ENHANCE PERFORMANCE OF TCP CONGESTION CONTROL USIG MODIFIED RED OVER WIRELESS NETWORK

Ankita Rathod[1] , A/Prof. Ravi Raval[2]

[1]Student, [2]A/Prof. , Computer Engineering

[1]Master of enginnering in Information Technology , [1]SOCET , [1]Ahmedabad , [1]India

Abstract : Faster data communication is today requirement in any wireless system. TCP is most widely used transport protocol. TCP provides reliable delivery of data in wired network. In wireless network, congestion event and non-congestion event are present. Traditional TCP always considers as packet is lost due to congestion. It fails to perform well in wireless network. Many TCP variants have been proposed for congestion control but they cannot distinguish reason of dropping packets that is either due to congestion or due to bit error. For Congestion reduction router queue play important role, like drop tail and RED. Here, study and comparative analysis of TCP variants has been done with different parameters like throughput, end to end delay, number of dropped packets and packet delivery fraction. By analyzing these comparisons conclude that NewReno give batter result With RED queue as compare to other but there is no solution for bit error and high priority packet drop over RED. It reduces congestion window every time when their congestion error but when there is a bit error then no need to reduce the transmission rate. MOD_TCPNR is capable to reduce the unnecessary reduction of congestion window and retransmissions caused by bit error or random error. MOD_TCPNR uses the ECN indication of TCP header for distinguishes issue of congestion from bit error and Congestion notification of high priority traffic. Simulation results shows that MOD_TCPNR gives better results compare to NewReno. Simulation has been performed in ns2.35.

Keywords: TCP, Congestion window, congestion error, Bit error.

1. INTRODUCTION

Now a day internet is highly used in every sector like government offices, schools, colleges, hospitals and other places. There are two types of networks first one is wired network and second one is wireless network. Development in wireless network is increased day by day which is mainly caused by two factors. One is “Anytime and anywhere” access and second is the significant use of communication. Today the internet is highly use for searching and exchange the information. Wireless network provides quality of service, effectiveness and highly secure. But wireless networks still are not ready for give appropriate result. In wireless networks data links have high error rates, lower bandwidth and longer delay. Wireless compatibility show up in many structures and hidden vast scope is throughput and separation.

Network congestion [9]–[13] is an important factor affecting network quality of service [14]–[17]. According to the work in [18], congestion occurs when the total demand for a resource, e.g., the link bandwidth, exceeds the capacity of the resource. Results of congestion include a high delay, wasted resources, and even global synchronization, i.e., the throughput drops to zero, and the response time tends to infinity [19]. The aim then would be to control congestion or, more ideally, avoid congestion. A congestion control scheme based on active queue management (AQM) has become a research hot spot in the industry, and the AQM mechanism is recommended on Internet routers to achieve the following goals: managing queue lengths to absorb short-term congestion (e.g., bursts), providing a lower interactive delay, and avoiding global synchronization [20]. Among the AQM mechanisms, the most typical scheme is random early detection (RED) [21] proposed by Floyd and Jacobson. The RED algorithm functions by detecting incipient congestion and notifying the transmission control protocol (TCP) by probabilistically dropping packets before the queue when a router fills up. Briefly, the algorithm works by maintaining an average queue size. As the average queue size varies between the minimum and maximum thresholds, the packet dropping probability linearly changes between zero and maximum drop probability *P*max. Thus, the packet dropping probability function is linear to the change of the average queue size. If the average queue size exceeds the maximum threshold, all arriving packets are dropped. Since the packet dropping mechanism is based on the moving average algorithm, RED can control the transient congestion by absorbing arrival rate fluctuations. Although RED is a significant improvement over simple Drop Tail [22] that simply drops all incoming packets when a Drop Tail queue is full, RED is particularly sensitive to the traffic load and the parameters of the scheme itself [23], [24].

Additionally, from the viewpoint of the network layer, the queue size is a decisive influence on the network throughput. More specifically, when an AQM scheme such as RED is deployed on Internet routers, since a network can provide lower delay service with a lower queue size, then its network throughput also lowers. Although Drop Tail can provide the network with a higher throughput before deadlock, it removes the ability to provide lower delay service [25]. Therefore, through analysing RED, this paper presents a linear congestion control Scheme Mod RED with its packet dropping probability function divided into three sections, maintaining a predictable average queue length to achieve better balance for the delay and the throughput between low and high traffic loads and with the performance of ModRED and that of RED compared using NS2.35 simulation [4].

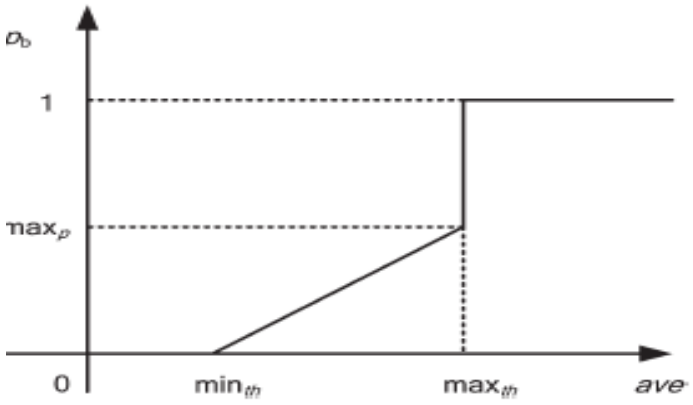


Fig. 1. RED's packet dropping probability curve.

This paper is organized as follows. The problems statement is presented in section II. The Mod RED scheme is presented in Section III. The proposed scheme's performance is assessed in Section IV using NS2.35 simulation. Section V concludes this paper. Section VI is future scope.

2.PROBLEM INTRODUCED

RED, a queue-based AQM mechanism and designed heuristically, is the first developed AQM scheme to be deployed in TCP/Internet Protocol networks for the replacement of Drop Tail. The initial objectives of RED is to detect incipient congestion, to achieve fairness among flows with differing levels of burstiness, to control the queue lengths to low values to minimize queuing delay, to prevent the correlation of packet drops and global synchronization, to minimize packet loss, and to provide a high link utilization [26].

The RED scheme drops packets with a certain probability by computing the average queue length (ave) to notify traffic sources about the early stages of network congestion. The average queue length is calculated as the result of the exponentially weighted moving average (EWMA) [27], which really acts as a low-pass filter that smoothes out the burstiness of the instantaneous queue length [28] to provide a more stable measure. The degree of smoothing is determined by weighting factor *wq*.

In addition, the average queue length is expressed as

$$ave = (1 - wq)ave + wqq. \quad (1)$$

In the aforementioned formula, q is the instantaneous queue length, and $wq \in [0, 1]$ is the weighting factor. In addition to EWMA weight *wq*, RED has three more parameters, i.e., minimum threshold minth, maximum threshold maxth, and the maximum dropping probability *P*max at maxth. If the average queue length is below minth, RED drops no packets. However, if the average queue length increases above minth but is below maxth, RED drops incoming packets with a probability proportional to the average queue length linearly. When the average queue length exceeds maxth, all the arriving packets are dropped. Fig. 1 depicts the RED control function curve, which is the packet dropping probability as a function of the average queue length [4].

Since the average queue length may reflect the network congestion, the packet dropping probability being the function of the average queue length is calculated by [4]

$$P_b = \begin{cases} 0, & ave \in [0, min_{th}] \\ max_p (ave - min_{th} / max_{th} - min_{th}), & ave \in [min_{th}, max_{th}] \\ 1, & ave \in [max_{th}, +\infty] \end{cases}$$

Although RED can eliminate Drop Tail defects such as deadlock, full queues, and global synchronization, there are still many deficiencies in RED. Forced drops or link underutilization will occur without reasonably set RED parameters. Multiple TCP global synchronization will also occur in certain traffic load environments, leading to queue oscillation, throughput reduction, and an intensified delay jitter [4].

Initially client enters into slow start phase of TCP. TCP New Reno cannot differentiate reason of segment loss through either congestion or bit error on wireless link. TCP synchronization problem still present with Drop Tail queue. High priority traffic drop unwantedly. RED queue has limitation of drop high IP precedence and different types of traffic.TCP New Reno is infers that all packet lost is due to congestion. In both the cases, size of congestion window is half. It might be unnecessary cut down the size of congestion window is half when loss is not due to congestion [4].

3. PROPOSED WORK

Here proposed new approach for finding reason of loss with the help of ECN flag of TCP Header. Mod_TCPNR use 1 bit ECN of TCP Header for strong indication of congestion. In our proposed work, distinguish drop of packets using ratio of timeouts and 3dupack. Base on this rate, find reason of dropping packets. Timeout occurs due to congestion at router side and 3 DUPACKs is received at receiver side due to bit error.

In TCP handle drop system, a sender sends a solitary fragment, and if the sender gets a fruitful affirmation from the beneficiary, it at that point sends two portions (that is, a "windows estimate" of 2). On the off chance that those two sections were recognized effectively, the sender sends four portions, expanding the window measure exponentially.

However, if one of the segments is dropped, the TCP flow goes into TCP slow start, where the window size is reduced to 1. The TCP flow then exponentially increases its window size until the window size reaches half of the window size when congestion originally occurred. At that point, the TCP flow's window size increases linearly.

TCP slow start is relevant to QoS, because when an interface's output queue is full, all newly arriving packets are discarded (that is, "tail dropped"), and all of those TCP flows simultaneously go into TCP slow start.

Note that the process of multiple TCP flows simultaneously entering TCP slow start is called global synchronization or TCP synchronization. When TCP synchronization occurs, the link's bandwidth is underutilized, resulting in wasted bandwidth.

RED is a mechanism that randomly drops packets before a queue is full. RED increases drop rate as the average queue size increases. The parameter sensitivity of RED has been addressed by several researchers and as a result, RED has been extended and enhanced by adopting many different approaches. The basic mechanism of RED, however, still remains same.

The motivation behind Arbitrary Early Identification (RED) is to anticipate TCP synchronization by arbitrarily disposing of parcels as an interface's yield line starts to fill. How forcefully RED disposes of bundles relies upon the present line profundity.

The following three parameters influence when a newly arriving packet is discarded:

Minimum threshold

Maximum threshold

3.1 Mark Probability Denominator (MPD)

The minimum threshold specifies the number of packets in a queue before the queue considers discarding packets. The probability of discard increases until the queue depth reaches the maximum threshold. After a queue depth exceeds the maximum threshold, all other packets that attempt to enter the queue are discarded.

However, the probability of packet discard when the queue depth equals the maximum threshold is 1/(MPD). For example, if the mark probability denominator were set to 10, when the queue depth reached the maximum threshold, the probability of discard would be 1/10 (that is, a 10 percent chance of discard).

On entry of every bundle, RED entryways ascertain normal line estimate (avg) utilizing Exponential Weighted Moving Normal (EWMA). In the event that avg is not exactly minth, the bundle is enqueued. On the off chance that avg is more than maxth, the parcel is dropped. Be that as it may, if avg is amongst minth and maxth, the bundle is dropped arbitrarily with a specific likelihood. The accompanying conditions demonstrate avg and bundle drop likelihood (pd) estimation of RED individually:

- **For finding average queue length**

$$\text{Avg queue} = (1-w_q) * \text{avg} + w_q * q_L$$

where $w_q \in [0,1]$ is the weighting factor

q_L = queue length

avg = previous avg queue

where oldavg is the average queue size during previous packet arrival; cur_q is the current queue size and Wq is time constant for avg queue size estimator for one second, so base on

C = link capacity calculate Wq in this manner.

$$w_q = 1 - exp(-1/C)$$

$$P_d = \begin{cases} 0, & ave \in [0, min_{th}] \\ max_p\,(ave - min_{th} / max_{th} - min_{th}), & ave \in [min_{th}, min_{th} + \Delta] \\ max_p\,(ave - min_{th} / max_{th} - min_{th}), & ave \in [min_{th} + \Delta, min_{th} + 2\Delta] \\ max_p\,(ave - min_{th} / max_{th} - min_{th}), & ave \in [min_{th} + 2\Delta, max_{th}] \\ 1, , & ave \in [max_{th}, +\infty] \end{cases}$$

The probability with which a packet is dropped is a linear function of avg. Hence when avg varies from minth to maxth, the drop probability varies from 0 to maximum drop probability maxp. If avg increases above maxth, drop probability becomes 1 i.e. all incoming packets are dropped. It is observed that sharply increasing the drop probability to 1 when avg crosses maxth results in high number of packet drops. Hence, a modified RED known as Gentle RED (GRED) is proposed by Floyd that varies drop probability from maxp to 1 when avg varies from maxth to twice maxth so as to reduce the number of packet drops.

Modified RED follows Additive Increase Multi-plicative Decrease (AIMD) policy to vary maxp.

Algorithm:

Every interval seconds;
If (avg > target and Maxp ≤ 0.5)
Increase Maxp:
Maxp ← Maxp + α;
Elseif (avg < target and Maxp ≥ 0.01)
Decrease Maxp:
Maxp ← Maxp * β;

Variable ;
Avg: average queue size

Fixed Parameters;
Interval; time; 0.5 seconds
Target; target of avg;
[Min + 0.4 * (Max – Min), Min + 0.6 * (Max – Min)]
α : increment ; min (0.01, Maxp/4)
β: decrease factor; 0.9

In this algorithm, use AIMD mechanism to vary Maximum probability of drop, so we can manage different TCP traffic with control window size. Proposed Algorithm used different min and max threshold. Each time define average queue length with variance of maximum probability.

At receiver end Mod_TCPNR check packet drop reason, either due to bit error or due to congestion. For that Mod_TCPNR observe consicutive packets arrival, if arrival rate is same for all incoming packet then no need to change flag status of 3rd ACK, if arrival rate is different, then set ECN bit high in ACK for congestion notification for sender. So sender reduce congestion window to half. Otherwise move to congestion avoidance phase as it is.

3.2 STEPS for FLOWCHART

Step 1: Packet Starts arriving in queue.
Step 2: Calculate min_{th} and max_{th} of queue with the help of queue length.
$min_{th} = q_{L/3}$
$max_{th} = min_{th} * 3$
Step 3: Calculate avg queue size with the help of
Avg queue = $(1-w_q) * avg + w_q * q_L$.
Step 4: On consideration of avg different conditions are checked.
Step 5: Calculate Δ for check the modified red conditions.
$\Delta = (max_{th} + min_{th}) / 3$
Step 5: In the last all the packets are dropped.
Step 6 : At the end update the cwnd (congestion window) at the source side with ECN flag.
Step 7: After Receiving the packet check the condition seq. no. < last ack.
Step 8: check Delay if there is no delay it is connection error if there is a delay it is congestion error.
Step 9 : if it is connection error than flag bit set to 1 and congestion window will be incremented.
Step 10: if it is congestion error than flag bit set to 0 and congestion window will be half.
Step 11: END.

4.ANALYSIS BY SIMULATION

4.1Network Simulator NS2.35

NS2 is open source tool. It is event driven software tool that is helpful for understanding dynamic nature of communication as well as use for simulation in wired and wireless network. Ns2 provides specification of network protocols and simulation of their corresponding behaviors. UNIX, LINUX, Windows, and MAC operating systems are platform for on which NS2 is run.

A user has required OTCL script for use NS2. An OTCL script does following tasks.

1) Initiation of event scheduler
2) By using network objects, it creates network topology.
3) Event scheduler informs sources when start transmission packet and when it will be stop

Event scheduler is one of major component in NS. The events includes packet ID which is individual for each packet with scheduled time and the pointer to an object that handles the event. Tasks performed by NS-2 event scheduler

- Managing simulation time
- Arrange event in event queue
- Execute network components in simulation.

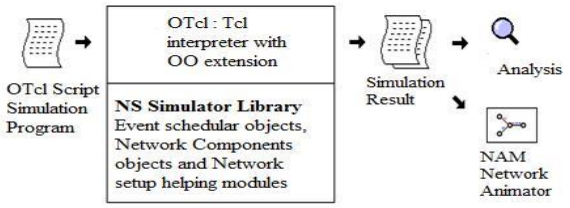


Fig 2. Network simulator [31]

4.2 Simulation Parameters

To perform the simulation and to achieve results so parameters should be set as shown in table 5.2.The total time duration is to be set 100 seconds. The boundary are of simulation is 500 × 500 meters. Simulation parameters show in below table.

Table 1. Simulation parameters

Parameters	Value
Mean packet size	1000 Bytes
Min threshold RED	5
Max Threshold RED	15
Wq time constant RED	0.001

Table2. Simulation Environment parameter

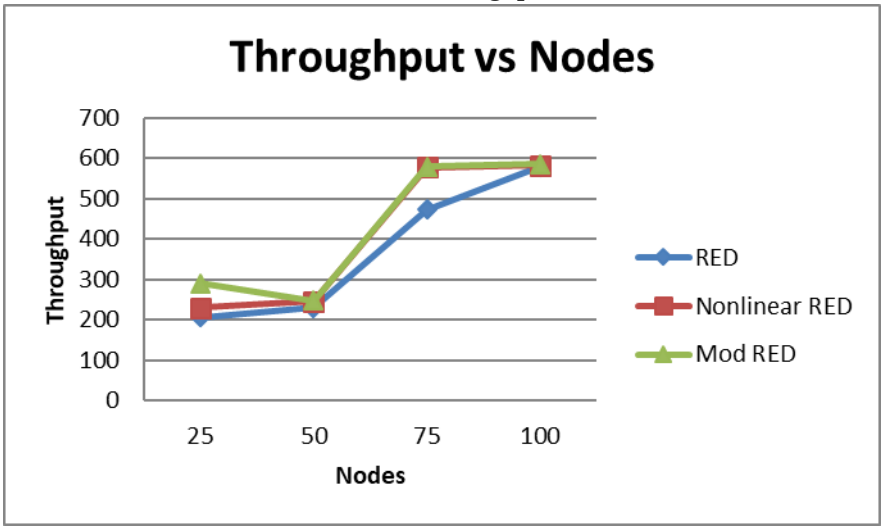
Environment Parameters	Value
Area	500 * 500 m
No. of Nodes	25,50,75,100
Time	29 sec
Antenna	Omni

4.3 Analysis Throughput v/s nodes (kbps)

Table 3. Throughput v/s nodes

Nodes	RED	Nonlinear RED	Mod RED
25	206.89	229.82	291.09
50	230.28	246.31	247.62
75	472.58	576.94	579.57
100	578.93	583.02	585.31

Chart 1. Throughput v/s nodes

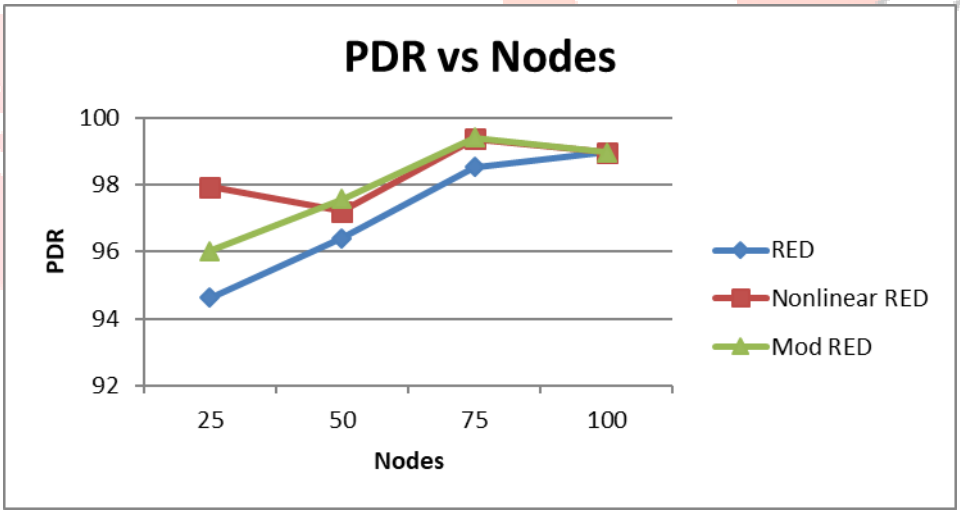


4.4 Analysis PDR v/s nodes (%)

Table 4. PDR v/s nodes

Nodes	RED	Nonlinear RED	Mod RED
25	94.63	97.94	96.02
50	96.4	97.2	97.57
75	98.52	99.37	99.41
100	98.96	98.96	98.97

Chart 2. PDR v/s nodes

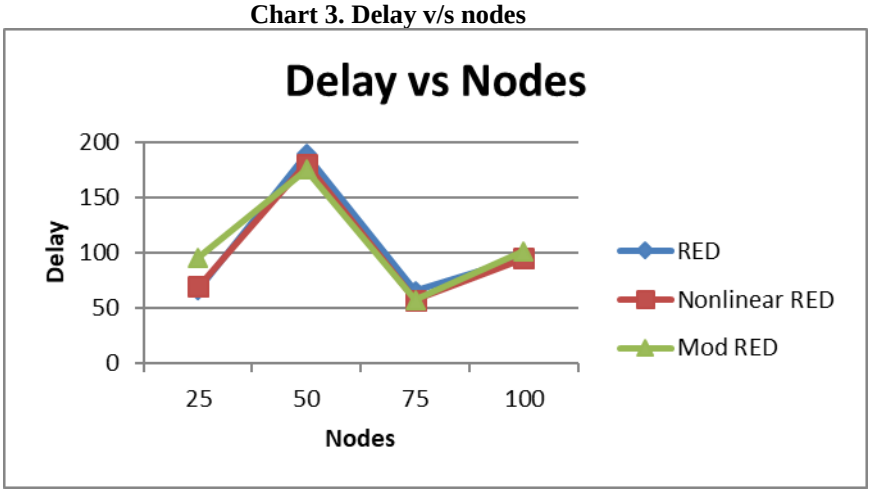


4.5 Analysis Delay v/s nodes (ms)

Table 5. Delay v/s nodes

Nodes	RED	Nonlinear RED	Mod RED
25	66.94	69.6	95.66
50	188.7	180.44	175.24
75	65.27	57.51	57.68
100	96.12	94.93	101.34

Chart 3. Delay v/s nodes

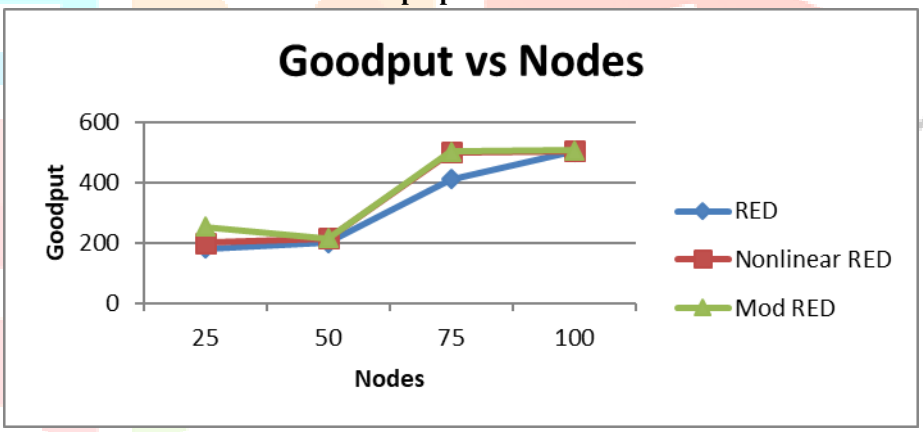


4.5 Analysis Goodput v/s nodes (kbps)

Table 6. Goodhput v/s nodes

Nodes	RED	Nonlinear RED	Mod RED
25	179.58	199.07	252.97
50	199.76	213.46	214.58
75	409.28	498.96	501.45
100	500.68	504.59	506.55

Chart 4. Goodputput v/s nodes



5> CONCLUSION:

Mod-TCPNR work on router as well as destination. Mod-TCPNR use ECN notification during congestion at router or destination end. Mod RED queue mechanism support different traffic load and vary min and max threshold of RED. Each time modify average queue length and AIMD approach define to Maximum probability of drop. So queue size does not reach at max threshold level. Using ECN notification during 3rd dupack or during congestion at router sender maintains Window size. Acknowledgment notify about maximum probability of drop, force to drop or congestion using last two bit of ECN notification. Proposed algorithm modify RED queue using additive increase and multiplicative decrease (AIMD) approach of maximum probability of drop.

At receiver end check drop reason either bit error or congestion and base on that MOD_TCPNR set ECN bit and sender maintain congestion window. Using this approach and with help of ns2 simulator, improve packet drop ratio, throughput parameters using various tcp connection.

6> FUTURE SCOPE:

- Implement other TCpP variants like TCP vegas,cubic etc.
- Reduce congestion using identify other reasons of packet drop at receiver end.
- Dynamic queue length will be implement in future.

REFERENCES AND BIBLIOGRAPHY

PAPERS:

[1] Luciano Mauro Arley Sup , Renato Mariz de Moraes , Adolfo Bauchspiess " Explicit Non-Congestion Notification: A New AQM Approach for TCP Networks " ieeexplore.ieee.org/document/7986462, 2017.

[2] Ayush Agarwal and Mohit P. Tahiliani " BCON: Back pressure based Congestion Avoidance Model for Named Data Networks " Wireless Information Networking Group (WiNG), ieeexplore.ieee.org/document/7947823, nov 2016.

[3] Marcos Talau, Mauro Fonseca , Anelise Munaretto and Emilio C. G. Wille "Early Congestion Control: A New Approach to Improve the Performance of TCP in Ad HocNetworks " ieeexplore.ieee.org/document/7810143, jan 2017.

[4] Chen-Wei Feng, Lian-Fen Huang, Cheng Xu, and Yao-Chung Chang " Congestion Control Scheme Performance Analysis Based on Nonlinear RED " ieeexplore.ieee.org/document/7018031, 2015.

[5] Shubhangi Rastogi, Hira Zaheer " Comparative analysis of queuing mechanisms: Droptail, RED and NLRED " https://link.springer.com/article/10.1007/s13278-016-0382-5,2016.

[6] Sai Prasad and Gaurav Raina "Analysis of TCP with an Exponential-RED (E-RED) queue management policy with two delays" ieeexplore.ieee.org/abstract/document/7162456,2015.

[7] Fahad Khan "A Comparative Analysis of TCP Tahoe, Reno, New-Reno, SACK and Vegas " http://www.academia.edu/5989428.

[8] MACURA, A[rijana]; MISSONI, E[duard] & KORDIC, Z[oran], "COMPARISON OF WESTWOOD, NEW RENO AND VEGAS TCP CONGESTION CONTROL" DAAAM International, Volume 23, No.1, ISSN 2304-1382 ISBN 978-3-901509-91-9, 2012

[9] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 314–329, Aug. 1998.

[10] N. F. Huang, G. Y. Jai, H. C. Chao, Y. J. Tzang, and H. Y. Chang, "Application traffic classification at the early stage by characterizing application rounds," *Inf. Sci.*, vol. 232, pp. 130–142, May 2013.

[11] L. Zhou and H. C. Chao, "Multimedia traffic security architecture for Internet of things," *IEEE Netw.*, vol. 25, no. 3, pp. 29–34, May/Jun. 2011.

[12] L. J. Zhang, D. Y. Gao, W. C. Zhao, and H. C. Chao, "A multilevel information fusion approach for road congestion detection in VANETs," *Math. Comput. Model.*, vol. 58, no. 5/6, pp. 1206–1221, Sep. 2013.

[13] Y. C. Chang, "Heterogeneous wireless sensor network with EPC network architecture for U-life environment," *J. Internet Technol.*, vol. 15, no. 4, pp. 647–655, Jul. 2014.

[14] Y. S. Yen, H. C. Chao, R. S. Chang, and A. Vasilakos, "Flooding-limited and multi-constrained QoS multicast routing based on the genetic algorithm for MANETs," *Math. Comput.Model.*, vol. 53, no. 11/12, pp. 2238–2250, Jun. 2011.

[15] C. F. Lai, H. G. Wang, H. C. Chao, and G. F. Nan, "A network and device aware QoS approach for cloud-based mobile streaming," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 747–757, Jun. 2013.

[16] C. Y. Chen, T. Y. Wu, W. T. Lee, H. C. Chao, and J. C. Chiang, "QoS based active dropping mechanism for NGN video streaming optimization," *Knowl. Eng. Rev.*, vol. 29, no. 4, pp. 484–495, Sep. 2014.

[17] N. N. Lu, H. K. Zhang, Y. C. Chang, and H. C. Chao, "IPas++: A novel accountable and scalable Internet protocol for future Internet," *J. Internet Technol.*, vol. 12, no. 5, pp. 769–780, Sep. 2011.

[18] R. Jain, "Congestion control in computer networks: Issues and trends," *IEEE Netw.*, vol. 4, no. 3, pp. 24–30, May 1990.

[19] M. Arpaci and J. A. Copeland, "An adaptive queue management method for congestion avoidance in TCP/IP networks," in *Proc. IEEE GLOBECOM*, 2000, vol. 1, pp. 309–315.

[20] J. Wang, L. Rong, and Y. Liu, "A robust proportional controller for AQM based on optimized second-order system model," *Comput. Commun.*, vol. 31, no. 10, pp. 2468–2477, Jun. 2008.

[21] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[22] R. Shorten, C. King, F. Wirth, and D. Leith, "Modeling TCP congestion control dynamics in drop-tail environments," *Automatica*, vol. 43, no. 3, pp. 441–449, Mar. 2007.

[23]S.Floyd, RED: Discussions of setting parameters, Nov. 1997. [Online]. Available: http://www.icir.org/floyd/REDparameters.txt

[24] B. Zheng and M. Atiquzzaman, "DSRED: Improving performance of active queu management over heterogeneous networks," in *Proc. IEEE ICC*, 2001, vol. 8, pp. 2375–2379.

[25] L. Hu and A. D. Kshemkalyani, "HRED: A simple and efficient active queue management algorithm," in *Proc. IEEE 13th ICCCN*, 2004, pp. 387–393.

[26] S. Ryu, C. Rump, and C. Qiao, "Advances in Active Queue Management (AQM) based TCP congestion control," *Telecommun. Syst.*, vol. 25,
no. 3/4, pp. 317–351, Mar. 2004.

[27] S.-B. Zhang, L. Gang, and K. Jun, "Study of congestion control algorithm based on neural network supervised control," *Appl. Res. Comput.*, vol. 27, no. 2, pp. 657–660, 2010.

[28] W. Wu, Y. Ren, and X. Shan, "Stability analysis on active queue management algorithms in routers," in *Proc. IEEE 9th Int. Symp. Model., Anal. Simul. Comput. Telecommun. Syst.*, 2001, pp. 125–132.

BOOKS:

[29] Andrew S. Tanenbaum and David J. Wetherall, *Computer Networks*, USA: Pearson, 2011, Pg:392-396.

[30]J.Postel, *"Transmission control protocol"*,RFC 793,Sep. 1981.

[31]Teerawat Issariyakul; Ekram Hossain, "Introduction to network simulator NS2" second edition, Springer.