Advanced Queue Management Algorithms for Computer Networks

Jalal Al-Frihat

Ministery of Education, p.o.box 1646

Amman,

JORDAN

jal_fr10@yahoo.com

Keywords. Fuzzy logic control, congestion control, active queue management, TCP/IP

Abstract: The use of the Internet for time-sensitive services, such as voice and video applications, requires a predictable quality of service. The TCP/IP differentiated services architecture was introduced to achieve such performance. Network congestion control, however, still remains a critical and high priority issue and a number of alternative schemes such as random early detection (RED) and its variants were proposed to handle congestion. This paper presents an advanced active queue management (AQM) scheme to provide congestion control in TCP/IP best-effort networks by using a fuzzy logic approach. This advance scheme uses linguistic knowledge to implement better understood nonlinear probability discard functions, achieve better differentiation for packet discarding behaviors for aggregated flows, thus providing a better service quality to different kinds of traffic whilst maintaining high utilization.

Mr. Jalal Al-Frihat was born on January 12, 1962. He received his MSc degree in control system and computer engineering from Automatic Control and Computers Faculty, Polytechnical Institute "Traian Vuia" Timisoara, Romania in 1987. As a PhD student his research interests include computer networks, security, network applications.

1. Introduction

It is generally accepted that the problem of network congestion control remains a critical issue of high priority, especially given the growing size, demand, and bandwidth of the increasingly integrated services network. As the growth of the Internet increases it becomes clear that the existing congestion control solutions deployed in the Internet transport control protocol are becoming less effective.

The existing TCP congestion avoidance mechanisms and its variants, while necessary and powerful, are not sufficient to provide good service in all circumstances. Basically, there is a limit to how much control can be accomplished from the edges of the network.

The RED algorithm [3] was proposed as an active queue management (AQM) approach. RED proposes a strategy for when to start dropping packets and which packets to drop. The RED approach can be contrasted with the tail drop (TD) queue management approach, employed by common Internet routers, where the policy for discarding arriving packets is based on the overflow of the output port buffer. Contrary to TD, active queue management mechanisms start dropping packets earlier in order to be able to notify traffic sources about the incipient stages of congestion. RED has been designed to replace TD and is currently implemented in commercially available routers, but it is not as yet widely deployed.

Apart from RED, many other variants of RED, such as A-RED [4], BLUE [2], REM [5] and other schemes were proposed. In this paper, the strength of fuzzy logic in controlling complex and highly nonlinear systems is used to address congestion control problems.

The application of fuzzy control techniques to the problem of congestion control in networks is suitable due to the difficulties in obtaining a precise mathematical model using conventional analytical methods..

There is increasing empirical knowledge gathered about RED and its variants, and several ‘rules of thumb’ have appeared in many papers.

In this paper it is highlighted the potential of the methodology based on a simple rule base and simulation examples. The fuzzy knowledge base is devised and based on heuristic methods and experience. Some comparison was also made between proportional integral (PI) and Adaptive REDed (A-RED) algorithms.

The paper is organized as follows. Section 2 presents RED algorithm. In Section 3 the Adaptive RED is reviewed. The Fuzzy logic control with explicit marking is developed in Section 4. Section 5 is dedicated for some simulation results and finally in Section 5 conclusions of this paper are presented.

2. Random Early Detection (RED)

Random Early Detection (RED) [1] is a proactive queue management technique, whereby a router discards one or more incoming packets before its buffers overflow. Though RED is primarily designed for use on a single FIFO queue, it can be extended to multiple queue systems such as fair queuing [4] in which RED is individually applied to each queue.

RED has been developed with four basic objectives [3]. First, it avoids congestion by detecting the onset congestion and potential for congestion rather than reacting to it. Second, it avoids bias against bursty traffic by acting probabilistically. Third, it avoids global synchronization by breaking the oscillation between heavy and light state of the network. Fourth, it maintains a bound on the average delay by controlling the average queue length. RED algorithm performs mainly two crucial steps; an average queue size, AvgQ, computation and a packet drop probability, P_a. As shown in the generalized RED algorithm in Figure 1, the average queue size is computed every time a new packet arrives at the output queue.

Upon packet arrival **do**
  Compute the average queue size AvgQ
    if $AvgQ \geq T_h$ **then**
      drop the packet
    else if $T_l \leq AvgQ < T_h$ **then**
      drop packet with probability P_a
    end if
end do

Figure 1: Random Early Detection Algorithm

To determine whether an incoming packet should be dropped, RED first computes the average queue size, AvgQ. It also uses two thresholds T_l and T_h; a lower and an upper threshold, respectively, to control the growth of AvgQ.

The average queue size, AvgQ, is used to filter out transient congestion at the router and avoid any bias against bursty traffic. AvgQ is computed using an *exponentially weighted moving average* (EWMA) of the previous queue lengths. If we have $Q \neq 0$

$$AvgQ = (1 - W_q)\, AvgQ + W_q Q \tag{1}$$

in other case m is computed as the ration between queue_idle_time and typical_transmission_time and

$$AvgQ = (1 - W_q)^m\, AvgQ \tag{2}$$

where, the RED parameter W_q determines how rapidly AvgQ changes in response to changes in Q. m is a linear function of time to account for the periods when the queue is empty. The algorithm estimates the number of packets that could have been transmitted by the router during the idle period, and then uses that in the computation of the AvgQ when a new packet arrives. It is done this way because RED computes AvgQ at the packet arrival time rather than at a fixed time interval.

Between $[T_l, T_h)$, RED assigns a probability of discard for an incoming packet that depends on:

(1) the proximity of AvgQ to T_h. As AvgQ approaches T_h , the probability of discard approaches its maximum value P_{max}. In order to incorporate this dependence, RED calculates a temporary probability P_b , that increases linearly from 0 when AvgQ = T_l to the maximum value P_{max} when AvgQ = T_h

$$P_b = P_{\max} \frac{AvgQ - T_l}{T_h - T_l} \tag{3}$$

(2) the count c of consecutive packets that escaped discard. As this count increases, the probability of discard also increases. This property is incorporated to not penalize bursty sources by spacing the discards quite evenly rather than in a cluster. Hence, the actual packet dropping probability, P_a is a function of P_b given by:

$$P_a = \frac{1}{P_b^{-1} - c} \tag{4}$$

where c is the number of packets since last discard. For a given value of c, P_a increases gradually from 0 to P_{max}. Keeping other variables constant and varying c, it can be observed that value of P_a increases very slowly and then rises dramatically until it reaches 1, making it force a more or less uniform spacing of discards.

3. Adaptive Random Early Detection (A-RED)

In Adaptive RED [4] it is argued that congestion notification does not directly depend on the number of connections multiplexed over the link. In order for an early detection algorithm to work effectively, congestion notification must be given at a rate sufficient enough to prevent packet loss due to buffer overflow, but not too high to cause an under utilization of the link.

Upon packet arrival ***do*** Update AvgQ
  if T_l < AvgQ < T_h ***then***
    status = Between
  else if AvgQ < T_l ***and*** status ≠ Below ***then***
    status = Below
    $P_{max} = P_{max} / \alpha$
  else if AvgQ > T_h ***and*** status ≠ Above ***then***
    status = Above
    $P_{max} = P_{max} \beta$
  end if
end do

Figure 2: Adaptive Random Early Detection Algorithm

A-RED [4] reduces the packet loss rate by inferring whether or not RED should become more or less aggressive. This is performed by examining the variations in the average queue length, AvgQ. As given in Figure 2, if AvgQ oscillates around T_l, then the early detection mechanism is being too aggressive. If AvgQ oscillates around T_h, then the early detection mechanism is being too conservative. The algorithm thus dynamically adjusts P_{max}, based on the region of the AvgQ oscillation. P_{max} is scaled using constant factors α and β depending on which threshold it crosses. P_{max} is multiplied by β when the early detection is being too conservative and is divided by α when the early detection is being too aggressive.

4. Fuzzy Logic Controler FLC Explicit Marking

The design of a fuzzy control system is based on a fuzzy logic controlled AQM scheme to provide congestion control in TCP/IP best-effort networks. The system model of FLC is shown in Figure 3, where all quantities are considered at the discrete instant kT, with T the sampling period, $e(kT) = q_r - q$ is the error on the controlled variable queue length, q, at each sampling period, $e(kT - T)$ is the error of queue length with a delay T (at the previous sampling period), p(kT) is the mark probability, and G_{in} and Go are scaling gains.

The proposed fuzzy control system is designed to regulate the queues of IP routers by achieving a specified desired queue size, qr, in order to maintain both high utilization and low mean delay. A fuzzy inference system (FIS) is designed to operate on router buffer queues, and uses linguistic rules to mark packets in TCP/IP networks.

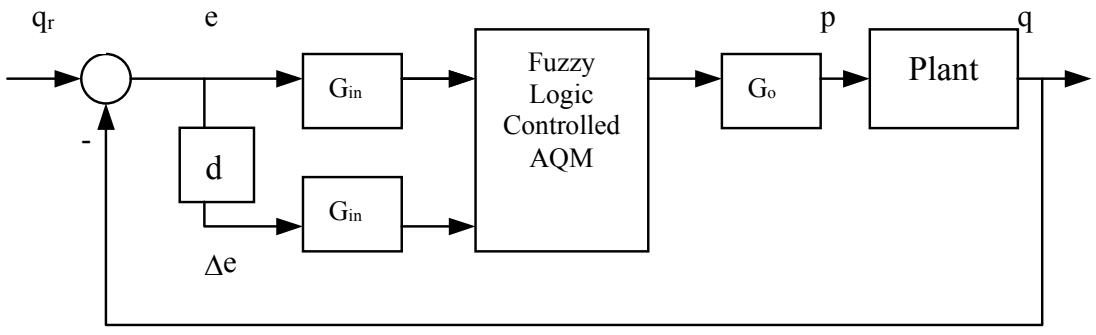

Figure 3: Fuzzy Logic Control Structure

As shown in figure 3, the FIS dynamically calculates the mark probability behavior based on two network-queue state inputs: the error on the queue length (i.e., the difference between the desired queue length and the current instantaneous queue length) for two consecutive sample periods.

The fuzzy controller is implemented with marking capabilities, so that FLC routers have the option of either dropping a packet or setting its ECN bit in the packet header, instead of relying solely on packet drops. The decision of marking a packet is based on the mark probability, which is dynamically calculated by the FIS.

The scaling gains, G_{in} and Go, shown in Figure 1, are defined as the maximum values of the universe of discourse of the FIS input and output variables, respectively. In order to achieve a normalized range of the

FIS input variables from -1 to 1, the input scaling gain Gin is set to be equal to -1/(qr–QueueBufferSize), if the instantaneous queue length is greater than the desired one; otherwise SGi is equal to 1/qr. The output scaling gain G_o is determined so that the range of outputs that are possible is the maximum, but also ensuring that the input to the plant will not saturate around the maximum.

The two input variables the error on the queue and the same with one sampling period delay are fuzzyfied on the universe -1 to 1 with five linguistic terms as follows negative big (NB), negative small (NS), yero (Z), positive small (PS), positive big (PB). The output variable, mark probability with the universe of discourse 0 to 1 has the following linguistic terms: zero(Z), very small (VS), small (S), big (B), very big (VB).

Following the approach in [4], Go is set to a value indicating the maximum mark probability (e.g. 10%) that can also be adjusted in response of changes of the queue length.

The FIS uses linguistic rules to calculate the mark probability based on the input from the queues shown in Table 1. Usually multi-input FISs can offer better ability to linguistically describe the system dynamics. We expect that we should be able to tune the system for a better performance, and improve the behavior of the queue, by achieving high utilization, low loss and delay. The dynamic way of calculating the mark probability by the FIS comes from the fact that, according to the error of queue length for two consecutive sample periods, a different set of fuzzy rules apply. Based on these rules and inferences, the mark probability is calculated more dynamically than other AQM approaches [3, 4].

This point can be illustrated based on the decision surface of the FIS used in the FLC scheme. The decision surface and the linguistic rules shown in Table 1 provides some insights on the operation of FLC. The mark probability behaviour under the region of equilibrium (i.e., where the error on the queue length is close to zero) is smoothly calculated. On the other hand, the rules are aggressive about increasing the probability of packet marking sharply in the region beyond the equilibrium point. These rules reflect the particular views and experiences of the designer, and are easy to relate to human reasoning processes and gathered experiences.

Table 1. Rule Base – linguistic Rules

p(kT)		e(kT-T)				
		NB	NS	Z	PS	PB
	NB	B	B	VB	VB	VB
	NS	VS	S	S	B	VB
e(kT)	Z	Z	Z	VS	VS	S
	PS	Z	Z	Z	VS	VS
	PB	Z	Z	Z	Z	Z

Usually, to define the linguistic rules of a fuzzy variable, Gaussian, triangular or trapezoidal shaped membership functions are used. Since triangular and trapezoidal shaped functions offer more computational simplicity, we have selected them for the rule base. Then, the rule base is fine tuned by observing the progress of simulation, such as packet marking and delay occurrences, and throughput curves. Any gain in throughput must be traded off by a possible increase in the delay experienced at the terminal queues. Alternatively, an adaptive fuzzy logic control method can be used, which is based on tuning the parameters of the fuzzy logic controller on line, using measurements from the system. The tuning objective can be based on a desired optimization criterion, for example, a trade-off between maximization of throughput with minimization of end-to-end delay experienced by the users.

The design of FLC aims to generally provide better congestion control and better utilization of the network, with lower losses and delays than other AQM schemes [3,4], especially by introducing additional input variables and on-line self-tuned rule base.

5. Simulation Results

In this section the performance and robustness of the proposed FLC AQM in a set of environments are evaluated, and compare with A-RED [4], and PI controller [5]. The version 2.1b9a of NS-2 [9] simulator was used in simulations. The network topology used is shown in Figure 4. The TCP/Reno with an advertised window of 240 packets is used.

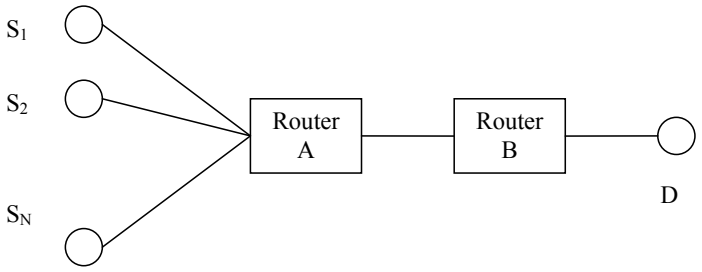

Figure 4: Network Topology

The size of each packet is 1000 bytes. The buffer size of all queues is 500 packets. AQM is used in the queues of the bottleneck link between router-A and router-B. All other links have a simple TD queue. All sources S_1, S_2,…, S_N with N flows are greedy sustained FTP applications and in the last simulation a web-like traffic is introduced. The links between all sources and router-A have the same capacity and propagation delay pair (C1, d1), whereas the pairs (C2, d2) and (C3, d3) define the parameters of the bottleneck link between router-A and router-B, and the link between router-B and the destination D, respectively.

The sampling period for FLC AQM is fixed to 0.006 sec as in [5]. The desired queue size of all AQM schemes is set to 200 packets, as this is used in [5] (for A-RED, we set the minimum threshold to 100 packets, and the maximum to 300, giving an average queue size of 200 packets). The simulation time is 100 sec. In the first experiment is investigated the effect of the traffic load factor N, by increasing N from 100 to 200, 300 and 400. The expected queuing delay experienced at router-A is 106.7 ms (15Mbps link capacity corresponds to 1875 packets/sec; for a queue size of 200 packets the expected mean delay is 200/1875 =0.1067. From simulation it can be concluded by noticing the packet losses as traffic load increases, that FLC has the lowest drops. FLC has stable and low packet loss over large traffic load. A-RED has the largest drops with a large increase of packet loss with respect to higher loads.

In the next experiment the utilization of the bottleneck link with respect to the mean queuing delay was investigated. FLC outperforms other AQM schemes on both high utilization and low mean delay, exhibiting a more stable, and robust behavior. The other AQM schemes have a poor performance as the number of traffic load increases, achieving much lower link utilization, and large queuing delays, far beyond the expected value. Table 2 lists the statistical results of the mean queuing delay and its standard deviation. It is clear that FLC has the lowest variance in queuing delay, resulting in a stable and robust behavior. On the other hand, the other AQM schemes exhibit very large queue fluctuations with large amplitude that inevitably deteriorates delay jitter.

Table 2: Mean Delay and Standard Deviation for First Experiment

		Mean-Delay (ms)	**Standard Deviation (ms)**
100 sources	PI	120.40	55.80
	FLC	104.33	26.24
	A-RED	107.53	73.84
200 sources	PI	145.22	87.55
	FLC	113.25	31.27
	A-RED	112.55	51.29
300 sources	PI	172.21	97.25
	FLC	122.54	37.54
	A-RED	121.35	52.21
400 sources	PI	184.27	99.22
	FLC	125.23	31.35
	A-RED	157.14	58.57

Next the performance of AQM schemes are investigated by introducing additional web-like traffic that can be seen as noise-disturbance to the network. The number of flows is kept to 100 for FTP applications, with an additional 100 web-like traffic flows. Two specific values of the desired queue size (i.e., 100 and 200 packets) were chosen to examine the robustness of the AQM schemes. For both cases the results are shown in Table 3 as the mean queuing delay, its standard deviation and packet losses.

Table 3: Mean Delay and Standard Deviation and Packets Drops for the Second Experiment

		Mean delay (ms)	Standard dev.(ms)	Drops(pack.)
Desired queue size 100	PI	71.60	45.27	1031
	FLC	56.71	22.62	169
	A-RED	57.15	41.68	1154
Desired queue size 200	PI	137.71	38.96	1023
	FLC	106.77	23.35	365
	A-RED	109.21	69.27	2715

FLC keeps a queuing delay close to the expected one with the lowest variance, while it exhibits the highest link utilization with the lowest drops. The other AQM schemes exhibit very large variations of the queue; consequently, this has the effect of having degraded link utilization with large number of drops.

6. Conclusions

In this paper an advanced active queue management algorithm, based on fuzzy logic is presented in comparison with some other AQM approaches. This algorithm for explicit marking is implemented in TCP/IP networks, using fuzzy logic techniques, in order to provide effective congestion control by achieving high utilization, low losses and delays. The proposed scheme is contrasted with a number of well-known AQM schemes through a wide range of scenarios.

The proposed fuzzy logic approach for congestion control is implemented with marking capabilities (either dropping a packet or setting its ECN bit). The design of the fuzzy knowledge base is kept simple, using a linguistic interpretation of the system behavior.

FLC controller is shown to exhibit many desirable properties, like robustness and fast system response, and behaves better than other AQM schemes in terms of queue fluctuations and delays, packet losses, with capabilities of adapting to high variability and uncertainty in network.

REFERENCES

1. Network Simulator, NS-2, Homepage, http://www.isi.edu/nsnam/ns/.
2. BRADEN, B**., Recommendations on Queue Management and Congestion Avoidance in the Internet**, IETF RFC2309, April 1998.
3. FENG W, KANDLUR D, SAHA D, SHIN K., **Blue: A New Class of Active Queue Management Algorithms**. Technical Report UM CSE-TR-387–99, 1999.
4. FLOYD S., V. JACOBSON, **Random Early Detection gateways for congestion avoidance**, IEEE/ACM Trans. on Networking, Aug. 1993.
5. FLOYD,S., R. GUMMADI, S. SHENKER, **Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management**, Technical report, ICSI, August 2001.
6. HOLLOT,C.V., V. MISRA, D. TOWSLEY, W.-B. GONG, **Analysis and Design of Controllers for AQM Routers Supporting TCP Flows** IEEE Trans. on Automatic Control, (47), 6, pp. 945-959, 2002.
7. PEDRYCZ, W., **Computational Intelligence: An Introduction**, CRC Press, 1998.
8. PITSILLIDES,A., A. SEKERCIOGLU, **Congestion Control**, Computational Intelligence in Telecommunications Networks, (Ed. W. Pedrycz, A. V. Vasilakos), CRC Press, pp.109-158, 2000.
9. PLASSER, E., T. ZIEGLER AND P. REICHL, **On the Non-linearity of the RED Drop Function**, in Proc. of Int. Conference on Computer Communication (ICCC), August 2002.
10. RAMAKRISHNAN, K., S. FLOYD, **The Addition of Explicit Congestion Notification (ECN) to IP**, IETF RFC 3168, September 2001.
11. SEKERCIOGLU, A., A. PITSILLIDES, A. VASILAKOS, **Computational intelligence in management of ATM networks**, Soft Computing Journal, 5(4), pp. 257-263, 2001.
12. ZIEGLER,T., S. FDIDA, AND C. BRANDAUER, **Stability Criteria of RED with TCP Traffic,** in Proc. of IFIP ATM&IP Working Conference, Budapest, 2001.