Fuzzy Logic Congestion Control in IEEE 802.11 Wireless Local Area Networks: A Performance Evaluation

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

***Abstract*—In 802.11 Wireless Local Area Networks, the wired and the wireless interfaces of the Access Point are characterized by the disparity in channel capacity. This presents a significant bottleneck for traffic flowing from the wired network to the wireless network. Therefore, the implementation of a congestion control mechanism in Access Points promises to be a suitable solution. Recently, a particle swarm optimized Fuzzy Logic Congestion Detection (FLCD) was proposed for IP 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 the Wireless Local Area Network environment. We compare its performance with that of ARED and the Droptail mechanism. Simulation results show that the FLCD algorithm helps to minimize UDP traffic delay, packet loss rates. In terms of throughput, the FLCD algorithm exhibits similar performance to the other schemes. Its UDP jitter is slightly higher than that of the Droptail mechanism but better than that of ARED.**

***Index Terms*——Access Point, Active Queue Management, Fuzzy Logic, Wireless Local Area Networks**

I. Introduction

IEEE 802.11 wireless Local Area Networks (WLANs) have gained strong popularity in a number of vertical markets, health-care, retail, manufacturing, warehousing and academic institutions. In contrast to traditional wired networks, these networks are characterized by mobility support, installation simplicity and flexibility, reduced cost-of-ownership and scalability. These factors have led to the widespread deployment of WLANs. Despite the growing popularity, the available bandwidth in these networks is much smaller than in wired local area networks because they are non-switched half duplex. These networks also suffer from interference from radio sources like microwave ovens, cordless phones and wireless computer devices such as Bluetooth. These networks also suffer from the hidden node problem which results in collisions and reduced channel performance [1].The maximum peak transmission rate possible in 802.11a/g stations is 54Mbps. However studies [2] have shown that, due to a large overhead per frame transmission, the maximum efficiency is only 50-60%. In addition, maximum channel throughput can only be achieved in close proximity to the Access point (AP). As distance from the Access point (AP) increases, throughput diminishes more or less rapidly due to effects of packet loss, reflection, diffraction and scattering. Studies [2] have also shown that the actual channel throughput also heavily depends on the frame payload size. When only frames are sent, the maximum throughput on the wireless channel on the wireless channel can drop below 1Mbps even at a data rate of 11 Mbps.

At present, an IEEE working group (IEEE802.11n) is working on a new standard which builds on the previous 802.11 standards by adding MIMO (Multiple-Input Multi-Output) in order to offer throughput wireless transmissions at rates greater than 100Mbps. The 802.11n will also offer a better operating distance than current 802.11 networks. While the ultimate advantage of this standard is obviously speed, it is reported in [3] that just like its preceding standards, maximum throughput can only be achieved in close proximity to the Access Point (AP). As distance increases from the Access Point (AP), throughput still diminishes more or less rapidly. It is further reported in [3] that in almost every case today, an Access Point is a shared medium: whatever throughput it can deliver is divided up among the users that connect to that one access point. This implies that for 10 users sharing one Access Point, the throughput per user translates to 10 Mbps. This new standard was designed in order to match the 100Mbps switched Ethernet capacity in the Distribution System (DS). However, new standards have arisen in switched Ethernet such that its capacity is bound to increase exponentially. While 1-Gigabit Ethernet is now widely available and 10-Gigabit products are becoming more available, the IEEE and the 10-Gigabit Ethernet Alliance [4] are working on 40, 100, or even 160 Gbps standards.

Clement .N. Nyirenda is with the Meraka Institute, Centre for Scientific and Industrial Research, Pretoria, South Africa (+27 72 1404564; e-mail: nyirendac@ ieee.org).

Dawoud S. Dawoud is with the Radio Access Technologies Centre, School of Electrical, Electronics and Computer Engineering, University of KwaZulu-Natal, Durban, South Africa. (e-mail: dawoudd@ ukzn.ac.za).

Therefore, the disparity in channel capacity between the wired and the wireless interfaces of the Access Point will continue to present a significant bottleneck in the downstream direction. The outgoing link will continue to be oversubscribed resulting in frequent buffer overflow. As a result, the implementation of a congestion control scheme in either the Gateway or the Access Point will always remain a viable solution for avoiding congestion and ensuring that delays for packets with real-time constraints are minimized in IEEE 802.11 WLANs.

The rest of the paper is organized as follows. Section II presents related work and the motivation for this work. Section III presents an overview of the self-organized particle swarm optimized Fuzzy Logic Congestion Detection (FLCD) algorithm. Section IV presents the implementation of the FLCD algorithm in the Simulation model. In Section IV, we also provide a discussion of results. The conclusion of this paper is presented in Section V.

II. Related Work And Motivation For This Work

In [5], a proxy-RED congestion control scheme is proposed for WLANs. The main idea of this scheme is to reduce overhead at the access point by implementing the Random Early Detection algorithm [6] at the gateway. The instantaneous queue length at the access point is sampled periodically to calculate the estimated average queue length, which is used for determining the drop/marking probability at the gateway. Results [5] show that this congestion control scheme significantly improves packet loss rate and goodput for a small buffer, and delay for a large buffer. The proxy-RED approach is based on the WLAN architecture in Figure 1.

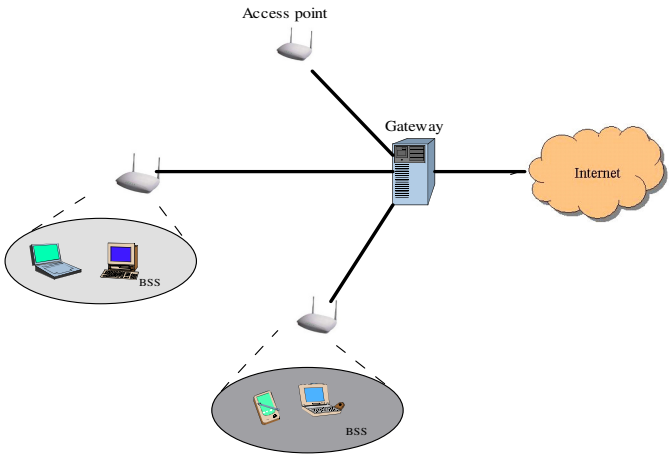

Figure 1. Gateway Centric WLAN

However, WLAN architectural trends [7] show that the WLAN architecture in Figure 1 is not highly scalable because the cost of deploying a large scale WLAN network dramatically increases as the network expands. An emerging architecture which presents a possible solution to the scalability problem is the (static) multi-hop.11 (*mesh*) network (See Figure 2).

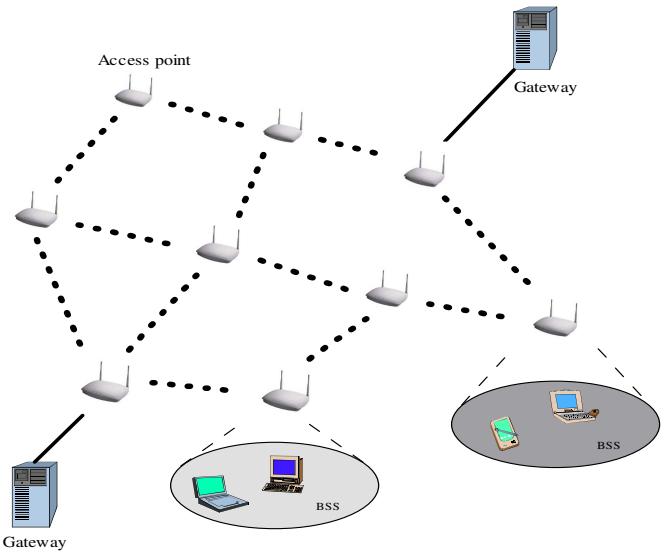

Figure 2. Wireless AP-to-AP Mesh Network

Interest in this approach is indicated not only by the Mesh Task Group within IEEE 802.11 but also mesh solutions offered by several companies [8]. The deployment of a proxy-based (gateway-centric) congestion control scheme in the future *wireless AP-AP mesh network* is not a scalable approach because the gateway is directly connected to one access point only. Apart from that there could be other gateways feeding the same WLAN. Therefore, congestion control schemes will still have to be implemented in the access point because the access point has direct control to a particular Base Service Set (BSS).

In wired networks, Fuzzy Logic Control (FLC) has been proposed for congestion control [9, 10, 11, 12]. Results show that Fuzzy Logic Controllers achieve lower packet loss rates, more stable queue lengths and higher goodput and link utilization than traditional Active Queue Management schemes based on RED [6], its variants [13] and control theoretic approaches [14].Fuzzy Logic Controllers exhibit better performance because they have the ability to cope with the difficulties, complexity and the uncertainty posed by the growth of the Internet. This can be compared with the fact that the difficulties experienced in obtaining a mathematical model of the Internet limit the capacity of the traditional Active Queue Management schemes to address the problem of congestion. These attributes motivated us to implement a recent mechanism known as the self-organized particle swarm optimized FLCD algorithm [11, 12] in IEEE 802.11 WLANs.

III. Overview of the Particle Swarm Optimized FLCD Architecture

A. *The FLCD Architecture*

The FLCD algorithm is composed of the Fuzzy Logic Control Unit (FLCU), the Probability Adjuster (PA) and the CHOKe

Activator (CA). The FLCD algorithm is given in a block diagram form in Figure 3.

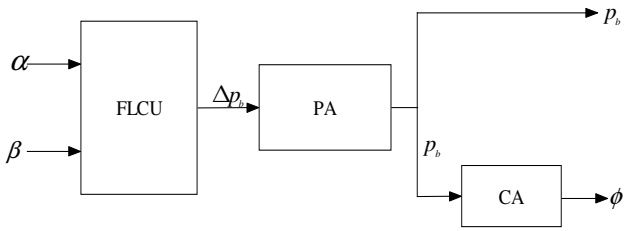

Figure 3. The FLCD Architecture

A single FIFO buffer in which all packets are treated equally is assumed. The queue status is sampled at a period τ 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 α which is the ratio of backlog with respect to the Buffer Size BS as follows

$$\alpha = q(t) / BS \tag{1}$$

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 τ. Let n denote the number of packets that arrive at the buffer during period τ, ω_1 denotes 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 β are determined as follows

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

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

The FLCU determines the change in packet marking/dropping probability Δp_b by using fuzzified values of parameters α and β. A rule base is used in the process of mapping the two inputs α and β to the output Δp_b based on their degrees of membership. A typical rule would be as follows: If α is *medium* and β is *medium* then Δp_b is zero. Two membership functions are defined for fuzzifying α and β. One membership function is defined for the defuzzification of Δp_b. The PA computes the new packet marking probability p_b as follows

$$p_b(t) = p_b(t-\tau) + \Delta p_b(t) \tag{4}$$

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 CHOKe Activator (CA) which uses $p_b(t)$ to generate a fuzzy parameter $\phi \in [0,1]$. Let P_{thresh} denote the CHOKe threshold then the fuzzy parameter ϕ is derived as follows

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

When $p_{thresh} > p_b$ (low congestion), ϕ is 0.0. During this period there is no CHOKe activity.

When $p_{thresh} \leq p_b$ (high congestion), the value of ϕ 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 ϕ. 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.

B. Parameter Optimization using Adaptive Multi-objective Particle Swarm Optimization (AMOPSO)

In order to design a mechanism that addresses all the major objectives of Internet congestion control, which are usually conflicting, the problem is modeled as a multi-objective (MO) problem [11]. Link utilization, packet loss rate, link delay and jitter are used as objectives. Various parameters are captured from the input and output membership functions of the FLCD algorithm in order to generate a particle vector. This particle vector is optimized by using the AMOPSO algorithm [15], a special case of Particle Swarm Optimization (PSO) [16]. A number of similar particles are randomly created and initialized within a decision variable space whose parameters are predefined. 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, AMOPSO 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*. After the optimization process, a simple linear Fuzzy inference algorithm [17] is used for extracting the best compromise solution from the Pareto Optimal Set. The best compromise solution is then used in the implementation of the practical FLCD algorithm.

C. Self Learning and Organization

The optimization process in the previous section is done offline and is based on a single optimization script. Therefore, the performance of the FLCD algorithm may not be optimal

under different network conditions due to the fact that the IP environment is characterized by dynamic traffic patterns. Self-learning and organization is employed in order to fine tune the FLCD algorithm in light of traffic variations, system dynamics and other disturbances without disrupting the structure of the optimized membership functions. In order to achieve this, the following mechanisms are used: an RTT based sampling mechanism and a self-learning and adaptation mechanism. The former adapts the rate at which the queue is sampled based on the link propagation delay. The latter adjusts the rule consequents of the FLCD algorithm based on the size of the queue and the amount of lost packets.

IV. Simulation Model And Results

To investigate the performance of the self-organized FLCD algorithm in WLAN, we run some simulations on the network topology shown in Figure 4.

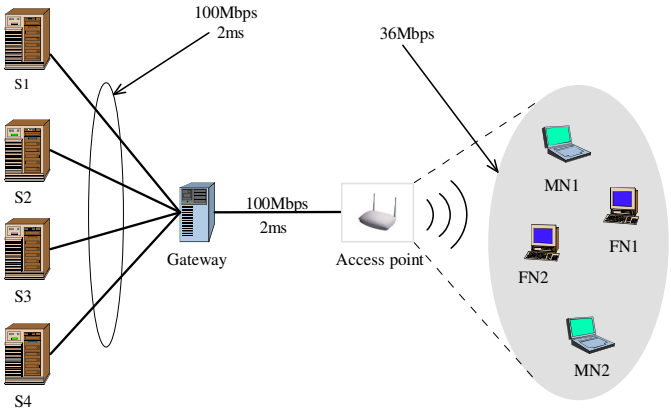

Figure 4. WLAN Simulation Topology

Servers S1, S2, S3 and S4 are connected to the Gateway which is connected to the Access Point. The bandwidth and propagation delay between the servers and the Gateway and between Gateway and the Access Point are set to 100Mbps and 2ms respectively. The Access Point feeds two fixed nodes (FN1 and FN2) and two mobile nodes (MN1 and MN2) in the 36Mbps WLAN architecture. We compare the performance of the self-organized FLCD algorithm with that of the Adaptive RED [13] and the Droptail mechanism which is used in WLAN networks at present. These schemes are configured in the Access Point at the WLAN interface. We use throughput, packet loss rate, UDP traffic delay and jitter as performance metrics.

We vary the buffer size by using 50,100, 150 up to 400 packets. The simulations run for 250 seconds. The standard web traffic generator included with NS-2.28 is used for our simulations, with the following parameter settings: an average of 30 web pages per session, an inter-page parameter of 0.8, an average page size of 10 objects, an average object size of 400 packets and a ParetoII shape parameter of 1.002. 48 FTP flows and 4 web traffic flows, each having 4 sessions, are configured to flow from the servers to all the WLAN nodes throughout the simulation period. To provide background traffic on the return path, we configure 4 web traffic flows, each having 4 sessions, to flow from the WLAN nodes to the servers. Audio, video and basic UDP traffic was generated based on Table I.

TABLE I
Real Time Traffic Simulation Parameters

	Audio	Video	Background
Packet Size	160 bytes	1280 bytes	1500 bytes
Packet Interval	20 ms	10 ms	12.5 ms
Flow Rate	8 Kbytes/s	128 Kbytes/s	120 Kbytes/s

Video traffic flows from Servers S1 and S2 to the mobile nodes in the interval [40s-70s]. Audio traffic flows from Servers S1 and S2 to the mobile nodes in the interval [130s-160s]. Basic UDP traffic flows from all the 4 Servers to all the WLAN nodes in the intervals [25s-30s] and [215s-220s]. Figure 5- Figure 8 show the results.

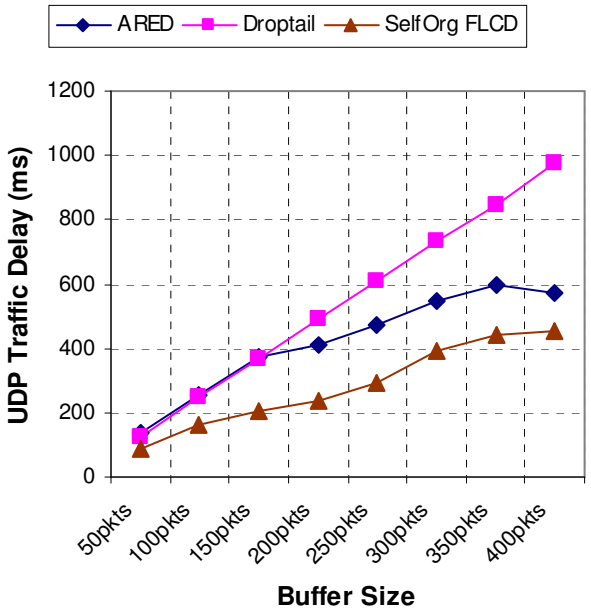

Figure 5: UDP Traffic Delay with varying Buffer Size

Figure 5 shows that the Self organized FLCD algorithm exhibits the lowest UDP traffic delay ranging from 85.7ms (for a buffer of 50 packets) to 455.7ms (for a buffer of 400 packets). ARED comes second with UDP traffic delay ranging from 137.76ms to 571.34ms (for a buffer of 400 packets).The Droptail mechanism comes third with the UDP traffic delay ranging from 125.47ms (for a buffer of 50 packets) to 974.6ms (for a buffer of 400 packets).In terms of averages, the UDP traffic delay performance is as follows: 282.587ms for the Self organized FLCD, 421.4ms for ARED and 548ms for Droptail. The Self organized FLCD algorithm achieves the lowest delay

because the optimization process used in its design enables it to maintain a very short queue [11]. The Droptail mechanism exhibits the highest delay because its queue is perpetually long. This happens because this mechanism does not employ any special mechanism to limit the length of the queue. This implies that more UDP packets would be discarded at the receivers in Droptail mechanism due to late arrival because of the high link delay. On the other hand, less UDP packets would be discarded at the receiver in the Self organized FLCD mechanism. This is an important characteristic because the packets that arrive at the receiver will have consumed expensive network bandwidth along the way. Dropping such packets at the receiver would be a huge loss. In Figure 5, we also observe that in all cases, UDP traffic delay increases with varying proportions as buffer size increases. This happens because queuing delay increases as buffer size increases.

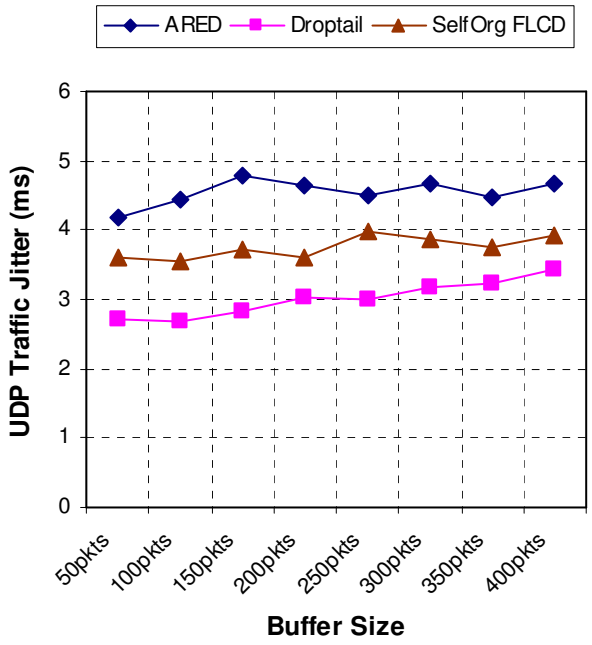

Figure 6. UDP Traffic Jitter with varying Buffer Size

Figure 6 shows that the Droptail algorithm exhibits the best UDP traffic jitter with an average value of 3.01ms.The Self organized FLCD algorithm comes second with an average value of 3.75ms while ARED comes third with an average value of 4.547ms The optimization results in [11] reveal that jitter and delay are non-commensurate i.e. they cannot be improved at the same time. This is proved again in this case because the Droptail mechanism exhibits the lowest UDP traffic jitter at the expense of UDP traffic delay while the Self organized FLCD algorithm exhibits the shortest UDP traffic delay at the expense of UDP traffic jitter. A closer look at the jitter performance shows that jitter values for the three mechanisms are very close such that practically their performance in terms of jitter would not be very different.

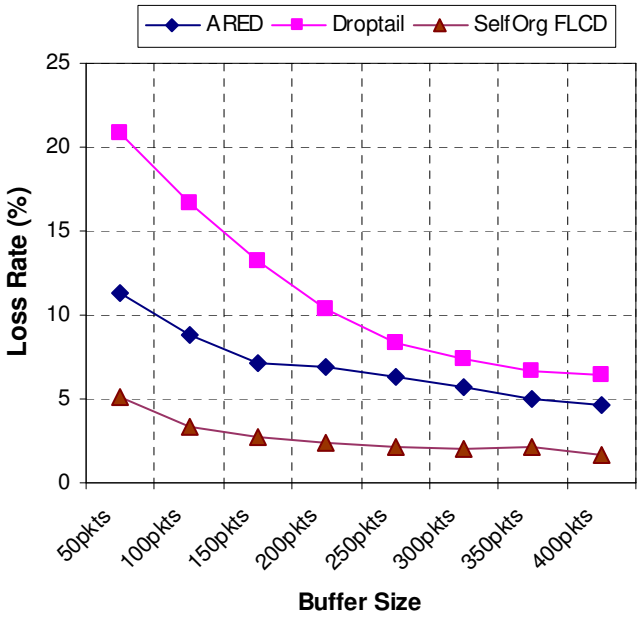

Figure 7. Packet Loss rate with varying Buffer Size

Figure 7 shows that the Self organized FLCD algorithm exhibits the lowest packet loss rate. This is more marked for small buffer sizes. On average, the FLCD algorithm exhibits an average loss rate of 2.687%. ARED, which ranks second, has an average loss rate of 6.98% while the Droptail mechanism ranks third with an average of 11.226%. The Droptail mechanism performs poorly because it does not detect incipient congestion and as a result it fails to minimize packet loss rates.

The Self organized FLCD achieves the best result because of several reasons. The first one is because of the strength of fuzzy logic in dealing with reasoning that is approximate rather than precise. This is a suitable characteristic for congestion control in IP-based networks where it is difficult to obtain a precise mathematical model by using conventional mathematical analytical methods such as ARED. Secondly, as opposed to ARED, which uses queue length only as a measure of congestion, the FLCD algorithm uses both queue length and packet arrival rate in determining the level of congestion. A combination of these parameters gives a better picture of the severity of congestion. Thirdly, the superior performance of the FLCD algorithm is attributed to impact of the optimization process [11] and the role of the self organization structures [12] which fine-tune the system based on the prevailing conditions.

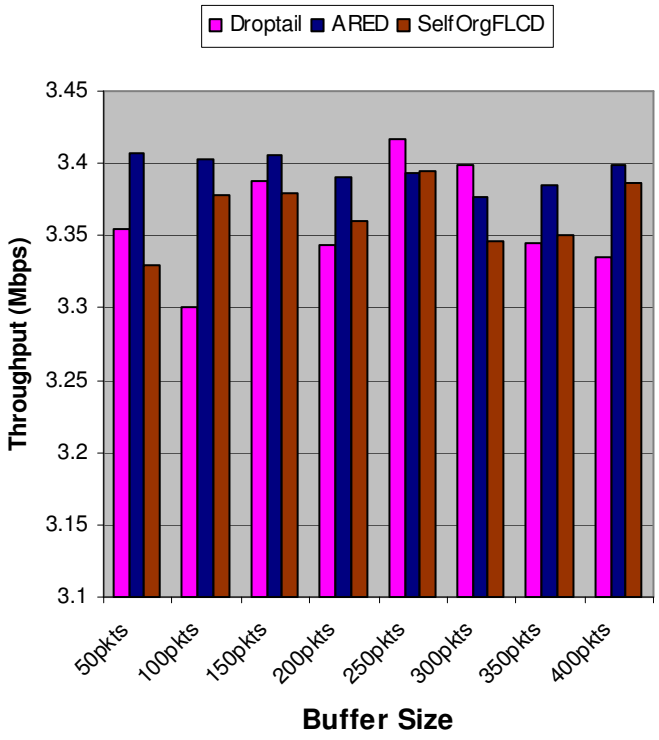

Figure 8.Throughput with varying Buffer Size

Figure 8 shows that all the three schemes exhibit low average throughput rates. The average values are as follows: 3.395Mbps for ARED, 3.366Mbps for the Self organized FLCD algorithm, 3.361Mbps for Droptail and 3.36Mbps for the basic Fuzzy AQM algorithm. These values account for less than 10% of the available bandwidth i.e.36Mbps. From these results, we observe that the introduction of an AQM in WLAN environments does not improve throughput significantly.

V. Conclusion

This paper has extended the concept of Fuzzy logic congestion detection to WLAN networks. In this implementation, the FLCD algorithm helps to minimize UDP traffic delay, packet loss rates. In terms of throughput, the FLCD algorithm exhibits similar performance to the other schemes. Its UDP jitter is slightly higher than the Drop-tail mechanism but better than that of ARED.

In future, we intend to focus on the design of a lightweight Fuzzy Logic Congestion Detection algorithm in order to reduce the processing overhead on the Access Points.

References

[1] J. Schiller, "Mobile Communications," ISBN 0-201-39386-2, 2000.

[2] Martin Kappes and Sachin Garg, "An Experimental Study of Throughput for UDP and VoIP Traffic in IEEE 802.11b Networks," *In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, March 2003

[3] Cox, John. "Wireless LAN throughput on the rise," Network World Fusion, 26 September, 2003 http://www.nwfusion.com/news/2003/092980211n.html.

[4] http://www.cisco.com/warp/public/cc/techno/media/lan/gig/tech/10gig_sd.htm

[5] S. Yi *et al.* "Proxy-RED: An AQM scheme for wireless LANS", in *Proc. ICCCN*, Chicago, Oct. 2004.

[6] Sally Floyd and Van Jacobson, "Random early detection gateways for congestion avoidance", *IEEE/ACM Transactions in Networking*, 1(4):397-413, August 1993

[7] Jing Zhu, and Sumit Roy, "802.11 Mesh Networks with Two-Radio Access Points", ICC 2005, May 2005, Seoul, Korea.

[8] http://www.meshnetworks.com

[9] F Ren *et al.*, "Design of a fuzzy controller for active queue management", *Computer Communications* 25 (2002) 874-883

[10] 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).

[11] C.N. Nyirenda, D.S. Dawoud, "Multi-objective Particle Swarm Optimization for Fuzzy Logic Based Active Queue Management", *In Proceedings of the 15th IEEE International Conference in Fuzzy Systems as part of the Fourth IEEE World Congress on Computational Intelligence (IEEE WCCI 2006)*, Vancouver, BC, Canada, pages: 2231 - 2238, 16-21 July 2006.

[12] C.N. Nyirenda, D.S. Dawoud, "Self-Organization in a Particle Swarm Optimized Fuzzy Logic Congestion Detection Mechanism for IP Networks", Submitted to *Scientia Iranica, International Journal of Science and Technology.*

[13] S. Floyd, R. Gummadi, S.Shenker. Adaptive RED: An Algorithm for increasing the robustness of RED's Active Queue Management. *Technical report ICSI*, August 2001.

[14] C. V. Hollot *et al*. "Analysis and Design for controllers for AQM Routers Supporting TCP Flows", *IEEE Transactions on Automatic Control,* vol. 47, no. 6 , pp. 945-959, June 2002.

[15] 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.

[16] J. Kennedy and R.C. Eberhart, "Particle swarm optimization," In *Proceedings IEEE ICNN*, 1995, pp. 1942-1948

[17] B. Zhao and Y. Cao, "Multiple objective particle swarm optimization technique for economic load dispatch," *Journal of Zhejiang University SCIENCE 2005*, 6A(5):pages 420-427