

# Performance Evaluation of the Particle Swarm Optimized Fuzzy Logic Congestion Detection Mechanism in Proportional Differentiated Services IP Networks

Clement N. Nyirenda, *Student Member, IEEE* and Dawoud S. Dawoud

**Abstract**—A Particle Swarm optimized Fuzzy Logic Congestion Detection (FLCD) was recently proposed for best-effort service networks. This algorithm synergistically combines the good characteristics of traditional Active Queue Management (AQM) algorithms and fuzzy logic based AQM algorithms. Its membership functions are designed automatically by using a Multi-objective Particle Swarm Optimization (MOPSO) algorithm in order to achieve optimal performance on all the major performance metrics of IP congestion control. In this paper we evaluate the performance of the FLCD algorithm in a Proportional differentiated services (PropDiffServ) network environment. Simulation results show that the PropDiffServ FLCD algorithm exhibits lower packet loss rates, higher link utilization and lower buffer occupancy for TCP traffic flows when compared with the Weighted Random Early Detection (WRED) algorithm. The FLCD algorithm also exhibits lower jitter and delay for real-time traffic.

**Index Terms**—Active Queue Management, Fuzzy Logic, Weighted Fair Queuing, Quality of Service

## I. INTRODUCTION

AS the Internet continues to evolve, it carries a wide range of traffic types such that the best-effort service cannot guarantee the diverse expectations of applications. Different applications have different Quality of Service (QoS) requirements. For instance, applications such as World Wide Web (www) and file transfers prefer low data loss rates while tolerating large delays. On the other hand multimedia applications (Voice over IP, Video-on-Demand etc.) require low delays but can tolerate a certain amount of loss rates. Therefore there is a need for careful QoS design by taking into account the different demands of different types of traffic classes and different subclasses in order to provide a fair service.

Differentiated Services (DiffServ) is a paradigm that has been proposed for QoS provisioning on the Internet [1]. Two different schemes exist for DiffServ: Absolute DiffServ [2] and relative DiffServ [3], [4]. The absolute DiffServ approach attempts to provide services with absolute profiles

but without per-flow parameter maintenance in the network core. The user receives an absolute service profile similar to that of a leased line as long as user traffic is under certain profile limitation. Relative DiffServ seeks to provide per-hop relative services to the traffic classes it supports. It does not guarantee the amount of network services provided to each class, but it intends to locally ensure that the QoS of higher priority classes will be better than that of the lower ones. There is currently a lot of research interest in relative DiffServ because of the implementation difficulties associated with absolute DiffServ.

Recently, a proportional DiffServ (PropDiffServ) model was proposed in support of relative DiffServ [4]. It ensures the quality spacing between classes of traffic to be proportional to certain pre-specified class differentiation parameters. The PropDiffServ link has a separate queue for every class. The congestion control mechanism by which DiffServ achieves the prioritization and differentiation of service can be separated into two independent and concurrent events: Scheduling and Active Queue Management (AQM). A scheduling algorithm determines the allocation of bandwidth in each class as well as packet transmission order based on pre-specified class differentiation parameters. The Weighted Fair Queue (WFQ) [5] algorithm is the commonest scheduling algorithm in PropDiffServ schemes. Separate AQM algorithms are configured on each of the queues (classes) in order to control the number of packets entering the network. Current proposals for AQM in PropDiffServ are based on extensions to the Random Early Detection (RED) [6] algorithm. These extensions are collectively known as multi-level RED (MRED). The weighted RED (WRED) is key MRED proposal that has been deployed in CISCO IOS routers [7].

Although WRED has been deployed in commercial routers today, it is often disabled by default. The RED algorithm on which it is based has a number of drawbacks: 1) it is difficult to configure its parameters; 2) it becomes unstable if traffic changes suddenly [8]; 3) it solely relies on queue length as an estimator of congestion. While the presence of a persistent queue indicates congestion, its length gives very little information as to the severity of congestion [9]. A plethora of algorithms based on mathematical analytical techniques has been proposed to address these drawbacks [10]. It has however been shown in [11], that as capacity or delay increases, all these schemes eventually become oscillatory and prone to instability. In [12], it is further pointed out that these schemes demonstrate instability with the introduction of high bandwidth-delay links because they still require a careful configuration of non-intuitive control

Clement N. Nyirenda and Dawoud S. Dawoud are with the Radio Access Technologies Centre, School of Electrical, Electronics and Computer Engineering, University of KwaZulu-Natal, Durban, South Africa (e-mail: [nyirendac@ukzn.ac.za](mailto:nyirendac@ukzn.ac.za), [dawoudd@ukzn.ac.za](mailto:dawoudd@ukzn.ac.za)).

parameters. As a result, they are non-robust to dynamic network changes. They exhibit greater delays than the target mean queuing delay with a large delay variation, and large buffer fluctuations, and consequently cannot control the router queue. Based on the fuzzy logic theory [13], several fuzzy logic AQM schemes [12], [14], [15] have been developed with satisfactory results compared to the traditional analytical approaches.

In this paper we extend the particle swarm optimized fuzzy logic congestion detection (FLCD) algorithm proposed in [16] to the PropDiffServ environment. The uniqueness of the FLCD algorithm with respect to its predecessors [12], [14], [15] lies in the fact its parameters are optimized by using the multi-objective particle swarm optimization (MOPSO) algorithm [17] in order to achieve optimal performance on all the major objectives of IP congestion control. The FLCD algorithm ensures fairness by employing a Fuzzy CHOCe (CHOose and Keep for responsive flows, CHOose and Kill for unresponsive flows) [18] which penalizes non-responsive flows and TCP unfriendly flows by dropping more packets from them as network congestion increases. We evaluate the performance of the PropDiffServ FLCD algorithm against the WRED algorithm. We use the WFQ algorithm proposed in [19] as a scheduling algorithm in both cases.

The rest of the paper is organized as follows. Section II presents an overview of the best-effort service particle swarm optimized FLCD algorithm. Section III presents the implementation of the FLCD algorithm in the PropDiffServ environment. In Section IV, we describe the simulation model and provide a discussion of results. The conclusion of this paper is presented in Section V.

## II. OVERVIEW OF THE BEST EFFORT SERVICE PARTICLE SWARM OPTIMIZED FLCD ALGORITHM

The FLCD algorithm is composed of the Fuzzy Logic Control Unit (FLCU), the Probability Adjuster (PA) and the CHOCe Activator (CA). Figure 1 shows the FLCD architecture.

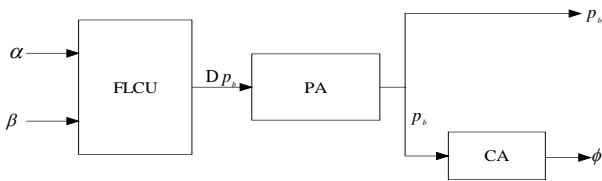


Fig. 1. The FLCD Architecture

A single FIFO buffer in which all packets are treated equally is assumed. The queue status is sampled at a period  $\tau$  of 0.002 seconds in order to obtain the queue-occupation size (backlog)  $q(t)$  and the traffic arrival rate  $r(t)$ . The backlog  $q(t)$  is translated into the backlog factor  $\alpha$  which is the ratio of backlog with respect to the Buffer Size  $BS$  as follows

$$\alpha = q(t) / BS \quad (1)$$

The packet arrival rate is determined by counting the actual number of packets that arrive at the buffer (both those that are queued and those that are dropped) during sampling period  $\tau$ . Let  $n$  denote the number of packets that arrive at the buffer during period  $\tau$ ,  $\omega_1$  denote the measuring weight and  $r_m$  the maximum packet arrival rate. The weighted

average packet arrival rate  $\overline{r(t)}$  and the packet arrival factor  $\beta$  are determined as follows

$$\overline{r(t)} = \omega_1 * \overline{r(t-\tau)} + (1-\omega_1) * n \quad (2)$$

$$\beta = \begin{cases} \overline{r(t)} / r_m & \overline{r(t)} < r_m \\ 1.0 & \overline{r(t)} \geq r_m \end{cases} \quad (3)$$

The FLCU determines the change in packet marking/dropping probability  $\Delta p_b$  by using the fuzzified values of parameters  $\alpha$  and  $\beta$ . The set of linguistic rules that govern the inference process in the FLCU is shown in Table I.

TABLE I  
The FLCU Rule Base

<b>if</b> $\alpha$ is <i>low</i> and $\beta$ is <i>low</i> <b>then</b> $\Delta p_b$ is <i>Negative Big</i> .
<b>if</b> $\alpha$ is <i>low</i> and $\beta$ is <i>medium</i> <b>then</b> $\Delta p_b$ is <i>Negative Small</i> .
<b>if</b> $\alpha$ is <i>low</i> and $\beta$ is <i>high</i> <b>then</b> $\Delta p_b$ is <i>Zero</i> .
<b>if</b> $\alpha$ is <i>normal</i> and $\beta$ is <i>low</i> <b>then</b> $\Delta p_b$ is <i>Negative Small</i> .
<b>if</b> $\alpha$ is <i>normal</i> and $\beta$ is <i>medium</i> <b>then</b> $\Delta p_b$ is <i>Zero</i> .
<b>if</b> $\alpha$ is <i>normal</i> and $\beta$ is <i>high</i> <b>then</b> $\Delta p_b$ is <i>Positive Small</i> .
<b>if</b> $\alpha$ is <i>high</i> and $\beta$ is <i>low</i> <b>then</b> $\Delta p_b$ is <i>Positive Small</i> .
<b>if</b> $\alpha$ is <i>high</i> and $\beta$ is <i>medium</i> <b>then</b> $\Delta p_b$ is <i>Positive Big</i> .
<b>if</b> $\alpha$ is <i>high</i> and $\beta$ is <i>high</i> <b>then</b> $\Delta p_b$ is <i>Positive Big</i> .

The PA computes the new packet marking probability  $p_b$  as follows

$$p_b(t) = p_b(t-\tau) + \Delta p_b(t) \quad (4)$$

Packets are either marked (if Explicit Congestion Notification is enabled) or dropped with a probability  $p_b$ . Responsive flows react to these events by reducing their sending rates thereby reducing congestion at the bottleneck link. In order to address the issue of fairness in light of non-responsive flows and network anomalies such as Denial of Service (DoS) attacks and routing loops which may dramatically flood the network as the responsive flows back off, we incorporate the CHOCe Activator (CA) which uses  $p_b(t)$  to generate a fuzzy parameter  $\phi \in [0,1]$ . Let  $P_{thresh}$  denote the CHOCe threshold then the fuzzy parameter  $\phi$  is derived as follows

$$\phi = \begin{cases} 0 & P_{thresh} > p_b \\ \left( \frac{p_b - P_{thresh}}{1 - P_{thresh}} \right)^2 & P_{thresh} \leq p_b \end{cases} \quad (5)$$

When  $P_{thresh} > p_b$  (low congestion),  $\phi$  is 0.0. During this period there is no CHOCe activity. When  $P_{thresh} \leq p_b$  (high congestion), the value of  $\phi$  increases rapidly such that more packets from non-responsive and TCP unfriendly flows are dropped at the bottleneck link. An arriving packet is picked probabilistically based on the value of  $\phi$ . This packet is compared with a randomly chosen packet from the buffer. If

they have the same flow ID, they are both dropped. Otherwise the randomly chosen packet is kept in the buffer (in the same position as before) and the arriving packet is queued if the buffer is not full; otherwise it is dropped.

Figure 2 shows the standard membership function that is used in the process of fuzzifying the parameters  $\alpha$  and  $\beta$ .

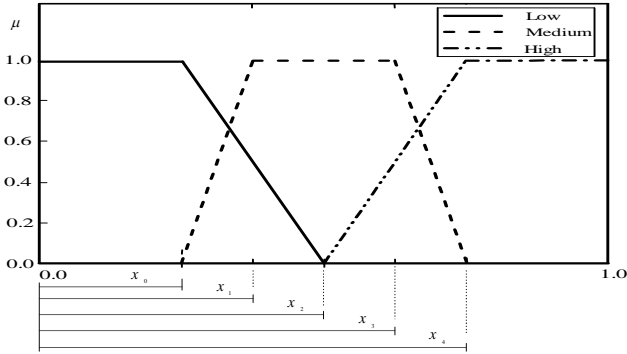


Fig. 2. Standard Membership Function for backlog and packet arrival factors

Figure 3 shows the membership function used in the defuzzification process in order to generate the change in packet marking/dropping probability  $\Delta p_b$ .

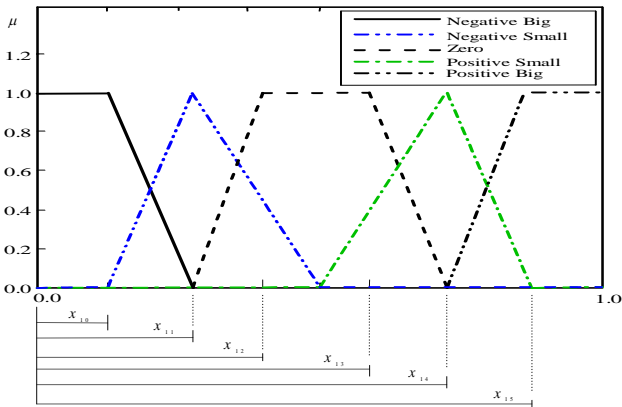


Fig. 3. Membership Function for change in packet marking probability

The 18-dimensional parameter vector  $P$  that determines the membership functions is expressed as follows

$$P = [x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}] \quad (6)$$

The definition of these elements is presented as follows:

1.  $x_0, x_1, x_2, x_3, x_4$  are parameters for the backlog factor ( $\alpha$ ) membership function (MF1) as shown in Fig. 2.
2.  $x_5, x_6, x_7, x_8, x_9$  are parameters for the packet arrival ( $\beta$ ) rate membership function (MF2) similar to Fig. 2 because  $\alpha$  and  $\beta$  use the same standard membership function.
3.  $x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}$  are parameters for the change in packet marking probability ( $\Delta p_b$ ) membership function (MF3) as depicted in Fig. 3.
4.  $x_{16}, x_{17}$  denote the maximum negative and positive variations ( $\Delta P_{neg}$  and  $\Delta P_{pos}$ ) of the change in packet marking probability. The output from the

defuzzification process which falls in the range  $[0, 1.0]$  is scaled to  $[\Delta P_{neg}, \Delta P_{pos}]$ .

Parameters for individual membership functions are always sorted in ascending order. For instance, for MF1 the following

$$x_0 < x_1 < x_2 < x_3 < x_4 \quad (7)$$

must always be true. The same applies to MF2 and MF3.

The parameters in equation (6) are modeled as single 18-dimensional particle based on which a number of similar particles are randomly created and initialized within a decision variable space whose parameters are predefined. The Adaptive MOPSO (AMOPSO) algorithm [17], which is a special case of the Particle Swarm Optimization (PSO) algorithm [20], is then used to optimize these particles. Each particle is viewed as a potential solution. The concept of PSO is that each particle randomly searches the decision variable space by updating itself with its own memory and the social information gathered from other particles. This is done over a number of generations/iterations. Unlike basic PSO which optimizes the particles based on a single objective function, MOPSO is tailored for multiple objective functions which are usually competing and non-commensurable. In this case, the optimization process generates a pool of non-dominated solutions called the *Pareto Optimal Set*. The optimization of the FLC algorithm is based on four objective functions: 1) maximizing link utilization; 2) minimizing packet loss rates; 3) minimizing link delay; 4) minimization jitter. After the optimization process, a fuzzy inference algorithm is used to draw the best compromise solution (particle) from the Pareto optimal set. This particle's parameters are used to configure the FLC algorithm. Results in [16] have shown that the FLC algorithm provides high link utilization whilst maintaining lower jitter and packet loss. The new approach also exhibits higher fairness and stability compared to its basic variant [14] and the Random Explicit Marking (REM) algorithm (REM) [21].

### III. FLC ALGORITHM IN PROPORTIONAL DIFFERENTIATED SERVICE NETWORKS

The implementation of a congestion detection mechanism in the Prop-DiffServ is shown in Figure 4. The number of service classes is denoted by  $N$ . When packet arrives at the Prop-DiffServ link, it is enqueued into a particular queue based on its class. Each queue is managed by a separate FLC algorithm. Therefore, each queue (class) is treated as a FIFO buffer just like in the best-effort FLC algorithm.

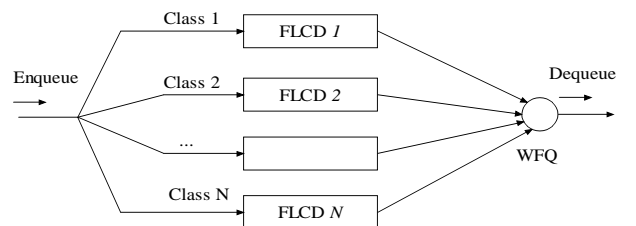


Fig. 4. Congestion Detection in the Proportional Differentiated Service Network

The Weighted Fair Queuing (WFQ) is used as the scheduling mechanism in the dequeuing routine. The WFQ algorithm is parameterized by a weight vector  $W$  where  $W_i$  is the proportion of capacity class  $i$  when there are packets available for transmission for all classes. This ensures that the performance of the low priority queues is guaranteed. The constraints for the weights are

$$W_i > 0 \quad (8)$$

and

$$\sum_{i=1}^N W_i = 1. \quad (9)$$

Recently, Joutsensalo *et al.* [19] have proposed an adaptive weighted fair queue based algorithm for channel allocation. The weights in gradient type algorithms are adapted by using revenue as a target function. We have used this version of WFQ along with the FLCD algorithm. The Nortel DiffServ implementation in NS-2.28 does not employ the WFQ scheduling algorithm. Therefore, we had to download the C++ source codes for the WFQ algorithm from [http://www.cc.jyu.fi/~sayenko/src/diffserv\\_wfq.diff](http://www.cc.jyu.fi/~sayenko/src/diffserv_wfq.diff) and <http://www.cc.jyu.fi/~sayenko/src/dsred.diff> and patch them to the DiffServ codes in NS-2.28.

#### IV. SIMULATION MODEL AND RESULTS

In this Section, we compare the performance of the PropDiffServ FLCD algorithm with the WRED algorithm. Figure 5 shows simulation topology used in this simulation.

We use the same WFQ scheduler [19] in both cases. We simulate both schemes using the Network Simulator (NS-2.28). Traffic flow is from sources  $S(1)$ ,  $S(2)$ ,  $S(3)$  and  $S(4)$  through the router to the Destination.

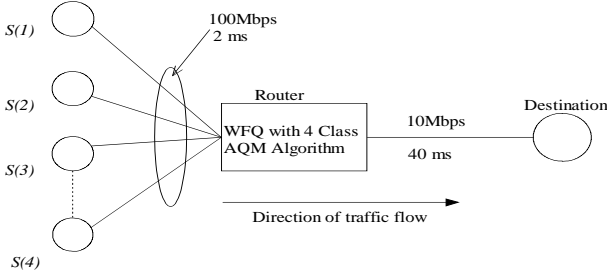


Fig. 5. Simulation Topology

A four-class PropDiffServ mechanism is implemented on the router. Class  $k$  is composed of traffic from source  $S(k)$ , where  $k = 1, 2, 3, 4$ . Traffic sources  $S(1) \dots S(4)$  are connected to the router through 100 Mbps, 2ms delay links. The router is connected to the destination node through a through 10 Mbps, 40ms delay link. The buffer size is set to 50 for each queue. ECN marking is employed for all TCP flows. Two experiments are conducted in our simulations.

##### A. Experiment 1: TCP Traffic simulation

The aim of this experiment is to compare the packet loss rate, buffer occupancy and link utilization metrics of the PropDiffServ FLCD algorithm with that of the WRED algorithm when traffic in all classes is generated by TCP sources. The simulation time for each scenario is 200sec. The simulation metrics for this experiment are presented in Table

I.

TABLE I  
Simulation parameters for Experiment 1

Class	$W_i$	Sources	Start	Stop
1	8/15	25 TCP	0	120
2	4/15	50 TCP	20	140
3	2/15	50 TCP	40	160
4	1/15	25 TCP	60	200

Table II shows the Packet loss rate and the link utilization metrics for the two algorithms. The FLCD algorithm reduces the packet loss rate by 65% while link utilization remains virtually the same in both cases. In traditional AQM algorithms, packet loss rate is directly proportional to link utilization. Any attempt to reduce packet loss rate results in drastic drops in link utilization. The situation is different with the FLCD algorithm because it is optimized to offer optimal performance on all the major metrics of IP congestion control.

TABLE II  
Packet Loss Rate and Link utilization metrics

Metric	FLCD	WRED
Packet loss rate	0.95%	2.73%
Link utilization	95.655%	94.956%

Figure 6 and Figure 7 show the aggregate backlog for the FLCD algorithm and the WRED algorithm respectively. The average values of backlog are 59.5 for FLCD and 79.04 for WRED. This implies that the buffer size requirement in the FLCD algorithm would be lower than in the WRED algorithm. The other implication is that the queuing delay incurred by packets in the FLCD algorithm would be lower than in the WRED approach. This characteristic is very important for real-time traffic.

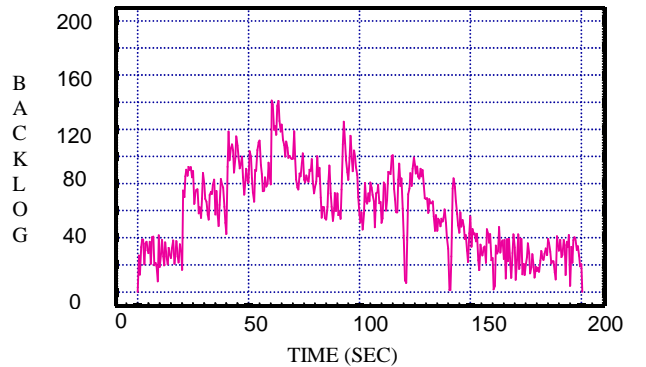


Fig. 6. Backlog for the FLCD algorithm

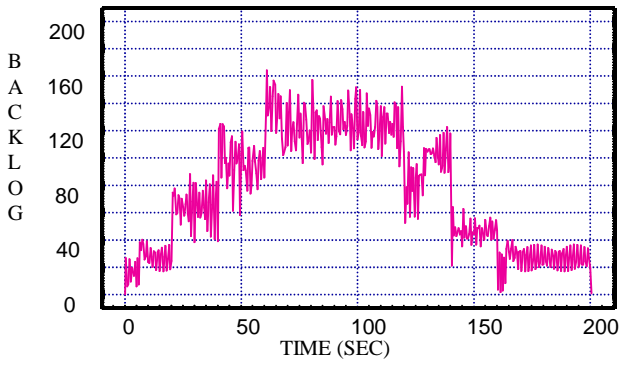


Fig. 7. Backlog for the FLCD algorithm

### B. Experiment 2: Realtime Traffic simulation

The aim of this experiment is to compare the jitter and delay metrics of the PropDiffServ FLCD with those of the WRED algorithm when one class carries higher priority video traffic while the rest of the classes carry TCP traffic of the same priority. NS-2.28's Constant Bit Rate (CBR) traffic generator is used in order to generate video traffic which is sent at a rate of 128Kbytes/sec. This approach has also been used in [22]. All the traffic flows start at 0sec and stop at 200sec. The other simulation parameters are presented in Table III. Figure 8 and Figure 9 show the results.

TABLE III  
Simulation parameters for Experiment III

Class	$W_i$	Sources
1	0.4	5,10,15,...,35 CBR UDP
2	0.2	40 FTP
3	0.2	40 FTP
4	0.2	40 FTP

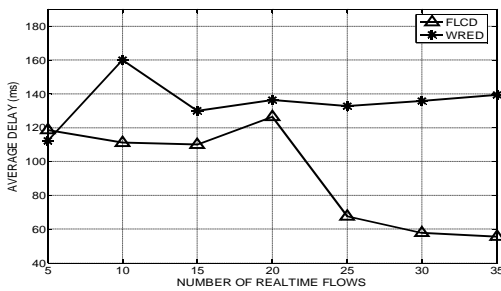


Fig. 8. Average Delay for Real-time traffic

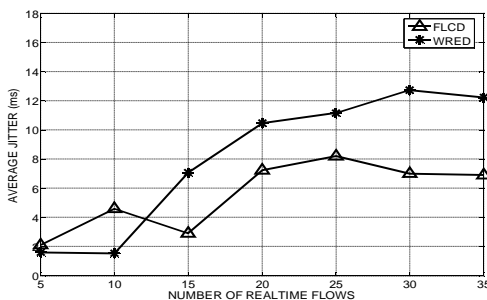


Fig. 9. Average Jitter for Real-time traffic

Figure 8 shows that the FLCD algorithm exhibits a lower

average delay. Its average delay is virtually constant when the number of real-time traffic flows is 20 or less. It however exhibits an exponential decrease in delay as the number of flows increases beyond 20. It finally saturates around 55ms which is just 15ms higher than the link propagation delay (40ms). The overall average delay in the FLCD algorithm is 92.59ms. The WRED algorithm is unstable when the number of real-time traffic sources is low. Its average delay is 112.28ms for 5 real-time flows. It then shoots to an overall high of 159ms for 10 real-time flows before stabilizing as the number of real-time flows increases. WRED registers an overall average delay of 135.25ms.

The exponential decrease in average delay in the FLCD algorithm is attributed to the fact that this algorithm becomes more aggressive in dropping real-time packets as their arrival rate increases. This helps to minimize queuing delay for the packets that have been queued successfully. This characteristic is very important for real-time traffic because this type of traffic operates within stringent time delays. If the arrival of a packet at the destination node is outside its time delay threshold, that packet is just dropped. Therefore, dropping packets aggressively as their arrival rate increases helps to get rid of useless late packets from the network.

Figure 9 shows that the WRED algorithm exhibits a lower jitter when the number of real-time flows is low i.e. 5 and 10. This advantage is however offset by the huge delay as shown in Figure 8. As the amount of real-time traffic increases, we observe that the FLCD algorithm outperforms the WRED algorithm. The overall average jitter is 5.57ms for FLCD and 8.112ms for WRED. This means that the intelligibility of real-time traffic is higher in the FLCD algorithm.

## V. CONCLUSION

This paper has extended the Particle Swarm Optimized Fuzzy Logic Congestion Detection (FLCD) algorithm to the Proportional Differentiated Service (PropDiffServ) architecture. Performance evaluation is carried out against the WRED algorithm. Results show that the FLCD approach achieves higher link utilization, lower packet loss rate, jitter and delay.

## REFERENCES

- [1] S. Blake *et al*, "Architecture for Differentiated Services," IETF RFC 2475, Dec 1998
- [2] V. Jacobson, K. Nichols, and K. Poduri, "An Expedited forwarding PHB," RFC 2598, 1999.
- [3] J. Heinanen *et al.*, "Assured Forwarding PHB group," RFC 2597, 1999
- [4] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. *IEEE/ACM Transactions on Networking*, 10(1):12–26, 2002.
- [5] C. Li, S. Tsao, M.C. Chen, Y. Sun and Y. Huang, "Proportional delay differentiation service based on weighted fair queuing," In *Proc. IEEE ICCN 2000*, 2000.
- [6] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM*

- Transactions in Networking*, 1(4):397-413, August 1993.
- [7] CISCO, "Weighted Random Early Detection on the Cisco12000 Series Router," [http://www.cisco.com/univercd/cc/td/doc/product/software/ios112/ios112p/gsr/wred\\_gs.htm](http://www.cisco.com/univercd/cc/td/doc/product/software/ios112/ios112p/gsr/wred_gs.htm): CISCO Ltd., 2002.
- [8] M.May, J.Bolot, C.Diot and B.Lyles, "Reasons not to deploy RED," In *Proceedings IWQoS'99*, June 1999, pp.260-262.
- [9] B. Wydrowski and M. Zukerman, "GREEN: Active Queue Management Algorithm for a Self Managed Internet," In *Proceedings of the ICC*, New York, vol.4, pp.2631-2635, 2002.
- [10] A. Bitorika, M. Robin, and M. Huggard, "A Survey of Active Queue Management Schemes," Trinity College. Dublin, Department of Computer Science, Tech. Rep., September 2003
- [11] S.H. Low, F. Paganini, J. Wang, S. Adlakha, and J.C. Doyle. Dynamics of TCP/AQM and a scalable control. In *Proc. of IEEE INFOCOM*, June 2002.
- [12] C. Chrysostomou *et al.* Fuzzy Logic Congestion Control in TCP/IP Best-Effort Networks, *2003 Australian Telecommunications Networks and Applications Conference (ATNAC 2003)*, Melbourne, Australia, 8 - 10 December 2003 (CD ROM - ISBN: 0-646-42229-4).
- [13] L.A. Zadeh, "Fuzzy Sets," *Inform. and Control*, vol.3, pp. 116-132, 1987
- [14] R. Fengyuan, R. Yong and S. Xiuming, "Design of a fuzzy controller for active queue management," *Computer Communications* 25 (2002) 874-883
- [15] C.Wang, B. Li, K. Sohraby and Y. Peng, "AFRED An Adaptive Fuzzy-based Control Algorithm for Active Queue Management," In *Proc. of the 28<sup>th</sup> Annual IEEE International Conference on Local Computer Networks* 2003
- [16] C.N. Nyirenda and D.S. Dawoud, "Multi-objective Particle Swarm Optimization for Fuzzy Logic Based Active Queue Management," *To appear in Proceedings of the IEEE International conference in Fuzzy Systems (FUZZ-IEEE 2006), World Congress on Computational Intelligence (WCCI 2006)*, Vancouver, British Columbia, Canada, 16-21 July 2006.
- [17] G.T. Pulido and C.A. Coello Coello, "Using Clustering Techniques to Improve the Performance of a Multi-Objective Particle Swarm Optimizer," In *Proceedings of the Genetic and Evolutionary Computation Conference*, Springer-Verlag, Lecture Notes in Computer Science Vol. 3102, pp. 700--712, Seattle, Washington, USA, June 2004
- [18] R. Pan, B. Prabhakar, and K. Psounis, "Choke - a stateless active queue management scheme for approximating fair bandwidth," *Proc. INFOCOM'00*, March 2000, pp. 942-951
- [19] J. Joutsensalo, T. Hämäläinen, M. Pääkkönen and A. Sayenko. Adaptive weighted fair scheduling method for channel allocation. *IEEE International Conference on Communication 2003 (ICC '03)*, Volume 1, pp. 228-232, 2003.
- [20] J. Kennedy and R.C. Eberhart, "Particle swarm optimization," In *Proceedings IEEE ICNN*, 1995, pp. 1942-1948.
- [21] S. Athuraliya, V.H. Li, S.H. Low, Q. Yin, "REM: Active Queue Management," *IEEE Network Magazine*, 15(3), pp.48-53, May 2001
- [22] M. Malli, Qiang Ni, Thierry Turletti, and Chadi Barakat, "Adaptive Fair Channel Allocation for QoS Enhancement in IEEE 802.11 Wireless LANs," *IEEE International Conference on Communications (ICC 2004)*, Paris, June 2004.

**Clement N. Nyirenda** received the B.Sc. degree in electrical engineering from the University of Malawi, Malawi in 2000. He is currently working toward the MSc (Eng) degree at the Radio Access Technologies Centre in the School of Electrical, Electronics and Computer Engineering, University of KwaZulu-Natal, Durban, South Africa. His current research interests lie in the application of evolutionary computation and fuzzy logic in computer networks.

**Dawoud S. Dawoud** received the B.Sc. and the M.Sc Degree from Cairo University in 1965 and 1969 respectively and Ph.D. degree from the University of Leningrad, Russia in 1972. He is a Professor of Computer Engineering in the School of Electrical, Electronics and Computer Engineering at the University of KwaZulu-Natal, Durban, South Africa. His current research interests lie in computer engineering