Fuzzy Logic Controller of Gentle Random Early Detection Based on Average Queue Length and Delay Rate

Mahmoud Baklizi, Hussein Abdel-Jaber, Ahmad Adel Abu-Shareha, Mosleh M. Abualhaj, and Sureswaran Ramadass

Abstract

This paper proposes a controller technique for early stage congestion detection at the router buffer in the networks. The proposed technique extends the well-known Gentle Random Early Detection (GRED) algorithm. Unlike GRED, which relies on parameter settings, such as minthreshold, maxthreshold and double maxthreshold, in order to obtain a satisfactory performance, the proposed technique depends on a fuzzy logic system which reduces the large dependency on parameter settings. The proposed technique uses the average queue length and the delay rate as input linguistic variables for a fuzzy logic system. The utilized fuzzy logic system produces a single output that represents a packet dropping probability, which in turn control and prevent congestion in early stage. The proposed technique and the well-know GRED and REDD1 algorithms were simulated using Java environment. The performance of the proposed technique has been evaluated and compared with regard to various performance measures, which are: mean queue length, throughput, average queuing delay, packet loss and packet dropping probability. The simulation results show that the proposed technique, in comparison with the existing algorithms, offers better performance results in terms of mean queue length, average queuing delay and packet loss. Therefore, this technique, generally, improves the network performance.

Corresponding Author: Mahmoud Baklizi is with the National Advanced IPv6 Center of Excellence (NAv6), Universiti Sains Malaysia, Penang 11800, Malaysia. E-mail: mbaklizi@nav6.org
Hussein Abdel-jaber is with the Department of Computer Information and Network Systems, The World Islamic Sciences and Education (W.I.S.E.) University, Jordan. E-mail: husein.abdeljaber@wise.edu.jo
Ahmad Adel Abu-Shareha is with the Department of Computer Science, The World Islamic Sciences and Education (W.I.S.E.) University, Jordan. E:mail: abushareha@wise.edu.jo
Mosleh M. Abualhaj is with the Department of Networks and Information Security, Faculty of Information Technology, Al-Ahliyya Amman University, Jordan. E-mail: m.abualhaj@ammanu.edu.jo
Sureswaran Ramadass is with the National Advanced IPv6 Center of Excellence (NAv6), Universiti Sains Malaysia, Penang 11800, Malaysia. E-mail: sures@nav6.org
Manuscript received 3 July 2012; revised 11 Sep. 2013; accepted 12 Feb. 2014.

Keywords: Congestion control, fuzzy logic, gentle random early detection, queuing systems.

1. Introduction

With the rapid growth of computer networks and Internet technologies, managing network resources such as bandwidth allocation and queue spaces in various networks is essential. When the networks fail to manage and keep up its resources, unfair bandwidth sharing among the network connections occurs. In such a case, some network connections may engage queue spaces more than others. Subsequently, these connections will increase their transmitting rates compared to others, this is what so called aggressive connections [1, 2]. When the transmission rate of some aggressive connections increases, the router queues built up, accordingly, router queues are overflowed and leads to unmanageable packets dropping. In such a case, the network is said to be congested.

Congestion is one of the major problems that challenge network performance [3, 4]. Congestion occurs at the buffers of the network routers when the amount of incoming packets exceeds the available network resources and the buffer can no longer handle all incoming packets [5]. Generally, congestion plays a major role in worsening computer network performance by increasing the packet dropping probability (Dp) and growing the packet loss probability (P_L). In addition, congestion may lead to an increase in the mean queue length (mql) and the packets average queuing delay (D), congestion may also cause an unbalanced share among the network sources which successfully degrade the amount of packets passing through the buffer of the routers, namely, the throughput (T) [1].

Early work in controlling congestion comes up with Drop Tail (DT) technique [6, 7]. DT control congestion using a fixed router buffer size assigned based on the network administrator awareness. Generally, there are two scenarios in which the DT is executed. First, DT sets the router buffers to the maximum in order to obtain a high T. However, this causes a drawback of high D. Second, DT sets its router buffer to relatively small length. In such a case the network resources managed by DT technique provide low D. However, this scenario accommodated with several drawback of high P_L, high D_p and low *T*. Finally, DT suffers also from other drawbacks such as lockout phenomenon and full router buffers. Generally, it has proven that DT degrades the performance of network [8].

Later on, Active Queue Management (AQM) methods have been developed to overcome the aforementioned DT problems and provide sufficient resource management [4, 9, 10]. Random Early Detection (RED), one of the most significant algorithms for congestion control, manages congestion before the router buffer overflows based on the computed average queue length (*aql*) and the calculated minimum and maximum thresholds values [8]. Generally, RED detects congestion as follows: when a packet arrives at the router buffer, RED computes *aql* of the underlying buffer and compares it with the minimum and maximum threshold positions. An *aql* value that is smaller than the minimum threshold gives a sign for no congestion, thus, the packet is passed to the queue and no packet is dropped. If the *aql* value is between the two thresholds, the arriving packet is dropped probabilistically to alleviate congestion at the underlying buffer. Finally, when the *aql* is above the maximum threshold, all arriving packets are dropped at a D_p value equal to one.

Based on the previously discussed scenarios, RED algorithm provides acceptable performance when the traffic load is steady. However, when the load increases suddenly, RED drops many subsequent packets which lead to reduce the network performance. In some other situation, because RED depends on the amount of traffic load in controlling congestion, the computed *aql* may become above the maximum threshold, and as a result, every arriving packet will be dropped. In addition to the previously discussed drawbacks, RED requires a superior parameter setting for the maximum and minimum thresholds, queue weight (*qw*) and maximum packet dropping probability (D_{max}), to ensure achieving a satisfactory performance. Parameter setting, however, is not necessary possible in actively changed networks.

Generally, the need for accurate parameter settings and the expected unbalanced load in most networks made RED inefficient technique. Consequently, this paper proposes a dynamic technique for congestion control based on the existing AQM algorithms and using Fuzzy Logic (FL) system that identifies congestion incipiently at router buffers. The purpose of the proposed technique is to improve the network performances when a high congestion situation occurs without the need for great parameter settings.

Fuzzy logic, which is commonly known as Computational Intelligence (CI), is one of the most important tools that have been used to control methods in communication data networks, as fuzzy logic is effective alternative for heavily parameterized systems [11]. Fuzzy logic approach in classical control theory is used either to alleviate the system's complex parameters in the mathematical model, or to simplify the model to some extent, in order to obtain some stability results, or to make model tractable for the controller design [12]. In addition, Fuzzy Logic Control (FLC) has been used successfully for controlling many systems in which analytical models are not easily obtainable or the model itself, if available, is too complex and possibly highly nonlinear [11].

For congestion control, in recent years, fuzzy logic has been used as a solution to several problems and has demonstrated the applicability of fuzzy logic to the problem of congestion control [13, 14]. FLC has been used due to its capability of qualitatively capturing the attributes of a control system based on observable phenomena. Thus, if the FLC is designed with a good (intuitive) understanding of the system, the limitations due to the complexity of the system's parameters can be avoided [11, 12].

Generally, the proposed technique uses Fuzzy Inference Process (FIP) as congestion detectors. FL is a set of mathematical expressions for knowledge representation [15-18]. The output of FL system, unlike the classical Binary Logic (CBL) [19], is a continuous truth value between (0-1) [15-18]. Fuzzy Logic Controller (FLC) is an expert system, implements a knowledge-based decision using some experience [16, 17]. FLC component process an input and produce an output by applying them into fuzzy linguistic rules. Generally, FLC has four steps (fuzzification, evaluation of the rules, aggregation the outputted rules, deffuzzification) [17, 20]. Those stapes are implemented and discussed in the rest of the paper as a basis for congestion control in the core of the proposed technique.

The rest of the paper is organized as follows. Section 2 presents previous related work. The proposed algorithm is discussed in Section 3. Section 4 presents the simulation information. The results of the developed simulation are discussed in Section 5. Finally, conclusions are stated in Section 6.

2. Related Work

Enormous algorithms for congestion control have been developed based on RED and other discrete-time queue analytical models to enhance the network performance. Gentle Random Early Detection (GRED) [21] and REDD1 [22] are the most powerful algorithms in the literature [23-25]. GRED was proposed by Floyd to overcome RED's limitations [10, 21, 26]. The main goal of the GRED algorithm, similar to RED, is to manage and control the congestion networks at the early stage. Although, GRED employs a similar approach used by RED in calculating the D_p, it depends on stabilizing the *aql* at a certain level based on three thresholds which are: minimum, maximum, and double maximum. Generally, GRED control congestion as illustrated in Figure 1.GRED pseudo-code is described in Algorithm 1.

Unfortunately, GRED does not perform well in dynamically change network. This is because it sets its parameters to specific values (i.e., parameterization), consequently, when a heavy congestion suddenly occurs while *aql* is less than the minimum threshold, *aql* will take time to adjust, which will likely leads to buffer overflow during the adjustment process. In which case, no packets are dropped, although the GRED router buffer overflows.

REDD1 was proposed, by Thiruchelvi and Raja [14], based on calculating the *aql* for every arriving packet, similar to RED and GRED. However, the dropping probability (DP) is calculated using FL as D_P= {zero, low, moderate, high}. The value of D_p is determined using *aql* and P_L, which are considered as two input linguistic variables. These variables are linked to a fuzzy set. The fuzzy sets as, *aql* = {conservative, middle, aggressive} and P_L = {few, medium, a lot} The REDD1 algorithm is aimed to offer fewer PL result than RED, also REDD1 decreases the RED algorithm dependency on its parameters, i.e. minimum and maximum thresholds.

Adaptive Fuzzy RED (AFRED) [27] developed a FL congestion control algorithm using a single input linguistic variable (current queue length) to produce a single output variable (dropping probability). The simulation results of [27] showed that AFRED outperforms RED in terms of the queue length and throughput. Meanwhile, several algorithms have been developed using Fuzzy Logic (FL) in association with AQM, such as those proposed by Chrysostomou [18] and Chrysostomou [28]. In general, several algorithms that implements association of FL and RED technique within TCP/IP, use different linguistic rules for each class of service. Most of these techniques use two input linguistic variables, which are current queue length and the change rate in the traffic load, and produce a single output linguistic variable, which is the packet dropping probability. The results reveal that the association of FL and RED outperforms RED with regard to the optimization of queue size and throughput [28-31].

Unfortunately, all the aforementioned related methods fail to implement a congestion control that can efficiently address the expected congestion cases encountered by the network resources, which in turn effect and waste the network resources.

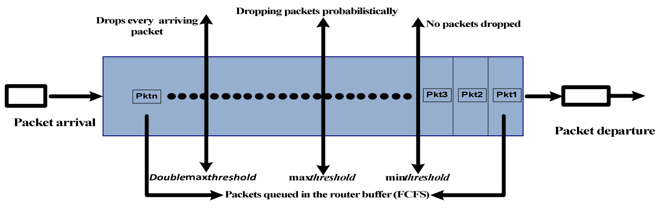


Figure 1. The single router buffer for GRED.

3. The Proposed Technique

The proposed technique for congestion control calculates the dropping probability based on the calculated *aql* and D. The proposed technique, unlike the existing technique, does not rely upon certain parameters settings; rather, this technique employs FIP as congestion detectors. In addition, the proposed technique aims at obtaining more satisfactory performance measure results when a heavy congestion occurs. Using Fuzzy logic, the proposed technique calculates the dropping probability of each arriving packets based on two input linguistic variables (*aql*, delay (D)), as illustrated in Figure 2. The proposed technique is described in Algorithm 1 and its parameters are defined in Table 1.

Algorithm 1: The Proposed Technique

```
1.  Begin
2.  SET $C$ = -1, $aql$ = 0.0 // Initialization stage
3.  FOR every arriving packet at a GRED router buffer,
    do //2nd Stage Calculate the aql for the arriving packet
    at the router buffer.
4.  Examine the queue status at the router buffer (e.g.
    empty or not
5.  IF, the queue at the router buffer = = empty, do
6.  Compute $n$, where $n = q(current\_time - idle\_time)$
7.  Set $aql = aql \times (1 - qw)\, n$
8.  ELSE
9.  Set $aql = aql \times (1 - qw) + qw \times q\_instantaneous$
10. End IF
11. END FOR
12. Check the congestion status at the router buffer //3rd
    stage
13. IF, aql < min_threshold, do
14. SET $D$ = 0 .0; // No packets have dropped
15. SET $C$ = — 1;
16. ELSE IF min threshold < aql & & aql < max threshold, do
17. SET $C = C + 1$;
18. Calculate $D_{init} = \frac{D_{max} \times (aql - \min threshold)}{\max threshold - \min threshold}$
19. Calculate $D_p = \frac{D_{init}}{(1 - C \times D_{init})}$
20. Drop arriving packet probabilistically in terms of its
    $D_p$ value;
```

```
21. SET $C = 0$;
22. ELSE IF $max\ threshold \le aql\ \&\ \&aql < double$ max threshold, do
23. SET $C = C + 1$;
24. SET $D_{init} = \mathrm{D}_{\max} + \frac{(1-\mathrm{D}_{\max})\times(aql - \max threshold)}{\max threshold}$
25. SET $D_p = \frac{\mathrm{D_{init}}}{(1 - \mathrm{C}\times\mathrm{D_{init}})}$
26. Drop arriving packet probabilistically in terms of its ($D_p$) value;
27. SET $C = 0$;
28. ELSE IF $aql \ge double$ max threshold, do
29. Drop every arriving packet with $D_p = 1$;
30. SET $C = 0$;
31. END IF
32. IF GRED router buffer becomes empty, do // 4th stage
33. SET idle time = current time ;
34. END IF
35. END
```

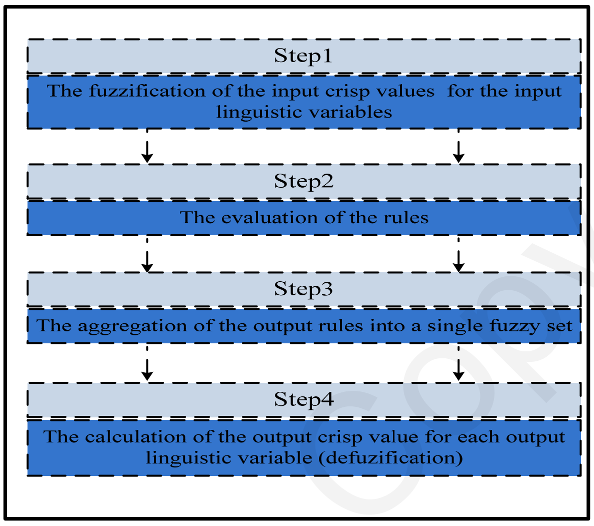

Step1
The fuzzification of the input crisp values for the input linguistic variables
Step2
The evaluation of the rules
Step3
The aggregation of the output rules into a single fuzzy set
Step4
The calculation of the output crisp value for each output linguistic variable (defuzification)

Figure 2. FIP to find the packet dropping probability.

Table 1. Adscription of parameter used.

Definitions	Description
current time	The current time.
idle time	The beginning waiting time at the router buffer.
n	The number of packets transmitted to the router buffer through an idle interval time.
C	A counter that represents the number of packets arrived at the router buffer and have not dropped since the last packet was dropped.
D*p*	The packet dropping probability.
D*init*	The initial packet dropping probability.
q_ instantaneous	The instantaneous queue length.
qw	The queue weight.
Dmax	The maximum value of Dinit.
q(time)	The linear function for the time.
T*aql*	target level for the aql
doublemaxthresh-old	is set to 2 x maxthreshold
K	Capacity of the buffer

The procedure for the proposed technique as illustrated in the figure is described as follows:

- *Step 1 (The fuzzification of the input crisp values (aql and delay)):* in this step the input crisp values are calculated to specify the membership degree for each crisp value. The fuzzy set range for each input linguistic value, based on the universe of discourse, can therefore be obtained. A crisp value denotes a numerical value placed on the universe of discourse.
- *Step 2 (Evaluation of the rule)*: in this step, the fuzzified input variables obtained//processed in the previous step is evaluated by applying them on the antecedent part of the rules. After every antecedent part is processed, the consequent part of every rule is then evaluated by obtaining the membership degree of the output variables. When multiple antecedent rules are found, the computation of all the antecedent rule parts is calculated using the fuzzy set operations [17, 32]. Then based on the results of the antecedent rules, the membership degree for every output linguistic rule is achieved.
- *Step 3 (Aggregating all the output rules into a single output rule (fuzzy set))*: Given that the degree of membership for each consequent rule part is obtained in the previous step, the combination of them into a single output rule is implemented in this step. This single output rule is called the single fuzzy set. The input for this step is a list of membership values for the output consequent rules and the output of this step is a fuzzy set for every output variable.
- *Step 4 (Defuzification)*: the final step of the FIP generates a crisp value for each output linguistic variable based on its fuzzy set. One of the popular defuzzification techniques is the center of gravity (COG) method [33], which aims to find out the point located on the center of the aggregate fuzzy set for each output linguistic variable. Formally, the COG can be defined according to equation (1) [17], for further information refer to [17].

$$COG = \frac{\Sigma_a^b F_S(S)\times S}{\Sigma_a^b F_S(S)} \qquad (1)$$

3.1 Fuzzy sets

Each linguistic variable in the FLC is associated with fuzzy sets in GREDFL. The following sets depict the fuzzy sets for the input and the output linguistic variables: *aql* = {conservative, middle, aggressive}, *D* = {Little, Average, Long} and D_P = {zero, low, moderate, high}. The fuzzy sets for each linguistic variable are chosen based on the behavior of their input linguistic variable. For example, if *D* input linguistic variable will be low, and this means average queuing delay for packets is low. Either average or a long represent a medium or a large average queuing delay for packets, respectively. Therefore, little, average and a long are the behaviors of the *D*.

3.2 Creating membership functions for GREDFL technique

After the fuzzy sets are identified, the membership functions will be generated. Generally, the membership function may take several shapes based on the problem at hand. For computational simplicity, the membership function of the linguistic variables, *aql* and D, are often considered as triangular or trapezoidal shaped. In the proposed GREDFL technique trapezoidal has been used. The chosen membership functions of the linguistic inputs and output values in the GREDFL controller, is shown in Figure 3 and 4. The amount of overlapping between the membership functions' areas is significant. The left and right half of the trapezoidal membership functions, for each linguistic value, are chosen to provide membership overlapping with adjacent membership functions. The overlapping of the fuzzy regions, representing the continuous domain of each control variable, contributes to a well-behaved and predictable system operation. Appendix A shows a scenario of how the memberships for both aql and D are created.

The sum of the membership's grades for an input value, which represents the linguistic values of a specific input variable, is always one. For the output variable, the membership functions at the outermost edges cannot be saturated for the GREDFL controller to be properly defined. The basic reason for this is that in fuzzy-based decision-making processes we seek to take actions that specify an exact value for the controlled system's input.

As illustrated in Figure 3, the final value of *aql* is K, where K represents the size of the router buffer as illustrated in table 1. On the other hand, in Figure 4, the final value of D is 2K, this value comes from K/beta, where beta represents the probability of packet departure, which equals 0.5.

Figure 5 displays the membership function of the D_p output linguistic variable. The assumption of membership functions for the D_p linguistic variables are similar to those in [14]. The boundaries of membership functions and fuzzy sets are chosen by domain experts in both FL and congestion control fields [17]. The consideration of a membership function for the *aql* linguistic variable is given as follows: *aql* will be in a conservative fuzzy set when its value is between zero and a 0.25 of the system capacity. However, the *aql* will be in the middle fuzzy set when its value is between 0.2 of system capacity and 0.75 of system capacity. Finally, the *aql* will be in the aggressive fuzzy set when its value is between 0.7 of system capacity and the finite capacity of system. Figures 7, 8 and 9 are either trapezoidal or triangular for simple computations [17].

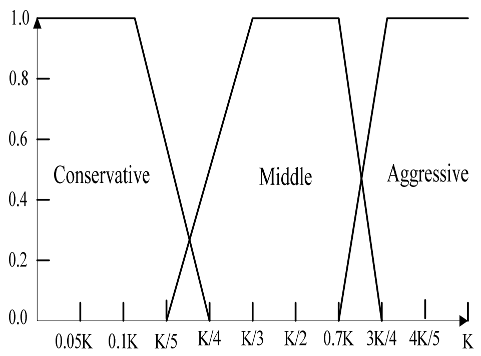


Figure 3. The memberships function of *aql*, where K represents the system capacity.

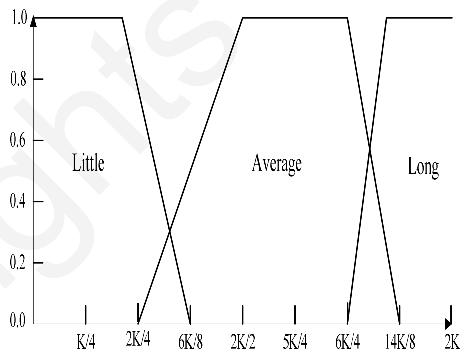


Figure 4. The memberships function of D.

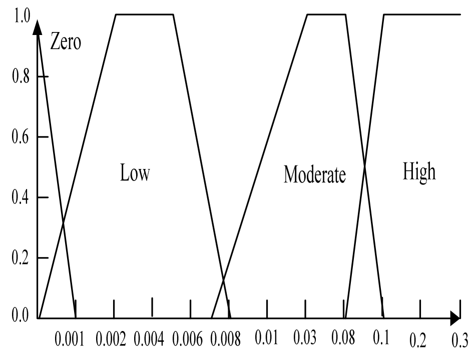


Figure 5. The memberships function of D_p.

3.3 The rules in GREDFL

In this section, the fuzzy logic, which captures human knowledge and experience about how to control congestion, is set up. Choosing simplest Multiple Input Single Output (MISO) controller leads to avoid the exponential increase of the rule base and decrease the complexity of the controller, when the number of input variables increases [12]. Generally, a good design of the rule-base in fuzzy logic is prepared based on two aims: First, completeness which means that all the conditions of the system behavior should be taken into the consideration, i.e., all arrangements of the input variables should produce an appropriate output values. Second, consistency which means that the rule base should not contain any illogicality. A set of rules is inconsistent if there are at least two rules with the same antecedents-part (input) with different consequent-part (output). However, to build the fuzzy rules, as shown in Figure 5, the input and output linguistic variables as well as all the fuzzy sets with their ranges on the universe of discourse must be known.

The knowledge-base for the fuzzy controller which is generated from IF-THEN control rules has the following form:{ IF *aql* is conservative and D is average THEN D_p is zero}. Where *aql* and D denote the linguistic variables associated with the two controller inputs, D*p* denotes the linguistic variable associated with the controller's output. The fuzzy rules are determined empirically to obtain the control signal according to the congestion in the router buffer. This relationship between the inputs and the output is mainly based on intuitive understanding and considerations (using expert knowledge) of the concept of congestion control. For example, if the *aql* is aggressive and the *D* is long then the output should be high in order that the system can respond quickly.

Table 2. The linguistic fuzzy rules of the proposed GREDFL algorithm based on *aql* and D.

IF *aql* is conservative and D is little THEN D*p* is zero
IF *aql* is conservative and D is average THEN D*p* is zero
IF *aql* is conservative and D is a long THEN D*p* is zero
IF *aql* is middle and D is little THEN D*p* is zero
IF *aql* is middle and D is average THEN D*p* is zero
IF *aql* is middle and D is a long THEN D*p* is low
IF *aql* is aggressive and D is little THEN D*p* is zero
IF *aql* is aggressive and D is average THEN D*p* is moderate
IF *aql* is aggressive and D is a long THEN D*p* is high

Table 2 indicates that if the *aql* is in a conservative fuzzy set, whatever the fuzzy set that the D belongs to it, the D*p* will be in a zero fuzzy set. In case that the *aql* is in a middle fuzzy set and the D is in either few or medium fuzzy set, the D*p* will be in a zero fuzzy set. However, if the D is in a lot fuzzy set, then the D*p* will be in a low fuzzy set. Finally, if the *aql* is in an aggressive fuzzy set, the D*p* result depends on the D fuzzy set. Hence, if the D is in a few fuzzy set, then the D*p* is in a zero fuzzy set, whereas if the D is in medium and a lot fuzzy sets, then the D*p* is in moderate and high fuzzy sets, respectively. In addition, the FIP is the main component in the GREDFL algorithm, which is used at each router buffer queue. The FIP employs two input variables (*aql*, D) to output a single output variable (D*p*). At any time a packet arrives at the router buffer queue in GREDFL, the FIP uses as congestion detector and controller at the router queues to derive the D*p* result through four steps that mentioned above in Figure 5. The first step is fuzzification in which the FIP takes the input crisp values of the *aql* and D to obtain their membership degrees. Now, based on the returned membership degrees, the fuzzy set for each input linguistic variable is determined on the universe of discourse. In other words, the fuzzification step determines the area to which each input linguistic variable belongs to based on its membership degree. After the fuzzification step, the rule body (IF-part) gets evaluated by applying the membership degrees of the input linguistic variables in the IF-part to obtain the membership degree of the output linguistic variable. Based on the membership degree of the output variable, the area which the output variable belongs to, can be determined. In the third step the membership degrees of the THEN-part of the rules are aggregated into a single fuzzy set. The final step is defuzzification, where the single aggregate fuzzy set of the output variable is inputted, then using the COG method [34, 35], the output crisp value for the D*p* is calculated.

4. Simulation

GRED, REDD1, and the proposed GREDFL are simulated based on a discrete-time queue that uses slot as a unit of time [25, 36]. Each slot may involve packet arrival and/or departure. The compared algorithms are simulated by applying them in a network consisting of a single router buffer node. Notably, both packet arrival and departure are implemented in single mode .The scheduling mode is first-come-first-served. The GRED, REDD1, and GREDFL simulations are implemented in Java on an i7 processor machine with 1.66 GHz and 4 GB RAM. In the conducted simulation, the probability of the arriving packets at the router buffer in a slot is denoted by α [36]. The probability of packet departure from the router buffer in a slot is denoted by β. Packet arrivals can be modeled using a Bernoulli process, whereas packet departures can be modeled using a geometrical distribution. Using geometrical distribution, packet inter-arrival times and service times are estimated to the values 1/ α and 1/β, respectively.

5. Performance Evaluation Results of the Fuzzy Logic Controller Algorithms

In this section, the proposed FLC algorithm (GREDFL) has been compared with GRED and REDD1algorithms according to different performance measures (*mql*, *T*, *D*, P_L, D*p*) to identify which algorithm offers the most satisfactory performance measure results.

For the parameter settings, GRED, and REDD1 are initiated using identical parameters at most. To create congestion and non-congestion scenarios at the buffer, the probability of packet arrival was set to several values; each value tends to create a congestion or non-congestion status. The buffer size room of 20 packets was used to detect congestion at small buffer sizes. The total number of slots used in the experiments was 2000000. This value allows the incorporation of accurate performance measures and encapsulates a period is terminated when the system reaches a steady state. The performance measure results of compared algorithms are obtained by running the algorithm simulations ten times with various random seeds, then taking the mean of the ten results. This ten runs due to remove the bias for any run results. The compared algorithm simulations implemented by a Java environment on a core 2 duo Centrino with 1024 MB RAM.

5.1 Mean queue length, throughput, and delay

Table 3 illustrates the output performances of RED, REDD1, and GREDFL using different probabilities of packet arrivals. The *mql* and *D* results for proposed GREDFL algorithm and all algorithms are identical up to certain value of the probability of packet arrival (e.g., 0.33). In such a low probability value, there is no congestion at their router buffers when the packet arrival probability value is either 0.18 or 0.33. However, when the packet arrival probability value increases such as 0.63, GRED algorithm give marginally small values for *mql* and *D* than either GREDFL or REDD1, additionally GREDFL slightly give smaller performance results than REDD1 with reference to mql and D.

This is due to REDD1 and GREDFL router buffers lose marginally larger number of packets than GRED when congestion occurs (packet arrival probability value = 0.63). In cases where the packet arrival probability value increases to be 0.78, GREDFL offers better *mql* and *D* results than GRED and REDD1. Furthermore, at this probability value of packet arrival, congestion increases, and GREDFL became better than other compared algorithms with regard to *mql* and *D* results since it is stabilized its *mql* and *D* at values lower than those of GRED and REDD1.

Moreover, when the value of packet arrival probability increases to be greater than 0.78 or 0.93, a heavy congestion situation occurs, GREDFL sustains its *mql* and *D* results at values smaller than of those of GRED and REDD1. Consequently, GRED and REDD1 produce slightly higher mql and D results than those of proposed GREDFL when high congestion has occurred. The *T* results under different packet arrival probability values. After analyzing, the *T* of the algorithms give similar *T* results, whether the probability of packet arrival is set to a value lower or higher than the probability of packet departure value. In other words, the algorithms offer similar *T* whether or not a heavy congestion situation has existed.

5.2 Packet loss and dropping probabilities

The proposed GREDFL algorithm is compared with the GRED, REDD1 algorithms in terms of P_L and D_p in this section. The goal of the conducted comparison is to show the quantity of packets loss and dropping at the router buffer in all compared algorithms. The packet loss probability (P_L) is the probability of packet loss due to a buffer overflow, and packet dropping probability (D_p) is the probability of dropping packets before a router buffer has full. The performances of GRED, REDD1 and GREDFL algorithms in terms of P_L and D*p* are shown in Figure 6 and Figure 7, respectively.

In Figure 6, the proposed GREDFL algorithm marginally produces better and least P_L performance results when the probability value of packet arrival is larger than the probability value of packet departure (existence of congestion). This is because the router buffer of GRED and REDD1 overflow more than that of GREDFL's router buffer. Moreover, REDD1 router buffer loses fewer packets than GRED when high congestion has appeared. When the value of packet arrival probability is smaller than the value of packet departure probability, all algorithms provide similar P_L results since either a light congestion or no congestion situation.

Figure 7 shows that the proposed GREDFL algorithm evidently drops more packets at the router buffer than either GRED orREDD1 algorithms when the probability of packet arrival is higher than the probability of packet departure, and this due to GREDFL router buffer loses fewer packets due to overflow than either router buffer of GRED or REDD1. Furthermore, GRED drops fewer packets than REDD1 since GRED loses packets due to overflow larger than those of REDD1.

Table 3. *mql*, *T* and *D* performance results of GRED, EDD1and proposed GREDFL.

	GRED			**REDD1**			**GREDFL**		
α	mql	T	D	mql	T	D	mql	T	D
0.18	0.457	0.1787	2.5604	0.457	0.1893	2.5604	0.4452	0.1858	2.4462
0.33	1.279	0.3277	3.903	1.279	0.3334	3.903	1.2118	0.3262	3.7244
0.48	6.1005	0.4689	13.009	7.299	0.4733	15.395	6.1788	0.4670	13.226
0.63	13.578	0.497	27.299	14.175	0.4996	28.372	14.094	0.4995	28.2131
0.78	14.7936	0.49885	29.6551	14.4143	0.4998	28.8371	14.233	0.4998	28.4723
0.93	14.9456	0.499	29.9316	14.7639	0.4998	29.5363	14.472	0.4998	28.9517

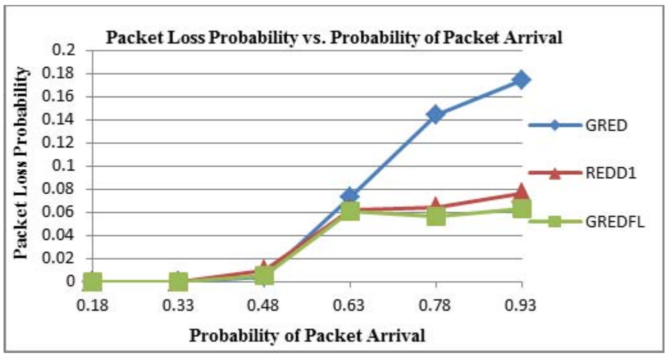


Figure 6. P_L vs. probability of packet arrival.

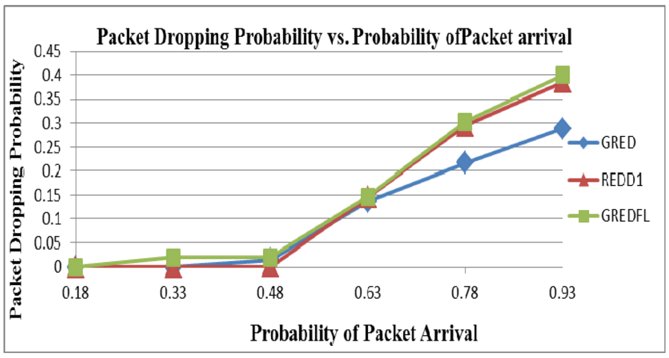


Figure 7. Dp vs. probability of packet arrival.

6. Conclusion

In this paper, an extension to the well-known GRED algorithm based on fuzzy logic controller has been proposed. The purpose of the proposed extension are as follows: First, to obtain a more satisfactory performance with respect to the mean queue length (*mql*) and the average queuing delay (*D*) when a heavy congestion occurs. Second, lose fewer packets by maintain the router buffers protected from overflow when a heavy congestion occurs. Third, eliminate the dependency on parameters setting, as compared to the existing algorithm, relying upon a FIP as a congestion measure.

The results of the proposed algorithm in comparison with GRED and REDD1) show that all the compared algorithms (GRED, REDD1 and the proposed algorithm) provide similar *mql, T* and *D* results when light congestion accrues. Whereas, when the packet arrival probability increases to a value near the packet departure probability value (0.63), GRED algorithm generates marginally better result than those of the REDD1 and the proposed algorithm. Furthermore, if the packet arrival probability value becomes near 0.78 and 0.93, the proposed technique generates slightly better results than those of GRED and REDD1 algorithms regarding *mql* and D. The results also show that, all the compared algorithms offer similar *T* in both congestion and no congestion cases. Generally, the proposed algorithm marginally outperforms the GRED and REDD1 algorithms for P_L when the value of the probability of packet arrival is larger than the value of the probability of packet departure or in the event of heavy congestion. Moreover, GRED and REDD1 drop fewer packets (D_p) at their router buffers than the proposed algorithm at such a case.

Appendix A: Membership Functions

if (InitialDelay>= 0.0 &&InitialDelay<= 2*Capacity / 4) {DelayStatus = 0; // Few} if (InitialDelay> 2*Capacity / 4 &&InitialDelay<= 6 * Capacity / 8) {DelayStatus = 0; // Few}
if (DelayStatus= 0) {
if (InitialDelay>= 0.0 &&InitialDelay<= 2*Capacity / 4) {DelayDegree0 = 1.0;}
if (InitialDelay> 2*Capacity / 4 &&InitialDelay<= 6 * Capacity / 8) {DelayDegree0 = (6 * Capacity / 8 - InitialDelay) / (6 * Capacity / 8 - 2*Capacity / 4); }}
if (InitialDelay> 2*Capacity / 4 &&InitialDelay<= 2*Capacity /2) {DelayStatus1 = 1; // Medium}
if (InitialDelay> 2*Capacity / 2 &&InitialDelay<= 6 * Capacity / 4) {DelayStatus1 = 1; // Medium}
if (InitialDelay> 6 * Capacity / 4 &&InitialDelay<= 14 * Capacity / 8) {DelayStatus1 = 1; // Medium}
if (DelayStatus1 = 1) {
if (InitialDelay> 2*Capacity / 4 &&InitialDelay<= 2*Capacity /2) {DelayDegree1 = ((InitialDelay - 2*Capacity / 4) / (2*Capacity / 2 - 2*Capacity / 4));}
if (InitialDelay> 2*Capacity / 2 &&InitialDelay<= 6 * Capacity / 4) {DelayDegree1 = 1.0;}
if (InitialDelay> 6 * Capacity / 4 &&InitialDelay<= 14 * Capacity / 8) {DelayDegree1 = (14 * Capacity / 8 - InitialDelay) / (14 * Capacity / 8 - 6 * Capacity / 4) ;}}
if (InitialDelay> 3 * Capacity / 2 &&InitialDelay<= 14 * Capacity / 8) {DelayStatus2 = 2; // Alot }
if (InitialDelay> 14 * Capacity / 8 &&InitialDelay<= 2*Capacity) {DelayStatus2 = 2; // Alot}
if (DelayStatus2 == 2) {
if (InitialDelay> 3 * Capacity / 2 &&InitialDelay<= 14 * Capacity / 8) {DelayDegree2 = ((InitialDelay - 3 * Capacity / 2) / (14 * Capacity / 8 - 3 * Capacity / 2));}
if (InitialDelay> 7 * Capacity / 8 &&InitialDelay<= 2*Capacity) {DelayDegree2 = 1.0;}}

Appendix B: Memberships Scenarios for aql

if (AverageQueueLength>= 0.0 &&AverageQueueLength<= Capacity / 5) {AverageQueueLengthStatus0 = 0; // Conservative}
if (AverageQueueLength> Capacity / 5 &&AverageQueueLength<= Capacity / 4) {AverageQueueLengthStatus0 = 0; // Conservative}
if (AverageQueueLengthStatus0 == 0) {
if (AverageQueueLength>= 0.0 &&AverageQueueLength<= Capacity / 5) {AverageQueueLengthDegree0 = 1.0;}
if (AverageQueueLength> Capacity / 5 &&AverageQueueLength<= Capacity / 4)

```
{AverageQueueLengthDegree0 = ((Capacity / 4 - Aver-
ageQueueLength) / (Capacity / 4 - Capacity / 5) ) ; }}
  if (AverageQueueLength>= Capacity / 5
&&AverageQueueLength< Capacity / 3)
{AverageQueueLengthStatus1 = 1; // Medium}
  if (AverageQueueLength>= Capacity / 3
&&AverageQueueLength<= Capacity / 2)
{AverageQueueLengthStatus1 = 1; // Medium}
  if (AverageQueueLength> Capacity / 2
&&AverageQueueLength<= 3 * Capacity / 4)
{AverageQueueLengthStatus1 = 1; // Medium}
  if (AverageQueueLengthStatus1 == 1) {
  if (AverageQueueLength>= Capacity / 5
&&AverageQueueLength< Capacity / 3)
{AverageQueueLengthDegree1 = ( (AverageQueueLength -
Capacity / 5) / (Capacity / 3 - Capacity / 5) );}
  if (AverageQueueLength>= Capacity / 3
&&AverageQueueLength<= Capacity / 2)
{AverageQueueLengthDegree1 = 1.0;}
  if (AverageQueueLength> Capacity / 2
&&AverageQueueLength<= 3 * Capacity / 4)
{AverageQueueLengthDegree1 = ( (3 * Capacity / 4 - Aver-
ageQueueLength) / (3 * Capacity / 4 - Capacity / 2) );}}
  if (AverageQueueLength>= (0.7 * Capacity)
&&AverageQueueLength< 3 * Capacity / 4)
{AverageQueueLengthStatus2 = 2; // Aggressive}
  if (AverageQueueLength>= 3 * Capacity / 4
&&AverageQueueLength<= Capacity)
{AverageQueueLengthStatus2 = 2; // Aggressive}
  if (AverageQueueLengthStatus2 == 2) {
  if (AverageQueueLength>= (0.7 * Capacity)
&&AverageQueueLength< 3 * Capacity / 4)
{AverageQueueLengthDegree2 = ( (AverageQueueLength -
0.7 * Capacity) / (3 * Capacity / 4 - 0.7 * Capacity) ) ; }
  if (AverageQueueLength>= 3 * Capacity / 4
&&AverageQueueLength<= Capacity)
{AverageQueueLengthDegree2= 1.0;}}
```

References

[1] D. Lin and R. Morris, "Dynamics of random early detection," in *ACM SIGCOMM*, New York, pp. 127-137, 1997.

[2] T. O. Lakshman et al., "SRED: Stabilized RED," in *IEEE INFOCOM*, pp. 1346-1355, 1999.

[3] G. Thiruchelvi and J. Raja, "A Survey on active queue management mechanisms," *International Journal of Computer Science and Network Security*, vol. 8, 2008.

[4] Welzl M., *Network congestion control: managing internet traffic*, John Wiley& Sons, August, 2005.

[5] A. S. Tanenbaum, *Computer Networks*, 4th ed.: Prentice Hall Ptr, 2002.

[6] C. Brandauer, G. Iannaccone, C. Diot, T. Ziegler, S. Fdida, and M. May, "Comparison of tail drop and active queue management performance for bulk-data and Web-like Internet," in *IEEE ISCC*, pp. 122-129,2001.

[7] R. Stanojevic, R. N. Shorten, and C. M. Kellet, "Adaptive tuning of drop-tail buffers for reducing queuing delays," *IEEE Communications Letters*, vol. 10, pp. 570-572, 2006.

[8] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. on Networking*, pp. 397-413, 1993.

[9] S. Athuraliya, S. H. Low, V. H. Li, and Q. Yin, "REM: active queue management," *IEEE Network*, vol. 15, pp. 48-53, 2001.

[10] J. Aweya, M. Ouellette, and D. Y. Montuno, "A control theoretic approach to active queue management," *Computer Networks*, vol. 36, pp. 203-235, 2001.

[11] A. M. Murshid, S. A. Loan, S. A. Abbasi, and A. R. M. Alamoud, "A novel VLSI architecture for a fuzzy inference processor using triangular-shaped membership function," *International Journal of Fuzzy Systems*, 2012.

[12] M.-S. Kim and S.-G. Kong, "Parallel structure fuzzy systems for time series prediction," *International Journal of Fuzzy Systems*, vol. 3, 2001.

[13] W. J. Chang, P. H. Chen, and C. T. Yang, "Robust fuzzy congestion control of TCP/AQM router via perturbed takagi-sugeno fuzzy models," *International Journal of Fuzzy Systems*, vol. 15, 2013.

[14] H. Abdel-jaber, M. Mahafzah, F. Thabtah, and M. Woodward, "Fuzzy logic controller of random early detection based on average queue length and packet loss rate," *Performance Evaluation of Computer and Telecommunication Systems, SPECTS. International Symposium, the Proceeding of the IEEE Explorer*, Edinburgh, UK, pp. 16-18, 2008.

[15] S. Ghosh, Q. Razouqi, H. J. Schumacher, and A. Celmins, "A survey of recent advances in fuzzy logic in telecommunications networks and new challenges," *IEEE Trans. on Fuzzy Systems*, vol. 6, pp. 443-447, 1998.

[16] G. J. Klir, "Fuzzy logic," presented at the *Potentials IEEE*, 1995.

[17] M. Negnevitsky, *Artificial intelligence*, Second ed., 2005.

[18] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338-353, 1965.

[19] M. Black, "Vagueness: An exercise in logical analysis," *Philosophy of Science*, vol. 4, pp. 427-455, 1990.

[20] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, pp. 1-13, 1975.

[21] S. Floyd. *Recommendations on using the gentle*

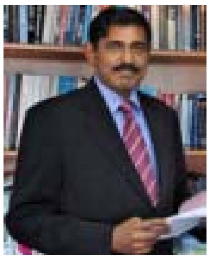
variant of RED, 2000. Available: http://www.aciri.org/floyd/red/gentle.html.
[22] H. Abdel-jaber, F. Thabtah, A. M. Daoud, J. Ababneh, and M. Baklizi, "Performance investigations of some active queue management techniques using simulation," *International Journal on New Computer Architectures and Their Applications*, vol. 2, 2012.
[23] J. Ababneh, H. Abdel-Jaber, F. Thabtah, W. Hadi, and E. Badarneh, "Derivation of three queue nodes discrete-time analytical model based on DRED algorithm," *7th International Conference on Information Technology: New Generations (ITNG)*, pp. 885-890, 2010.
[24] H. Abdel-Jaber, M. E. Woodward, F. A. Thabtah, and M. Al-Diabat, "Modelling BLUE active queue management using discrete-time queue," *Proceedings of the International Conference of Information Security And Internet Engineering (ICISIE'07)*, London, pp. 568-573, 2007.
[25] H. Abdel-Jaber, M. Woodward, F. Thabtah, and A. Abu-Ali, "Performance evaluation for DRED discrete-time queuing network analytical model," *Journal of Network and Computer Applications*, vol. 31, pp. 750-770, 2008.
[26] 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*, Aug. 1, 2001.
[27] C. Wang, B. Li, K. Sohraby, and Y. Peng, "AFRED: An adaptive fuzzy-based control algorithm for active queue management," *Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks (LCN'03)*, IEEE Computer Society, Washington, DC, USA, 2003, pp. 12-20.
[28] C. Chrysostomou, A. Pitsillides, L. Rossides, and A. Sekercioglu, "Fuzzy logic controlled RED: congestion control in TCP/IP differentiated services networks," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 8, pp. 79-92, 2003.
[29] C. Chrysostomou, A. Pitsillides, G. Hadjipollas, A. Sekercioglu, and M. Polycarpou, "Fuzzy explicit marking for congestion control in differentiated services networks," *Proceedings of the 8th IEEE International Symposium on Computers and Communication (ISCC'03)*, pp. 312-319, 2003.
[30] C. Chrysostomou, A. Pitsillides, L. Rossides, M. Polycarpou, and A. Sekercioglu, "Congestion control in differentiated services networks using Fuzzy-RED," *IFAC Control Engineering Practice (CEP) Journal*, 2003.
[31] R. Loukas, S. Kohler, P. Andreas, and T. G. Phuoc, "Fuzzy RED: congestion control for TCP/IP Diff-Serv," *Proceeding of 10th Mediterranean Electrotechnical Conference, MEleCon*, pp. 19-22, 2000.
[32] S. Dick, "Toward complex fuzzy logic," *IEEE Trans. on fuzzy systems*, vol. 13, pp. 405-414, 2005.
[33] P. Hajek, "Metamathematics of fuzzy logic," *Computational Intelligence: An Introduction*, 1998.
[34] S. Tasaka, *Performance analysis of multiple access protocol*, MIT Press, April, 1986.
[35] D. d. Van, *Logic and Structure*, Berlin: Springer, 1994.
[36] M. E. Woodward, *Communication and computer networks: Modeling with discrete-time queues*, London: IEEE Computer Society Press (Los Alamitos, Calif.), 1994.

Mahmoud Khalid Baklizi is a researcher pursuing his Ph.D. in Computer Science at the National Advanced IPv6 Center of Excellence in University Sains Malaysia. He received his first degree in Computer Science from Yarmouk University, Jordan, 2002 and his Master degree in Computer Information System from the Arab Academy for Banking and Financial Sciences, Jordan in 2008. His research area of interest includes Congestion control.

Hussein Abdel-Jaber is a head of departments of Computer Information Systems (CIS). He received his first degree in Computer Science from Philadelphia University, Jordan, 2003 and his Master degree in Mobile Computing from the University of Bradford, UK in2004. His research area of interest includes Congestion control.

Ahmad Abu Shreha is a researcher in departments of Computer Information Systems (CS). He received his first degree in Computer Science from Al Al-Bayt University (AABU), Jordan, 2004 and his Master degree from Universiti Sains Malaysia (USM) – Malaysia, 2006. And his phd degree from Universiti Sains Malaysia (USM) – Malaysia, 2012.

Mosleh Mohammad Salem Abualhaj is a researcher in Faculty of Information Technology, Assistant Professor in Ahlya Amman University. He received his first degree Philadelphia University, Jun-2004, Computer and Computer Information Systems, and his Master degree from The Arab Academy for Banking and Financial Sciences, Jul-2007, Computer Information Systems. And his Ph.D. degree from Universiti Sains Malaysia (USM) – Malaysia, 2012.



Sureswaran Ramadass is a Professor and the Director of the National Advanced IPv6 Centre (NAV6) at Universiti sains Malaysia. Dr. Sureswaran obtained his BsEE/CE (Magna Cum Laude) and Masters in Electrical and ComputerEngineering from the University of Miami in 1987 and 1990 respectively. He obtained his Ph.D. from Universiti Sains Malaysia (USM) in 2000 while serving as a full time faculty in the School of Computer Sciences. Dr. Sureswaran's recent achievements include being awarded the Anugerah Tokoh Negara (National Academic Leader) for Innovation and Commercialization in 2008 by the Minister of Science and Technology. He was also awarded the Malaysian Innovation Award by the Prime Minister in 2007. Dr. Sureswaran is also the founder and headed the team that successfully took Mlabs Systems Berhad, a high technology video conferencing company to a successful listing on the Malaysian Stock Exchange in 2005. Mlabs is the first, and so far, only university based company to be listed in Malaysia.