Chapter Five

Fuzzy Logic Controller Algorithms Based on RED

5.1 Introduction

In the last few decades, researchers have concentrated on controlling the congested router buffers in fixed networks such as the internet [66, 67, 68, 73, 100, 117, 120]. Normally, the internet does its best to deliver packets as fast as possible to their destinations (best effort) [77, 117], but there is no guarantee that these packets will safely arrive. Further, the internet relies on the router buffers to queue packets in order to serve them. When the queue at a particular router buffer builds up due to the arrival of several packets, it starts to drop some arriving packets probabilistically to reduce the congestion.

As discussed in Chapter 2, numerous researchers have proposed different AQM methods to overcome the congestion problem in wired networks. For instance, RED was developed to alleviate congested router buffers in the internet [46]. Nevertheless, as mentioned in Subsection 2.4.3.2, RED suffers from drawbacks such as the tuning of parameters (qw, $D_{\max}$, $\min threshold$ and $\max threshold$), where RED must tune its parameters to specific values to obtain a satisfactory performance. To overcome some of RED's limitations, many researchers have developed congestion control methods based on RED such as GRED [50], ARED [49] and SRED [90].

In the past few years, some researchers have adopted Fuzzy Logic (FL) controller techniques based on different AQM methods because of the difficulty of designing new AQM approaches [105, 122]. For example, in order to enhance the BLUE algorithm performance, [122] proposed a FL technique based on the BLUE algorithm [39, 43] where the authors used two input linguistic variables (current queue length and packet loss rate) to obtain a single output linguistic variable (packet dropping probability). The experimental results obtained in [122] revealed that their technique performs better than the original BLUE with reference to queue length, throughput and packet loss rate.

Moreover, the authors of [27, 28, 29, 76] designed FL techniques based on RED within TCP/IP differentiated service networks. They used multiple classes of service, and different linguistic rules for each class of service. They also employed two input linguistic variables (current queue length, the change rate in the traffic load) in order to produce a single output linguistic variable (packet dropping probability). The experimental results of [27, 28, 29, 76] have shown that their techniques outperform RED with regard to throughput and the queue size performances. Moreover, [116] developed a FL congestion control algorithm called Adaptive Fuzzy RED (AFRED) that depends on a single input linguistic variable (current queue length) to derive a single output variable (dropping probability). The simulation results of [116] showed that AFRED outperforms RED in terms of the throughput and in maintaining the queue length as small as possible.

In this chapter, three AQM algorithms based on FL and RED are developed to achieve the following aims:

1) Provide a more satisfactory performance measure results than the RED algorithm with respect to mean queue length (mql) and average queueing delay (D) when a high congestion is present
2) Lose fewer packets due to buffer overflow than the RED algorithm when a high congestion exists.
3) Decrease the RED algorithm dependency on its parameters, i.e. $\min threshold$, $\max threshold$. So the proposed FL algorithms will not depend on $\min threshold$ and $\max threshold$ as congestion boundaries, rather they will rely on a fuzzy inference process (FIP) as a congestion measure.

The proposed methods are similar to an expert system developed to model the human experience in decision making [55, 93]. In the developed FL methods, two linguistic rules called error and trial [85, 122] and theory [85, 122] are employed to evaluate the input and output linguistic variables. The FL methods discussed in this chapter rely on a FIP to find the packet dropping probability in four steps. These steps are: 1) The fuzzification of the input crisp values for the input linguistic variables, 2) The evaluation of the rules, 3) The aggregation of the output rules into a single fuzzy set, and 4) The calculation of the output crisp value for each output linguistic variable (defuzification). Subsection 5.2.3 discusses these steps further.

The proposed methods are based on the RED algorithm. The first method is called REDD1 and it utilises two input linguistic variables (overflow packet loss probability, average queue length). The second method is called REDD2 and it uses two input linguistic variables (throughput, average queue length), and the last method is called REDD3 and it uses three input linguistic variables (overflow packet loss probability, throughput, average queue length). In all the proposed techniques, the output linguistic variable is the packet dropping probability. The proposed REDD1, REDD2 and REDD3 algorithms use an average queue length as an input linguistic variable rather than the instantaneous queue length since they are developed based on the RED algorithm [46].

This chapter is structured as follows: Section 5.2 gives a review on FL, the proposed algorithms (REDD1, REDD2 and REDD3) are described in Sections 5.3, 5.4 and 5.5, respectively. Section 5.6 is devoted to the performance evaluation and statistical analyses of the FL methods and RED with respect to several performance measures. Lastly, the chapter summary is presented in Section 5.7.

5.2 Fuzzy Logic (FL)

Fuzzy logic (FL) can be defined as a set of mathematical expressions for knowledge representation with reference to membership values (also called membership degrees) [3, 23, 71, 76, 89, 109, 125]. Additionally, FL is a subset of computational intelligence, an active research field that deals with processing numerical data [17, 85, 124]. Unlike the classical binary logic (CBL) [123], which depends solely on two values (0, 1) and do not accepts values between 0 and 1, FL relies on multiple values between 0 and 1 including both 0 and 1. Since FL accepts values between 0 and 1, these values can be partly true or partly false. The CBL deals only with 0 and 1, where these values are also accepted in FL, thus the CBL is a special case of FL.

5.2.1 Fuzzy Sets

A fuzzy set is a field in mathematics [72, 88]. Consider *S* as a CBL set that contains *s* elements. An element *s* either belongs to $S\ (s \in S)$ or does not belong to $S\ (s \notin S)$, and this CBL set is often named the classical set theory (CST). All elements have "1" membership value if they belong to CST or "0" otherwise, and therefore the CST deals only with (0, 1). However, in the FL set, the ultimate aim is not whether *s* belongs to the FL set *S*, rather *s* may be partly 0 or partly 1 (a floating value between [0, 1]), and this *S* is called the Fuzzy Set Theory (FST) [33, 87]. One can declare the fuzzy set as a set that has fuzzy boundaries and its contents are obtained from a domain expert [85]. Also, the fuzzy set can be defined as a set that determines the boundaries of linguistic values on the universe of discourse (x-axis).

As example, consider a weight that consists of five sets, i.e. "very skinny", "skinny", "average", "fat", and "very fat". Using the CST, if the element belongs to the "very skinny" binary set, it is understood that its membership value would equal 1, and the membership values for the other binary sets are zeros. In other words, in the binary set theory, each element belongs to just a single binary set, whereas, in the FST, an element could belong to multiple fuzzy sets (partly true for some fuzzy sets and partly false for the others). Furthermore, the degree of membership for this element in each fuzzy set lies in the range [0.0, 1.0]. Now going back to the example, Figure 5.1 shows the five CST binary sets. The weights for the "very skinny", and the "skinny" binary sets are [0-45], [46-60], respectively. Further, the weights of the "average" and the "fat" binary sets are [61-85], [86-115], respectively. Lastly, the weights of the "very fat" binary set are [116-150].

On the other hand and according to the FST, the five fuzzy sets are obtained as follows: The elements of "very skinny" fuzzy set fall in the range [0-50]. The "skinny" fuzzy set elements are represented in the interval [40-60], and the elements of "average" fuzzy set are between 50 and 90. Finally, the "fat" and "very fat" fuzzy sets elements are between [80-120] and [100-150], respectively. The fuzzy sets for the weight example are depicted in Figure 5.2.

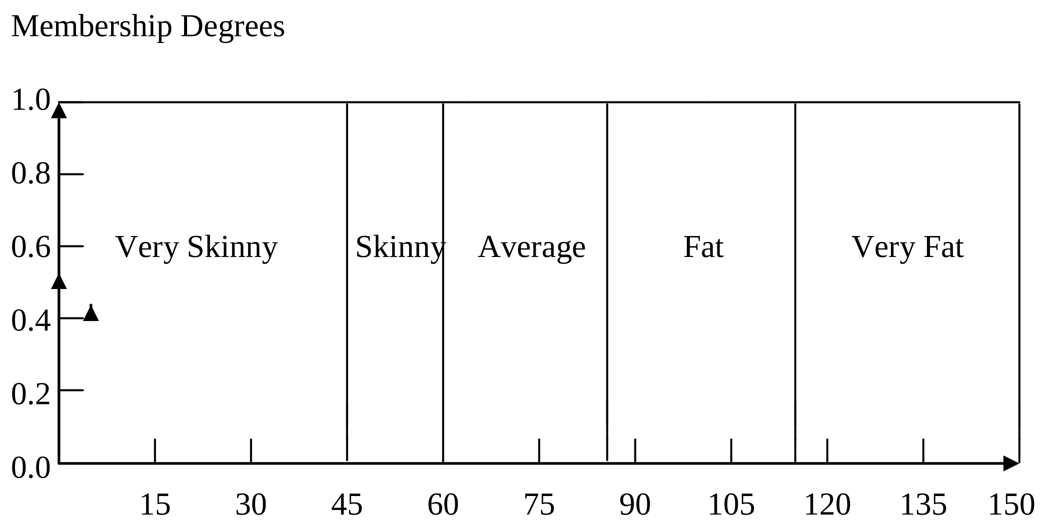

Figure 5.1: The binary sets for CST.

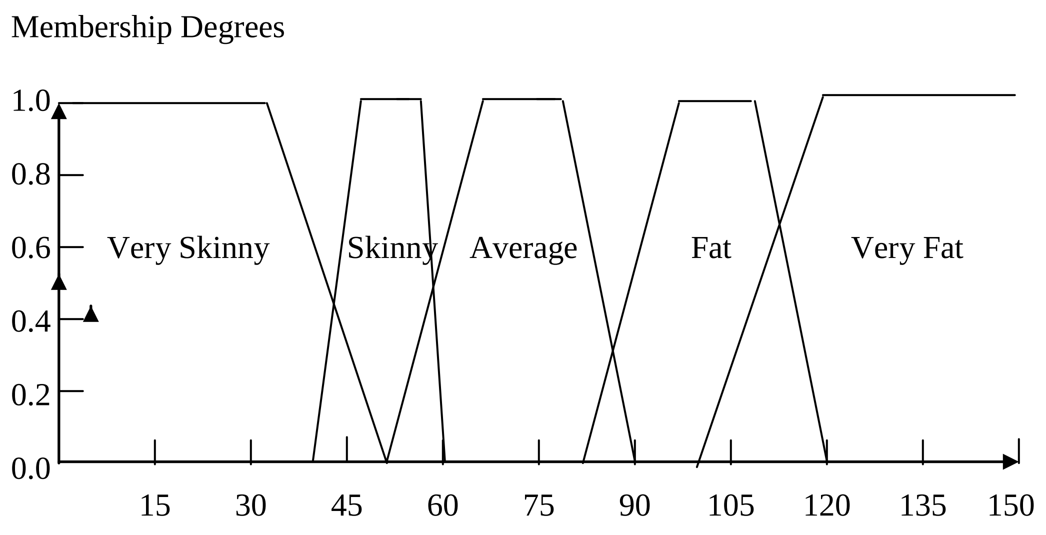

Figure 5.2: The fuzzy sets for FST.

Now, consider a person who weight’s 59 kilograms and belongs to the "skinny" binary set according to Figure 5.1, this leads us to say this person is skinny. If one picks the weight 61 from the same figure, it can be noted this weight belongs to the "average" binary set. This means that a person who weight 61 is classified as an "average" weight person. However, it is abnormal that two weights with just 2 kilograms difference are classified into two different classes. In other words, it is irregular to say that a 59 kilograms person is skinny and a 61 kilograms person is an average one. In order to resolve such issues, the fuzzy sets, displayed in Figure 5.2, are utilised to classify weights. According to Figure 5.2, a person who is 59 kilograms could be partly skinny and partly average. The weight 59 in the "skinny" and the "average" fuzzy sets has 0.6 and 0.475 membership degrees, respectively. Further, the 61 weight belongs solely to the "average" fuzzy set and its membership degree equals to 0.675.

Formally and according to [93], let *S* be the x-axis that has the entire elements *s*, it is also called the universe of discourse. Further, it is assumed that *B* is a binary set in *S*. The function that calculates the binary values in CST is called the characteristic function [85] and is defined in equation (5.1).

$F_B(s): S \to 0,1$, where

$$F_B(s) = \begin{cases} 0, if & s \notin B \\ 1, if & s \in B \end{cases} \quad \text{.......................................} \quad (5.1)$$

According to the FST, let *B* denotes a fuzzy set in the universe of discourse *S*, the membership function employed to identify the *B* fuzzy set is defined in equation (5.2).

$F_B(s): S \to [0,1]$, where... (5.2)

$F_B(s) = 0$ if s does not belong to B.

$F_B(s) = 1$ if s completely belongs to B.

$1 > F_B(s) > 0$ if s partly belongs to B.

5.2.2 Fuzzy Rules

A fuzzy rule can be identified as a conditional statement in the form of "IF c is C THEN o is O". "c" and "o" are denoted the linguistic variables and both are used in the IF-part (antecedent) and THEN-part (consequent), respectively. "C" and "O" represent the linguistic values that are derived from the fuzzy sets based on the universe of discourse. Each linguistic variable has its own universe of discourse, and for every universe of discourse there are fuzzy sets, i.e. 0-150 in the example of Figure 5.2. It should be noted that both the antecedent and the consequent parts may contain multiple conditions. Figures 5.3 and 5.4 exhibit an example for the antecedent and the consequent parts, respectively.

IF delay is Low and service is great THEN no. of customers is large.

IF delay is high and service is bad THEN no. of customers is low.

Figure 5.3: The fuzzy rules with multiple antecedent parts.

IF restaurant is full THEN food is delicious and service is excellent.

Figure 5.4: The fuzzy rules with multiple consequent parts.

After the input and output linguistic variables with their fuzzy sets on the universe of discourse are specified by the domain expert. The choice of the fuzzy linguistic rules is done through using designing methods such as error and trial and theory [85, 122]. The error and trial method relies upon the experience, and knowledge of the domain expert. After a domain expert had introduced the fuzzy linguistic rules, he/s examined whether the rules are suitable or not to the system. If they are suitable to the system then the expert accepts them, otherwise, he/s may reject them. The rules selection process will repeat again until all suitable rules are given. On the other hand, the theory method depends on tuning the ultimate parameters of the system such as packet loss rate, throughput, etc, by a domain expert. The combination of the error and trial and theory designing methods may produce better fuzzy linguistic rules than either utilising just a single method.

5.2.3 Fuzzy Inference Process

Fuzzy inference is the process of planning to assign inputs in order to achieve outputs utilising the FST [3, 56, 78]. The most well known fuzzy inference method is Mamdani [78], which is implemented in four steps.

1) Fuzzification of the input variables.

2) Evaluation of the rules.

3) Aggregation all the output rules into a single fuzzy set.

4) Defuzification.

The details of these steps are given in the next four subsections.

5.2.3.1 Fuzzification of the Input Variables

In the fuzzification step, the input crisp values are obtained for the input linguistic variables 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.

5.2.3.2 Evaluation of the Rule

In this step, the fuzzified input variables are given using their membership degrees and can be evaluated by applying them on the antecedent part of the rule. After the antecedent part is processed, the consequent part 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 [35, 85]. Then based on the results of the antecedent rules, the membership degree for every output linguistic rule is achieved.

5.2.3.3 Aggregating All the Output Rules into a Single Output Rule (Fuzzy Set)

In this step, the degree of membership for each consequent rule part is obtained, and then the combination between them into a single output rule is done. 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.

5.2.3.4 Defuzification

This is the final step of the fuzzy inference process, which generates a crisp value for each output linguistic variable. The input for this step is a fuzzy set for every output linguistic variable and the output of this step is a crisp value for each output linguistic variable. One of the popular defuzification techniques is the centre of gravity (COG) method [57, 93], which aims to find out the point located on the centre of the aggregate fuzzy set for each output linguistic variable. Formally, the COG can be defined according to equation (5.3) [85], for further information refer to [85].

$$COG = \frac{\sum_{a}^{b} F_B(s) \times s}{\sum_{a}^{b} F_B(s)} \quad \text{.........} \quad (5.3)$$

5.3 A Proposed Fuzzy Logic Controller Based On Packet Loss Probability and Average Queue Length

This section introduces the first fuzzy logic controller (FLC) method called (REDD1), which adopts the RED strategy in calculating the aql. The algorithm is based on two input linguistic variables (aql, packet loss probability (P_L)) that are used to compute a single output linguistic variable (D_P). Each linguistic variable in the FLC is associated with fuzzy sets. Figure 5.5 depicts the fuzzy sets for the input and the output linguistic variables, respectively. The fuzzy sets for each linguistic variable are selected based on the behaviour of their linguistic variable, for example the P_L input linguistic variable could be few, and this means few packets were lost due to router buffer overflow. Either $medium$ or $alot$

represent a medium or a large number of packets were lost. So few, $medium$ and $alot$ are the behaviours of the P_L.

The aims of the REDD1 algorithm are:

1. Provide a more satisfactory performance measure results than the RED algorithm with respect to mean queue length (mql) and average queueing delay (D) when a high congestion is present.

2. Lose fewer packets due to overflow than the RED algorithm when a high congestion occurs.

3. Decrease the dependency of the RED algorithm on its parameters, i.e. $\min threshold$, $\max threshold$. So the proposed REDD1 algorithm does not depend on $\min threshold$ and $\max threshold$ as congestion boundaries, rather it relies on a FIP.

$aql = \{$ conservative, middle, aggressive $\}$.

$P_L = \{$ few, medium, a lot$\}$.

$D_P = \{$ zero, low, moderate, high $\}$.

Figure 5.5: The fuzzy sets of input and output linguistic variables of the REDD1 algorithm.

Every fuzzy set has a range from the universe of discourse that clarifies its boundaries. The optimum choice of set boundaries is not considered here and would be a topic for further investigation. The assumption of the membership functions of the aql and the P_L input linguistic variables are shown in Figures 5.6 and 5.7, respectively. Moreover, Figure 5.8 displays the membership function of the D_P output linguistic variable. The assumption of membership functions for the P_L and D_P linguistic variables are similar to those in [122], the boundaries of membership functions and fuzzy sets are chosen by domain experts in both FL and congestion control fields [85]. The consideration of a membership function for the aql linguistic variable is given as follows: The 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 5.6, 5.7 and 5.8 are either trapezoidal or triangular for simple computations [85, 122].

After the fuzzy sets and their elements are presented, one can build the fuzzy rules. To build the fuzzy rules shown in Figure 5.9, 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 person who builds the fuzzy linguistic rules is a domain expert who utilises his knowledge and experience in the congestion control and FL fields. Details about building the fuzzy linguistic rules have been explained in Subsection 5.2.2.

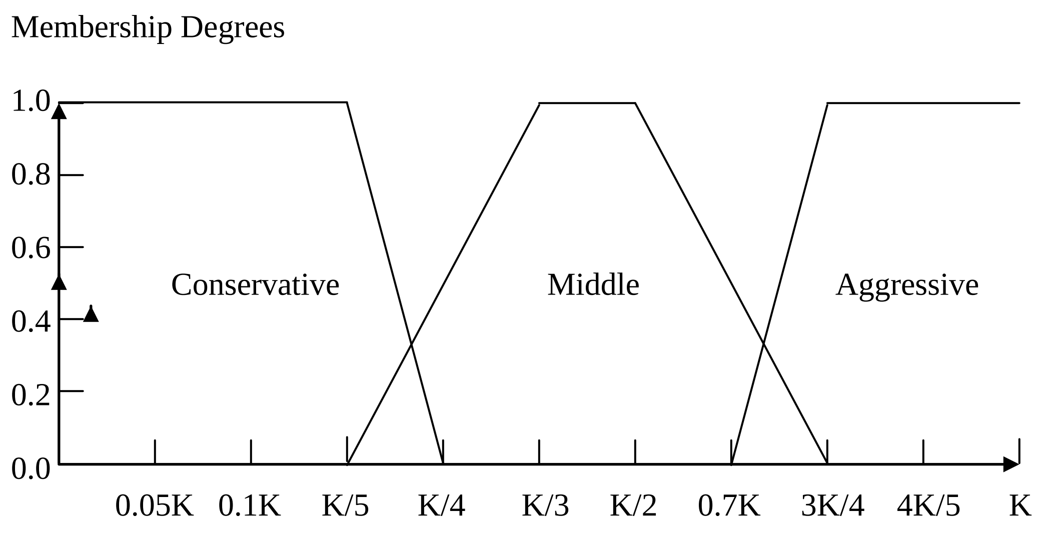

Figure 5.6: The membership function of aql , where K represents the system capacity .

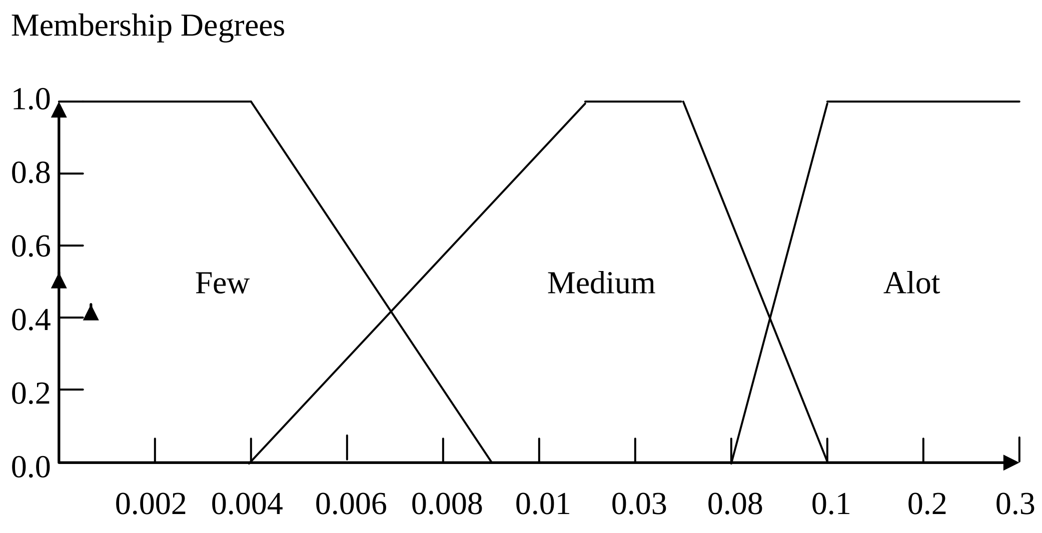

Figure 5.7: The membership function of P_L .

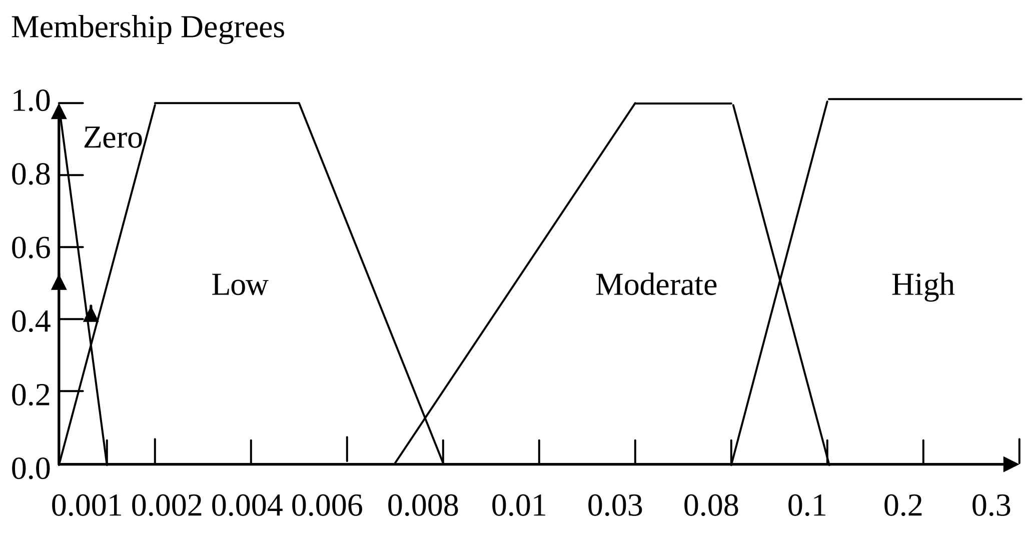

Figure 5.8: The membership function of D_P.

IF aql is conservative and P_L is few THEN D_P is $zero$.

IF aql is conservative and P_L is $medium$ THEN D_P is $zero$.

IF aql is conservative and P_L is a lot THEN D_P is $zero$.

IF aql is $middle$ and P_L is few THEN D_P is $zero$.

IF aql is $middle$ and P_L is $medium$ THEN D_P is $zero$.

IF aql is $middle$ and P_L is a lot THEN D_P is low.

IF aql is $aggressive$ and P_L is few THEN D_P is $zero$.

IF aql is $aggressive$ and P_L is $medium$ THEN D_P is moderate.

IF aql is $aggressive$ and P_L is a lot THEN D_P is $high$.

Figure 5.9: The linguistic fuzzy rules of the proposed REDD1 algorithm based on aql and P_L.

Figure 5.9 highlights that each fuzzy rule consists of two parts, the rule antecedent, and the rule consequent. In the REDD1 method, the combination of the theory and trail and error methods are used to construct the fuzzy linguistic rules used during the process of obtaining the D_P results. Furthermore, the main component of the REDD1 algorithm is the FIP, which is implemented at each router queue. The FIP employs two input variables (aql , P_L) to output a single output variable (D_P). Whenever a packet arrives at the router queue in this algorithm, the FIP uses as congestion detector and controller at the router queues to derive the D_P result through four steps. The first step is fuzzification in which the FIP takes the input crisp values of the aql and P_L 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 [93, 112, 114, 122], the output crisp value for the D_P is calculated.

Figure 5.9 indicates that if the aql is in a conservative fuzzy set, whatever the fuzzy set that the P_L 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 P_L is in either few or medium fuzzy set, the D_P will be in a zero fuzzy set. However, if the P_L 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 P_L fuzzy set. Hence, if the P_L is in a few fuzzy set, then the D_P is in a zero fuzzy set, whereas if the P_L is in medium and a lot fuzzy sets, then the D_P is in moderate and high fuzzy sets, respectively.

5.4 A Proposed Fuzzy Logic Controller Based on Throughput and Average Queue Length

In this section, a FLC method based on the RED algorithm called REDD2 is presented. This method depends on two input linguistic variables (aql ,T) to generate a single output linguistic variable (D_P). The aims of developing this FLC method are similar to those of REDD1. REDD2 uses REDD1's strategy in computing the aql , and has the same input and output linguistic variables (aql , D_P) of REDD1. Therefore, the fuzzy sets of aql and D_P for REDD1 are also used by the REDD2 algorithm. The fuzzy sets of the aql and the D_P were given as in Section 5.3 (Figure 5.5), and the fuzzy set of the T variable is displayed in Figure 5.10. Moreover, the membership functions of the aql and the D_P were given in Section 5.3 (Figures 5.6 and 5.8), and the membership function of the T input linguistic variable is shown in Figure 5.11. The diagrams of the T membership function are similar to those of the aql and the D_P (Triangular and trapezoidal). The member function of the T is considered by a domain expert in both FL and congestion control fields, and presented as follows: The T can be in a small fuzzy set when its value is between 0.0 and 0.6. While the T is in a medium fuzzy set when its value is between 0.4 and 0.8. Lastly, when the T value is equal to or greater than 0.8, the T will be in a big fuzzy set. The T membership function with its values seems logical. For instance, assume T=0.3, this T value is a small and it is logical to be in a small fuzzy set. Additionally, consider T=0.7 then 0.9, it is logical that the T is a medium then a big, respectively. Therefore, the T is in a medium fuzzy set then in a big fuzzy set. Further, the optimal selection of membership function for T is not considered here and can be a topic for further research. As discussed previously in Section 5.3, the fuzzy linguistic rules for the REDD2 method are depicted in Figure 5.12.

$$T = \{small, medium, big\}.$$

Figure 5.10: The fuzzy sets of the T linguistic variable in the REDD2 algorithm.

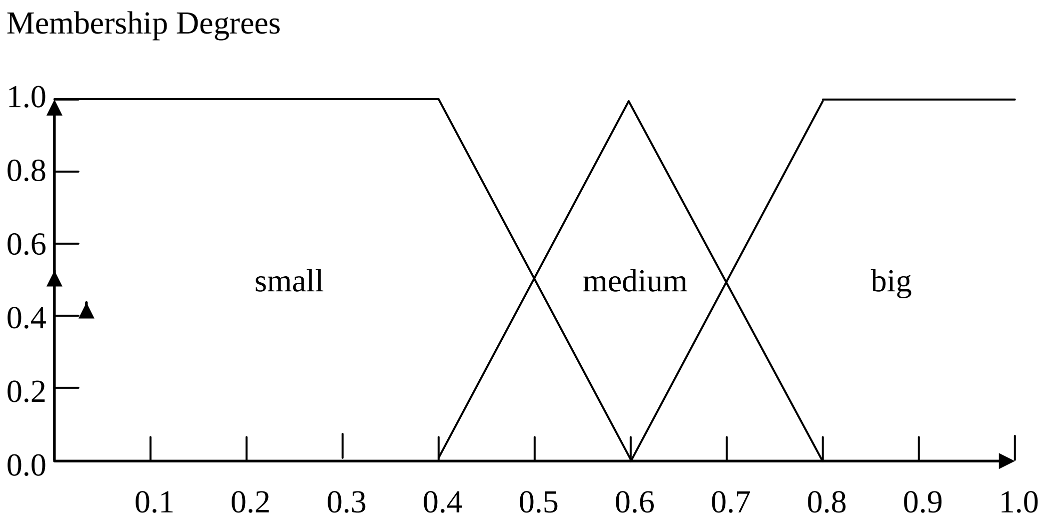

Figure 5.11: The membership function of T.

Similar to the REDD1, the FIP plays a major role in discovering and controlling the congested router buffers in the REDD2 method. Thus, the FIP can be applied at the REDD2 router buffer. The FIP uses both the aql and the T as input linguistic variables aiming to produce an output crisp value for the D_P (output linguistic variable). Finally, REDD2 uses the same strategy (FIP four steps) employed by REDD1 and discussed at the end of Section 5.3 to derive the crisp value for the output linguistic variable (D_P).

IF aql is conservative and T is $small$ THEN D_P is $zero$.

IF aql is conservative and T is $medium$ THEN D_P is $zero$.

IF aql is conservative and T is big THEN D_P is $zero$.

IF aql is $middle$ and T is $small$ THEN D_P is low.

IF aql is $middle$ and T is $medium$ THEN D_P is $zero$.

IF aql is $middle$ and T is big THEN D_P is $zero$.

IF aql is $aggressive$ and T is $small$ THEN D_P is $high$.

IF aql is $aggressive$ and T is $medium$ THEN D_P is moderate.

IF aql is $aggressive$ and T is big THEN D_P is $zero$.

Figure 5.12: The linguistic fuzzy rules of the presented REDD2 technique based on aql and T.

5.5 A Proposed Fuzzy Logic Controller Based on Combined Packet Loss Probability, Throughput and Average Queue Length

The method presented in this section, i.e. REDD3, combines both the REDD1 and the REDD2 methods. REDD3 utilises three input linguistic variables (P_L, T, aql) to produce a single output linguistic variable (D_P). REDD3 computes the aql in the same way as REDD1 and REDD2 and its fuzzy sets with their boundaries on the universe of discourse are also similar to those of REDD1 and REDD2. Furthermore, the membership functions for the input and the output linguistic variables for the REDD3 algorithm are displayed in Figures 5.6, 5.7, 5.8 and 5.11. The fuzzy linguistic rules shown in Figure 5.13 of the REDD3 are built the same way as in Section 5.3. The fuzzy linguistic rules for the REDD3 algorithm are designated by merging both REDD1 and REDD2 fuzzy linguistic rules together as follows:

IF aql is conservative and T is $small$ and P_L is few THEN D_P is $zero$.
IF aql is conservative and T is $small$ and P_L is $medium$ THEN D_P is $zero$.
IF aql is conservative and T is $small$ and P_L is a lot THEN D_P is $zero$.
IF aql is conservative and T is $medium$ and P_L is few THEN D_P is $zero$.
IF aql is conservative and T is $medium$ and P_L is $medium$ THEN D_P is $zero$.
IF aql is conservative and T is $medium$ and P_L is a lot THEN D_P is $zero$.
IF aql is conservative and T is big and P_L is few THEN D_P is $zero$.
IF aql is conservative and T is big and P_L is $medium$ THEN D_P is $zero$.

IF aql is conservative and T is big and P_L is a lot THEN D_P is $zero$.

IF aql is $middle$ and T is $small$ and P_L is few THEN D_P is $zero$.

IF aql is $middle$ and T is $small$ and P_L is $medium$ THEN D_P is low.

IF aql is $middle$ and T is $small$ and P_L is a lot THEN D_P is moderate.

IF aql is $middle$ and T is $medium$ and P_L is few THEN D_P is $zero$.

IF aql is $middle$ and T is $medium$ and P_L is $medium$ THEN D_P is $zero$.

IF aql is $middle$ and T is $medium$ and P_L is a lot THEN D_P is low.

IF aql is $middle$ and T is big and P_L is few THEN D_P is $zero$.

IF aql is $middle$ and T is big and P_L is $medium$ THEN D_P is $zero$.

IF aql is $middle$ and T is big and P_L is a lot THEN D_P is $zero$.

IF aql is $aggressive$ and T is $small$ and P_L is few THEN D_P is low.

IF aql is $aggressive$ and T is $small$ and P_L is $medium$ THEN D_P is moderate.

IF aql is $aggressive$ and T is $small$ and P_L is a lot THEN D_P is $high$.

IF aql is $aggressive$ and T is $medium$ and P_L is few THEN D_P is low.

IF aql is $aggressive$ and T is $medium$ and P_L is $medium$ THEN D_P is moderate.

IF aql is $aggressive$ and T is $medium$ and P_L is a lot THEN D_P is $high$.

IF aql is $aggressive$ and T is big and P_L is few THEN D_P is $zero$.

IF aql is $aggressive$ and T is big and P_L is $medium$ THEN D_P is $zero$.

IF aql is $aggressive$ and T is big and P_L is a lot THEN D_P is $zero$.

Figure 5.13: The linguistic fuzzy rules of the REDD3 technique based on aql, T and P_L.

Similar to both the REDD1 and REDD2 algorithms, the FIP discussed in Section 5.3 is utilised by the REDD3 method to control the congested router buffers. The FIP employs three input linguistic variables (P_L, T, aql) to produce an output crisp value for a single output linguistic variable (D_P).

5.6 Performance Evaluation of the Fuzzy Logic Controller Algorithms

In this section, the proposed FL algorithms (REDD1, REDD2, REDD3) have been compared with the classic RED algorithm according to different performance measures (mql, T, D, P_L, D_p, P_{Loss}) in order to achieve the following goals:

1) Identifying the algorithm that offers the most satisfactory performance measure results with reference to (mql, T, D, P_{Loss}).

2) The algorithm that loses and drops (P_L, D_p) fewer packets at its router buffer, where the packet loses is due to router buffer overflows.

The performance measures are obtained by setting the packet arrival probability to several different values. Consequently, a decision about which algorithm is offering more satisfactory performance measure results is only given based on the parameter values of packet arrival probability.

The performance measure results of RED, REDD1, REDD2 and REDD3 algorithms are obtained after the system reaches to a steady state. Every compared algorithm has been run ten times in order to perform statistical analysis. In particular, each performance measure has ten results generated from the ten runs for the compared algorithms. Moreover, every value for the packet arrival probability was run ten times for statistical analysis purpose. The statistical analysis results describe how much the data (the mean results of every ten runs) are dispersed from their mean. The proposed FL algorithms (REDD1, REDD2, REDD3) implemented by simulation in a Java environment on a 1.66 Centrino with 1024 MB RAM. This section is structured as follows: The mean values for (mql, T, D, P_{Loss}) and (P_L, D_p) of the RED and proposed FL algorithms based on the ten runs are given in Subsections 5.6.1 and 5.6.2, respectively. The statistical analysis results for the proposed FL and RED algorithms are given in Subsection 5.6.3. Finally, the chapter summary is in Section 5.7.

5.6.1 Performance Evaluation of the RED and the Fuzzy Logic Algorithms

The proposed REDD1, REDD2 and REDD3 algorithms are compared with the RED algorithm in terms of the following performance measures: mql, T, D and P_{Loss} to decide which algorithm offers more satisfactory performance measure results. The decision about which algorithm has more satisfactory performance measure results depends on the values of the packet arrival probability. The parameters of RED were set as in Subsection 3.1.2. The proposed FL algorithm's parameters (Probability of packet arrival and departure, $BufferCapacity$ includes packets in the service (System capacity), qw, $D_{\max}$,Number of slots), have been set similar to the RED's values. It should be noted that since the proposed FL algorithms use the FIP, the $\min threshold$ and the $\max threshold$ are not applicable.

The results for calculation purposes are taken only are performed after the system reaches a steady state. The results of the performance measures are carried out based on the values of the packet arrival probability and these values vary from 0.18 to 0.93. Each compared algorithm has been run ten times with the seed changing for each run. This analysis describes the results for the ten runs. To be more specific, each algorithm was run ten times for each packet arrival probability value with the sees changed. Then, for each performance measure the mean result of the ten runs is computed. This mean results denote the performance measure results with reference to mql, T, D and P_{Loss} that are shown in Figures (5.14-5.17) respectively. Particularly, Figures 5.14 and 5.15 depict mql and T results versus the packet arrival probability values in the proposed FL and RED algorithms, respectively. The results of D and P_{Loss} for the proposed FL and RED algorithms are illustrated in Figures 5.16 and 5.17, respectively.

It is shown in Figures 5.14 and 5.16 that the RED and proposed FL algorithms offer similar mql and D results when 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 increases to be very close to the packet departure probability, i.e. 0.48, all the compared algorithms except the REDD1 give similar mql and D results and their results are slightly better than that of REDD1. This is since the REDD1 algorithm loses a marginally larger number of packets than the rest of the compared algorithms when a light congestion occurs. In cases where the packet arrival probability increases to be 0.63, RED, REDD2 and REDD3 algorithms offer better mql and D results than REDD1. Furthermore, at this probability of packet arrival value, the congestion increases, and REDD2 and

REDD3 algorithms become better than the RED algorithm with regard to mql and D since they stabilise their mql at values lower than those of RED.

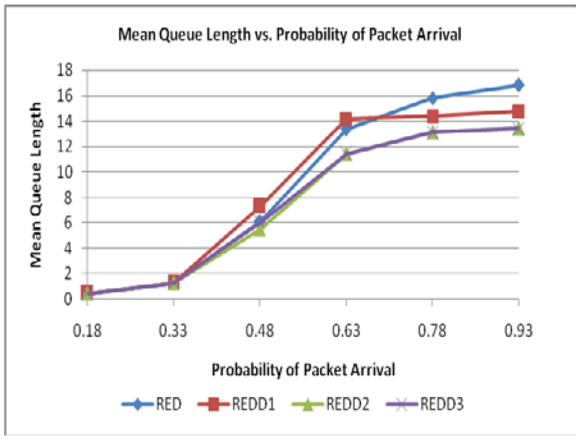

Figure 5.14: mql vs. probability of packet arrival.

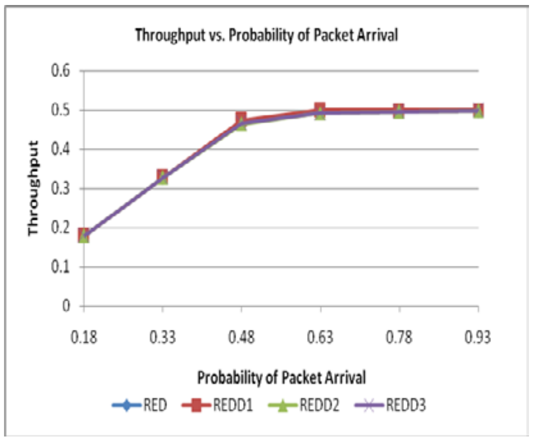

Figure 5.15: T vs. probability of packet arrival.

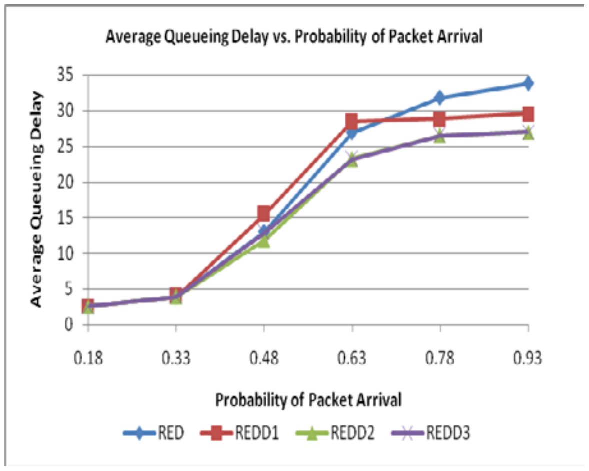

Figure 5.16: D vs. probability of packet arrival.

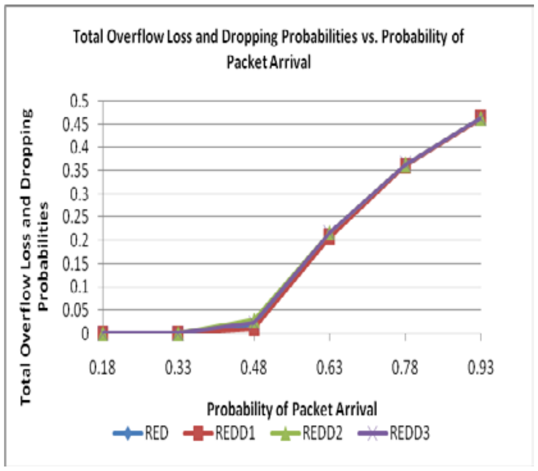

Figure 5.17: P_{loss} vs. probability of packet arrival.

Moreover, when the packet arrival probability increases to 0.78 or 0.93, a heavy congestion occurs, and REDD1, REDD2 and REDD3 algorithms sustain their mql and D results at values smaller than those of RED. Consequently, the proposed algorithms produce better mql and D results than those of RED. Also at these packet arrival probability values, REDD2 and REDD3 algorithms outperform REDD1 in terms of the mql and D results since they lose larger number of packets than REDD1. Finally, the proposed REDD2 and REDD3 algorithms provide similar mql and D results whether the packet arrival probability value is high or low.

Figures 5.15 and 5.17 indicate that regardless the packet arrival probability value, RED and proposed FL algorithms present similar T and P_{Loss} results. Moreover, the T result is increased as long as the packet arrival probability value increases but is still less than the packet departure probability value. Conversely, if the packet arrival probability $>$ packet departure probability, the T result will stabilise at the value of packet departure probability (0.5).

5.6.2 The Overflow Loss and Dropping Probability Results For the RED, REDD1, REDD2 and REDD3

This Subsection gives a comparison between the proposed FL algorithms and RED according to the loss probability (P_L) for packets and packet dropping probability (D_p) to identify which algorithm loses and drops fewer packets at its router buffer. The RED's parameters were set as in Subsection 3.1.2, whereas the parameters of the REDD1, REDD2 and REDD3 algorithms have been set as in Subsection 5.6.1. The mean results of P_L and D_p for the compared algorithms are shown in Figures 5.18 and 5.19, respectively.

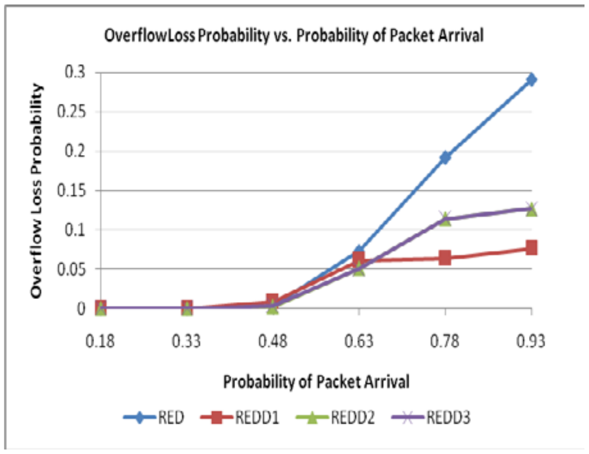

Figure 5.18: P_L vs. probability of packet arrival.

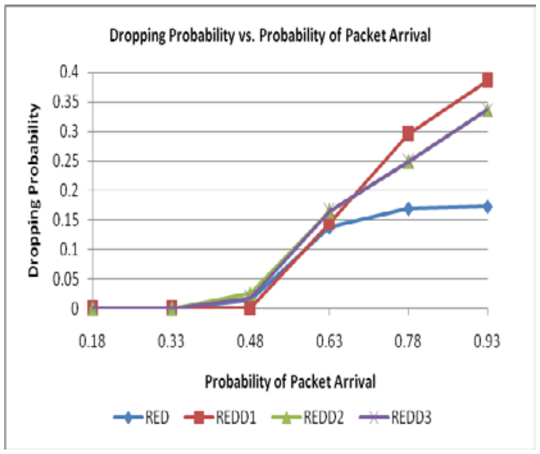

Figure 5.19: D_p vs. probability of packet arrival.

It is observed in Figures 5.18 and 5.19 that RED, REDD1, REDD2 and REDD3 algorithms lose (P_L) and drop similar amount of packets when the packet arrival probability is less than the packet departure probability or the packet arrival probability value equals 0.63. Also, it is apparent that the REDD1, REDD2 and REDD3 algorithms lose fewer packet than RED when a heavy congestion is present (packet arrival probability value = 0.78 or 0.93). This is since the RED router buffer overflows at times larger than the times of the proposed FL algorithms. In addition at 0.78 or 0.93 values for the probability of packet arrival, REDD1 loses fewer packets than either REDD2 or REDD3 since its router buffer overflows at times lower than those of the REDD2 and REDD3 algorithms.

It is obvious in Figure 5.19 that the RED algorithm drops fewer packets than the proposed FL algorithms when the packet arrival probability value equals 0.78 or 0.93. This is because the RED's router buffer overflows at times higher than those of the proposed FL algorithms. In addition, at these packet arrival probability values, REDD2 and REDD3 algorithms drop fewer packets than REDD1 algorithm since REDD2 and REDD3 router buffers overflow at times lower than those of REDD1. Moreover, Figures 5.18 and 5.19 show that the proposed REDD2 and REDD3 algorithms lose and drop similar amounts of packets at their router buffers whether the value of packet arrival probability is high or not. This packet loss is due to overflowing of the REDD2 and REDD3 routers buffers.

5.6.3 Statistical Analysis of RED, REDD1, REDD2 and REDD3

The statistical analysis in each compared algorithm consists of results based on ten runs for every performance measure in order to derive statistical measurements such as the mean, variance (σ^2), standard deviation (σ), and the 95% confidence intervals (CI). These statistical measurements describe how the performance measures are dispersed from their mean.

In Subsection 3.1.3, the RED's statistical analysis was discussed. The statistical analysis details for one of the proposed FL algorithms (REDD3) are introduced in this subsection. This is because REDD3 covers both REDD1 and REDD2 algorithms (combines both of them). The statistical analysis measurements of REDD3 algorithm are computed according to the packet arrival probability. Tables [5.1-5.6] show the statistical analysis for the performance measures of REDD3 versus the probability of packet arrival values. Specifically, Tables [5.1-5.3] indicate the statistical analysis concerning mql, T and D, respectively. Whereas the statistical analysis regarding P_L, D_p and P_{Loss} is illustrated in Tables [5.4-5.6], respectively.

It is clear from Tables [5.1-5.6] that the statistical measurement results (σ^2, σ and 95% CI, upper limit, lower limit) are extremely small. This reflects that the results for every performance measure are not largely dispersed from their mean. Thus, a more accurate mean result is suggested. Also, the statistical measurement results of the RED (see Subsection 3.1.3) and the proposed REDD1 and REDD2 algorithms are also extremely small. Furthermore, it is observed in Tables [5.1-5.6] that the largest deviations occur when the value of packet arrival probability was set to 0.48. This is because this value is very close to the service rate (0.5) and hence small random fluctuations in the input can cause congestion, even though the mean arrival rate is strictly less than the service rate.

Table 5.1: Mean mql vs. probability of packet arrival with statistical analysis for the REDD3 method.

REDD3 Method						
Probability of Packet Arrival	0.18	0.33	0.48	0.63	0.78	0.93
Mean mql	0.457	1.279	6.007	11.423	13.135	13.426
σ^2	2.640E-06	0.0003	0.121	0.0008	6.399E-05	0.0006
σ	0.0016	0.018	0.349	0.029	0.007	0.024
95% CI	0.001	0.011	0.216	0.018	0.004	0.015
Upper Limit	0.458	1.290	6.223	11.442	13.140	13.442
Lower Limit	0.456	1.267	5.791	11.405	13.130	13.411

Table 5.2: Mean T vs. probability of packet arrival with statistical analysis for the REDD3 method.

REDD3 Method						
Probability of Packet Arrival	0.18	0.33	0.48	0.63	0.78	0.93
Mean T	0.178	0.327	0.467	0.493	0.496	0.498
σ^2	3.469E-07	1.946E-06	5.339E-06	3.093E-08	2.228E-09	5.814E-10
σ	0.0005	0.001	0.0029	0.0001	4.721E-05	2.411E-05
95% CI	0.0003	0.0008	0.001	0.0001	2.926E-05	1.494E-05
Upper Limit	0.179	0.328	0.468	0.493	0.496	0.498
Lower Limit	0.178	0.326	0.466	0.492	0.496	0.498

Table 5.3: Mean D vs. probability of packet arrival with statistical analysis for the REDD3 method.

REDD3 Method						
Probability of Packet Arrival	0.18	0.33	0.48	0.63	0.78	0.93
Mean D	2.560	3.903	12.846	23.167	26.444	26.944
σ^2	0.0001	0.002	0.484	0.002	0.0002	0.002
σ	0.010	0.047	0.695	0.054	0.014	0.048
95% CI	0.006	0.029	0.431	0.033	0.009	0.030
Upper Limit	2.566	3.932	13.277	23.200	26.453	26.974
Lower Limit	2.554	3.873	12.414	23.133	26.435	26.914

Table 5.4: Mean P_L vs. probability of packet arrival with statistical analysis for the REDD3 method.

REDD3 Method						
Probability of Packet Arrival	0.18	0.33	0.48	0.63	0.78	0.93
Mean P_L	0	0	0.004	0.050	0.114	0.127
σ^2	0	0	6.552E-11	4.267E-07	2.391E-07	5.104E-07
σ	0	0	8.094E-06	0.0006	0.0004	0.0007
95% CI	0	0	5.017E-06	0.0004	0.0003	0.0004
Upper Limit	0	0	0.004	0.051	0.114	0.127
Lower Limit	0	0	0.003	0.050	0.114	0.126

Table 5.5: Mean D_p vs. probability of packet arrival with statistical analysis for the REDD3 method.

REDD3 Method						
Probability of Packet Arrival	0.18	0.33	0.48	0.63	0.78	0.93
Mean D_p	0	0	0.018	0.165	0.248	0.336
σ^2	0	0	7.0003E-05	2.904E-06	8.222E-07	2.625E-06
σ	0	0	0.008	0.001	0.0009	0.001
95% CI	0	0	0.005	0.001	0.0005	0.001
Upper Limit	0	0	0.023	0.166	0.249	0.337
Lower Limit	0	0	0.013	0.164	0.248	0.335

Table 5.6: Mean P_{Loss} vs. probability of packet arrival with statistical analysis for the REDD3 method.

REDD3 Method						
Probability of Packet Arrival	0.18	0.33	0.48	0.63	0.78	0.93
Mean P_{Loss}	0	0	0.022	0.216	0.363	0.464
σ^2	0	0	7.012E-05	4.899E-06	7.251E-07	1.322E-06
σ	0	0	0.008	0.002	0.0008	0.001
95% CI	0	0	0.005	0.001	0.0005	0.0007
Upper Limit	0	0	0.027	0.217	0.363	0.464
Lower Limit	0	0	0.017	0.214	0.362	0.463

5.7 Chapter Summary

In this chapter, three new FL controller algorithms have been proposed (REDD1, REDD2, REDD3) based on the known RED algorithm in order to achieve the following goals:

1. Obtain a more satisfactory performance measure results with respect to the mean

queue length (mql) and the average queueing delay (D) than the RED algorithm when a heavy congestion occurs.

2. Lose fewer packets than RED due to overflow their router buffers when a heavy congestion occurs.
3. Decrease the dependency of RED on its parameters as congestion boundaries, instead the proposed FL algorithms rely upon a FIP as a congestion measure.

The results of the compared algorithms (RED, REDD1, REDD2, REDD3) with reference to the performance measures are derived through setting the packet arrival probability to different values. So the decision about which algorithm is better in terms of the performance varying is based on the packet arrival probability values. Also, the statistical analysis results for every performance measure of the RED and the proposed FL algorithms are obtained. The following can be observed from the comparison of RED and the FL algorithms:

- The proposed REDD2 and REDD3 algorithms offer similar performance measure results ($mql, T, D, P_L, D_p, P_{Loss}$) regardless the packet arrival probability value.
- All the compared algorithms (RED, REDD1, REDD2, REDD3) provide similar mql and D results when the packet arrival probability value is either 0.18 or 0.33. Whereas, when the packet arrival probability increases to a value near the packet departure probability value (0.48), the RED, REDD2, and REDD3 algorithms generate similar results with respect to mql and D and these results are marginally better than those of the REDD1 algorithm. Furthermore, if the packet arrival probability value becomes

near 0.63, the compared algorithms excluding REDD1 remain giving better mql and D results than REDD1, and both REDD2 and REDD3 slightly better than the RED with regard to mql and D. Lastly, when the packet arrival probability value was set to 0.78 or 0.93, then the proposed FL algorithms outperform the RED algorithm regarding mql and D. Also, at these packet arrival probability values both REDD2 and REDD3 outperform REDD1.

- All the compared algorithms offer similar T and P_{Loss} results regardless the packet arrival probability value.
- All the compared algorithms suffer loss due to overflow at their router buffers and drop similar amount of packets when the packet arrival probability value $<$ the packet departure probability value or the packet arrival probability value = 0.63. The proposed FL algorithms lose fewer packets due to overflow their routers buffers than the RED algorithm when the packet arrival probability value is equal to 0.78 or 0.93 (when heavy congestion occurs). On the other hand, the RED algorithm outperformed the proposed FL algorithms in terms of D_p when the packet arrival probability value is equal to 0.78 or 0.93.
- The RED and the proposed FL algorithms provide very small values for the statistical measurement results (σ^2, σ and 95% CI, upper limit, lower limit) for every performance measure. Therefore, the results for every performance measure are not largely dispersed from their mean.
- The largest deviations for all the considered algorithms were obtained when the packet arrival probability value was set to 0.48 since this is a value approaching the service

rate (0.5) and small fluctuations in the input may derive the system into a congestion situation.