Indonesian Journal of Electrical Engineering and Computer Science
Vol. 23, No. 3, September 2021, pp. 1419~1431
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v23.i3.pp1419-1431

Survey on: A variety of AQM algorithm schemas and intelligent techniques developed for congestion control

Amar A. Mahawish, Hassan J. Hassan
Computer Engineering Department, University of Technology, Iraq

Article Info

Article history:

Received May 24, 2021
Revised Jul 30, 2021
Accepted Aug 7, 2021

Keywords:

AQM
Fuzzy logic
GA
PID controller
RED

ABSTRACT

The congestion on the internet is the main issue that affects the performance of transition data over the network. An algorithm for congestion control is required to keep any network efficient and reliable for transfer traffic data of the users. Many Algorithms had been suggested over the years to improve the control of congestion that occurs in the network such as drop tail packets. Recently there are many algorithms have been developed to overcome the drawback of the drop tail procedure. One of the important algorithms developed is active queue management (AQM) that provides efficient congestion control by reducing drop packets, this technique considered as a base for many other congestion control algorithms schema. It works at the network core (router) for controlling the drop and marking of packets in the router's buffer before the congestion inception. In this study, a comprehensive survey is done on the AQM Algorithm schemas that proposed and modification these algorithms to achieve the best performance, the classification of AQM algorithms based on queue length, queue delay, or both. The advantages and limitations of each algorithm have been discussed. Also, debate the intelligent techniques procedure with AQM algorithm to achieve optimization in performance of algorithm operation. Finally, the comparison has been discussed among algorithms to find the weakness and powerful of each one based on different metrics.

Corresponding Author:

Amar A. Mahawish
Computer Engineering Department
University of Technology
Baghdad-571-15-11, Iraq
Email: ce.19.17@grad.uotechnology.edu.iq

1. INTRODUCTION

The internet today widely used by many applications and provides a variety of information using standard communication protocols such as TCP/IP. The increasing user's demand on the internet makes a challenge in providing services and quality of service "QoS". The congestion problem is the main factor that affects internet performance. The congestion phenomena occur when many users send a large amount of data on the same link that exceeded its capacity. The congestion effect QoS by increase latency and drop of sending packets.

The congestion control "CC" is a mechanism and technique used to reduce congestion on the wired and wireless networks as in [1] by using routing information strategy to enhance the internet performance. The CC allows a sender to adjust its rate based on measuring the congestion status in the network. There are two approaches to solve this problem: prevent congestion before it occurs or detects and remove congestion after it occurs.

The transmission control protocol/internet protocol "TCP/IP" is the internet's primary protocol. The TCP congestion control work in the same concept of additive-increase, multiplicative-decrease "AIMD" [2]. The AIMD has two lines, the first one is called the efficiency line, and the second line is called the fairness line. For optimal control, the two flows must reach the optimal point, the intersection point of the efficiency line and the fairness line. The TCP uses an end to end flow control to avoid congestion [3]. The TCP has four phases of congestion control algorithms, slow start "SS", congestion avoidance "CA", fast retransmit, and fast recovery.

There are two parameters besides SS and CA, the congestion window "cwnd" and advertise window "rwnd". The cwnd is sender-side that specify the number of packets the sender can send before receiving acknowledge "ACK", while the rwnd is receiver-side that determine the number of packets that can accept. The TCP control selects the minimum of cwnd and rwnd for transmission data. The TCP sender used slow start and congestion avoidance behavior as shown in Figure 1. In this case the TCP slowly send packets into network to estimate the capacity of links through routing, this procedure used in order to avoid congesting the network. The slow start threshold "ssthr" parameter allows the controller to specify which phase is SS or CA. the SS increasing exponentially by duplicate the cwnd value while the CA was increasing linearly by increasing cwnd by one. To determine the new ssthr, the cwnd divide in half. This new ssthr decided when all packets dropped. The TCP data transmission in unknown network capacity, so the SS can quickly discover the network link capacity available for safe transmission.

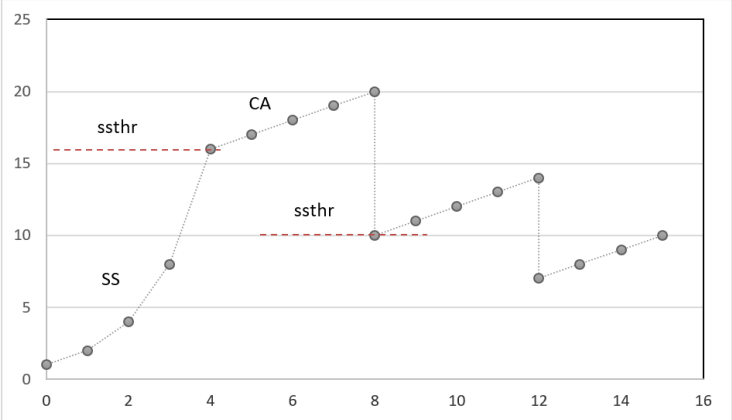

Figure 1. TCP congestion control behavior

The TCP uses the sliding window method to throttle the sender rate, which is a certain number of packets that the sender can send before ACK is received. This method is used to maximize the utilization of bandwidth. The initial value of cwnd is initial window "IW". The IW value specifies the size of data transmission, whether slow or aggressive data transmission. The reason behind increasing IW as found in [4] to 10 due to internet traffic, mostly is web traffic which has short-lived connections. The increasing IW is helpful to increase internet performance by probe the available bandwidth on the network.

The Queuing, marking, and dropping [5] are found in any network system using finite memory "buffer" to protect itself from congestion collapse. The queuing means the packets buffered in a queue when space in that buffer available. The system drains its buffer based on schedule algorithms (such as first-in-first-out "FIFO"). If the sender sends data at a rate higher than outgoing link capacity, the packets that reside in the buffer will increase, causing a buffer overflow, which produces congestion in the network. The active queue management "AQM" algorithms proposed to start marking or drop the packet before the queue full to handle congestion. The sender interprets the dropped packet as a signal to which congestion occurs, and reducing its rate is required. The type of service "ToS" is specified in the second byte of the IPv4 header and the traffic class in the IPv6 header to classify the traffic types flow. For example, critical network traffic needs low latency like voice or streaming media, while non-critical network traffic needs best effort services like file transfer or web traffic.

This review study will examine the development of AQM algorithms and their advantage and drawback of each algorithm. Section 2 has a brief description of AQM and classification algorithms. The Section 3, some essential algorithms based on queue length have been discussed. Section 4 lists algorithms that achieved the best performance when different types of services flow based on queuing delay. Section 5

studied the algorithms that utilize the advantages of both classes of AQM schemas based on queuing length and queuing delay. Section 6 described the intelligent optimization procedures to enhance the AQM algorithms. Section 7 comparison between discussed algorithms and the conclusion is the last section.

2. ACTIVE QUEUE MANAGEMENT AQM ALGORITHMS

The active queue management "AQM" [6] is a technique used to manage congestion in network devices (such as router's queue) before the buffer full. The AQM was developed to eliminate the drop-tail "DT" FIFO queue drawbacks. First, the DT drawbacks discard the new incoming packets when the buffer overflow, so the congestion notification is done when the queue becomes full. Second, the DT may not achieve farness queue "FQ", which means a few connections sized the queue buffer, which prevents the other new connections. Third, the large packets burst may increase the latency of a small burst of packets. Finally, the DT makes control loop synchronization. The AQM has been improved [7] to enhance the performance of internetwork.

The AQM was designed to achieve less packet loss, low queuing delay, and high link capacity utilization. AQM ensures dropping or marking packets before a router's buffer become full. AQM work as queue size regulator in control theory point of view. The input to AQM algorithm is level of congestion in queue and the output the probability of marking or dropping the packets. AQM send early feedback to sender for increase or reduce the sending packets to avoid dropping packets as well as high utilize the available bandwidth.

There are many AQM schemas was proposed to achieve these goals. The Random Early Detection was the earliest algorithm designed as AQM congestion control. The AQM algorithms can be classified into three main methods to measure network devices' congestion Figure 2. The first method is based on queue length "QL", which measures how many packets reside in the router's queue. The second method is based on the meantime that a packet spends in a queue. The third method is based on both the length and time delay of packets in the queue.

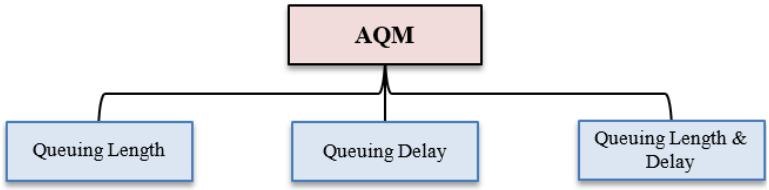

Figure 2. Classification of AQM algorithms

3. AQM ALGORITHMS BASED ON QUEUING LENGTH

In this AQM schemas, the controller reaction is based on measuring the network device's queue. This schema is the earliest algorithm to adopt AQM behavior to mark/drop packets before the queue full. The overall advantage of these schemas is eliminating the DT mechanism's drawback and improving the QoS of the internet. Simultaneously, the general disadvantage is the problematic tuning of the parameter to achieve an optimal result.

3.1. Random early detection-red

RED is designed as AQM technique [8], [9] to perform packet queue management to avoid drawback of DT and improve the packet transmission during network congestion. The RED algorithm computes the average queue size "*avg*" and compared it with the specific threshold. The RED determined the probabilistic mark/drop for each new incoming packets if the *avg* less than the minimum threshold "min_{th}" the probability drop "P_d" is zero and there are no packets drop when the *avg* exceeded the maximum threshold "max_{th}" the probability reaches to one, and all new incoming packets dropped. The *avg* queue length can be calculated by using exponential weighted moving average "EWMA" algorithm as shown in (1).

$$avg = (1 - W_q)avg + w_q * q \quad (1)$$

Where W_q is predefined weight queue coefficient and is the queue length. When the value of *avg* between min_{th} and max_{th} the probability assigned to incoming packets to drop these packets can be calculates by (2).

$$if\ min_{th} < avg \leq max_{th} \quad then\ the \quad P_d = max_p \frac{avg - min_{th}}{max_{th} - min_{th}} \tag{2}$$

Where the max_p is a maximum dropping probability. The RED algorithm is shown in Figure 3. The drop probability in (2) drived from semicontinuous piecewise linear function [10] of the average queue length as shown in Figure 3, so when the *avg* increase the P_d increases until reach to maximum value (which is P_d) and then all incoming packets will drop.

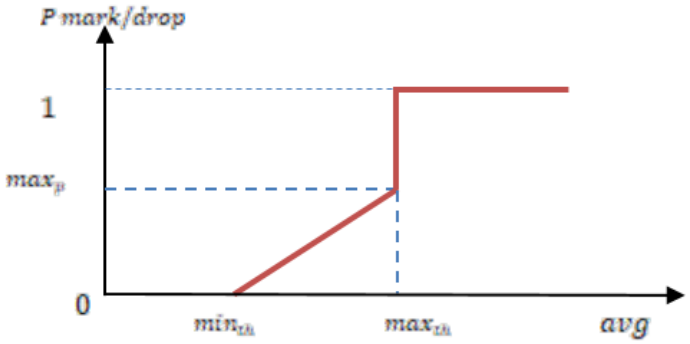

Figure 3. RED algorithm

The advantages of the RED algorithm have overcome the problem of the full queue and global synchronization. The disadvantages are the complex tuning parameter, unfair queue by having the same drop rate perform to all traffics type because it cannot differentiate between ToS and RED unable to stabilize the computation of *avg* value between two the thresholds when traffic load rapidly change. The RED algorithm have been developed to improve their performance as shown in next paragraphs.

The fair random early detection "FRED" [11] was designed to eliminate RED's unfair drop when different traffics types are used. In RED, all connections will have the same loss rate when queue length exceeded the threshold. The slow rate connection using less than its fair bandwidth will have the same packet loss rate as a higher rate connection, which is unfair utilizing the bandwidth for this kind of connection. The FRED allows fairness for a different type of traffic connection by assigning minimum packets of each flow that would enable buffering before any drop. The FRED maintains *avg* for each flow type, if *avg* exceeded the thresholds the drop probability will calculate to mark/drop packets of this flow so that the FRED will penalize the aggressive flow.

The weighted random early detection "WRED" [12] was designed to drop packets based on traffic flow weight or priority in addition to multiple virtual queues and queue threshold for each virtual queue. The IP precedence in the IPv4 header is used to give priority for each packet and distinguish among services flows. This precedence value increases or decreases based on the importance of the packet. The work of WRED is to drop lower-priority packets by setting various drop-probability functions for each priority level. Each traffic class has a virtual queue and a queue threshold. The CISCO routing platforms support this algorithm in recent series products [13].

An adoptive random early detection "ARED" was suggested by [14], [15] to solve RED parameter tuning. The ARED performs self-configuring of RED parameters based on mixed traffic types. The ARED adopting autotune of max_p to keep *avg* target range between min_{th} and max_{th}. There are many modifications build over ARED [16] by making it more robust for deployment in a network device. This modification, for example, not only keeping the *avg* between two thresholds but to keep *avg* in halfway between min_{th} and max_{th} to increase the performance of the algorithm.

Another modification of RED is gentle random early detection "GRED" [17], [15], which used gentle parameters to improve RED's behavior. The GRED avoid a sudden change in the drop probability from max_p to 1 when the *avg* exceeded the max_{th}. The modification on RED implement by adding three thresholds min_{th}, max_{th}, and $doublemax_{th}$. In GRED gradual varying of drop packets probability from max_p to 1 when the *avg* exceeded max_{th} to $doublemax_{th}$. The three threshold level min_{th}, max_{th}, and $doublemax_{th}$ are used to give more robust to tuning max_p and *avg* parameters. The GRED was designed to eliminate the problem of stabilizing the *avg* value at a certain level. The adaptive GRED "AGRED" in [18] was developed from GRED to increase performance by determining the initial drop using a specific formula. The initial drop starts from max_p to 0.5 as the *avg* length moves from max_{th} to $doublemax_{th}$. Enhanced AGRED "EAGRED"

[19] was proposed to improve the response of the router's buffer to drop packets as well as this algorithm gives better performance by reducing delay and packet loss. The EAGRED, unlike the AGRED and GRED, when using the fixed value of queue weight (e.g., 0.002) to calculate the *avg* which may lead delay in congestion response.

The static probability may cause high drop packets when the drop probability value is increased, while the network's performance decreases due to congestion when the probability value is small. The dynamic algorithm was proposed to calculate drop packets based on congestion status. The dynamic GRED "DGRED" [20] same as GRED by having $doublemax_{th}$ but provides a faster response to congestion events and enhances the network's performance when using a different type of services flow. The DGRED depends on the three-state markov modulated bernoulli arrival process (MMBP-3) to support three types of the traffic flow by performing correlation among these network traffics. An extension of DGRED is stabilized dynamic GRED "SDGRED" [21], [22] to provide dynamically stabilizing the *avg* between min_{th} and max_{th}. This dynamic algorithm uses dynamic increase or decrease of max_{th} and $doublemax_{th}$ value by stabilizing the *avg* value around min_{th} to adjust the calculation of dropping probability.

3.2. CHOKe

There are different types of connection flow [7], the flow that responds to congestion control signal called TCP-friendly flows, while the other flows do not respond to reduce the flow rate when congestion occurs this named unresponsive flows or aggressive flows such as UDP video or voice connection. The fairness queue "FQ" will not achieve when these two types of flow sharing the same bandwidth. As a result, the aggressive flow will size the most available bandwidth when congestion occurs.

To save TCP-friendly flows and to achieve FQ among different flow connection types, the CHOKe algorithm [23] was designed. The CHOKe algorithm follows the drop candidate packet procedure. This procedure work when the average queue size *avg* exceeded the min_{th}, so, the arriving new packet is compared with a randomly select packet from the buffer, the selected packet called candidate, if they are from the same flow type, both of them are dropped. Otherwise, the candidate packet is kept in the buffer, and the latest arriving packet is dropped based on probability value. The drop probability value is computed as in the RED algorithm, which is affected by the buffer's congestion level. As in RED, there are no packets drop if the *avg* less than min_{th} or all the new arriving packets dope if the *avg* passed the max_{th}. The aggressive flow has more packets in the buffer than the TCP-friendly flow when congestion occurs. So the chance to select candidate packet from aggressive flow higher than the other flow types and this high drop rate from this type of flow. The advantage of this algorithm has a low processing cost. The disadvantage is the pre-flow type information needed for its process. The CHOKe is a stateless technique about the number of aggressive flows and produces less performance when multiple aggressive flows present.

The CHOKe-RH "CHOKe with recent drop history" algorithm [24] was designed to protect the TCP-friendly flows and eliminate the limitations CHOKe algorithm, which spans from stateless to stateful. The CHOKe-RH uses two phases of comparisons, the first phase is the initial comparison, and the second phase is the penalty for aggressive flows. This technique store history information that has recently dropped packets flow-ids and then use it to penalize the aggressive flows.

3.3. Hash table and circular buffer-HTCB

The HTCB [25] is a stateful congestion control algorithm designed to improve the AQM stateless schemas' performance. The stateless algorithms features are simple, less processing requirement and, few storage resources need. The stateless algorithm's performance reduced when managing an aggressive flow, so the stateful was suggested to eliminate the lack of managing aggressive flow during congestion control.

The HTCB consists of two parts, a fixed size of hash table "HT" and circular buffer "CB". The HT is a data structure that records abstract information about aggressive flow connections. This information calculates by hashing and will contain the IP address and port number of both peers in addition to flow packets rate and drop rate of aggressive flow. While the CB records information of the new arriving packets such as IP address and port number for both peers. This algorithm's work by inspecting the new arriving packets if it belongs to flow found in HT, then the packet will be dropped passed on probability. In the second phase of this algorithm that if the flow type of arriving packet is not found in HT, so the arriving packet will be compared with a randomly selected packet from CB if its same flow type will register in HT.

4. AQM ALGORITHMS BASED ON QUEUING DELAY

The AQM algorithms based on queue length, such as RED, suffered from difficult tuning parameters for different types of flow services connections (such as TCP-friendly and unresponsive). These services have diverse sending data rate speeds, variance round trip time "RTT" and link types (such as fiber

optic and satellite). Also, the RED manages the queue length, which implicitly affects latency. The based queue length algorithms have a lack to improve the performance of latency-sensitive applications (or services) that have RTT. Therefore, many algorithms have recently been suggested to implement control congestion based on latency or RTT to determine the level of congestion instead of depending on queue length and eliminating the problems in queue based algorithm.

4.1. Closed-loop controller

These controller was used in many research areas such as DC motors [26], [27], Robotics [28], Fuel Cell [29], and others fields. Also in AQM congestion network [30] used the controllers by using control system feedback such as proportional, integral, and/or derivative "PID". The control system model in AQM focused on stabilizing the variation of different application flow and reducing the steady-state error. The control system uses constant gain to adjust the AQM parameter to achieve high performance in controlling congestion.

The proportional integral controller enhanced "PIE" [31] was designed to perform congestion control with feedback control loop connection based on latency. The PIE work uses three basic steps: random dropping, drop probability updating, and latency calculation, as shown in Figure 4. The drop packets, according to probability drop "P_d" calculate by (3), will affect based on latency value, whether it increases or decrease dropping. The PI factors are two constant gain K_p and K_i for Proportional and Integral respectively, used to adjust the optimal calculation of probability drop and reduce error study state "e", where error calculated by the difference between target queue q_t and current queue q. While the latency calculation is done by using one of two ways: either using little's law (current queue delay = queue byte length / dequeue rate) or by using other ways such as record Time-Stamp the packets when inserting in the queue, then use this TimeStamp to obtain latency when this packet leaves the queue. Same as the RED, the PIE drop all packets when it exceeded the target latency. The PIE algorithm's advantages are inherited from the classical PI controller method, which eliminates the steady-state error, autotuning parameter based on congestion level, and achieves stability.

$$P_d = K_p * (qdelay_{current} - qdelay_{ref}) + K_i * (qdelay_{current} - qdelay_{old}) \quad (3)$$

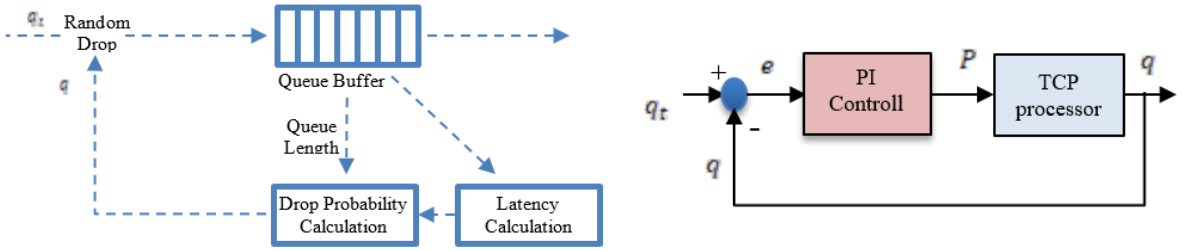

Figure 4. The PIE structure; (a) details view, (b) general view of PI controller

The delay-based PI controller enhanced by adaptive CHOKe "D-PAC" [32], was designed as delay base algorithm to achieve fairness queue among various services flows types, by using Adaptive CHOKe and PI controller. The adaptive CHOKe used to penalize aggressive flows to discipline fairness with slow rate flows, while the PI used to keep the latency acceptable with different flow services and eliminate the steady-state error. As mentioned above about CHOKe mechanism work, selecting one or more packet as a candidate to compare with the new incoming packet if match then discards the candidate and incoming packets. While the PI calculates the latency and probability of drop as mention above in the PIE algorithm.

The PD-AQM [33], [34] controller algorithm was used based on TCP window size and current queue length within time. The value of current queue length *q(t)* change based on the number of applications flow, window size *w(t)* and, RTT (4), where the RTT is measured by both propagation delay and queuing delay. The drop probability p is measured based on current queue q and target q_t value (queue length error). The proportional and derivative gain is used to eliminate the error and also to calculate the drop probability (5), where these gain effect by window size, queuing delay and, link capacity. This algorithm provides high link utilization, faster settling time, and a slight overshoot.

$$q(t) = \frac{no.of\ flow\ connections}{RTT} * w(t) - link\ capacity \quad (4)$$

$$p = K_p(q(t) - q_t) + K_d q(t) + p_0 \quad (5)$$

4.2. CoDel

The controlled delay active queue management "CoDel AQM" [35], same as PIE was used queue delay (latency) to implement congestion control and keep delay in the queue as minimum as possible. The CoDel is parameter-less based on the concept of two keys: avoiding bad queue (adds delay) and enhancing good queue (high bandwidth utilization). The goal of CoDel is high bandwidth utilization with minimal delay. This algorithm implements by calculation the sojourn time (the time that packets have spent in the queue) of a standing queue (the number of packets enqueues). If the sojourn packet's time exceeded the target queuing delay time, then the algorithm starts dropping packets. The target time acts as a threshold that specifies that the maximum queuing delay can be acceptable above the algorithm mark/drop packets and target time measure from interval time. The interval time should be at least an RTT to avoid packet drop misbehavior and not a lot higher than RTT to prevent the unnecessary delay in detection congestion. So the suitable suggested interval time is the maximum RTT among all sharing applications connection.

Fair queue CoDel "FQ-CoDel" [36], [37], is an extension of the CoDel algorithm that provides fairness among different flows services connections. Also, it's considered a hybrid algorithm of packet scheduler and queue management to avoid congestion control. The FQ-CoDel is nearly parameter-less to improve the performance and efficiency of congestion control. This algorithm classifies the incoming packets (the packets flow services distinguished base on IP address and port number of both peers parameters) in multiple different queues, then apply the CoDel on each queue. The FQ-CoDel contains two parts, the scheduler to select the queue for dequeue the packets, and the CoDel to perform the congestion control in the selected queue. The scheduler part gives priority and fairness to low rate services connections.

An adaptive CoDel with interval tuning "ACoDel-IT" and adaptive CoDel with target and interval tuning "ACoDel-TIT" algorithms were proposed for autotuning the parameter to stabilize queuing delay and increase the performance of the network [38]. While the ACoDel-TIT has an additional enhanced performance by reducing the drop packets and high bandwidth utilization. Both algorithms have the same procedure of CoDel for calculating drop probability. The CoDel has fix parameter, while the adaptive algorithm has an autotuning parameter (interval and target time) based on network conditions. In the ACoDel-IT only the interval has been adaptive, while ACoDel-TIT both interval and target time have auto tuning. ACoDel-TIT can more effectively stabilize the queueing delay and provide higher performance in link utilization than the ACoDel-IT.

5. AQM ALGORITHMS BASED ON QUEUING LENGTH AND DELAY

Many services and applications work over the network, which forms heterogeneous flow connections [39]. The IPv4 header has TOS and traffic class in the IPv6 header to differentiate these services. The heterogeneity results from different flow rate as well as the different end to end delay (round trip time "RTT"). For example, the media streaming application (e.g., video) has a higher flow rate than other applications, while the satellite communication has a large RTT than other connections. Due to this variation of flow rate and RTT, the congestion control algorithms based on the delay of a packet are required to improve algorithms' performance based on queue length, such as RED schema.

The enhanced random early detection "ENRED" [40] was used to minimize the queue delay and loss rate of the packet by keeping the average queue size *avg* minimal. The low pass filter was used to make the queue more stable by allowing the algorithm to take action to be meaningful. Reacting faster leads to oscillations and instability while responding more slowly makes the system tardy. The ENRED depends on queue weight based on queue size and burst delay in a queue. This algorithm reduces the *avg* of the queue by using a small queue size, so, this leads to less delay as well as a low loss rate.

Delay-controller random early detection "DcRED" [41] is an extension of the RED algorithm by using the delay feature while preserving the original character such as min_{th} and max_{th}. The modification in DcRED using a delay of a packet in the queue to calculate the drop probability.

An adaptive AQM [42] work based on trade-off between queuing delays and link utilization "TODU". The TODU goal is to achieve low queue delay when different types of services are connected over the network. In TODU, the virtual queue was maintained, which has a capacity smaller than the actual queue. The mark/drop packets in the actual queue are done when the virtual queue overflows, while the arriving packets into the actual queue will update the state of the virtual queue. When the new packet arrives with a size that exceeds the available capacity in a virtual queue, this packet will mark/drop to the actual queue; otherwise, it adds and then updates the virtual queue.

The RED-exponential technique "RED-E" [43] was proposed as a modified RED algorithm. The RED-E uses non leaner behavior to drop packets to manage heterogeneous services flow types. The RED-E used min_{th}, $max_{th,}$ and instantaneous queue length to measure the congestion level. When *avg* exceed the min_{th} the drop probability calculations based on packets arriving rate or RTT by using the exponential formula. The higher arriving rate means that the flow has a higher delay in the queue. In (6) and (7) show the RED-E modified from (1) of RED (remove the dependent on the static value of max_p), where the C is the packets arriving rate that base on the RTT of the flows.

$$initial\ Drop\ probability = \frac{e^{avg} - e^{min_{th}}}{e^{min_{th}} - e^{max_{th}}} \quad (6)$$

$$P_d = \frac{initial\ Drop\ probability}{1 - C \times initial\ Drop\ probability} \quad (7)$$

6. AQM WITH INTELLIGENT OPTIMIZATION

To improve the AQM algorithm schemas, an intelligent procedure was combined with it. Many intelligent aspects were available such as machine learning, neural network, genetic algorithm and fuzzy value. The intelligent method is used to enhance the AQM performance by optimizing the select and tuning parameter during heterogeneous traffic flows to reduce the delay more than traditional AQM.

6.1. AQM with neural network algorithm

The neural network "NN" mimics the human brain-behavior for learning and decision making; recently, NN was used with the AQM algorithm to achieve the queue length stabilize with high link utilization. NN's essential parts are inputs, weights, activation function, and output, where the activation function performs summing calculation while weight is updated to achieve optimization decision. The NN can be classified based on neuron interconnections (such as Feedforward, Feedback, and Recurrent) and weight update learning (e.g., supervised, unsupervised, reinforcement learning). many research studies have been applied the NN with AQM to enhanced congestion control performance and predict future congestion levels.

The feed-forward neural network AQM "FFNN-AQM" [44] was designed to deal with heterogynous traffics flow and to predict the future queue length value. The FFNN-AQM weights update based on time-varying to achieve stabilize queueing length. It adopts self-learning to predict future queue length using an activation function (in this study, activation function was a soft sign). As shown in Figure 5, the feedback control algorithm, as described in the previous section, mapped to the NN controller to calculate the probability drop. There is two input (target queue and current queue length), where the constant gain used as weight and this weight update by using gradient descent. The stability of queue length in this algorithm is achieved by maintaining the current queue length close to the target queue. In this algorithm, in addition to stability, the settling time becomes faster, as well as high utilization of bandwidth and low delay. In [45], the explicitly congestion notification "ECN" was used with Reinforcement learning and the decision making based on inferred rest of path congestion to tuning the algorithm parameter (which is based on markov decision process "MDP"). This algorithm has a set of level congestion sate as well as a set of actions to achieve the target parameter.

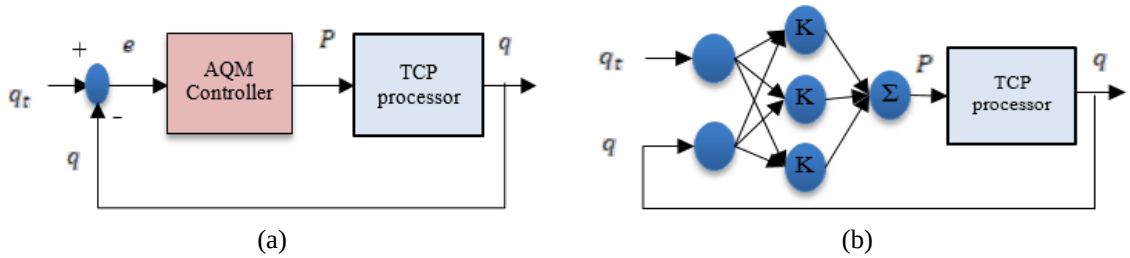

Figure 5. Convert the AQM feedback controller to neural controller; (a) AQM feedback controller, (b) Neural network controller

6.2. AQM with genetic algorithm

A genetic algorithm "GA" technique is used to search optimization from the population (best solution), which follows Darwin's natural evolution theory. The population represents as a chromosome (such as binary encoding), and the operations used in GA are reproduction, selection, crossover and, mutation. many research and studies in network congestion control with GA help, such as AQM with GA.

The genetic algorithm with proportional and integral controller "GA-PI" [46] was designed to optimize the PI controller's solution. As discussed above, the PI controller is closed-loop feedback to eliminate the error study state "e" caused by latency of different flow connections, and this action by using two factors constant gain K_p and K_i. In GA-PI algorithm used GA to optimize the selection of these factors, as shown in Figure 6. The two PI factors represented as binary concatenation, GA operations work by select higher fitness after that the crossover and mutation applied that allow a variety of population (K_p and K_i) to select the best. Also there is many other study in this field such as ant colony optimization "ACO" to optimize tuning PID AQM parameters [47]. While other study used particle swarm optimization (PSO)-based PID (proportional–integral–derivative) "PSOPID" [48]. The PSO used to find best position from global position by using velocity and position. These feature of PSO used to select appropriate parameters of PID's gain to reach zero steady state error in different type of services. The algorithm result show low in rise time and reduced the settling time and this mean low delay and faster settling time.

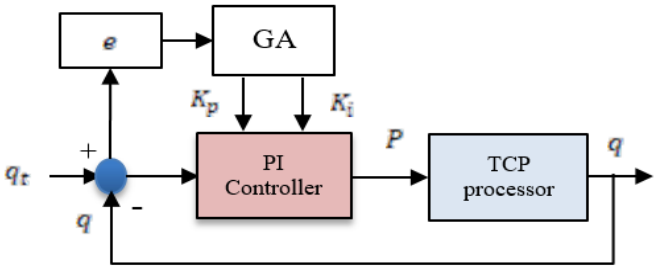

Figure 6. GA-PI congestion controller

6.3. AQM with Fuzzy Logic algorithm

The fuzzy logic "FL" have membership values between 0 and 1 while crisp value has only two value only 0 and 1 or true and false. FL follows the if-else rule that takes the decision based on imprecise and vagueness information. Many works implement the FL with the AQM algorithm to enhance the congestion control parameter and achieve the optimization result. The FL takes a crisp value as input (for example, x value) then transforms it into a fuzzy set by splitting it into different stages as in Table 1 to optimize decisions [49].

Table 1. Fuzzy membership value

Stage	Description
LN	x is Large Negative
MN	x is Medium Negative
SN	x is Small Negative
Z	Zero
SP	x is Small Positive
MP	x is Medium Positive
LP	x is Large Positive

The fuzzy proportional integral derivative (FPID) controller [50], [51] was designed to optimize the PID-AQM algorithm parameters, and these parameters are three constant gain K_p, K_i and K_d. Where the FL adjusted these parameters to achieve optimal control on congestion when a different type of service flows used over a network. The error "e" is the difference between actual and desired queue length, and e_c is the change in flow rate, both e, and e_c used as input to FL block, and the three constant gain as optimizing output of this block as shown in Figure 7.

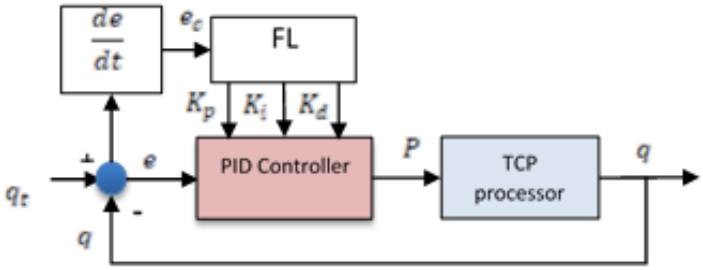

Figure 7. FPID congestion controller

There are many other study that mixed two or more optimization technique to improve optimization of AQM parameter. In [52] fuzzy proportional-Integral (FPI) controller was designed with genetic algorithm (GA) used to tuning the FPI parameters. The FPI can achieve good performance such as the desired queue length, fast response and high link utilization. While the GA used to optimize in select the FPI parameters.

In [53] used FL controller with particle swarm optimization (PSO), social spider optimization (SSO) and ant colony optimization (ACO) algorithms then make comparison to show how optimize PID gains. While in [54] design a linear quadratic (LQ)-servo controller as an AQM and its parameters tuned by using the particle swarm optimization (PSO) method, which provide more stabilization with faster settling time and small delay.

7. COMPARISON BETWEEN THE DISCUSSED ALGORITHMS

This review compare the algorithms discussed above with each other to analyze the improvement performance of these algorithms. In [55] the comparison have been hold based on varying the propagation delay and the link capacity. While in this study, the AQM schemas are briefly discussed and compared based on several metrics, as shown in Table 2. The Low "L", Moderate "M", High "H", and Very High "V.H" have been used to assign the value of a specific algorithm deal with the metric. Fairness means how an algorithm achieves fairness among different flows (TCP-friendly or UDP aggressive flow), while the complexity of tuning parameters of this algorithm. Each algorithm can achieve the Throughput, Link utilization, and Loss rate of packets. Finally, the stability of an algorithm obtained when flow dramatically changed within time.

Table 2. Comparison of AQM algorithms

AQM Method	Algorithm name	Fairness	Complexity	Throughput	Link utilization	Loss Rate	Queue Stability
Queueing Length	RED	L	H	L	M	H	L
	FRED	H	V.H	H	H	L	M
	WRED	L	H	M	M	M	M
	ARED	L	H	M	H	M	H
	GRED	L	H	M	L	M	M
	EAGRED	L	H	H	M	M	M
	DGRED	L	H	H	M	L	H
	SDGRED	L	V.H	H	M	L	V.H
	CHOKe	M	M	M	M	M	M
	CHOKe-RH	H	V.H	H	M	L	H
	HTCB	H	V.H	H	M	M	L
Queuing Delay	PIE	M	M	M	H	L	V.H
	D-PAC	H	H	M	H	L	V.H
	PD-AQM	M	H	M	H	M	H
	CoDel	M	L	M	H	M	L
	FQ-CoDel	V.H	M	H	M	M	L
	ACoDel-IT	M	M	H	H	M	H
	ACoDel-TIT	H	M	H	H	M	H
Queuing Length & Delay	ENRED	M	M	M	M	M	H
	DcRED	M	M	H	M	L	M
	TODU	L	H	H	H	L	H
	RED-E	L	M	L	M	M	H

There is not easy to say which algorithm is best because each algorithm is developed to deal with applications, services, and flow rate. The main classification of AQM schemas was either based on queuing

length, delay, or managing congestion. As mentioned previously the AQM schemas classified into three groups the first one based on queue length such as the RED which is the basic algorithm that used early to manage congestion in router queue. The drop probability increase when the average queue length increased. The RED algorithm has acceptable performance as compared with DT method, but difficult to tuning the value of two thresholds when different type of traffic and applications used over internet. The second group of AQM schemas is based on queuing delay such as CoDel which depend on sojourn time that the packets spend in queue (the queue delay is the timestamp between enqueuer and dequeuer of packet). The PID controller used delay of packets in queue to solve the congestion. The PID same as CoDel but use extra parameters (gain constant) to enhanced the performance. The PID is a widely used controller due to its simple structure, easy implementation, robust nature and the less number of parameters to be tuned. The third schemas of AQM is combine the advantage of two previous schemas by providing stable queue length with different type of traffics. Other AQM algorithms developed with intelligent techniques to achieve the optimal result for tuning PID parameters such as AQM with fuzzy logic, neural network, or genetic algorithms. These algorithms have recently become an interesting field for many researchers to get the best performance in congestion control networks [56].

8. CONCLUSION

This survey lists some AQM algorithms developed to control congestion in a network, and these algorithms are classified based on metrics used to measure the congestion level. Also, the survey discusses each algorithm's work, starting with RED as the first algorithm developed to implement AQM and its development to enhance the performance of a network, and ending with the optimization congestion control based on intelligent techniques to obtain high performance in a network. Finally, a comparison has been made based on several metrics to evaluate each algorithm’s performance and how improved to control congestion in the network.

REFERENCES

[1] W. M. Lafta, A. A. Alkadhmawee and M. A. Altaha, "Best strategy to control data on internet-of-robotic-things in heterogeneous networks," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 2, pp. 1830-1838, April 2021, doi: 10.11591/ijece.v11i2.pp1830-1838.

[2] M. Welzl, "Source behaviour with binary feedback," in *Network Congestion Control Managing Internet Traffic*, pp. 16-18, 2005.

[3] M. Allman, V. Paxson and E. Blanton, "TCP Congestion Control," *Network Working Group*, RFC5681, September 2009.

[4] J. Chu, N. Dukkipati, Y. Cheng and M. Mathis, "Increasing TCP’s Initial Window," *Internet Engineering Task Force (IETF)*, RFC6928, April 2013.

[5] F. Baker and R. Pan, "On Queuing, Marking, and Dropping," *Internet Engineering Task Force (IETF)*, RFC7806, April 2016.

[6] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet," *Network Working Group*, RFC2309, p. 17, April 1998, doi: 10.17487/RFC2309.

[7] F. Baker and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management," *Internet Engineering Task Force (IETF)*, RFC7567, p. 31, July 2015, doi: 10.17487/RFC7567.

[8] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397-413, August 1993, doi: 10.1109/90.251892.

[9] S. Sharma, D. Jindal and R. Agarwal, "Analysing Mobile Random Early Detection for Congestion Control in Mobile Ad-hoc Network," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 8, no. 3, pp. 1305-1314, June 2018, doi: 10.11591/ijece.v8i3.pp1305-1314.

[10] J. N. Hooker, "Chapter 15 - Operations Research Methods in Constraint Programming," in *Foundations of Artificial Intelligence*, vol. 2, P. v. B. T. W. Francesca Rossi, Ed., Elsevier, pp. 527-570, 2006, doi: 10.1016/s1574-6526(06)80019-2.

[11] D. Lin and R. Morris, "Dynamics of Random Early Detection," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 4, p. 127-137, October 1997, doi: 10.1145/263109.263154.

[12] R. Molenaar, "https://networklessons.com," cisco, 2013 - 2021. [Online]. Available: https://networklessons.com/cisco/ccie-routing-switching-written/wred-weighted-random-early-detection#.

[13] CISCO, "Queue Limits and WRED," in *QoS Modular QoS Command-Line Interface Configuration Guide*, Cisco IOS XE 17, 2020.

[14] W.-C. Feng, D. Kandlur, D. Saha and K. Shin, "A self-configuring RED gateway," *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, vol. 3, March 1999, pp. 1320-1328, doi: 10.1109/INFCOM.1999.752150.

[15] N. Hamadneh, M. Al-Kasassbeh, I. Obiedat and M. BaniKhalaf, "Revisiting the Gentle Parameter of the Random Early Detection (RED) for TCP Congestion Control," *Journal of Communications*, vol. 14, no. 3, p. 7, March 2019, doi: 10.12720/jcm.14.3.229-235.
[16] S. Floyd, R. Gummadi and S. Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management," *AT&T Center for Internet Research at ICSI*, p. 12, August 2001.
[17] S. Floyd, "Recommendation on using the gentle variant of RED," *ICSI Networking Group Technical Report*, March 2000.
[18] M. Baklizi, H. A.-. jaber, S. Ramadass and N. A. a. M. Anbar, "Performance Assessment of AGRED, RED and GRED Congestion Control Algorithms," *Information Technology Journal*, vol. 11, pp. 255-261, 2012, doi: 10.3923/itj.2012.255.261.
[19] M. Baklizi and J. Ababneh, "Performance Evaluation of the Proposed Enhanced Adaptive Gentle Random Early Detection Algorithm in Congestion Situations," *International Journal of Current Engineering and Technology*, vol. 6, no. 5, p. 8, October 2016.
[20] M. Baklizi, J. M. Ababneh, M. Abualhaj, N. Abdullah and R. Abdullah, "Markov-modulated bernoulli dynamic gentle random early detection," *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 20, October 2018.
[21] M. Baklizi, "Stabilizing Average Queue Length in Active Queue Management Method," *IJACSA*, vol. 10, no. 3, p. 7, 2019, doi: 10.14569/ijacsa.2019.0100310.
[22] M. Baklizi, "Weight Queue Dynamic Active Queue Management Algorithm," *Symmetry*, vol. 12, no. 12, p. 2077, December 2020, doi: 10.3390/sym12122077.
[23] V. V. Govindaswamy, G. Zaruba and G. Balasekaran, "Analyzing the accuracy of CHOKe hits, CHOKe misses and CHOKe-RED drops," in *2008 Canadian Conference on Electrical and Computer Engineering*, Canada, 2008, doi: 10.1109/CCECE.2008.4564501.
[24] Z. Hussain, G. Abbas and U. Raza, "CHOKe with Recent Drop History," in *13th International Conference on Frontiers of Information Technology (FIT)*, Islamabad, 2015, doi: 10.1109/FIT.2015.37.
[25] S. Hu, J. Sun, Q. Xu and J. Kong, "A Fairness-driven Active Queue Management Algorithm with Hash Table and Circular Buffer," *2020 Chinese Control And Decision Conference (CCDC)*, August 2020, pp. 2502-2506, doi: 10.1109/CCDC49329.2020.9164827.
[26] F. A. Hasan and L. J. Rashad, "Fractional-order PID controller for permanent magnet DC motor based on PSO algorithm," *International Journal of Power Electronics and Drive Systems (IJPEDS)*, vol. 10, no. 4, pp. 1724-1733, December 2019, doi: 10.11591/ijpeds.v10.i4.pp1724-1733.
[27] H. K. Khleaf, A. K. Nahar and A. S. Jabbar, "Intelligent control of DC-DC converter based on PID-neural network," *International Journal of Power Electronics and Drive Systems (IJPEDS)*, vol. 10, no. 4, pp. 2254-2262, December 2019, doi: 10.11591/ijpeds.v10.i4.pp2254-2262.
[28] F. A. Raheem, B. F. Midhat and H. S. Mohammed, "PID and Fuzzy Logic Controller Design for Balancing Robot Stabilization," *IJCCCE*, vol. 18, no. 1, p. 11, April 2018, doi: 10.33103/uot.ijccce.18.1.1.
[29] W. T. J. Al-Rubaye, A. S. Al-Araj and H. A. Dhahad, "Digital PID Control Law Design for Fuel Cell Model Based on FPGA Emulator System," *IJCCCE*, vol. 20, no. 3, July 2020, doi: 10.33103/uot.ijccce.20.3.5.
[30] F. Al-doraiee, S. Al-Qaraawi and H. J. Hassan, "Design and Implementation of Adaptive Wavelet Network PID Controller for AQM in the TCP Network," *IJCCCE*, vol. 13, no. 1, 2013.
[31] R. Pan, P. Natarajan, F. Baker and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem," *Internet Engineering Task Force (IETF)*, RFC8033, February 2017, doi: 10.17487/RFC8033.
[32] S. Hu, J. Sun, Z. Liu and Q. Xu, "A PI Queueing Delay Controller Enhanced by Adaptive CHOKe for AQM," *IEEE Access*, vol. 6, pp. 57219-57229, October 2018, doi: 10.1109/ACCESS.2018.2872952.
[33] S. K. Bisoy and P. K. Pattnaik, "Design of feedback controller for TCP/AQM networks," *Engineering Science and Technology*, vol. 20, no. 1, pp. 116-132, February 2017, doi: 10.1016/j.jestch.2016.10.002.
[34] A. Puerto-Pi˜na and D. Melchor-Aguilar, "Stability Analysis of PD AQM control for delays models of TCP networks," *International Journal of Control*, p. 24, November 2020, doi: 10.1080/00207179.2020.1849804.
[35] K. Nichols, V. Jacobson, A. McGregor and J. Iyengar, "Controlled Delay Active Queue Management," *Internet Engineering Task Force (IETF)*, RFC8289, p. 25, January 2018.
[36] T. Hoeiland-Joergensen, P. McKenney, d. taht, J. Gettys and E. Dumazet, "The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm," *Internet Engineering Task Force (IETF)*, RFC8290, p. 25, January 2018, doi: 10.17487/RFC8290.
[37] F. Zampognaro, "Enabling CoDel AQM with TCP Cubic connections over satellite links," *ISAECT*, pp. 1-6, 2019, doi: 10.1109/ISAECT47714.2019.9069695.
[38] J. Ye and K.-C. Leung, "Adaptive and Stable Delay Control for Combating Bufferbloat: Theory and Algorithms," *IEEE Systems Journal*, vol. 14, no. 1, pp. 1285-1296, March 2020, doi: 10.1109/JSYST.2019.2929157.
[39] D. Papadimitriou, M. Welzl, M. Scharf and B. Briscoe, "Open Research Issues in Internet Congestion Control," *Internet Research Task Force (IRTF)*, RFC6077, p. 51, February 2011, doi: 10.17487/RFC6077.
[40] A. Hamdy, A. El-Sayed, A. El-Sayed, Z. Elsaghir and I. Z. Morsi, "Enhanced Random Early Detection (ENRED)," *International Journal of Computer Applications*, vol. 92, no. 9, April 2014.
[41] A. A. Abu-Shareha, "Controlling Delay at the Router Buffer using Modified Random Early Detection," *IJCNC*, vol. 11, no. 6, p. 14, November 2019, doi: 10.5121/ijcnc.2019.11604.

[42] H. Wang, "Trade-off queuing delay and link utilization for solving bufferbloat," *Science Direct*, vol. 6, no. 4, pp. 269-272, December 2020, doi: 10.1016/j.icte.2020.05.008.
[43] H. Abdel-Jaber, "An Exponential Active Queue Management Method Based on Random Early Detection," *Journal of Computer Networks and Communications*, vol. 2020, p. 11, 2020, doi: 10.1155/2020/8090468.
[44] S. K. Bisoy and P. K. Pattnaik, "An AQM Controller Based on Feed-Forward Neural Networks for Stable Internet," *Arabian Journal for Science and Engineering*, vol. 43, p. 3993-4004, August 2018, doi: 10.1007/s13369-017-2767-9.
[45] C. A. Gomez, X. Wang and A. Shami, "Intelligent Active Queue Management Using Explicit Congestion Notification," *IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1-6, doi: 10.1109/GLOBECOM38437.2019.9013475.
[46] M. Q. Sulttan, M. H. Jaber and S. W. Shneen, "Proportional-integral genetic algorithm controller for stability of TCP network," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 6, pp. 6225-6232, December 2020, doi: 10.11591/ijece.v10i6.pp6225-6232.
[47] H. I. Ali and K. S. Khalid, "H-infinity Based Active Queue Management Design for Congestion Control in Computer Networks," *IJCCCE*, vol. 12, no. 3, p. 9, September 2014.
[48] H. I. Ali and K. S. Khalid, "Swarm intelligence based robust active queue management design for congestion control in TCP network," *IEEJ Trans. Elec. Electron Eng.*, vol. 11, pp. 308-324, February 2016, doi: 10.1002/tee.22220.
[49] H. M. Kadhim and A. A. Oglah, "Congestion Avoidance and Control in Internet Router Based on Fuzzy AQM," *Engineering and Technology Journal*, vol. 39, no. 2, pp. 233-247, 2021, doi: 10.30684/etj.v39i2A.1799.
[50] L. Lin, Y. Shi, J. Chen and S. Ali, "A Novel Fuzzy PID Congestion Control Model Based on Cuckoo Search in WSNs," *Sensors*, vol. 20, no. 7, p. 16, March 2020, doi: 10.3390/s20071862.
[51] M. K. Oudah, M. Q. Sulttan and S. W. Shneen, "Fuzzy type 1 PID controllers design for TCP/AQM wireless networks," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 21, no. 1, pp. 118-127, January 2021, doi: 10.11591/ijeecs.v21.i1.pp118-127.
[52] M. Z. Al-Faiz and A. M. Mahmood, "Fuzzy-Genetic Controller for Congestion Avoidance in Computer Networks," *Iraqi Journal of Computers, Communication, Control & Systems Engineering*, vol. 11, no. 2, pp. 20-27, 2011.
[53] H. M. Kadhim and A. A. Oglah, "Interval type-2 and type-1 Fuzzy Logic Controllers for congestion avoidance in internet routers," *International Conference on Sustainable Engineering Techniques*, 2020, p. 15, doi: 10.1088/1757-899x/881/1/012135.
[54] S. S. Sabry and T. M. Nayl, "Particle Swarm Optimization Based LQ-Servo Controller for Congestion Avoidance," *Iraqi Journal of Computers,Communications,Control and Systems Engineering*, vol. 19, no. 1, p. January, 2019, doi: doi:10.33103/uot.ijccce.19.1.8.
[55] K. Okokpujie, C. Emmanuel, O. Shobayo, E. Noma-Osaghae and I. Okokpujie, "Comparative analysis of the performance of various active queue management techniques to varying wireless network conditions," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 1, pp. 359-368, February 2019, doi: 10.11591/ijece.v9i1.pp359-368.
[56] M. A. Al-majeed and L. Saud, "A Comparative Study among Four Controllers Intended for Congestion Control in Computer Networks," *ASRJETS*, vol. 41, no. 1, pp. 333-355, 2018.