FAVQCHOKE: TO ALLOCATE FAIR BUFFER TO A DYNAMICALLY VARYING TRAFFIC IN AN IP NETWORK

K.Chitra[1] and Dr.G.Padmavathi[2]

[1]Department of Computer Science, D.J.Academy for Managerial Excellence, Coimbatore, India
chitrakandaswamy@yahoo.com
[2]Department of Computer Science, Avinashilingam University for Women, Coimbatore, India

ABSTRACT

In IP networks, AQM attempts to provide high network utilization with low loss and low delay by regulating queues at bottleneck links. Many AQM algorithms have been proposed, most suffer from instability of queue, bursty packet drop, require careful configuration of control parameters, or slow response to dynamic traffic changes and unfairness. The deployment of active queue management techniques such as RED based is used that results in increased bursty packet loss and unfairness caused by an exponential increase in network traffic. The inherent problem with these queue management algorithms is that they all use queue lengths as the indicator of the severity of congestion. In order to solve this problem, a new active queue management algorithm called FAVQCHOKe is proposed. In this paper, arrival rate at the network link is maintained as a principal measure of congestion to improve the transient performances of the system and ensures the entire utilization of link capacity. In addition this proposed algorithm uses queue length and flow information that enhances fairness. This characteristic is particularly beneficial to real-time multimedia applications. Further, FAVQCHOKe achieves the above while maintaining high link utilization and low packet loss. This paper discusses about the inherent weaknesses of current techniques and how the proposed algorithm overcomes the weaknesses and ensures high degree of effectiveness in the performance of the system.

KEYWORDS

Congestion, Dynamic Traffic, IP Network, Fairness, Load

1. INTRODUCTION

Internet congestion occurs when the aggregate demand for a resource (e.g., link bandwidth) exceeds the available capacity of the resource. Thus resulting effects of such congestion include long delays in data delivery, wasted resources due to lost or dropped packets, and even possible congestion collapse, in which all communication in the entire network comes to an end. It is therefore obvious that in order to maintain good network performance, certain mechanisms must be provided to prevent the network from being congested for any noteworthy period of time.

Two approaches to handling congestion are congestion control and congestion avoidance. The former is reactive in which congestion control typically comes into play after the network is overloaded, that is, congestion is detected. The latter is proactive in that congestion avoidance comes into play before the network becomes overloaded, that is, when congestion is expected. In general and throughout this article, the term congestion control is used to denote both approaches**.**

DOI : 10.5121/ijdps.2011.2106

Congestion control involves the design of algorithms to statistically limit the demand-capacity mismatch, or dynamically control traffic sources when such a mismatch occurs. Currently, usage of the Internet is dominated by transmission control protocol (TCP) traffic such as remote terminal, FTP, Web traffic, and electronic mail. Although these applications can tolerate either packet delay or packet losses rather gracefully, congestion remains a major problem that leads to poor performance. Internet is still evolving as a high-performance network providing ubiquitous services, including real time voice/video. Accordingly, many congestion control approaches have been proposed. However, current Internet congestion control methods results in unsatisfactory performance including multiple packet losses and low link utilization as the number of users and the size of the network increases. Active Queue Management (AQM), network algorithms executed by network components such as routers, detect network congestion, packet losses, or incipient congestion, and inform traffic sources either implicitly or explicitly.

The first AQM algorithm RED detects congestion by observing the queue state. In RED [1] packet drop probability is linearly proportional to queue length. The AQM algorithm RED drops packets before a queue becomes full. This reduces the number of packets dropped. RED and its variant uses queue length as a congestion indicator that results in certain drawbacks. In order to overcome the difficulty of relying only on queue length to identify the level of congestion various other AQMs are introduced with different congestion indicators.

To overcome the problems with RED, REM [2] was proposed. This AQM scheme attempts to make user input rates equal to the network capacity. In case of high congestion, sources are indicated to reduce their rates. In contrast to RED, REM decouples congestion measure from performance measure which stabilizes the queue around its target independent of traffic load leading to high utilisation and low delay. AQM schemes like GREEN [3], AVQ [4] also depend on arrival rate to control the congestion in the router. AVQ uses only the traffic input rate for the measure of congestion. This provides early feedback of congestion.

Another AQM scheme BLUE [5] does not use queue length as a congestion metrics. BLUE uses packet loss and link utilization as a congestion indicator. In LRED [6] packet loss ratio is used to design more adaptive and robust AQM. It uses the instantaneous queue length and packet loss ratio to calculate the packet drop probability.

AQMs that used only the congestion metric faced the problem of unfairness in handling the different types of traffic. To overcome this problem, FRED [7] was proposed that improved uniformity by constraining all flows to occupy loosely equal shares of the queue's capacity. To overcome the shortcomings of the FRED, CHOKe [8] was designed. This paper considers the advantages of the queue-based, load-based and flow-based algorithms to combine into an improvised AQM algorithm. This improvised AQM algorithm called Flow based AVQCHOKe (FAVQCHOKe) tries to achieve its objective of improving overall performance

In the next section, a broad study of all possible AQM schemes is presented to identify the basic schemes that exist and their classification based on congestion metric and flow information. In section 3 the FAVQCHOKe algorithm is discussed to bring out the advantages of the proposed algorithm.

2. BACKGROUND

During the recent years research activities have come out with various congestion avoidance mechanisms in Internet to improve Internet traffic. But each of these mechanisms is ineffective especially in heavy traffic network. That has made research a continuous process in identifying the best Active Queue Management algorithm. Congestion in routers results in high packet loss leading to high cost that is reduced by the various existing AQM schemes.

The existing schemes use various factors or metrics to detect congestion. These factors are used to estimate congestion in the queue based on which various AQM algorithms are proposed in the past few years. As discussed in [9], the schemes are based on congestion metrics like Queue-length, Load, both Queue and Load, others like Loss rate. Further some of these schemes also use flow information along with various congestion metrics like Queue-length, Load, both Queue and Load, others like Loss rate metrics to analyze and control the congestion in routers more accurately. Considering these factors AQM schemes are categorized based on congestion metrics without flow information and with flow information as shown in Fig 1.

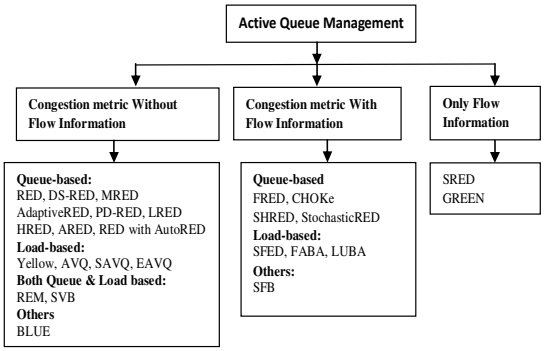

Figure 1 Classification of AQM Schemes

The first well known AQM scheme proposed is RED. It tries to avoid problems like global synchronization, lock-out, bursty drops and queuing delay that exists in the traditional passive queue management i.e Droptail scheme. In this AQM scheme, the dropping probability depends on various parameters like min_{th}, max_{th}, Q_{ave} and w_q. These parameters must be tuned well for the RED to perform better. However, it faces weaknesses such as accurate parameter configuration and tuning. This becomes a major disadvantage for the RED algorithm. Though RED avoids global synchronization but fails when load changes dramatically. Queue length gives minimum information regarding the severity of congestion. RED does not consider the packet arrivals from the various sources, which is also a very important measure for the congestion indication. RED based AQMs like DSRED [10], MRED [11], AdaptiveRED [12] tried to remove the problems of RED. DSRED, MRED showed better performance than RED. AdaptiveRED tried to eliminate the problem of parameter tuning by adapting the parameters. Though RED and its variant were simple to handle, the difficulty with it is the parameter tuning problem. RED based AQMs are vulnerable to unresponsive flows dominating a routers queue.

To overcome this problem, FRED was proposed that improved uniformity by constraining all flows to occupy loosely equal shares of the queue's capacity. AQMs that used only congestion metric and not flow information faced the problem of unfairness in handling the different types of traffic. FRED is based on instantaneous queue occupancy of a given flow. It provides better protection than RED for adaptive flows and isolating non-adaptive greedy flows. Combination of Flow and congestion metric based AQMs like CHOKe, SFB [13], SFED [14], FABA [15], StoRED [16] were proposed to allocate fair buffer between flows considering the effects of misbehaving or non-responsive flows.

To remove the implementation cost of FRED, CHOKe (CHOose and Keep for responsive flows, and CHOose and Kill for unresponsive flows) algorithm was designed that penalizes misbehaving flows by dropping more of their packets. So CHOKe tries to bring fairness for the flows that pass through a congested router. CHOKe provides much better fairness than FRED

but penalizes high bandwidth flows and does not handle unresponsive flows in case of few packets. Flow based AQMs with congestion metric are able to discriminate responsive and non-responsive flows. The malicious flows are identified which might cause congestion at the router.

AVQ [17] and YELLOW [18] used only input rate as the congestion indicator to demonstrate that it performed well in terms of link utilisation and packet loss. In AVQ, the virtual queue is updated, when a packet arrives at the real queue to indicate the new arrival of the packet. When the virtual queue or buffer overflows, the packets are marked / dropped. The virtual capacity of the link is modified such that total flow entering each link achieves a desired utilization of the link. This is done by aggressive marking when the link utilization exceeds the desired utilization and less aggressive when the link utilization is below the desired utilization.

REM used both input rate and queue length that illustrated very high utilization but very low throughput compared to Queue based RED. This scheme stabilizes both the input rate around link capacity and the queue around a small target independent of the number of users sharing the link. BLUE used packet loss and link utilization as congestion indicator to give a very high throughput and, high utilisation with low queue length stability.

3. PROPOSED ALGORITHM

The proposed algorithm is motivated by the need for a stable queue size and fair bandwidth allocation irrespective of the varying traffic and congestion characteristics of the n flows. As discussed in Introduction, some of the algorithms arrive at a stable queue size and some of them bring in fairness when the shared link has n flows. The unstable queue size results in high queue oscillation due to the parameter tuning problem in queue based AQMs. We are motivated to identify a scheme that penalizes the unresponsive flows with the stable queue size.

The proposed algorithm - Flow based AVQCHOKe (FAVQCHOKe) involves both the congestion factors - queue length and load factor including the flow information. As mentioned in the previous section, queue length gives minimum information regarding the severity of congestion. This proposed algorithm does consider the packet arrivals from the various sources, which is also a very important measure for the congestion indication. Another problem that exists with most AQMs is vulnerability to unresponsive flows resulting in dominating a routers queue. FAVQCHOKe algorithm tries to remove this problem that exists in the other AQMs.

FAVQCHOKe in Fig. 2 uses a virtual queue and feeds the virtual queue sizes to the CHOKe algorithm. In this proposed algorithm, the CHOKe algorithm reacts to the congestion level of the virtual queue rather than the actual queue. The virtual queue is adjusted using the desired link utilisation rather than the parameter using w_q and max_p. This proposed algorithm adjusts the incoming traffic according to the desired link utilization. As discussed in CHOKe, average queue size and drop probability is calculated using w_q and max_p respectively. This is removed in this proposed algorithm. It also indicates two thresholds on the virtual queue, a minimum threshold min_{th} and a maximum threshold max_{th}. The virtual capacity of the virtual queue is smoothened and calculated using the parameter *alpha*. a*lpha* in Fig. 3 is a low-pass filter for the calculation of actual capacity. The recommended value is 0.5 as discussed in [19]. The range of processing is defined by the parameters: max_capacity and min_capacity. The value for the max_capacity is chosen to be high to increase the capacity. The min_capacity can be chosen too low as it does not affect the range of the actual capacity. Then the virtual capacity is updated based on the max_capacity and min_capacity. The virtual queue size VQ is calculated based on the serviced bytes. Then the queue size is calculated using the virtual queue size. The queue size is compared with the two thresholds for every arriving packet.

```
Initialisation:
count = -1
last_measure = curr_time
prev_tx_bytes = bytes transmitted
For every packet arrival {
        /* Calculate virtual queue size */
delta_time = curr_time – last measure
If delta_time > 1
        tx_bytes = bytes transmitted
        output_rate = (tx_bytes – prev_tx_bytes) * 8000 / delta_time
        prev_tx_bytes = tx_bytes
                /* Smoothen virtual capacity */
        v_capacity = alpha * output_rate * (1.0 – alpha) * v_capacity
                /* Update virtual capacity */
        v_capacity = MAX(MIN(max_capacity, v_capacity), min_capacity)
        serviced_bytes = v_capacity / 1000 / 8 * delta_time
        if VQ > serviced_bytes
                VQ = VQ – serviced_bytes
        Else
                VQ =0
                Q_time = curr_time
Last measure = curr_time
q_size = VQ /1500
if (q_size < min_th)
        Forward the new packet
Else
        Select randomly a packet from the queue for their flow id
        Compare arriving packet with a randomly selected packet.
        If they have the same flow id
                                Drop both the packets
        Else
                if (q_size ≥ max_th)
                    Calculate the dropping probability p_a
                    Drop the packet with probability p_a
                Else
                    Drop the new packet
}
```

Figure 2 Pseudocode of FAVQCHOKe

Variables:	
p_a	: current packet-marking probability
p_b	: temporary marking probability
q_size	: current virtual queue size
VQ	: Virtual Queue size
Fixed parameters:	
min_{th}	:minimum threshold for queue
max_{th}	:maximum threshold for queue
min_capacity	
max_capacity	: Range of Processing capacity
alpha	: loss-pass filter

Figure 3 Parameters in FAVQCHOKe

If queue size is less than min_{th}, every arriving packet is queued. If queue size is greater than max_{th}, every arriving packet is dropped. This results in queue size below max_{th}. When the queue size is greater than min_{th}, every arriving packet is compared with a randomly selected packet from the virtual queue for their flow id. If they have the same flow id, both are dropped. Otherwise the randomly selected packet is placed in the same position in the buffer. The arriving packet is dropped with a probability depending on the virtual queue size.

To achieve good throughput and reasonable average queue length with RED based algorithm requires careful tuning of both w_q and max_p. Adapting max_p 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. In case if too small a value of w_q, performance is in terms of queuing delay and too large a value leads to decreased throughput. These problems do not exist with this proposed algorithm as these two parameters are not used. This algorithm tries to achieve good throughput, reasonable average queue length resulting in reduced packet drop and minimised queuing delay. In addition, FAVQCHOKE is also able to isolate non-adaptive greedy traffic more effectively.

FAVQCHOKe queue size ODE is modeled with M/D/1 assumption that brings out the main characteristic of the FAVQCHOKe algorithm which is deterministic service time. While actual queue size ODE is modeled with M/M/1 assumption for RED. The following ODEs are used to model RED, AVQRED and FAVQCHOKe and the details are available in [19], [20], [21] and [22].

$$\frac{dx}{dt} = \frac{\ln(1-\propto)}{\delta} x(t) - \frac{\ln(1-\propto)}{\delta} q(t)$$

$$\frac{dq}{dt} = \lambda(t)\big(1-p(t)\big) - u\,\frac{q(t)}{1+q(t)}$$

$$\frac{d\lambda}{dt} = \big(1-p(t)\big)\,\frac{m}{R^2}\,p(t)\,\frac{\lambda}{2m}$$

$$\frac{ds}{dt} = -\tau \cdot u + \lambda \cdot (1-p)$$

$$\frac{dl}{dt} = u.\left(\frac{q(t)}{1+q(t)}\right) - l\,(t-1)$$

Variables

$\frac{dx}{dt} = Weighted\ average\ queue\ size$

$\frac{dq}{dt} - Actual\ Q\ size$

$\frac{d\lambda}{dt} = Arrival\ rate$

$\frac{ds}{dt} = FAVQCHOKe\ and\ AVQRED\ queue\ size\ from\ the\ Lindley\ equation$

$\frac{dl}{dt} = Link\ Utilisation$

Parameters

$\propto = Queue\ weight\ for\ weighted\ average$

$\delta = Delta\ time\ between\ each\ packet\ arrival$

$\lambda = Aggregate\ arrival\ rate$

$p = AQM\ packet\ loss\ probability$

$u = Service\ rate\ with\ uniform\ distribution\ between\ [80\%, 100\%]$

$m = Number\ of\ concurrent\ TCP\ connections$

$R = Round\ trip\ delay$

$\tau = Target\ link\ utilisation\ \%$

To compute the marking probability, either q in case of RED or s when AVQRED or FAVQCHOKe is given to the following equation:

$$p(x) = \begin{cases} 0, & 0 \le x \le \min_{th} \\ \dfrac{x - \min_{th}}{\max_{th} - \min_{th}} \max_{p,} & \min_{th} \le x \le \max_{th} \\ 1, & \max_{th} \le x \end{cases}$$

FAVQCHOKe considers input rate that provides early feedback of congestion resulting in good link utilisation. In case of rate-based AQM, the queue length is less sensitive to the number of connections. So queue length stability is provided by this algorithm. Routers with this AQM will provide better protection than other AQMs for adaptive (fragile and robust) flows.

EXPERIMENTATION

In this section, we will use the packet-simulator ns-2 to simulate the FAVQQCHOKe algorithm. In this simulation the network topology in Fig. 4 is with a single link of capacity 1Mbps that drops packet according to the AQM algorithm. The congestion link is in between the two routers R1 and R2. The link is shared by n TCP flows and n UDP flows. End hosts are connected to the routers using a 10Mbps link. All links have a small propagation delay of 1ms so that the delay introduced is by the buffer delay rather than the transmission delay. The TCP flows are derived from FTP sessions which transmit large size files. The UDP hosts send packets at a constant bit rate of 2 Mbps. In the simulation setup we consider 32 TCP flows and 1 UDP flow in the network. The minimum threshold min_{th} in the FAVQQCHOKe scheme is set to 100 and the maximum threshold max_{th} to be twice the min_{th} and the physical queue size is fixed at 300 packets.

RED and other AQMs are unable to penalize unresponsive flows. As the packets dropped from each flow over a period of time is almost the same. Consequently the misbehaving traffic like UDP can take up a large % of the link bandwidth and starve out TCP friendly flows as in Fig 5. FAVQCHOKe identifies and penalizes misbehaving flows effectively compared to the existing AQMs as in Table 1 with the help of input rate and the flow information.

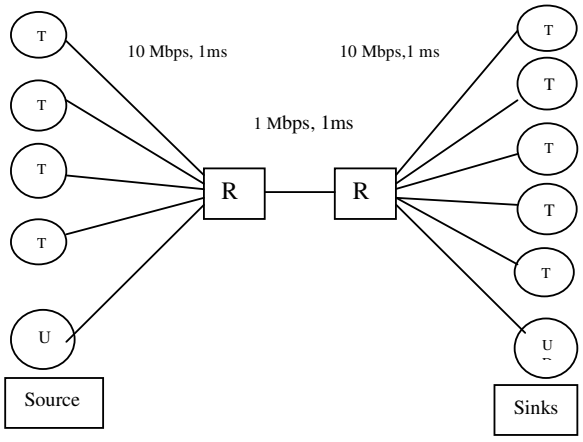

Fig.4 Network Topology

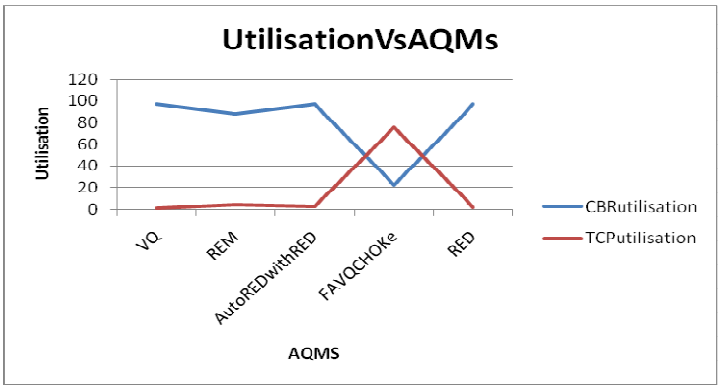

Fig 5 CBR and TCP Utilisation of other AQMs with FAVQCHOKe

Table 1 CBR and TCP Utilisation of other AQMs with FAVQCHOKe

	In %	
	CBRutilisation	**TCPutilisation**
VQ	96.9	1.2
REM	88.82	3.95
AutoREDwithRED	97.29	2.7
FAVQCHOKe	22	76
RED	97.16	2.35

CONCLUSIONS

A new rate-flow based AQM method, FAVQCHOKe is designed to improve the performance of congested routers in IP networks. FAVQCHOKe requires the queue, load and flow information to adapt the queue size with regard to the dynamics of traffic in routers. This algorithm enhances the way virtual capacity is adapted to the varying traffic types in IP networks. This AQM also protects adaptive flows from non-adaptive flows resulting in good service under varying traffic. FAVQCHOKe algorithm considers the advantages of queue based, load based AQMs and flow based algorithm to provide the QOS requirements of the IP networks. So the algorithm improves the performance of AQM under heavy load and guarding the adaptive flows from nonadaptive flows to achieve best QOS to all users by simulation.

REFERENCES

[1] S. Floyd and V. Jacobson, "*Random early detection gateways for congestion avoidance",* IEEE/ACM Trans. Networking, vol. 1, pp. 397–413, Aug. 1993.

[2] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, "*REM*: *Active queue management*," IEEE Network Mag., vol. 15, pp. 48–53, 2001

[3] Wu-chun Feng, Apu Kapadia , Sunil Thulasidasan,, "*GREEN: Proactive Queue Management over a Best-Effort Network*", IEEE GlobeCom, Taipei, Taiwan, November 2002

[4] S. Kunniyur, R.Srikant, "*Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management*", Proceedings of ACM SIGCOMM, San Diego, 2001

[5] W. Feng, D.D. Kandlur, D. Saha, D. Saha, "*The Blue active queue management algorithms",* IEEE/ACM Transactions on Networking 2002

[6] C.Wang, J. Liu, B. Li, K. Sohraby, Y. H. Lou, "*LRED: A Robust and Responsive AQM Algorithm Using Packet Loss Ratio Measurement* ", January 2007, vol. 18 no. 1

[7] D.Lin, R.Morris, "Dynamics of Random Early Detection". Proceedings of ACM SIGCOMM October 1997.

[8] Pan R., Prabhakar.B, and Psounix.k, "*CHOKe*, *a Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation*", IEEE INFOCOMM, Feb 2000

[9] K.Chitra, G. Padmavathi, "*Classification and performance of AQM-based Schemes*", International Journal of Computer Science and Information Security, Vol 8, No. 1, 2010

[10] Bing Zheng, Mogammed Atiquzzaman, "*DSRED: An Active Queue Management Scheme for Next Generation Networks*" Proceedings of 25th IEEE conference on Local Computer Networks LCN 2000,November 2000

[11] Jahoon Koo., Byunghun Song., Kwangsue Chung., Hyukjoon Lee., Hyunkook Kahng.,"*MRED: A New Approach To Random Early Detection*" 15th International Conference on Information Networking, February 2001

[12] S.Floyd., R.Gummadi,S.Shenkar and ICSI, "*Adaptive RED*: *An algorithm for Increasing the robustness of RED's active Queue Management*", Berkely,CA [online] http:www.icir.org/floyd/red.html

[13] Wu-chang Feng, Dilip D. Kandlur, Debanjan Saha, Kang G. Shin, **"***Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness*", IEEE INFOCOM 2001

[14] A. Kamra, S. Kapila, V. Khurana, V. Yadav, H.Saran, S.Juneja, R.Shorey, "*SFED: a rate control based based active queue management discipline*", IBM India Research Laboratory Research report # 00A018, November 2000.

[15] A. Kamra., Huzur Saran., Sandeep Sen., Rajeev Shorey, "*Fair Adaptive Bandwidth allocation: a rate control based active queue management discipline*", Computer Networks, July 2003

[16] S. Chen, Zhen Zhou,, Brahim Bensaou., "*Stochastic RED and its applications*" ICC 2007

[17] S. Kunniyur, R.Srikant, "*Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management*", Proceedings of ACM SIGCOMM, San Diego, 2001

[18] Chengnian Long., Bin Zhao., Xinping Guan., Jun Yang., "*The Yellow active queue management algorithm*", Computer Networks, November 2004

[19] D. J. Byan, J.S. Baras, "*A new rate based active queue management: Adaptive Virtual Queue RED*", Fifth Annual Conference on Communication Networks and Services Research, 2007

[20] P. Kuusela, P. Lassila, J. Virtamo, P.Key, "*Modeling RED with Idealised TCP sources*", 9th IFIP cinference on Performance Modelling ad evaluation of ATM & IP networks 2001

[21] P. E. Lassila, J. T.Virtamo, "*Modeling the dynamics of the RED algorithm*", in Proceedings of QofIS'00, pp 28-42, September 2000

[22] V.Misra, V.Gong, and D. Towsley, "*A fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED*", in Proceedings of ACM SIGCOMM, August 2000.

Authors

K.Chitra received her B.Sc (C.Sc) from Women Christian College, Chennai and M.Sc from Avinashilingam University for Women, Coimbatore in 1991 and 1993 respectively. And, she received her M.Phil degree in Computer Science from Bharathiar University, Coimbatore in 2005. She is pursuing her PhD at Avinashilingam University for Women. She is currently working as a Lecturer in the Department of Computer Science, D.J.Academy for Managerial Excellence, Coimbatore. She has 12 years of teaching experience. Her research interests are Congestion Control in Networks and Network Security.

Dr. Padmavathi Ganapathi is the Professor and Head of the Department of Computer Science, Avinashilingam University for Women, Coimbatore. She has 21 years of teaching experience and one year Industrial experience. Her areas of interest include Network security and Cryptography and real time communication. She has more than 80 publications at national and International level. She is a life member of many professional organizations like CSI, ISTE, AACE, WSEAS, ISCA, and UWA. She is currently the Principal Investigator of 5 major projects under UGC and DRDO