Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 8, August 2013, pg.136 – 144

RESEARCH ARTICLE

P-CHOKe: A Piggybacking-CHOKe AQM Congestion Control Method

G. Sasikala[1], E. George Dharma Prakash Raj[2]

[1]Department of Computer Science, Bharathidasan University, Trichy, TamilNadu, India
[2]Department of Computer Science, Bharathidasan University, Trichy, TamilNadu, India

[1] rajkala87@gmail.com, [2] georgeprakashraj@yahoo.com

Abstract— The Active Queue Management (AQM) is a technique that consists of ECN (Explicit Congestion notifications) in internet routers. Congestion is an important issue which researcher focuses on in the TCP network environment. AQM is a router – based mechanism for early detection of congestion inside the network. This paper provides an analysis of congestion metric with flow information in queue based AQM algorithms (FRED, AdaptiveCHOKe) and compare these two algorithms and proposed a new algorithm based on AdaptiveCHOKe named as P-CHOKe.

Key Terms: - Internet; Queue; Congestion; AQM

I. INTRODUCTION

Congestion in Internet occurs when the link bandwidth exceeds the capacity of available routers. This results in long delay in data delivery and wasting of resources due to lost or dropped packets. The increased use of multimedia applications also results in bursty flows in the internet. So there is a requirement of regulating bursty flows in the very large network, the Internet. To regulate these bursty flows, resource allocation must be done efficiently. It is known from [2] routing algorithms focus on two main concepts namely Queue Management and Scheduling.

Queue Management in routers plays an important role in taking care of congestion. Two approaches are adopted to solve this problem. First one is Congestion Avoidance preventive technique, which comes into play before network is congested by overloading. Second is Congestion Control, which comes into play after congestion at a network has occurred and the network is overloaded.

A Congestion Avoidance is scheme is a proactive one that maintains the network in a state of low delay and high throughput by keeping the average queue size low to accommodate bursty traffic and transient congestion. It makes Transfer Control Protocol (TCP) responsive to congestion, as TCP will back off its transmission rate when it detects packet loss. A Congestion Control scheme is a reactive scheme that reacts after the congestion occurs. To remove such problems, Active Queue Management (AQM) has been introduced in recent years.

1.1. Active Queue Management

The essence of Internet congestion control is that a sender adjusts its transmission rate according to the congestion measure of the networks. There are two approaches to accomplish this [3]. One is a source algorithm that dynamically adjusts the transmission rate in response to the congestion along its path; the other one is a link

algorithm that implicitly or explicitly conveys information about the current congestion measure of the network to sources using that link. In the current internet, the source algorithm is carried out by AQM scheme at routers.

1.2. Queue Based AQM Schemes

In queue – based schemes, [4] congestion is observed by average or instantaneous queue length and the control aim is to stabilize the queue length. The drawback of queue – based schemes is that a backlog is inherently necessitated.

It contains two parts. First one is Congestion with flow information. Second one is congestion metric without flow information.

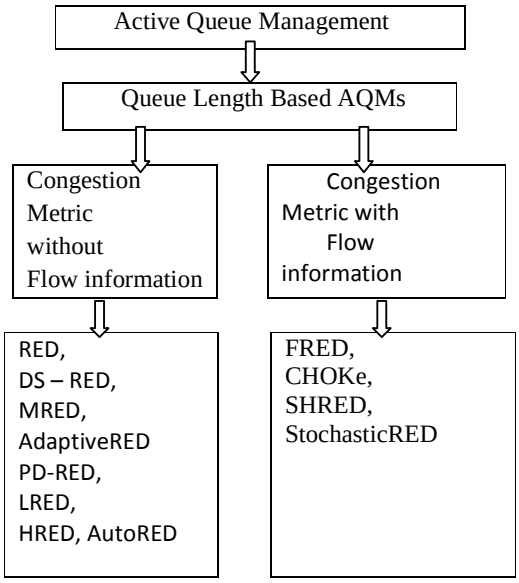

Fig.1.Category of queue based AQMs

II. EXISTING WORK

In this section the two Existing queue-length based AQM algorithms described.

2.1 FRED (Flow based Random Early Detection):-

FRED is a modified version of RED, which uses per-active-flow accounting to make different dropping decisions for connections with different bandwidth usages. FRED only keeps track of flows that have packets in the buffer, thus the cost of FRED is proportional to the buffer size and independent of the total flow numbers.

FRED has some interesting features than the RED. They are (i) penalizing non-adaptive flows by imposing a maximum number of buffered packets and surpassing their share to average per-flow buffer usage. (ii) Protecting fragile flows by deterministically accepting flows from low bandwidth connections. (iii) Providing fair sharing for large numbers of flows by using "two packet-buffer" when buffer is used up. (iv) Fixing several imperfections of RED by calculate average queue length at both packet arrival and departure (which also causes more overhead).

Two parameters introduced into FRED: minq and maxq, which are minimum and maximum number of packets that each flow is allow to buffer. FRED uses a global variable avgcq to estimate it. The average per-active-flow buffer usage. It maintains a count of buffer packets qlen.

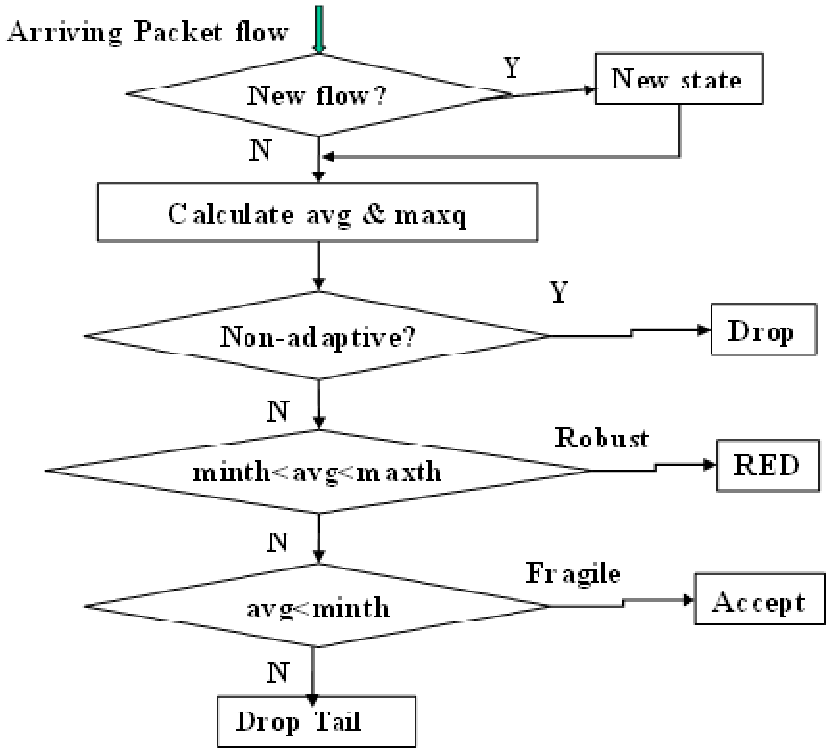

Fig.2. Flowchart for FRED algorithm

Pseudo code for FRED algorithm is as follows,

For each arriving packet P:
Calculate average queue length
Obtain connection ID of the arriving packet: flowi connectionID(P)
If flowi has no state table then
Qleni = 0
Strikei = 0
End if
Compute the drop probability like RED: p maxp
Maxth-avg
Maxh-minth
Maxq minth
If(avg_maxth) then
Maxq 2
End if
If (qlen_maxqII(avg_maxth && qleni>2_avgcq)ll(qleni_avgcq && strikei>1)) Then
ikei = strikei + 1
Drop arriving packet and return
End if
If (minth_avg<maxth) then
If (qleni_max(minq,avgcq))then
Drop packet P with a probability p like RED
End if
Else if (avg<minth) then
Return
Else
Drop packet P
Return
End if


```
If (qleni==0) then
Nactive = Nactive + 1
End if
Enqueue packet P
For each departing packet P:
Calculate average queue length
If (qleni==0) then
Nactive = Nactive – 1
Delete state table for flow i
End if
If (Nactive) then
Avgcq =avg/Nactive
Else
Avgcq=avg
End if
```

Pseudo code for FRED

2.2 *AdaptiveCHOKe*:-

The AdaptiveCHOKe is an extension of CHOKe (CHOose and Keep responsive flows, CHOose and Kill unresponsive flows). It enforces the concept of queue-based and flow information. It is desirable for AQM schemes to act without storing a loss of information otherwise it becomes an overhead and non-scalable. This algorithm also calculates the average queue size of the buffer for every packet arrival. It also indicates two thresholds on the buffer, a minimum threshold minth and a maximum threshold maxth.

The pseudo code for AdaptiveCHOKe is as follows,

```
For every packet arrival {
Calculate Qavg
If (Qavg < minth) {
Forward the new packet
}
Else
{
Select randomly a packet from the queue for their flowid
Compare arriving packet with a randomly selected packet.
If they have same flowid {
Drop both the packet
}
Else {
If (Qavg>=maxth) {
Calculate the dropping probability Pa
Drop the packet with probability Pa
}
Else
{
Drop the new packet
}
}
}
}
```


Variables:

Qavg : average queue size
Pa : current packet-marking probability
Q : current queue size
Pb : temporary marking or dropping probability
Wq : queue weight
Maxp : maximum dropping probability

Fixed Parameters:

Minth : minimum threshold for queue
Maxth : maximum threshold for queue

Pesudo code for AdaptiveCHOKe

The average queue size is compared with these thresholds for every arriving packet. If average queue size is less than minth, every arriving packet is queued. If average queue size is greater than maxth, every arriving packet is dropped. This results in queue size below maxth. When the average queue size is greater than minth, every arriving packet is compared with a randomly selected packet from the queue for their flowid. If they have the same flowid, both are dropped. Otherwise the randomly selected packet is placed in the same position in the buffer. In this algorithm, the arriving packet is dropped with a probability depending on the average queue size.

AdaptiveCHOKen which aims to protect well-behaved flows from misbehaving flow and adaptive flows from non-adaptive flows. It also obtains high utilization, low queuing delay and low packet loss with well adaptively tuned parameters.

To achieve good throughput and reasonable average queue length with RED based algorithm requires careful tuning of both Wq and maxp. Adapting maxp controls the relationship between the average queue size and the packet drop probability and helps in maintains a steady average queue size in the presence of varying traffic.

The definition of weighting parameter Wq is written as a product of three functions as follows

$$Tt = Pt\,(1-Pt)$$

$$Jt = \frac{2\,(5.923 + (Qt-Qavg, t-1)}{\text{In}\,(5.923 + (Qt-Qavg, t-1)}$$

$$Kt = \frac{1}{Bz}$$

The Tt represents the dynamics of congestion characteristics of the network, Jt represents the dynamics of congestion characteristics of the network and also time dependent function, Kt represents the time independent parameter and it allows normalization of instantaneous queue size changes with respect to the buffer size. The Fig.3. Shows the flowchart for AdaptiveCHOKe algorithm

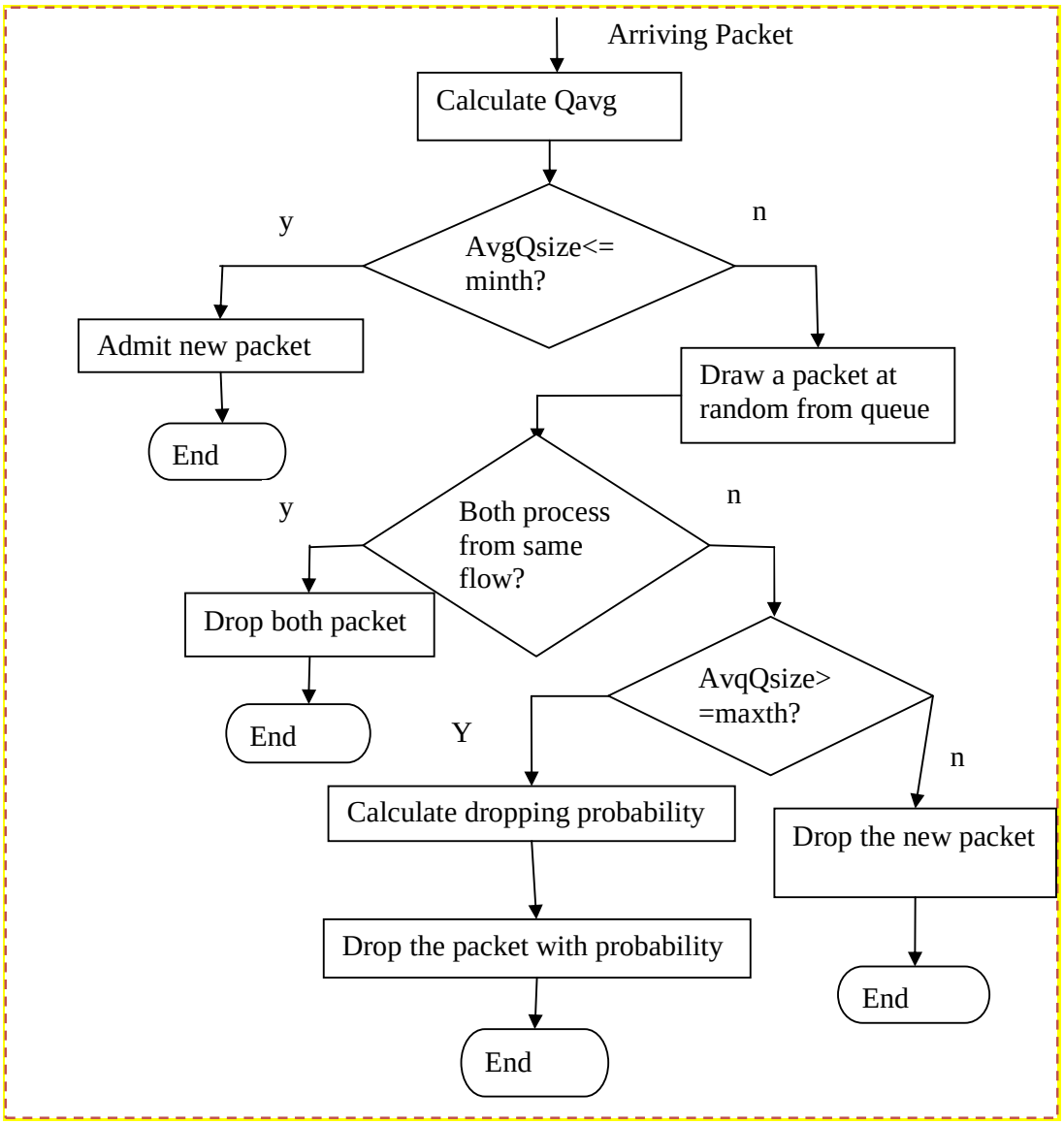

Fig.3. Flowchart for A-CHOKe

III. PROPOSED WORK

The proposed algorithm is motivated by the need for a stable operating point for the queue size and fair bandwidth allocation irrespective of the dynamic traffic and congestion characteristics of the n flows. We are motivated to identify a scheme that penalizes the unresponsive flows with the stable queue size.

P-CHOKe (PiggybackingCHOKe) enforces the concept of queue-based and flow information. The Fig.4. Shows the P-CHOKe algorithm flowchart and the following steps are involved in P-CHOKe algorithm,

Step1: Start Procedure
Step2: For every packet arrival
Step3: Forward the new packet to the gateway
Step4: Select randomly a packet from the queue for their flowid
Step5: The gateway compare arriving packet with a randomly selected packet
Step5: If there is equal flowid
Step6: Drop the packet
Step7: Otherwise forward the packets to the receiver

Step8: The receiver sends the ack to the gateway
Step9: Gateway collects all those acknowledgments
Step10: Gateway sends the ack to the receiver based on the flowid
Step11: End Procedure

In this P-CHOKe algorithm the n senders sends the packets to the gateway or router. The router collects all the packets and then it sends it to the receiver. Then the receiver sends acknowledgement to the router, it collects all the acknowledgements and then sends to the receiver based on their flowid. The Fig.4. shows the flowchart for P-CHOKe.

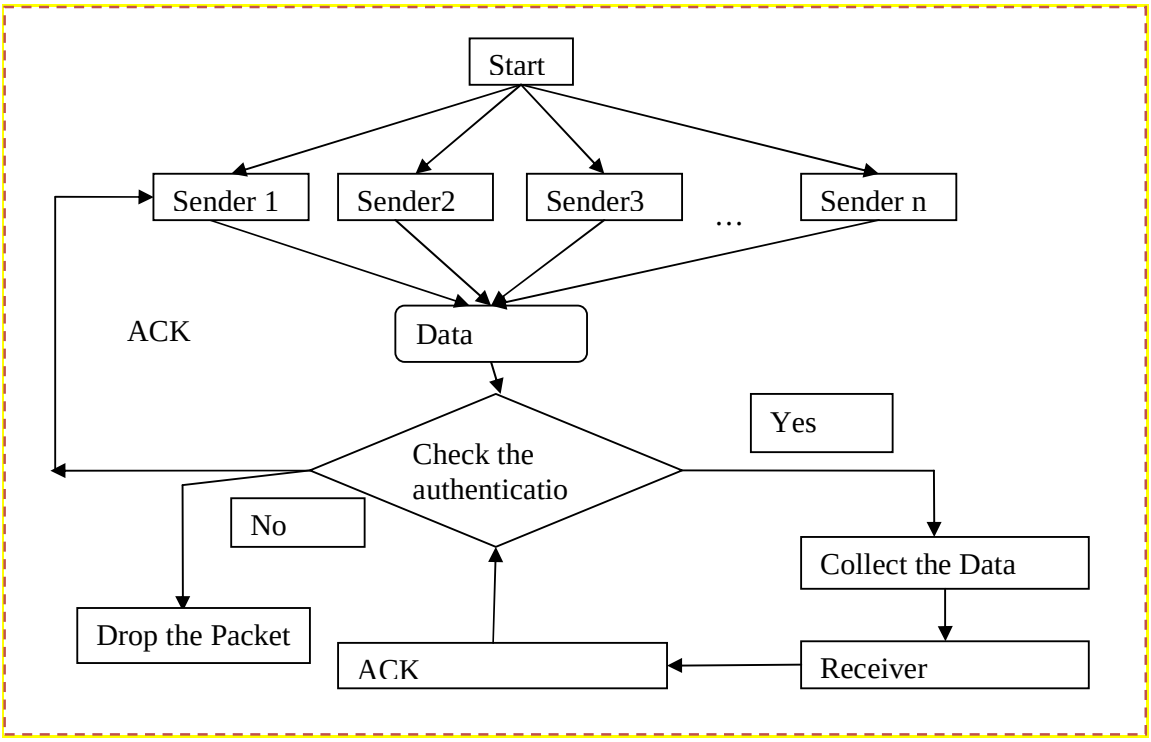

Fig.4. Flowchart for P-CHOKe

IV. RESULTS AND CONCLUSIONS

The Experimentation is done by Network Simulator2. Fig.5, Fig.6, and Fig.7 shows that the major result of the simulation. The P-CHOKe provides high link utilization (throughput) than the other two (FRED and A-CHOKe) algorithms in fig.5. In fig.6 shows that P-CHOKe provides better packet delivery ratio than the FRED and A-CHOKe. In fig.7 shows that P-CHOKe provides the low queue delay compared with FRED and A-CHOKe.

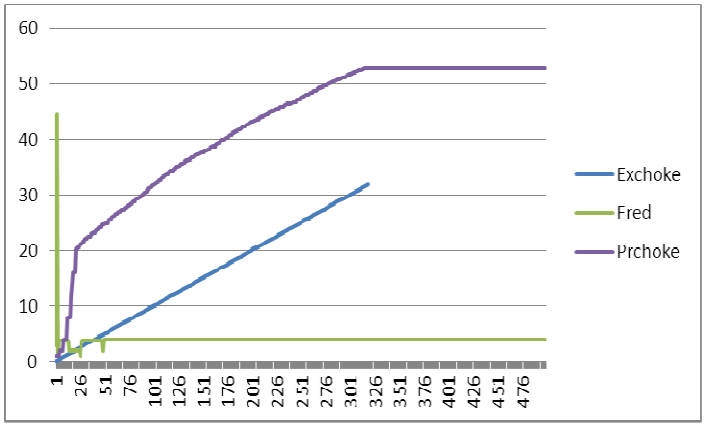

Fig.5. Throughput for FRED,A-CHOKe and P-CHOKe

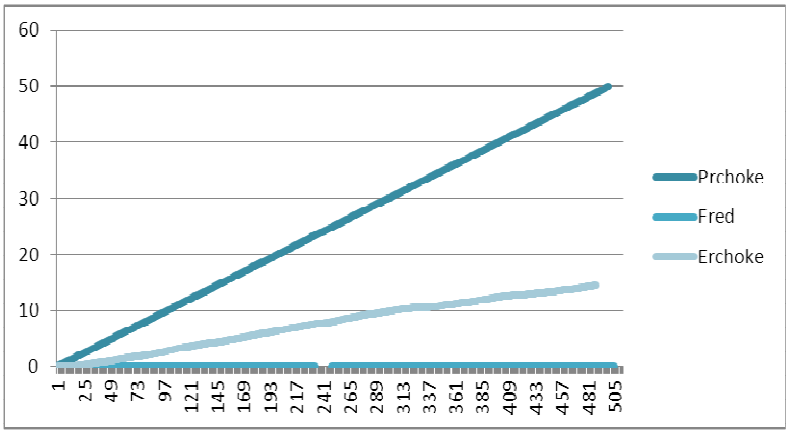

Fig.6. Packet deliver ratio for FRED, A-CHOKe and P-CHOKe

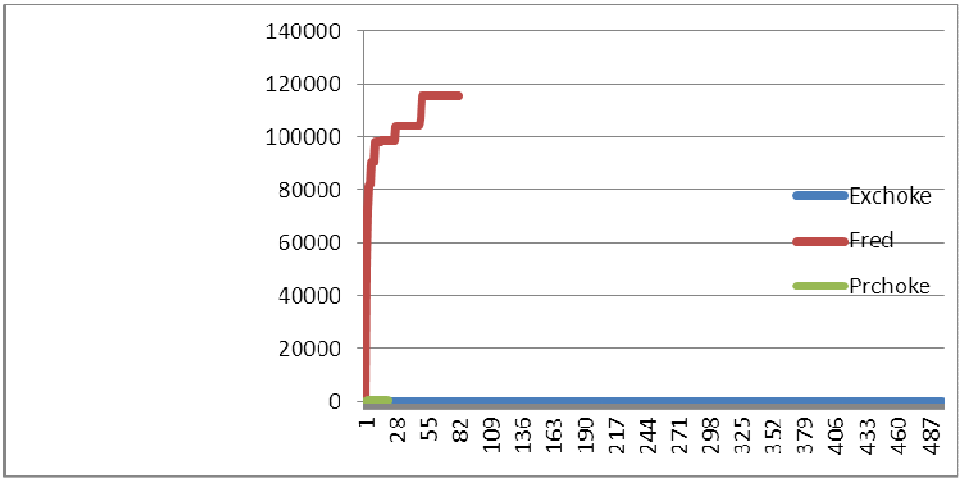

Fig.7. Queue delay for FRED,A-CHOKe and P-CHOKe

The following table (Table.1.) provided the performance analysis of FRED, A-CHOKE and P-CHOKe algorithms based on the parameters or metrics (throughput, packet delivery ratio, queue delay and time consuming).

Algorithm	Throughput	Packet delivery ratio	Queue delay	Time
FRED	Medium	Low	High	High
A-CHOKe	Medium	Medium	Low	Medium
P-CHOKe	High	High	Low	Less

Table.1. Performance analysis of FRED, A-CHOKe, and P-CHOKe

This paper compared three AQM algorithms (FRED, A-CHOKe, P-CHOKe). From this comparison P-CHOKe provides better result than the FRED and P-CHOKe. From the Simulation it is found that P-CHOKE obtains high Packet delivery Ratio, low queuing delay, high throughput, and less process time than the existing AdaptiveChoke and FRED schemes.

REFERENCES

[1] K.Chitra and Dr.G.Padmavathi, " Adaptive CHOKe: An Algorithm to Increase the Fairness in Internet Routers", International Journal on Advanced Networking and Applications , Vol:01,Issue:06,2010.

[2] Alireza Gharegozi, " Greedi Flow Control by FRED Active Queue Management Mechanism", IPCSIT, Vol:7, 2011.

[3] G.F.Ali Ahammed and Reshma Banu, " Analyzing the Performance of Active Queue Management Algorithms",. IJCNC, Vol:2, No:1, 2010.
[4] Dr.G.Padmavathi and K.Chitra, " Classification and Performance of AQM – Based Schemes for Congestion Avoidance", IJCSIS, Vol:8, No:1, 2010.
[5] Aos Anas Mulahuwaish, Kamalrulnizam Abu Bakar, Kayhan Zar Ghafoor, " A Congestion Avoidance Approach in Jumbo Frame – Enabled IP Network" , IJACSA, Vol:3, No:1, 2012.
[6] G.Thiruchelvi and J.Raja, " A Survey On Active Queue Management Mechanisms", IJSCNS, Vol.8, No.12, December 2008.
[7] Shakeel Ahmad, Adli Mustafa, Bashir Ahmad, Arjamand Bano and Al-Sammarraie Hosam, " CMPARATIVE STUDY OF CONGESTION CONTROL TECHINIQUES IN HIGH SPEED NETWORKS", IJCSIS, Vol.6, No.2, 2009.
[8] Alshimaa H.Ismail, Zeiad Elsagheer, and I.Z.Morsi," Survey on Random Early Detection Mechanism and Its Variants", IOSRJCE, vol.2, Issue.6, Aug-2012.
[9] S.Dijkstra, " Modeling Active Queue Management Algorithms Using Stohastic PetriNets", Master Thesis, 2004.
[10] Abhinav Kamra,Huzur Saran, Sandeep Sen, and Rajeev Shorey, " Fair Adaptive Bandwidth Allocation: a rate based active queue management discipline", Elsevier, Computer Networks 44, 2004.
[11] M. Claypool., Robert Kinicki., Mathew Hartling.,"Active Queue Management for Web Traffic", IEEE International Conference on Performance, Computing and Communication 2004 .
[12] B. Braen., Clark,D.,et.al "Recommendations on queue management and congestion avoidance in the Internet", IETF RFC (Information)2309.April 1998 .
[13] Pawel Mrozowski and Andrzej Chydzin Ski, " Stability of AQM Algorithms on Low Congestion Scenarios", International Journal of Applied Mathematics and Informatics, Vol:3, Issue:1, 2009.
[14] Aos Anas Mulahuwaish, Kamalrulnizam Abu Bakar, Kayhan Zar Ghafoor, " A Congestion Avoidance Approach in Jumbo Frame – Enabled IP Network" , IJACSA, Vol:3, No:1, 2012.

