Improving the Adaptability of AQM Algorithms to Traffic Load Using Fuzzy Logic

Zhi Li, Zhongwei Zhang, Ron Addie
Department of Mathematics & Computing
University of Southern Queensland
zhili, zhongwei, addie@usq.edu.au

Fabrice Clérot
France Télécom Research and Development, Lannion, France
fabrice.clerot@rd.francetelecom.com

Abstract

In order to avoid the sensitivity of RED mechanism to the parameter settings and traffic load, Adaptive RED (ARED) has been proposed to self-tune the parameters to adjust to the current traffic conditions. More importantly, ARED also potentially achieves another goal to keep average queue length around a predefined queue length with certain deviation. If so, average delay will be offered to Internet users and at the same time high link utilization will be achieved. However, ARED algorithm merely makes average queue length loosely converge to the target length with low responsiveness and low stability. One reason for this is ARED uses a fixed increase step-size to adjust the maximum marking/dropping probability to the current traffic load under a variety of traffic scenarios.

In this paper, using artificial intelligence (AI) technology fuzzy logic (FL), a FL controller has been designed to adaptively obtain an increase step-size to adapt the maximum marking/dropping probability to the current traffic conditions. The simulation results show the proposed FL scheme achieves the desired target queue length with better performance than ARED, while keeping very similar drop rate and link utilization.

1 Introduction

Even though TCP's flow-control and congestion-control mechanisms are capable of detecting information about the capacity of TCP receivers and congestions in networks, it is not enough to avoid congestion occurrences. For instance, from the time when packets are dropped from a router because queues overflow, to the time when the sending TCP realizes that they have been

dropped, many packets sent on the way to their destination are likely to be dropped for the same reason. Therefore, the goodput of a TCP connection is decreased dramatically, and at the same time network resources are wasted. Alternative approaches for congestion control are router or gateway based, and carry out active queue management. Random Early Detection (RED) randomly notifies sending TCP hosts involved in connections passing through a congested link. After receiving the information, the sending TCPs decrease the transmission rate to avoid or alleviate congestions. RED gateway algorithm outperforms traditional so-called drop-tail queue management method, since it prevents many problems of drop-tail, such as global synchronization, low goodput, and low link utilization [6]. However, as has been pointed out in [2], it is difficult to find a set of suitable value for the RED parameters under different congestion scenarios. The reason is, under different traffic load, different parameter settings are required. Moreover, because of the constantly changing nature of network communication, constant auto-tuning of these parameters is needed.

Adaptive RED (ARED) has been presented in [3] to self-tune these parameters. The goals of ARED and also RED are to achieve low drop rate, high link utilization, and low average delay. If so, the end users are satisfied with high goodput and low average delay, while the network resources are utilized efficiently. To achieve these goals, the ARED mechanism attempts to maintain the average queue length around a desired target queue length with a certain deviation.

However, the ARED algorithm merely results in the average queue length to loosely converge to the target length with low responsiveness and low stability. One reason for this is that ARED uses a fixed increase step-size to adjust the maximum marking/dropping probability max_p. In ARED, an additive-increase multiplicative-decrease (AIMD) policy is used. When average queue length is below the target, ARED uses multiplicative-decrease. On the other hand, when it is above the target, additive-increase with a fixed step-size is utilized to adjust max_p to the current traffic conditions. The adoption of the fixed increase step-size is based on experiments and may not be optimal to all the traffic scenarios.

In this paper, a controller based on fuzzy logic (FL) has been designed to control average queue length, which has the capability of adaptively choosing an increase step-size for adapting max_p to the current traffic load. In section 2, RED and ARED mechanisms are briefly introduced. Our FL adaptive RED (FLARED) is presented in section 3 to improve the performance of ARED. Simulation results are given in section 4. Finally, concluding remarks and future work are discussed in section 5.

2 Active Queue Management Algorithms

The problem of parameter settings is one of RED's main weaknesses. All the parameters in RED are listed in Table 1. Of these, w_q is the weight for estimating the exponential weighted average queue length, *ave*. In the RED mechanism, a gateway marks incoming packets with a certain probability based on *ave* and max_p in the ECN field of an IP packet header. Otherwise, it simply drops the incoming packets with the same probability [4]. When *ave* is below min_{th}, the incoming packets are never marked/dropped. In [4], when *ave* is above max_{th}, all the incoming packets are marked/dropped, while in a later paper [3], *gentle mode* is adopted, in which the marking/dropping rate increases linearly from max_p at max_{th} to 1 at $2max_{th}$.

Table 1: RED Parameters

Parameter	Function
w_q	Weight for estimating average queue
max_p	Maximum marking/dropping probability
min_{th}	Lower threshold
max_{th}	Upper threshold

RED's performance is sensitive to traffic load and to RED parameter settings. The parameters need to be set carefully to get better performance of low drop rate and high utilization under different congestion scenarios, otherwise, it might have the same or even worse performance compared with drop-tail [2]. The ARED algorithm has been presented in [3] to self-tune the parameters to current traffic conditions.

The objective of ARED is to keep *ave* around a target queue length with a certain deviation or a target range of queue length, where the desired target queue length is the trade-off between throughput and delay decided by a network operator. In this way, ARED achieves low drop rate, high utilization, and low average delay. This is the objective of auto-tuning the parameters. After the determination of the desired target queue length, all the parameters, except max_p, are set automatically [3]. For instance, w_q is configured based on the link speed. Or, if preferred, the network operator can configure them by hand. The parameter max_p is left to be the key parameter to keep *ave* around target queue length $(min_{th} + max_{th})/2$ with a certain deviation of $0.1 * (max_{th} - min_{th})$. And thus, the target range is $[min_{th} + 0.4(max_{th} - min_{th}), min_{th} + 0.6(max_{th} - min_{th})]$. AIMD policy is adopted to adjust max_p every fixed time interval greater than a typical round-trip time. This is, when *ave* is less than the lower bound of the target range, decrease max_p by $max_p\beta$; and when *ave* is greater than the upper bound of the target range, increase max_p by $max_p+\alpha$. Moreover, in order to avoid oscillation, the parameter α is set to a constant value or $0.25max_p$, whichever is smaller, and the default value for the constant value is 0.01. Therefore, when

max_p is greater than 0.04, the constant value will be in charge of increasing max_p, whenever an increase action is required. The default value for constant value β is 0.9(see [3] for details).

The problem of the RED algorithm is average queue size *ave* has too much oscillation from below min_{th} to above max_{th}. The problem has been alleviated to an extent by ARED. The performance comparison of RED and ARED for controlling *ave* is illustrated in Figure 1, which is the combined results of Figure 11 and 13 in [3].

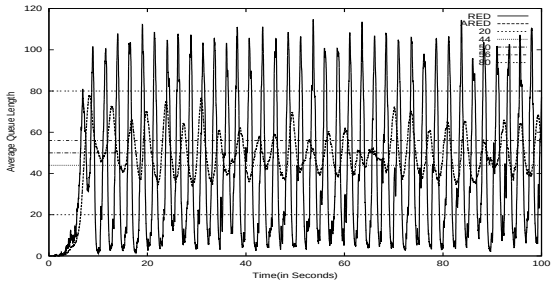

Figure 1: *ave* of RED and ARED

In Figure 1, *ave* of ARED is around target length 50 packets, and is totally controlled in range $[min_{th}, max_{th}]$, i. e. [20, 80], whereas *ave* of RED exhibits excessive oscillation and far beyond this range. As a result, ARED has better performance than RED in terms of achieving low drop rate, high utilization, and low average delay simultaneously. Figure 1 also shows that ARED loosely converges to the target range [44, 56] with low responsiveness and low stability.

Despite its merit, ARED has a problem that, with a fixed increase step-size α, it is hard for ARED algorithm to reach an optimal value of max_p to quickly respond the changes of traffic load under a variety of traffic scenarios. Consequently, the adoption of the fixed increase step-size partially affects the performance of ARED with loose convergence to the target queue length. Study has been done with different values of increase step-size α, and the conclusion is that for different traffic scenarios, different values of α would be preferred. More detail will be given in section 4.

3 An Active Queue Management Mechanism Using FL: FLARED

Variable increase step-size can be generated by the use of fuzzy logic (FL). In general, FL has the capability of mimicking how field experts would make decisions [5]. And thus, it provides a simple way to arrive at a definite conclusion merely based upon vague, ambiguous, imprecise, noisy, or missing input information.

FL is a powerful tool to provide solutions for controlling a complex system. In networking area, there is a variety of traffic mixes including different round-

trip time (RTT), different traffic types, and long-lived and short-lived traffic flows. Accurately modeling what is going on in a router's output buffer is hard, though FL can incorporate networking expert knowledge to generate sensible solutions. In order to improve the performance of the active queue management algorithm ARED, a mechanism using fuzzy logic (FL) has been proposed to auto-configure the increase step-size in ARED.

The hypothesis is that the problem of queue management in routers can be addressed by dealing with different levels of *ave* and detecting its changing tendency. However, such knowledge has not been applied in ARED. An FL system can readily recognize the degree of the deviation between current *ave* and target length, and then takes an appropriate action with the consideration of the tendency of *ave*.

FL is based on the concepts of membership functions and fuzzy if-then rules. In the following sections, we introduce the membership functions and if-then rules used in our FL adaptive RED (FLARED) in turn. In this FL control system, *max-min* defuzzification and the *center of gravity* method are adopted.

3.1 FL Membership Functions

Our proposed FL adaptive RED (FLARED) mechanism aims to keep *ave* around a desired target range of queue length by adaptively choosing increase step-size of max_p based on the current traffic conditions. The parameter *ave* is a good indicator for adjusting max_p. A big *ave* means that the current value of max_p is too small and vice versa. The difference between the current value of *ave* and the preceding one in the last time interval, δave, is the feedback from the last control action, and it also indicates the trend of traffic load. FLARED includes an FL control system with input variables *ave* and δave and one output variable, increase step-size of max_p, δmax_p, as described in Figure 2. Note that when *ave* first jumps above the target range, a fixed increase step-size is used. A small value, 0.003, is chosen for the fixed first jump increase step-size, and let FLARED do the auto-configuration of this parameter thereafter.

Figure 2: FL controller

L(ow), M(edium), M(edium)H(high), and H(high) are used as linguistic variables of input *ave*. Figure 3 shows the membership functions of *ave*. Here, the parameters $a1$, $a2$, $a3$, and $a4$ are $min_{th} + 0.6\,(max_{th} - min_{th})$, $min_{th} + 0.8\,(max_{th} - min_{th})$, max_{th}, and $2max_{th}$, respectively.

NE(negative), ZE(zero), and PO(positive) are utilized to describe input δave. The membership functions are given in Figure 4. The parameters $d1$ and $d2$ are symmetric in FLARED and equal $0.1\,(max_{th} - min_{th})$. For the case

where min_{th} and max_{th} are 5 and 15, $d1$ is 1, while it is 6 when min_{th} and max_{th} are 20 and 80,respectively.

Similarly, Ze(zero), PS(positive small), PM(positive medium), PMB(positive medium big), PMMB(positive medium medium big), and PB(positive big) are the linguistic variables for δmax_p. Figure 5 shows the membership functions.

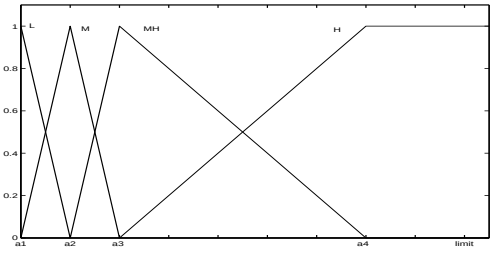

Figure 3: Membership functions for *ave*

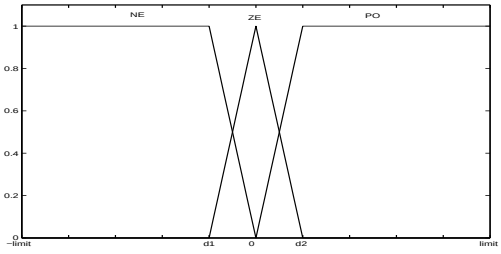

Figure 4: Membership functions for δave

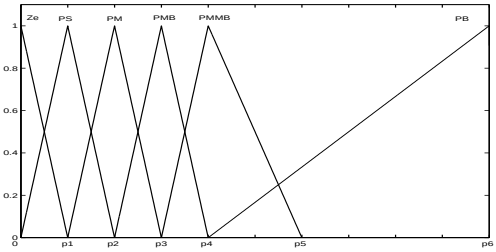

Figure 5: Membership functions for δmax_p

3.2 FL Rules

Table 2 shows the fuzzy if-then rules in FLARED. To make the fuzzy rules easy to understand, we discuss the following three cases.

- δave is NE. This means the last action of δmax_p does take effect, and control the increase of *ave* to some degree, but not sufficiently. In this case, be cautious to increase δmax_p, since the trend of traffic load is decreasing.
- δave is ZE. This means more traffic has arrived. Increase δmax_p according to the current *ave*.
- δave is PO. In this case, much more traffic has arrived. Take a stronger action to avoid congestion occurrences.

In FLARED, we have finer classification for the input variable *ave* between $min_{th} + 0.6\,(max_{th} - min_{th})$ and max_{th}. Therefore, parameters $p1$, $p2$, and $p3$ are fairly close together. And also a big value is preferred to deal with the situation of big increases. The parameters for output δmax_p are as follows. $p1 = 0.005$, $p2 = 0.01$, $p3 = 0.015$, $p4 = 0.025$, $p5 = 0.05$, and $p6 = 0.2$.

Table 2: FL rules

Rule no.	ave	δave	δmax_p
1	L	NE	Ze
2	L	ZE	PS
3	L	PO	PM
4	M	NE	PS
5	M	ZE	PM
6	M	PO	PMB
7	MH	NE	PM
8	MH	ZE	PMB
9	MH	PO	PMMB
10	H	NE	PMB
11	H	ZE	PMMB
12	H	PO	PB

4 Simulations

FLARED has the capability to auto-configure increase step-size according to the current traffic conditions. In this section, we examine stability and responsiveness of FLARED with the *ns-2* simulator [1]. The simulation results are given by comparison of ARED and FLARED. Note that Tahoe TCP is used instead of SACK TCP, and both gateways and hosts do not use Explicit Congestion Notification (ECN) field in the IP packet header. The reason is to cooperate with the current TCP version.

4.1 Experiments

A dumbbell topology is used in this simulation. The simulation scenario is similar to the one related to Figures 12 and 14 in [3]. In addition, rush hour is considered to detect the stability and responsiveness of FLARED. ARED and FLARED will be respectively installed in gateways at the two ends of the bottleneck link as active queue management mechanisms. Traffic is mixed, having long-lived and short-lived traffic flows in either direction through the bottleneck link. The bottleneck link is 15Mbps. At each end of the link are gateways with 160 packets output buffer size. 50 traffic flows are utilized as forward traffic, while half of these traffic flows are in the other direction. 20,000 web mice (short-lived TCP connections) randomly appear on either direction of the bottleneck. Moreover, rush hour is simulated as another 50 traffic flows are in the forward direction together with 25 in the other direction during simulation time 40s to 80s. In this simulation, the target length is 50 packets, while 20 and 80 packets are used as min_{th} and max_{th}, respectively.

The measurement used here are frequency, average queue size *AVE*, drop rate in gateways and link utilization. Frequency measures how often *ave* falls in the target range during measurement time. AVE is the average of queue size during measurement time. Drop rate is calculated by the ratio of dropped

Table 3: Simulation results

Scheme	Frequency(%)	AVE(packets)	Drop rate(%)	Link utilization(%)
ARED	38.80	49.01	2.18	90.08
FLARED	53.65	50.27	2.26	89.98

packets number to the total arrival packets number, while the ratio of the amount of packets passing through the gateway to the number of total packets able to be supported by the link is computed as link utilization. The measurement time used in all the simulations starts at simulation time 20s. Since *ave* is normally between max_{th} and max_{th}, the simulation results of ARED and FLARED have similar drop rate and link utilization.

The simulation results of ARED and FLARED are shown in Table 3. And also the dynamics of *ave* and max_p with ARED and FLARED is shown in Figures 6, 7, Figures 8 and 9 respectively. Both Table 3 and Figure 8 show that the *ave* of FLARED falls in the target range [44, 56] with higher frequency than that of ARED without much cost in decreasing link utilization and drop rate. The dynamics of max_p in Figure 9 illustrates that FLARED can quickly respond to the increase of traffic load, and at a certain traffic load level, steadily adjust max_p without too much oscillation.

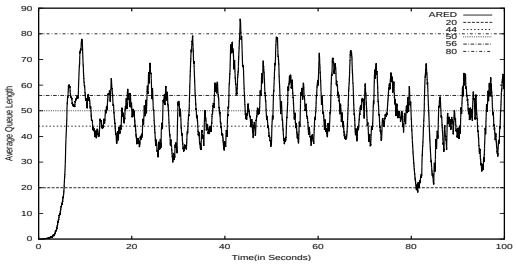

Figure 6: *ave* of ARED

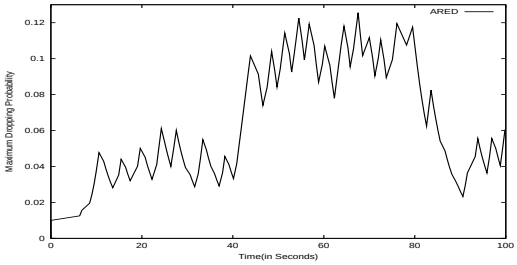

Figure 7: max_p of ARED

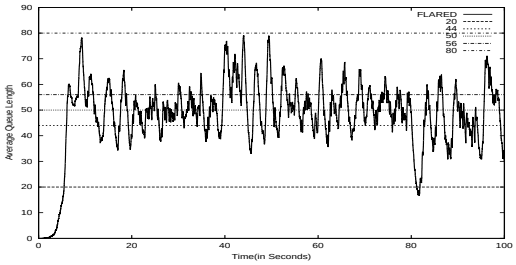

Figure 8: *ave* of FLARED

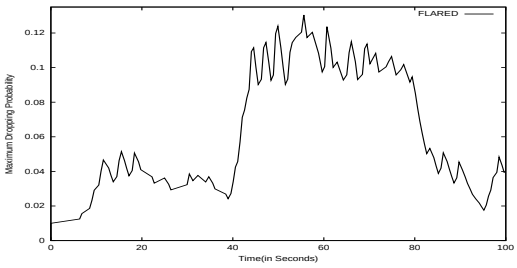

Figure 9: max_p of FLARED

Table 4: Simulation results with different α

Scenario	α=0.003		α=0.01	
	Frequency(%)	AVE(packets)	Frequency(%)	AVE(packets)
1	13.58	15.33	27.73	13.09
2	32.00	10.20	35.40	9.00
3	55.33	52.57	41.90	49.10
4	48.54	54.77	53.76	50.08
Scenario	α=0.1		FLARED	
	Frequency(%)	AVE(packets)	Frequency(%)	AVE(packets)
1	52.08	10.62	39.48	12.35
2	34.31	8.65	50.23	9.24
3	37.94	46.46	50.25	50.12
4	44.58	47.02	53.87	51.31

4.2 Comparison of ARED and FLARED

In this subsection, we replicate some simulations related to Figure 6 - Figure 14 in [3] with different increase step-size value α, and show the performance of FLARED for comparison.

In the four simulation scenarios, a dumbbell topology is used. In the first two scenarios, the bottleneck link is 1.5Mbps with 20ms-delay, buffer size is 35 packets, and three long-lived TCP connections exist. In the first scenario, 20 new traffic flows arrive almost simultaneously at simulation time 25s, whereas 15 traffic flows finish transmission simultaneously at simulation time 25s in the second scenario. For the big increase in scenario one, Table 4 shows that a big value of α is suitable, while there is not much difference between a variety of α values in scenario two. In the last two scenarios, the bottleneck link of 15Mbps and 120ms-delay is used, and buffer size of the attached gateway is 160 packets. In the third scenario, only one-way traffic is considered, while two-way and both long-lived and short-lived traffic flows are used in the fourth scenario. In these two cases, a large value of α is not preferred.

Once again, we use frequency, average queue size (AVE) as metrics. Simulation results are given in Table 4 to show the performance of ARED with different α and FLARED for comparison. Note that simulation results for drop rate and link utilization of ARED with different α and FLARED in all the cases are similar.

As shown in Table 4, even though our mechanism FLARED can't achieve the best performance all the time, it certainly improves the quality of ARED in general. In terms of frequency, the result of FLARED is best or approaching the best result in all the scenarios. For AVE, the result of FLARED is in the desired average queue length range except in the first case, where the results of both frequency and AVE with FLARED are better than those of ARED with the default value 0.01 for increase step-size α.

5 Conclusion

ARED self-tunes the parameters to adjust to current traffic conditions. However, the adoption of fixed increase step-size is not an appropriate way to search for an optimal value of maximum dropping probability max_p, while dealing with a variety of network topologies and traffic mixes. Consequently, it detracts from ARED's performance in terms of achieving target queue length.

In this paper, the performance improvement of adaptive RED (ARED) is explored. A scheme using fuzzy logic (FL) has been presented to automatically obtain an appropriate increase step-size based on the current situation in the output queue. The simulation results show that FLARED is able to improve the performance of ARED in both stability and responsiveness.

Certainly, there is more improvement required to get ideal performance, that is to control *ave* within the target range with high frequency. More effort will be put into the future work to get the ideal performance of active queue management on the Internet. An adaptive time scale instead of fixed adjusting time interval will be studied to improve the responsiveness. A way for adjusting parameter weight for estimating average queue w_q to the changes of traffic load in order to more precisely describe average queue length will be explored. The FL controller will be further explored by combining other artificial intelligence technologies such as neural networks and genetic algorithms to self-tune the parameters in FLARED and choose FL rules on-line. The better performance of FLARED suggests also the potential to utilize FLARED in differentiated services to automatically adjust the parameters of queue management to achieve better network performance.

References

[1] The network simulator - ns-2. *http://www.isi.edu/nsnam/ns/.*

[2] W. Feng, D. D. Kandlur, D. Saha, and K. Shin. A self-configuring RED gateway. In *Proceedings of INFOCOM 99*, volume 3, pages 1320–1328, 1999.

[3] S. Floyd, R. Gummadi, and S. Shenker. Adaptive red: An algorithm for increasing the robustness of red, 2001.

[4] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[5] L. A. Zadeh. Fuzzy sets. *Information and control*, (8):338–353, 1965.

[6] L. Zhang, S. Shenker, and D. Clark. Observations on the dynamics of a congestion control algorithm: The effects of two-way traffic. In *SIGCOMM*, pages 133–147, 1991.