NEW ACTIVE QUEUE MANAGEMENT MECHANISM FOR REDUCING PACKET LOSS RATE

Mrs S. Malarvizhi
Professor , Department of M.C.A,
K.S.Rangasamy College of Technology,
Tiruchengode -637215
E-Mail: smalarvizzhi@yahoo.com

Dr. Madheswaran .M
The Principal,
Muthayammal Engineering College,
Rasipuram

ABSTRACT

Due to exponential increases in internet traffic, Active Queue Management (AQM) has been heavily studied by numerous researchers. Virtual queue-based marking schemes have been recently proposed for Active Queue Management (AQM) in Internet routers. In an effort to improve performance of congested gateways, a new Active Queue Management (AQMNEW) algorithm was developed by feeding virtual queue size to the RED algorithm. The objective of the new algorithm is to improve overall performance by keeping link utilization high, link utilization stable, queuing delay low and consecutive packet drop rate low. This paper shows the objective is met by comparing AQMNEW with six other well known AQM methods. To provide fair comparisons, the AQM parameters are fine-tuned by exploring many different parameter settings. The simulation results conclude that AQMNEW improves overall performance by 8 to 25%.

Keywords **-** Congestion, Gateway, Network, Traffic generator, Virtual queue.

1. INTRODUCTION

In the modern day Internet, there has been a strong demand for quality of service (QoS) and fairness among flows. As a result, in addition to the sources, the links are also forced to play an active role in congestion control and avoidance. Random Early Discard (RED) [8] was originally proposed to achieve fairness among sources with different burst-ness and to control queue lengths. RED allows for dropping packets before buffer overflow. Another form of congestion notification that has been discussed since the advent of RED is Explicit Congestion Notification (ECN) [7]. ECN has been proposed to allow each link to participate in congestion control by notifying users when it detects an onset of congestion. Upon detecting incipient congestion, a bit in the packet header is set to one for the purpose of notifying the user that a link on its route is experiencing congestion. The user then reacts to the *mark* as if a packet has been lost. Thus, the link avoids dropping the packet (thereby enhancing good-put) and still manages to convey congestion information to the user.

Internet Protocol (IP) over Satellite (IPoS) has been commercially available for the last few decades. Due to its high availability and mobility, IPoS has been attractive to areas where terrestrial services aren't available as well as enterprises with scattered branch offices. One big barrier that IPoS has faced is its high propagation delay between earth stations and satellite. A typical round trip time (RTT) for a two-way geosynchronous satellite is around 600 msec. One aspect of the problem is the TCP slow start [21] phase where it takes a long time (RTTxlog2xSSTHRESH) to reach the maximum congestion window threshold (maximum rate at which the sender sends traffic); and the other aspect is that the maximum throughput of 65,535x8/RTT is too low when the TCP Window Scale option is not supported. Even when the TCP Window Scale option is supported, unless all nodes support the option, fair bandwidth sharing becomes an issue. TCP Spoofing or Performance Enhancing Proxy (PEP) [22] has been practiced by most of IPoS providers to overcome this problem with TCP. For consistency, the term PEP will be used throughout this paper. The basic idea of PEP is to buffer at least one round trip worth of data by locally acknowledging the data. Usually buffering only one round trip worth of data isn't enough because one has to account for queueing delays associated with congestion and bandwidth allocations.

To provide ECN marks or drop packets in order to control queue lengths or provide fairness, the routers have to select packets to be marked in a manner that conveys information about the current state of the network to the users. Algorithms that the routers employ to convey such information are called *Active Queue Management (AQM)* schemes. An AQM scheme might mark or drop packets depending on the policy at the router. In this paper, we use the term "marking" more generally to refer to any action taken by the router to notify the user of incipient congestion. The action can, in reality, be ECN-type marking or dropping (as in RED) depending upon the policy set for the router.

Active Queue Management (AQM) is an algorithm that detects and reacts to congestion to avoid queue overflows. There are generally two ways to react to congestion: signal congestion to traffic sources explicitly by setting Explicit Congestion Notification (ECN) [7] bits; or signal congestion to traffic sources implicitly by dropping

packets. ECN is not used in our study due to the following reasons:

1. The problems that we are trying to solve are not due to packet drops between gateway and senders.

2. ECN marking after PEP may seem to avoid retransmissions over satellite and fix the queueing instability problem but it is too late to enforce ECN bits when data are already acknowledged without ECN bits by PEP.

When applying AQM to satellite networks, the following need to be considered:

1. The source of congestion is different in satellite networks. i.e. In satellite networks, congestion arises mainly due to the satellite link capacity, not due to the processing capacity. Therefore, gateways in satellite networks become congested when the offered load is greater than the allowed transmit rate
whereas gateways in terrestrial networks often become congested when the offered load is greater than the processing capacity.

2. Monitoring and marking packets after PEP is not a good idea because it involves retransmissions over satellite link.

3. Monitoring (with real-queue-based AQM) and marking packets before PEP is not a good idea because the receive queue will never be congested when the congestion bottleneck is the spacelink capacity, not the processing capacity. This isn't true for virtual-queue-based AQMs such as Adaptive Virtual Queue (AVQ) [9], but they have the global synchronization (consecutive packet drops) problem.

4. Monitoring after PEP and marking packets before PEP is not a good idea because the high buffering between congested queue (transmit queue towards satellite) and receive queue may result in unstable queueing behavior.

Due to exponential increases in internet traffic, congestion has been an unavoidable attribute to many Internet service providers for the last decade and improving performance of congested gateways has been an active research interest. This paper contributes an effort to improve overall performance of congested gateways by proposing a new rate-based AQM algorithm, Active Virtual Queue Random Early Detection (AQMNEW). Active Queue Management (AQM) is an algorithm that detects and reacts to congestion to avoid queue overflows. There are generally two ways to react to congestion: signal congestion to traffic sources explicitly by setting Explicit Congestion Notification bits, or signal congestion to traffic sources implicitly by dropping packets. A new AQM algorithm was developed by overlaying a virtual queue on top of the real queue and applying the RED algorithm to the virtual queue instead of the real queue. The objective of the new algorithm is to improve overall performance by keeping link utilization high, link utilization stable, queuing delay low and consecutive packet drop (mark) rate low. This paper compares AQMNEW with six well-known AQM methods, RED [1], ARED [2], BLUE [3], GKVQ [6], AVQ [5], and PI [4]. To provide fair comparisons, the parameters for all seven AQM methods are fine-tuned before the comparisons. The results conclude that AQMNEW improves overall performance by 8 to 25%.

This paper is organized as follows: Section 2 provides an overview of the six different Active Queue Management methods. Section 3 discusses the proposed work of new Active Queue Management algorithm, AQMNEW. Section 4 discusses the mathematical model of new Active Queue Management algorithm, AQMNEW. Section 5 provides the simulation framework and methodologies. Section 6 provides the results. Finally, section 7 concludes the paper.

2. RELATED WORK

The RED [1] algorithm computes the marking probability when the weighted queue size falls between *minth* and *maxth* parameters. The marking probability becomes higher as the weighted queue size gets closer to *maxth* (becomes 1 if it's greater than *maxth*), and it also becomes higher as the distance between each marking gets bigger. Parameter tuning is required for *wq* and *maxp*. *wq* controls the weighted average queue size which then determines how quickly the algorithm reacts to congestion. Reacting too quickly may result in oscillations in queue size, and reacting too slowly may result in queue overflows. *maxp* is a scaling factor for the marking probability which also controls how quickly the algorithm reacts to congestion.

The intuition behind the ARED [2] algorithm is to self adjust *maxp* according to the target average queue size every half second. The results revealed that the adjustment of *maxp* does not find the value that best represents the desired packet loss rate of the gateway. i.e. *maxp* reduction is so aggressive that it always stays at a lower value than the desired packet loss rate resulting in the average queue size to exceed *maxth*. Therefore, the ARED algorithm was slightly modified to have a hysteresis requirement where decreasing *maxp* by β requires the queue size to stay under the target queue size for *n* times.

Unlike RED and ARED, BLUE [3] relies on packet loss and idle events of the queue when adjusting its marking probability. It increases or decreases by a configured value, *d1* or *d2* respectively, when the queue size exceeds a threshold or it becomes empty. It does so no more than once every configured interval, *freeze_time*.

Gibbens-Kelly Virtual Queue (GKVQ) [6] maintains a virtual queue whose service rate is the

desired link utilization. When an incoming packet exceeds the virtual queue limit, it drops or marks the packet. Adaptive Virtual Queue (AVQ) [5] maintains the same virtual queue whose capacity is dynamically adjusted. The virtual capacity is adjusted by adding the number of bytes that could have been serviced between the last and the current packets minus the bytes that were just received. The problems with these two schemes are global synchronization and queue overflows when virtual capacity exceeds the actual processing capacity. Although AVQ seems to solve the latter by adjusting the virtual capacity, there is an assumption that the actual processing capacity is static which could lead to high queuing delays and queue overflows if the processing allocation shifts – a common phenomenon in real networks. However, as long as the virtual capacity does not exceed the actual processing capacity, GKVQ and AVQ yield low queuing delays and stable link utilizations.

The main idea behind the Proportional Integrator (PI) [4] algorithm is that the queue length slope determines the packet marking probability, and the queue is regulated by the desired queue length.

3. PROPOSED WORK

The problem with AVQ is global synchronization where consecutive packet drops occur due to its tail-drop nature of packet marking. When packets are dropped consecutively, multiple TCP connections will react to the drops simultaneously resulting in global oscillatory link utilization amongst multiple TCP connections.

This problem is severer with RED due to its oscillatory queueing behavior . When the transmit queue congestion level and the senders congestion windows are not synchronized, the RED region will likely be exceeded resulting in tail-drop behavior. The motivations behind AQMNEW are to fix the problems of AVQ: global synchronization (consecutive packet drops) and inaccurate virtual capacity which leads to high queuing delays.

NEWAVQ solves the asynchronous queueing behavior problem by both monitoring and marking at the receive queue. Monitoring and marking at the receive queue is possible because NEWAVQ constructs a virtual queue which can be placed anywhere.

The AQMNEW algorithm constructs a virtual queue and feeds the virtual queue sizes to the RED algorithm instead of feeding the weighted average queue sizes to it. AQMNEW reshapes the incoming traffic according to the desired link utilization because the RED algorithm reacts to the congestion level of the virtual queue which is serviced by the desired link utilization. The *AQMNEW Algorithm* highlights the AQMNEW parameters in bold.

Note that *wq* and *maxp* are no longer in the algorithm because their functionalities are replaced by the desired link utilization in AQMNEW. *alpha* is a low-pass filter for the actual capacity calculation and choosing a small number such as 0.05 is recommended since changes in processing capacity are not drastic. *min_capacity* and *max_capacity* define the range of processing capacity. They define the RED region of the virtual queue which indirectly controls the real queue length. One big advantage of applying RED to the virtual queue is that the algorithm works well even with a small queue (because the RED region for the virtual queue can still be big) whereas having a small queue makes RED resemble a tail-drop queue.

The implementation complexity of the AVQ scheme is comparable to RED. RED performs averaging of the queue length, dropping probability computation and random number generation to make drop decisions. We replace these with the virtual capacity calculation in AVQ.

AVQ is a primarily a rate-based marking, as opposed to queue length or average queue length-based marking.

Initialization
count <- -1
last_measure <- curr_time
prev_tx_bytes <- bytes_tx
For each packet arrival
/* Calculate virtual queue size */
delta_time <- curr_time - last_measure
if delta_time > 1
/* Compute actual output rate in bps */
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)
/* Compute number of bytes that could have been processed and transmitted */
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
/* Feed VQ size to the RED algorithm */
if min_{th} < q_size < max_{th}
count <- count + 1
pb <- (q_size - min_{th}) /(max_{th} - min_{th})

pa <- pa / (1 - count * pb)
With probability pa:
Mark the arriving packet
count <- 0
else if max_{th} <= q_size
Mark the arriving packet
count <- 0
else
count <- -1
/* b = number of bytes in current packet */
VQ <- VQ + b + 1

4. MATHEMATICAL MODEL

The actual queue size ODE was modeled with M/M/1 assumption and the AQMNEW queue size ODE was modeled with M/D/1 assumption to expose the main characteristic of the AQMNEW algorithm which is deterministic service time.

Equations:

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

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

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

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

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

Variables:

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

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

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

$$\frac{d\lambda}{dt} = Actual\ Q\ size$$

$$dl = linkutiliz\ ation$$

Parameters:

α = Queue weight for weighted average

δ = Delta time between each packet arrival

λ = 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

τ = Target link utilization %

Either q (when RED or ARED is used) or s (when AQMNEW is used) is fed to the following equation to compute the marking probability:

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

5. SIMULATION FRAMEWORK

The actual gateway code from *Hughes Network Systems* was used to evaluate the AQM methods. The software was simplified to have only one queue for receiving packets and one thread that de-queues and transmits the packets. The seven AQM methods were implemented at the receive queue and "drop" was chosen for the marking method. AQM was implemented only on one queue which is assumed to be the only congestion queue in our network scenarios. To eliminate the unnecessary system complexity, satellite specific features such as TCP spoofing, per-user queuing and output rate limiting were removed. Although the software serves as a gateway in satellite networks, it was modified to be a generic IP Gateway. The simulation environment was constructed using two IP Gateways (one serves as a router to the internet and the other serves as modems for N different users) and a traffic generator called *Spirent*®.

Fig.2 illustrates the connectivity of the simulation setup. A delay simulator was inserted between the two gateways. The round trip time (RTT) between the client IP Gateway and the

Spirent® is 4 msec, the RTT between the client IP Gateway and the AQM IP Gateway is 20~40 msec, and the RTT between the AQM IP Gateway and the *Spirent*® is 40~80 msec resulting in an end to end RTT of 104~204 msec.

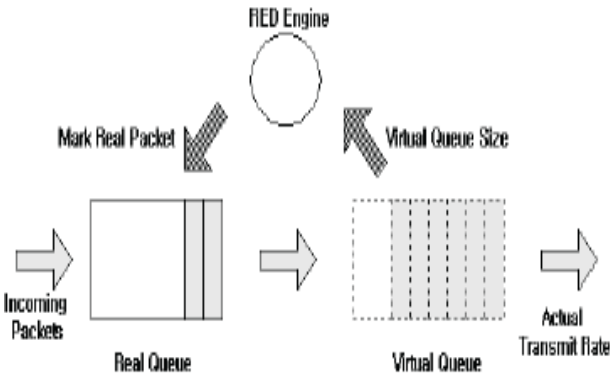

Fig. 1. AQMNEW flow diagram

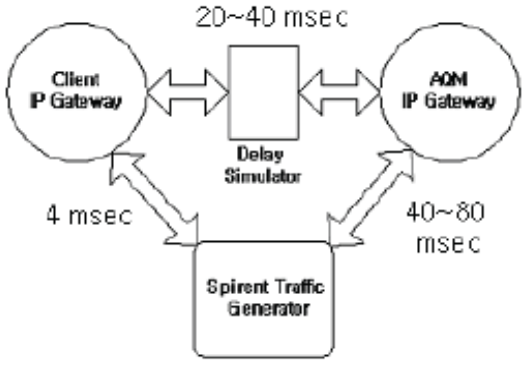

Fig. 2. Simulation Set-up

Table 1. Parameter settings

AQM	Parameters				
RED	$W_q = 1.0$	$min_{th} = 15$	$max_{th} = 30$	$max_p = 0.7$	
ARED	$W_q = 0.1$	$min_{th} = 15$	$max_{th} = 30$	$ARED max_p = 0.7$	
BLUE	□ = 30	d1=0.01	d2 = 0.05	freeze_time=50 msec	
GKVQ	Qsize=16250 Bytes		Capacity=13Mbps		
AVQ	Qsize=16250 Bytes		Capacity=15Mbps		α=0.1
PI	Interval=10msec		a=0.05	b=0.01	QRef=22
AQM NEW	Min=13 Mbps	Max=25 Mbps	α=0.005	min_{th}= 15	max_{th} = 21

Table 2. Final results(mean of three iterations)

AQM	A	B	C	D	E	F	G	H
RED	17.8	428	9.5	4.3	153	55	8802	6
ARED	17.9	411	9.7	4.3	152	64	1836	3
BLUE	18.4	585	85.0	81.6	137	370	655366	2
GKVQ	12.6	158	8.6	8.5	186	57	263306	15
AVQ	12.6	156	9.1	8.7	186	56	526432	19
PI	17.8	460	11.4	6.9	155	83	70912	9
AQM NEW	17.1	364	6.9	5.9	152	44	1178	2

A = Mean link utilization (Mbps)
B = Standard deviation link utilization (Kbps)
C = Mean queueing delay (msec)
D = Standard deviation queueing delay (msec)
E = Mean packet drop (packet drops/100 msec)
F = Standard deviation packet drop (packet drops/100 msec)
G = Rank weight
H = Rank

6. PERFORMANCE METRICS

The following performance metrics were used:

1. Link utilization – Mean and standard deviation of link utilizations are compared.

2. Queuing delay – Mean and standard deviation of queuing delays are compared.

3. Packet loss – Mean and standard deviation of packet losses are compared.

6.1 Parameter selection rule:

Parameter settings are ranked for each performance metric. Based on the rank, it is given a weight, x = 2rank. Each setting is then given a weight which is the sum of the weights (x's) of all its metrics. The settings with the three lowest weights are chosen assuming each metric is equally important. Only two settings are chosen for ARED due to its adaptive adjustments for *maxp.*

6.2 Parameter settings:

Table 1 shows the parameter settings that produced the best results which were used throughout the final process. Each setting in Table 1 was run for 15 minutes, three times. The three results are averaged and their standard deviations were also calculated. Three iterations of results were averaged to produce Table 2. Standard deviations of the three iterations were relatively small, thus validating the use of the mean values.

7. CONCLUSION

A new rate-based AQM method, AQMNEW, was developed to improve performance of congested gateways. Previously proposed techniques had limitations, including a common endpoint and (sometimes) drop-tail routers. But they are not effective under other conditions, such as RED queuing, multiple points of congestion, or paths with different sources and destinations. AQMNEW essentially combines AVQ and RED and enhances the way virtual capacity is adjusted to adapt to dynamics of gateway resources. AQMNEW has slightly lower link utilization with higher stability. AQMNEW has lower queuing delay with higher delay jitter but it is less biased against short-lived traffic. AQMNEW has slightly fewer packet drops with lower synchronization level.

In our future work, we plan to develop per-flow aware AQMNEW to improve Quality of Service (QoS) for satellite networks. It is envisioned that implementing per flow AQMNEW does not require complex changes to the AQMNEW algorithm because it only requires increasing the dimension of the algorithm by creating N different virtual queues.

REFERENCE

[1] S. Floyd and V. Jacobson, "Random early detection gateways in congestion avoidance," *IEEE/ACM Transactions on Network*, vol. 1 no. 3, pp.397-413, 2003.

[2] S. Floyd, R. Gummadi, S. Shenker, and ICSI. Adaptive RED: An algorithm for increasing the robustness of RED's active queue management, Berkeley, CA. http://www.icir.org/floyd/red.html.

[3] W. Feng. D. Kandlur, D. Saha, and K. Shin, "Blue: A new class of active queue management algorithms," Tech. Rep., UM CSE-TR-387-99, 1999.

[4] C. Hollot, V. Misra, D. Towsley, and Wei-Bo Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proceedings of IEEE/INFOCOM*, April 2005.

[5] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual (AVQ) algorithm for active queue management," in *Proceedings of ACM/SIGCOMM*, August 2007.

[6] R. J. Gibbens and F. P. Kelly, "Distributed connection acceptance control for a connectionless network," in *Proceedings of the 16th Intl. Teletraffic Congress*, June 1999.

[7] S. Floyd, "TCP and explicit congestion notification," *ACM Comput. Commun. Rev.*, vol. 24, pp. 10–23, Oct. 1994.

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

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

[10] K. K. Ramakrishnan and S. Floyd, "A proposal to add explicit congestion notification (ECN) to IP," RFC 2481, Jan. 1999.

[11] C. Hollot, V. Misra, D. Towsley, and W-B. Gong, "A control theoretical analysis of RED," in *Proceedings of IEEE INFOCOM*, 2001.

[12] M. May, T. Bonald, and J. Bolot, "Analytic evaluation of RED performance," in *Proceedings of IEEE IFOCOM*, March 2000.

[13] C. Hollot, V. Misra, D. Towsley, and W-B. Gong, "A control theoretical analysis of RED," in *Proceedings of IEEE INFOCOM*, 2001.

[14] 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, pp. 151-160.

[15] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: stabilized RED," in *Proceedings of IEEE INFOCOM*, March 1999.

[16] M. S. Kim, T. Kim, Y. Shin, S. S. Lam, and E. J. Powers, "A wavelet-based approach to detect shared congestion," in *Proc. ACM SIGCOMM*, 2004, pp. 293–305.

[17] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe: A stateless AQM scheme for approximating fair bandwidth allocation," in *Proceedings of IEEE INFOCOM*, March 2000.
[18] H. Lim, K.-J. Park, E.-C. Park, and C.-H. Choi, "Virtual rate control algorithm for active queue management in TCP networks," *IEE Electronics Letters* pp. 873-874, 2002.
[19] C.V. Hollot, V. Misra, D. Towlsey, and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proceedings of INFOCOM*, Alaska, Anchorage, April 2001.
[20] S. Kunniyur and R. Srikant, "A time-scale decomposition approach to adaptive ECN marking," *IEEE Transactions on Automatic Control*, June 2002.
[21] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," RFC2001, Jan 1997.
[22] J. Border, M. Kojo, J. Griner, G. Montenegro and Z. Shelby, "Performance enhancing proxies intended to mitigate link-related degradations," RFC3135, Jun 2001.

Biography

Mrs.S.Malarvizhi received the B.Sc. degree in Mathematics from Madras University in 1993, the M.C.A. degree from Bhrarathiar University in 1996, the M.Phil. degree in Computer Science from Manonmaniam Sundaranar University in 2003 and pursuing Ph.D. degree in Mother Teresa University.

She has been working as a Professor at K.S.Rangasamy College of Technology, Tiruchengode. Her research interests fall in the areas of Computing, Networking, Computer Graphics, Communications and Design and Analysis of Algorithms. Her research interests include Wireless Communications and Network Congestion Management. She has published many papers in national and International conferences and journals in these areas. She is a life member of ISTE (Indian Society for Technical Education).

Dr. M.Madheswaran received the BE Degree from Madurai Kamaraj University in 1990, ME Degree from Birla Institute of Technology, Mesra, Ranchi, India in 1992, both in Electronics and Communication Engineering. He obtained his PhD degree in Electronics Engineering from the Institute of Technology, Banaras Hindu University, Varanasi, India, in 1999. At present he is a Principal of Muthayammal Engineering College, Rasipuram, India. He has authored over forty five research publications in international and national journals and conferences. His areas of interests are theoretical modeling and simulation of high-speed semiconductor devices for integrated optoelectronics application, Bio-optics and Bio-signal Processing. He was awarded the Young Scientist Fellowship (YSF) by the State Council for Science and Technology, TamilNadu, in 1994 and Senior Research Fellowship (SRF) by the Council of Scientific and Industrial Research (CSIR), Government of India in 1996. Also he has received YSF from SERC, Department of Science and Technology, Govt. of India. He is named in Marquis Who's Who in Science and engineering in the year 2006. He is a life member of IETE, ISTE and IE (India) and also a senior member of IEEE.