



A self-tuning controller for queuing delay regulation in TCP/AQM networks

Ghasem Kahe¹  · Amir Hossein Jahangir²

Published online: 7 November 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract

AQM router aims primarily to control the network congestion through marking/dropping packets which are used as congestion feedback in traffic sources to balance their flow rate. However, stabilizing queuing delay and maximizing link utilization have been considered as the main control objectives, especially in media dominated networks. Usually, most of the AQM algorithms are designed for a nominal operating point. However, time-varying nature of network parameters frequently violates their robustness bounds. In this paper, a self-tuning compensated PID controller is proposed to address the time-varying nature of network conditions caused by parameter variations and unresponsive connections. The proposed scheme consists of network parameter estimation and a self-tuning AQM. Traffic load, network delay, and bottleneck link capacity are the time-varying network parameters whose variation effects should be compensated by the controller gains adaptation. As the controller gains are simply and directly obtained from the dynamic model, the obtained self-tuning controller can reasonably adapt itself to different operating conditions, while preserving the simplicity of the PI controllers. Packet-level simulations using ns2 show the outperformance of the developed controller for both latency regulation and resource utilization.

Keywords TCP networks · Congestion control · AQM · Queuing delay

1 Introduction

Due to the widespread and increasing use of the Internet, lacking any effective and operative flow and congestion control may again be vulnerable to congestion collapse which causes abrupt degradation of network performance. In the intermediate network, the active queue management (AQM) router controls the congestion in the network through marking/dropping the incoming packets, while in the entrance (traffic sources), this signal is used as a feedback for flow control, in which the flow rate is regulated according to the available bandwidth.

Historically, TCP and AQM have evolved together and have had a symbiotic relationship. When it comes to con-

gestion control, the related algorithms have evolved in such a way to cover the diversity and widespread use of today's Internet (see Fig. 1). In the early days of the Internet, the passive algorithms sufficed to resolve the congestion. As the Internet has expanded widely, active algorithms based on heuristic methods have been developed to manage the traffic congestion intelligently. However, the problematic growth of the Internet necessitates more advanced algorithms which could consider the dynamic behavior of the network. At the same time, control and optimization theory provided the tools to explore the congestion problems and dynamic behavior of the network and therefore resolve them appropriately. Implementation and computation simplicity, along with acceptable performance put the linear control theory such as PI and PIE, as the basis method that can meet most of the AQM requirements [4]. As shown in Fig. 1 and according to the outcome of the reviewed literature, we focus on the self-tuning linear control method which tries to regulate the queuing delay.

The disturbances caused by unresponsive flows such as UDP and short-lived HTTP connections seriously affect the performance of the aforementioned AQM controllers and must be taken into account in AQM controller design. Moreover, neglecting the time-varying delay and restrictions

✉ Ghasem Kahe
kahe@ari.ac.ir

Amir Hossein Jahangir
jahangir@sharif.edu

¹ Aerospace Research Institute, Ministry of Science, Research, and Technology, Tehran 14665-834, Iran

² Department of Computer Engineering, Sharif University of Technology, Tehran 14588-89694, Iran

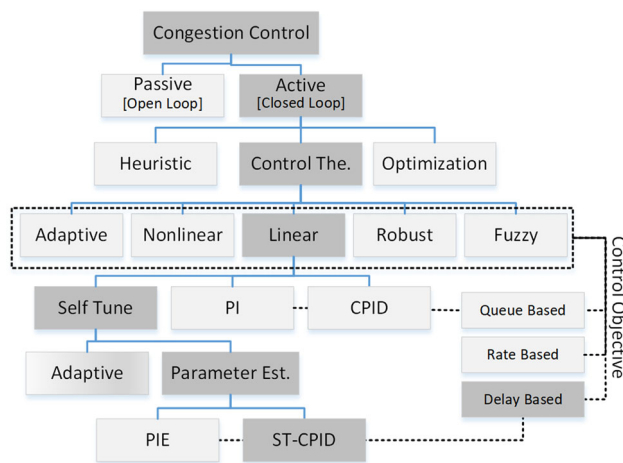


Fig. 1 Evolution of AQM algorithms and our selected approach

therefrom leads to undesirable behavior of the controller. In this paper, a self-tuning compensated proportional-integral-derivative (ST-CPID) controller is proposed, which covers the restrictions caused by network parameter variations. The scheme has two features: network parameter estimation and self-tuning AQM. A parameter-varying second-order dynamic compensator is designed to retrieve unstable internal dynamics caused by Padé approximation of the time-varying network delay. Moreover, the desired dynamics characteristics can be achieved through the dynamic compensator. The controller gains are obtained directly from the state-space model coefficients, and therefore, it does not require complex analysis and tuning approaches like the Ziegler–Nichols method. Moreover, different control objectives can be specified through the desired dynamic for the tracking error. Traffic load (N), network delay (R) and bottleneck link capacity (C) are the time-varying network parameters with which the controller must adapt itself. The network parameters are estimated from measurements made locally at the congested router. The aggregation of unresponsive flows leads to a noisy estimation of the network parameters. Due to the robustness characteristics of the proposed controller concerning to small variations of network parameters and also avoiding frequent and unnecessary retuning of the controller, a change detection algorithm is implemented not only to de-noise the estimated values but also to track fast the network variations. Simulation results using ns2 show the effectiveness of the proposed controller. The main contributions of this paper are summarized as follows:

- An improved change detection algorithm for estimating the traffic load and link capacity to have an accurate yet fast tracking filter.

- A straightforward and easy to implement self-tuning controller whose gains are obtained directly from the state-space dynamic model.
- A flexible controller which allows specifying the desired dynamics for tracking errors according to the desired objectives.

The organization of the paper is arranged as follows. The related works are reviewed in Sect. 2. In Sect. 3, the dynamic model of the TCP/AQM network is described. In Sect. 4, the estimation algorithms are explained. The design and digital implementation aspects of the compensated PID controller are presented in Sects. 5 and 6, respectively. In Sect. 7, the proposed self-tuning AQM controller is evaluated using ns2 and compared with the baseline controller, PIE. Conclusion and future works are finally presented in Sect. 8.

2 Related works

In its early days, the coverage of the Internet was limited to a few specific organizations and the traffic congestion was not an issue. Therefore, passive algorithms were considered for the queue management, which simply drop the incoming packets as the queue become full (DropTail). Fundamental problems in such passive algorithms have led to the emergence of active queue management algorithms which apply some level of intelligence based on heuristic and intuitive methods. These improvements convinced the internet engineering task force (IETF) to recommend the deployment of AQM algorithms in routers [4] such as RED [13]. Experimental results have shown that RED does not deal adequately with network heterogeneity and its false congestion avoidance scenario may escalate further bandwidth underutilization [54]. Moreover, the tuning problem of RED and its sensitivity to the network conditions [7] have limited its adequate use. Therefore, many alternatives such as RED with optimal weight parameter [56], FRED [29], DRED [2], and SRED [38] have been proposed so far. Based on RED, an adaptive queue management is proposed in [26] which incorporates both the average and rate of change of the queue length to resolve the problem of excessive packet dropping in RED due to the frequent crossing of the maximum threshold value. Each of such algorithms has focused on a particular problem of RED with none being able to offer a comprehensive solution. Based on CHOKe [39], a novel fairness-driven AQM scheme, CHOKeH, have been presented in [1] to identify and restrict unresponsive traffic from dominating the bandwidth.

In this category, more advanced AQM algorithms have been developed using heuristic optimization approach. An ant colony-based approach called ant-inspired level-based congestion control (AILCC) is presented in [43] to man-

age bandwidth issues effectively. To meet various bandwidth demands from numerous applications, AILCC focuses primarily on providing an efficient congestion control mechanism. An adaptive traffic prediction based AQM (ATPAQM) algorithm has been proposed in [36] in which the combination of the Kalman filtering model and the online noise estimation is used to predict traffic accurately. The traffic prediction results are used to reflect congestion indication in the proposed AQM algorithm.

Meanwhile, without considering the network dynamic behavior and its time-varying nature, the RED algorithm, its alternatives, and the aforementioned heuristic algorithms cannot provide the appropriate efficiency and performance and cannot guarantee stable behavior over a wide range of network conditions [8]. Development of the control theory in engineering and optimization and resource allocation theory in the economy have opened up new horizons to explore the congestion problem. Using optimization and resource allocation theory, network congestion has been investigated from a new point of view. Authors of [31] compare four neural AQM schemes, namely, Neuron PID [47], AN-AQM [46], FAPIDNN [53], and NRL [58], versus ARED, REM, and PI together with a modified PI scheme named IAPI [48] over a wide range of conditions and scenarios. Their simulation results for ARED, REM, and IAPI show large queue fluctuations in long-delay networks. Therefore, these cannot be considered as effective schemes in long-delay networks. They demonstrate that the neuron-based schemes have faster convergence to queue length target, with smaller queue length jitter, while no proof of stability is available for them [31].

Computation and implementation complexity along with the hardly tuning problem of optimization-based algorithms divert the research to the control theory. To incorporate all aspects of the AQM algorithm, the control theory has been taken into account with [35], boosting such an approach through proposing a fluid-flow dynamic model of TCP/AQM networks. Based on the developed dynamic models, various AQM controllers have been proposed using different control approaches: proportional-integral (PI) and proportional-integral-differential (PID) controllers [19,20,44], nonlinear controllers [15], and robust controllers [32,42,57]. Among them, PI controllers are gaining more attention because of their implementation and computation simplicity [49]. A methodology to compute a non-fragile PI AQM controller is provided in [34] in which a new method for tuning the parameters of PI/AQM controller has been proposed. Comparison of various PI controllers designed for the delay-model of TCP/AQM in [49] shows that the non-fragile PI AQM [34] is less robust compared with the others, because their proportional or integral gain margins are relatively large. Although these controllers fulfill the desired objectives for a nominal operating condition, the high variation of network operating conditions frequently violates their

robustness and performance. Moreover, the computational complexities and implementation problems of the proposed schemes [32,42,57] are contrary to the recommended features for AQM in [4]. Furthermore, conservative designs for the worst-case conditions lead to their degraded performance.

Network parameters are time-varying. Unresponsive short-lived flows, user datagram protocol (UDP) connections, the requirements of service level agreement (SLA), and quality of service (QoS) frequently alter the bottleneck link capacity experienced by long-lived flows. The effective number of users utilizing each link is also usually variable. These variations frequently violates the robustness bounds of the controller. The time-varying nature and heterogeneity of network parameters necessitate a self-tuning structure for the AQM controllers leading to the development of various adaptive and self-tuning AQM controllers, which is essential for the future Internet architecture [11]. A self-tuning PI (STPI) controller is presented in [55] in which the link capacity and the traffic load are estimated, and then the parameterized PI controller is automatically retuned for the new operating condition. Authors of [52] propose a self-configuring PI controller in terms of the estimated traffic load. In both of the latter works, the controller gains are obtained through the phase and gain margin analysis which does not lead to a straightforward tuning method. An adaptive RED (ARED) is proposed in [12,14] in which the RED parameters are tuned automatically according to the network conditions. The low-pass filter used for queue averaging leads to the sluggish and indolence behavior of the RED algorithm and its alternative, ARED [19].

Adaptive control has also been used to develop AQM algorithms. The model predictive control scheme is used in [33,50,51] to develop an adaptive controller based on the identification of network dynamics. In [3], an optimization scheme based on online stochastic approximation has been proposed to dynamically tune the parameters of AQMRD [26]. The optimization problem, which must be solved over a prediction and controlling horizon in each sampling time instant, complicates the computation and implementation aspects of them, which is crucial for deployment in edge and core routers. A self-tuning AQM controller has been proposed in [6] based on the pole placement approach to adapt the controller to traffic load changes. Without considering unstable internal dynamics caused by Padé approximation of the time-varying delay, the internal dynamics may conduct the system to the undesired states.

The growing concern about the multimedia in today's Internet, yields more sensitivity about the end-to-end latency experienced by end users [28]. Accordingly, IETF has released some delay sensitive AQM algorithms such as CoDel [37] and PIE [40,41]. Unlike their predecessors which are sensitive to the queue length, these algorithms directly control the queuing delay to meet the requirements for real-

time and delay-sensitive applications. Using timestamps for each packet, CoDel directly controls latency and drops the dequeued packet if the queuing delay blows over the predefined setting. Due to the expensive requirements of CoDel for implementation and operation, a lightweight algorithm, PIE (Proportional Integral Controller Enhanced), has also been proposed, which benefits from the advantages of both RED and CoDel: it is easy to be implemented like RED, while it directly controls latency like CoDel [40]. Evaluation of CoDel, PIE, and ARED for various static and dynamic scenarios in [23,45] shows that PIE achieves significantly lower delays than the other two in static scenarios, while CoDel and ARED recover significantly faster than PIE from traffic changes. Experimental evaluations using real-world implementations in a wired testbed in [28] show that ARED performs worse than CoDel and PIE only when the number of traffic flows on the bottleneck link is very small. PI2 [9] provides an alternate and simpler design and implementation to the PIE algorithm. The heuristic scaling steps for controller gains in PIE are replaced by squaring the controller output instead, which has less complexity and improves stability in some cases. While the PIE controls the congestion very aggressively, resulting queue length oscillation, the PI2 encodes the drop/mark probability into a congestion signal, which is either the square of drop/mark probability for the case of classic controls like TCP Reno, or no encoding is needed for scalable controls like DCTCP. A similar structure is derived analytically in the proposed controller in which, the unstable internal dynamics is compensated according to the desired dynamics and then the PID controller is used to control the congestion. In this paper, we try to cover some of the PIE problems using a compensated PID controller with self-tuning capability. The controller gains are obtained directly from the state-space model which can be computed according to the network parameters. The obtained controller preserves the simplicity of PIE while covering more accurately the objectives with comparable fewer fluctuations.

3 TCP/AQM dynamic model

Various dynamic models have been developed for the TCP/AQM networks. The model given in [35] is used widely incorporating fluid-flow and stochastic differential equations to describe the dynamic behavior of the TCP/AQM networks. In this model, the congestion window size and the packet latency are used as state variables.

$$\begin{cases} \dot{W}(t) = \frac{1}{R(t)} - \frac{W(t)}{2} \frac{W(t-R(T))}{R(t-R(T))} P(t-R(T)) \\ \dot{R}(t) = N \frac{W(t)}{CR(t)} - 1 \end{cases} \quad (1)$$

The following nomenclature will be used hereafter:

$W(t)$: TCP congestion window size (packets)
 $q(t)$: router's queue length (packets)
 $R(t)$: round trip time (s) = $q(t)/C + \tau_p$,
 C : link capacity (packets/s)
 τ_p : propagation delay (s)
 N : traffic load (number of TCP sessions)
 $P(t)$: packet drop probability
 q_{max} : queue capacity (packets)
 W_{max} : maximum congestion window size (packets)

The congestion avoidance phase of TCP protocol is modeled in the first equation of (1); the $1/R(t)$ term, models the additive increase of congestion window length in response to received packet acknowledgement, while the $W(t)/2$ term models the multiplicative decrease of congestion window length in response to packet marking/dropping probability $P(t)$. The dynamics of queuing delay is modeled in the second equation of (1).

4 Estimation of network parameters

Traffic load, network delay, and bottleneck link capacity are time-varying network parameters. In this section, the traffic load is estimated based on the dynamic behavior of congestion window length. The bottleneck link capacity is estimated by measuring the rate of departed packets. The network delay is composed of a fixed but heterogeneous terms; propagation delay, and a time-varying term, queuing delay. Since the realistic network has heterogeneous TCP flows with different propagation delays, the propagation delay should be assumed as a bounded quantity. The queues of non-bottleneck links are almost empty and their queuing delays are negligible. The queuing delay of the bottleneck link can be estimated from measurements made locally at the congested router.

4.1 Change detection algorithm

Although a fast filter follows quickly the fast variation of the signal, it can not attenuate the signal noise. Though, a slow filter suppresses the signal noise accurately, it lags from the fast variation of the signal. Therefore, there is an inconsistency between accuracy and speed of the filter, which can be compromised using a change detector along with adaptive filter. Change detector is a nonlinear filter used for signals with sharp and sudden variations such as traffic load and link capacity [30]. For example, an algorithm combining a Kalman filter and a change detection algorithm (CUSUM) is proposed in [22] for RTT estimation. Furthermore, due to the robustness characteristics of the proposed controller with respect to the small variations of the network parameters and also in order to avoid frequently and unnecessary retuning of the controller, which leads to the instability and sluggish

behavior, a change detection algorithm is developed to not only remove the estimation noise, but also to track the fast changes of the network conditions.

By using a change detector with an adaptive filter which is tuned according to the robustness characteristics, the controller is retuned whenever a considerable change of the network parameters is announced. The proposed change detector decides according to both level and speed of parameter changes and is composed of two parts: distance measure and stopping rule. It is typically a modified version of CUSUM along with recursive least square (RLS) or least mean square (LMS) filter [16]. The parameters are estimated using RLS or LMS, and the distance measure is computed according to the last change. The stopping rule works based on the defined threshold, and the choice of threshold depends not only on the robustness bounds but also on the accuracy requirement of the controller. Furthermore, it must avoid the frequent retuning of the controller, which may lead to the sluggish behavior of the system. According to the robustness characteristics of the proposed algorithm, if the estimated network parameters (traffic load or link capacity) varies for more than 10 percent, the change detector alarms for re-tuning the controller. Therefore, in our simulations the threshold set to 10 percent of the network parameter value. These algorithms are described more in the next sections.

4.2 Estimation of traffic load

Estimation of the effective number of users has many applications in communication networks. Router assisted congestion controllers, such as XCP [27], R-XCP [17] and RCP [10], use the traffic load to advertise the congestion window to traffic sources. In this paper, the traffic load is estimated according to a refined version of TCP/AQM dynamic model (1). Revising (1) in terms of traffic rate $X(t)$ and using $W(t) = X(t)R(t)$, yields:

$$\begin{cases} \dot{W}(t) = +X(t - R(t))(1 - P(t - R(t)))\frac{1}{X(t)R(t)} \\ \quad - X(t - R(t))P(t - R(t))\frac{X(t)R(t)}{2} \\ \dot{q}(t) = +NX(t) - C \end{cases} \quad (2)$$

The first term in the first equation of (2) models the additive increase while the second term models the multiplicative decrease of congestion window length. The achieved dynamic model has an equilibrium point which is obtained by zeroing the dynamics of both congestion window length and queue length as follows:

$$\begin{cases} 0 = \frac{1-P_0}{R_0} - \frac{X_0^2 R_0 P_0}{2} \\ 0 = N_0 X_0 - C \end{cases} \quad (3)$$

Using the first equation of (3) and substituting R_0 , the transmission rate of each source in equilibrium point is obtained by:

$$X_0 = \frac{1}{\tau_p + q_0/C} \sqrt{\frac{2(1 - P_0)}{P_0}} \quad (4)$$

As it can be seen, the equilibrium transmission rate is achieved in terms of P_0 and q_0 , which are measurable locally in the router. According to the second equation of (3), the traffic load is obtained as the ratio of the bottleneck link capacity over the estimated transmission rate as follows:

$$\hat{N} = \frac{\hat{C}}{X_0} \quad (5)$$

where \hat{N} is the estimated traffic load. While the traffic load is noisy due to short-lived flows, it also has abrupt changes that necessitate the change detector not only to remove the noise but also to track the fast variations of the signal. The distance measure g_k^N is computed according to both level and rate of changes of the traffic load, and stopping rule ζ_N is determined according to the robustness characteristics of the controller with respect to the traffic load uncertainties.

An adaptive filter (normalized least mean square—NLMS) with change detection [16] is used to attenuate noise as well as to track fast the abrupt changes of the traffic load. Algorithm 1 shows the CUSUM NLMS filter for estimation of the traffic load \hat{N} . In order to lower the influence of the input signal amplitude on the gradient noise, the NLMS algorithm is used in which the step size λ_N is scaled and normalized by the variance of the input signal X . In this algorithm, X_k

Algorithm 1 The CUSUM NLMS Filter for Estimation of the Traffic Load

```

 $\lambda_k \leftarrow \frac{\lambda_N}{1+X_k^2}$ 
 $\hat{N}_{k+1} \leftarrow \hat{N}_k + \lambda_k X_k (Y_k - N_k X_k)$ 
 $\epsilon_N \leftarrow Y_k - N_k X_k$ 
 $g_k^N \leftarrow \max(g_{k-1}^N + \epsilon_N - \zeta_N, 0)$ 
if  $g_k^N > h_N$  then
    alarm to retune the controller
     $g_k^N \leftarrow 0$ 
     $\hat{N}_{k+1} \leftarrow C/X_k$ 
end if
    
```

is the traffic rate obtained from (4), $Y_k = \hat{C}$ is the measured link capacity, λ_N is the step size, and λ_k is the normalized step size.

4.3 Estimation of link capacity

Due to the other rate-varying traffic such as UDP, SLA, and QoS requirements, the bottleneck link capacity is time-

varying. This parameter can be estimated simply by measuring the rate of departed packets which is equal to the ratio of departed packets to the router’s busy time. The bottleneck link capacity can also be estimated using spectral and statistical analysis of traffic. Packet transmission frequency is impressed from the packet size and link bandwidth. This frequency can be seen strongly in the spectral representation of the aggregate traffic observed at a downstream monitoring point [18]. Therefore, using the power spectral density of the traffic, the bottleneck link capacity could be estimated. Both of these estimation schemes for link capacity are noisy along with sharp variations which necessitate its filtering using an adaptive filter (Recursive Least Square - RLS) with change detector [16]. Algorithm 2 shows the CUSUM RLS filter for estimation of the link capacity. The distance measure g_k^C is computed according to both level and rate of changes of link capacity and stopping rule ζ^C , is defined according to the robustness characteristics of the controller with respect to the link capacity uncertainties. In this algorithm, d_k is the

Algorithm 2 The CUSUM RLS Filter for Estimation of Link Capacity

```

 $Y_k \leftarrow \frac{d_k}{\Delta t}$ 
 $\hat{C}_k \leftarrow \lambda_c \hat{C}_{k-1} + (1 - \lambda_c) Y_k$ 
 $\epsilon_c \leftarrow Y_k - \hat{C}_{k-1}$ 
 $g_k^c \leftarrow \max(g_{k-1}^c + \epsilon_c - \zeta_c, 0)$ 
if  $g_k^c > h_c$  then
    alarm to retune the controller
     $g_k^c \leftarrow 0$ 
     $\hat{C}_{k+1} \leftarrow Y_k$ 
end if
    
```

departed packet size, Δt is the sampling interval, λ_c is the forgetting factor, and \hat{C}_k is the estimated link capacity.

5 Design of the self-tuning AQM controller

Having the network dynamic model, the objective of this study is to design an AQM controller capable of achieving asymptotic stability of the TCP/AQM network. The AQM controller aims to stabilize the queuing delay and jitters through accurate regulation of the packet latency (queuing delay) whose accurate operation also improves the other objectives: maximizing the link utilization and minimizing the packet loss. Thereby, it can achieve predictable delay according to the QoS requirements. The other control objective is the robustness of the congestion control algorithm against the disturbance on the network parameters which are time-varying. The packet marking/dropping probability, $P(t)$, is used as the control action, and the router queuing delay is the measurable output. The time-varying network parameters are network delay, traffic load and bottleneck link

capacity in which the controller should adapt itself with their variations.

5.1 Delay-based dynamic model

We develop the controller based on a linearized parameter-varying delay-based dynamic model. Assuming the traffic load $N(t)$, and delay term $R(t)$ are constant, the linearized model is obtained as follows [19]:

$$\begin{cases} \delta \dot{W}(t) = -\frac{N}{CR_0^2} (\delta W(t) + \delta W(t - R_0)) \\ \quad -\frac{1}{R_0^2} (\delta R(t) - \delta R(t - R_0)) \\ \quad -\frac{C^2 R_0}{2N^2} \delta P(t - R_0) \\ \delta \dot{R}(t) = +\frac{N}{CR_0} \delta W(t) - \frac{1}{R_0} \delta R(t) \end{cases} \tag{6}$$

Ignoring the internal delay as the high frequency small gain parasitic terms [19], the transfer function becomes:

$$F(s) = \frac{\delta R(s)}{\delta P(s)} = -\frac{\frac{C}{2N} e^{-R_0 s}}{\left(s + \frac{2N}{CR_0^2}\right) \left(s + \frac{1}{R_0}\right)} \tag{7}$$

which comprises of two stable poles $(-1/R_0, -2N/(CR_0^2))$ and an input/output delay (R_0). In order to handle the input/output delay and considering it in controller design with a finite-dimensional form, it is approximated with a first order Padé rational function as:

$$e^{-sR_0} = \frac{1 - \frac{R_0}{2}s}{1 + \frac{R_0}{2}s} \tag{8}$$

The unstable zero transfers the dynamic system into a non-minimum phase system. The non-minimum characteristics caused by unstable internal dynamics restrict the direct application of conventional control techniques. To circumvent this restriction, the unstable internal dynamics should be compensated through an appropriate compensator not only to capture the unstable zero, but also to achieve an appropriate dynamics for the tracking error. The achieved third order system is simplified in order to utilize the well-known form of the second order system dynamics. Hence, the reduced order model is obtained as follows:

$$F(s) = \frac{\frac{R_0 C}{4N} \left(s - \frac{2}{R_0}\right)}{\left(s + \frac{2N}{CR_0^2}\right) \left(s + \frac{1}{R_0}\right)} \tag{9}$$

The state-space equivalent of (9) is acquired as:

$$\begin{cases} \dot{x}_1(t) = x_2 \\ \dot{x}_2(t) = -a_0 x_1(t) - a_1 x_2(t) + u(t) \\ y(t) = b_0 x_1(t) + b_1 x_2(t) \end{cases} \tag{10}$$

where R_0 is the delay parameter, $a_0 = (2N)/(CR_0^3)$, $a_1 = 1/R_0 + 2N/(CR_0^2)$, $b_0 = -C^2/(2N)$ and $b_1 = (C^2R_0)/(4N)$. Since, the relative degree of the obtained second-order system (10) is $r = 1$, the overall stability of the whole system depends upon not only to the stability of input/output pair, but also to the stability of the internal dynamics. The normal form transformation [21] is used to distinguish the internal dynamics from the input/output dynamics as follows:

$$W(t) = [\xi(t) \eta(t)]^T \leftrightarrow \Phi(x) = [y(t) x_1(t)]^T$$

The obtained model is:

$$\begin{cases} \dot{\eta}(t) = a_{11}\eta(t) + a_{12}\xi(t) \\ \dot{\xi}(t) = a_{21}\eta(t) + a_{22}\xi(t) + \beta u(t) \\ y(t) = \xi(t) \end{cases} \quad (11)$$

where $a_{11} = -b_0/b_1$, $a_{12} = 1/b_1$, $a_{21} = -b_0^2/b_1 - a_0b_1 + a_1b_0$, $a_{22} = b_0/b_1 - a_1$ and $\beta = b_1$ are the corresponding coefficients. The obtained zero dynamics of the system has an unstable eigenvalue which is equal to $2/R_0$. Therefore, the internal dynamics is unstable for all positive time delays and must be considered carefully in the controller design.

5.2 AQM controller design

A parameter-varying dynamic compensator, which is achieved according to the tracking error and the unstable internal dynamic, can cover the problem of non-minimum phase system limitations caused from Padé approximation of the delay. Figure 2 shows the proposed structure consisting of the compensator ($E(s)$) and controller ($C(s)$). Primarily, using the proposed method in [25] the unstable internal dynamics of the delay-based dynamic model described in Sect. 5.1, is compensated as follow:

$$\left[\frac{d^2}{dt^2} + m_1 \frac{d}{dt} \right] \eta(t) + n_0 e(t) = 0. \quad (12)$$

The acquired dynamic compensator has two unknown coefficients: n_0 and m_1 . These two parameters should be determined according to the tracking error and the unstable internal dynamics. By designating a second order dynamic system with suitable characteristics as the desired error

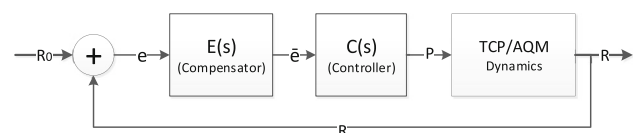


Fig. 2 AQM controller structure

dynamics, the unknowns can be calculated. Rewriting (12) and using $\dot{\eta}(t) = \bar{e}(t)$, the dynamic compensator with parameter-varying characteristics is achieved as:

$$\dot{\bar{e}}(t) + m_1 \bar{e}(t) + n_0 e(t) = 0 \quad (13)$$

While the proposed compensator captures the unstable internal dynamics and conducts the closed-loop system into a desired region of state-space, it allows to specify an appropriate dynamics for tracking error. Then, according to the acquired compensator (12), the PID controller is obtained as follow [25]:

$$u(t) = k_p \bar{e}(t) + k_d \dot{\bar{e}}(t) + k_i \int_0^t \bar{e}(\rho) d\rho \quad (14)$$

where \bar{e} is the filtered and compensated error based on the dynamic compensator (12). The PID control parameters (proportional k_p , derivative k_d , and integral k_i) are obtained as $k_p = a_1$, $k_d = 1$, and $k_i = a_0$, which are related directly to the state-space parameters in (10). Hence, the controller parameters are obtained directly from the state-space model coefficients and the achieved PID controller does not require to be tuned over the system operating envelop.

5.3 Parametrization of AQM controller

The PID controller parameters are directly obtained from the state-space model of TCP/AQM network. Hence the achieved controller is excellently parameterized in terms of network parameters, and it does not require further gain calculation approaches like the Ziegler–Nichols method. The compensator is designed not only to stabilize the internal dynamics, but also to specify an appropriate dynamics for the tracking error. Therefore the parameters of the compensator are computed according to the unstable internal dynamics and the desired dynamics for the tracking error. According to the control objectives, the desired dynamics for the tracking error is specified through a well-known second order dynamic system $\ddot{e}(t) + 2\zeta w_n \dot{e}(t) + w_n^2 e(t) = 0$. To calculate the compensator unknowns, we equalize the achieved dynamic model for compensator (12) and the desired dynamic model for the tracking error. Hence, the compensator parameters, m_1 and n_0 , are obtained in terms of ζ and w_n and internal dynamics parameters, a_{11} and a_{12} as follows:

$$\begin{cases} m_1 = 2\zeta w_n + \frac{w_n^2}{a_{11}} \\ n_0 = \frac{a_{12} w_n^2}{a_{11}} \end{cases} \quad (15)$$

We investigate the effect of these two parameters (w_n and ζ) on the correct operation of ST-CPID. To do this, we simulate the ST-CPID for different w_n and ζ : w_n from 1.0 to 6.0 and ζ from 0.1 to 1.0, respectively. Figure 3 shows the result of

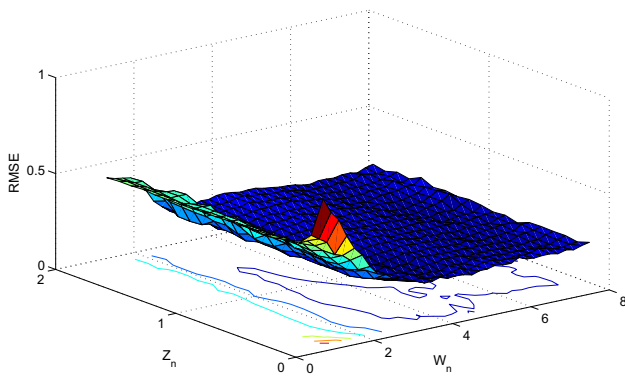


Fig. 3 RMSE of queuing delay with respect to the w_n and ζ

these simulations. As it can be seen, the root mean square error (RMSE) of the queuing delay is less affected by the variation of compensator parameters. However, according to these results, it is recommended that w_n be between 3.5 and 6 and ζ be between 0.5 and 0.8.

6 Packet-level implementation

Selecting the zero-order-hold transform and the sampling interval T_s , the PID controller (14) in discrete time domain is achieved as follows:

$$p(k) = d_0\delta R(k) + d_1\delta R(k-1) + d_2\delta R(k-2) + p(k-1) \quad (16)$$

where $u(k) = p(k)$, $\bar{e} = \delta R(k) = R(k) - R_{ref}$, $d_0 = k_p + k_d/T_s = a_1 + 1/T_s$, $d_1 = -k_p - 2k_d/T_s + k_i T_s = -a_1 - 2/T_s + a_0 T_s$, $d_2 = k_d/T_s = 1/T_s$, and R_{ref} is the desired queuing delay. The sampling frequency $F_s = 1/T_s$ is typically selected 10 times of the loop bandwidth [19]. Similarly, the compensator (13) in discrete time domain is obtained as follows:

$$\bar{e}(k) = a_c \bar{e}(k-1) + b_c e(k-1) \quad (17)$$

where $a_c = 1 - m_1 T_s$ and $b_c = n_0 T_s$. At every sampling time instance, the estimation of the traffic load and the link capacity are updated, filtered out, and then the controller is retuned accordingly. Due to the robustness characteristics of the controller, whenever a considerable change is announced through the change detector, the controller is retuned for the new operating conditions. Hereinafter, the self-tuning compensated PID controller is called ST-CPID.

7 Packet-level evaluations

In this section, the proposed controller and its adaptive scheme are evaluated under various simulation scenarios

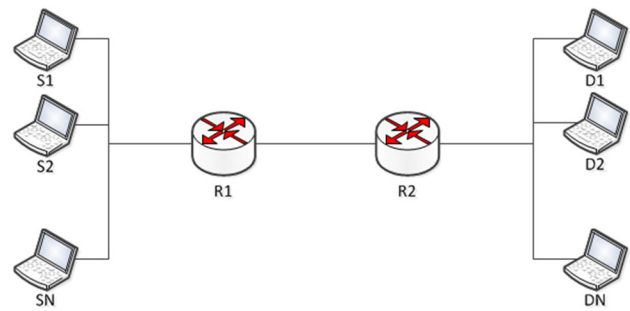


Fig. 4 Single bottleneck dumbbell topology

through network simulator, ns2, with a sampling frequency of $F_s = 100$ Hz. The simulation scenarios are regulated according to ones used in [55] and other related papers. The proposed self-tuning compensated PID controller (ST-CPID) is evaluated using a single-gateway dumbbell topology (see Fig. 4) which is widely used to evaluate the AQM controller [33,50]. However, we extend the evaluations to the case of a network topology with multiple bottlenecks to show the scalability of the proposed AQM controller.

7.1 Varying traffic load

In the first simulation scenario, the proposed controller is evaluated under varying traffic load using the well-known single bottleneck dumbbell topology, where $C = 50$ Mb/s or $C = 6250$ packets/s, link delay ranges uniformly between 80 and 260 ms, $packet_size = 1000$ byte, and $N(t)$ is time-varying: $N(0 : 60) = 50$, $N(60 : 120) = 100$, $N(120 : 180) = 200$, $N(180 : 240) = 300$, $N(240 : 300) = 400$, and $N(t_1 : t_2)$ is the traffic load from t_1 to t_2 . Using the proposed change detector, the estimated value of the traffic load is continuously monitored, and by detecting a considerable change in the traffic load, the controller is retuned for the new operating point. Figure 5 depicts the output of change detector along with the estimator. Despite the high-frequency oscillations of the estimated value, retuning is just applied when the change is considerable and its variance remains approximately bounded.

As it can be seen in the figure, the change detector not only removes the noise accurately, but also tracks the fast variations of the traffic load at time instances of $t = 60, 120, 180$, and 240 s. Using the estimated value of the traffic load, the controller adapts itself to the variations of the traffic load. Figures 6 and 7 give the evolution of control action $P(t)$ and queuing delay versus time for both ST-CPID and PIE and Table 1 shows the performance criteria. The figures show that, the ST-CPID controller regulates accurately the queuing delay under varying traffic load from light to heavy traffic load. While ST-CPID shows a stable behavior under varying

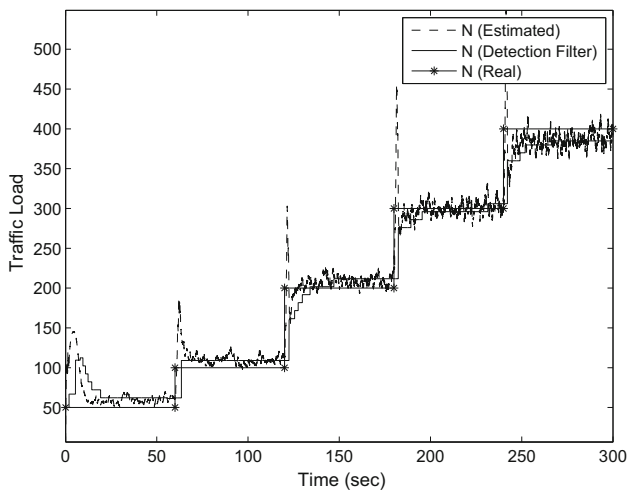


Fig. 5 Evaluation of change-detection algorithm on the estimated traffic load in a network with 50, 100, 200, 300, and 400 FTP flows at time interval 0–60, 60–120, 120–180, 180–240, and 240–300 s, respectively

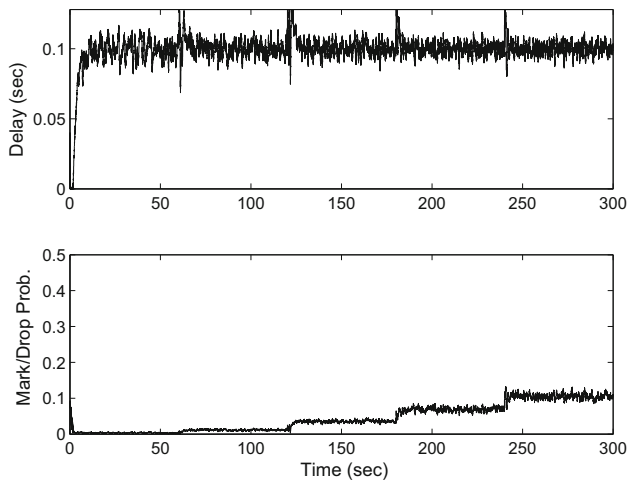


Fig. 6 Evaluation of the ST-CPID to regulate the queuing delay under varying traffic load with 50, 100, 200, 300, and 400 FTP flows at time interval 0–60, 60–120, 120–180, 180–240, and 240–300 s, respectively

traffic load, the PIE shows sluggish and near unstable around light traffic load.

We extend the evaluation of the proposed controller under more realistic conditions: noisy and time-varying traffic load. The aggregated short-lived flows on high-bandwidth links behave as a Gaussian noise [24]. The simulation setup is not changed except that $C = 15$ Mbps, link delay ranges uniformly between 160 and 240 ms, and the short-lived http flows with rate 200 flows/s are additionally generated using the Pack-Mime module from Bell-Labs [5]. Figure 8 gives the time evolution of the real traffic load, the estimated traffic load, and the filtered traffic load. Despite the noisy traffic, the estimation algorithm follows the real value but overestimates the number of long-lived flows. Since, the aggregation of short-lived flows behaves as a Gaussian noise [24], an increase in

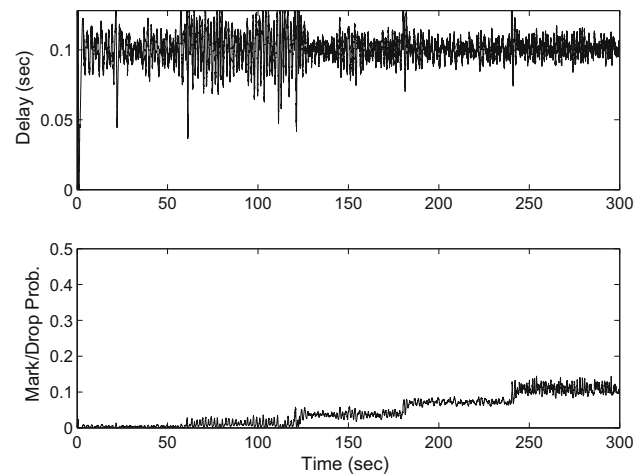


Fig. 7 Evaluation of PIE to regulate the queuing delay under varying traffic load with 50, 100, 200, 300, and 400 FTP flows at time interval 0–60, 60–120, 120–180, 180–240, and 240–300 s, respectively

Table 1 Performance measures under varying traffic load

	avg(s)	std(s)	RMSE	Loss%	Util%
STCPID	0.0996	0.010	0.10	0.046	0.99
PIE	0.1004	0.012	0.12	0.048	0.99

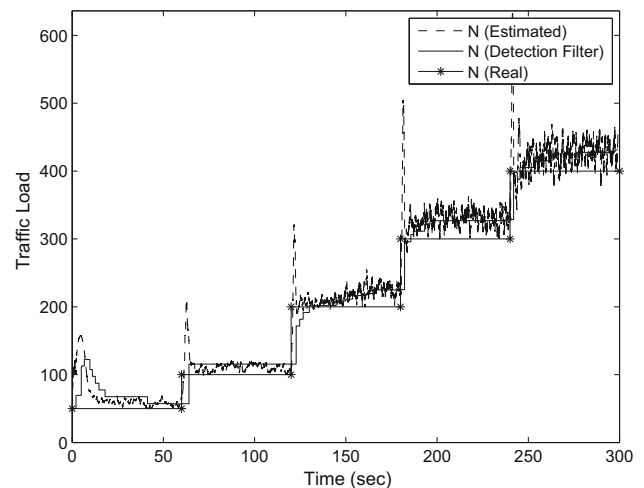


Fig. 8 Evaluation of change-detection algorithm on the estimated traffic load under noisy traffic and in a network with 50, 100, 200, 300, and 400 FTP flows at time interval 0–60, 60–120, 120–180, 180–240, and 240–300 s, respectively

the rate of incoming traffic can be seen. This increase shows itself as additional long-lived flows in the estimation process. Figure 8, gives the time evolution of the estimated traffic load and the change detector. When the traffic is a combination of responsive and unresponsive flows, the estimated value becomes noisy. So the change detector discovers the considerable changes and avoids the frequently retuning of the controller for small changes.

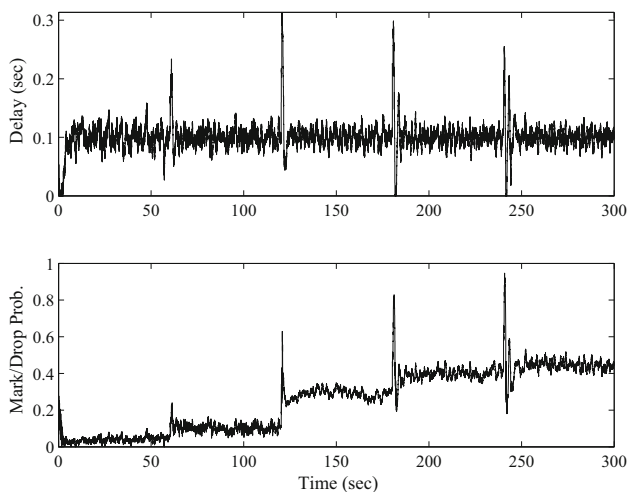


Fig. 9 Evaluation of ST-CPID to regulate the queuing delay under varying and noisy traffic load with 50, 100, 200, 300, and 400 FTP flows at time interval 0–60, 60–120, 120–180, 180–240, and 240–300 s, respectively

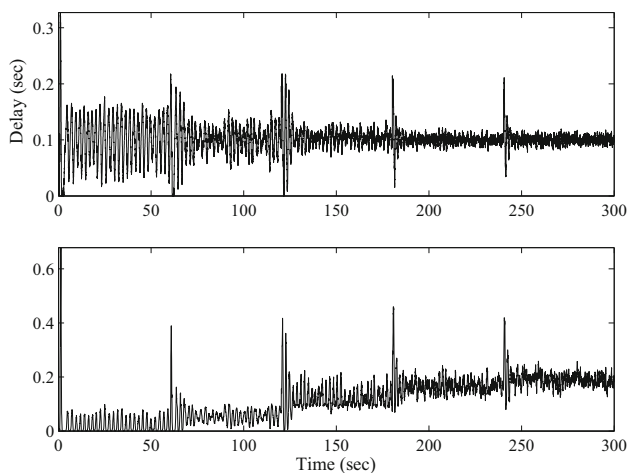


Fig. 10 Evaluation of PIE to regulate the queuing delay under varying and noisy traffic load with 50, 100, 200, 300, and 400 FTP flows at time interval 0–60, 60–120, 120–180, 180–240, and 240–300 s, respectively

Table 2 Performance measures under varying traffic load

	avg. delay (s)	std. delay (s)
ST-CPID	0.1006	0.020
PIE	0.1018	0.026

Figures 9 and 10 show the variation of control action and queuing delay versus time for both ST-CPID and PIE. The proposed controller adapts itself with the variation of traffic load and regulates the queuing delay accurately on the desired value, but on the other side, PIE tries to regulate the delay with oscillatory behavior (see Table 2).

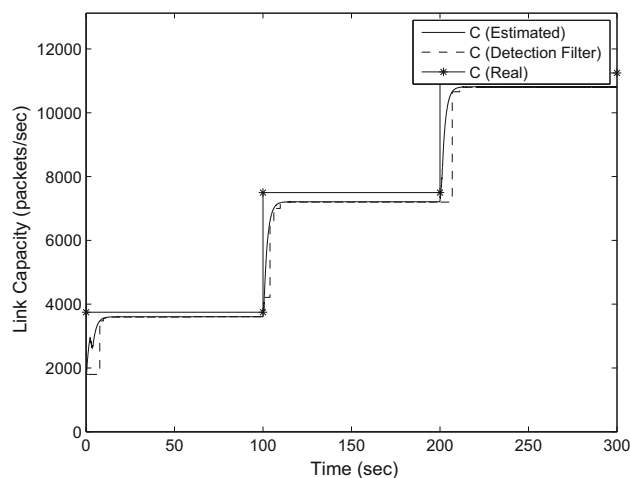


Fig. 11 Evaluation of change-detection algorithm on estimated link capacity in a network with 15, 30, and 45Mbps at time interval 0–100, 100–200, and 200–300 s, respectively

7.2 Varying link capacity

Unresponsive short-lived flows, UDP connections, and the scheduling algorithm frequently alter the bottleneck link capacity experienced by long-lived flows. In the second series of simulations, the proposed controller is evaluated under varying bottleneck link capacity. The simulation setup is like the first scenario but instead of traffic load, the bottleneck link capacity is varying. At time $t = 100$ s and $t = 200$ s, the bottleneck link capacity is increased from $C = 15$ Mbps to $C = 30$ Mbps and $C = 45$ Mbps, respectively. A change detector is designed to detect a considerable and steady change of the estimation. By detecting a considerable change in bottleneck link capacity, the controller is returned for the new operating conditions. Figure 11 shows the variation of estimated link capacity and change detector versus time. To regulate the queuing delay accurately over varying link capacity, the AQM controller should set different queue length according to the varying link bandwidth. As presented in Fig. 12, both the ST-CPID and PIE change the queue length with respect to the link bandwidth, to minimize the queuing delay variation. Figures 13 and 14 show the time evolution of control action and queuing delay for both the proposed controller and the baseline controller PIE under varying link capacity. The ST-CPID regulates the queuing delay accurately with less fluctuations (see Table 3).

7.3 Multiple bottleneck topology

In this section we evaluate the ST-CPID under more realistic network topology, known as multiple bottleneck network topology (see Fig. 15)). The aforementioned topology has also been used in various related papers [15,50]. This topology consists of a total of 6 routers (R_1 to R_6) with 120 FTP

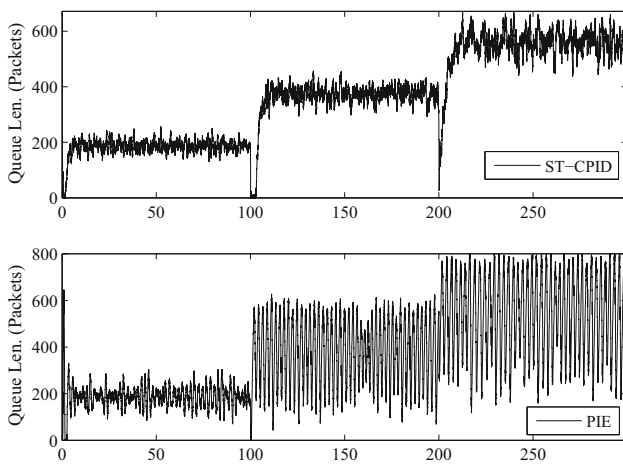


Fig. 12 Evolution of queue length over varying link capacity

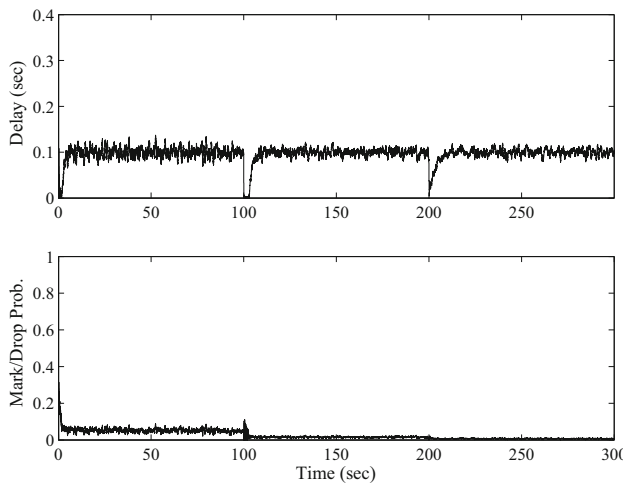


Fig. 13 Evaluation of ST-CPID to regulate the queuing delay under varying link capacity with 15, 30, and 45 Mbps at time interval 0–100, 100–200, and 200–300 s, respectively

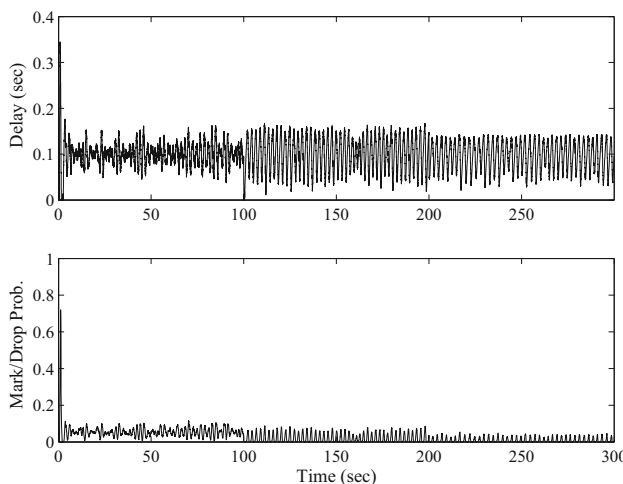


Fig. 14 Evaluation of PIE to regulate the queuing delay under varying link capacity with 15, 30, and 45 Mbps at time interval 0–100, 100–200, and 200–300 s, respectively

Table 3 Performance measures under varying link capacity

	avg. delay (s)	std. delay (s)
ST-CPID	0.096	0.018
PIE	0.095	0.036

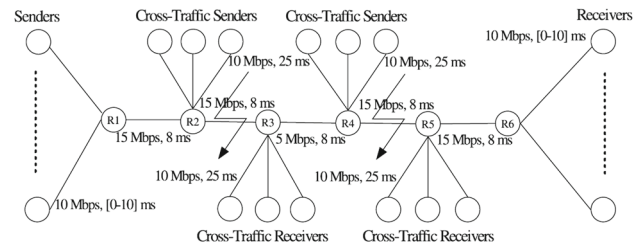


Fig. 15 Multiple bottleneck network topology [50]

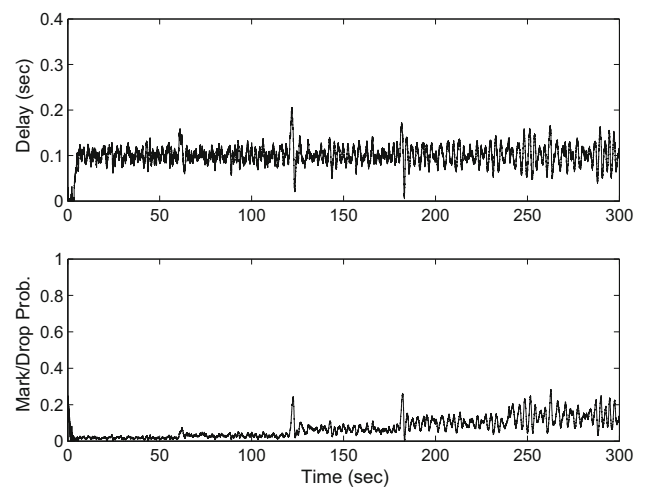


Fig. 16 Evaluation of ST-CPID to regulate the queuing delay in a multiple bottleneck link network with 50, 100, 200, 300, and 400 FTP flows at time interval 0–60, 60–120, 120–180, 180–240, and 240–300 s, respectively

flows from left to right, and 30 FTP cross-flows. It consists of two bottleneck links: link $R_2 \leftrightarrow R_3$ and $R_4 \leftrightarrow R_5$. Since the other links are not bottleneck, their queuing delay are almost close to zero. Link $R_2 \leftrightarrow R_3$ and link $R_4 \leftrightarrow R_5$ have similar behavior. Therefore, we just investigate the behavior of link $R_2 \leftrightarrow R_3$. The queuing delay of $R_2 \leftrightarrow R_3$ for both PIE and ST-CPID is shown in Figs. 16 and 17. The figures show that the ST-CPID and PIE are scalable for multiple bottleneck network topology and compared to the PIE, ST-CPID has more stable behavior.

7.4 Mixed TCP sources

We extend the evaluation of the proposed controller with a mix of different TCP sources (Reno, SACK, and Vegas). Since different types of TCP have approximately similar

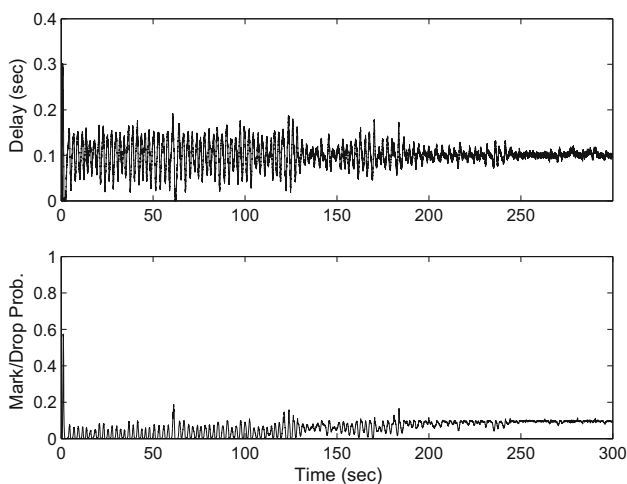


Fig. 17 Evaluation of PIE to regulate the queuing delay in a multiple bottleneck link network with 50, 100, 200, 300, and 400 FTP flows at time interval 0–60, 60–120, 120–180, 180–240, and 240–300 s, respectively

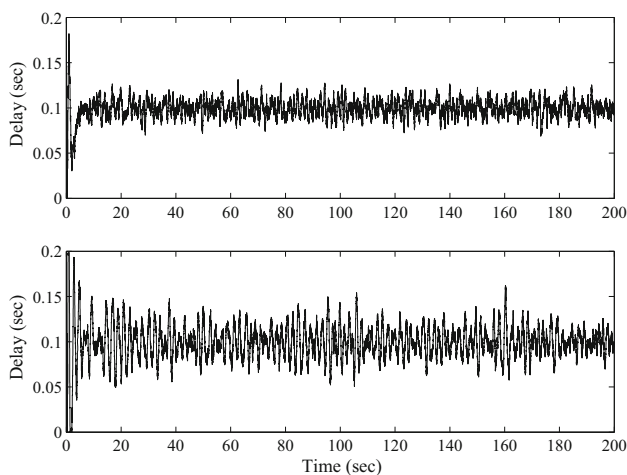


Fig. 18 Evolution of queuing delay with respect to time for ST-CPID (top) and PIE (bottom) with mix of different TCP sources

Table 4 Performance measures for a network with mixed TCP traffic

	avg(s)	std(s)	RMSE	Loss%	Util%
STCPID	0.099	0.012	0.12	0.013	0.998
PIE	0.099	0.020	0.20	0.015	0.997

dynamic behavior in the steady state, there are little changes compared to the simple traffic. The queuing delay for both PIE and ST-CPID is shown in Fig. 18. As it can be seen, ST-CPID behaves more stable than PIE. Table 4 also shows the outperformance of ST-CPID.

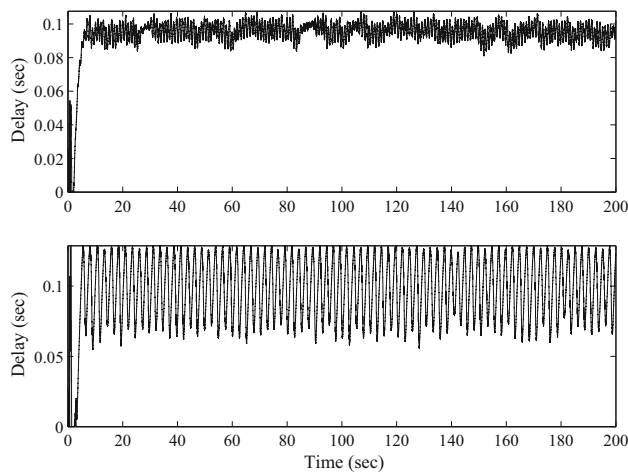


Fig. 19 Evolution of queuing delay with respect to time for ST-CPID (top) and PIE (bottom) with OC-12 link

Table 5 Performance measures for a network with OC-12 link

	avg(s)	std(s)	RMSE	Loss%	Util%
STCPID	0.0940	0.011	0.12	0	0.997
PIE	0.0936	0.023	0.24	0.001	0.992

7.5 OC-12 Bottleneck link

In this section, we evaluate the proposed controller under a high-speed bottleneck link, Synchronous Optical Networking (SONET) fiber optic networks, OC-12. OC-12 lines are commonly used by ISPs as Wide area network (WAN) connections, as the backbones of many regional ISPs, web hosting companies or smaller ISPs buying service from larger ones. Therefore, the traffic load on this link is high. The proposed controller is evaluated and compared with the PIE on dumbbell topology. The bottleneck link is OC-12 with capacity of 622 Mbps and the traffic load is 1000 of FTP flows. Figure 19 gives the time evolution of queuing delay of the ST-CPID and PIE. Regarding the PIE, the ST-CPID shows more stability in the tracking error in terms of mean and standard deviation (see Table 5). The link utilization of STCPID is approximately 0.5% higher than that of PIE which equals about 3.1 Mbps. The standard deviation of queuing delay for PIE is 0.023sec, almost twice that of STCPID 0.011 which for OC-12 link become about.

In high speed link, the queue dynamics $(NW(t)/R(t) - C)$ is more affected by the dynamics of congestion window length which is controlled through the marking/dropping feedback of AQM. However, by increasing the number of TCP flows (N), which is reasonable in WAN connections, the queue dynamics become more stable and is under the influence of cumulative behavior of many TCP flows. The PIE aims to keep queuing delay to a target (τ_0) by updating

the drop probability p which consists of a proportional and an integral part, weighted respectively by the gain factors β and α :

$$p(t) = p(t - T) + \alpha(\tau(t) - \tau_0) + \beta(\tau(t) - \tau(t - T)) \quad (18)$$

where $\tau(t)$ is the queuing delay. The proportional gain β is the main correction of p and the integral gain α is typically smaller than the proportional one. In high speed links, the growing speed of the queue ($\tau(t) - \tau(t - T)$) is high (the proportional term). Therefore p is more affected by the proportional terms and leads to the sluggish behavior of the queue length or queuing delay in the PIE.

8 Conclusion

Regulating accurately the queuing delay yet keeping high the link utilization and keeping low the packet loss rate are the main objectives of a communication network which are achieved through AQM controller. However, highly varying dynamics of the network, does not allow most of the AQM controllers to meet these objectives well. Moreover, frequently re-tuning of an AQM controller with respect to the small variations of the network parameters, leads to the sluggish behavior of the controller. In this paper a self-tuning compensated PID controller is proposed which considers the time-varying nature of delay and network parameters. Moreover, the parameter-varying dynamic compensator increases the robustness against noise and uncertainties caused from unresponsive connections. Due to harsh variations of the network parameters, the proposed change detector provides an accurate yet fast tracking filter for the estimated parameters. Packet-level evaluations using ns2 show that the proposed controller is able to tune itself accurately and quickly with respect to the harsh variations of the network parameters. Since the controller parameters are obtained directly from the state-space model coefficients, identification of the network dynamics with a linear state-space model provides directly the controller gains and it is not necessary to estimate each network parameter individually. This is regulated as the future work of this research.

References

1. Abbas, G., Manzoor, S., & Hussain, M. (2018). A stateless fairness-driven active queue management scheme for efficient and fair bandwidth allocation in congested internet routers. *Telecommunication Systems*, 67(1), 3–20.
2. Awaya, J., Ouellette, M., & Montuno, D. (2001). An optimization-oriented view of random early detection. *Computer Communications*, 24(12), 1170–1187.
3. Bhatnagar, S., Patel, S., & Karmeshu, (2018). A stochastic approximation approach to active queue management. *Telecommunication Systems*, 68(1), 89–104.
4. Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., & Zhang, L. (1998). Recommendations on queue management and congestion avoidance in the internet. In *RFC 2309*.
5. Cao, J., Cleveland, W., Gao, Y., Jeffay, K., Smith, F., & Weigle, M. (2004). Stochastic models for generating synthetic HTTP source traffic. *IEEE INFOCOM*, 3, 1546–1557.
6. Chen, Q., & Yang, O. (2004). On designing self-tuning controllers for AQM routers supporting TCP flows based on pole placement. *IEEE Journal on Selected Areas in Communications*, 22(10), 1965–1974.
7. Christiansen, M., Jeffay, K., Ott, D., & Smith, F. (2001). Tuning RED for web traffic. *IEEE/ACM Transactions on Networking*, 9(3), 249–264.
8. Chrost, L., & Chydzinski, A. (2016). On the deterministic approach to active queue management. *Telecommunication Systems*, 63(1), 27–44.
9. De Schepper, K., Bondarenko, O., Tsang, I. J., & Briscoe, B. (2016). PI2: A Linearized AQM for both classic and scalable TCP. In *Proceedings of the 12th international conference on emerging networking experiments and technologies* (pp. 105–119.) ACM.
10. Dukkupati, N. (2008). Rate control protocol (RCP): Congestion control to make flows complete quickly. PhD thesis, Stanford, CA, USA.
11. Esaki, H. (2010). A consideration on R&D direction for future internet architecture. *International Journal of Communication Systems*, 23(6–7), 694–707.
12. Feng, W. C., Kandlur, D., Saha, D., & Shin, K. (1999). A self-configuring RED gateway. In *Proceedings of eighteenth annual joint conference of the IEEE computer and communications societies (INFOCOM99)* (Vol. 3, pp. 1320–1328).
13. Floyd, S., & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4), 397–413.
14. Floyd, S., Gummadi, R., Shenker, S., et al. (2001). Adaptive RED: An algorithm for increasing the robustness of REDs active queue management. Preprint, available at <http://www.icir.org/floyd/papers.html>.
15. Guan, X., Yang, B., Zhao, B., Feng, G., & Chen, C. (2007). Adaptive fuzzy sliding mode active queue management algorithms. *Telecommunication Systems*, 35(1), 21–42.
16. Gustafsson, F., & Gustafsson, F. (2000). *Adaptive filtering and change detection* (Vol. 1). Londres: Wiley.
17. He, L., & Zhou, H. (2017). Robust Lyapunov–Krasovskii based design for explicit control protocol against heterogeneous delays. *Telecommunication Systems*, 66(3), 377–392.
18. He, X., Papadopoulos, C., Heidemann, J., Mitra, U., & Riaz, U. (2009). Remote detection of bottleneck links using spectral and statistical methods. *Computer Networks*, 53(3), 279–298.
19. Hollot, C., Misra, V., Towsley, D., & Gong, W. (2002). Analysis and design of controllers for AQM routers supporting TCP flows. *IEEE Transactions on Automatic Control*, 47(6), 945–959.
20. Hong, Y., & Yang, O. W. W. (2006). Adaptive AQM controllers for IP routers with a heuristic monitor on TCP flows. *International Journal of Communication Systems*, 19(1), 17–38.
21. Isidori, A. (1997). *Nonlinear Control Systems*. New York: Springer.
22. Jacobsson, K., Hjalmarsson, H., Möller, N., & Johansson, K. H. (2004). Estimation of RTT and bandwidth for congestion control applications in communication networks. In *IEEE CDC*. Paradise Island, Bahamas: IEEE.

23. Jarvinen, I., & Kojo, M. (2014). Evaluating CoDel, PIE, and HRED AQM techniques with load transients. In *IEEE 39th conference on local computer networks (LCN)* (pp. 159–167).
24. Kahe, G., & Jahangir, A. (2013). On the Gaussian characteristics of aggregated short-lived flows on high-bandwidth links. In *27th International conference on advanced information networking and applications workshops (WAINA)* (pp. 860–865).
25. Kahe, G., Jahangir, A. H., & Ebrahimi, B. (2014). AQM controller design for TCP networks based on a new control strategy. *Telecommunication Systems*, 57(4), 295–311.
26. Karmeshu, P. S., & Bhatnagar, S. (2017). Adaptive mean queue size and its rate of change: Queue management with random dropping. *Telecommunication Systems*, 65(2), 281–295.
27. Katabi, D., Handley, M., & Rohrs, C. (2002). Congestion control for high bandwidth-delay product networks. In *Proceedings of the 2002 conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM '02)*, pp. 89–102). New York, NY: ACM.
28. Khademi, N., Ros, D., & Welzl, M. (2014). The new AQM kids on the block: An experimental evaluation of CoDel and PIE. In *IEEE conference on computer communications workshops (INFOCOM WKSHPS)* (pp. 85–90).
29. Kim, W. J., & Lee, B. G. (1998). FRED: Fair random early detection algorithm for TCP over ATM networks. *Electronics Letters*, 34(2), 152–154.
30. Ko, E., An, D., Yeom, I., & Yoon, H. (2012). Congestion control for sudden bandwidth changes in TCP. *International Journal of Communication Systems*, 25(12), 1550–1567.
31. Li, F., Sun, J., Zukerman, M., Liu, Z., Xu, Q., Chan, S., et al. (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, 274–299.
32. Manfredi, S., di Bernardo, M., & Garofalo, F. (2009). Design, validation and experimental testing of a robust AQM control. *Control Engineering Practice*, 17(3), 394–407.
33. Marami, B., & Haeri, M. (2010). Implementation of MPC as an AQM controller. *Computer Communications*, 33(2), 227–239.
34. Melchor-Aguilar, D., & Niculescu, S. I. (2009). Computing non-fragile PI controllers for delay models of TCP/AQM networks. *International Journal of Control*, 82(12), 2249–2259.
35. Misra, V., Gong, W. B., & Towsley, D. (2000). Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. *ACM SIGCOMM Computer Communication Review*, 30, 151–160.
36. Na, Z., Guo, Q., Gao, Z., Zhen, J., & Wang, C. (2012). A novel adaptive traffic prediction AQM algorithm. *Telecommunication Systems*, 49(1), 149–160.
37. Nichols, K., & Jacobson, V. (2012). Controlling queue delay. *Queue*, 10(5), 20:20–20:34.
38. Ott, T. J., Lakshman, T. V., & Wong, L. H. (1999). SRED: Stabilized RED. In *Eighteenth annual joint conference of the IEEE computer and communications societies (INFOCOM 1999)* (Vol. 3, pp. 1346–1355).
39. Pan, R., Prabhakar, B., & Psounis, K. (2000). CHOKe—A stateless active queue management scheme for approximating fair bandwidth allocation. In *Proceedings IEEE INFOCOM 2000* (Vol. 2, pp. 942–951).
40. Pan, R., Natarajan, P., Piglione, C., Prabhu, M., Subramanian, V., Baker, F., & VerSteeg, B. (2013). PIE: A lightweight control scheme to address the bufferbloat problem. In *IEEE 14th international conference on high performance switching and routing (HPSR)* (pp. 148–155).
41. Pan, R., Natarajan, P., Baker, F., & White, G. (2017). Proportional integral controller enhanced (PIE): A lightweight control scheme to address the bufferbloat problem. In *Internet requests for comments (No RFC 8033)*.
42. Quet, P. F., & Ozbay, H. (2004). On the design of AQM supporting TCP flows using robust control theory. *IEEE Transactions on Automatic Control*, 49(6), 1031–1036.
43. Reddy, C. P., & Venkata Krishna, P. (2015). Ant-inspired level-based congestion control for wireless mesh networks. *International Journal of Communication Systems*, 28(8), 1493–1507.
44. Ryu, S. (2004). PAQM: an adaptive and proactive queue management for end-to-end TCP congestion control. *International Journal of Communication Systems*, 17(8), 811–832.
45. Schwardmann, J., Wagner, D., & Kühlewind, M. (2014). Evaluation of ARED, CoDel and PIE. In *Proceedings of the 20th EUNICE/IFIP WG 6.2, 6.6 workshop on advances in communication networking* (pp. 185–191). Cham, Rennes, France: Springer International Publishing. https://doi.org/10.1007/978-3-319-13488-8_17.
46. Sun, J., & Zukerman, M. (2007). An adaptive neuron AQM for a stable Internet. *International conference on research in networking* (pp. 844–854). Atlanta: Springer.
47. Sun, J., Chan, S., Ko, K. T., Chen, G., & Zukerman, M. (2006). Neuron PID: A robust AQM scheme. In *Proceedings of ATNAC*, Melbourne, Australia (pp. 259–262).
48. Sun, J., Chan, S., & Zukerman, M. (2012). IAPI: An intelligent adaptive PI active queue management scheme. *Computer Communications*, 35(18), 2281–2293.
49. Üna, H., Melchor-Aguilar, D., Üstebay, D., Niculescu, S. I., & Özbay, H. (2013). Comparison of PI controllers designed for the delay model of TCP/AQM networks. *Computer Communications*, 36(10), 1225–1234.
50. Wang, P., Chen, H., Yang, X., & Ma, Y. (2012). Design and analysis of a model predictive controller for active queue management. *ISA Transactions*, 51(1), 120–131.
51. Wang, P., Zhu, D., & Lu, X. (2017). Active queue management algorithm based on data-driven predictive control. *Telecommunication Systems*, 64(1), 103–111.
52. Wu, W., Ren, Y., & Shan, X. (2001). A self-configuring PI controller for active queue management. In *Asia-Pacific conference on communications (APCC)*.
53. Yan, Q., & Lei, Q. (2011). A new active queue management algorithm based on self-adaptive fuzzy neural-network PID controller. In *International conference on internet technology and applications (iTAP)*, Wuhan, China (pp. 1–4).
54. Zhang, C., Khanna, M., & Tsaoussidis, V. (2004). Experimental assessment of RED in wired/wireless networks. *International Journal of Communication Systems*, 17(4), 287–302.
55. Zhang, H., Towsley, D., Hollot, C. V., & Misra, V. (2003). A self-tuning structure for adaptation in TCP/AQM networks. In *Proceedings of the ACM international conference on measurement and modeling of computer systems (SIGMETRICS)* (SIGMETRICS '03, pp. 302–303). New York, NY: ACM
56. Zheng, B., & Atiquzzaman, M. (2008). A framework to determine the optimal weight parameter of RED in next-generation internet routers. *International Journal of Communication Systems*, 21(9), 987–1008.
57. Zheng, F., & Nelson, J. (2009). An h_∞ approach to the controller design of AQM routers supporting TCP flows. *Automatica*, 45(3), 757–763.
58. Zhou, C., Di, D., Chen, Q., & Guo, J. (2009). An adaptive AQM algorithm based on neuron reinforcement learning. In *IEEE international conference on control and automation*, Christchurch, New Zealand (pp. 1342–1346).



Ghasem Kahe is assistant professor in the Aerospace Research Institute, Ministry of Science, Research, and Technology. He received his Ph.D. in Computer Engineering from Sharif University of Technology, Tehran, Iran, in 2013. His main research interests include Computer Networks, Congestion Control, and Quality of Services.



Amir Hossein Jahangir obtained his Ph.D. in industrial informatics from Institut National des Sciences Appliquées, Toulouse, France in 1989. Since then he has been with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran, and has served during his career as the Head of the Department, Head of Computing Center, and is now an associate professor and the director of Network Evaluation and Test Laboratory, an accredited internationally recognized laboratory in the field of network equipment test. His fields of interest comprise network equipment test and evaluation methodology, network security, and high performance computer architecture.