American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)
ISSN (Print) 2313-4410, ISSN (Online) 2313-4402
© Global Society of Scientific Research and Researchers
http://asrjetsjournal.org/

A Comparative Study among Four Controllers Intended for Congestion Control in Computer Networks

Maryam Abd Al-majeed[a]*, Laith Saud[b]

[a,b]*Control and System Engineering Department, University of Technology, Baghdad, Iraq*
[a]*Email: Maryam_85_1993@yahoo.com*
[b]*Email: Laithjasim15@yahoo.com*

Abstract

Computer networks efficiency is an vital part of today's information services technology, with this comes multiple issues, among them is the congestion problem. This paper will discuss the designing and evaluating of four controllers to deal with this issue. The design starts with modeling the Transmission Control Protocol /Active Queue Management (TCP/AQM) which is intended for dynamics modeling of the average TCP window size and the queue size in the bottleneck router. Apart from modeling, the work comprises of two parts. In the first, three controllers Random Early Detection, Proportional-Integral and Proportional-Integral-Derivative (RED, PI, and PID) are designed, tested, evaluated, and compared among each other, with the use of the TCP/AQM model developed. The second part considers designing a fuzzy logic based online tuned PID controller and comparing its performance with a PID controller tuned offline with three tuning methods, Ziegler Nichols (Z-N), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO). The Integral Square Error (ISE) is used as the objective function for optimization. The controllers' performance is evaluated using the following parameters for system's response, rise time, settling time, and maximum peak overshoot. The performance of the controllers is also examined by applying a disturbance as an exceptional condition. To test and evaluate the controllers, the system as all is implemented using MatLab (Version 2014). The results obtained indicated that the PID gave a better performance, compared to the RED and the PI, in following changes in the desired queue level, and in reducing the loss of packets. The PID gave a settling time 20% lesser than that of the PI and 60% lesser than that of the RED. Regarding the tuning methods, and under the settings considered for each in this work, the ACO-PID gave the least overshoot (1.545%) compared to the others methods [ZN-PID (40%), PSO-PID (13.85%), Fuzzy-PID (5%)].

----

* Corresponding author.

The PSO and ACO managed to cause great reduction in settling time (t_s) and rise time (t_r). The ratios of t_s and t_r of PSO-PID to PID before tuning are (16.5%), (23.43%) and the ratio of t_s and t_r of ACO-PID to PID before tuning are (11.5%), (44.56%). The intelligent tuning methods [PSO & ACO] gave better t_r and t_s compared to Fuzzy or Ziegler–Nichols. Despite the indicated relative performance of the Fuzzy PID controller, it has some important privileges. Firstly, it is an online tuning method, as it continuously adapts the PID controllers' parameters as long as the system is running. Secondly, its performance can still be improved by optimizing the fuzzy part. Thirdly, it represents a nonlinear controller (as its parameters are changing), and so it can even suit the nonlinear model.

Keywords: Random Early Detection (RED); Proportional-Integral (PI); Proportional-Integral-Derivative (PID); Fuzzy Logic (FL); Ziegler Nichols Method (ZN); Particle Swarm Optimization (PSO); and Ant Colony Optimization (ACO).

1. Introduction

The vital role computers networks play in modern technology is enormously increasing on a global scale, and so does the bandwidth size. Hence, networks efficiency, which depends on many factors and issues, is an important concern to be considered. One of the issues is to avoid congestion [1]. During congestion, large amounts of packets experience delay or can even be dropped, and the occurrence of several congestion problems can cause degradation of the throughput and large packet loss rate. The resulting congestion will decrease efficiency and reliability of the whole network and at very high traffic, performance will completely fall and may cause no packets to be delivered [2].

Avoiding congestion and guaranteeing the reception of transmitted data and in minimum time as possible is an important requirement. However due to the massive amount of data transmitted, it is difficult to achieve this requirement perfectly. As a compromise, best solutions are sought out by researchers instead of perfect solutions to avoid data loss and unacceptable delays.

The TCP, which is a protocol used for computers communication over the Internet, has a mechanism which avoids congestion in computer networks. TCP detects congestion by checking acknowledgements or time-out processing and adjusts TCP window sizes of senders. But this control method has a shortcoming that it used to avoid congestion after congestion appears on the computer networks. AQM schemes have been proposed to complement the TCP role for network congestion control [3]. The most common AQM objectives are: efficient queue utilization (to minimize the occurrences of queue overflow and underflow, thus reducing packet loss and maximizing link utilization), queuing delay (to minimize the time required for a data packet to be serviced by the routing queue), and robustness (to maintain closed-loop performance in spite of changing conditions) [4].

Different efforts have been done so far related to active queue management. Misra and his colleagues [5] developed a non-linear mathematic model to help in obtaining the expected transient behavior of networks with AQM routers supporting TCP flows. In reality, with this model, the congestion control problem turned into a control problem of a plant with the possibility of using the rich control theory so far developed. Later, Hollot and his colleagues [6] approximated this model to a linear model using small-signal linearization about an operating point. The linearization was intended to simplify the problem, in a justified way, and gain the benefit of using feedback control theory for linear systems.

After developing the TCP/AQM system model, a significant part of efforts was devoted to design controllers to deal with the congestion problem in computer networks. Different controllers and design methods have been used for this purpose including using different tuning methods to tune the controllers' parameters for best performance and based on different performance indices and optimization objectives.

Hollot and his colleagues [7] proposed a PI controller as a congestion control scheme based on linear control theory. As a model, they adopted the previously developed linearized model of TCP and AQM. For congestion avoidance and control in intermediate nodes, Waskasi and his colleagues [8] designed a PI controller with its parameters being dynamically adapted, using an ANN (Artificial Neural Network), to compensate for changes in the system. Seeking performance improvement over that of the classical PI controller, Al-Faiz and his colleagues [9] suggested a Fuzzy PI controller for congestion avoidance in computer networks. Their work starts with manual tuning of controller parameters before a Genetic Algorithm (GA) is used to optimize them. Alvarez [10] tried a methodology to design a PID controller with linear gain scheduling that allows a network with wireless links to deal with control congestion under a variety of configurations. Salim and his colleagues [11] used the "MATLAB/Nonlinear Control Design Blockset (NCD)" to tune the of the (PI) controller they used for congestion avoidance. Besides introducing the conventional H_∞ controller as an AQM scheme, Ali and his colleagues [12] used the classical PID controller with two ways of optimization, PSO and ACO.

This paper consists of two efforts. The first one presents three controllers to control congestion and track the desired queue level. The controllers are RED, PI, and PID. The effort includes testing and evaluating the controllers with a comparison among their performance. The design starts with modeling the TCP/AQM and then designing and testing the controllers. The second effort includes designing a fuzzy logic based online tuned PID controller and comparing its performance with a PID controller tuned offline with three tuning methods, the classical (Z-N) tuning method, and the intelligent optimization methods (PSO) and (ACO). The objective function used in the PSO and ACO is the Integral Square Error (ISE). Evaluation of the controllers' performance is based on the following performance indices: rise time (t_r), settling time (t_r), and maximum peak over shoot (M_p).

The rest of this paper is organized as follows: Section 2 covers developing the TCP/AQM model and the control objectives. The RED, PI, and PID controllers design is given in Section 3. Section 4 is devoted for the design of the fuzzy logic based online tuned PID controller, and the algorithms used for offline tuning of the PID controller. Section 5 presents the simulation results for the controllers' performance and tuning algorithms. Finally, brief conclusions are provided in Section 6.

2. Mathematical Model of the TCP/AQM

Different approaches have been used to model the TCP like, renewal theory, fixed point, fluid models, processor sharing, and control theoretic [13]. Here, the model developed by Hollot and his colleagues [6] is adopted. This model represents a simplified model of the one developed by Misra and his colleagues [5] which is a nonlinear dynamic model and is based basically on fluid-flow and stochastic differential equations. The simplified model ignores the TCP timeout mechanism.

The non-linear differential equations for the model are given as:

$$\dot{W}(t) = \frac{1}{R(t)} - \frac{W(t)W(t-R(t))}{2R(t-R(t))}p(t-R(t)) \qquad (1)$$

$$\dot{q}(t) = \frac{W(t)}{R(t)}N(t) - C \qquad (2)$$

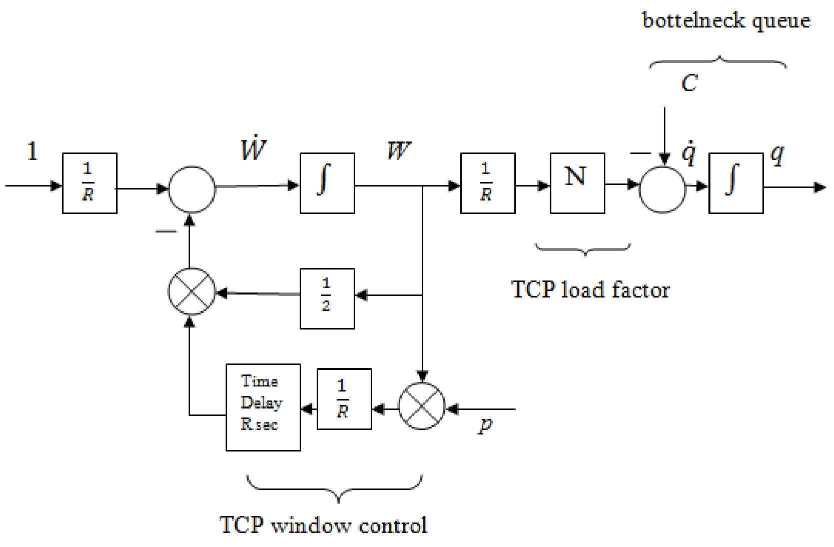

Figure 1: A block-diagram of the TCP's congestion-avoidance flow-control mode.

Where $\dot{W}(t)$ an $\dot{q}(t)$ denotes the time-derivative of $W(t)$ and $q(t)$ respectively.

W = average TCP window size (packets);

q = average queue length (packets);

R(t) = round trip time;

C = link capacity (packets/sec.);

N = load factor (Number of TCP session);

p = probability of packet drop/mark;

Figure (1) shows these differential equations in the block diagram which highlights TCP window-control and queue dynamics.

In order to linearize the model, it is assumed that the number of TCP sessions and the link capacity are constant, and then an approximated linearized model can be developed by small-signal linearization about an operating point (W_o, p_o, q_o). The linearized model is given by equation (3):

$$\begin{aligned} \delta \dot{W}(t) &= -\frac{2N}{R_0^2 C}\delta W(t) - \frac{R_0 C^2}{2N^2}\delta p(t-R_0) \\ \delta \dot{q}(t) &= \frac{N}{R_0}\delta W(t) - \frac{1}{R_0}\delta q(t) \end{aligned} \qquad (3)$$

Taking the Laplace transform of equation (3) and after rearrangement, the following transfer functions can be obtained:

$$P_{tcp}(s) = \frac{W(s)}{p(s)} = \frac{\frac{R_0 C^2}{2N^2}}{s + \frac{2N}{R_0^2 C}} \qquad (4)$$

$$P_{queue}(s) = \frac{q(s)}{W(s)} = \frac{\frac{N}{R_0}}{s + \frac{1}{R_0}} \qquad (5)$$

The block diagram of the linearized AQM control system is shown in Figure (2), where, P_{tcp} (s) is the transfer function of TCP, P_{queue} (s) is the transfer function of the router queue, and C(s) denotes the transfer function of the controller.

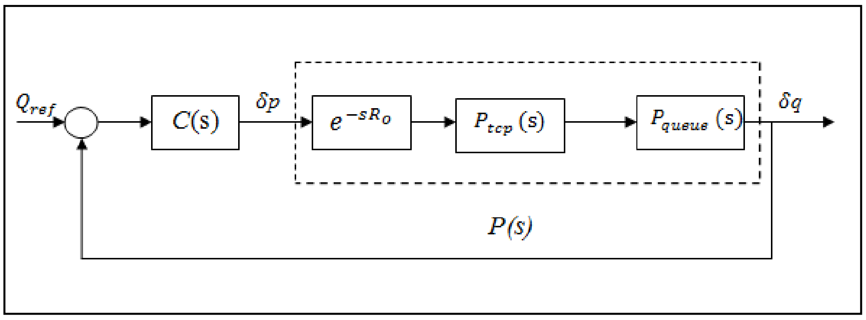

Figure 2: Block diagram of a linearized AQM model-based feedback control.

The overall plant transfer function becomes:

$$P(s) = P_{tcp}(s)P_{queue}(s)e^{-sR_o} \qquad (6)$$

which can be expressed as:

$$P(s) = \frac{W(s)}{p(s)} = \frac{\frac{C^2}{2N}}{(s+\frac{2N}{R_0^2 C})(s+\frac{1}{R_0})} \times \exp(\text{-s}R_0) \quad (7)$$

The model in equation (7) is a general one, and for the purposes of testing the controllers' performance and simulation work, a specific model for the TCP/AQM is determined from this equation by assuming the following values for the network parameters: N = 60, C = 3750 packet/sec, and R_0 = 0.246 sec, and the resulting model is:

$$P(s) = \frac{117187.5}{(s^2 + 4.594S + 2.13)} \times \exp(\text{-}0.246\text{s}) \quad (8)$$

3. The RED, PI, and PID controllers

The following section, the RED, PI, and PID controllers will be introduced briefly.

3.1 Random Early Detection Controller [14]

The Random Early Detection (RED) scheme was initially described and analyzed by Floyd and Jacobson in 1993. The main idea behind this algorithm is that it starts dropping packets randomly before the buffer gets full, the transfer function of RED is:

$$C_{red}(s) = \frac{L_{red}}{{}^{s}/_{K}+1} \quad (9)$$

Where

$$L_{red} = \frac{p_{max}}{max_{th} - min_{th}} \quad (10)$$

$$K = \frac{log_e(1-\alpha)}{\delta} \quad (11)$$

min_{th} : The minimum threshold, which the queue length must exceed before any dropping or marking is done.

max_{th} : The maximum threshold, if the queue length size is exceeded, all the incoming packets are dropped.

$p_{\ max}$: The maximum marking/dropping probability, which determines how aggressively the queue marks or drops packets when congestion occurs.

$\alpha > 0$ is the queue averaging parameter.

δ is the sample time.

e in log_e is a constant equal to 2.71828.

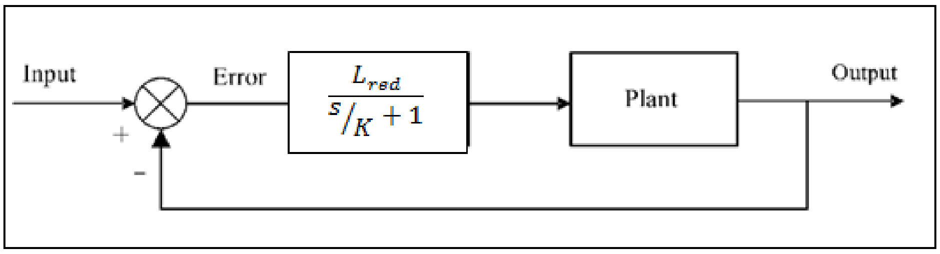

Figure 3: The block diagram of a RED controller based feedback system.

Figure (3) shows RED block diagram in the system, where the input represents the reference value of the queue length (Qref), and the output represents the actual value of the queue length. The plant represents the TCP/AQM model.

3.2 PI Controller

The PI controller is described by the following expression **[7]**:

$$C_{PI}(s) = k_p + \frac{k_i}{s} \tag{12}$$

Where Kp and *Ki* are the proportional and the integral gain coefficients. A block diagram for a feedback PI based controller system is shown in Figure (4).

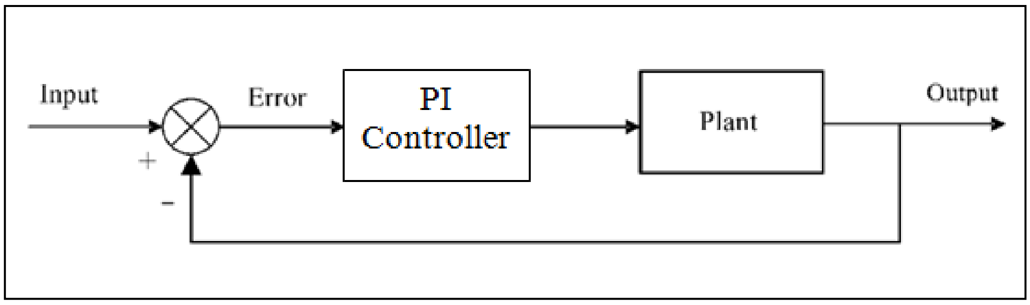

Figure 4: Block diagram of a feedback PI based controller system.

3.3 PID controller

The PID controller is a widely used controller due to its simple structure, easy implementation, robust nature and the less number of parameters to be tuned. The PID controller is expressed by [15]:

$$C(s) = k_p + \frac{k_i}{s} + k_d s \tag{13}$$

Or

$$C(s) = k\left(1 + \frac{1}{Tis} + Tds\right) \tag{14}$$

Where k_p, k_i, and k_d are the proportional gain, the integral gain, and the derivative gain respectively. The proportional gain makes the controller respond to the error while the integral gain helps to eliminate steady state error and derivative gain to prevent overshoot. The block diagram of controller and model can be shown in figure (5).

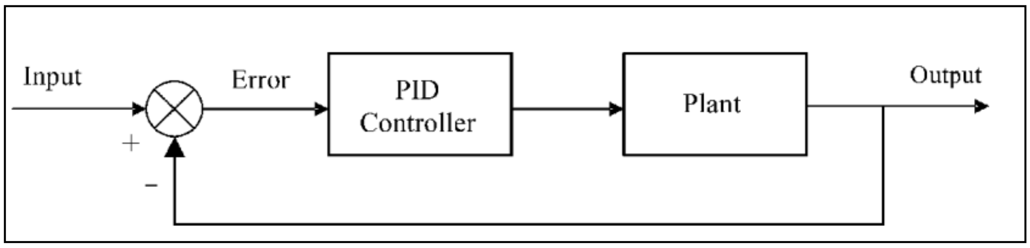

Figure 5: Block diagram of a feedback PID based controller system.

4. Tuning the PID controller

The goal of tuning is to determine optimum values for the PID controller parameters that meet closed loop system performance specifications, and the robust performance of the control loop over a wide range of operating conditions should also be ensured. Four tuning methods are used here starting with a conventional tuning method (Ziegler-Nichols), followed by two modern methods, the Particle Swarm Optimization (PSO-PID), and the Ant Colony Optimization (ACO-PID). All of these three methods are used offline. The fourth method represents a fuzzy logic based online tuning method for the PID, which as a result constitutes a fuzzy PID controller.

4.1 Ziegler-Nichols (Z-N) tuning method [16]

In 1942 Ziegler and Nichols, described a simple mathematical procedure for tuning the PID controller. There are two types of Z-N method, one is open loop system based, and the other is closed loop system based. Here, the second type is used. The Z-N closed-loop tuning method requires the determination of the ultimate gain and the ultimate period.

Kcr = Ultimate gain (minimum gain with P-only control that causes system to cycle continuously).

Pcr = Ultimate period of oscillation.

The procedure of tuning is as follows (using only proportional feedback control):

1. Reduce the integrator and derivative gains to 0.

2. Increase Kp from 0 to some critical value Kp=Kcr at which sustained oscillations occur.

3. Note the value Kcr and the corresponding period of sustained oscillation, Pcr.

Then, the controller gains are specified according to table 1.

Table 1: Ziegler Nichols Recipe for closed loop.

PID Type	K_p	T_i	T_d
P	0.5 K_{cr}	∞	0
PI	0.45 K_{cr}	$\frac{P_{cr}}{1.2}$	0
PID	0.6 K_{cr}	$\frac{P_{cr}}{2}$	$\frac{P_{cr}}{8}$

4.2Tuning of the PID controller with PSO [17]

Particle Swarm Optimization (PSO) is one of the optimization techniques and a kind of evolutionary computation technique. The technique is derived from research on swarm such as bird flocking and fish schooling. In the PSO algorithm, instead of using evolutionary operators such as mutation and crossover to manipulate algorithms, for m-variable optimization problem, a flock of particles are put into the m-dimensional search space with randomly chosen velocities and positions knowing their best values.

The position $X_{i,m}$ of the particle and the previous best position for the particle is represented as $Pbest_{i,m}$. The best performing particle among the swarm population is represented as $gbest_{i,m}$ and the velocity of each particle in each m-dimension is represented as $V_{i,m}$. The new velocity and position for each particle can be calculated from its current velocity and distance, respectively. The velocity and position of each particle can be calculated as shown in the following equations:

$$V_{i,m}^{(t+1)} = W.V_{i,m}^{(t)} + c_1 * rand() * (Pbest_{i,m} - x_{i,m}^{(t)}) + c_2 * rand() * (gbest_m - x_{i,m}^{(t)}) \quad (15)$$

$$x_{i,m}^{(t+1)} = x_{i,m}^{(t)} + v_{i,m}^{(t+1)} \quad (16)$$

Where

i	-number of particles in the group.
m	-dimension.

t	-pointer of iterations (generations).
$V_{i,m}^{(t+1)}$	-velocity of particle at iteration.
W	-inertia weight factor.
c_1, c_2	-acceleration constant.
$rand\ (n)$	-random number between 0 and 1.
$x_{i,m}^{(t)}$	-current position of particle at iterations.
$Pbest_{i,m}$	-best previous position of the particle.
$gbest_m$	-best particle among all the particles in the population.

The following parameters have been used for tuning the PID controller using the PSO:

1) The parameters of each individual in the PSO algorithm are k_p, k_i, and k_d.

2) Swarm size equal to 10.

3) Inertia weight factor =1.5.

4) C_1=2 and C_2=2.

5) Iteration is set to 10.

6) Using ISE fitness function.

The ISE (Integral Squared Error) fitness function equation is :

$$ISE = \int_0^\infty e^2(t)dt \qquad (17)$$

4.3 Tuning of controller with ACO [17]

ACO (Ant colony optimization) algorithm is based on the behavior of ants. When an ant travels in a path, it leaves a substance called pheromone. The other ant which follow will take the path with the most pheromones traces present. This path is again marked with their own pheromones. The pheromone gets evaporated over time. The probability (P_{ij}^A) of choosing a node *j* at node *i* is defined in the equation (18). At each generation of the algorithm, the ant constructs a complete solution using equation (18), starting at source node.

$$P_{ij}^{A}(t)=\frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{i,j\in T^{A}}[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}]^{\beta}} \qquad (18)$$

Where:

$\eta_{ij} = \frac{1}{K_j}$, j =[P,I,D] representing heuristic functions.

$\tau_{ij}(t)$: pheromone matrix.

α : constants that determine the relative influence of the pheromone values.

β : constant that determine the heuristic values on the decision of the ant.

T^{A} : is the path effectuated by the ant at a given time.

The quantity of pheromone $\Delta\tau_{ij}^{A}$ on each path may be defined as:

$$\Delta\tau_{ij}^{A} = \begin{cases} \frac{L_{min}}{L_A} & if\ i,j \in T^{A} \\ 0 \end{cases} \qquad (19)$$

When all ants complete one iteration, the pheromone trail is updated using the following equation (20):

$$\tau_{ij}(t)=p\tau_{ij}(t-1)+\sum_{A=1}^{NA}\Delta\tau_{ij}^{A}(t) \qquad (20)$$

The pheromone updating rule was meant to simulate the change in the amount of pheromone due to both the addition of new pheromone deposited by ants on the visited edges and to pheromone evaporation, where:

NA : number of ants.

P : the evaporation rate (a coefficient of persistence of the trial during a cycle such that (1-U) represents the evaporation of the trail between two generations).

The following ACO parameters have been used for tuning the PID controller parameters:

1) The PID parameters in the ACO algorithm are k_p, k_i, and k_d.

2) Number of ants is 25.

3) The Evaporation parameter =0.5.

4) Number of iterations is set to 20.

5) Using ISE as fitness function (Given in equation (17)).

4.4The Fuzzy PID tuning method [18]

Fuzzy rules represent a main part in the fuzzy tuning system and can be evaluated from the human experience and knowledge about the system. They are set, in tuning case, to adjust the PID controller parameters for achieving the best system response. In engineering application, fuzzy logic has the following characters: 1) Fuzzy logic is flexible; 2) Fuzzy logic is based on natural language, and the requirement for intensive reading of data is not very high; 3) Fuzzy logic can take full advantage of expert information; 4) Fuzzy logic is easy to combine with traditional control technique. The Fuzzy-PID controller based system is shown in figure (6).

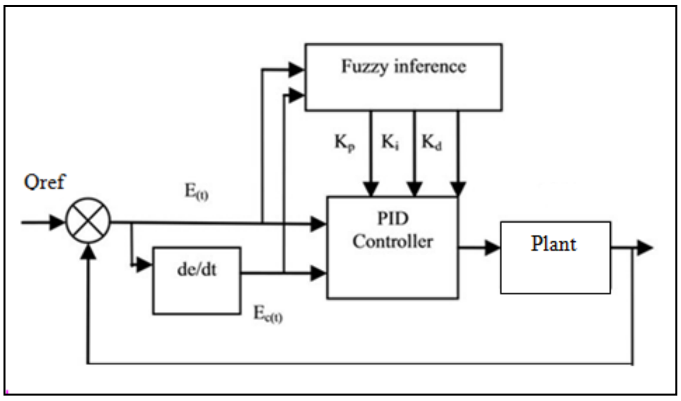

Figure 6: The structure of a fuzzy PID controller based feedback control system.

Choosing the suitable memberships is an issue itself, and many things need to be decided about it among which is their number and shape. Here seven membership are selected to represent the input and output.

The Universe of Discourse (UoD) of each input and output control variable is decomposed into seven fuzzy sets that have the linguistic values: Negative Big (NB), Negative Medium (NM), Negative Small (NS), Zero (Z), Positive Small (PS), Positive Medium (PM), Positive Big (PB).

The ranges chosen for the fuzzy unit In/Out signals are given in figures (7,8,9,10).

For the input variables E and E_c the range is [−2, 2]. The range for the output K_p is [0 , 2.5], The range for the output K_i is [0 , 5], and the range for the output K_i is [0 , 30].

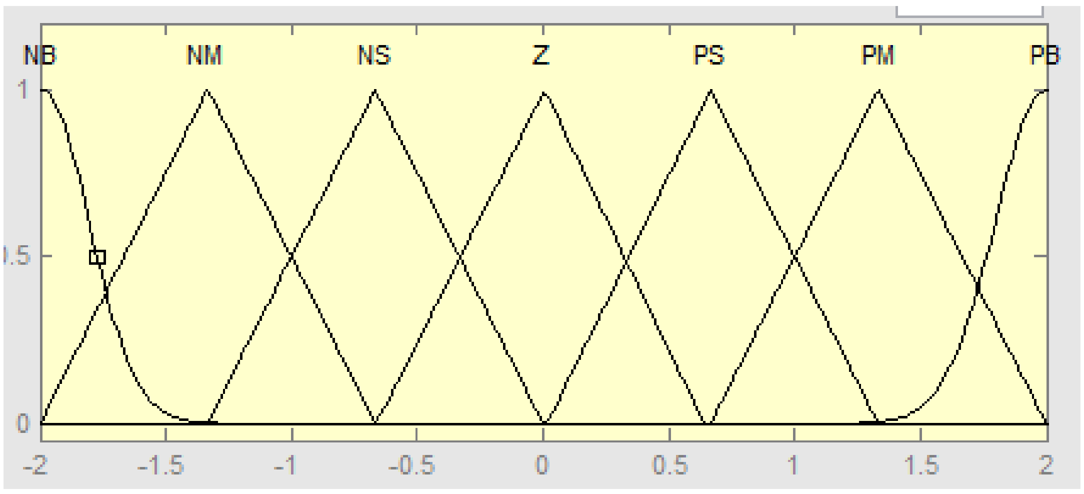

Figure 7: E, E_c membership function curves.

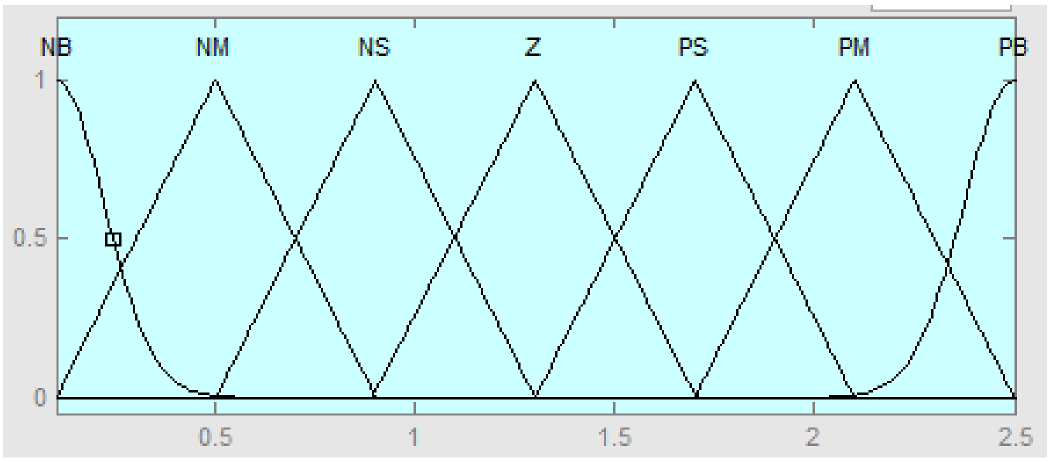

Figure 8: Kp membership function curves.

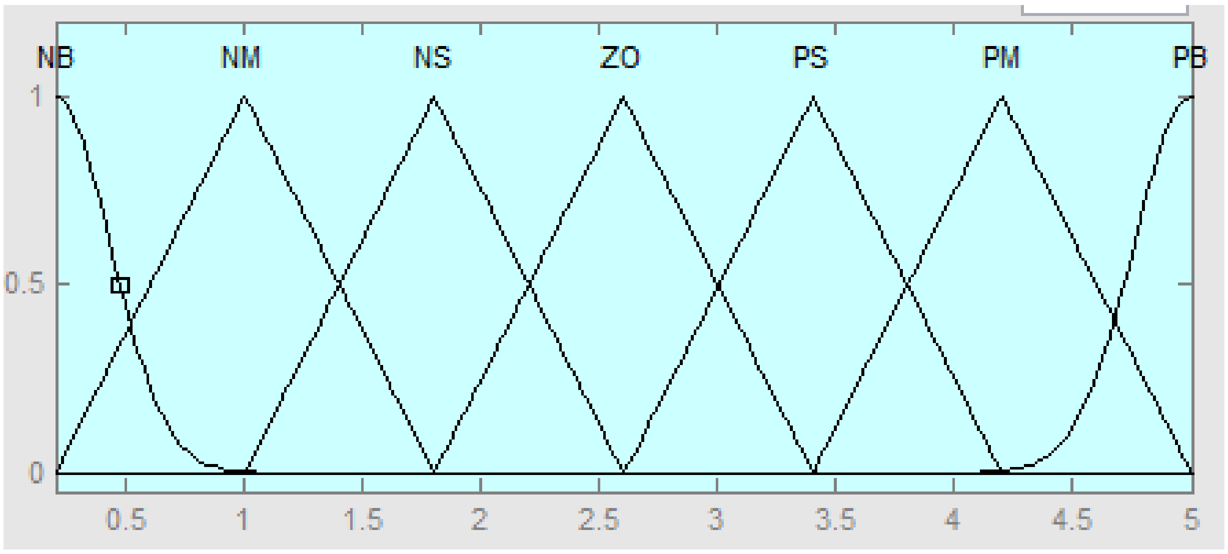

Figure 9: Ki membership function curves.

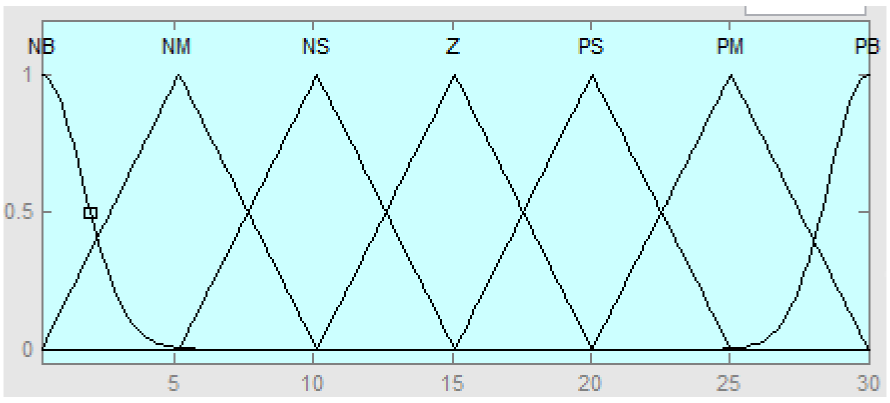

Figure 10: Kd membership function curves.

After defining the membership functions and the range for each input and output signal, we define the rule base. Because we used two inputs each contains seven membership functions, we needed 49 rules as shown in tables (2), (3) and (4).

Table 2: K_p Fuzzy control rules.

Ec \ E	NB	NM	NS	Z	PS	PM	PB
NB	PB	PB	PM	PM	PS	Z	Z
NM	PB	PB	PM	PS	PS	Z	NS
NS	PM	PM	PM	PS	Z	NS	NS
Z	PM	PM	PS	Z	NS	NM	NM
PS	PS	PS	Z	NS	NS	NM	NM
PB	PS	Z	NS	NM	NM	NM	NB
PB	Z	Z	NM	NM	NM	NB	NB

Table 3: K_i Fuzzy control rules.

E Ec	NB	NM	NS	Z	PS	PM	PB
NB	NB	NB	NM	NM	NS	Z	Z
NM	NB	NB	NM	NS	NS	Z	Z
NS	NB	NM	NS	NS	Z	PS	PS
Z	NM	NM	NS	Z	PS	PM	PM
PS	NM	NS	Z	PS	PS	PM	PB
PM	Z	Z	PS	PS	PM	PB	PB
PB	Z	Z	PS	PM	PM	PB	PB

Table 4: K_d Fuzzy control rules.

E Ec	NB	NM	NS	Z	PS	PM	PB
NB	PS	NS	NB	NB	NB	NM	PS
NM	PS	NS	NB	NM	NM	NS	Z
NS	Z	NS	NM	NM	NS	NS	Z
Z	Z	NS	NS	NS	NS	NS	Z
PS	Z	Z	Z	Z	Z	Z	Z
PM	PB	NS	PS	PS	PS	PS	PB
PB	PB	PM	PM	PS	PS	PS	PB

5. Simulation and Results

Test 1: The first simulation was done to test the performance of the three controllers (RED, PI, and PID) in the system when the input (desired queue) equals to 200 packets. Figures (11, 12, and13) show the responses of the three controllers, which show good response for the controllers in tracking the desired queue level.

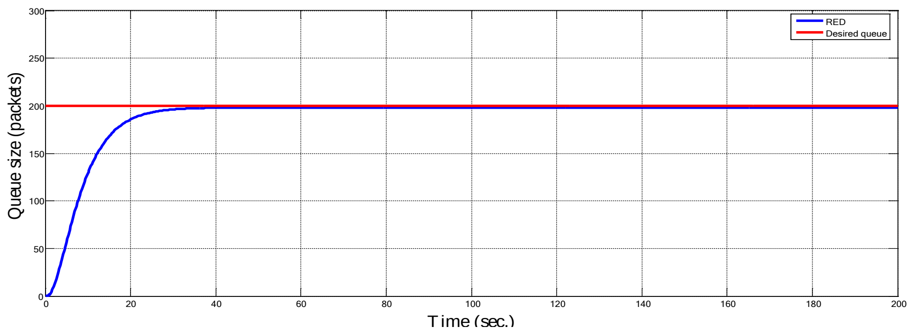

Figure 11: System response when using RED.

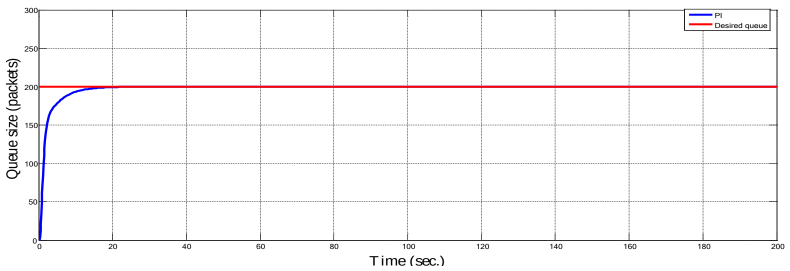

Figure 12: System response with PI controller.

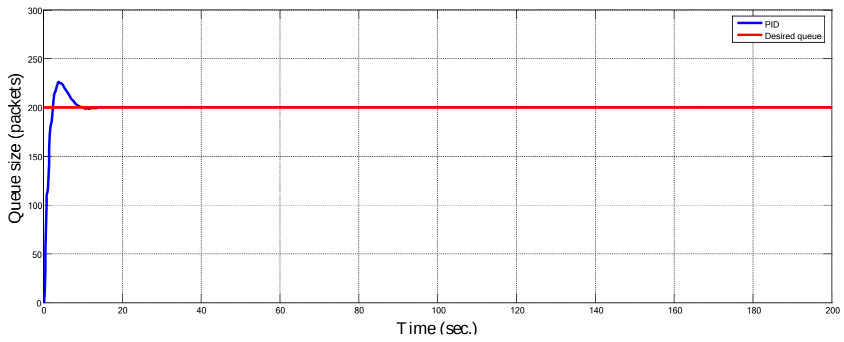

Figure 13: System response with PID controller.

Test 2: The performance of the controllers is tested when the desired queue change every 50 seconds as in equation (21). The response of these controllers are shown in figures (14, 15, and 16). The results obtained indicates that the PID controller gave the best performance, compared to RED and PI, in tracking the changing queue level and decreasing the lost packets.

$$q_{ref} = \begin{cases} 300 & 0 < t < 50; \\ 200 & 50 < t < 100; \\ 500 & 100 < t < 150; \\ 200 & 150 < t < 200; \end{cases}$$

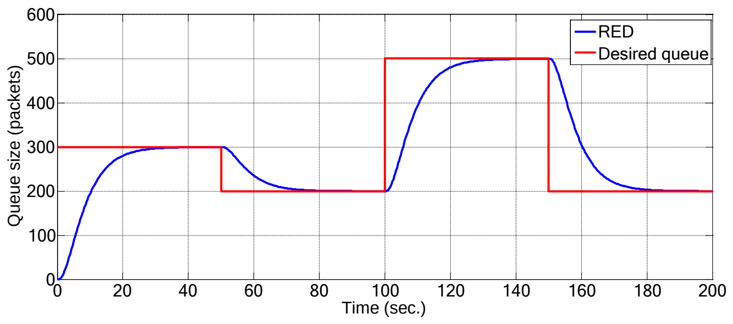

Figure 14: System response with RED.

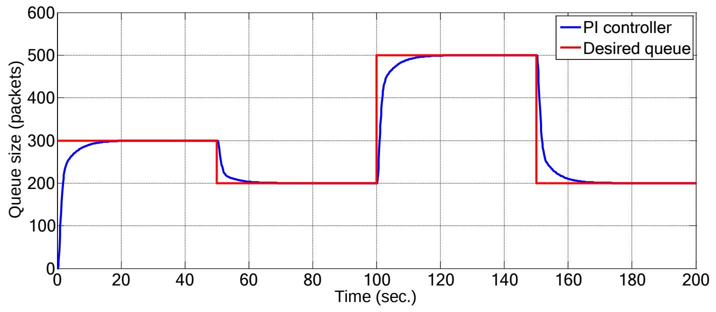

Figure 15: System response with the PI controller.

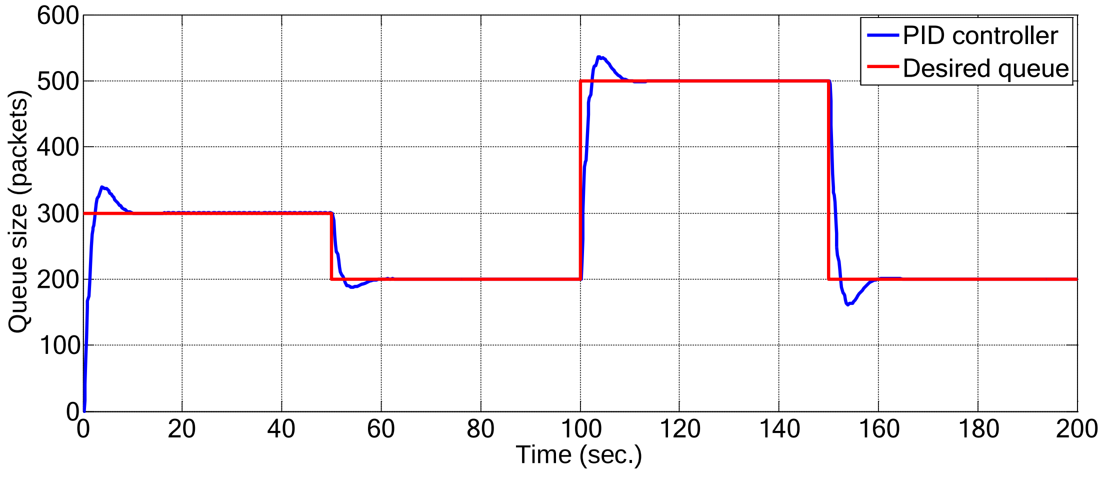

Figure 16: System response with the PID controller.

Test 3: To get an even better performance with the PID controller, it is tuned using the ZN, PSO, and ACO methods. The responses obtained after tuning are shown in figures (17, 18, and 19).

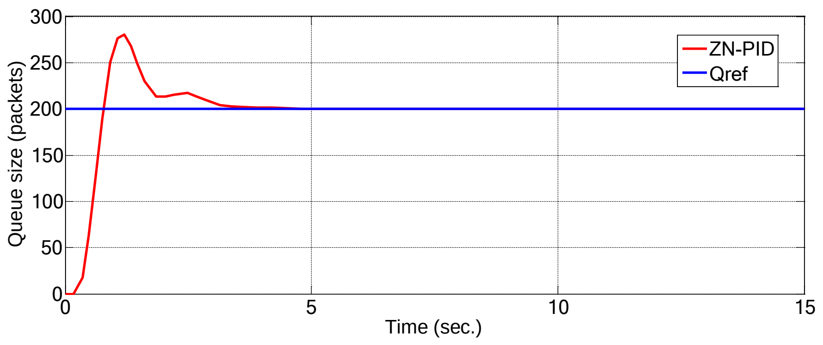

Figure 17: System response with ZN- tuned PID controller.

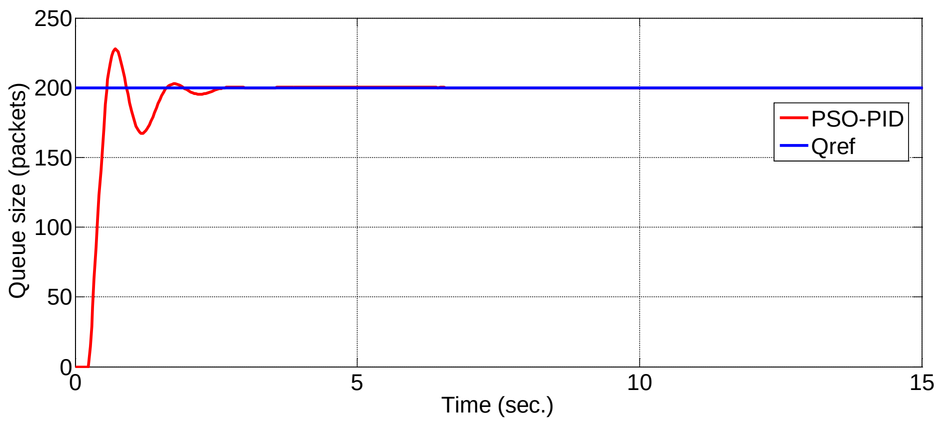

Figure 18: System response with PSO-PID controller

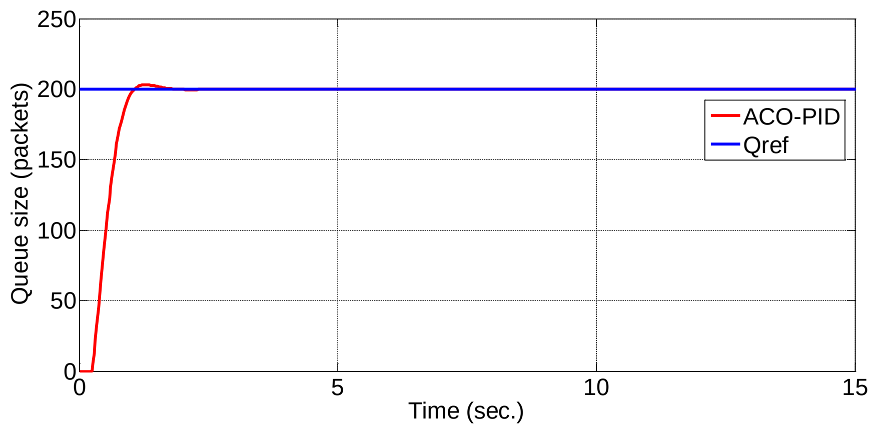

Figure 19: System response with ACO-PID controller

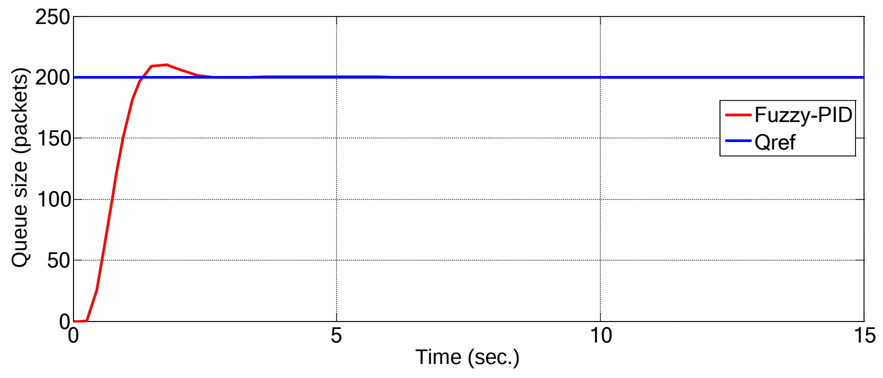

Figure 20: System response with Fuzzy-PID controller

A comparison among the PID tuning methods showed that the tuning method (ZN) led to a large overshoot (40%) compared to the other tuning methods, on the contrary of ACO that gave the least overshoot (1.545%). The PSO and Fuzzy methods gave (13.85%) and (5%) overshot, respectively. These observations are summarized in table (5).

Table 5: Comparison result of different tuning methods.

Methods	Settling time (Ts)	Rise time (Tr)	Peak overshoot Mp
PID (trial and error tuning)	16	2.39	26.5
ZN-PID	6	0.77	80
Fuzzy-PID	2.65	1.32	10
PSO-PID	2.64	0.56	27.7
ACO-PID	1.84	1.065	3.09

Test 4: To examine the performance of the system in an exceptional condition, a disturbance is added as a step equals to 50 which starts at second 50 and stays for the whole remaining response time. Figure (21) shows the block diagram of the system with the disturbance, and figure (22) shows the response of the system error as well as the disturbance signal.

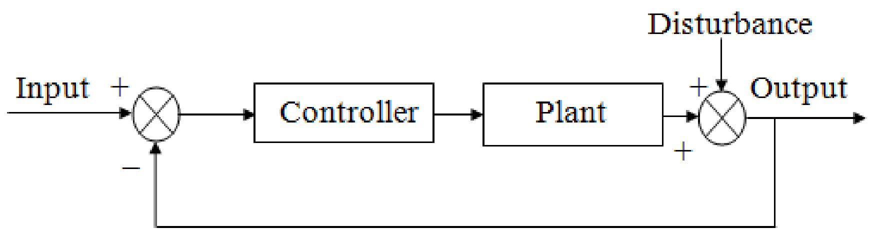

Figure 21: The block diagram of the system with the disturbance.

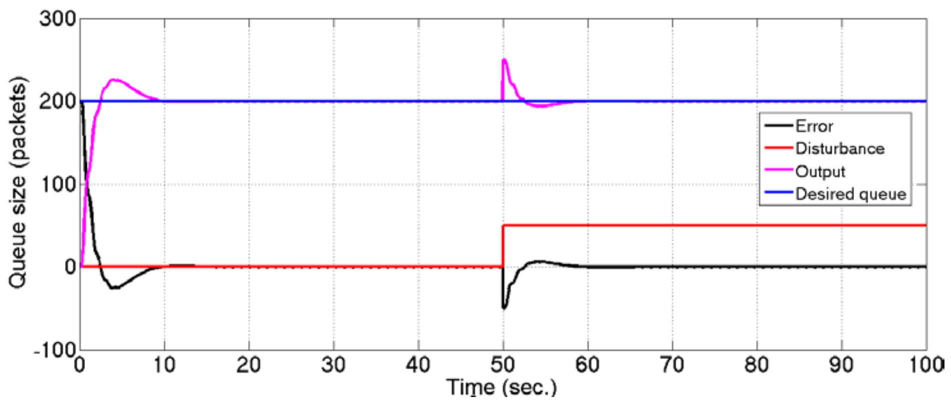

Figure 22: The responses for the desired queue, the output, the error, and the disturbance.

Figures (23) to (29) show the response of the system when a disturbance is added at time equal to 50 seconds. The tests are carried out with different controllers to check their performance.

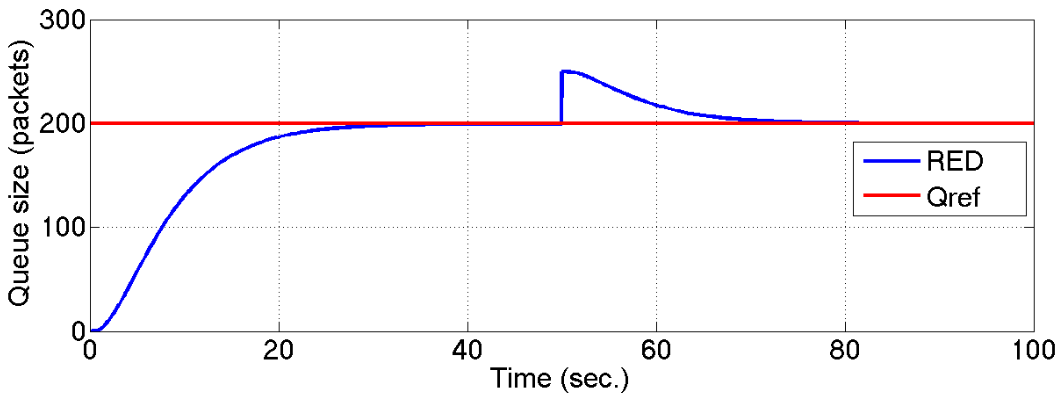

Figure 23: The system response with RED controller in case of disturbance.

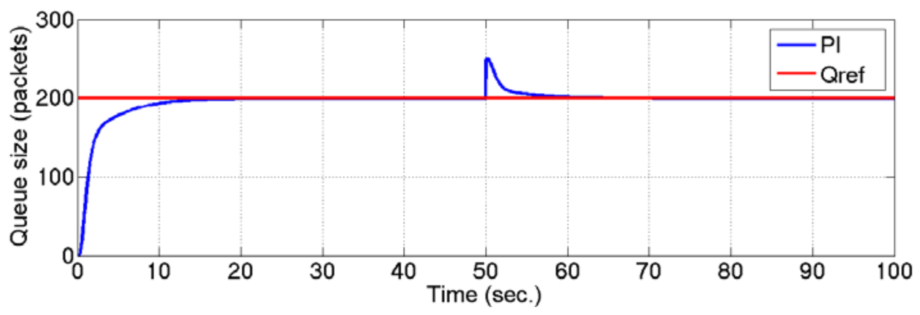

Figure 24: The system response with PI controller in case of disturbance.

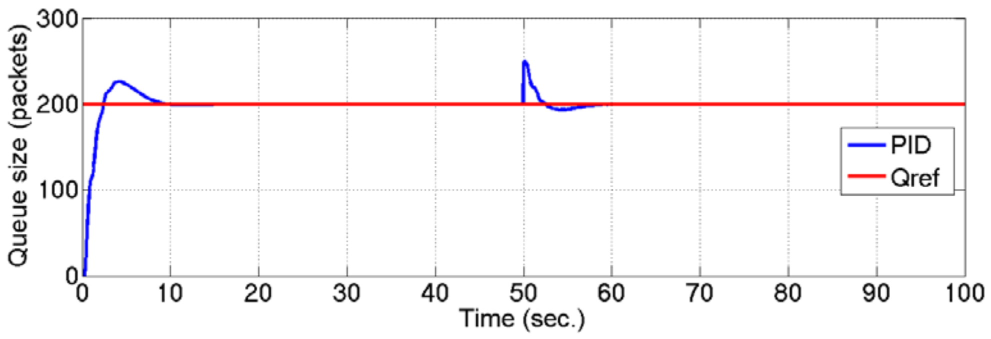

Figure 25: The system response with PID controller in case of disturbance.

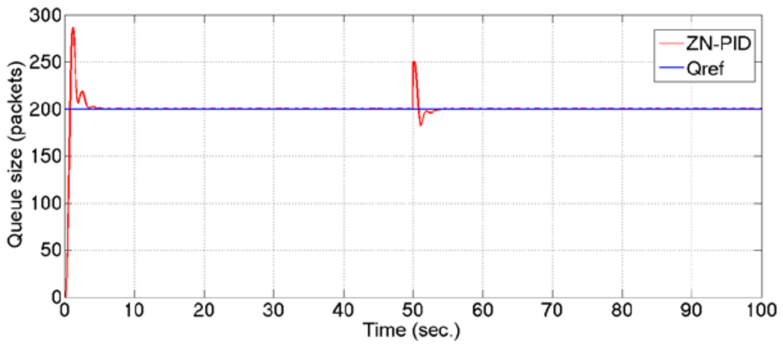

Figure 26: The system response with ZN-PID controller in case of disturbance.

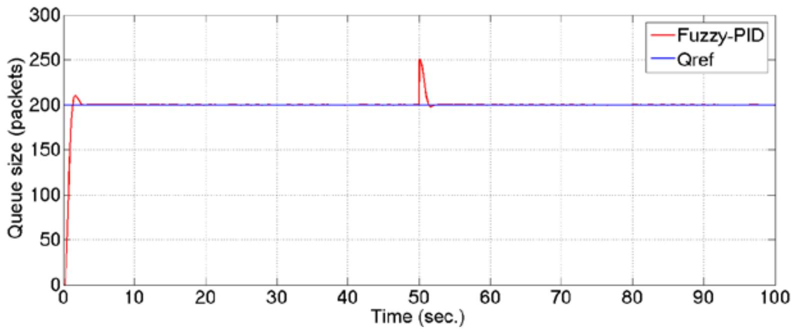

Figure 27: The system response with FPID controller in case of disturbance.

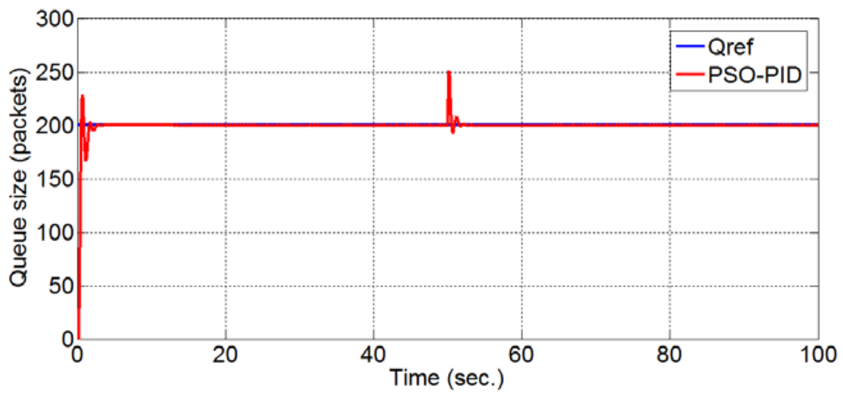

Figure 28: The system response with PSO-PID controller in case of disturbance.

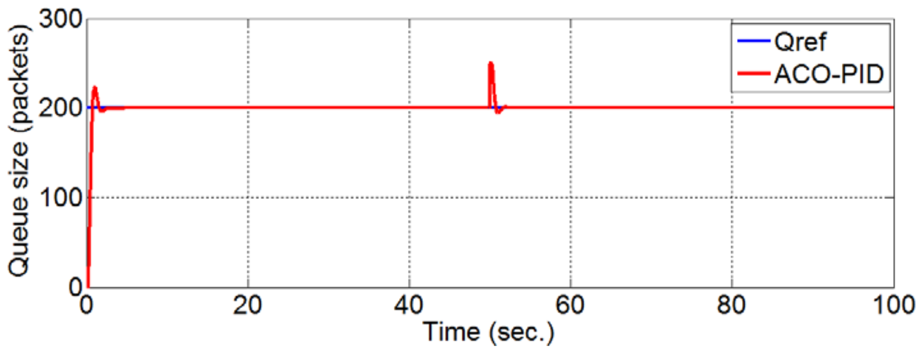

Figure 29: The system response with ACO-PID controller in case of disturbance.

Table (6) shows the time that each controller takes for settling down the response and tracking the queue reference again when the disturbance take place. These figures and the table show that when a disturbance occurs, RED takes the longest time to settle down the response again to track the queue reference with time (t_s= 35 sec.), and the Fuzzy-PID gives the smallest settling time compared to others (t_s= 2.4 sec.).

Table 6: The settling time after disturbance for each controller.

Controller	The time to settle down the disturbance.
RED	35 sec.
PI	20 sec.
PID	10 sec.
ZN-PID	4.5 sec.
Fuzzy-PID	2.4 sec.
PSO-PID	3.5 sec.
ACO-PID	2.55 sec.

6. Conclusion

Based on the design and simulation results, the followings are concluded:

- The PID has a better performance, than the RED and the PI, in following the changes in the desired queue level and in reducing the loss of packets. The PID gave a settling time 20% lesser than that of the PI and 60% lesser than that of the RED.
- A comparison between the three methods used to tune the PID controller, and under the settings considered in this work for each method, indicates that:

- The ACO-PID gave the least overshoot (1.545%) compared to the others methods [ZN-PID (40%), PSO-PID (13.85%), Fuzzy-PID (5%)].

- PSO-PID gave the least settling time and rise time compared to all other tuning methods.

- The PSO and ACO managed to cause great reduction in settling time and rise time. The ratios of t_s and t_r of PSO-PID to PID before tuning are (16.5%), (23.43%). And the ratio of t_s and t_r of ACO-PID to PID before tuning are (11.5%), (44.56%).

- The intelligent tuning methods [PSO & ACO] gave better t_r and t_s compared to Fuzzy or Ziegler–Nichols.

- The comparison between controllers when adding disturbance showed that the Fuzzy-PID gives the smallest settling time compared to others (t_s= 2.4 sec.).

- Despite the relative performance of the Fuzzy PID controller indicated above, it is very important to state the following points:

- A very main positive feature of the Fuzzy tuning method is that it is an online tuning method, as it continuously adapts the PID controllers' parameters as long as the system is running.

- The Fuzzy PID performance can still be improved by optimizing the Fuzzy part.

- In reality, the Fuzzy-PID controller represents a nonlinear controller (as its parameters are changing), and so it can even suit the nonlinear model.

References

[1] V. Jacobson and M. J. Karels. "Congestion Avoidance and Control." in Proc.ACM SIGCOMM, 1988, pp. 314-329.

[2] G. F. Ahammed and R. Banu. "Analyzing the Performance of Active Queue Management Algorithms." in (IJCNC) International Journal of Computer Networks & Communications, Vol.2, No.2, March 2010.

[3] T. Azuma, T. Fujita, and M. Fujita, "Congestion control for TCP/AQM networks using state predictive

control." EEE Transactions on Electronics, Information and Systems, Vol. 125, No.9, pp. 1491–1496, 2005.

[4] C. V. Hollot, V. Misra, D. Towsley and W. Gong, "Analysis and Design of Controllers for AQM Routers Supporting TCP flows." IEEE Transactions on Automatic Control, Vol. 47, No. 6, pp. 945-959, 2002.

[5] V. Misra, W. B. Gong and D. Towsley, " Fluid-Based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED." in Proc. ACM Sigcomm, 2000, pp. 151-160.

[6] C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, " A Control Theoretic Analysis of RED." in Proceedings of IEEE INFOCOM, 2001.

[7] C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, " On Designing Improved Controllers for AQM Routers Supporting TCP Flows." in Proceedings of IEEE INFOCOM, 2001, pp. 1726–1734.

[8] M. Y. Waskasi, M. J. Yazdanpanah and N. Yazdani, " a New Active Queue Management Algorithm Based On Neural Networks PI." Control and Intelligent Processing Center of Excellence Electrical and Computer Engineering Department, University of Tehran, 2005

[9] M. Z. Al-Faiz and A. M. Mahmood, " Fuzzy-Genetic Controller for Congestion Avoidance in Computer Networks." IJCCCE, Vol. 11, No. 2, 2011.

[10] T. Alvarez, " Design of PID Controllers for TCP/AQM Wireless Networks." in Proc. the World Congress on Engineering, 2012, pp. 1273-1280.

[11] S. T. Salim and A. M. Mahmood, " Design of On-Line Tuned Controller for Congestion Avoidance in Computer Networks." Eng. &Tech. Journal, Vol. 31, No. 3, 2013.

[12] H. I. Ali, and K. S. Khalid, " Swarm intelligence based robust active queue management design for congestion control in TCP network." IEEJ Transactions on Electrical and Electronic Engineering, Vol. 11, pp. 308-324, 2016.

[13] J. Olsen, " Stochastic modeling and simulation of the TCP protocol." Uppsala Dissertations in Mathematics, Uppsala University, Sweden, 2003.

[14] Ch. Koutsimanis, and P. G. Park, " Active Queue Management – A router based control mechanism." Scientific report, 2006.

[15] V. Chopra, S. K. Singla, and L. Dewan, "Comparative Analysis of Tuning a PID Controller using Intelligent Methods," Acta Polytechnica Hungarica, Vol. 11, No. 8, 236-249, 2014

[16] B. R. Copeland, "The Design of PID Controllers using Ziegler Nichols Tuning," Internet: http://educypedia.karadimov.info/library/Ziegler_Nichols.pdf, Mar. 2008[Apr. 20017].

[17] B. Nagaraj and N. Murugananth, "A comparative study of PID controller tuning using GA, EP, PSO and ACO," Journal of Automation, Mobile Robotics & Intelligent Systems, Vol. 5, No. 2, 2011.

[18] J. Jin and H. Huang, "Study on fuzzy self-adaptive PID control system of biomass boiler drum water," Journal of bio-energy systems, Vol. 3, No. 1, 2013.