its drawbacks. The design of RED and many of its variants, though intuitive, has been, for the most part, heuristic. As a result, parameter-tuning has been one of their main limitations. Some researchers discovered that by applying more formal andrigorous techniques as found in control theory (whether it be classical control, modern control, optimal control or nonlinear control), this limitation may be alleviated if not eliminated. Other researchers have also invented AQM schemes based upon optimization techniques in the context of congestion control. With the increasingly rapid march to convergence, i.e., data, voice, video and mobility, supported by a common IP platform that is shared by a growing heterogeneous set of communication technologies, the primary focus has shifted from congestion control (though still very important) to the more holistic theme of quality-of-service (QoS) provisioning.

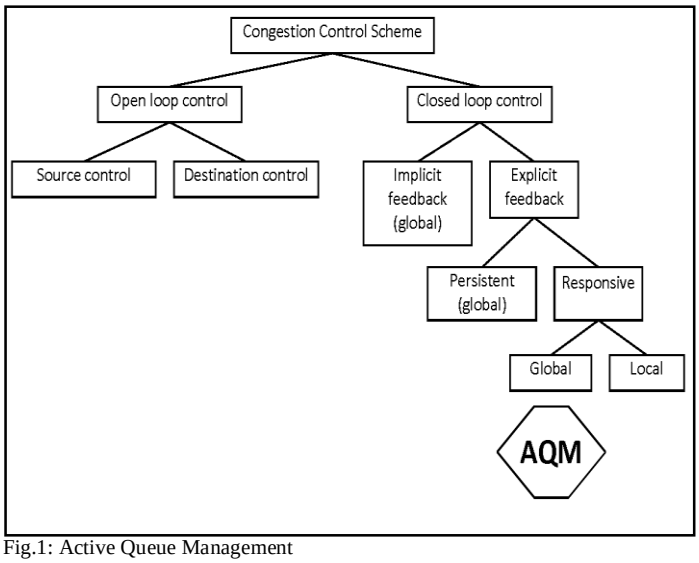

Fig.1: Active Queue Management

The main thrust of the latter is to have the network simultaneously and efficiently service the diverse requirements of the different types of traffic flows. In this new (and broader) context, the role of AQM is to serve as a mechanism for service differentiation. In the DiffServ architecture, in particular, it works in conjunction with other QoS mechanisms such as traffic conditioning and packet scheduling so that their combined effect would be to, in an average sense, distinguish one network service from another in terms of overall end-to end delay, delay variation or jitter,

packet loss and bandwidth according to mutually agreed upon service level agreements (SLAs). Based on the current specifications for DiffServ, the main candidate AQM scheme is based on RED (specifically RIO-C (RED In/Out and Couple)) having a different set of parameter values for each drop precedence. However, it may be beneficial to capitalize on the vast AQM research that already exists, exploring those feasible alternative schemes and approaches that can be used in the DiffServ context so as to improve network performance and QoS.

II. DESIGN GUIDELINES

This section summarizes some of the design goals and guidelines for RED gateways. The main goal is to provide congestion avoidance by controlling the average queue size. Additional goals include the avoidance of global synchronization and of a bias against bursty traffic and the ability to maintain an upper bound on the average queue size even in the absence of cooperation from transport layer protocols. The first job of a congestion avoidance mechanism at the gateway is to detect incipient congestion. As defined in [8], a congestion avoidance scheme maintains the network in a region of low delay and high throughput. The average queue size should be kept low, while fluctuations in the actual queue size should be allowed to accommodate bursty traffic and transient congestion. Because the gateway can monitor the size of the queue over time, the gateway is the appropriate agent to detect incipient congestion. Because the gateway has a unified view of the various sources contributing to this congestion, the gateway is also the appropriate agent to decide which sources to notify of this congestion. In a network with connections with a range of roundtrip times, throughput requirements, and delay sensitivities, the gateway is the most appropriate agent to determine the size and duration of short-lived bursts in queue size to be accommodated by the gateway. The gateway can do this by controlling the time constants used by the low-pass filter for computing the average queue size. The goal of the gateway is to detect incipient congestion that has persisted for a "long time" (several roundtrip times). The second job of a congestion avoidance gateway is to decide which connections to notify of congestion at the gateway. If congestion is detected before the gateway buffer is full, it is not necessary for the gateway to drop packets to notify sources of congestion. In this paper, we say that the gateway marks a packet, and notifies the source to reduce the window for that connection. This marking and notification can consist of dropping a packet, setting a bit in a packet header, or some other method understood by the transport protocol. The current feedback mechanism in TCP/IP networks is for the gateway to drop packets, and the simulations of RED gateways in this paper use this approach. One goal is to avoid a bias against bursty traffic. Networks contain connections with a range of burstiness, and gateways such as Drop Tail and Random Drop gateways have a bias against bursty traffic. With Drop Tail gateways, the more bursty the traffic from a particular connection, the more likely it is that the gateway queue will overflow when packets from that connection arrive at the gateway [7]. Another goal in deciding which connections to notify of congestion is to avoid the global synchronization that results from notifying all connections to reduce their windows at the same time. Global synchronization has been studied in networks with Drop Tail gateways and results in loss of throughput in the network. Synchronization as a general network phenomena has been explored in [8]. In order to avoid problems such as biases against bursty traffic and global synchronization, congestion avoidance gateways can use distinct algorithms for congestion detection and for deciding which connections to notify of this congestion. The RED gateway uses randomization in choosing which arriving packets to mark; with this method, the probability of marking a packet from a particular connection is roughly proportional to that connection's share of the bandwidth through the gateway. This method can be efficiently implemented without maintaining per-connection state at the gateway. One goal for a congestion avoidance gateway is the ability to control the average queue size even in the absence of cooperating sources. This can be done if the gateway drops arriving packets when the average queue size exceeds some maximum threshold (rather than setting a bit in the packet header). This method could be used to control the average queue size even if most connections last less than a roundtrip time (as could occur with modified transport protocols in increasingly high speed networks), and even if connections fail to reduce their throughput in response to marked or dropped packets.

III. RED ALGORITHM

The RED algorithm calculates the average queue size using a low pass filter with an exponential weighted moving average. The average queue size is compared to two thresholds: a minimum and a maximum threshold. When the average queue size is less than the minimum threshold, no packets are marked. When the average queue size is greater than the maximum threshold, every arriving packet is marked. If marked packets are, in fact, dropped or if all source nodes are cooperative, this ensures that the average queue size does not significantly exceed the maximum threshold. When the average queue size is between the minimum and maximum thresholds, each arriving packet is marked with probability pa, where pa is a function of the average queue size *avg*. Each time a packet is marked, the probability that a packet is marked from a particular connection is roughly proportional to that connection's share of the bandwidth at the router. The detailed algorithm for RED. Essentially, RED algorithm has two separate parts. One is for computing the average queue size, which determines the degree of burstiness that will be allowed in the router queue. It takes into account the period when the queue is empty (the idle period) by estimating the number m of small packets that could have been transmitted by the router during the idle period. After the idle period, the router computes the average queue size as if *m* packets had

arrived to an empty queue during that period. The other is used to calculate the packet-marking probability and then determine how frequently the router marks packets, given the current level of congestion. The goal is for the router to mark packets at fairly evenly spaced intervals, in order to avoid biases and avoid global synchronization, and to mark packets sufficiently frequently to control the average queue size.

Initial ization
avg <- 0
count <- -1
For each packet arrival
if the queue is non-empty
avg ←$(1-w_q)*avg+w_q*q$
else
m←*f* (time-q_time)
avg←$(1-\omega q)m\times avg$
If minth ≤ avg < maxth
Increment count
$P_b=(avg-minth)/(max- minth)*max_p$
$P_a= p_b/(1-count)*p_b$
with probability pa :
mark the arriving packet
count <- 0
Else if maxth < avg
mark the arriving packet
count <- 0
Else count <- -1
When queue become empty
q_time <- time

Notations:
[1] Saved Variables:
avg: average queue size
q_time: start of the queue idle time
count: packets since last marked packet
[2] Fixed Parameters:
ω_q *:* queue weight
minth: minimum threshold for queue
maxth: maximum threshold for queue
maxp: maximum value for p_b
[3] Other:
p_a: current packet-marking probability
q: current queue size
Time: current time

As avg varies from minth to maxth, the packet-marking probability pb varies linearly from 0 to maxp :

$$P_b= (avg-minth)/(max-minth)*max_p \quad (1)$$

The final packet-marking probability *pa* increases slowly as the count increases since the last marked packet:

$$P_a=P_b/(1-count)*P_b \quad (2)$$

this ensures that the gateway does not wait too long before marking a packet. The gateway marks each packet that arrives at the gateway when the average queue size avg exceeds maxth.

IV. QUEUEING MODELING FOR RED

Various analysis approaches have been proposed to model RED mechanism and evaluate its performance. Three different models are to be examined. In this section, classic queueing theory issued to study the benefits (or lack thereof) brought about by RED. In the subsequent section, a different feedback control models will be discussed. Thomas Bonald et al.3 use classic queueing theory to evaluate RED performance and quantify the benefits (or lack thereof) brought about by RED. Basically, three major aspects of RED scheme, namely the bias against bursty traffic, synchronization of TCP flows, and queuing delays, are studied in details and compared with those of Tail Drop scheme to evaluate the performance of RED.

A. Bias against Bursty Traffic

We consider a router with buffer size of K packets. A typical drop functions for RED scheme and Tail Drop scheme are listed below. Drop function of RED and Tail Drop scheme.

Drop function for RED scheme*:*

$$(avg – minth)/ (max – minth)*max_p = p_b \quad (3)$$

1 if avg < minth

D(avg) = 0 if avg > maxth

Drop function for TAIL DROP scheme:
0 if q < maximum buffer size
d_q= 1 if q > maximum buffer size (4)

V. RED IMPACT ON INTERNET FLOW

Usually, TCP connections/flows can be modeled as bursty traffic, while UDP-based application can be considered as smooth traffic. Since TCP has congestion control mechanism implemented at the end host, TCP connection should respond to the packet dropping after a round trip time (RTT). Meanwhile, UDP host neglects the packet loss and keeps pumping data into network and let the upper layer application take care of congestion and perhaps further retransmission. However, this does not necessarily mean that RED algorithm has no impact on UDP application. In fact, since RED algorithm is implemented in routers instead of end hosts, it has impact on all kinds of Internet traffic, including both the TCP and UDP connections. So, it makes sense to compare how different the influence RED algorithm has on TCP flows from that on UDP-based applications. The key observations are listed below. First, the overall loss rate suffered by TCP connections when going from Tail Drop to RED will not change much, but that the loss rate suffered by UDP/IP

telephony applications (whether they are rate adaptive or not) will increase significantly. Second, average delay suffered by the UDP packets would be much lower than with Tail Drop, which is a key benefit in telephony applications. However, the delay variance is such that the end-to-end delay, including the playout delay at the destination, does not reflect the gain RED brought to the mean delay. We can expect the audio quality perceived at the destination to be mediocre at best.

VI. MEASURE AVERAGE QUEUE LENGTH

The RED gateway uses a low-pass filter to calculate the average queue size. Thus, the short-term increases in the queue size that result from bursty traffic or from transient congestion do not result in a significant increase in the average queue size. The low-pass filter is an exponential weighted moving average (EWMA):

$$Avg <- (1-w_q)avg + w_q.q \quad (5)$$

The weight determines the time constant of the low-pass filter. The following sections discuss upper and lower bounds for setting . The calculation of the average queue size can be implemented particularly efficiently when is a (negative) power of two.

A. An upper bound for w_q

If w_q is too large, then the averaging procedure will not filter out transient congestion at the gateway.

Assume that the queue is initially empty, with an average queue size of zero, and then the queue increases from 0 to L packets over packet arrivals. After the Lth packet arrives at the gateway, the average queue size avg_l is

$$Avg_l = \sum_{i=1}^{L} iw_q (1-w_q)^{L-I}$$
$$=L+1+\{(1+w_q)^{L+1}-1\}/w_q \quad (6)$$

Figure 2 shows the average queue size avg_l for a range of values for w_q and L. The x-axis shows w_q from0.001 to 0.005, and the -axis shows from 10 to 100.For example for w_q=0.001 after a queue increase from 0 to 100 packets, the average queue size avg_{100} is 4.88 packets. Given a minimum threshold and given that we wish to allow bursts of packets arriving at the gateway, then w_q should be chosen to satisfy the following equation for $avg_l < min_{th}$:

$$L+1+\{(1-w_q)^{L+1}-1/w_q < min_{th} \quad (7)$$

Given min_{th}=5 and L=50, for example it is necessary to choose w_q=0.0042.

This derivation uses the following identity

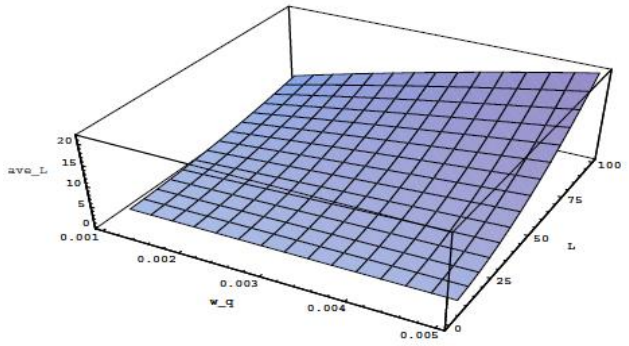

Fig.2: Average queue size for w_q

B. A lower bound for w_q

RED gateways are designed to keep the calculated average queue size average below a certain threshold. However, this serves little purpose if the calculated average is not a reasonable reflection of the current average queue size. If w_q, is set too low, then average responds too slowly to changes in the actual queue size. In this case, the gateway is unable to detect the initial stages of congestion. Assume that the queue changes from empty to one packet, and that, as packets arrive and depart at the same rate, the queue remains at one packet. Further assume that initially the average queue size was zero. In this case it takes $-1/\ln(1-w_q)$ packet arrivals (with queue remaining at one) until the average queue size avg reaches 0.63=1-1/e [35]. For w_q=0.001, this takes 1000 packet arrivals; for w_q = 0.002, this takes 500 packet arrivals; for w_q=0.003, this makes 333 packet arrivals

C. Setting minth and maxth

The optimal values of min_{th} and max_{th} depends on the desired average queue size.. If the typical traffic is fairly bursty, then min_{th} must be correspondingly large to allow the link one packet would result in unacceptably low link utilization. The discussion of the optimal average queue size for a particular traffic mix is left as a question for future research utilization to be maintained at an acceptably high level. For the typical traffic in our simulations, for connections with reasonably large delay-bandwidth products, a minimum threshold of one packet would result in unacceptably low link utilization. The discussion of the optimal average queue size for a particular traffic mix is left as a question for future research. The optimal value for max_{th} depends in part on the maximum average delay that can be allowed by the gateway. The RED gateway functions most effectively when max-min is larger than the typical increase in the calculated average queue size in one roundtrip time. A useful rule-of-thumb is to set max_{th} to at least twice min_{th} we compare two methods for calculating the final packet marking

probability, and demonstrate the advantages of the second method. In the first method, when the average queue size is constant the number of arriving packets between marked packets is a geometric random variable;in the second method the number of arriving packets between marked packets is a uniform random variable. The intial packet dropping capability is calculated as:

$$p_b \leftarrow max_p(avg\text{-}min_{th})/(max_{th}\text{-}min_{th})$$

The parameter max_{th} gives the maximum value for the packet-marking probability p_b achieved when the

average queue size reaches the maximum threshold.

Method 1. Geometric random variables

In Method 1, let each packet be marked with probability p_b Let the intermarking time X be the number of packets that arrive, after a marked packet, until the next packet is marked. Because each packet is marked with probability p_b:

$$Prob\ [X=n] = (1-p_b)^{n-1}p_b \quad (8)$$

Thus with Method 1, is a geometric random variable with parameter p_b and $E[X]=1/p_b$ With a constant average queue size, the goal is to mark packets at fairly regular intervals. It is undesirable to have too many marked packets close together, and it is also undesirable to have too long an interval between marked packets. Both of these events can result in global synchronization, with several connections reducing their windows at the same time, and both of these events can occur when X is a geometric random variable.

Method 2. Uniform random variable

A more desirable alternative is for X to be a uniform random variable from $\{1,2\ldots\ldots 1/p_b\}$. This is achieved if the marking probability for each arriving packet is $p_b/(1\text{-}count.p_b)$, where count is the number of unmarked packets that have arrived since last marked packet.

$$prob[x=n] = p_b/1\text{-}(n\text{-}1)p_b\ \pi(1\text{-}p_b/1\text{-}ip_b)=p_b \quad (9)$$
$$= p_b \text{ for } 1 < n < 1/p_b.$$

And

$$Prob[X=n] = 0 \text{ for } n > 1/p_b. \quad (10)$$

VII. EVALUATION OF RED GATEWAYS

Several general goals have been outlined for congestion voidance schemes [14, 16]. In this section we describe how our goals have been met by RED gateways.

A. Congestion Avoidance

If the RED gateway in fact drops packets arriving at the gateway when the average queue size reaches the maximum threshold, then the RED gateway guarantees that the calculated average queue size does not exceed the maximum threshold. If the weight for the EWMA procedure has been set appropriately then the RED gateway in fact controls the actual average queue size. If the RED gateway sets a bit in packet headers when the average queue size exceeds the maximum threshold, rather than dropping packets, then the RED gateway relies on the cooperation of the sources to control the average queue size.

B. Appropriate time scale

After notifying a connection of congestion by marking a packet, it takes at least a roundtrip time for the gateway to see a reduction in the arrival rate. In RED gateways the time scale for the detection of congestion roughly matches the time scale required for connections to respond to congestion. RED gateways don't notify connections to reduce their windows as a result of transient congestion at the gateway.

C. No global synchronization

The rate at which RED gateways mark packets depends on the level of congestion. During low congestion, the gateway has a low probability of marking each arriving packet, and as congestion increases, the probability of marking each packet increases. RED gateways avoid global synchronization by marking packets at as low a rate as possible.

D. Simplicity

The RED gateway algorithm could be implemented with moderate overhead in current networks.

E. Maximizing global power

The RED gateway explicitly controls the average queue size. It shows that for simulations with high link utilization, global power is higher with RED gateways than with Drop Tail gateways. Future research is needed to determine the optimum average queue size for different network and traffic conditions.

F. Fairness

One goal for a congestion avoidance mechanism is fairness. This goal of fairness is not well-defined, so we simply describe the performance of the RED gateway in this regard. The RED gateway does not discriminate against particular connections For the RED gateway, the fraction of marked packets for each connection is roughly proportional to that connection's share of the bandwidth. However, RED gateways do not attempt to ensure that each connection receives the same fraction of the total throughput, and do not explicitly control misbehaving users. RED gateways provide a mechanism to identify the level of congestion, and RED gateways could also be used to identify connections using a large share of the total bandwidth. If desired, additional mechanisms could be added to RED gateways to control the throughput of such connections during periods of congestion.

Appropriate for a wide range of environments

The randomized mechanism for marking packets is appropriate for networks with connections with a range of roundtrip times and throughput, and for a large range in the number of active connections at one time. Changes in the load are detected through changes in the average queue size, and the rate at which packets are marked is adjusted correspondingly. The RED gateway's performance is discussed further in the following section. Even in a network where RED gateways signals congestion by dropping marked packets, there are many occasions in a TCP/IP network when a dropped packet does not result in any decrease in load at the gateway. If the gateway drops a data packet for a TCP connection, this packet drop will be detected by the source, possibly after a retransmission timer expires. If the gateway drops an ACK packet for a TCP connection, or a packet from a non- TCP connection, this packet drop could go unnoticed by the source. However, even for a congested network with a traffic mix dominated by short TCP connections or by non-TCP connections, the RED gateway still controls the average queue size by dropping all arriving packets when the average queue size exceeds a maximum threshold.

VIII. CHALLENGES AND ISSUES

The RED active queue management algorithm allows network operators to simultaneously achieve high throughput and low average delay. However, the resulting average queue length is quite sensitive to the level of congestion and to the RED parameter settings, and is therefore not predictable in advance. Delay being a major component of the quality of service delivered to their customers, network operators would naturally like to have a rough a priori estimate of the average delays in their congested routers; to achieve such predictable average delays with RED would require constant tuning of the parameters to adjust to current traffic conditions. Our goal is to solve this problem with minimal changes to the overall RED algorithm. To do so, we revisit the Adaptive RED proposal of Feng *et al.* from 1997 [6, 7]. We make several algorithmic modifications to this proposal, while leaving the basic idea intact, and then evaluate its performance using simulation. We find that this revised version of Adaptive RED, which can be implemented as a simple extension within RED routers, removes the sensitivity to parameters that affect RED's performance and can reliably achieve a specified target average queue length in a wide variety of traffic scenarios. Based on extensive simulations, we believe that Adaptive RED is sufficiently robust for deployment in routers.

IX. CONCLUSION

In this review paper we have reviewed that RED parameters max_{th} and min_{th} and the probability of dropping packet with random variable and uniform variable methods helps maintaining predictable queue size. But due to parameter over sensitivity RED somehow along the path fails to meet the expectation as we need to tradeoff between utilization and delay.

ACKNOWLEDGEMENT

The authors are grateful to Mr. Abhinav Bhandari ,Assistant Professor, Department of Computer Engineering ,Punjabi University , Patiala for their support and guidance.

X. REFERENCES

[1] Active queue management; a survey, Richelle Adams, member IEEE.

[2] J. Aweya, M. Ouellette, D. Y. Montuno and A. Chapman. Enhancing TCP Performance with a Loadadaptive RED Mechanism. International Journal of Network Management, V. 11, N. 1, 2001

[3] J. Aweya, M. Ouellette, D. Y. Montuno and A. Chapman. A Control Theoretic Approach to Active Queue Management. Computer Networks 36, 2001.

[4] W. Feng, D. Kandlur, D. Saha, and K. Shin. Techniques for Eliminating Packet Loss in Congested TCP/IP Network. U. Michigan CSE-TR-349-97,November 1997.

[5] W. Feng, D. Kandlur, D. Saha, and K. Shin. A Self-Configuring RED Gateway. Infocom, Mar 1999.

[6] Victor Firoiu and Marty Borden. A Study of ActiveQueue Management for Congestion Control. Infocom, pages 1435–1444, 2000.

[7] S. Floyd. RED: Discussions of Setting Parameters, November1997.http://www.aciri.org/floyd/REDparameters.txt.

[8] S. Floyd, V. Jacobson. Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on Networking (TON) August 19932309

[9] RFC: Recommendations on Queue Management and Congestion Avoidance in the Internet

[10] T. Bonald, M. May, and J. C. Bolot. Analytic evaluation of RED performance. IEEE INFOCOM 2000

[11] V. Firoiu, M. Borden. A Study of Active Queue Management for Congestion Control. IEEE INFOCOM 2000

[12] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. ACM SIGCOMM '98.

[13] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. A Stchastic Model of TCP Reno Congestion Avoidance and Control. Technical Report CMPSCI TR 99-02, Univ. of Massachusetts, Amherst, 1999.

Karanpreet Singh Virk is a M.Tech student at University college of engineering, Punjabi University Patiala. His research interests include network security, RED algorithm and DDoS attacks.

Jatinder Singh Sidhu is M.Tech student at University College of engineering, Punjabi university Patiala. His research interests include network security, Intrusion detection, Wireless Sensor Network.

Lovepreet Kaur is a M.Tech student at University college of engineering at Punjabi university Patiala. Her research interests include network security and DDoS attacks.

Abhinav Bhandari is Asstt. Professor at University College of engineering, Punjabi university Patiala. His research interests include Genetical Algorithm and Soft Computing Algroithms in Network Security