Asian Journal of Research in Computer Science

Volume 17, Issue 2, Page 37-50, 2024; Article no.AJRCOS.111554
ISSN: 2581-8260

Packets' Congestion Management with Fuzzy Admission Control Policy for Differentiated Service Networks

Adio T. Akinwale [a], Adesina S. Sodiya [a], Simeon A. Akinleye [b] and Adegbuyi D. Gbadebo [c*]

[a] Department of Computer Science, Federal University of Agriculture, Abeokuta, Nigeria.
[b] Department of Mathematics, Federal University of Agriculture, Abeokuta, Nigeria.
[c] Department of Computer Science, Lagos State University of Education, Lagos, Nigeria.

Authors' contributions

This work was carried out in collaboration among all authors. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/AJRCOS/2024/v17i2418

Open Peer Review History:
This journal follows the Advanced Open Peer Review policy. Identity of the Reviewers, Editor(s) and additional Reviewers, peer review comments, different versions of the manuscript, comments of the editors, etc are available here: https://www.sdiarticle5.com/review-history/111554

Original Research Article

Received: 08/11/2023
Accepted: 12/01/2024
Published: 16/01/2024

ABSTRACT

This study presents a priority-based admission control system for ensuring stability in a differentiated service network using a fuzzified admission policy. Packets of varying sizes of data were transmitted from *N* sources through a traffic conditioner which categorizes the packets into two independent sources, consequently classifying them into "high" and "low" priorities. While class *A* packets are not denied admission into the buffer, this is not the case with packets of class *B.* Arrivals from both sources follow a Poisson distribution process of $\alpha(k) = (\lambda^k / k!)e^{-\lambda}$. It is assumed that $r > h/\mu$ in order to avoid a situation in which a class B arrival is denied admission while the system is empty. The arrival rates $\lambda_1 \in [0, \mu)$ and $\lambda_2 \in [0, \alpha)$ serve as fuzzy inputs with four linguistic values while the output is a decision, *d.* Simulation started with an initial state, zero and system

**Corresponding author: E-mail: gbadeboad@lasued.edu.ng;*

performance for the first 300time units was monitored. Results indicate that the admission controller admits arriving class B packets provided that the value of y ≤ 4 while it denied admission to arriving class *B* packets when y > 4, thus giving a threshold policy of $^{y=4}$. Arrivals denied admission are dropped and transmitted to the "tree manager" via the bottleneck router. These packets are arranged as nodes in an AVL tree structure which adopts tree properties to manage and transmit nodes to the buffer based on node rotations.

Keywords: Tree structure; node; transmission; tree manager; network traffic.

1. INTRODUCTION

A differentiated service (DiffServ) is a class of service model used to specify and control Internet Protocol (IP) network traffic by class. The goal is to ensure that certain types of traffic that require relatively uninterrupted flow of data get precedence over other kinds of traffic. It is one of the most advanced network solutions for managing network traffic as it aims at classifying traffic into multiple classes and treating them differently. Such classification and differentiation are especially useful when there is a shortage of networking resources [1].

DiffServ is a computer networking architecture that specifies a mechanism for classifying and managing network traffic and providing quality of service on modern IP networks. It could, for example, be used to provide low-latency to critical network traffic such as voice or streaming media while providing best-effort service to non-critical services such as web traffic or file transfers [2].

Quality of service is the ability to provide different priorities to different applications, users or data flows in order to guarantee a certain level of performance such as a required bit rate, delay or delay variation, packet loss or bit error rates. It is important for real-time streaming of multimedia applications such as voice over IP, multiplayer online games, etc since these services often require fixed bit rate and are delay-sensitive [2]. In essence, the quality of service is especially important in networks with limited resources such as in cellular data communication.

When too many packets are transmitted through a network, congestion may occur. At very high traffic, performance may collapse, resulting in non-delivery of packets thereby making the bursty nature of network traffic one of the root causes of congestion. Other causes of network congestion may include ill-configuration of networks and slow routers [3]. This can be corrected either by open-loop or closed-loop methods. In the open-loop method, congestion is prevented by carefully having a good design of the network parameters and structures. With the closed-loop method, the system is monitored to detect congestion, pass this information to where action can be taken and adjust system operation to correct the problem. This is because packets' dropping wastes network resources used for carrying the packet from its source to the router experiencing congestion.

The most common "drop strategy" used for differentiating queue mechanisms is "drop-arrivals", which is otherwise known as "drop-tail" [3]. With this approach, packets are dropped only as they arrive since already queued packets are never dropped. When offering loss-rate differentiation, the drop arrivals strategy could delay packets' dropping. With drop-arrivals, the router waits until packets tagged with higher drop precedence levels arrive and are consequently dropped. Delayed drops can result in complex loss patterns and more jitter compared to dropping packets immediately at congestion [4]. When drops are delayed, the transmission control protocol (TCP) reduces its sending rate later than if drops were made immediately when congestion first occurred. Moreover, TCP increases its sending rate until packet loss is detected exponentially at slow-start and linearly at congestion avoidance.

According to [5], advancement in telecommunication technology is now being directed at adapting traffic patterns to network congestion control. As a result, congestion control methods are modeled such that transmission rates increase linearly when there are no congestion signals. In essence, when congestion is detected, transmission rate decreases by a multiplicative factor. This is the case of TCP in the Internet as congestion is detected at the source, through signals such as packet losses or some negative acknowledgment

mechanisms [6]. However, continuous increase in the rate of data transmission over networks has made it necessary to ensure that network congestions are addressed as they occur.

While TCP congestion control mechanism is used to prevent congestion collapse, Active Queue Management (AQM) schemes have been proposed to complement the TCP network congestion control [7]. In AQM schemes, performance of two thresholds over single threshold is reported. Two thresholds can always be adjusted to give a lower delay for the same throughput. The AQM scheme used with priority structures is able to provide better quality of service, reduce traffic congestion and packet delays [8].

A study on single-server finite capacity queueing model under the <*p*−*F*> policy was investigated by [9]. Results from the study established a steady-state analytical formulae for various performance indices. A finite state-dependent queueing system with admission control *F*-policy and general retrial attempt was proposed by [10]. The researchers used a recursive method to find system performance indices and derived the cost function. In a similar research, [11] studied Zadeh's extension principle as well as fuzzy Markov chains, thereby generating a general solution for queueing systems in a fuzzy environment.

An analysis of a queueing model for priority classes using triangular fuzzy numbers and fuzzy set theory was studied by [12]. In this, the authors investigated the queueing model for *F*-policy using fuzzy parameters. The researchers used a mathematical programming approach to derive the membership function of the performance measures. An investigation into the admission control for customers using *F*-policy in a queueing system was studied by [13] which involves mathematical modeling and computation of real-time problems as it relate to queue network. This was done by using the Markovian admission control policy at startup.

The management problem of queueing system with *F*-policy and a startup time to solve the problem of admission in a queueing system was studied by Zhang et al. [14]. The researchers used the supplementary variable technique and recursive method for obtaining the steady-state probability distribution of the number of customers in the system. A cost function was also constructed to find the best management admission control *F*-policy at the best price.

2. LITERATURE REVIEW

This section is discussed under two sub-headings: related studies and priority scheduling.

2.1 Related Studies

Studies on congestion control in queue networks are enormous in literature. In a study on methods of avoiding queue overflow and reducing queue delay at evolved-NodeB (eNodeB) in Long-Term Evolution networks using congestion feedback mechanism, [15] suggested the use of controlled delay in which the channel rate is changed so that the channel queue capacity can be adapted based on data weight. Result showed that the proposed algorithm outperformed Random Early Detection (RED) gateway. Similarly, Miao [16] proposed an improvement of service quality using queue method in Internet topology. The study adopted the First-In, First-Out (FIFO), Random RED and Per-Connection Queue (PCQ) methods. Results indicated that RED performed better compared to PCQ and FIFO when several users were simultaneously downloading data in the queue.

The performance of two adaptive TCP models were compared with RED and fixed-parameter Proportional Integral (PI) on a MATLAB platform by Okokpujie [17]. Results indicated that the two adaptive TCP models performed better than the fixed-parameter PI and RED controllers. In a similar study, an Adaptive Queue Management algorithm with Random Dropping (AQMRD) was proposed by Patel et al. [18]. It incorporates information about the average queue size and its rate of change to threshold level that falls in-between the minimum and maximum thresholds. Results indicated that the AQMRD was able to minimize packets' dropping by managing the difference between minimum and maximum thresholds.

While several researches had shown that RED can improve TCP performance under certain parameter settings and network circumstances, yet the basic RED algorithm is subject to several inadequacies including sensitivity to its control parameters, bandwidth unfairness and low throughput [19]. To overcome some of these

problems, some researchers had proposed several variants of RED while others suggested making modifications to it. RED with Reconfigurable Maximum Dropping Probability which aims at average queue size reduction and delay time scheduling without any effect on the rate of packet dropping and link utilization via simulation using OPNET was proposed by Al-Allaf and Jabbar [20]. Results indicated that the RRMDP has a more controllable average queue size, which led to lower queuing delay without affecting the link utilization and throughput. In addition, the controllable average queue size keeps the router queue away from buffer overrun, even in the case of severe congestion.

While there are enormous studies on the application of series of methods to solve packets' admission problem in queue networks, to the best of our knowledge, there is none which had addressed the problem by the assignment of priority to individual packets while also adopting fuzzy logic approach to process the crisp variables so that admission control policy is based on output. In addition, packets of lower priorities which are dropped are managed in such a way that it is re-transmitted as soon as the buffer can accommodate it. In essence, this study proposes a congestion management model in which packets with lower service priorities are dropped and are arranged in a tree structure and can be re-transmitted when the server becomes idle. Packets dropped are arranged as nodes in an Adelson-Velsky and Landis (AVL) tree structure. These packets are rotated from time to time based on tree properties and are re-transmitted based on rotations once the buffer is ready to admit them. With this approach, packets dropped are not lost but rather arranged based on sizes in the tree structure. This method ensures that high-priority packets have minimal delay, if any, in the process of admission into the buffer at any point in time.

2.2. Priority Scheduling

Multimedia applications such as video streaming could be affected by delay of individual packets. Since the transmitted content (video frame or voice samples) need to play continuously on the receiver side, a delay in only a fraction of the packets could result in a stall or errors in the video or garbling of the received speech signal [21]. This had led to the development of mechanisms and architectures such as DiffServ which enhances the ability of network operators to provide different classes of service to different types of network flows [22]. Consequently, it is possible for an Internet Service Provider to treat packets that it could identify as belonging to a multimedia traffic flows preferentially by giving such a higher priority in packet queues or guaranteeing a minimum delay and flow throughput.

Priority scheduling processes a packet with higher priority before all packets with a lower one whenever there is one in queue. This can be pre-emptive by interrupting the current processing of a low-priority packet that is currently being processed even if it has a lower priority than another packet arriving during this service time. In order to model this system, the average waiting time for a queue system with two priorities is derived and then providing the solution for the general case [23]. If Dst_i is the service time distribution of packet priority class *i* while λ_i is the arrival rate of packets of this class, then $\rho_i = \lambda_i.E[\mathrm{Dst}_i]$, with $\rho = \sum_i \rho i < 1$ is necessary for the system to remain stable.

In order to distinguish the condition in which an arriving packet is of high priority (priority class 1) or of a low priority (priority class 2), the average waiting time of a priority class 1 packet is:

$$E[D_{q1}] = E[N_{q1}].E[D_{s1}] + E[D_r]$$

$$E[D_{q1}] = \lambda_1.E[D_{q1}].E[D_{s1}] + E[D_r]$$

$$E[D_{q1}] = \rho_1.E[D_{q1}] + \lambda/2\, E[D_s^2]$$

$$E[D_{q1}] = \frac{\lambda E[D_s^2]}{2(1-\rho_1)} \quad (1)$$

In this case $E[D_r]$ is the residual service time for the packet being served upon arrival. Since this service is not preemptive, it may be a packet of any priority. This can be expressed as $E[D_r]$ in terms of the general service time distribution. However, the number of packets in the buffer that need to be processed before the arriving packet is the number of packets $E[N_{q1}]$ of the high priority class, since it takes precedence over the lower priority packets. The high-priority packets only need $E[D_{s1}]$ as the average service

time. Consequently, an arriving packet of low priority will wait until:

a. the server complete service with the packet currently being served;
b. the packets of both low and high priority found in the queue upon arrival have been served; and
c. The packets of high priority that arrived during the waiting period have been served.

The average waiting time of the low priority packets, according to [24] can be expressed as:

$$E[D_{q2}] = E[N_{q1}].E[D_{s1}] + E[N_{q2}].E[D_2] + \lambda_1.E[D_{q2}].E[D_{s1}] + E[D_r]$$

This can be simplified as:

$$E[D_{q2}] = \lambda_1.E[D_{q1}].E[E_{s1}] + \lambda_2.E[D_{q2}].E[D_{s2}] + \lambda_1.E[D_{q2}].E[D_{s1}] + \frac{\lambda}{2}E[D_s^2]$$

$$E[D_{q2}] = \rho_1 E[D_{q1}] + \rho_2 E[D_{q2}] + \rho_1 E[D_{q2}] + \frac{\lambda}{2}E[D_s^2] \quad (2)$$

Further simplification of (2) gives (3):

$$E[D_{q2}](1-\rho_2-\rho_1) = \rho_1 E[D_{q1}] + \frac{\lambda}{2}E[D_s^2]$$

$$E[D_{q2}] = \frac{\rho_1 E[D_{q1}] + \frac{\lambda}{2}E[D_s^2]}{(1-\rho_2-\rho_1)}$$

$$E[D_{q2}] = \frac{\rho_1 \frac{\lambda}{2}E[D_s^2]}{(1-\rho_1)} + \frac{\lambda}{2}E[D_s^2]}{(1-\rho_2-\rho_1)}$$

$$E[D_{q2}] = \frac{\lambda E[D_s^2]}{2(1-\rho_2-\rho_1)(1-\rho_1)} \quad (3)$$

The average waiting time of the low priority packets is given in (3) and it differentiates it from the high priority packets.

3. METHODS AND MATERIALS

This section is discussed under the following sub-sections: problem formulation, tree structure and packets' characteristics, node arrangement of packets in the AVL tree, packets' re-transmission from tree to buffer, design of fuzzy admission control policy, simulation as well as results and discussion.

3.1 Problem Formulation

It is conventional for the TCP flow control mechanism to slow down the sending rates to avoid packet losses when the network becomes congested. This is one of the inbuilt design features of the TCP flow control mechanism [25]. In a differentiated service network, whenever there is a signal of incipient congestion, the controller denies admission to packets having low service preferences by dropping them to pave way to packets with higher service preference. Consider a single-server queuing system having two classes of packets' arrivals as depicted in Fig. 1.

In Fig. 1, there are *N* arrival sources. These arrivals are then classified into independent Poisson arrivals of class A and class B based on priority by the traffic conditioner. Fig. 2 is the schematic structure of the traffic conditioner.

A traffic conditioner is a part of a network node that takes the node's ingress as its input and places the packets in its output so as to satisfy service requirements' set [24]. The classifier selects packets from a traffic stream based on the content of some portion of the packet header; the meter measures the temporal properties of the stream of packets selected by a classifier against the traffic profile specified in the traffic conditioning agreement. Consequently, it passes state information to the marker, which identifies whether a packet is of high priority or otherwise, while the dropper manages incoming stream by dropping packets that are of low priority, if their admission would result in system congestion or delay service to packets of higher priority.

Packets arrive at the buffer at rates λ_1 and λ_2 for classes *A* and *B* respectively while the buffer has unlimited capacity and the order of service is insignificant. An exponential server services the packets at rate μ. Class *A* packets enter the buffer without restriction while Class *B* packets could either be admitted or denied admission and consequently dropped into the bottleneck router. The system receives a fixed reward (*r)* for each admitted arrival of packet and pays a holding cost (*h)* per arrival per unit time in the system.

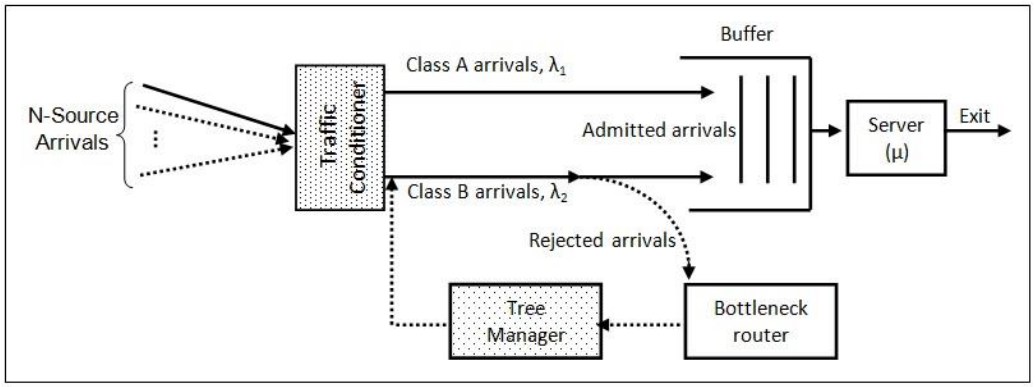

Fig. 1. A single-server queue with arrival and congestion control

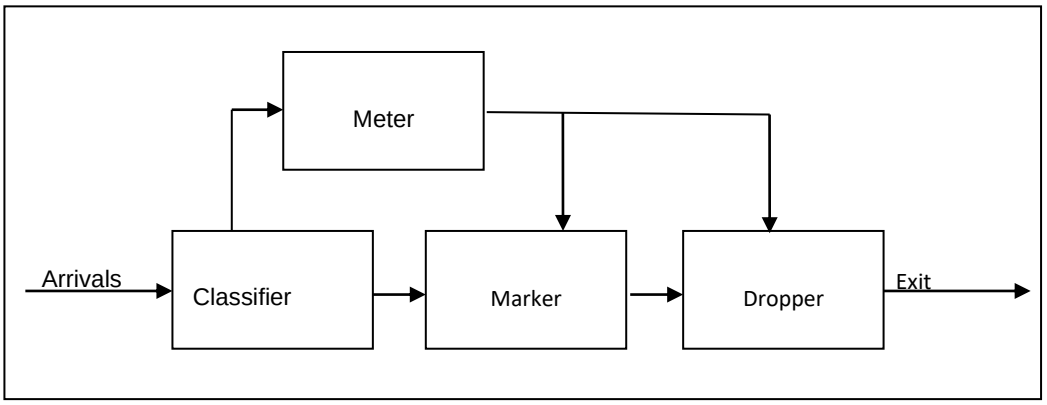

Fig. 2. Schematic structure of a traffic conditioner

3.2 Tree Structure and Packets' Characteristics

A tree is an abstract data type which simulates a hierarchical structure with a root value and subtrees of children with a parent node, represented as a set of linked nodes [26]. It is a non-linear and hierarchical data structure consisting of a collection of nodes such that each node of the tree stores a value which is a list of references to the nodes i.e. the children. Generally, a tree consists of a root and zero or more subtrees i.e. $T_1, T_2, ..., T_n$, such that there is an edge from the root of the tree to the root of each subtree [27]. In the design of the proposed model, the structure adopted is an AVL tree structure with several nodes which represent network packets. An AVL tree is a self-balancing binary search tree, where the difference between heights of left and right subtrees for any node cannot be more than one. The difference between the heights of the left subtree and the right subtree for any node is known as the balance factor (bf) of the node.

A packet carries important information such as size, time-to-live, source, destination IP address, checksum for error detection, 16-bit identification number, etc. It may also indicate whether or not the packet can be fragmented while also including information about re-assembling fragmented packets, i.e. fragmentation offsets [28]. However, since packets are treated as nodes in the tree, packet's information considered in the design of the proposed model is the node size. The node size refers to the size of packet (in bytes), treated as nodes in the tree. The minimum size of an internet protocol packet is 21 bytes (which includes 20 bytes for the header and 1 byte of data) while the maximum size is 65,535 bytes. However in practice, most packets are about 1500 bytes. For the purpose of this model, the size of packets, treated as nodes does not include the size of the header but rather the size of data only. The node size of each packet is arranged in the AVL tree structure.

3.3 Node Arrangement of Packets in the AVL Tree

If it is assumed that the following packets sizes (in megabytes): 43, 54, 21, 17, 90, 36, 49, 28 and 46 were dropped and transmitted to the bottleneck router as a result of incipient congestion. The packets are arranged as nodes in the AVL as depicted in Fig. 3.

It is important to note that the balance factor is indicated for each node in Fig. 3. If another node with size 25 megabytes is dropped and consequently transmitted to the bottleneck router, it is added as a node to the existing tree. With this addition, the new tree is depicted in Fig. 4.

3.4 Packets' Re-Transmission from Tree to Buffer

Transmission from the AVL tree is done by considering the root node in the tree. The root node is 43 and this is the first to be removed and transmitted to the buffer for service. Once node 43 is removed, the new structure of the tree is as depicted in Fig. 5.

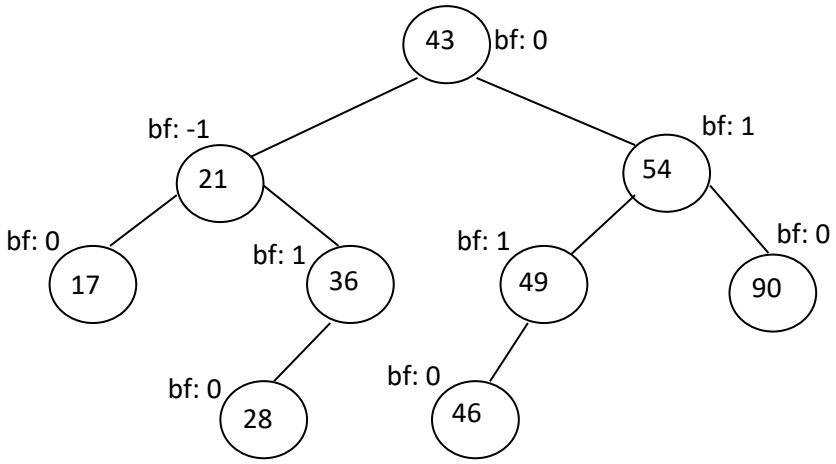

Fig. 3. A balanced AVL tree structure for dropped packets arranged as nodes

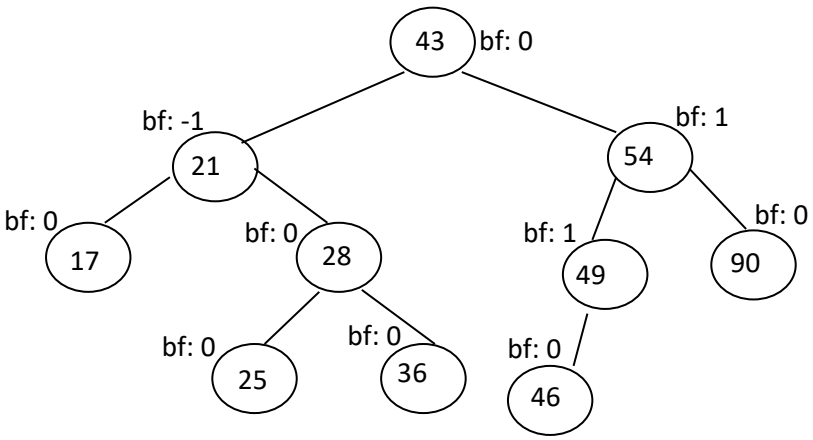

Fig. 4. The balanced AVL tree structure after the insertion of node 25

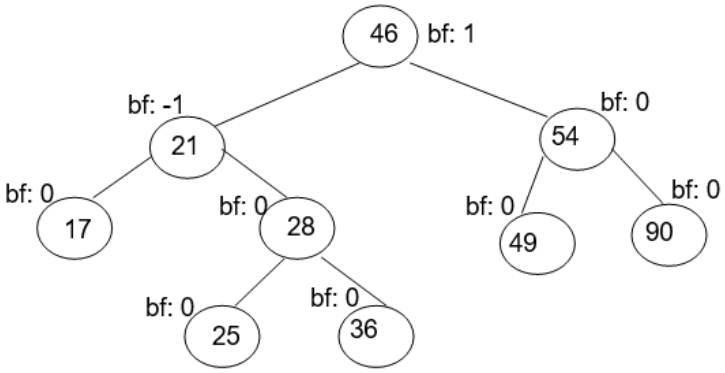

Fig. 5. Resulting AVL tree structure after the removal of node 43

The process of removal of node from the tree and consequent transmission to the buffer for service takes the same form for other nodes until all nodes in the tree are transmitted one after the other.

3.5 Design of the Fuzzy Admission Control Policy

The problem of admission control in a queue system requires proper planning for the system to perform optimally. A system based on fuzzy logic theory is an appropriate means to perform this task. Such a system is a combination of fuzzy numbers / membership functions and fuzzy "If-Then" rules that supports representation of complex systems into a processable linguistic model that deals efficiently with the characteristic of uncertainty / inexactness of human thinking and perception in most real-world problems [29,30].

In the proposed system, the decision epochs at which packets' arrival are controlled coincide with the arrival times of class B packets. As a result, the state of the system at the decision epochs is described by the total number, *y* of class A and class B packets in the system including the one in service, if any. Hence, *y = 0, 1,* …., *n.* In order to avoid a trivial situation in which a class B packet is denied admission even while the system is empty, it is assumed that $r > h/\mu$ .With this, it is beneficial to the system to admit a class B packet when the system is empty. For this case, the crisp rule for *y = 0* could be expressed as:

> *"If there are no packets in the system, then a class B packet seeking admission into the system is admitted"*.

Similarly, it is necessary to consider cases where $y \geq 1$. As only class *A* packets are uncontrolled, then the system is stable if $\lambda_1 < \mu$. When the arrival rate is high, it becomes necessary to reject class *B* packets so as to avoid queuing

delays and costs. The number of packets in the buffer, *s* is chosen such that $s = y - 1, s = 0, 1, \ldots$.

The arrival rates $\lambda_1 \in [0, \mu)$ and $\lambda_2 \in [0, \alpha)$ serve as fuzzy inputs with four linguistic values "zero", "fairly positive", "positive" and "highly positive" represented as "ZE", "FP", "PO" and "HP" respectively. The decision, (*d* =1, 0) is the fuzzy output in which "1" indicates "admit a class *B* packet" while "0" indicates "reject a class *B* packet". There are three main arguments on the basis of which the fuzzy rules are formulated. These are as follows:

i. A higher arrival rate of class B packets weakens the decision, *d:* "admit a class B packet". In this case, it is optimal to accept new packets as it merely incur holding costs.
ii. As arrivals increases, the holding cost becomes greater than the reward. Consequently, packets are denied admission.
iii. Class A arrivals, λ_1 negatively affects the decision to admit a class B arrival in anticipation of future uncontrolled class A arrivals.

Since there are four linguistic values and three crisp input, we have 4^3 i.e. 64 rule combinations. However only rule combinations with output decision "Admit" and another 10 rule combinations with output decision "Reject" were randomly chosen. These are as follows:

*Rule Number 1: If **s** is ZE and **λ₁** is ZE and **λ₂** is ZE, then **Admit** Arrival*
*Rule Number 2: If **s** is ZE and **λ₁** is ZE and **λ₂** is FP, then **Admit** Arrival*
*Rule Number 3: If **s** is ZE and **λ₁** is ZE and **λ₂** is PO, then **Admit** Arrival*
*Rule Number 4: If **s** is ZE and **λ₁** is FP and **λ₂** is PO, then **Reject** Arrival*
*Rule Number 6: If **s** is ZE and **λ₁** is FP and **λ₂** is ZE, then **Reject** Arrival*
*Rule Number 7: If **s** is ZE and **λ₁** is FP and **λ₂** is FP, then **Admit** Arrival*
*Rule Number 11: If **s** is FP and **λ₁** is ZE and **λ₂** is FP, then **Admit** Arrival*
*Rule Number 13: If **s** is FP and **λ₁** is FP and **λ₂** is ZE, then **Admit** Arrival*
*Rule Number 15: If **s** is ZE and **λ₁** is PO and **λ₂** is FP, then **Admit** Arrival*
*Rule Number 16: If **s** is ZE and **λ₁** is PO and **λ₂** is ZE, then **Admit** Arrival*
*Rule Number 17: If **s** is ZE and **λ₁** is ZE and **λ₂** is HP, then **Reject** Arrival*
*Rule Number 19: If **s** is FP and **λ₁** is PO and **λ₂** is HP, then **Admit** Arrival*
*Rule Number 23: If **s** is FP and **λ₁** is ZE and **λ₂** is ZE, then **Reject** Arrival*
*Rule Number 27: If **s** is ZE and **λ₁** is HP and **λ₂** is ZE, then **Reject** Arrival*
*Rule Number 29: If **s** is PO and **λ₁** is ZE and **λ₂** is HP, then **Admit** Arrival*
*Rule Number 32: If **s** is ZE and **λ₁** is PO and **λ₂** is HP, then **Reject** Arrival*
*Rule Number 39: If **s** is FP and **λ₁** is FP and **λ₂** is FP, then **Reject** Arrival*
*Rule Number 42: If **s** is FP and **λ₁** is FP and **λ₂** is PO, then **Reject** Arrival*
*Rule Number 45: If **s** is PO and **λ₁** is ZE and **λ₂** is ZE, then **Reject** Arrival*
*Rule Number 49: If **s** is HP and **λ₁** is ZE and **λ₂** is ZE, then **Reject** Arrival*

The membership functions for λ_1 is shown in Fig. 6.

The output, d is represented with a singleton membership function assigns membership value "1" to "Admit an arriving packet" and value "0' to "Reject an arriving packet". The membership functions for output, *d* is represented with an impulse function as shown in Fig. 9.

Mathematically, this is formulated as:

$$\mu(d) = \begin{cases} 1, & \text{if } d = 1, \\ 0, & \text{otherwise} \end{cases}$$

It implies from the rule *if s is HP and* λ_1 *is ZE and* λ_2 *is ZE, then d is REJECT. Hence HP* for *s* is expressed in (4):

$$y_{\lambda 1} = y_{\lambda 1} = 0 = {r\mu}/{h} \quad (4)$$

For λ_1, the rule *if s is ZE and* λ_1 *is HP and* λ_2 *is ZE, then d is REJECT*. The threshold value can be calculated using (5):

$$\lambda_1, s = \lambda_2 = 0 = {\mu(r\mu - 2h)}/{h} \quad (5)$$

Since λ_2 is bounded from above by μ, it can be expressed as (6):

$$\lambda_2, s = \lambda_2 = 0 = \mu \quad (6)$$

The admission and rejection of packets by the admission controller is a semi-Markov decision process. This makes it possible for the system to admit a packet provided that the number *y* of customers in the buffer is below a threshold value. This implies that the optimal threshold y_i is expressed in (7):

$$y_1 \leq \frac{r\mu}{h} < y_i + 1,$$

However, the socially optimal threshold y_s is given by:

$$\frac{y_s(1-\rho)-\rho(1-\rho^{ys})}{(1-\rho)^2} \leq \frac{r\mu}{h} < \frac{(y_s+1)(1-\rho)-\rho(1-\rho^{ys+1})}{(1-\rho)^2} \quad (7)$$

where $\rho = \lambda / \mu$

3.6 Simulation, Results and Discussion

The fuzzy controller to an M/M/1 queuing system with two arrival streams with parameters: λ_1 = 0.25, λ_2 = 0.05, μ = 0.15, h = 2 and r = 50 was considered. The simulation began with an initial state *y* = 0, and the system performance for the first 300 time units is as depicted in Fig. 10.

An observation of figure 10 indicates that the number of customers in the system, i.e. *y* does not exceed 4. This implies that the admission controller will admits an arriving packet (i.e. d=1) into the buffer provided that the value of y ≤ 4. On the other hand, the admission controller will deny admission (i.e. d = 0) to an arriving class B packet into the system if the value of y > 4. Consequently, it is obvious that the threshold value, $y_s = 4$. The implication of this is that the two categories of packets in the system are service-prioritized.

Packets with priority value less than or equal to 4 are treated as high-preference service arrivals while those with priority value greater than 4 are treated as low-preference service arrivals. This is the basis of classification and treatment when incipient congestion is signaled. If the system discovers that the admission of a low-priority packet will affect the admission and / or service of a high-priority packet, such arrival is dropped into the bottleneck router and consequently transferred to the tree manager where it is treated as a node in an AVL tree structure and later re-transmitted into the buffer when it is safe to do so.

The decision epochs at which arrivals are controlled has a direct relationship with the arrival times of class *B* packets. It is obvious therefore that the state of the system at the decision epochs is a reflection of the total number of packets in the buffer, including the one in service. The management of admission of packets in the system is therefore a semi-Markov decision process. This is why it possible for the system to admit a packet provided that the number *y* of customers in the buffer is below the threshold value of: $y_1 \leq \frac{r\mu}{h} < y_i + 1,$ With this, it is optimal to admit a class *B* packet provided the buffer is empty. In this case, the value of reward r, is expressed as: r > h/μ, consequently encouraging the admission of a class *B* packet since it has no effect on any other class of arrival.

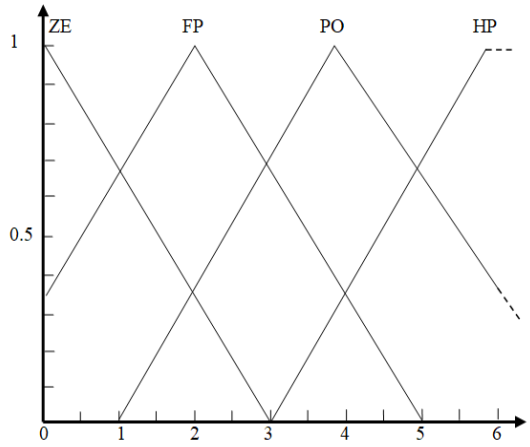

Fig. 6. The membership functions for λ_1

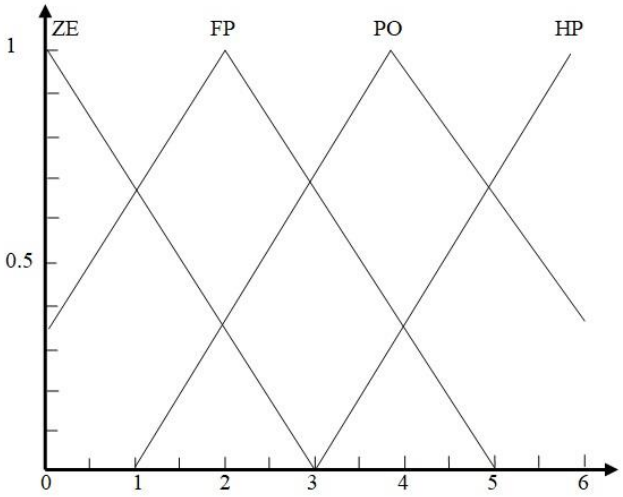

Fig. 7. Membership functions for λ_2

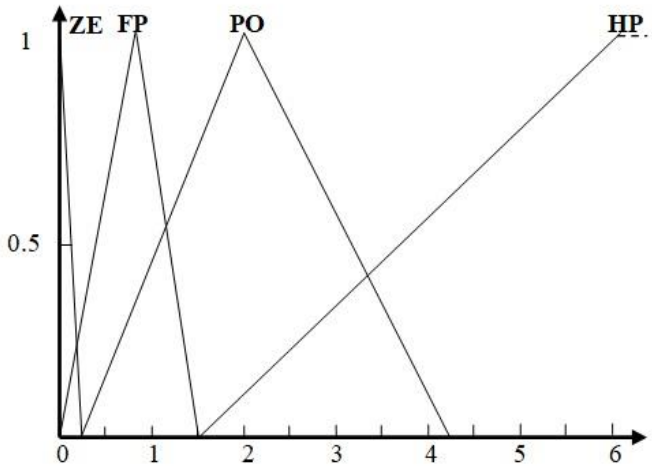

Fig. 8. Membership functions for *s*

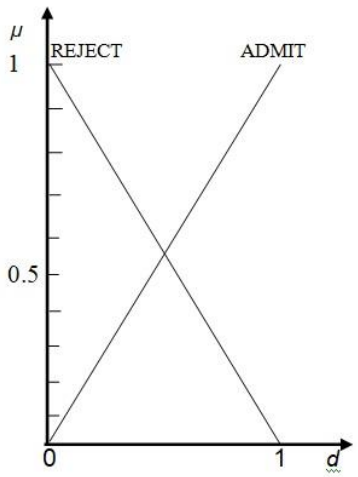

Fig. 9. Membership functions for output, *d*

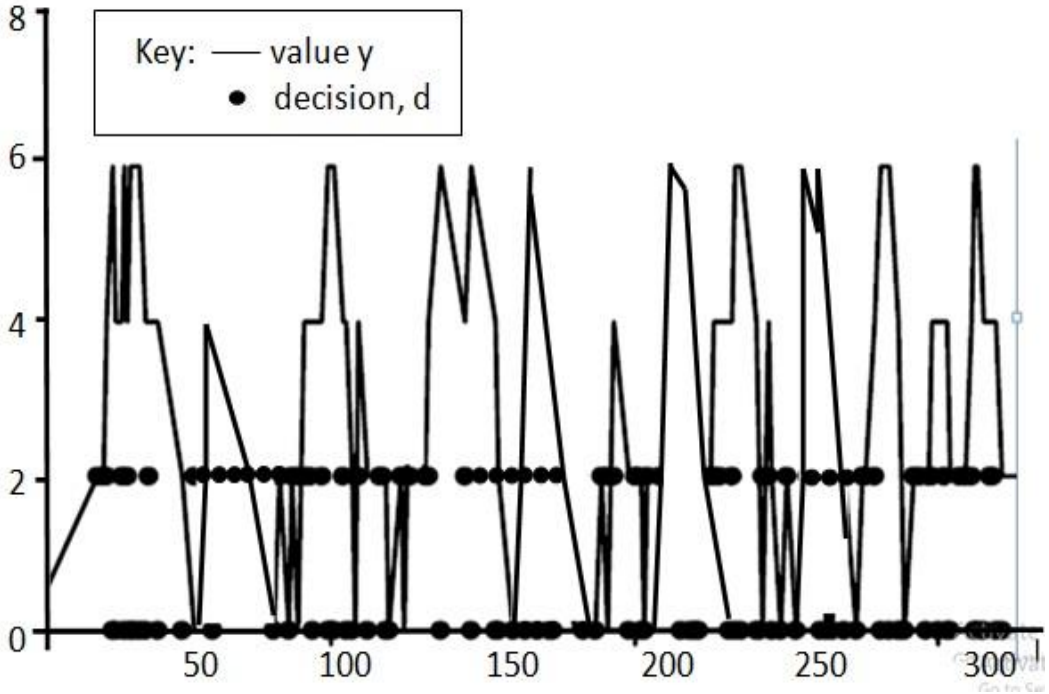

Fig. 10. The fuzzy control policy for the evolution of *y* and decision, *d*

4. CONCLUSION

The study considered a priority-based admission control system for managing congestion in a differentiated service network. Packets of varying sizes of data were transmitted from *N* sources and classified into two independent sources of high and low priorities. While class *A* packets are not denied admission into the buffer, this is not the case with class B packets. Arrivals denied admission are dropped and consequently transmitted to the tree manager via the bottleneck router so that the packets could be arranged as nodes in an AVL tree structure which adopts tree properties to manage and transmit nodes to the buffer based on node rotations. Consequently, decision epochs on which admission policy is based has direct relationship with the arrival times of class *B* packets. This implies that the control of admission is a semi-Markov decision process which is based on the characteristics of the crisp inputs being processed using a fuzzy logic controller to arrival at optimal decisions as far as

admission control is concerned.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Hoiland-Jorgensen T, McKenney P, T'aht D, Gettys J. and Dumazet E. The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm. RFC 8290; 2018.
2. Iyengar J, Thompson M. QUIC: A UDP-Based multiplexed and secure transport. Internet Draft Work. Internet Society; 2018.
3. Hoiland-Jorgensen T, T'aht D, Morton J. Piece of CAKE: A Comprehensive Queue management solution for home gateways. In Proceedings of IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN 2018). (Washington, District of Columbia, USA); 2018;37–42.
   DOI: 10.1109/LANMAN.2018.8475045.
4. Miao W. Stochastic Performance analysis of network function virtualization in future internet, IEEE Journal on Selected Areas in Communications. 2019;37(3):613-626.
5. Floyd S. Gummadi R, Shenker S. 2001. Adaptive RED: An Algorithm for increasing the robustness of REDs Active Queue Management". ICSI Technical Report.
   Available: http://www.icir.org/oyd
6. Hamdi MM, Mahdi HF, Abood MS, Mohammed RQ, Abbas AD, Mohammed AH. Review on queue management algorithms in large networks. 2nd International Scientific Conference of Engineering Sciences (ISCES) 2020; 110.
   DOI: 10.1088/1757-899X/1076/1/012034
7. Luo Y, Zhou R, Liu J, Qiu S, Cao Y. An Efficient and self adapting colour image encryption algorithm based on chaos and interactions among multiple layers. Multimedia Tools and Applications. 2018;*77*(20):26191-26217.
8. Li RJ, Lee E. Analysis of Fuzzy Queues. Computers and mathematics with applications; 1989;17(7):1143–1147.
9. Yeh C, Lee YT, Chang CJ, Chang FM. Analysis of a two-phase queue system with < *P, F* > - Policy. Quality Technology and Quantitative Management. 2017;14(2): 178–194.
10. Floyd S, Jacobson V. Random Early detection Gateways forCongestion Avoidance. IEEE/ACM Transactions on Networking. 1993;1(4):397-403.
11. Meena RK, Kumar P. Performance analysis of markov retrial queueing model under admission control F-Policy. Mathematical Modeling and Computation of Real-Time Problems. 2021. 65–78. CRC Press.
12. Negi D, Lee E. Analysis and Simulation of Fuzzy Queues. Fuzzy Sets and Systems. 1992;46(3):321–330.
13. Devaraj J, Jayalakshmi D. A Fuzzy Approach to Priority Queues. International Journal of Fuzzy Mathematics and System. 2012;2(4):479–488.
14. Jain M, Sanga SS. Admission Control for finite capacity queueing model with general retrial times and State-dependent Rates. Journal of Industrial and Management Optimization. 2020;16(6):2625.
15. Zhang R, Phillis YA, Kouikoglou VS. Fuzzy control of queueing systems. Springer Books. Springer-Verlag London Limited; 2005.
16. Adesh N, Renuka A. Avoiding queue overflow and reducing queueing delay at eNodeB in LTE Networks Using Congestion Feedback Mechanism, Computer Communications. 2019;146(1): 131-143.
17. Miao W. et al. Stochastic performance analysis of network function virtualization in future internet, IEEE Journal on Selected Areas in Communications. 2019;37(3):613- 626.
18. Okokpujie KO, Chukwu EC, Noma-Osaghae E, Okokpujie IP. Novel active queue management scheme for routers in wireless networks. International Journal on Communications Antenna and Propagation. 2018;8(1):53-61.
19. Patel S, Bhatnagar S. Adaptive Mean Queue Size and its Rate of Change: Queue Management with Random Dropping. Telecommunication Systems 2017;65(2):281-295.
20. Kamal AA, Murshed M. Adaptive RED with Dynamic Threshold Adjustment, Research Report, Lowa State University. 2005:1-42.
21. Al-Allaf AF, Jabbar AA. RED with Reconfigurable maximum dropping probability. International Journal of

Computing and Digital Systems. 2019;8 (1):61-72.
22. Johansson I. TCP HyStart Patch Deployment. Post on IETF TCPM Working Group Mailing List; 2015. Available:https://www.ietf.org/mail archive/web/tcpm/ current/msg09675.html
23. Jarvinen I. Congestion control and active queue management during flow startup. Doctoral Dissertation in the Department of Computer Science, University of Helsinki, Kumpula. 2018:102-104.
24. Clark D, Fang W. Explicit allocation of best effort packet delivery service. IEEEYACM Transactions on Networking. 1998;6(4): 362 – 373.
25. Bellalta B, Occhsner S. Analysis of packet queueing in telecommunication networks. Course Notes. (B.Sc.) Network Engineering. 2nd Year. 2011:104-106
26. Jiang H, Liu Z, Wang Y, Lee K, Rhee I. Understanding Buffer-bloat in Cellular Networks. In: Proceedings of 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design (CellNet '12). (Helsinki, Finland). 2012:1–6. DOI: 10.1145/2342468.2342470.
27. James S, Prakash P, Nandakumar R. The Tree List: Introducing a Data Structure. International Journal of Recent Technology and Engineering (IJRTE) 2019;7(6):1093- 1095.
28. Gbadebo AD, Abimbola GO, Iposu ON. and Salami IA. A Treap-Based congestion control model for M/G/k Queue Network. Proceedings at the First International Conference of the College of Science and Information Technology, Tai Solarin University of Education, Ijebu-Ode, Ogun State, Nigeria. 2023;102-110.
29. Dordal PL. An introduction to computer networks. Department of Computer Science, Loyola University, Chicago. 2021: 537.
30. Amaitik N. The Basics of Fuzzy Systems Technology: A Complete Tutorial. Research Gate; 2020. Available:https://www.researchgate.net/publication/345503118

Peer-review history:
The peer review history for this paper can be accessed here:
https://www.sdiarticle5.com/review-history/111554