The Journal of
Mathematics and Computer Science

Available online at
http://www.TJMCS.com

The Journal of Mathematics and Computer Science Vol .2 No.3 (2011) 436-447

Design Novel AQM Schemes By Using Artificial Intelligence Technologies

S. Ghasempour[1], M. Hedayati[2], S. H. Kamali[3], R. Shakerian[4]
Department of Mathematics, Payam Noor University, Amol, Iran,
S_Ghasempoor@mpnu.ac.ir
Islamic Azad University, Ghaemshahr Branch,
Hedayati_maysam@yahoo.com
Islamic Azad University, Qazvin Branch,
Prjkamali@live.com
Young Researchers Club, Islamic Azad University, Ayatollah Amoli Branch,
R.shakerian@gmail.com

Received: July 2010, Revised: October 2010
Online Publication: January 2011

Abstract
Due to the dynamic nature and complexity of TCP congestion control, the AQMs leave some opportunity for improvement. The objective of this paper is to design novel AQM schemes which achieve efficiency and robustness by using AI technologies, in particular FL. In this paper, we elaborate on the approach of developing AQM using FL. First, we present our AQM design and innovations in terms of the traffic load factor and the application of FL for AQM. After describing the structure of a generic FL controller (FLC) which directs an FLC design, the two proposed FL-based AQM (FLAQM) algorithms are then presented to realize proactive queuing in turn. Finally we show the analysis of the efficiency and feasibility of our proposed FLAQM algorithms.

Keywords: AQM, FLAQM Algorithm, Traffic Load Factor, Generic FLC.

1. Introduction
The size and speed of the Internet have been growing ever since its inception in the 1960s, and so has the complexity of its traffic. With the emergence of optical fibers And

[1] Corresponding author: Saber Ghasempour, Department of Mathematics, Payam Noor University, Amol, Iran.
[2] Islamic Azad University, Ghaemshahr Branch, Ghaemshahr, Iran.
[3] Islamic Azad University, Qazvin Branch, Qazvin, Iran.
[4] Young Researchers Club, Islamic Azad University, Ayatollah Amoli Branch, Amol, Iran.

microprocessors running at billions of instructions per second, the future Internet Would be expected to have high speed with cheap and infinite bandwidth, and no Communication delay other than the speed of light. However, this is unlikely at least in the medium term [1]. There are a variety of network topologies, protocols, and traffic patterns. Traffic control thus has to be in place for dealing with congested links, and to provide a certain quality of service (QoS) to meet different user requirements and preferences.

Despite its immensity and heterogeneity, the Internet is remarkably stable. This Robustness of the Internet can be attributed to the widespread Transmission Control Protocol (TCP) protocol. The first congestion collapse in the mid 1980s was solved by the congestion avoidance mechanisms introduced by Van Jacobson [2]. These mechanisms took the form of a modification of the TCP protocol, and this modification is often referred to as TCP's congestion avoidance strategy. Since then, traffic control has been the focus of researchers in the networking area. The current Internet primarily deploys TCP end-to-end congestion avoidance algorithms to manage traffic and prevent any congestion collapses. It is worth mentioning that the other important transport protocol, UDP (User Datagram Protocol), does not have a congestion avoidance strategy, but it is expected that users will not resend lost UDP packets. The policy, which is not enforced, is also part of the solution of the congestion collapse problem of the Internet.

Due to the dynamic nature and complexity of TCP congestion control, the AQMs leave some opportunity for improvement. Fuzzy logic (FL) has been used in various areas and achieved many successes. Research has revealed that there are a numbers of areas in traffic control where we can explore the use of artificial intelligence (AI) technologies such as FL. The objective of this paper is to design novel AQM schemes which achieve efficiency and robustness by using AI technologies, in particular FL. In this paper, we elaborate on the approach of developing AQM using FL. First, we present our AQM design and innovations in terms of the traffic load factor and the application of FL for AQM. After describing the structure of a generic FL controller (FLC) which directs an FLC design, the two proposed FL-based AQM (FLAQM) algorithms are then presented to realize proactive queuing in turn. The performance evaluation of the two FLAQM algorithms in comparison to that of Drop Tail and some widely accepted AQM methods mentioned is conducted via extensive simulations. Finally, the discussion of the parameter configuration of FLAQM is given.

2. Design Rationales

A common principle used by most existing AQM schemes is match rate clear buffer; this implies the simultaneous achievement of low queuing delay and high link utilization. With this control principle in mind, we have also considered the other two issues in AQM design: selection of congestion indicators and calculation of packet dropping probability.

2.1 Traffic Load Factor

Most existing AQM schemes have deployed queue length or input rate, or both, for Congestion indication. Our approach is to design a new concept called traffic load factor. The traffic load factor, denoted as z, is the ratio of input rate to target capacity, where target capacity or expected traffic input rate is the leftover link capacity after draining the remaining packets in an output buffer. If input rate is measured at fixed intervals, the mathematical definition of z is given as follows.

$$z = \frac{input\_pkts}{t\arg et\_capacity}$$

$$t\arg et\_capacity = fraction \times link\_capacity$$

$$fraction = \begin{cases} \max(\gamma, 1 - \frac{Q - Q_0}{link\_capacity}) & Q > Q_0 \\ 1 & 0 \le Q \le Q_0 \end{cases} \quad (1)$$

$$link\_capacity = bdw \times dur$$

Where *input pkts* is the total input packets during the measurement period *dur*, γ is the maximum percentage of link capacity for draining the existing queue, *bdw* is the link capacity, Q is the instantaneous queue length at the end of the last measurement period, and Q_0 is a predefined target queue length. Note that Q_0 can be gained from a specified queuing delay by a network operator.

The traffic load indicator is a function of queue length and input rate. Using such a variable enables detection of impending congestion and reflects congestion severity. The adoption of the load factor as a congestion indicator is inspired by a successful Rate-based scheme for controlling ABR flows in ATM networks in [3]. There are some advantages to using z as a congestion indicator. First, z is dimensionless so that a control algorithm based on such a measure is robust against link capacity changes [4]. Second, to gain z, target capacity is first computed by the capacity used for clearing the buffer subtracted from the link capacity, i.e. target capacity = link capacity - capacity for clearing buffer. The calculation of target capacity is easily extended to deal with the scenario where best-effort traffic coexists with other QoS traffic with reserved bandwidth and available capacity for best-effort traffic is Ever-changing.

2.2 Application of Fuzzy Control for AQM

A TCP/IP network can be regarded as a feedback control system with a QM controller and the TCP/QM traffic plant. Control theory thus can be applied for the design of AQM. Some approaches based on classical control theory have been proposed such as PI [5] and VS [6]. It is pivotal for the schemes in this category to establish a mathematical model for TCP dynamics. However, there are some inherent limitations in the modeling process. Simplified models, such as through linearization, are required by certain control technologies. Some assumptions, such as exclusive long-lived connections and a delay-free control system, apply. Additionally, some parts of TCP dynamics are ignored, for example the slow-start phase and timeout of TCP. Therefore, any classical control-theory-based approaches potentially fail to achieve good performance and system stability. The weakness of classical control theory in the design of AQM is due to the complex and nonlinear nature of the control system and the difficulty in mimicking the dynamic behavior of the TCP/AQM plant in the form of any sound mathematic models.

A TCP/IP network is undoubtedly a complex nonlinear system. The complexity and nonlinear features result from many factors including the nonlinear property of TCP dynamics, dynamic traffic mix, and network heterogeneity ranging from the RTTs and the life-time a connection is experiencing, to protocols at each network layer, to different TCP versions. Given such a complex nonlinear control, FL is a better alternative control solution. In fuzzy control theory, nonlinearity is handled by rules, membership functions, and the inference process (these concepts will be given later), which results in improved performance and system stability with simpler implementation and reduced design costs. The idea behind FL is to emulate the cognitive inference process that human beings deploy in their decision-making and problem-solving in a way that input signals stimulate logic inference utilizing human heuristic knowledge and certain actions are taken accordingly. FL thus is able to apply expert knowledge to solve complex nonlinear problems without the need for precise and comprehensive information and the mathematical model of controlled objects. Many applications have been found for the use of

FL for traffic/congestion control in computer networks. For instance, in ATM networks, FL approaches range from ABR flow control [7], to frame discard mechanisms [8], to policing [9, 10, 11]. However, to our best knowledge, there are only a few studies in AQM using FL [12, 13].

Combining the use of the traffic load factor for congestion notification and FL to yield packet dropping probability, two FLAQM algorithms are derived. The first proposed FLAQM, FLAQM (I), uses z as an input directly, while the second FLAQM (II) uses its reciprocal z' instead. In addition, the changes of z and z' are used to capture traffic load trend in FLAQM (I) and FLAQM (II), respectively. The intention of using the reciprocal of z in FLAQM (II) is to implicitly realize the input normalization to achieve system stability and robustness.

3. A Generic FLC

The idea behind FLCs is to provide a means of converting a linguistic control strategy based on domain expert knowledge into an automatic control strategy in environments where either the processes are too complex for analysis by conventional quantitative techniques, or the available sources of information are interpreted qualitatively, inexactly, or uncertainly [14, 15]. A generic FLC comprises four building blocks: (1) a fuzzification interface, (2) an inference engine, (3) a knowledge base, and (4) a defuzzification interface as shown in Fig. 1. Note that in practice the values of the process states inputting to an FLC are crisp and the control outputs also require a crisp value. The functionalities of each block are given as follows. For more details refer to [16, 17, 18, and 19].

The fuzzification interface performs two functions including scaling and fuzzification. It performs a scale transformation or an input normalization, which maps the physical values of the process state variables or input variables into a normalized universe of discourse (normalized domain). When a non-normalized domain is used then there is no need for the scaling function.

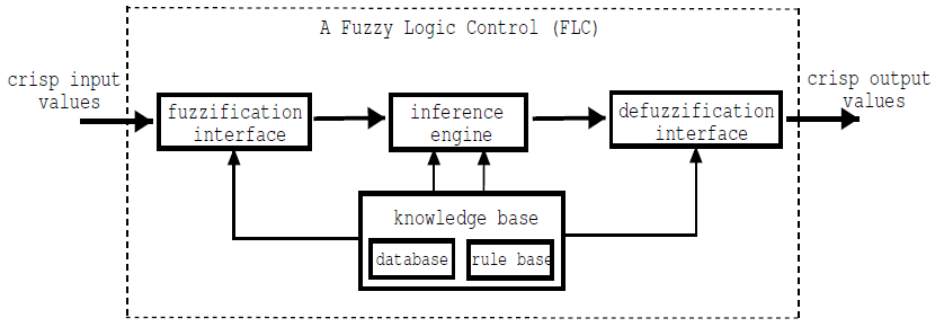


Figure 1. The structure of a generic FLC.

It performs the function of fuzzification that converts input data into a fuzzy set F, characterized with a membership function μF so that every element u from the universe of discourse U has a membership degree $\mu F(u) \in [0,1]$ to F. One widely adopted fuzzification strategy is 'singleton fuzzification', which produces a fuzzy set F for a crisp input u_0 with a membership function $\mu F(u)$ defined by

$$\mu F(u) = \begin{cases} 1, & u=u_0 \\ 0, & otherwise \end{cases} \quad (2)$$

The knowledge base provides a rule base and a database.

4. FLAQM

The FLAQM controller is designed to conduct queue management in a bottleneck output queue and control the behavior of the closed-loop feedback system as shown in Fig. 2. FLAQM calculates dropping probability pr based on the measurements from feedback signals. Two FLAQM algorithms are proposed in this study, namely FLAQM (I) and FLAQM (II). FLAQM (I) directly uses traffic load factor z and its change Δz as inputs, whereas FLAQM (II) inputs the values of the reciprocal of z, z' and its change $\Delta z'$. Note that valid feedback from the TCP/IP networks is delayed for at least one RTT for each connection. After packets reach their destinations, the corresponding ACK (acknowledgment) packets are received by their sources, and the sources respond to the dropping probability pr to adjust the amount of packets sent to the network.

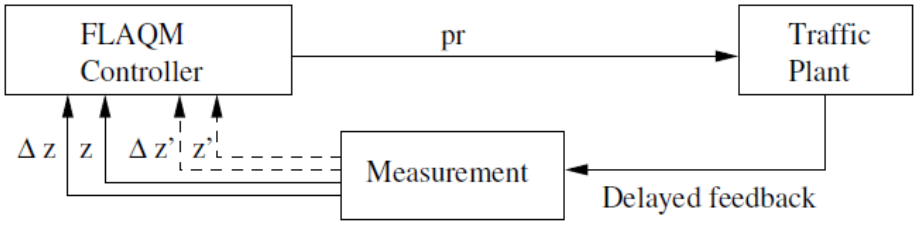


Figure 2. The closed-loop feed back system with the FLAQM controller.

Therefore, FLAQM is time-based in that measurement of feedbacks or the input values of the FLC and subsequent computation of the dropping probability *pr* are carried out in a fixed time interval. An appropriate value for the interval is carefully chosen to allow for transient traffic conditions with the arrival of bursty data on the one hand and the ability to react quickly to impending congestion on the other hand. Here the interval is set to be at least the maximum RTT of all the active connections to avoid system oscillation. The two proposals of FLAQM are presented in the following.

4.1 The FLAQM (I) Controller and Traffic Load Factor

FLAQM (I) aims to determine an appropriate value by which the routers drop the incoming packets based on the feedback information about traffic load and its trend. More specifically, the input variables of the FLAQM controller are traffic load factor z and its change Δz. It is clear that the set-point for the measured plant output, z, is 1 in that the input rate equals the target link capacity. Thus, the steady-state operating region toward which the FLAQM attempts to drive the closed-loop feedback system is in the neighborhood of z = 1. In order to achieve high link utilization of the network, the neighborhood of z = 1 is set as the range of $[1,1+\delta]$, where δ is a constant. The FLAQM controller copes with three cases in the network as shown in Fig. 3. If load factor z is beyond the set-point of the system, multiplicative decrease (MD) action is taken with negative Δz by using the MD FLAQM controller, while additive increase (AI) is applied with positive Δz by using the AI FLAQM controller. Otherwise, the traffic in the network is not overloaded and thus it is not necessary to do any extra control, but to add the incoming packets in the output queue. We have tried other designs, such as AIAD. However, the AIMD design appears to be steadier. The pseudo-code of dropping probability calculation in FLAQM (I) is given in Algorithm 1.

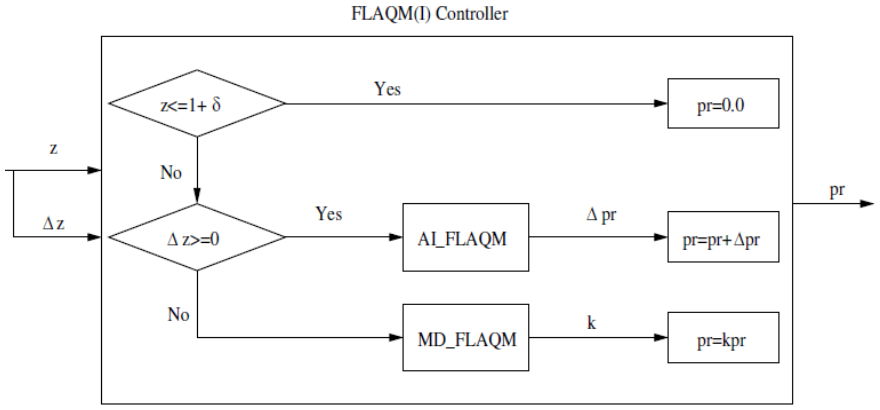


Figure 3. The structure of FLAQM (I).

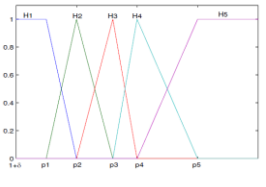
Algorithm 1 Dropping probability in FLAQM (I):

```
# pr is dropping probability
 if (z <= 1 + δ )
        Pr = 0.0
 else
  if ( Δz  < 0)
        k = MD_FLAQM (z, Δz )
        pr = k * pr
  else
     Δpr = AI_FLAQM (z, Δz )
        pr = pr +  Δpr
  endif;
endif;
```

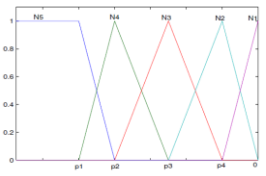
4.2 Design of two FLCs in FLAQM (I)

There are a set of membership functions for each FLC. For simplicity and effectiveness, triangular and trapezoidal shapes are chosen for these membership functions. Both FLCs use load factor z and Δz as inputs. For the MD_FLAQM controller, the output variable is coefficient k for a multiplicative decrease of pr, whereas for the AI FLAQM controller, Δpr is the output variable for an additive increase of pr. Fig. 4 shows the membership functions used in the MD FLAQM controller, whereas Fig. 5 shows the membership functions of the AI FLAQM controller. The linguistic values of the input z are *Hi,* $i = 1,2,...,5$ and traffic load increases with i. The input Δz is characterized by *Ni,* $i = 1,2,...,5$ where negative N specifies that the current traffic load has decreased when compared with its previous value and its magnitude increases with *i*, and *Pi,* $i = 0,1,...,5$ where positive *P* specifies that the traffic load is getting heavier than before and its magnitude increases with i. For the MD_FLAQM controller, the output k is described by *MDi,* $i = 0,1,...,5$ increasing with i, while the linguistic values of the output Δpr in the AI FLAQM controller are *AIi,* $i = 0,1,...,5$ also increasing with i. All the membership functions are characterized by their own shapes (triangle or trapezoid) and parameters indistinguishably denoted as *pi,* $i = 1 or 2,...$ such as $H1(-\infty,-\infty, p1, p2)$, $H2(p1, p2, p3)$, and $H5(p4, p5, \infty, \infty)$ Note that in the MD FLAQM controller, the effective universe of discourse for the input z is [p1, p5] and the counterpart for Δz is [p1, 0]. Likewise, in the AI FLAQM controller, a decision has been made for the effective universe of discourse for its inputs.

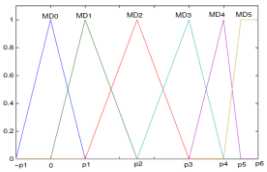
The end points of such an effective universe of discourse specify the "saturation points" at which the outermost membership functions are saturated for input universes of discourse, or beyond which the outputs will not move for the output universe of discourse [20]. The concept of effective universe of discourse makes intuitive sense as at some point the domain expert would just group all extreme values together in the linguistic description so that the membership functions at the outermost edges appropriately characterize "greater than" for the right side or "less than" for the left hand side.



(a) Membership functions for z

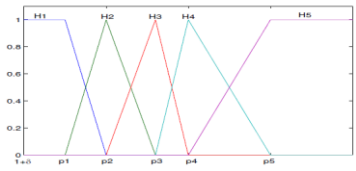


(b) Membership functions for Δz

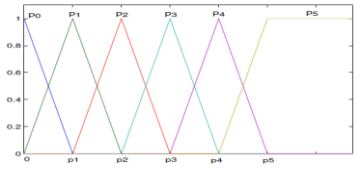


(c) Membership functions for k

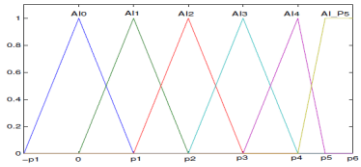
Figure 4. MD FLAQM in FLAQM (I).



(a) Membership functions for z



(b) Membership functions for Δz



(c) Membership functions for Δpr

Figure 5. AI FLAQM in FLAQM (I).

4.3 FL Rules in FLAQM (I)

Table 1 and Table 2 show the fuzzy if-then rules in the MD_FLAQM and AI FLAQM controllers respectively. FLAQM (I) operates upon a value of z beyond $1+\delta$, where impending congestion occurs. The principle of selecting fuzzy rules is the larger load factor z is away from the steady-state operating region $[1,1+\delta]$ and the more _z is away from zero, the more strong action is taken, and vice versa. For instance, in the case that z is around $1+\delta$ and Δz is only slight greater than 0, the slightly increased control action is taken for the dropping probability pr. So, we have if z is H1 and Δz is P0, then AI0. The other FL rules are derived in a similar way, which are obtained by expertise and the try and error method. The rule base of MD FLAQM consists of 25 fuzzy rules. Although we call the active controller under the situation of $z>1+\delta$ and $\Delta z<0$ a multiplicative decrease (MD) FLC, it does maintain or even increase dropping probability *pr* sometimes, based on network conditions. MD4 is set as an unchanged control action. The first thing is to judge under which situation of both inputs z and Δz the dropping probability pr would be unchanged. If input z is high and Δz is low, the control value of dropping probability pr has to be increased to cope with sustained high traffic load conditions. Otherwise, if traffic load dramatically reduces to a certain level, the decreased control signal of pr has to been taken. For the AI FLAQM controller, 30 fuzzy rules are adopted in its rule base.

In this FLC, the control value of pr is always increased. In the case that z is around $1+\delta$ and Δz is around 0, the slightly increased control action is taken. The FL rules are obtained by expertise and the try-and-error method. It is worth mentioning that the automation of the

control parameters defining the membership functions and even FL rules of the FLC System will be the future study of this project, since the preliminary investigation and application of FL for AQM has been successful.

Table 1. FL rules of MD FLAQM in FLAQM (I).

$\Delta z / z$	**H1**	**H2**	**H3**	**H4**	**H5**
N5	MD0	MD1	MD2	MD2	MD3
N4	MD1	MD2	MD3	MD3	MD4
N3	MD2	MD3	MD3	MD4	MD5
N2	MD3	MD3	MD4	MD5	MD5
N1	MD4	MD4	MD5	MD5	MD5

Table 2. FL rules of AI FLAQM in FLAQM(I).

$\Delta z / z$	**H1**	**H2**	**H3**	**H4**	**H5**
P0	AI0	AI0	AI1	AI1	AI2
P1	AI0	AI1	AI2	AI2	AI3
P2	AI1	AI2	AI2	AI3	AI4
P3	AI1	AI2	AI3	AI3	AI4
P4	AI2	AI2	AI3	AI4	AI5
P5	AI2	AI3	AI4	AI4	AI5

4.4 FL Rules in FLAQM (II)

Some fuzzy "if-then" rules are employed to capture the imprecise modes of reasoning that play an essential role in the human ability to make decisions in uncertain and imprecise environments. Table 3 and Table 4 shows the fuzzy rules in the MD FLAQM and AI FLAQM controllers, respectively. The rule base of MD FLAQM consists of 20 fuzzy rules. Although we call the active controller under the situation of $z' < 1+\delta$ and $\Delta z' > 0$ a multiplicative decrease FLC, it does maintain or even increase dropping probability pr sometimes, based on network conditions. MD4 is set as an unchanged Control action. The first thing is to judge under which situation of both inputs z' and $\Delta z'$ the dropping probability pr would be unchanged. Afterwards, if both inputs z' and $\Delta z'$ are low, the control value of dropping probability pr has to be increased to cope with sustained high traffic load conditions. Otherwise, if traffic load dramatically reduces to a certain level, the decreased control signal of pr has to been taken. For the AI FLAQM controller, 15 fuzzy rules are adopted in its rule base. In this FLC, the control value of pr is almost always increased. The exception is that in the case of $z' = 1+\delta$ and $\Delta z' = 0$, the unchanged control action is taken with plausible intuition. The principle of selecting these rules in Table 4 is that the more load factor z' is away from the steady-state operating region $[1,1+\delta]$ and the more $\Delta z'$ is away from zero, the more strong action is taken, and vice versa.

Table 3. FL rules of MD FLAQM in FLAQM (II).

$\Delta z' / z'$	**H1**	**H2**	**H3**	**H4**	**H5**
P1	MD7	MD7	MD6	MD5	MD4
P2	MD7	MD6	MD5	MD4	MD3
P3	MD6	MD5	MD4	MD3	MD2
P4	MD5	MD4	MD3	MD2	MD1

Table 4. FL rules of MD FLAQM in FLAQM (II).

$\Delta z' / z'$	**H1**	**H2**	**H3**	**H4**	**H5**
N1	AI7	AI6	AI5	AI4	AI3
N2	AI6	AI5	AI4	AI3	AI2
N3	AI5	AI4	AI3	AI2	AI1

5. Performance of FLAQM

In this section, we investigate the performance of the proposed FLAQM algorithms compared with that of the traditional Drop Tail (or DT), RED [51], and ARED [50]. The reason to choose RED to do performance comparison here is that RED is not only

a benchmark and the IETF default mechanism for buffer management, but also widely studied by the network research community and even Cisco System, a leading networking equipment supplier, has specified its RED implementation. Since the existing AQM schemes do not provide any significant advantage over Drop Tail for realistic traffic load models, ARED with the feature of auto-configuration is selected here as another representative AQM. Two simulation experiments have been carried out. In the simulations, the traffic pattern is composed of extremely long FTP connections which last the whole simulations, and Web traffic. By varying the number of active extremely long flows, performance comparison is conducted with different traffic loads in the first experiment, while changed traffic load conditions are simulated in the second experiment. In addition, all the traffic experiences the same propagation delay on the links from the server side to the client side with 40ms from the serves to R1, and 1ms from R3 to the clients.

The same traffic pattern has been input to the network with different queue management strategies in the bottleneck output buffer including Drop Tail, RED, ARED, and our proposed FLAQM algorithms. The parameter settings in the FLAQM (I) and FLAQM (II) controllers used in the simulations are shown in Table 5 and Table 6, respectively. Additionally, Fig. 6 and 7, and Fig. 8 and 9 illustrate the decision surfaces for AI FLAQM and MD FLAQM with these parameter choices in FLAQM (I) and FLAQM (II), respectively.

Table 5. FL rules of MD FLAQM in FLAQM (II).

Target Queue Length	60 Packets
Update Interval	0.5s
δ	0.05
Parameters of Z	P1=1.1, p2=1.5, p3=2.0, p4=2.5, p5=3.0
Parameters of ΔZ **in MD_FLAQM**	P1=-2.0, p2=-1.0, p3=-0.5, p4=-0.2

Parameters of Δz in AI_FLAQM	P1=0.2, p2=0.5, p3=1.0, p4=1.5, p5=2.0
Parameters of k in MD_FLAQM	P1=0.8, p2=0.85, p3=0.9, p4=1.0, p5=1.1, p6=1.15
Parameters of Δpr in AI_FLAQM	P1=0.01, p2=0.02, p3=0.03, p4=0.04, p5=0.05, p6=0.06

Table 6. Parameter settings of the FLAQM (II) controller.

Target Queue Length	60 Packets
Update Interval	0.5s
δ	0.05
Parameters of z'	P1=0.25, p2=0.5, p3=0.75, p4=1.05
Parameters of $\Delta z'$ in MD_FLAQM	P1=0.25, p2=0.5
Parameters of Δz in AI_FLAQM	P1=-0.5
Parameters of k in MD_FLAQM	P1=0.5, p2=0.95, p3=1.0, p4=1.05, p5=1.1, p6=1.15, p7=1.2
Parameters of Δpr in AI_FLAQM	P1=0.01, p2=0.02, p3=0.03, p4=0.04, p5=0.05, p6=0.06, p7=0.07

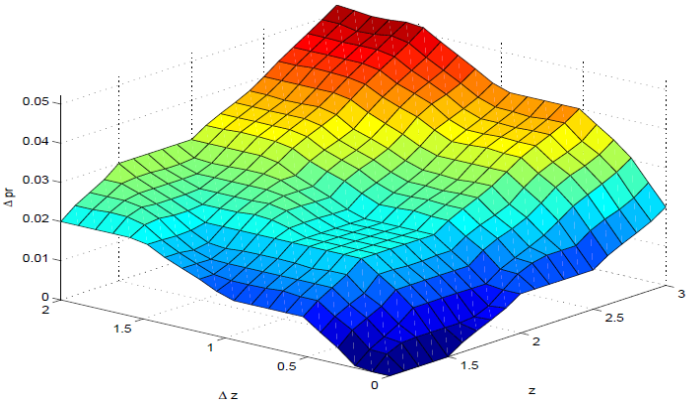


Figure 6. Decision surface of AI FLAQM in FLAQM (I).

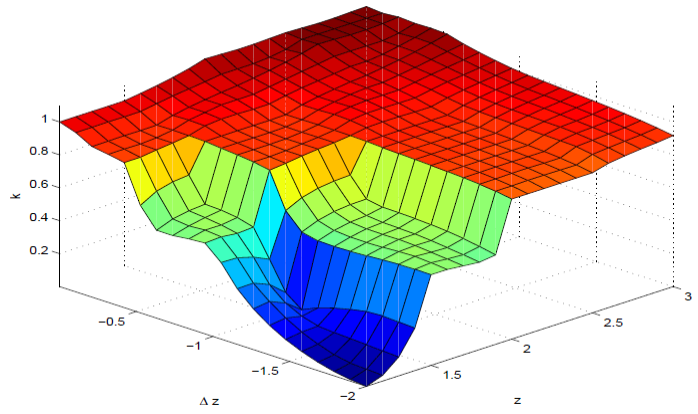


Figure 7. Decision surface of MD FLAQM in FLAQM (I).

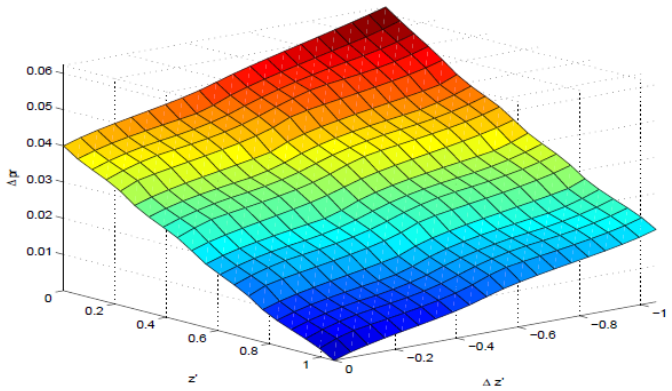


Figure 8. Decision surface of AI FLAQM in FLAQM (II).

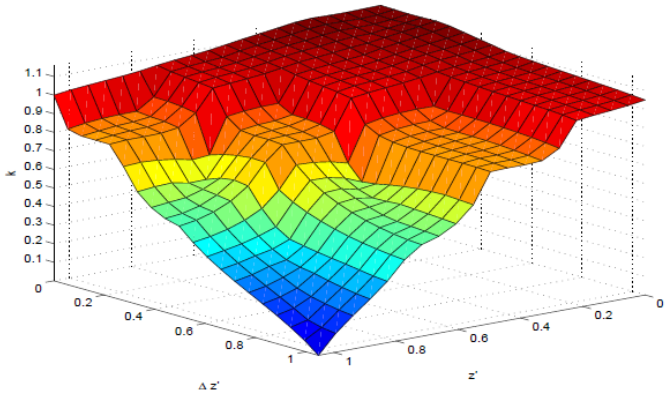


Figure 9. Decision surface of MD FLAQM in FLAQM (II).

6. Conclusion

The In this paper, we have proposed two novel AQM schemes: FLAQM (I) and FLAQM (II), by using one artificial intelligent (AI) method, fuzzy logic (FL). Via extensive simulations, both FLAQM schemes outperform the other well-known queue management strategies such as Drop Tail, RED, and ARED. Also, due to the achievement of scaling or normalization of the traffic load factor z' and its change $\Delta z'$, FLAQM (II) has improved the performance of FLAQM (I), in which normalization of its inputs is omitted since it is hard to know the maximum value of the inputs with different traffic load conditions. Although with an estimated approximate value for the maximum the performance of FLAQM (I) is reasonable, FLAQM (II) theoretically is expected to improve stability and performance. Thus, without specification, the term 'FLAQM' is used to stand for FLAQM (II).

One more point we would like to discuss here is the calculation of fraction in target capacity. At the end of each measurement period, when the Instantaneous queue length Q is less than or equal to a predefined value Q_0, fraction is set as 1 in the next period. Alternatively, the situation of $Q \le Q_0$ can be understood as more capacity left for accommodating incoming packets. However, this causes system oscillation in simulations. Thus, fraction = 1 is a better choice in the case of $Q \le Q_0$. More work still needs to be done to get an even better performance of FLAQM to realize the full potential advantages of FLC in complex nonlinear problem solving. Future work will consider the integration of other AI techniques such as neural networks (NNs)

and Genetic Algorithms (GAs) to achieve the self-tuning of the parameters in the two FLCs: MD FLAQM and AI FLAQM.

References

[1] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture:an Overview. RFC1633, 1994.
[2] V. Jacobson. Congestion Avoidance and Control. In Proceedings of ACM SIGCOMM, pages 314–329, 1988.
[3] S. Kalyanaraman et al. The ERICA Switch Algorithm for ABR Traffic Management in ATM Networks. IEEE/ACM Transactions on Networking, 8(1):87–98, 2000.
[4] G. Gopalakrishnan et al. Robust Router Overload Control Using Acceptance and Departure Rate Measures. In Proceedings of the 18th International Teletraffic Congress (ITC), Berlin, Germany, September, 2003.
[5] C. V. Hollot et al. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In Proceedings of IEEE INFOCOM, pages 1726–1734, 2001.
[6] P. Yan, Y. Gao, and H. Ozbay. Variable Structure Control in Active Queue Management for TCP with ECN. In Proceedings of ISCC, Kemer-Antalya, Turkey, June 2003. Accepted by IEEE Transactions on Control Systems Technology.
[7] R. Q. Hu and D. W. Petr. A Predictive Self-tuning Fuzzy-logic Feedback Rate Controller. IEEE/ACM Transactions on Networking, 8(6):697–709, 2000.
[8] H. H. Lim and B. Qiu. Performance Improvement of TCP using Fuzzy Logic Prediction. In Proceedings of International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), number 20 21, pages 152–156, Nashville, USA, November 2001.
[9] V. Catania et al. A Comparative Analysis of Fuzzy versus Conventional Policing Mechanisms for ATM Networks. IEEE/ACM Transactions on Networking, 4(3):449–459, 1996.
[10] A. Kandel et al. ATM Traffic Management and Congestion Control using Fuzzy Logic. IEEE Transactions on Systems, Man, and Cybernetics, Part C, 29(3):474–480, 1999.
[11] Z. Li and Z. Zhang. An Application of Fuzzy Logic to Usage Parameter control in ATM Networks. In Proceedings of the 1st International Conference on Fuzzy Systems and Knowledge Discovery: Computational Intelligence for the E-Age, 2002.
[12] C. Chrysostomou et al. Fuzzy Logic Congestion Control in TCP/IP Best Effort Networks. In Proceedings of the Australian Telecommunications, Networks and Applications Conference (ATNAC), Melbourne, Australia, December 2003.
[13] Z. Li et al. Improving the Adaptability of AQM Algorithms to Traffic Load Using Fuzzy Logic. In Proceedings of the Australian Telecommunications, Networks and Applications Conference (ATNAC), Melbourne, Australia, December 2003.
[14] C. C. Lee. Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part I. IEEE Transactions on Systems, Man and Cybernetics, 20(2):404–418, 1990.
[15] C. C. Lee. Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part II. IEEE Transactions on Systems, Man and Cybernetics, 20(2):419–435, 1990.
[16] J. R. Jang. ANFIS: Adaptive-Network-Based Fuzzy Inference System. IEEE Transactions on Systems, Man, and Cybernetics, 23:665–684, 1993.
[17] D. Driankov, H. Hellendoorn, and M. Reinfrank. An Introduction to Fuzzy Control (Second Edition). Springer, 1996.
[18] K. M. Passino and S. Yurkovich. Fuzzy Control. ADDISON-WESLEY, 1998.
[19] S. Floyd, R. Gummadi, and S. Shenker. Adaptive RED: An Algorithm for Increasing the Robustness of RED. Technical report, 2001. http://citeseer.nj. nec.com/floyd01adaptive.html.
[20] S. Floyd and V. Jacobson. Random Early Detection Gateways for Congestion Avoidance. IEEE/ACM Transactions on Networking, 1(4):397–413, 1993.