A comparative simulation study of TCP/AQM systems for evaluating the potential of neuron-based AQM schemes

Li, Fan; Sun, Jinsheng; Zukerman, Moshe; Liu, Zhengfei; Xu, Qin; Chan, Sammy; Chen, Guanrong; Ko, King-Tim

Published in:
Journal of Network and Computer Applications

Published: 01/05/2014

Document Version:
Post-print, also known as Accepted Author Manuscript, Peer-reviewed or Author Final version

Published version (DOI):
10.1016/j.jnca.2014.01.005

Publication details:
Li, F., Sun, J., Zukerman, M., Liu, Z., Xu, Q., Chan, S., Chen, G., & Ko, K.-T. (2014). A comparative simulation study of TCP/AQM systems for evaluating the potential of neuron-based AQM schemes. *Journal of Network and Computer Applications*, *41*(1), 274-299. https://doi.org/10.1016/j.jnca.2014.01.005

A comparative simulation study of TCP/AQM systems for evaluating the potential of neuron-based AQM schemes

Fan Li[a], Jinsheng Sun[b], Moshe Zukerman[a,*], Zhengfei Liu[b], Qin Xu[b], Sammy Chan[a], Guanrong Chen[a], King-Tim Ko[a]

[a] *Department of Electronic Engineering, City University of Hong Kong, Hong Kong*
[b] *Department of Automation, Nanjing University of Science & Technology, Nanjing, Jiangsu Province, PRC*

Abstract

In recent years, several neuron-based active queue management (AQM) schemes have been proposed. Such schemes exhibit important attributes including fast convergence with high accuracy to a desired queue length. This paper presents extensive comparative simulation results for four neural AQM schemes, namely, Neuron PID, AN-AQM, FAPIDNN, NRL, versus three traditional AQM schemes (ARED, REM and PI) together with a modified PI scheme named IAPI over a wide range of conditions and scenarios. For all schemes, we test their performance in various environments. Through extensive numerical comparisons, we demonstrate that the neuron-based schemes generally achieve faster convergence to queue length target, with smaller queue length jitter. We further demonstrate one order of magnitude reduction in the standard deviation of the end-to-end packet delay by the four neuron-based schemes over the other schemes, when the queueing delay is the dominant delay component. These advantages of neuronal schemes deserve recognition despite the fact that no proof of stability is available for such schemes.

Keywords: AQM, Internet congestion control, Neuronal system, Simulation

1. Introduction

Internet congestion control aims to achieve efficient resource utilization, acceptable packet loss and stable operation. At the early age of the Internet, the end-to-end Transmission Control Protocol (TCP) [1] addressed the congestion control problem by the combination of the following two mechanisms: Window-based Flow Control and Drop Tail.

*Corresponding author. Tel: +(852) 3442-4243 Fax: +(852) 3442-0562
Email address: m.zu@cityu.edu.hk (Moshe Zukerman)

At TCP sender side, window-based flow control mechanism comprises two phases, namely, slow start phase and congestion avoidance phase. Each connection maintains a congestion window *cwnd* [2], which limits the number of packets that can be sent at each round trip time (RTT). Initially, a connection is in the slow start phase, with *cwnd* being equal to 1. Upon successful transmission of every window of packets, *cwnd* is doubled. This process continues until *cwnd* has reached the predetermined threshold (*ssthresh*), then the sender switches to congestion avoidance phase. Otherwise, any packet loss during the slow start will trigger the sender to switch to the congestion avoidance phase. During the congestion avoidance phase, the sender dynamically adjusts *cwnd* according to Additive Increasing Multiplicative Decreasing (AIMD) algorithm [2]. Drop Tail mechanism is used inside IP routers. Data packets enter a router's buffer according to first-come-first-server manner. When the buffer becomes full, those overflowed packets are discarded based on Drop Tail mechanism. However, the combination of Drop Tail and window-based flow control can easily lead to TCP synchronization [3], in which different TCP connections experience packet loss and decrease their sending rates simultaneously, then the network becomes unstable and exhibits a high packet loss rate.

1.1. The Development of AQM Systems

In light of the aforementioned disadvantages, Active Queue Management (AQM) was proposed to overcome the shortcomings inherent in the above two mechanisms. Moreover, various real-time applications such as VoIP, video streaming, and online gaming are gaining popularity in today's Internet. These applications may tolerate some packet loss, but are more sensitive to delay and jitter. The User Datagram Protocol (UDP), which does not respond to congestion, is more suitable and has been widely adopted by these applications. If the proportion of real-time UDP traffic is low relative to non-real-time TCP traffic, the two traffic types can co-exist, so that the non-responsive real-time delay sensitive UDP connections achieve their target QoS because the non-real time TCP traffic may slow down their sending rates. In this way, both types of traffic achieve their QoS. In Section 3, we illustrate such interaction of UDP and TCP traffic streams. We note that if non-responsive UDP traffic becomes dominant, other measures such as connection (call) admission control [4] will be required. However, this is beyond the scope of the present paper.

In the past two decades, AQM has been an active research area with many schemes proposed and evaluated. (See for example [5–23] and references therein.) Random Early Detection (RED) [5, 24], the first AQM, compares the pre-defined threshold and the average queue size, calculated by a low-pass filter with an exponential weighted moving average. The dropping probability is then adjusted according to the average queue size: the larger the average queue size is, the higher the dropping probability will be. Athuraliya *et al.* [8] proposed Random Exponential Marking (REM) to improve RED by considering both queue length and link input rates (loads). Applications of control theory also led to some efficient AQM schemes. Hollot *et al.* [9] proposed a PI controller and provided guidelines for designing stable PI controllers. Ryu *et al.* [10] adopted a PID

controller to detect the incipient congestion and control the congestion proactively. In [12], Agrawal and Granelli proposed a queue length based scheme with a PID controller designed by a linear quadratic regulator method. We chose PD controller for AQM in [13] to improve stability and robustness against traffic load fluctuation. Additionally, PD controller was used as a supplementary mechanism of RED to accelerate transient response and decrease the queue length variance in steady state [14].

A number of schemes using heuristic methods to determine the dropping probability of incoming packets have also been proposed. In [15], packet loss and link idle events were used to determine the dropping probability of the incoming data packets. A rate based AQM scheme called AVQ [16] was designed to set up a virtual queue to provide early feedback. Proposed by Wang and Li *et al.*, RAQM [17] considered aggregate TCP sources' rate as a rate metric in both "queue dependent" and "queue independent" modes. Other works reported in the literature discussed and analyzed various effects in real networks and settings of controller parameters. For example, [18] discussed the nonlinear behavior of queue dynamics due to different TCP-RED parameter settings. In [19], an algorithm was proposed to alleviate the negative effects caused by large delays. In [21], an AQM scheme called GREEN was proposed to achieve low delay and packet drop probability without sacrificing the link utilization. Santi and Fonseca [22] designed optimal AQM schemes for TCP variants in high speed environment.

Despite the extensive AQM research effort, many AQM proposals do not achieve optimal congestion control operation, partly because they use fixed parameters which are not sufficiently adaptive to the time-varying and nonlinear network conditions. To mitigate the limitations of AQM schemes that are based on fixed parameters and to improve their performance, significant research has been conducted. For example, to improve the performance of RED, Zheng and Atiquzzaman [25] provided a method for determining the maximum drop probability of RED, based on a TCP channel model and traffic behavior. Tahiliani *et al.* [26] proposed the Cautious Adaptive RED (CARED) scheme to adapt Max_p based on the level of traffic load without increasing significantly the complexity of the original RED algorithm. In general, an effective and reliable AQM scheme should be able to regulate the queue to avoid congestion at bottleneck links. Furthermore, it should be able to tune its parameters "intelligently" according to the network and traffic conditions. Based on the above observations, we believe that applying the artificial neural systems to AQM is an effective way to achieve intelligence and self-adaptation to time varying environment.

1.2. Artificial Neural Networks, Machine Learning and AQM

Artificial neural network (ANN) models and machine learning algorithms have been widely used in the fields of networking and computer science to solve a wide range of problems that are not very amenable to solutions by conventional rule-based programming. ANNs are inspired by biological central nervous systems. ANNs can self-learn and self-adapt, and have shown ability in controlling nonlinear complex systems. An ANN consists of layers of interconnected

"artificial neurons" (herein called *neurons*). These neurons are the basic processing elements of an ANN. Figure 1 illustrates a multi-layer neural network. Each circle represents a neuron. As shown in the figure, there are three layers inside the network. The first two neurons on the left form the input layer which receives the input vector $X = [x_1, x_2]^T$. The three middle neurons manipulate the input data, form the hidden layer. The rightmost neuron collects the data from the hidden layer and then summing them together to yield the output. To represent the output, a common approach is a nonlinear neuron weighted sum which can be written as $y = K \sum_i \omega_i x_i$, where K is the neuron gain and ω_i is the neuron weight.

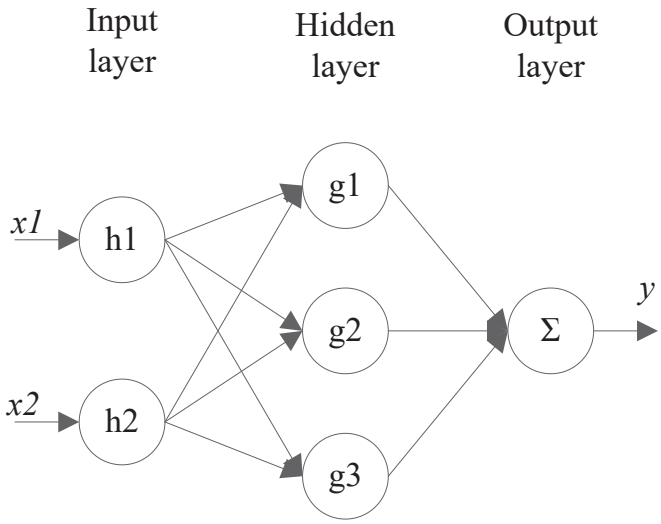

Figure 1: A Typical Three-layer Neural Network

The intelligence and self-adaption of ANNs are achieved by machine learning algorithms. There are three basic types of learning algorithms used by ANNs: supervised learning, unsupervised learning and reinforcement learning.

- **Supervised Learning**: In supervised learning, the training example pairs $(x_i,\ y_i)$ are given to the learner, where x_i and y_i represent input and output values, respectively. The learner learns by identifying patterns in the training data to predict a function f such that $f(x_i) = y_i$. To evaluate the mismatch of the general function f derived by the learner, the mean square error $\sum_i (y_i - f(x_i))^2$ is commonly used as a cost function. The goal of the supervised learning is to find the function f such that the mean square error $\sum_i (y_i - f(x_i))^2$ is minimized.
- **Unsupervised learning**: Unlike the supervised learner, the unsupervised learner is given the input data without the corresponding output,

namely, the input data are not "classified" or "tagged" beforehand. Therefore, the unsupervised learner itself needs to find the internal structure among the input data. One example of unsupervised learning is data clustering, which classify the data into groups according to their shared similarities.

- **Reinforcement Learning**: In reinforcement learning, the input data with the corresponding output are not given to the learner. Here, feedback (with positive or negative reward) from an external environment is used instead of the mean square error evaluation. The goal of reinforcement learning is to obtain the maximum reward from the external environment by tuning the neuron weights.

Readers are referred to [27] for more details on machine learning algorithms. Normally, AQM schemes cannot achieve optimal performance due to unpredictable traffic variability. Using ideas based on the above learning strategies has the potential to improve parameter tuning efficiency and system performance.

In recent years, there has been increasing interest in neural network based AQM schemes. In [28], the so-called Neural Network Model Predictive Control (NN-MPC) that controls TCP flows was proposed. NN-MPC exhibits better transient and steady behaviours over the classical PI controller. Hariri and Sadati [29] proposed a scheme called Neural Network-RED (NN-RED), which improves RED by using a neural network as a prediction tool to determine the future values of the queue size and dropping probability of incoming packets. Another advantage of NN-RED is that it involves less parameters than RED. Cho *et al.* [30] proposed an adaptive neural queue management for TCP transmission. This scheme classifies the network traffic condition into three different traffic scenarios: light traffic, medium traffic and heavy traffic, and it determines the output (incoming packet dropping probability) by a Dynamic Bayesian Network (BDN) stochastic process. Neuron PID [31] aims to adaptively tune the parameters of a PID controller. It uses the queue length error (the error between the actual queue length and the target queue length) as the neuron input. This scheme adopts the so-called associative learning which is based on both supervised and unsupervised learning. The particular unsupervised learning used is Hebbian Learning. Then, in the scheme called Adaptive Neuron AQM (AN-AQM) [32], the mismatch between the total sending rate of all sources and the capacity of the bottleneck (called *sending rate mismatch*) as well as the queue length error are used as neuron input. Such modification improves the performance in realistic environment including long delay networks and disturbance by UDP flows. Zhou *et al.* [33] implemented an AQM scheme named Neuron Reinforcement Learning (NRL). NRL uses both the queue length error and sending rate mismatch as the neuron input, and then it applies the reinforcement learning rule to derive an appropriate dropping probability as the output. Yan and Lei [34] implemented a Fuzzy Assisted PID controller based on Neuron Network (FAPIDNN). In FAPIDNN, the packet dropping probability is

derived by a neural network based PID controller, where the learning rate η of the neural network is calculated by a fuzzy controller.

1.3. Contribution and Organization

Publications that include survey of AQM schemes include [35, 36] and [37] which is the most comprehensive survey of AQM so far. Nonetheless, these survey papers do not consider the newly proposed neuron-based schemes discussed in this paper. Besides survey papers, several comparative studies have evaluated the performance metrics of various AQM schemes under different network conditions. Koo *et al.* [38] provided a comparative study of five AQM schemes (Drop Tail is employed as benchmark) using performance metrics including delay and loss rate. Reddy and Ahammed [39] compared RED and Drop Tail in two ways: throughput and fairness index. However, the existing comparative studies do not consider the neuronal AQM schemes studied here either.

In this paper, we aim to show the potential of neuronal AQM schemes by a simulation-based comparative study of AQM systems involving neuronal and traditional schemes. We consider performance metrics including link utilization, packet drop ratio, and mean and standard deviation of the queue length and end-to-end packet delay, respectively. Although Drop Tail is adopted by the current Internet, we do not include Drop Tail in our study as many existing studies have already demonstrated the significant performance improvement achieved by the traditional AQM schemes over Drop Tail. The focus of this paper is the performance comparison of AQM schemes considering link utilization, queue and delay statistics. Another important aspect is fairness. Note that all the AQM schemes considered in this paper do not aim to achieve fairness and mark the packets randomly without considerations to which flow they belong. To improve fairness, a certain discrimination between various types of flows is needed, e.g. to compensate flows with longer RTT. There are special AQM schemes which focus on the fairness issue such as FRED [40] and Choke [41], compromising efficiency and performance for the benefit of fairness. Although fairness is not a main focus of the paper, we have measured the simulation results using the Jains fairness index [42] for all the AQM schemes considered in the paper, and found them to be very close to each other. In particular, for various cases where all RTTs are the same, the fairness indices we have obtained were in the range of 0.86-0.92 and when the variance of the RTT increases (we considered two groups where the RTTs of one group are 20 times longer than the other) the indices for all schemes were in the range of 0.52-0.60. This is consistent with the fact that these schemes do not discriminate between flows to achieve fairness.

Given the large number of available AQM schemes, we consider a representative sample of key traditional schemes, plus a recently proposed scheme that provides an example of enhancement of the traditional schemes, plus four neuron-based schemes. In particular, we consider the following eight AQM schemes: ARED [6], REM [8], PI [9], IAPI [43], Neuron PID [31], AN-AQM [32], FAPIDNN [34] and NRL [33]. RED, REM and PI are key milestones in the history of the development of AQM schemes, and the choice of ARED instead of RED is because the former is known to improve the performance of

the latter. Also, our choices make this paper consistent with other AQM comparative studies such as [44]. IAPI is a modification of the PI controller, which achieves faster convergence rate and smaller queue length jitter. The remaining four AQM schemes are neuron-based, which exhibit very fast convergence and accuracy.

The remainder of the paper is organized as follows. Section 2 gives a brief description of each AQM scheme selected for comparison. Section 3 provides simulation results of each AQM scheme in various simulation conditions including instantaneous queue length plotted by figures and numerical performance metrics shown by tables. Finally, Section 4 concludes the paper.

2. AQM Schemes Selected for Comparison

In this section, we provide a brief description of each one of eight AQM schemes.

2.1. ARED

Random Early Detection (RED) compares the average queue length avg to two predefined thresholds: min_{th} and max_{th}. The average queue length avg is calculated by the following Exponential Weighted Moving Average (EWMA) algorithm

$$avg = (1 - w_q)avg + w_q * q, \tag{1}$$

where w_q is a predefined weight coefficient. Then the dropping probability p of the incoming packets depends on the degree of congestion level reflected by avg, which obtained by

$$p = max_p(avg - min_{th})/(max_{th} - min_{th}), \tag{2}$$

where max_p is predefined maximum dropping probability.

The maximum dropping probability max_p and queue weight w_q are fixed in original RED. The Adaptive RED (ARED) algorithm auto-tunes these two parameters dynamically.

Algorithm 1 [6] below describes how the maximum dropping probability max_p is modified in every fixed interval (0.5 second) by comparing the average queue size (avg) and the target queue size.

The predefined weight coefficient w_q of ARED is calculated by the following equation

$$w_q = 1 - \exp(-1/C), \tag{3}$$

where C [packet/sec] is the link capacity. As in [6], we assume that all packets are of a specified default size. The original RED set the weight coefficient w_q as a fixed value equal to 0.002, while ARED tunes this parameter as a function of the link capacity.

Algorithm 1 The auto-tuned parameters of ARED

Fixed parameters:
interval: time (0.5 seconds);
β: decrease factor (0.9);

Variables:
avg: average queue size calculated by EWMA algorithm;
target: target for *avg*;
α: increment; $min(0.01, max_p/4)$;

Auto-tuned parameters:
For every *interval* seconds:
if ($avg > target$ and $max_p \leq 0.5$) **then**
  increase $max_p \leftarrow max_p + \alpha$
else if ($avg < target$ and $max_p \geq 0.01$) **then**
  $max_p : max_p \leftarrow max_p * \beta$
end if

2.2. REM

Random Exponential Marking (REM) determines the dropping probability by considering the combination of queue length error and sending rate mismatch, such combination is named as "price". The probability updating process is as follows

$$p(t+1) = Max(0, p(k) + \gamma(\alpha_l(q(k) - q_{ref}) + x_l(k) - c_l(k))), \tag{4}$$

where γ and α_l are constant coefficients, $q(k) - q_{ref}$ is the queue length error and $x_l(k) - c_l(k)$ is the sending rate mismatch.

2.3. PI

PI AQM [9] is based on the PI controller that updates the dropping probability as in the continuous form

$$p(t+1) = p(t) + k_p(q(t) - q_{ref}) + k_i \int_0^t (q(t) - q_{ref})dt, \tag{5}$$

where $q(t)$ is the instantaneous queue length, q_{ref} is the target queue value, k_p and k_i are the proportional and integral coefficients, respectively.

In practice, the discrete-time version is used. In particular, the dropping/marking probability of the PI controller for the kth time-interval is recursively given by

$$p(k) = p(k-1) + \Delta p(k), \tag{6}$$

and the increment of the dropping probability $\Delta p(k)$ is updated by

$$\Delta p(k) = K_p(e(k) - e(k-1)) + K_I T e(k), \tag{7}$$

where $e(k)$ is the queue length error derived by $q(k) - q_{ref}$; K_p and K_I are the proportional and integral coefficients, respectively; and T is the sampling period of the AQM system.

2.4. IAPI

While the parameters k_p and k_i in PI controller are fixed, to achieve faster convergence to the queue length target, Intelligent Adaptive PI (IAPI) improves adaptability in the following three ways:

1. IAPI introduces the adaptive integral coefficient k_i adaptively determined by the following equation

$$k_i = k_{i0}(1 + \frac{|e(k)| - e^*}{2e^*}), \tag{8}$$

   where e^* is the preset error threshold, $e(k) = q(k) - q_{ref}$ is the queue length error, k_{i0} is the integral coefficient of original PI controller [9].
2. IAPI constantly monitors the queue length error and determines k_i to achieve fast convergence to the target queue length.
3. IAPI achieves smaller fluctuation of queue length at steady state by updating the dropping probability based on every packet's arrival rather than a fixed time interval, using the formula $p'(t) = \frac{q(t)}{q_{ref}} p(t)$.

2.5. Neuron PID

Neuron PID [31] is essentially a PID controller but with neuron tuned weights, which overcomes the limitation of the fixed parameters of a pure PID controller. Neuron PID adopts the associative learning strategy including Hebbian learning and supervised learning.

Figure 2 shows the block diagram of Neuron PID. The PID transfer converts the queue length error $e(k)$ into three components: proportional component $x_1(k) = e(k) - e(k-1)$, integral component $x_2(k) = e(k)$ and derivative component $x_3(k) = e(k) - 2e(k-1) + e(k-2)$.

Inside the neuron, according to Hebb [45], the learning rule for a neuron is formulated as

$$w_i(k+1) = w_i(k) + d_i y_i(k) \tag{9}$$

where $d_i > 0$ is the learning rate and $y_i(k)$ is the learning strategy. The associative learning strategy is as follows

$$y_i(k) = e(k)p(k)x_i(k). \tag{10}$$

By (9) and (10), we obtain

$$w_i(k+1) = w_i(k) + d_i e(k)p(k)x_i(k) \tag{11}$$

where $e(k)$ is used as teacher's signal, $p(k)$ is the neuron output (also the dropping probability for the TCP plant).

The reclusive relationship for the dropping probability of the Neuron PID scheme is written as

$$p(k) = p(k-1) + \frac{K \sum_{i=1}^{3} w_i(k) x_i(k)}{\sum_{i=1}^{3} w_i(k)} \tag{12}$$

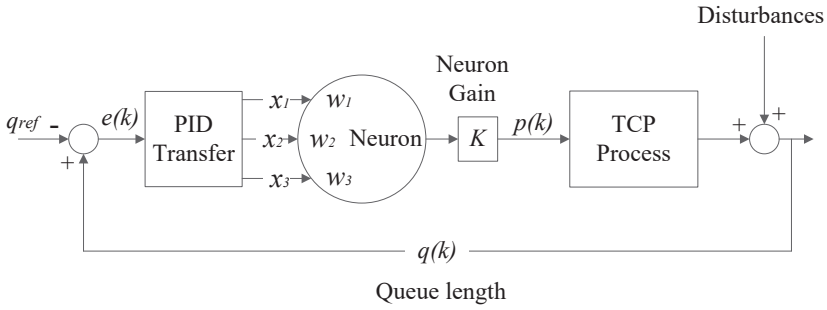

Figure 2: The block diagram of Neuron PID

2.6. AN-AQM

Adaptive Neuron AQM (AN-AQM) [32] is an extension of Neuron PID, In addition to considering queue length error, it also considers normalized sending rate mismatch $\gamma(k) = \frac{r(k)}{C} - 1$ as a congestion measurement, where r(k) is the input rate of the buffer at the bottleneck link, C is the capacity of the bottleneck link. Compared to Neuron PID, AN-AQM extends the inputs of neuron from three to six including the other three inputs of sending rate mismatch. By considering both queue length error and sending rate error, AN-AQM slightly improves the performance of Neuron PID in more realistic environment such as long delay network and varying TCP connections.

Figure 3 provides the block diagram of AN-AQM. Observe that the normalized sending rate mismatch $\gamma(k)$ is converted into three corresponding PID components, x_4, x_5 and x_6.

2.7. FAPIDNN

Fuzzy Assisted PID controller based on Neuron Network (FAPIDNN) employs a neural network to tune parameters of the PID controller. FAPIDNN differs from Neuron PID (and AN-AQM) in that FAPIDNN adopts fuzzy controller to dynamically tune the learning rate of the neural network instead of associative learning. FAPIDNN considers q_{ref} and $q(k)$ as inputs.

Figure 4 shows the block diagram of FAPIDNN scheme. The output of Fuzzy controller $\eta(k)$ is the learning rate of the Neuron PID controller. The operation of fuzzy controller is given in [34].

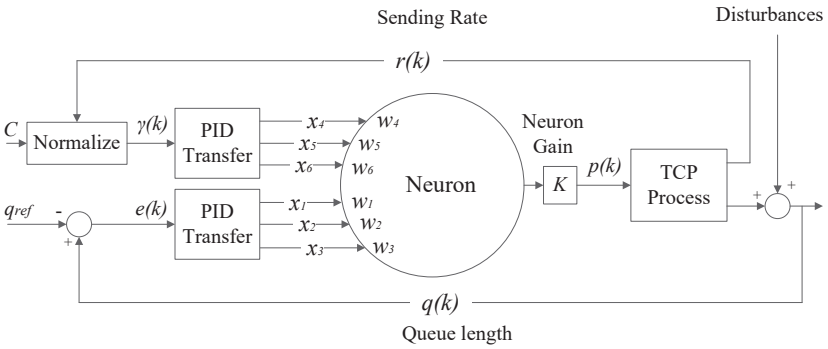

Figure 3: The block diagram of AN-AQM

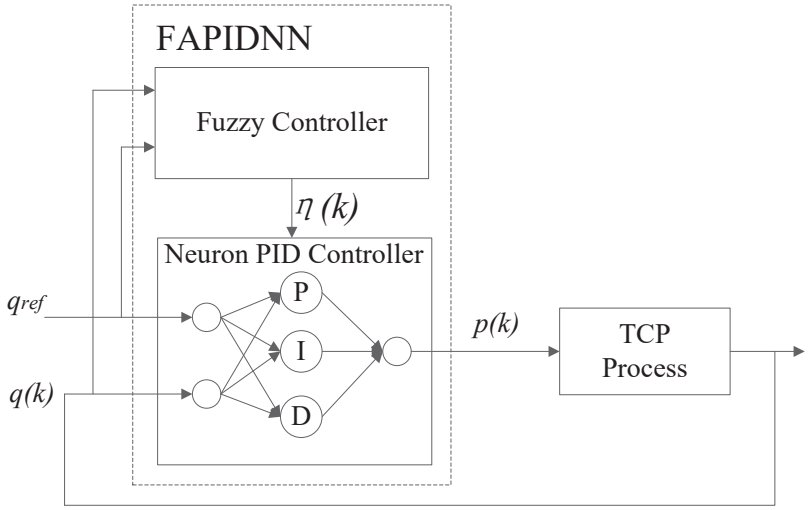

Figure 4: The block diagram of FAPIDNN

2.8. NRL

Neuron Reinforcement Learning (NRL) [33] is different from three above neuron-based AQM schemes, it does not introduce PID controller to preprocess the input of neuron, while it directly uses the neuron reinforced learning to adjust the TCP/AQM control plant. NRL considers both queue length error and sending rate error as the neuron inputs. The weights of the neuron w_1 and

w_2 are adjusted in real time by reinforcement learning rule, which, according to simulation results, improves the queue length stability in steady state.

Figure 5 shows the block diagram of NRL scheme. In the figure, vector $z(t)$ is the neuron output vector which is the product of neuron weight vector $\omega(t)$ and input vector $e(t)$

$$z(t) = \omega_1(t)[q(t) - q_{ref}] + \omega_2(t)[x(t) - C] = \omega^T(t)e(t), \tag{13}$$

where vector $e(t)$ contains two parts: the queue length error $e_1(t) = q(t) - q_{ref}$ and sending rate mismatch $e_2(t) = x(t) - C$. Then vector $z(t)$ is processed by the activation function $f(\cdot)$, where the activation function is given by

$$f(z(t)) = y(t) = \frac{1 - e^{-z(t)}}{1 + e^{-z(t)}}. \tag{14}$$

The activation function $f(\cdot)$ has the output $y(t) \in [-1, 1]$. Then $y(t)$ is truncated into $[0, 1]$, which serves as the dropping probability.

The reinforce learning strategy is applied to adjust the neuron weights vector $\omega(t)$.

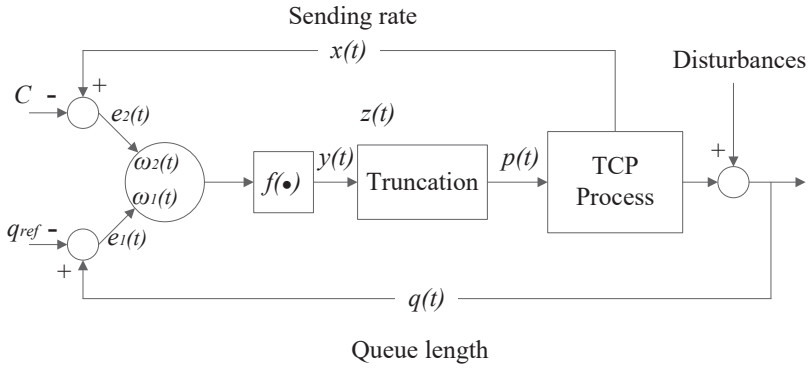

Figure 5: The block diagram of NRL

3. Performance Evaluation and Comparison

In this section, we provide extensive *NS2* [46] simulation results to demonstrate the performance attributes of the aforementioned eight AQM schemes. Our simulations and comparisons will examine the following attributes:

1. the ability to control the queue length to quickly converge to a given queue length target;

2. robustness to different round trip propagation time (RTPT) and effectiveness for a long delay network;
3. robustness to traffic loading under fixed and dynamic TCP connections scenarios, to bottleneck link capacity, and to impact of traffic noise (UDP);
4. performance in the topology with multiple bottlenecks.

These attributes of each AQM scheme are evaluated by the following five metrics: 1) instantaneous queue length; 2) the efficiency of the bottleneck link; 3) packet drop ratio; 4) average queue length and 5) standard deviation of the queue length. The instantaneous queue length illustrates whether an AQM scheme is able to stabilize the queue length to the target Q_T and how much time this stabilization takes. Efficiency of the bottleneck link is defined as the ratio of the average number of data packets passing through the bottleneck link per second to the capacity of the bottleneck link in terms of packets per second. Packet drop ratio in a given link is defined as the ratio of dropped packets to the total number of packets that arrive at the link. In the following simulations, the instantaneous queue length will be shown by figures, and the other four performance metrics will be shown by tables.

Another important performance metric is the mean packet delay including retransmissions subject to a given propagation time. We will compare the various AQM schemes in terms of their mean and standard deviation of the end-to-end packet delay.

3.1. Single bottleneck topology

The single bottleneck network topology used in the simulation is shown in Figure 6. The only bottleneck link is the common link between the two routers. The other links are assumed to have sufficient capacity to carry their traffic. Router B uses the tested AQM scheme (among ARED, REM, PI, IAPI, Neuron PID, AN-AQM, FAPIDNN and NRL) and Router C uses Drop Tail. The sources use TCP/Reno. In the following simulations, unless mentioned otherwise, the following parameters are used as default: the packet size is 1000 bytes, the capacity of the common link is 45 Mb/s, the round trip propagation time is 80 ms, and the buffer size is 900 packets (twice the bandwidth-delay product of the network). In our simulation experiments, Explicit Congestion Notification (ECN) at the AQM is always off. Therefore, the outputs of AQM schemes are dropping probabilities, packets are dropped instead of only being marked. The TCP connections always have data to send as long as their congestion windows permit. The receiver's advertised window size is set sufficiently large so that TCP connections are not constrained at the destination. The ack-every-packet strategy is used at the TCP receivers. The target queue length is set at 300 packets for all AQM schemes.

For ARED, the parameters are set as: $min_{th} = 15$, $max_{th} = 585$ and $w_q = 0.002$, and other parameters are set the same as in [6]: $\alpha = 0.01$, $\beta = 0.9$, $intervaltime = 0.5$ s. For REM, the default parameters of [8] are used: $\phi = 1.001$, $\alpha = 0.1$, $\gamma = 0.001$, the sampling interval is 2 ms. For PI controller, the default parameters in *NS2* are used: $a = 1.822 \times 10^{-5}$, $b = 1.816 \times 10^{-5}$

and the sampling frequency $w = 170$. The parameters of Neuron PID and AN-AQM follow [31] and [32]. For Neuron PID, the sampling time $\delta t = 0.001$s, neuron gain $K = 0.01$, learning rates $d1 = 0.000002$, $d2 = 0.0000001$, and $d3 = 0.0000001$. For AN-AQM, the sampling time δt and neuron gain K are the same value as Neuron PID's, the learning rates are set as: $d1 = 0.00001$, $d2 = 0.00001$, $d3 = 0.00001$, $d4 = 0.0001$, $d5 = 0.0001$, $d6 = 0.0001$. Two other neuron-based AQM schemes' parameters are set as below: For FAPIDNN, we set $K_P = 2.58392 \times 10^{-8}$, $K_I = 2.98929 \times 10^{-8}$, $K_D = 3.50735 \times 10^{-10}$, learning rate $\eta = 8.5 \times 10^{-5}$ and momentum gene $\alpha = 0.005$, which are the same as in [34]. For NRL, we keep the same parameters as in [33], i.e., $\alpha = 0.001$, $\gamma = 0.98$, $\theta_1 = 10$, $\theta_2 = 0.02$.

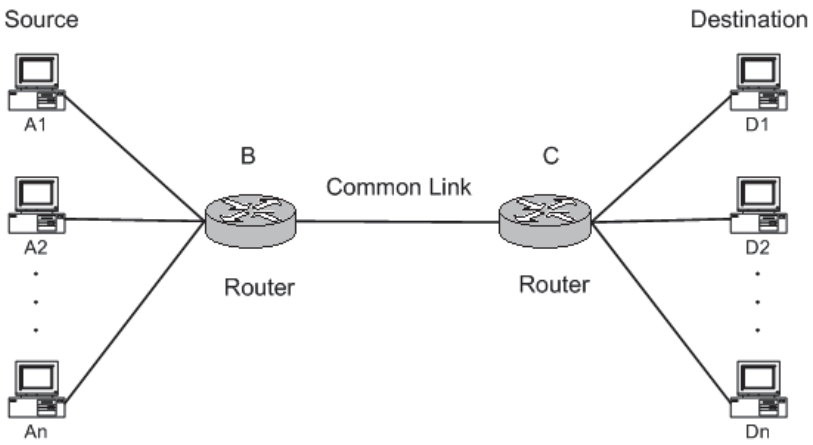

Figure 6: The single bottleneck topology

3.2. Performance for a constant number of TCP connections

This simulation experiment tests whether AQM schemes can control and stabilize the queue length at an arbitrarily chosen target for different loads and link capacities. The sources start data transmission at time 0. Figures 7 and 8 present the instantaneous queue lengths for 500 and 1500 TCP connections, respectively. Tables 1 and 2 provide the link efficiency, average packet drop ratio, average queue length and queue length standard deviation (STD) of all eight AQM schemes.

The results demonstrate that the four neuron-based AQM schemes (Neuron PID, AN-AQM, FAPIDNN and NRL) and IAPI converge to the target queue length Q_T much faster than the other three traditional AQM schemes (ARED, REM and PI). In addition, the queue length jitter of the four neuron-based AQM schemes is much smaller than three traditional schemes, IAPI maintains

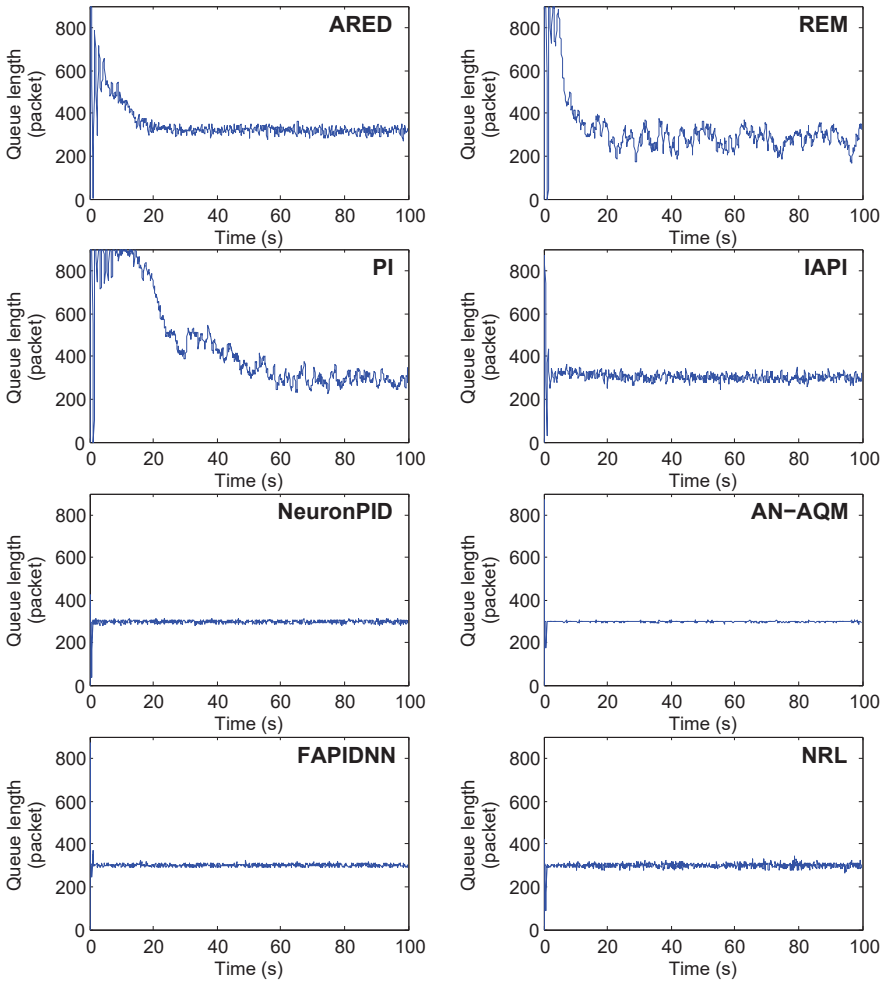

Figure 7: Queue length variations for 500 TCP connections with the queue length target of 300

relative large queue STD than the four neuron-based schemes, but better than the three traditional schemes.

When there are 500 TCP connections, all schemes are successful in controlling the queue length to the target value. Neuron PID has the lowest STD of the queue length among all schemes, and provides the lowest packet drop ratio among the four neuron-based schemes. FAPIDNN scheme has the average queue length equal to 300.30, which is most close to the target value. Although PI has the lowest packet drop ratio, it has the longest transient time which is around 40 seconds, and largest STD of the queue length.

Table 1: Performance for 500 TCP connections with target queue length 300 packets

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	95.24%	15.88%	347.28	82.76
REM	95.09%	16.37%	320.05	130.43
PI	95.14%	14.82%	454.72	214.07
IAPI	95.28%	16.68%	306.58	42.71
Neuron PID	95.28%	17.22%	296.88	16.28
AN-AQM	95.28%	18.05%	298.02	21.12
FAPIDNN	95.28%	19.27%	300.30	21.39
NRL	95.28%	19.18%	298.64	16.67

Table 2: Performance for 1500 TCP connections with target queue length 300 packets

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	95.29%	22.64%	728.72	34.36
REM	95.26%	24.74%	324.78	194.52
PI	95.29%	21.70%	539.08	252.60
IAPI	95.29%	25.85%	309.92	45.57
Neuron PID	95.29%	24.78%	297.34	19.03
AN-AQM	95.29%	27.04%	298.39	20.04
FAPIDNN	95.29%	28.58%	301.25	26.57
NRL	95.06%	25.74%	274.70	20.86

In the second run of the simulation, where there are 1500 TCP connections, ARED oscillates around the value of 700 packets that is even higher than maximum threshold (585 packets) set in its algorithm. This means that it fails in controlling the queue. PI and REM can be considered successful in controlling the queue to its target. However, their fluctuations are relatively large (their STD of the queue length are 252.60 and 194.52, respectively). Neuron PID again provides lowest packet drop ratio of four neuron-based schemes and lowest STD of the queue length of all schemes. FAPIDNN's average queue length is most close to the target value.

IAPI, the modified version of PI controller, improves the performance metrics of average queue length and queue length's STD of the three traditional AQM schemes in both simulation runs. IAPI also inherits the advantage of low packet drop ratio from PI controller. It is interesting to notice that PI achieves lowest packet drop ratio but with largest STD of the queue length in both simulation runs. Although packet loss due to full buffer normally increases with queue length variability, the packet drop ratio which depends on the AQM algorithm does not necessarily increase with such variability. This is illustrated by the

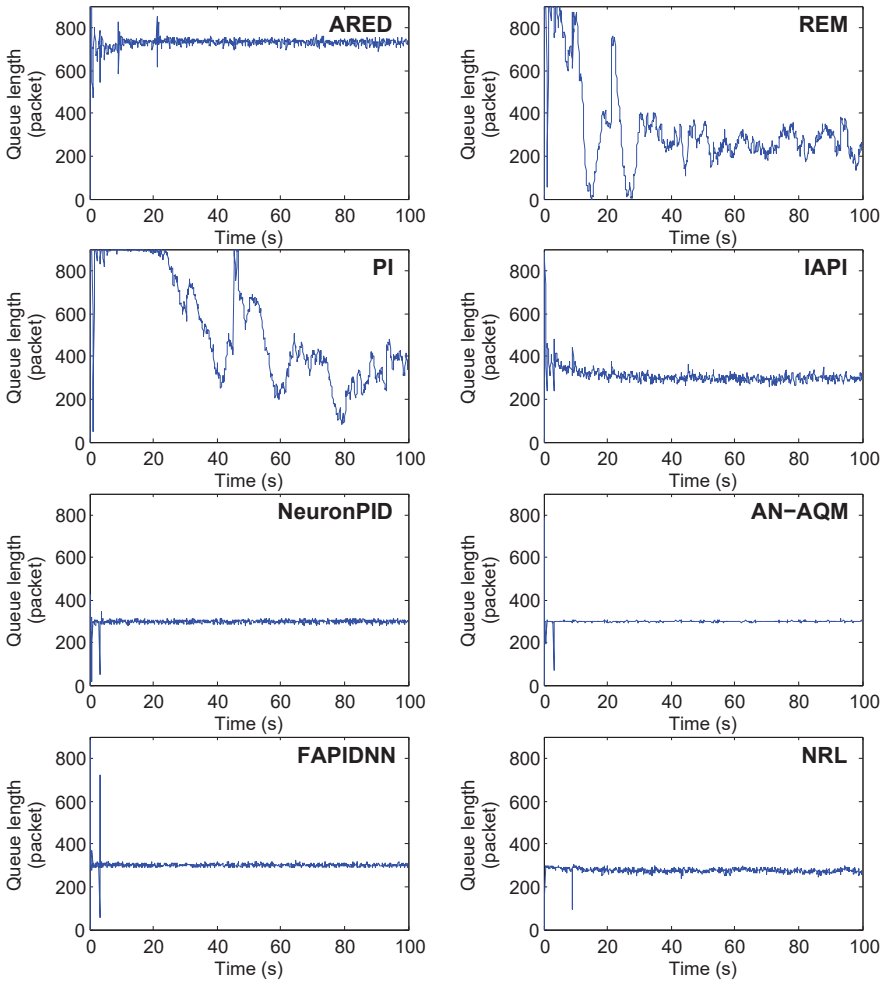

Figure 8: Queue length variations for 1500 TCP connections with the queue length target of 300

numerical results presented for this especially the lowest packet drop ratio of PI and its very large queue length variability.

Note that for the four neuron-based schemes, there are glitches in the queue dynamics within the first 10 seconds, and then the queue length stays close to the desired value. This is an indication that neuron-based schemes have a very short transient period before reaching the steady state. Moreover, when we examine closely the queue length dynamics within the transient period, we observe that the queue length fluctuations are very large initially, then small and then very large again, forming the glitches. This is because during this period,

the neurons are still learning. The system behaviour at any time is based on the limited input for learning up to that time instant, so it may fluctuate very wildly. However, once the neurons have learned sufficiently, the queue length reaches steady state and fluctuates closely to the target.

For lower queue target values, AQM schemes have less time to respond and have to act faster than the case with the higher queue target values. We now test the ability of all AQM schemes to stabilize the queue, in the case where the queue target value Q_T is reduced to 50 packets with 500 and 800 TCP connections. The results of instantaneous queue length are depicted in Figure 9 and Figure 10, while other performance metrics are given in Table 3 and Table 4.

Table 3: Performance for 500 TCP connections with target queue length 50 packets

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	95.26%	18.11%	74.64	21.68
REM	94.20%	18.24%	73.14	157.64
PI	94.95%	16.97%	233.18	275.23
IAPI	95.28%	18.34%	66.88	33.53
Neuron PID	95.14%	18.45%	47.72	7.54
AN-AQM	95.28%	20.17%	49.04	27.52
FAPIDNN	95.28%	21.08%	51.73	28.02
NRL	94.27%	20.90%	57.50	65.74

Table 4: Performance for 800 TCP connections with target queue length 50 packets

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	95.25%	21.84%	92.07	11.83
REM	93.06%	22.35%	76.24	182.75
PI	94.77%	20.52%	265.35	325.24
IAPI	95.29%	22.58%	70.03	34.72
Neuron PID	95.15%	20.77%	47.85	7.96
AN-AQM	95.29%	23.37%	48.79	16.89
FAPIDNN	95.29%	25.29%	51.97	28.38
NRL	94.09%	24.45%	65.53	75.31

Having this new target queue length, the four neuron-based schemes outperform other four schemes in both transient time and the stability of the queue length.

Based on the results presented in Figure 9, the queue dynamics of ARED fluctuates between two levels: 50 and 100 during the 100 second run, and fails to converge to the target queue length of 50 within this period. In the next run

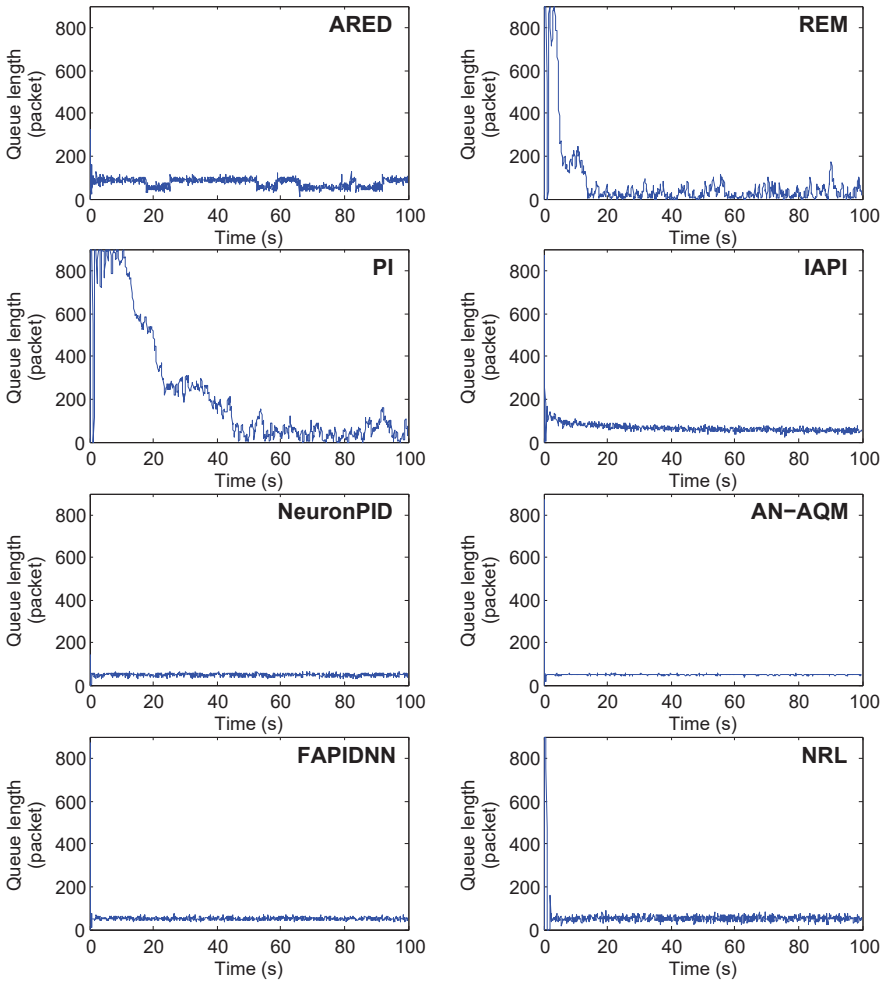

Figure 9: Queue length variations for 500 TCP connections with the queue length target of 50

where there are 800 TCP connections, the queue dynamics of ARED fluctuates around the value of 100. This value is larger than the maximum threshold (75 packets) set in its algorithm therefore ARED fails in controlling the queue this time. Although PI provides the lowest packet drop ratio, its average queue length (265.35) is far from the target value (50). The sluggish response of PI leads a quite long transient time, which is around 50 seconds, hence PI can not be considered as effective in controlling the queue in such 100 seconds time. Compared with PI, REM improves the performance in reducing both response time and fluctuation of the queue length. IAPI significantly improves

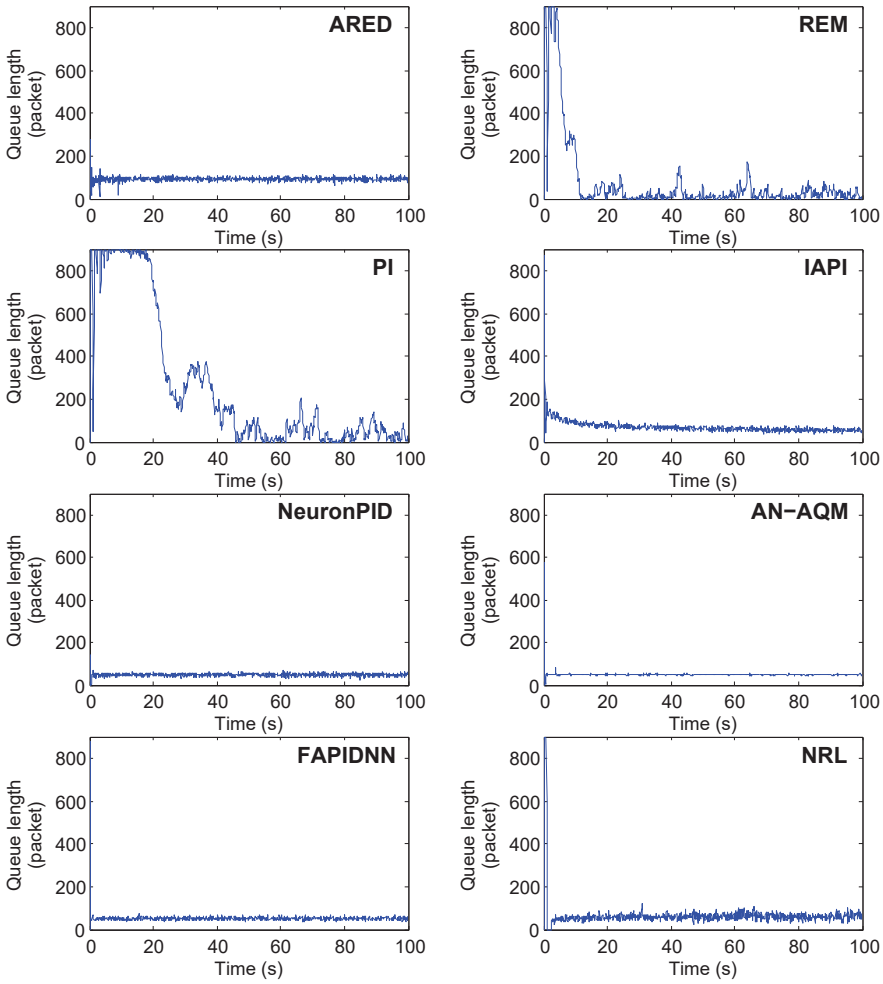

Figure 10: Queue length variations for 800 TCP connections with the queue length target of 50

the fluctuation of queue length, and it maintains the average queue length equal to around 70.

Among four neuron-based schemes, Neuron PID achieves the lowest packet drop ratio and smallest queue length's STD. While the average queue length of AN-AQM is most close to the target value.

3.3. Performance for a different bottleneck link capacity

In order to examine the performance of the AQM schemes under different link capacities, we now consider the cases where the capacity of the bottleneck

is firstly set to 15 Mb/s, and then set to 115 Mb/s, for 800 TCP connections, while the other parameters remain the same. The instantaneous queue length are plotted in Figure 11 and Figure 12, and other performance metrics are given in Table 5 and Table 6.

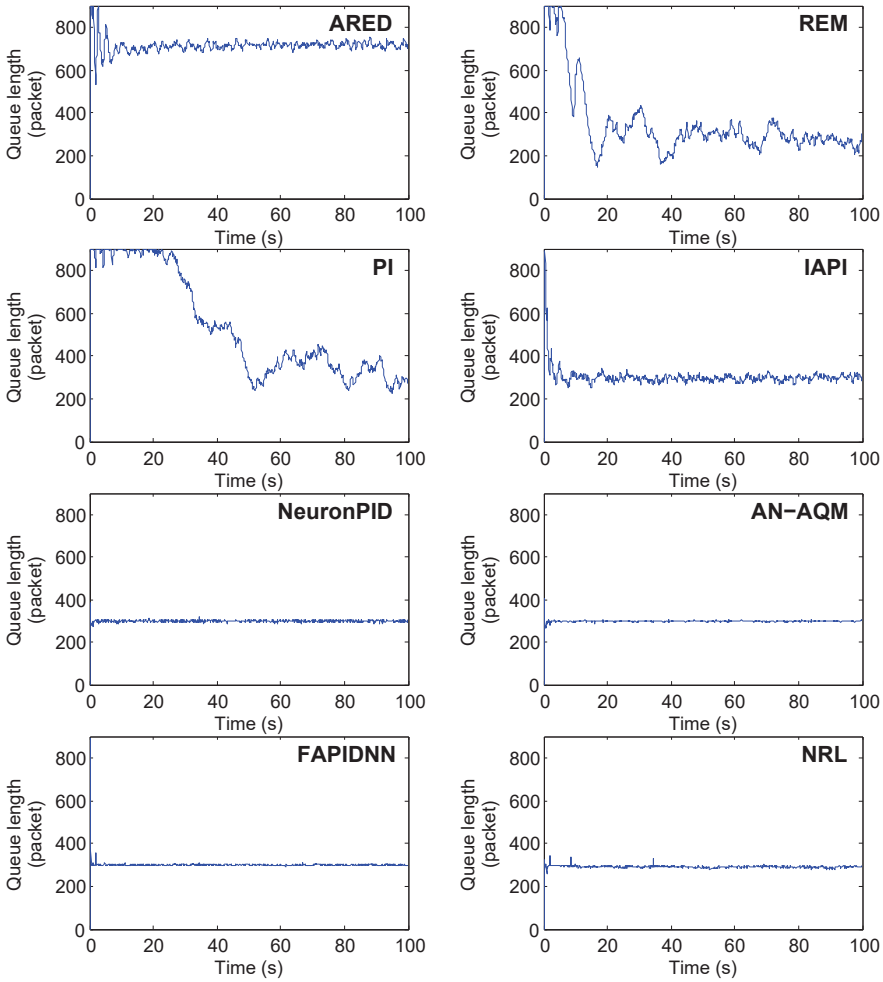

Figure 11: Queue length variations for 800 TCP connections with the bottleneck link capacity of 15 Mb/s

As can be seen from Table 5 and Table 6, AN-AQM achieves the smallest packet drop ratio and STD of the queue length in the first run where the capacity of the bottleneck link is 15 Mb/s, while Neuron PID achieves the best packet drop ratio and STD of the queue length in the second run, which demonstrates that AN-AQM has improved the performance of Neuron PID in a more

Table 5: Performance for 800 TCP connections with link capacity of 15 Mb/s

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	95.30%	21.12%	713.73	37.96
REM	95.30%	25.14%	343.95	165.15
PI	95.30%	22.03%	536.49	241.98
IAPI	95.30%	26.44%	301.97	51.61
Neuron PID	95.30%	24.31%	298.82	10.65
AN-AQM	95.30%	22.71%	298.21	10.36
FAPIDNN	95.30%	29.14%	301.08	26.43
NRL	95.30%	27.19%	290.23	10.55

Table 6: Performance for 800 TCP connections with link capacity of 115 Mb/s

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	95.15%	12.96%	330.72	80.55
REM	95.04%	13.15%	297.22	150.27
PI	95.05%	12.54%	422.59	225.17
IAPI	95.18%	13.29%	306.05	41.75
Neuron PID	95.18%	13.72%	294.42	19.09
AN-AQM	95.22%	14.94%	292.68	24.41
FAPIDNN	95.27%	16.43%	300.93	26.92
NRL	95.26%	10.86%	251.62	27.52

stringent environment with low link capacity. FAPIDNN shows its advantage in its average queue length in both two runs.

The queue of ARED fluctuates around the value of 700 packets instead of the target value in the first run. PI achieves the lowest packet drop ratio as well but with the largest transient time and STD of the queue length. Compared with PI, REM shortens the transient time and decreases the STD of the queue length. IAPI improves the performance of PI significantly in average queue length and queue length's STD, but IAPI has a slightly higher packet drop ratio.

3.4. Performance for different propagation time

Now the impact of propagation time on the performance metrics is investigated. Two simulations have been performed, there are 800 TCP connections with different propagation times in both simulations. In the first one, the round trip propagation times are uniformly distributed between 20 and 140 ms, and in the second one, they are uniformly distributed between 150 and 250 ms. Figures 13 and 14 present the instantaneous queue length for these two simulations. Table 7 and 8 show corresponding performance metrics, respectively.

The results show that ARED fails to control the queue in both simulation runs because the queue length of ARED fluctuates at the value of 700 packets

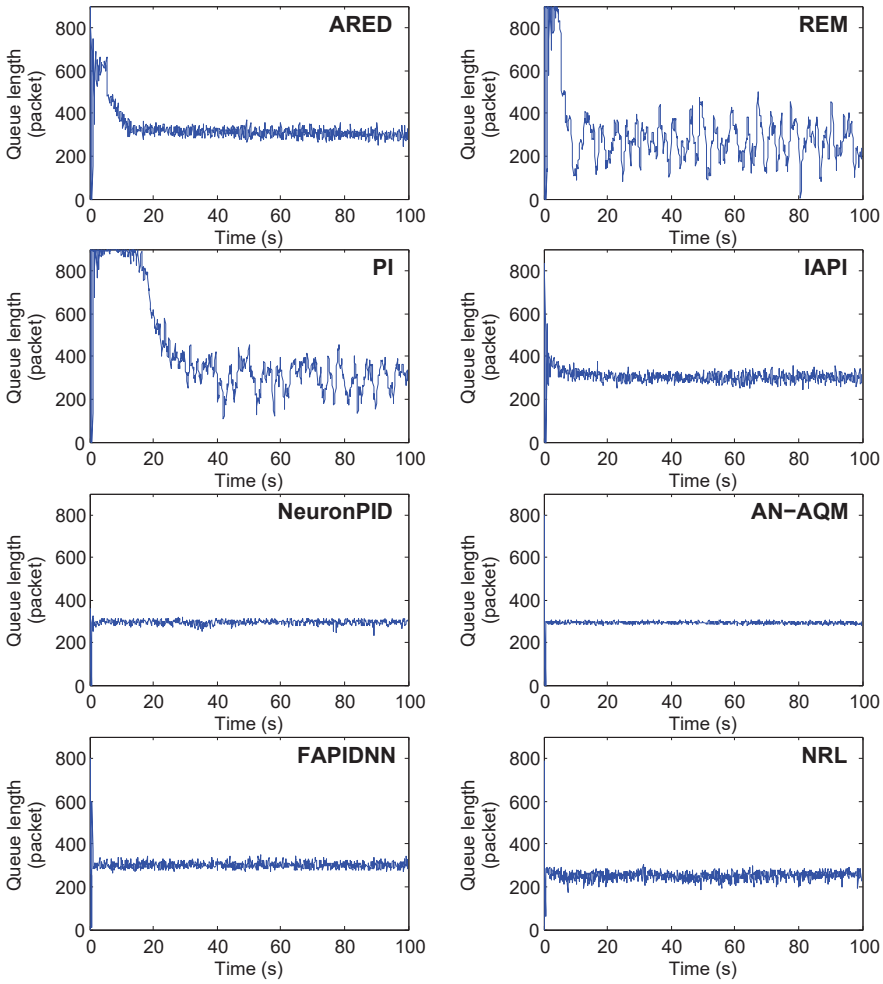

Figure 12: Queue length variations for 800 TCP connections with the bottleneck link capacity of 115 Mb/s

instead of the target value. PI continues its attributes in low packet drop ratio but large fluctuation of the queue length. REM improves the performance metrics of average queue length and queue STD.

Compared with three traditional AQM schemes, in both two runs, IAPI shows superb performance metrics of average queue length and queue length's STD while still keep relatively low packet drop ratio.

Four neuron-based schemes have close performance metrics in such experiment with different propagation times, AN-AQM shows the best performance of queue length's STD in the first run and Neuron PID shows the best performance

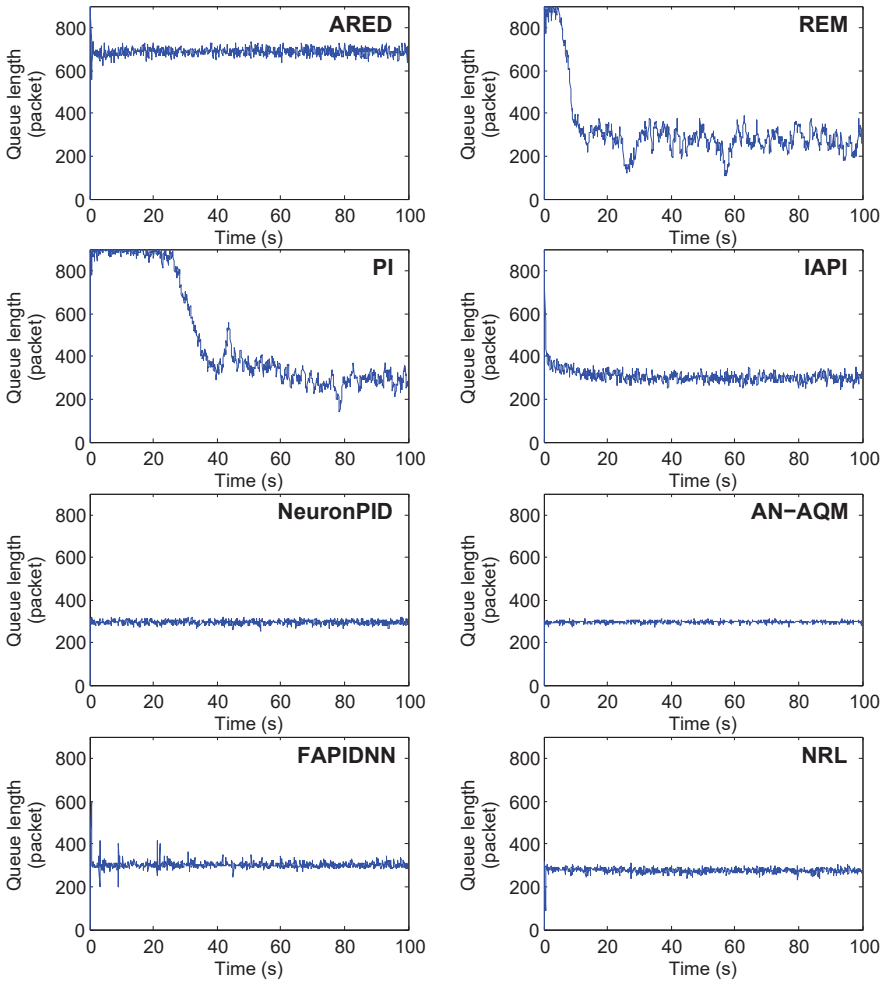

Figure 13: Queue length variations for 800 TCP connections with the round trip propagation time distributed randomly between 20 ms to 140 ms

of packet drop ratio and queue length's STD in the second run. FAPIDNN achieves the best average queue length in both two runs.

3.5. Performance for TCP connections which start and stop dynamically

This group of simulations contains two parts, in which TCP connections start and stop in two different manners: deterministically and randomly.

There are two simulation runs in the first part where the number of active TCP connections is varied deterministically. The number of TCP connections is increased from 500 to 1500 in the first run and decreased from 1500 to 500

Table 7: Performance metrics for 800 TCP connections with the round trip propagation time distributed randomly between 20 ms to 140 ms

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	95.35%	18.15%	687.26	30.01
REM	95.35%	20.99%	322.04	158.33
PI	95.35%	19.62%	495.96	255.45
IAPI	95.35%	21.08%	308.33	42.01
Neuron PID	95.35%	22.00%	294.95	13.49
AN-AQM	95.35%	23.09%	296.24	10.93
FAPIDNN	95.16%	22.23%	300.92	25.67
NRL	95.33%	24.05%	274.80	15.07

Table 8: Performance metrics for 800 TCP connections with the round trip propagation time distributed randomly between 150 ms to 250 ms

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	95.17%	12.81%	634.24	67.02
REM	94.96%	15.62%	318.01	139.51
PI	95.01%	14.65%	449.24	221.60
IAPI	95.18%	15.95%	304.54	47.56
Neuron PID	95.19%	17.30%	292.94	18.38
AN-AQM	95.18%	17.72%	292.10	23.89
FAPIDNN	94.65%	19.08%	300.59	47.63
NRL	95.16%	19.18%	274.17	21.41

in the second run. In each of the simulation runs, a group of 100 connections start (or stop), at the same time, in each 10 seconds interval. The instantaneous queue lengths are plotted in Figure 15 and Figure 16, respectively. Table 9 and Table 10 show the performance metrics numerically.

As can be seen from Figures 15 and 16, for ARED scheme, in the first simulation run, where the TCP connections increases abruptly in 10 seconds interval, the queue dynamics of ARED scheme shows small glitches on the boundaries between intervals due to the transient behaviour, it stabilizes the queue at around 400 packets at the end of the process. In the second simulation run, where the TCP connections decrease abruptly, besides the glitches on the boundaries between intervals, the queue length fluctuates at around 700 packets and suddenly drops to 300 packets in the last 10 seconds interval. Based on our experiments, we observe that ARED is not successful in stabilizing the queue length at the target, but it exhibits lower STD of queue length than PI and REM. PI, REM and IAPI behave as in previous experiments. PI achieves best packet drop ratio, REM has a better average queue length and decreased queue length's STD. IAPI show both improved average queue length and queue STD while still maintains

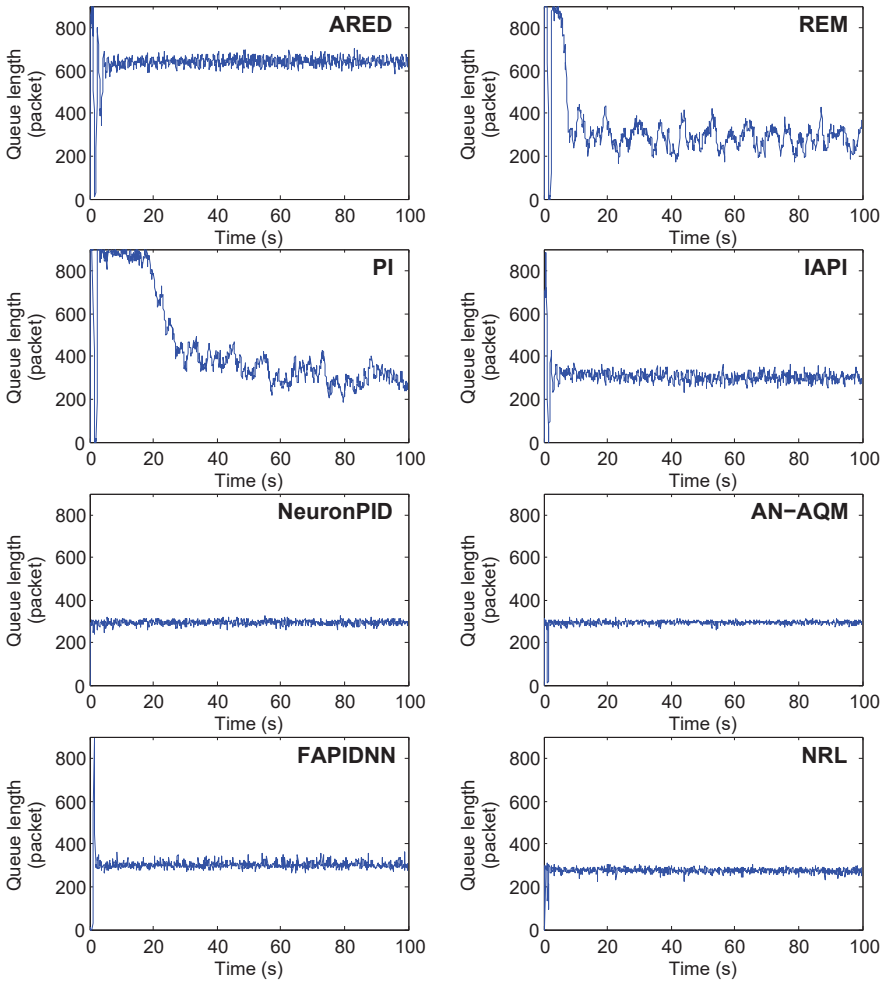

Figure 14: Queue length variations for 800 TCP connections with the round trip propagation time distributed randomly between 150 ms to 250 ms

a good packet drop ratio. As for the four neuron-based schemes, Neuron PID shows the best in packet drop ratio and queue STD. FAPIDNN gives the best average queue length.

Next, two simulations involving random start and stop times are performed, thus simulating staggered connection setup and termination. In the first run, the initial number of connections is set to 300 and, in addition, 1200 connections have their start-time uniformly distributed over a period of 100 seconds. In the second run, the initial number of connections is set to 1500, out of which 1200 connections have their stop-time uniformly distributed over a period 100 sec-

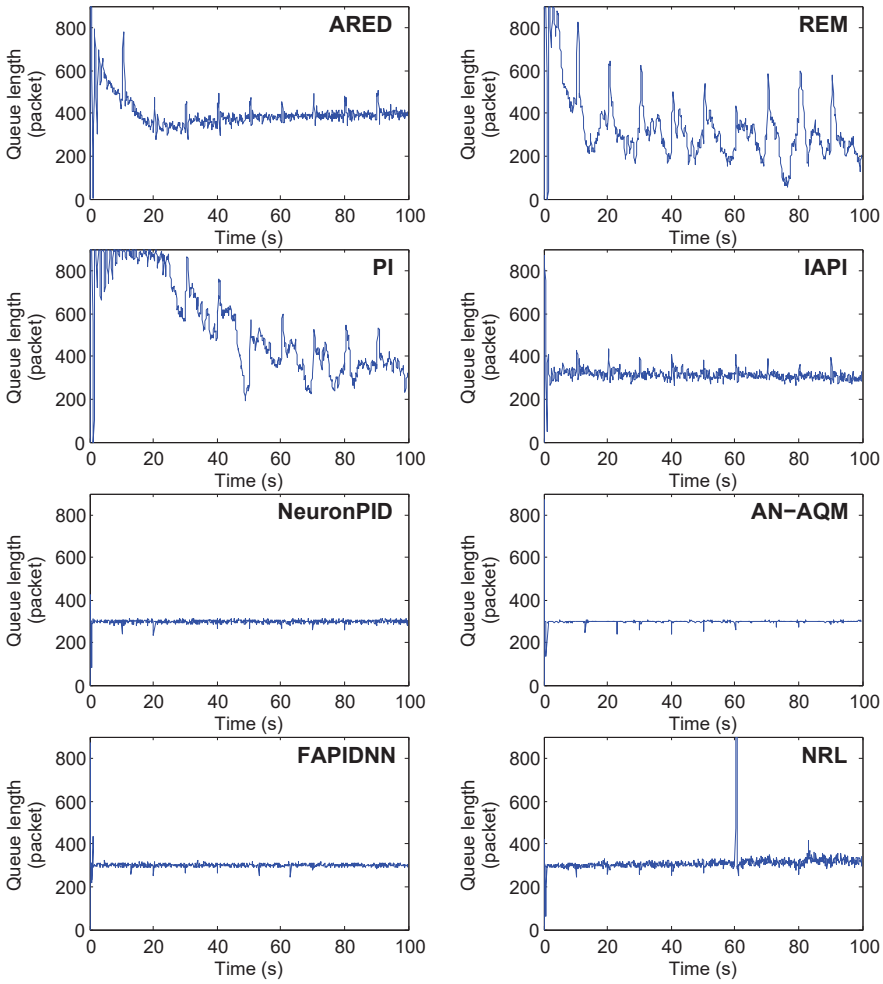

Figure 15: Queue length variations for the number of TCP connections increasing abruptly from 500 to 1500

onds. The instantaneous queue lengths are plotted in Figure 17 and Figure 18. The numerical performance metrics are given in Table 11 and Table 12.

The results of the random case are similar to the previous deterministic case. Note that the glitches on the boundaries between intervals do not appear in this random case. For ARED, when the TCP connection increase randomly, the queue length is stabilized at round 400 packets; when TCP connection decrease randomly, the queue length fluctuates in the range of [600, 700]. The NRL scheme achieves the lowest STD of the queue length in the case where the TCP connections increasing randomly from 300 to 1500, however, it fails to control

Table 9: Performance metrics for the number of TCP connections increasing abruptly from 500 to 1500

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	95.25%	21.84%	400.16	77.67
REM	95.11%	21.93%	331.08	156.57
PI	95.14%	20.09%	544.79	218.30
IAPI	95.28%	22.58%	316.40	43.99
Neuron PID	95.28%	23.24%	297.68	15.63
AN-AQM	95.28%	23.78%	298.05	21.91
FAPIDNN	95.28%	25.14%	300.41	22.25
NRL	95.28%	25.25%	311.01	47.32

Table 10: Performance metrics for the number of TCP connections decreasing abruptly from 1500 to 500

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	95.29%	19.90%	660.88	122.23
REM	95.26%	21.31%	312.37	195.94
PI	95.29%	18.39%	472.13	277.32
IAPI	95.29%	22.54%	299.14	48.49
Neuron PID	95.29%	21.64%	296.95	18.96
AN-AQM	95.29%	23.38%	298.03	20.03
FAPIDNN	95.29%	25.39%	300.98	26.53
NRL	95.06%	20.85%	279.39	20.68

Table 11: Performance metrics for the number of TCP connections increasing randomly from 300 to 1500

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	95.18%	22.01%	397.91	59.00
REM	95.13%	22.44%	330.04	134.56
PI	95.11%	20.70%	537.33	242.28
IAPI	95.24%	22.54%	319.64	41.22
Neuron PID	95.28%	23.39%	297.55	16.73
AN-AQM	95.28%	23.64%	298.24	16.44
FAPIDNN	95.28%	25.42%	301.56	24.89
NRL	95.28%	23.69%	278.50	15.65

the queue in the case where the TCP connections decreasing randomly. This reveals a undesired effect of NRL.

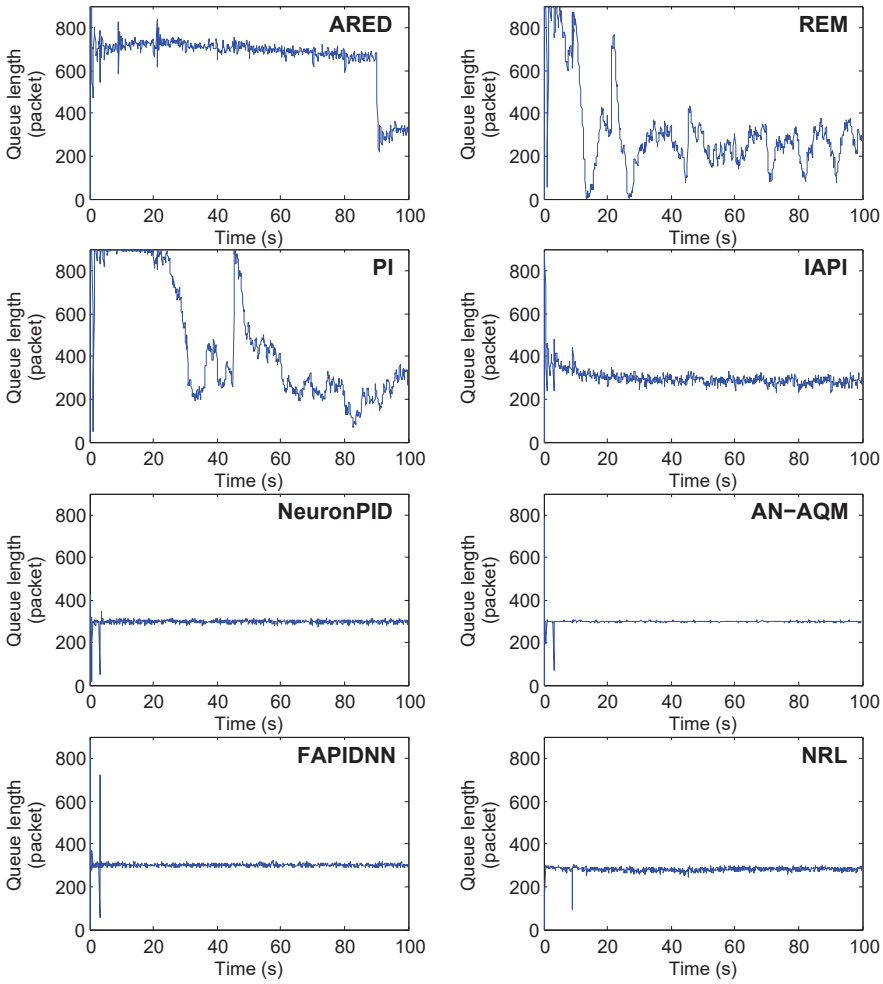

Figure 16: Queue length variations for the number of TCP connections decreasing abruptly from 1500 to 500

The estimated reward value V^* in the NRL scheme, depends on both queue length error $e_1(t) = q(t) - q_{ref}$ and sending rate mismatch $e_2(t) = x(t) - C$. When the number of TCP connections decreases abruptly, $e_1(t)$ and $e_2(t)$ will also decrease rapidly, hence the estimated reward value $V^* = \omega_1(t)e_1(t) + \omega_2(t)e_2(t)$ will take a significantly smaller value. The dropping probability is determined by $p = \frac{2}{1+e^{V^*}} - 1$. Consequently, the dropping probability will change to 0, and then the queue length will deviate from the target value. In this particular case, the simulation results show that from time 9.2 seconds onwards, NRL maintains the dropping probability at value 0. Afterwards, the

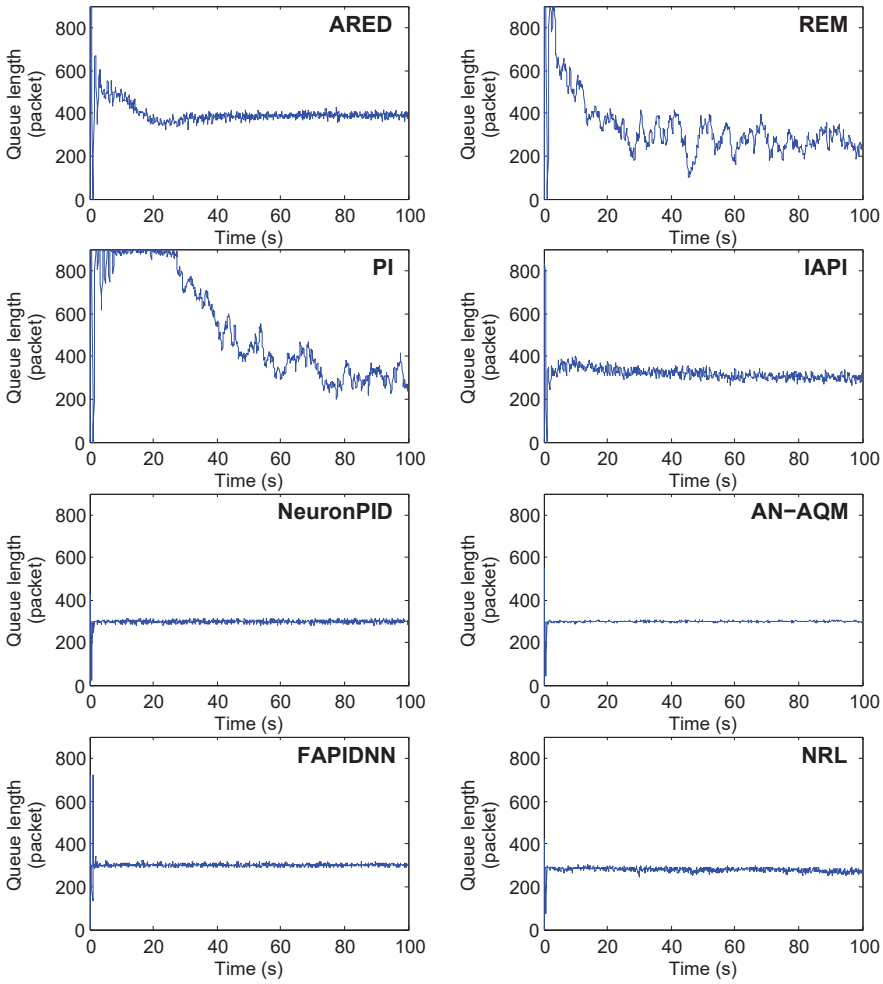

Figure 17: Queue length variations for the number of TCP connections increasing randomly from 300 to 1500

reinforcement learning is not able to tune the neuron weights $\omega_1(t)$ and $\omega_2(t)$ to derive a "normal" estimated reward V^*, from which the correct dropping probability can be determined. Therefore, the queue length fluctuates around the value of 900, which is the buffer size. This undesired effect of NRL requires a very abrupt decrease in the number of connections which occurred in the present case that involves random decrease of connection rather than the regular decrease that characterizes the previous cases. Note that a significant drop in the number of connections may be a very rare event, so the undesired effect we have uncovered may not be a major drawback of NRL.

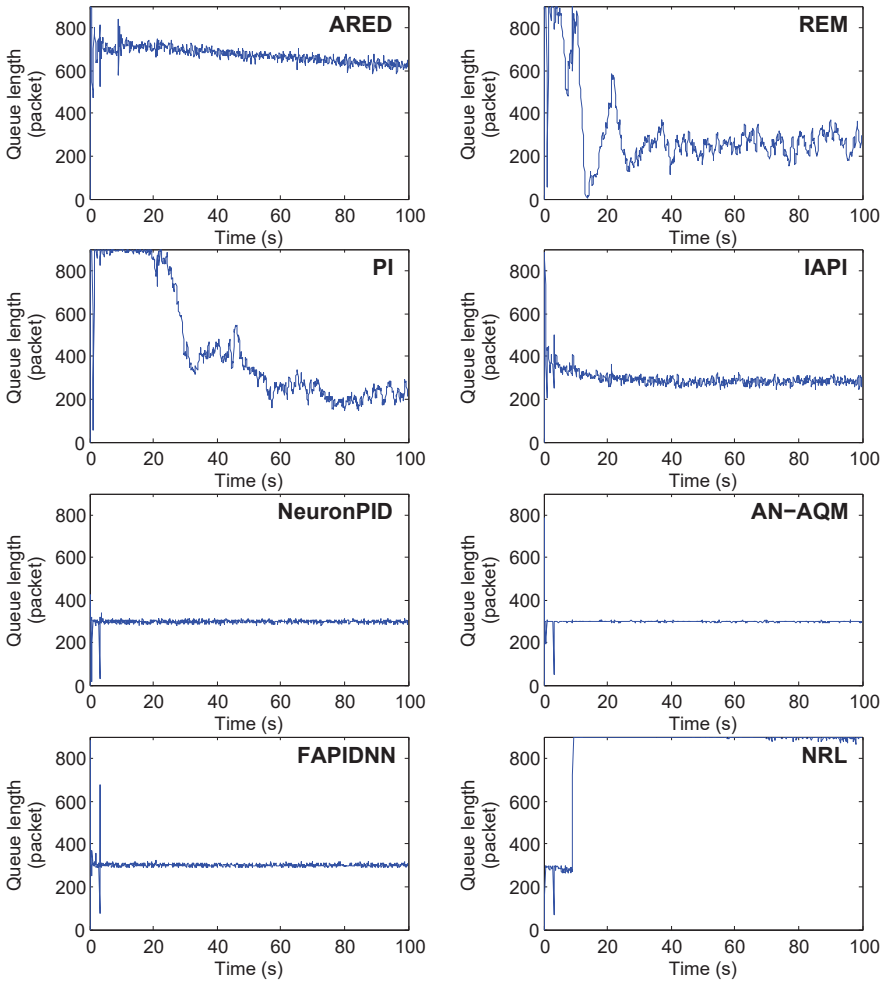

Figure 18: Queue length variations for the number of TCP connections decreasing randomly from 1500 to 300

3.6. Performance for a long-delay network

Simulations in [19] have shown that AQM schemes, such as PI, RED and REM, are unstable when the RTPT is 400 ms. In this experiment, the performance of AN-AQM for a long-delay network is tested. There are 800 TCP connections and the RTPTs are 500 ms. The simulation results are given in Figure 19 and numerical performance metrics are given in Table 13.

The results of ARED, REM and IAPI show large queue fluctuation during the whole simulation period, therefore they cannot be considered as effective schemes in such long-delay network environment. PI can be regarded as a

Table 12: Performance metrics for the number of TCP connections decreasing randomly from 1500 to 300

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	95.29%	17.24%	671.30	44.07
REM	95.27%	19.18%	310.58	182.48
PI	95.29%	17.20%	448.73	270.06
IAPI	95.29%	20.41%	296.10	47.83
Neuron PID	95.29%	20.06%	297.17	19.25
AN-AQM	95.29%	21.24%	297.91	20.23
FAPIDNN	95.29%	23.10%	300.96	25.78
NRL	95.06%	14.84%	838.77	182.64

Table 13: Performance metrics for 800 TCP connections with the round trip propagation time of 500 ms

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	93.26%	5.59%	360.06	202.75
REM	90.29%	6.18%	311.35	297.46
PI	92.90%	5.85%	345.51	171.68
IAPI	92.91%	6.13%	256.99	195.53
Neuron PID	92.61%	7.08%	278.71	59.08
AN-AQM	93.02%	7.49%	282.04	56.87
FAPIDNN	93.37%	10.21%	300.01	89.54
NRL	92.48%	9.08%	269.25	58.63

stable scheme but it enters the steady state after 30 seconds with a relatively large STD of the queue length equal to 171.68.

The four neuron-based schemes are still effective in stabilizing the queue length at the target value for TCP connections with long propagation time, where all of them enter the steady state after around 10 seconds. Since a neuron learning processes needs to compare the output (queue length and total sending rate at the bottleneck link) with a certain cost or reward function to tune its neuron weights and other parameters, while the long propagation time delays the reading of the output. Consequently, the transient times for all neuron schemes become longer in this experiment. Note that AN-AQM performs slightly better than Neuron PID in both average queue length and queue length's STD.

3.7. Performance for TCP connections mixed with UDP connections

This group of simulations considers the impact of the disturbances caused by UDP connections. There are 800 TCP connections mixed with 400 UDP connections. The round trip propagation times of TCP connections are uniformly distributed between 50 and 500 ms and the propagation delay of UDP is

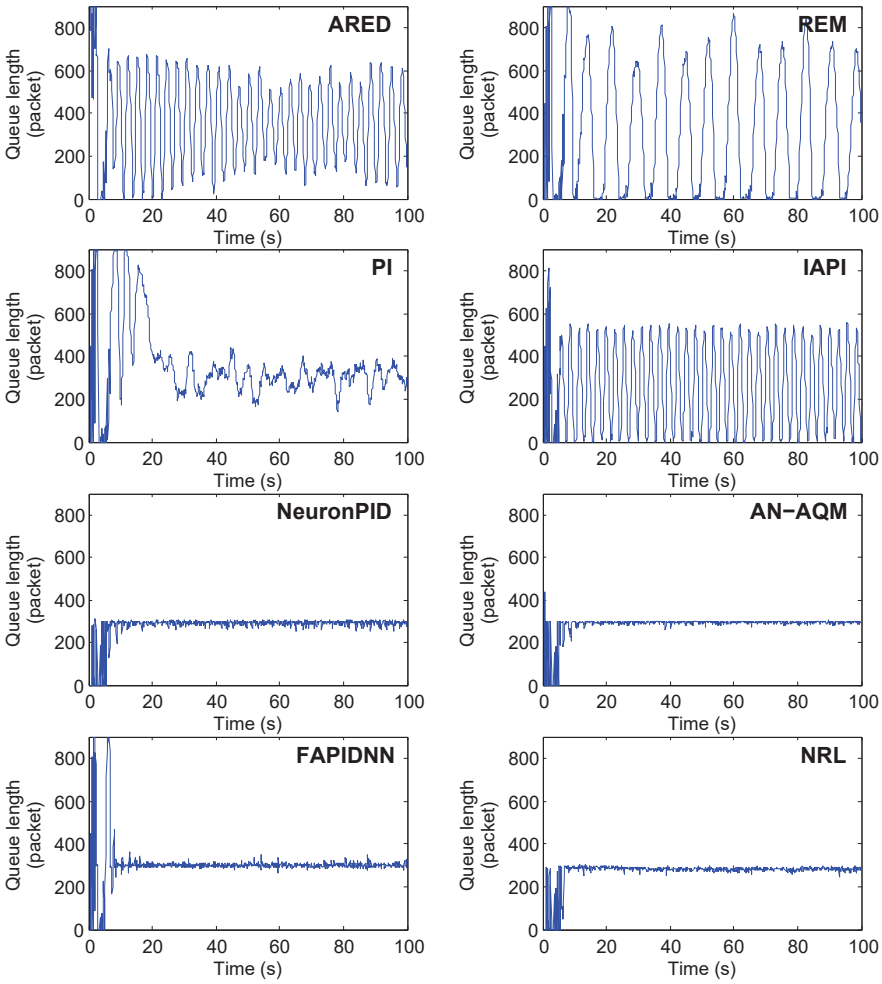

Figure 19: Queue length variations for 800 TCP connections with the round trip propagation time of 500 ms

uniformly distributed between 30 to 250 ms. Each of the UDP sources follows an exponential ON/OFF traffic model, both idle and burst times have a mean of 500 ms. The packet size is set at 500 bytes, and the sending rate during on-time is 64 kb/s. Figure 20 provides the queue dynamics and Table 14 presents the performance metrics of each schemes numerically.

The results are similar with the results obtained in the group of simulations in which the round trip propagation times of TCP connections are uniformly distributed between 150 ms and 250 ms. ARED fails to control the queue under the disturbance caused by UDP flows. PI and REM are able to regulate the

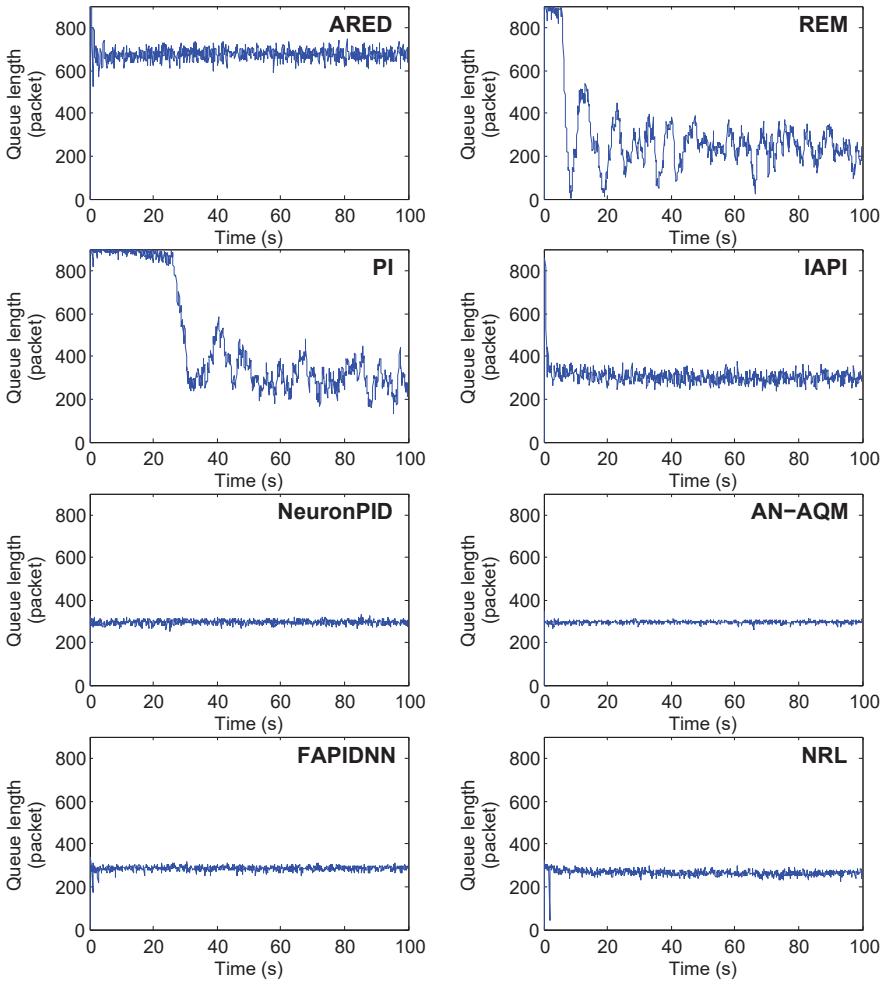

Figure 20: Queue length variations for 800 TCP connections in the presence of additional UDP flows

queue length but with larger fluctuations which equal to 257.55 and 172.57. IAPI reduces such large queue length jitter and maintains the second lowest packet drop ratio among all schemes. AN-AQM provides the best average queue length and queue length's STD among the four neuron-based schemes, which demonstrate that AN-AQM has improved the performance in more realistic network environment including long delay and mixed by disturbance by UDP flows.

Table 14: Performance metrics for 800 TCP connections in the presence of additional UDP flows

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	95.29%	16.50%	672.99	43.55
REM	95.29%	18.55%	285.18	172.57
PI	95.29%	17.65%	475.33	257.55
IAPI	95.29%	18.82%	305.88	43.34
Neuron PID	95.29%	19.57%	294.81	16.66
AN-AQM	95.29%	19.81%	295.11	14.79
FAPIDNN	95.24%	20.66%	284.75	19.97
NRL	95.27%	21.27%	264.81	19.13

3.8. Multiple bottlenecks

The simple single bottleneck topology is now extended to multiple bottlenecks in this part. The new bottleneck network topology is presented in Figure 21. There are two bottlenecks in this topology, the first one is between Routers B and C, and the other is between Routers D and E. The link capacity of the two bottlenecks is 45 Mb/s and the capacity of other links is 100 Mb/s. There are three traffic groups. The first group has N TCP connections traversing all bottleneck links, the second group has N_1 TCP connections traversing the bottleneck link between Routers B and C, and the third group has N_2 TCP connections traversing the bottleneck link between Routers D and E. The round trip propagation times of the first group are 80 ms, and for the second and third groups, they are 100 ms and 150 ms, respectively.

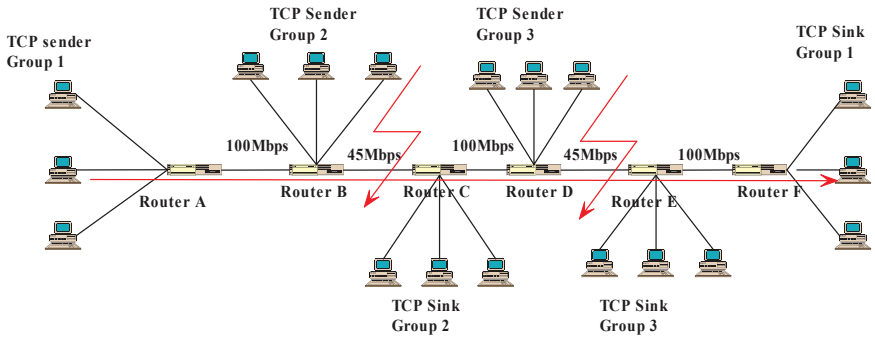

Figure 21: The multiple bottleneck network topology

Two simulation tests have been performed in this multiple bottleneck environment. In the first one, $N = 500$, $N_1 = 200$, and $N_2 = 200$, and in the second one, $N = 100$, $N_1 = 800$, and $N_2 = 400$. Figure 22 and Figure 23 present

the queue lengths for the first case, Figure 24 and Figure 25 present the queue lengths for the second case. Table 15 and Table 16 present the numerical performance metrics of the first case, Table 17 and Table 18 present the numerical performance metrics for the second case.

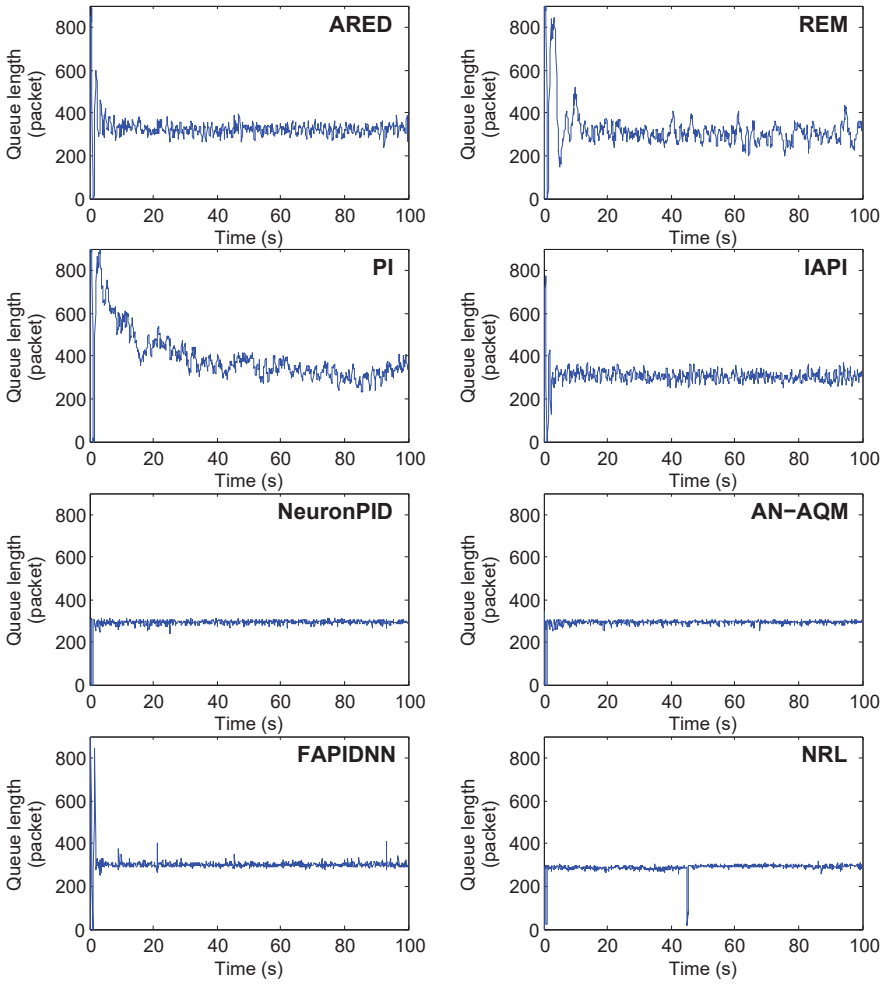

Figure 22: Queue length variations of Router B in multiple bottleneck network for $N = 500$, $N_1 = 200$, and $N_2 = 200$

The results are similar with the common outcome in the previous subsections, which demonstrate that neuron-based schemes are effective and outperform traditional AQM schemes in transient time and queue variance. ARED fails to control the queue at router B where there are 900 TCP connections traversing the first bottleneck. PI exhibits the lowest packet drop ratio at both

Table 15: Performance metrics of Router B in multiple bottleneck network for $N = 500$, $N_1 = 200$, and $N_2 = 200$

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	95.04%	8.82%	324.81	53.08
REM	95.08%	9.22%	315.79	92.69
PI	95.08%	8.05%	389.58	119.69
IAPI	95.06%	9.30%	304.88	44.81
Neuron PID	95.12%	9.91%	291.81	26.50
AN-AQM	95.12%	9.11%	291.72	26.83
FAPIDNN	95.18%	12.76%	301.14	41.36
NRL	95.21%	11.41%	287.61	25.91

Table 16: Performance metrics of Router D in multiple bottleneck network for $N = 500$, $N_1 = 200$, and $N_2 = 200$

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	94.75%	5.74%	310.57	57.13
REM	94.81%	5.95%	310.79	101.50
PI	94.83%	5.43%	354.78	108.73
IAPI	94.87%	5.92%	301.38	50.14
Neuron PID	94.94%	6.74%	287.69	32.43
AN-AQM	94.88%	6.23%	287.89	32.90
FAPIDNN	95.08%	10.00%	300.51	43.97
NRL	95.03%	8.42%	289.24	28.52

Table 17: Performance metrics of Router B in multiple bottleneck network for $N = 100$, $N_1 = 800$, and $N_2 = 400$

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	95.27%	17.44%	679.45	43.86
REM	95.27%	20.30%	320.37	154.84
PI	95.27%	18.81%	495.19	252.56
IAPI	95.27%	20.82%	306.71	41.06
Neuron PID	95.26%	20.52%	296.52	20.37
AN-AQM	95.27%	22.78%	297.70	14.48
FAPIDNN	95.27%	23.48%	300.58	17.47
NRL	95.20%	23.64%	311.78	24.36

Router B and Router D in two simulation runs. IAPI shows superb performance metrics in multi-bottleneck networks since its average queue length is even bet-

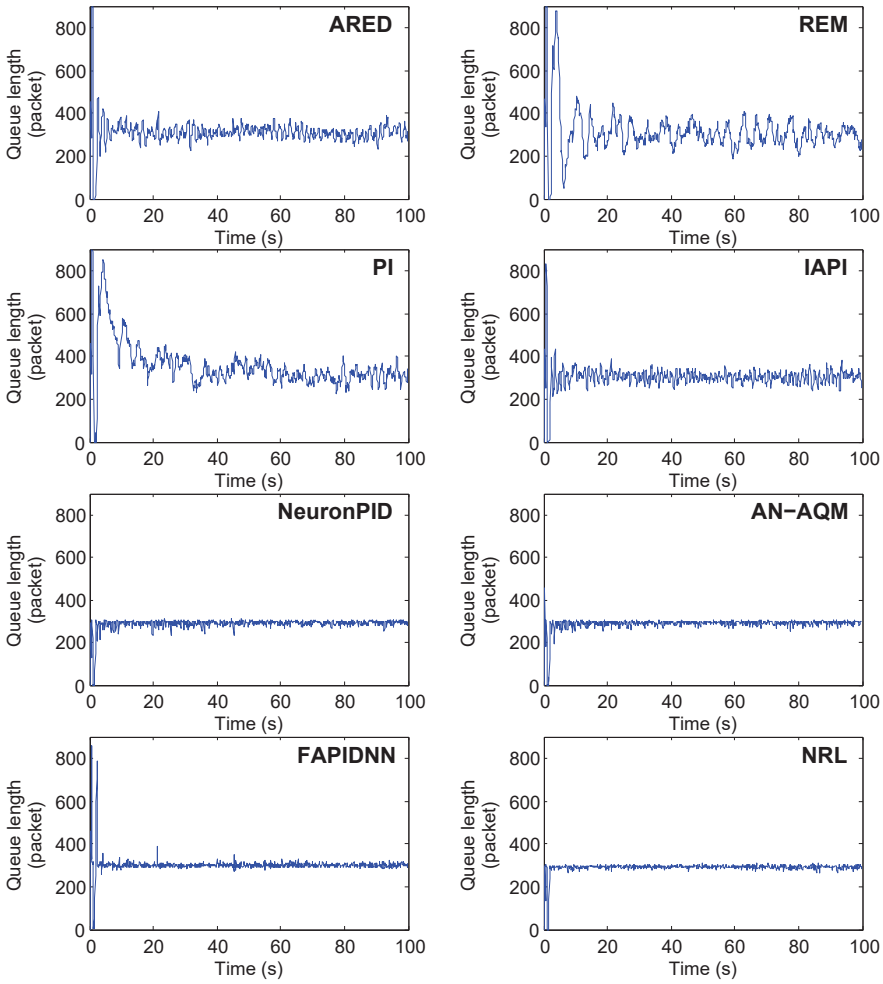

Figure 23: Queue length variations of Router D in multiple bottleneck network for $N = 500$, $N_1 = 200$, and $N_2 = 200$

ter than most of neuron-based schemes and significantly lower queue jitter than the three traditional AQM schemes, and it is also able to achieve lower packet drop ratio than all neuron-based schemes.

The four neuron-based schemes stabilize the queue to the target value more quickly and accurately than other four traditional schemes. Comparing Table 15 and Table 17, it can be found that the performance metrics in Table 17 is not so good as in Table 15, this is because the traffic load has been increased , totally 700 TCP flows traverse Router B in the first run and totally 900 TCP flows traverse Router B in the second run. FAPIDNN achieves best average queue

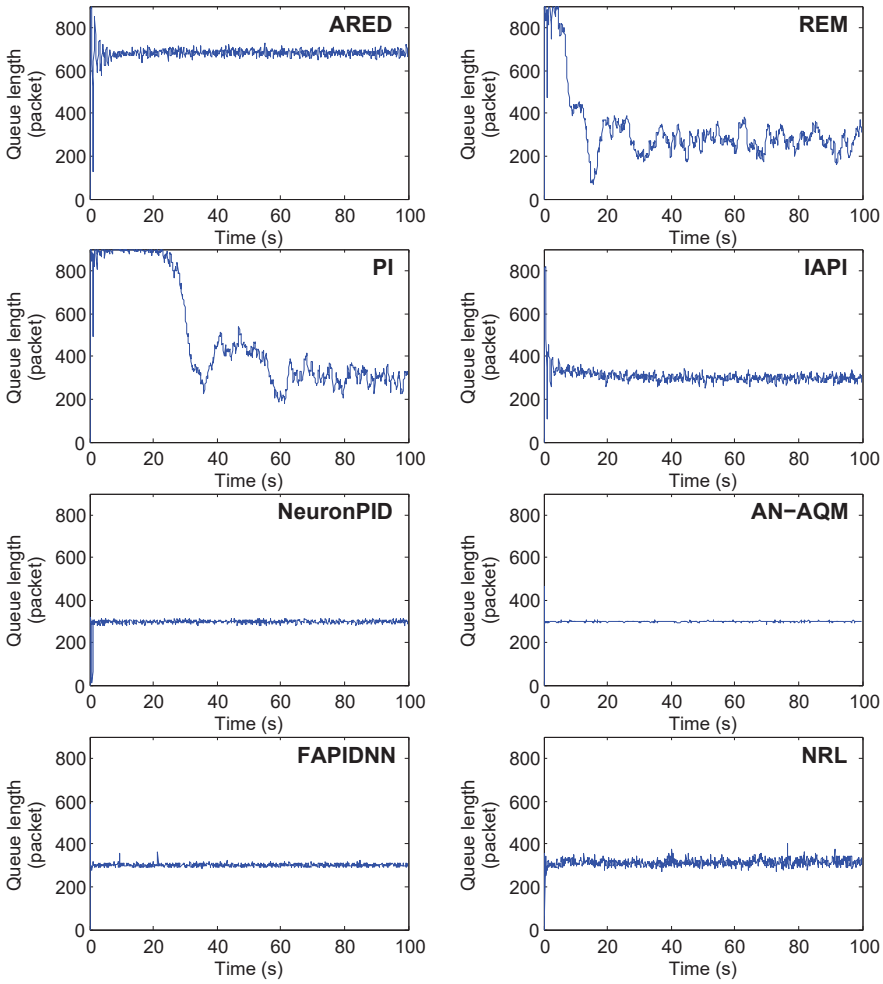

Figure 24: Queue length variations of Router B in multiple bottleneck network for $N = 100$, $N_1 = 800$, and $N_2 = 400$

length at two routers in both two runs, Neuron PID and AN-AQM provide relatively small packet drop ratio among the four neuron-based schemes.

The simulations so far demonstrate that each AQM scheme maintains a similar link efficiency (around 95%) in various environments. As for the three traditional AQM schemes, ARED fails to control the queue in many simulation experiments including those involving heavy traffic load, varying traffic load (especially when the number of TCP connections decreases) and long delay network, therefore raising concerns regarding the adaptability of ARED despite the fact that it has better queue length STD than PI and REM. PI provides

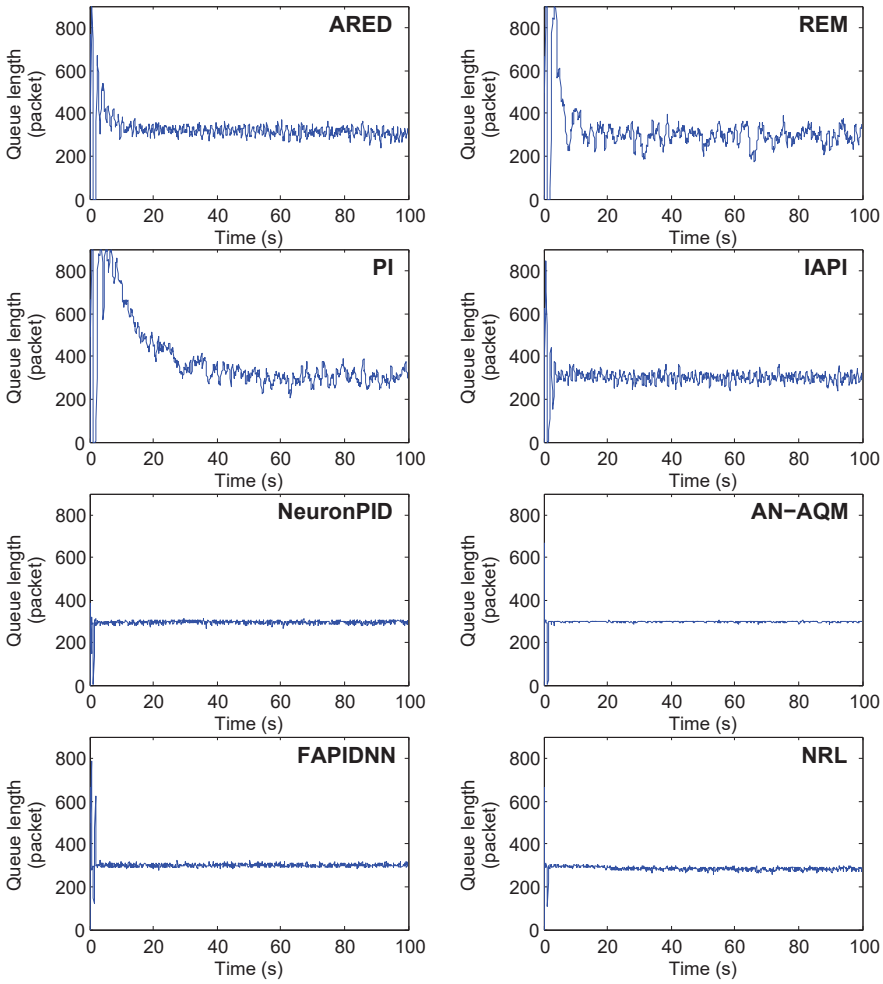

Figure 25: Queue length variations of Router D in multiple bottleneck network for $N = 100$, $N_1 = 800$, and $N_2 = 400$

the best performance in packet drop ratio. However, PI converges to the steady state in around 40 seconds and with fairly large STD of the queue length. Compared with PI, REM improves the performance by shortening the transient time to around 20 seconds and achieving smaller STD of the queue length. Note that the REM algorithm is based on both queue length and sending rate of sources, while the ARED algorithm and PI algorithm are only queue length based. Evolved from PI controller, IAPI shows significantly improvement in average queue length and queue length's STD, meanwhile IAPI provides relatively small packet drop ratio compared to the four neuron-based schemes.

Table 18: Performance metrics of Router D in multiple bottleneck network for $N = 100$, $N_1 = 800$, and $N_2 = 400$

	Link efficiency	Packet drop ratio	Average queue length	Queue length STD
ARED	94.86%	9.16%	324.78	66.61
REM	94.68%	9.55%	313.40	107.22
PI	94.70%	8.77%	384.68	162.08
IAPI	94.99%	9.65%	301.27	48.92
Neuron PID	95.13%	10.91%	293.29	25.24
AN-AQM	95.15%	11.95%	295.36	27.27
FAPIDNN	95.21%	14.75%	300.23	29.68
NRL	95.20%	14.54%	283.78	21.11

On the other hand, the four neuron-based AQM schemes have significant improvement in response time and queue jitter in all simulation runs. Neuron PID, AN-AQM and FAPIDNN successfully control the queue in all simulations. On the average, Neuron PID shows best performance in average queue length and queue length's STD, AN-AQM provides a slightly better performance in terms of packet drop ratio and queue STD than Neuron PID in more realistic network environment. FAPIDNN shows its advantage of average queue length, which is most close to the target value in most cases. Unlike the other three neuron-based schemes, NRL does not use a PID controller, and we have uncovered an undesired effect in the case where the number of TCP connections decreases randomly from 1500 to 300.

3.9. Packet delay statistics

Statistics of packet delay are important measures for user quality of service and experience. One performance metric we have considered is the time an AQM converges to the target queue length and the system fluctuation and stability around the target queue length. The motivation for maintaining constant desired queue length is primarily related to achieving predictable packet delay as perceived by the user. Another important factor that affects packet delay and its predictability is the RTT. Packet delay is a function of queueing delay, propagation time and the drop probability. The AQM can control the queueing delay and to some extent the drop probability process. However, it can never affect propagation time. These imply that there are scenarios where an AQM function cannot significantly affect the statistics of the packet delay. To illustrate such a scenario, consider a case where the propagation time is significantly longer than the queueing delay, and the variance of the packet delay is large because a variable number of retransmissions of the same packet is required due to a high and variable packet drop ratio. In such a case, having stable (almost constant) queue length will not have significant effect on the mean and variance of the packet delay. On the other hand, in a case where the queueing delay is

a dominant part of the packet delay, then lower level of queue length fluctuations will significantly reduce packet delay variance. As we have observed, the neuron-based AQM schemes are able to converge faster to the desired queue length and maintain it with less fluctuations. We will now present simulation results for the packet delay for all eight AQM schemes that will demonstrate the aforementioned effects in the cases where the propagation time is the dominant delay and when the queueing delay is dominant.

The end-to-end packet delay is defined as the time it takes for a packet sent from its source to its destination. It consists of two parts: queueing delay and propagation delay. To evaluate such performance metric, we compute by simulation the mean and STD of the end-to-end packet delay for each one of the AQM schemes under consideration.

The first set of simulations continues to be based on the network environment set in Subsection 3.2, which contains 1500 TCP connections with queue length target of 300 packets and the RTT is 80 ms. The results are provided in Table 19.

Table 19: Mean packet delay and its STD for 1500 TCP connections with the queue length target of 300 packets

	Mean packet delay (ms)	Mean packet delay STD
ARED	65.7236	37.3184
REM	59.0167	34.0178
PI	55.5232	31.5602
IAPI	59.9277	34.5685
Neuron PID	55.6888	31.7256
AN-AQM	56.8330	32.5722
FAPIDNN	57.6399	33.1321
NRL	56.2802	32.1737

The second set of simulations continues to be based on the network environment set in Subsection 3.5, where the number of TCP connections increased abruptly from 500 to 1500. Table 20 provides the corresponding result:

After conducting the simulation for the other scenarios, we observe that the mean packet delay and their STD are of the same order of magnitude for all AQM schemes. This is mainly because the propagation time is significantly longer than the queueing delay, and therefore the advantage of the neural network based AQM schemes over the traditional schemes in better controlling and stabilizing the queue length and the queueing delay is not effective in reducing the mean and variance of the packet delay.

Next, we consider an extreme case where the queueing delay is significantly longer than the propagation delay, where we set the RTT to be 2 microsecond, and where the other settings are the same as in the first group of simulations. The results are shown in Table 21:

From the above simulations, we find that when the queueing delay is significantly longer than the propagation delay, the four neuronal schemes exhibit

Table 20: Mean packet delay and its STD for the number of TCP connections increasing abruptly from 500 to 1500

	Mean packet delay (ms)	Mean packet delay STD
ARED	64.5770	36.8784
REM	57.1489	32.8302
PI	55.1649	31.3562
IAPI	57.0849	32.7856
Neuron PID	54.8945	31.1401
AN-AQM	55.1393	31.3359
FAPIDNN	55.9217	31.9410
NRL	55.8129	31.8587

Table 21: Mean packet delay and its STD for the RTT is 2 microsecond (μs)

	Mean packet delay (ms)	Mean packet delay STD
ARED	0.1235	0.0058
REM	0.0551	0.0330
PI	0.0913	0.0428
IAPI	0.0526	0.0077
Neuron PID	0.0504	0.0032
AN-AQM	0.0506	0.0033
FAPIDNN	0.0511	0.0045
NRL	0.0466	0.0036

reduction of one order of magnitude of the mean packet delays' STD relative to the traditional schemes, which implies a more predictable queueing delay. Note that the STDs of the packet delay of ARED and IAPI are one order of magnitude smaller than those of REM's and PI's. We must comment here that the actual values of the end-to-end delay statistics are very low, and will not adversely affect any quality of experience measures. However, the results here do demonstrate a behavior that is very likely to characterize scenario with larger end-to-end delay (due to larger queue length and longer propagation time). Note that such scenarios may involve a large number of packets and simulations of such cases will be computationally prohibitive.

4. Conclusion

We have presented an extensive study that provides simulation results over a wide range of conditions and scenarios. We have compared three traditional AQM schemes (ARED, PI, REM), one scheme that enhances PI (IAPI), and four recently proposed neuron-based AQM schemes (Neuron PID, AN-AQM,

FAPIDNN and NRL). Based on the comparison, we can draw the following conclusions:

- The four neuron-based schemes have exhibited faster convergence, better accuracy, smaller queue length jitter and better adaptability than the three traditional AQM schemes. As can be seen from the results, the convergence speeds of four neuron-based schemes are in the same level. The queue length averages of the neuron-based schemes are closer to the target value in the various scenarios that we have considered than the other four schemes, especially FAPIDNN shows the best accuracy in hitting the target average queue length. Once the queue has been stabilized to the target queue length, the queue length jitter of the neuron-based schemes is smaller than that of the other four schemes. Neuron PID exhibits the least queue length STD in most cases. Adaptability is another advantage of neuron-based schemes. All the neuron-based schemes are effective except that we have discovered a potentially rare scenario where NRL fails to control the queue length. In all simulation cases, the AQM schemes we considered exhibit similar mean packet delay, the reason is that the RTT takes the major part of the end-to-end delay, and the effect of queueing delay is negligible. However, when the queueing delay becomes the dominant factor of the delay, neuron-based schemes show much smaller variance, indicating that the queueing delays of four neuron-based schemes are more stable.

- Despite the strong empirical evidence for the superiority of neuron-based schemes, an unavoidable disadvantage is the lack of proof of stability of neuronal systems. Unlike the cases of traditional AQM schemes such as PI, it is hard to provide stability proof of neuronal systems due to their non-linear characteristics. Another important issue is the complexity. Compared to traditional AQM schemes, neuron system involve more parameters such as learning rates and neuron gain; also, FAPIDNN uses fuzzy controller to tune the parameters of neuronal network. This will increase the cost of both equipment and operation by requiring more energy due to additional processing requirement.

- From our simulations, we also observe certain differences in performance among the AQM systems that we use as benchmarks. The three traditional AQM schemes all have fixed parameters, which mean they cannot tune their parameters according to the highly variable network environment. Such limitation leads to low adaptability and sluggish response. For example, ARED successfully controls the queue when the traffic load is light and when capacity of the bottleneck is sufficient, but it fails in environments involving heavy and varying traffic load and limited bottleneck capacity. PI is stable in most cases, but its transient time is longest among all schemes. Compared to PI, REM considers both queue length error and sending rate mismatch, therefore it is able to shorten its transient time and to decrease the fluctuations of the queue length. The key point

of improving the performance of traditional schemes lies in the ability to tune parameters adaptively. IAPI is able to tune one of its parameters k_i to accelerate its response so it converges to the target value much faster than the three traditional schemes, such coefficient k_i is also auto-tuned according to the varying queue length error to achieve smaller fluctuations of the queue length.

In conclusion, the results demonstrate potential for neuronal AQM systems although more research is required to understand performance and stability consequences under realistic network environment.

Acknowledgments

This work was jointly supported by grants from the Australian Research Council (Grant Number DP0559131), the Natural Science Foundation (No. 60974129, No. 70931002), and the Natural Science Foundation of Jiangsu Province, China (No. BK2009388), and City University of Hong Kong (Project No. 9380044, No. 9041794, No. 7003017). We also thank Dr. Andrew Leung for his helpful comments on neural networks.

References

[1] J. Postel, RFC-793: Transmission Control Protocol, http://tools.ietf.org/html/rfc793 (Sep. 1981).

[2] M. Allman, V. Paxson, E. Blanton, RFC-5681: TCP Congestion Control, http://tools.ietf.org/html/rfc5681 (Sep. 2009).

[3] T. Bonald, M. May, J. C. Bolot, Analytic evaluation of RED performance, in: Proc. IEEE INFOCOM '00, Tel Aviv, Israel, 2000, pp. 1415–1424.

[4] H. G. Perros, K. M. Elsayed, Call admission control schemes: A review, IEEE Communications Magazine 34 (11) (1996) 82–91.

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

[6] S. Floyd, R. Gummadi, S. Shenker, Adaptive RED: An algorithm for increasing the robustness of RED, http://www.icir.org/floyd/papers/adaptiveRed.pdf (2001).

[7] J. Aweya, M. Ouellette, D. Montuno, A control theoretic approach to active queue management, Computer Networks 36 (2-3) (2001) 203–235.

[8] S. Athuraliya, S. H. Low, V. H. Li, Q. Yin, REM: Active queue management, IEEE Network 15 (3) (2001) 48–53.

[9] C. V. Hollot, V. Misra, D. Towsley, W. B. Gong, On designing improved controllers for AQM routers supporting TCP flows, in: Proc. IEEE INFOCOM '01, Anchorage, AK, USA, 2001, pp. 1726–1734.

[10] S. Ryu, C. Rump, C. Qiao, A predictive and robust active queue management for Internet congestion control, in: Proc. IEEE ISCC '03, Kemer Antalya, Turkey, 2003, pp. 991–998.

[11] Q. Chen, O. W. W. Yang, On designing self-tuning controller for AQM routers supporting TCP flows based on pole placement, IEEE Journal on Selected Areas in Communications 22 (10) (2004) 1965–1974.

[12] D. Agrawal, F. Granelli, Redesigning an active queue management system, in: Proc. IEEE GLOBECOM '04, Dallas, TX, USA, 2004, pp. 702–706.

[13] J. Sun, G. Chen, K. T. Ko, S. Chan, M. Zukerman, PD-controller: A new active queue management scheme, in: Proc. IEEE GLOBECOM'03, San Francisco, USA, 2003, pp. 3103–3107.

[14] J. Sun, K. T. Ko, G. Chen, S. Chan, M. Zukerman, PD-RED: To improve the performance of RED, IEEE Communication Letters 7 (8) (2003) 406–408.

[15] W. Feng, D. Kandlur, D. Saha, K. Shin, The blue active queue management algorithms, IEEE/ACM Transactions on Networking 20 (4) (2002) 513–528.

[16] S. S. Kunniyur, R. Srikant, An adaptive virtual queue (AVQ) algorithm for active queue management, IEEE/ACM Transactions on Networking 12 (2) (2004) 286–299.

[17] C. Wang, B. Li, Y. T. Hou, K. Sohraby, K. Long, A stable rate-based algorithm for active queue management, Computer Communications 28 (15) (2005) 1731–1740.

[18] P. Ranjan, E. H. Abed, R. J. La, Nonlinear instabilities in TCP-RED, IEEE/ACM Transactions on Networking 12 (6) (2004) 1079–1092.

[19] F. Ren, C. Lin, B. Wei, A robust active queue management algorithm in large delay networks, Computer Communications 28 (5) (2005) 485–493.

[20] J. Sun, M. Zukerman, K. T. Ko, G. Chen, S. Chan, Effect of large buffers on TCP queueing behavior, in: Proc. IEEE INFOCOM '04, Hong Kong, 2004, pp. 751–761.

[21] B. Wydrowski, M. Zukerman, GREEN: An active queue management algorithm for a self managed Internet, in: Proc. IEEE ICC '02, New York, USA, 2002, pp. 2368–2372.

[22] J. de Santi, N. L. S. da Fonseca, Design of optimal active queue management controllers for HSTCP in large bandwidth-delay product networks, Computer Networks 55 (12) (2011) 2772–2790.

[23] R. Barzamini, M. Shafiee, A. Dadlani, Adaptive generalized minimum variance congestion controller for dynamic TCP/AQM networks, Computer Communications 35 (2) (2012) 170–178.

[24] B. Braden, et al., RFC-2309: Recommendations on Queue Management and Congestion Avoidance in the Internet, http://tools.ietf.org/html/rfc2309 (Apr. 1998).

[25] B. Zheng, M. Atiquzzaman, A framework to determine bounds of maximum loss rate parameter of RED queue for next generation routers, Journal of Network and Computer Applications 31 (4) (2008) 429–445.

[26] M. P. Tahiliani, K. Shet, T. Basavaraju, CARED: cautious adaptive RED gateways for TCP/IP networks, Journal of Network and Computer Applications 35 (2) (2012) 857–864.

[27] M. Mohri, A. Rostamizadeh, A. Talwalkar, Foundations of Machine Learning, MIT Press, 2012.

[28] K. Rahnamai, K. Gorman, A. Gray, Model predictive neural control of TCP flow in AQM network, in: Fuzzy Information Processing Society, 2006. NAFIPS 2006. Annual meeting of the North American, 2006, pp. 493–498.

[29] B. Hariri, N. Sadati, NN-RED: An AQM mechanism based on neural networks, Electronics Letters 43 (19) (2007) 1053–1055.

[30] H. C. Cho, S. M. Fadali, H. Lee, Adaptive neural queue management for TCP networks, Computers & Electrical Engineering 34 (6) (2008) 447–469.

[31] J. Sun, S. Chan, K.-T. Ko, G. Chen, M. Zukerman, Neuron PID: A robust AQM scheme, in: Proc. ATNAC 2006, Melbourne, Australia, 2006, pp. 259–262.

[32] J. Sun, M. Zukerman, An adaptive neuron AQM for a stable Internet, in: Proc. IFIP Networking '07, Atlanta, USA, 2007, pp. 844–854.

[33] C. Zhou, D. Di, Q. Chen, J. Guo, An adaptive AQM algorithm based on neuron reinforcement learning, in: Proc. IEEE ICCA 2009, Christchurch, New Zealand, 2009, pp. 1342–1346.

[34] Q. Yan, Q. Lei, A new active queue management algorithm based on self-adaptive fuzzy neural-network PID controller, in: Proc. iTAP 2011, Wuhan, China, 2011, pp. 1–4.

[35] G. Chatranon, M. A. Labrador, S. Banerjee, A survey of tcp-friendly router-based AQM schemes, Computer Communications 27 (15) (2004) 1424 – 1440.

[36] G. Thiruchelvi, J. Raja, A survey on active queue management mechanisms, International journal of Computer Science and Network Security 8 (12) (2008) 130–145.

[37] R. Adams, Active queue management: A survey, IEEE Communications Surveys and Tutorials 15 (3) (2013) 1425–1476.

[38] J. Koo, S. Ahn, J. Chung, A comparative study of queue, delay, and loss characteristics of AQM schemes in QoS-Enabled networks, Computing and Informatics 22 (2003) 317–335.

[39] T. B. Reddy, A. Ahammed, Performance comparison of active queue management techniques, Journal of Computer Science 4 (12) (2008) 1020–1023.

[40] D. Lin, R. Morris, Dynamics of random early detection, ACM SIGCOMM Computer Communication Review 27 (4) (1997) 127–137.

[41] S. Wen, Y. Fang, H. Sun, Differentiated bandwidth allocation with TCP protection in core routers, IEEE Transactions on Parallel and Distributed Systems 20 (1) (2009) 34–47.

[42] R. Jain, A. Durresi, G. Babic, Throughput fairness index: An explanation, ATM Forum/99-0045.

[43] J. Sun, S. Chan, M. Zukerman, IAPI: An intelligent adaptive PI active queue management scheme, Computer Communications 35 (18) (2012) 2281–2293.

[44] A. Bitorika, M. Robin, M. Huggard, C. M. Goldrick, A comparative study of active queue management schemes, in: Proc. IEEE ICC 2004, Paris, France, 2004, pp. 1–6.

[45] Y. Du, N. Wang, A PID controller with neuron tuning parameters for multimodel plants, in: 2004 International Conference on Machine Learning and Cybernetics, Shanghai, China, 2004, pp. 3408–3411.

[46] The NS simulator and the documentation, http://www.isi.edu/hsnam/ns/.