Hindawi Publishing Corporation
Scientific Programming
Volume 2016, Article ID 5796907, 10 pages
http://dx.doi.org/10.1155/2016/5796907

Research Article

Infinite Queue Management via Cascade Control for Industrial Routers in Smart Grid IP Networks

Ku-Hwan Kim,[1] Hoang-Linh To,[2] Won-Joo Hwang,[3] and Jung-Tae Lee[1]

[1]*Department of Electrical and Computer Engineering, Pusan National University, Pusan 46241, Republic of Korea*
[2]*Department of Information and Communication System, Inje University, Gyeongnam 50834, Republic of Korea*
[3]*Department of Information and Communications Engineering, HSV-TRC, Inje University, Gyeongnam 50834, Republic of Korea*

Correspondence should be addressed to Won-Joo Hwang; ichwang@inje.ac.kr

Received 28 April 2016; Accepted 17 August 2016

Academic Editor: Kun Hua

Copyright © 2016 Ku-Hwan Kim et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Smart grid applications experience an extremely wide range of communication delay. Data flows of those applications are normally aggregated at industrial network routers in substations, form infinite (long) queues termed bufferbloat issue, and might damage the operation of transmission control protocol. The default queue management scheme, DropTail, in such routers just drops packets if queue is full while the others in literature are mostly based on one-loop feedback control where an optimal point of performance between queue length and drop rate is limited. In this paper, we study the problem of managing a long queue of industrial router at substation under heterogeneous smart grid networks. Specifically, we propose an enqueue-dequeue dropping cascade control using a two-loop design method to control both window size and queue length. Moreover, our proposal can be easily implemented into router firmware with provided discrete expressions. Finally, our simulation results are presented to validate the possible benefits that can be gained from cascade control and compare the existing queue management methods as well.

1. Introduction

1.1. Overview. Smart grid is going to be a geographically widespread system soon, which integrated many diverse real-time applications (e.g., sensor) together with energy management applications. Smart grid, which is a digitally-enhanced version of the traditional electric grid, provides infrastructure to support diverse services such as finance, information, and electrical delivery among consumers, assets, and those users who have authorized access. A conceptual model for smart grid communication is presented in [1]. At the customer house, energy consumption information is monitored and then sent to service provider for further analysis and better support. At the operator side, data is collected to make decision according to demand response or energy saving.

With an emerging smart grid, various applications can be enabled to improve quality of service and consumer satisfaction. Some popular examples are building automation, automated meter reading, outage and restoration management, and electric vehicles. Most of the smart grid applications have a strict latency requirement in the range of 100 milliseconds to 5 seconds [2]. In [3], the delay ranges of different smart grid applications were approximately reported. For example, teleprotection applications allow traffic delay from 8 to 10 ms while they allow traffic delay for more than one second for interval measurement from smart meters. To respect it, we also need a fast communication infrastructure that can handle a huge amount of exchanging data and is able to provide a near real-time response. Latency is defined as the time interval between when the state occurred and when it was acted upon by a smart grid application [2]. Various applications own different latency requirements that depend on the kind of system response it is dealing with. In such scenario, network equipment including switches, Internet wired/wireless routers, and operating systems (O/S) are the fundamental components to provide a fast and reliable smart grid communication. Table 1 lists latency and bandwidth requirements for some specific applications in smart grid networks [4].

In such a heterogeneous environment with multiple types of applications, special equipment like industrial routers

Table 1: A wide range of delays for some smart grid applications.

Application	Latency	Bandwidth
Metering	0–15 min	10–100 kbps
Information exchange	5–30 s	14–100 kbps
Electric transportation	2 s–5 min	100 kbps

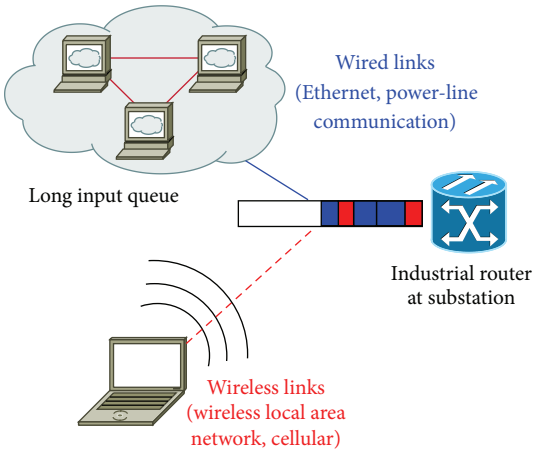


Figure 1: Long input queue at industrial router at substation.

at substations are necessary to handle. Commercially, we have seen an example product from the company, Virtual Access, that is, GW2027, which is an industrial router with applications like machine-to-machine (M2M), telemetry, supervisory control and data acquisition (SCADA), roadside, and wireless. Figure 1 illustrates such smart grid network situation in detail, where industrial router has to process data from both wired and wireless links. Among several great software features like remote configuration and fault reporting and so forth we, however, realize that a scheme to manage the input queue of the industrial router is not specified in the data sheet [5] and is usually Drop Tail, a simple scheme which drops packets in case of full buffer and allows packets otherwise.

1.2. Motivation. In general, each part of every network equipment or substation industrial router is usually preinstalled or configured with some amount of buffering, whether it is enough or not, to handle bursts of arriving packets and departing packets to the next link. It is important to ensure good utilization of the network link, especially in cases where arriving rate is greater than departing rate causing bottleneck point to be built up. The buffer then absorbs high-rate traffic packets which wait and are later served on the slower outgoing links [7]. Not enough buffering results in high dropping rate of most of the packets and low egress link utilization. For example, if a user transfers files using transmission control protocol (TCP), the user satisfaction is measured by how quickly the file transfer can complete, which is directly related to how effectively the protocol can utilize network links, that is, more buffering is better. Moreover, as the cost of buffers/memory keeps reducing, it is foreseen that large buffering would be put into every piece of network equipment and even preconfigured in operating systems. However, large buffering results in long latency experience for users as well. Besides, most of TCP schemes (Reno, New Reno, CUBIC, etc.) use loss-based congestion control. It means that a loss-based TCP sender continues increasing its congestion window size (i.e., sending rate) until a loss signal is detected by time-out or incorrect acknowledgment packets. In [8], it is suggested that "the widely used rule of thumb leads to buffers which are much larger than they need to be" and could be reduced by scarifying a small amount of bandwidth utilization. Using too much buffering, it obviously takes longer time for loss detection and therefore TCP may function incorrectly. The excessive long latency and the damage of TCP due to too much buffering are two major consequences of a problem recently termed as *bufferbloat*.

The up-to-date solutions to the bufferbloat issue, or debloat, consist of traffic shaping, TCP window size modification, and active queue management (AQM). In the scope of this paper, we concentrate on AQM algorithms because they can manage queuing delay efficiently, by either a direct or an indirect way. Most of Linux kernels and some open source routers (CeroWRT) have been integrated with some modern AQM algorithms [9, 10]. For example, depending on where the bottlenecks are, a Linux user can change the computer to use a different AQM qdisc (queuing discipline) via a `sysctl` or the `tc` (traffic control) command. Literature has witnessed several coexisting AQM candidate algorithms including Controlled Delay (CoDel) [11], Stochastic Flow Queuing CoDel (sfqCoDel) [12], and Proportional Integral Enhanced (PIE) [13] that have been proposed using varied theory tools (e.g., optimization, queuing, and control theory). Among them, the control theory approach owns some advantages to be implemented in smart grid network equipments, for example, parameterization controllers and well reference tracking. Theoretically, control-based AQM algorithms exploit the additive increase multiplicative decrease (AIMD) model of TCP and the continuous fluid-flow queue approximation [14, 15]. Then the specific controller's parameters are designed according to a closed-loop transfer function of the whole system. We, however, realize that these models almost design one-loop control for queue length only which has some problems of large overshoots and unacceptable lags (delay) in mixed-traffic scenarios. More loops operating in different network layers should improve the whole system performance, especially the input delay transient (overshoot) behaviour of the aforementioned debloat schemes.

1.3. Our Contributions. In this paper, we develop a novel active queue management using cascade control framework in control theory that is able to reduce large overshoot and therefore improve delay transient behaviour. To the best of our knowledge, our work is the first attempt to adapt cascade control method to bufferbloat research field. Using the well-known AIMD and fluid queue model [14], we decompose transfer functions into an inner and an outer loop. The inner one adjusts window size based on changing of traffic and

feedback window size and the outer one mission adjusts queue length based on feedback queue length value at time instant. Each loop's parameters are tuned using optimal gain and phase margin method [16]. This scheme operates at both the network transport layer (adjusting TCP congestion window size) and the network link layer (adjusting queue length). A difficulty when considering this cascade design is the interaction in time scale between two loops. The other side effect is complexity due to two additional controllers. Nevertheless, we show that better delay performance results can be achieved with EDDC via both numerical and simulation results (Section 5.2). Our primary contributions are summarized as follows:

(i) We develop a cascade control active queue management (EDDC) scheme by decomposing the TCP/AQM fluid-flow model into two cascaded loops for two independent controllers.

(ii) By applying the reliable optimal gain and phase margin tuning method, we derive a set of parameters for these controllers based on a given example from single-loop AQM design.

(iii) Our simulations are conducted in network simulator ns-2 using different smart grid application models, including file data transmission (FTP), voice over IP (VoIP), and video conference, which are close to the realistic smart grid applications. In most of cases, the related-delay results using EDDC are well controlled around the target delay (10 ms) while the large overshoot phenomenon disappears.

2. Related Works

In this section, we discuss the features of the famous existing AQM schemes: one mature (RED) and two modern AQM algorithms (CoDel, sfqCoDel) which are designed specifically for bufferbloat mitigation.

Random Early Detection (RED) is one of the first viable AQM algorithms [17]. However, it is proved to be so difficult to configure properly that hardly anybody uses it, even though many carrier grade routers implement it. With an appropriate set of parameters, RED is an effective algorithm. However, dynamically predicting this set of parameters was found to be difficult. As a result, RED has not been enabled by default, and its present use in the Internet is limited [18]. Other AQM algorithms have been developed since RFC2309 was published, some of which are self-tuning within a range of applicability. Hence, while this memo continues to recommend to deploy AQM, it no longer recommends that RED or any other specific algorithm is used as a default; instead it provides recommendation on how to select appropriate algorithms and that a recommended algorithm is able to automate any required tuning for common deployment scenarios.

"Controlled Delay" (CoDel) AQM algorithm [11] tries to address the problems that RED could not. First, the input signal into the algorithm (sojourn time versus average queue length) is of a different quality; second, CoDel (in its plain form) does drop/mark on dequeue (departure queue), while RED drops/marks on enqueue (arrival queue). Therefore TCP congestion control loop using CoDel responds much quicker than using RED; thus the reaction by the sender will probably be timely and relevant for that congestion epoch. With RED, the congestion signal (lost packet) has to traverse the filled-up buffer first; thus, the control loop time is much larger (includes the instantaneous queue length of the buffer) and is further delayed by the averaging going on. By design, CoDel does not need to be tuned specifically for one particular drain rate (bandwidth) of the queue unlike RED; so it adjusts much better to variable bandwidth MACs (e.g., Wi-Fi and DOCSIS link).

CoDel itself, however, drops packet of a group during each interval without considering the packet's priority level or user side. Recently, Stochastic Flow Queue CoDel (sfqCoDel) [12] is a promising AQM algorithm which demonstrates a satisfied performance to mitigate bufferbloat for users, for example, increase 10x speed of a network under load [9], and it has been implemented in open source routers (i.e., OpenWRT [10]). The sfqCoDel is a hybrid of deficit round-robin scheduling (DRR) and CoDel. It is renamed from "Fair Queuing" to "Flow Queuing" because flows that build a queue are treated differently than flows that do not. It stochastically classifies incoming packets into different queues; each queue is managed by the CoDel AQM algorithm. Packet ordering within a queue is preserved since queues have FIFO ordering [19].

3. System Model

3.1. TCP State-Space Model. We exploit the TCP fluid-flow model described by nonlinear differential equations that has been extensively studied in network routers interacting with TCP sources (e.g., [14, 20, 21]). This model captures the additive increase multiplicative decrease (AIMD) feature from TCP [22], without slow start and time-out mechanisms. However, this lack only affects initial start-up of the TCP system. Once the system reaches the stable point, the differential equations solver is able to track changes in the network well [23].

$$\begin{aligned}\dot{w}(t) &= f(w, w_R, q, p)\\ &= \frac{1}{R(t)} - \frac{w(t)}{2}\frac{w(t-R(t))}{R(t-R(t))}p(t-R(t));\end{aligned} \tag{1}$$

$$\dot{q}(t) = g(w, q) = N\frac{w(t)}{R(t)} - C_l, \tag{2}$$

where w is average TCP window size (packets); q is queue length at cable modems (packets); $p(\cdot)$ is packet dropping probability function ($0 \le p \le 1$); C_l is transmission capacity of link l (packets/sec); R is round-trip time (sec); $R(t) = T_p + q(t)/C_l$ with T_p being propagation delay; and N is number of TCP sessions.

The first equation (1) describes the TCP-Reno-sender behaviour based on AIMD while the second one (2) models a fluid queue, which allows traffic arrivals to be continuous rather than discrete, as in a classic queuing model like M/M/1

queue. The operating point (w_0, q_0, p_0) of model (1) can be derived at with $\dot{w} = 0$ and $\dot{q} = 0$ as follows

$$\begin{aligned} w_0^2 p_0 &= 2, \\ w_0 &= \frac{R_0 C_l}{N}, \\ R_0 &= T_p + \frac{q_0}{C_l}. \end{aligned} \tag{3}$$

By doing linearization around operating points (w_0, q_0, p_0) [14], with $\delta w \doteq w - w_0$; $\delta q \doteq q - q_0$; and $\delta p \doteq p - p_0$, a linearized small signal model is obtained as:

$$\begin{aligned} \delta \dot{w}(t) &= \frac{\partial f}{\partial w}\delta w + \frac{\partial f}{\partial w_R}\delta w_R + \frac{\partial f}{\partial q}\delta q + \frac{\partial f}{\partial p}\delta p, \\ \delta \dot{q}(t) &= \frac{\partial g}{\partial w}\delta w + \frac{\partial g}{\partial q}\delta q. \end{aligned} \tag{4}$$

By defining system state vectors as $x(t) = \left[\delta w(t) \;\; \delta q(t)\right]^T$; control input vector as $u(t) = \delta p(t)$; external disturbance as $\omega(t)$; and system output vector as $y(t) = \delta q(t)$, we then rewrite the TCP system as a continuous time-invariant type using state-space modeling approach.

$$\begin{aligned} \dot{x}(t) &= Ax(t) + A_d x(t - R_0) + Bu(t - R_0) + G\omega(t), \\ y(t) &= Cx(t), \end{aligned} \tag{5}$$

where matrices A, A_d, B, C, and G represent state (system), input, output, and disturbance matrix, respectively, as follows:

$$\begin{aligned} A &= \begin{bmatrix} \frac{-N}{R_0^2 C_l} & 0 \\ \frac{N}{R_0} & \frac{-1}{R_0} \end{bmatrix}, \\ A_d &= \begin{bmatrix} \frac{-N}{R_0^2 C_l} & 0 \\ 0 & 0 \end{bmatrix}; \\ B &= \begin{bmatrix} \frac{-R_0 C_l^2}{2N^2} \\ 0 \end{bmatrix}, \\ C &= \begin{bmatrix} 0 & 1 \end{bmatrix} \\ G &= \begin{bmatrix} 0 & 0 \end{bmatrix}. \end{aligned} \tag{6}$$

The final system model equations which map the input (traffic flow) to the output (queue length) are

$$\begin{aligned} P(s) &= P_{\text{TCPwin}}(s) \cdot P_{\text{queue}}(s) \cdot e^{-sR_0} \\ &= \left[\frac{w(s)}{p(s)}\right] \cdot \left[\frac{q(s)}{w(s)}\right] = \left[\frac{A}{s+B}\right] \cdot \left[\frac{C}{s+D}\right] e^{-sR_0}. \end{aligned} \tag{7}$$

Using the above state-space TCP/IP system model, there are several approaches to design an efficient controller with the following requirements:

(i) Queue length is as short as possible.

(ii) Sensitivity to network parameters is low.

(iii) Link utilization is high, which means we do not waste the cost of link.

(iv) Packet drop rate is low.

We, however, admit a trade-off to satisfy all of those aforementioned requirements. For example, a shorter queue length comes along with a higher packet drop rate. Our algorithm in the next section seeks for an optimal point.

3.2. Optimization Formulation of Infinite Buffer (Bufferbloat). The classic M/M/1 queuing model well represents the infinite buffer or bufferbloat phenomenon of smart grid substation routers. Optimizing the performance of even simple queue like M/M/1 is a difficult problem because of nonlinearity of objective functions and constraints and running time scales exponentially with the problem size. In [24], a convex optimization problem was proposed for M/M/1 queues and with purpose of minimizing the state probability p_k only. However, minimizing queue length is the main purpose when dealing with the bufferbloat issue. Therefore, we formulate our optimization problem to directly optimizing queue length of infinite buffer, with constraints following the aforementioned TCP state-space model.

$$\begin{aligned} \underset{w,q}{\text{minimize}} \quad & \dot{q}(t) = g(w, q) = N\frac{w(t)}{R} - C_l \\ \text{subject to} \quad & \dot{w}(t) = f(w, q, p) \\ & 0 < p(t) < 1. \end{aligned} \tag{8}$$

The optimization variables are w and q, and the constant parameters are N, C_l, and R.

4. Proposed Algorithm

In this section, we propose an algorithm, named an Enqueue-Dequeue Dropping Cascade control scheme (EDDC). We present two main design steps of our proposed algorithm from the above TCP state-space system model. The first step is to design in the continuous time domain. The second step is transform dropping probability formula into discrete domain so that we are able to implement it into the smart grid industrial router firmware (Figure 2).

4.1. EDDC in Continuous Time Domain. We design two controllers for each loop. The inner controls drop probability p_2 based on traffic information and the outer controls drop probability p_1 based on difference between measured average queue length and reference queue length.

(i) Inner Loop. An important design requirement is that the inner loop controller should behave quickly. From (7), the inner control objective is a linear first-order type:

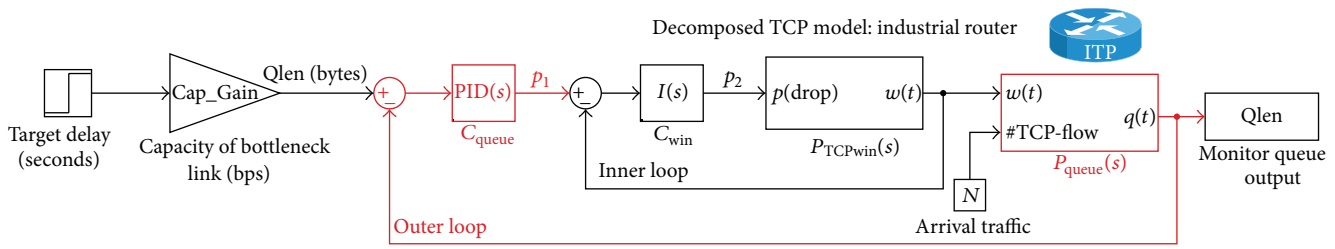


Figure 2: Proposed control framework for queue management in industrial router.

$P_{TCPwin}(s) = k_I/(1 + Ts)$, where $k_I = A/B$ and $T = 1/B$. Hence, we design an integral I controller for inner loop:

$$C_{win}(s) = \frac{1}{T_{C_{win}} s}. \tag{9}$$

The close-loop transfer function of the inner loop is

$$I(s) = \frac{w(s)}{p_1(s)} = \frac{P_{TCPwin}(s) C_{win}(s)}{1 + P_{TCPwin}(s) C_{win}(s)} = \frac{k_I}{T_{C_{win}} s(1 + Ts) + k_I}. \tag{10}$$

The $I(s)$ transfer function is converted into frequency domain, with ω as frequency:

$$|I(j\omega)| = \frac{k_I}{\sqrt{\left(k_I - T_{C_{win}} T \cdot \omega^2\right)^2 + \left(\omega \cdot T_{C_{win}}\right)^2}} \Longleftrightarrow$$
$$I(j\omega)^2 = \frac{k_I^2}{k_I^2 + \left(T_{C_{win}}^2 - 2k_I \cdot T_{C_{win}} T\right)\omega^2 + T_{C_{win}}^2 \cdot T^2 \omega^4}. \tag{11}$$

One of quality requirements of closed-loop control system, which is represented by $I(s)$, is that the output is same to the input signal or the controller C_{win} should bring $|I(j\omega)| = 1$, $\forall\omega$, which can be called optimal gain and phase margin tuning method [16]. However, due to several reasons of real system, that requirement is rarely satisfied for all frequencies ω. An acceptable design is that $|I(j\omega)| \approx 1$, in a wide band of low frequencies ω. Hence, we propose to choose $T_{C_{win}}$ such that $T_{C_{win}}^2 - 2k_I \cdot T_{C_{win}} T = 0$ or $T_{C_{win}} = 2k_I T$. This close-form expression of $T_{C_{win}}$ is used to make decision for controller C_{win}.

(ii) Outer Loop. The close-loop transfer function of the outer loop is

$$O(s) = \frac{q(s)}{q_{ref}} = \frac{I(s) P_{queue}(s) C_{queue}(s)}{1 + I(s) P_{queue}(s) C_{queue}(s)}. \tag{12}$$

The outer control objective into zero-pole form is

$$I(s) P_{queue}(s) = \frac{k_I}{T_{C_{win}} s(1 + Ts) + k_I} \cdot \frac{C}{s + D} = \frac{k_O}{\left(1 + T_{1_O} s\right)\left(1 + T_{2_O} s\right)\left(1 + T_{3_O} s\right)}, \tag{13}$$

where $k_O = C/D$; $T_{1_O} T_{2_O} = T T_{C_{win}}/k_I$; $T_{1_O} + T_{2_O} = T_{C_{win}}/k_I$; and $T_{3_O} = 1/D$.

For the outer loop, the objective function is linear third-order type, due to inclusion of $I(s)$. Hence, we choose proportional-integral-derivative (PID) controller by using the same method at the inner loop design, or $|O(j\omega)| \approx 1$.

$$C_{queue}(s) = k_{p_O}\left(1 + \frac{1}{T_{i_O} s} + T_{d_O} s\right), \tag{14}$$

with $k_{p_O} = (T_{1_O} + T_{2_O})/2k_O T_{3_O}$, $T_{i_O} = T_{1_O} + T_{2_O}$, and $T_{d_O} = (T_{1_O} \cdot T_{2_O})/(T_{1_O} + T_{2_O})$.

4.2. EDDC in Discrete Time Domain

(i) Discrete Outer Loop Controller

$$\frac{p_1(s)}{e_1(s)} = C_{queue}(s) = k_{p_O}\left(1 + \frac{1}{T_{i_O} s} + T_{d_O} s\right), \tag{15}$$

where $p_1(s)$ is the output of the outer loop controller and $e_1(s)$ is the difference between measured queue length and desired queue length at the outer loop. We apply backward difference method to get the z-transform of the outer loop controller $C_{queue}(s)$:

$$p_1(t) = e_1(t) k_{p_O}\left[1 + \frac{T_s}{T_{i_O}\left(1 - z^{-1}\right)} + T_{d_O}\frac{\left(1 - z^{-1}\right)}{T_s}\right]. \tag{16}$$

Transforming again to time domain, we obtain

$$p_1(t) = p_1(t-1) + k_{p_O}\left[e_1(t) - e_1(t-1)\right] + \frac{k_{p_O} T_s}{T_{i_O}} e_1(t) + \frac{k_{p_O} T_{d_O}}{T_s}\left[e_1(t) - 2e_1(t-1) + e_1(t-2)\right], \tag{17}$$

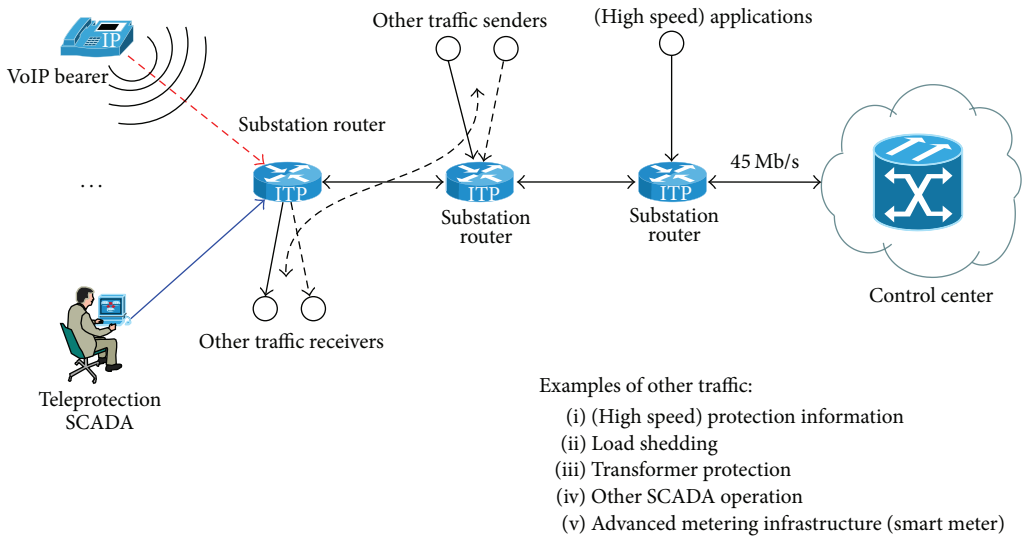


Figure 3: Cross-traffic simulation topology for smart grid applications.

or a compact form of the outer loop controller C_{queue}

$$p_1(t) = p_1(t-1) + a_O \cdot e_1(t) - b_O \cdot e_1(t-1) + c_O \cdot e_1(t-2). \quad (18)$$

(ii) Discrete Inner Loop Controller

$$\frac{p_2(s)}{e_2(s)} = C_{\text{win}} = \frac{1}{T_{C_{\text{win}}} s}, \quad (19)$$

where $p_2(s)$ is the output of the inner loop controller and $e_2(s)$ is the difference between p_1 and congestion window size $w(s)$. Doing similarly as discrete outer loop steps above, we obtain a compact form of the inner loop controller C_{win} in discrete domain as follows, with $a_I = T_s/T_{C_{\text{win}}}$:

$$p_2(t) = p_2(t-1) + a_I \cdot e_2(t-1). \quad (20)$$

5. Performance Evaluation

5.1. Simulation Setup. NS2 is a discrete event network simulator, which simulates packet-level events and facilitates the development of communication network scenarios considering the new protocols involved, either wired or wireless technologies. We use and modify the NS2 TCP evaluation tool originally developed by Wang et al. [25] for our AQM algorithm evaluation purpose. We also develop our own EDDC source code files based on the basic proportional integral (PI) algorithm code from NS2 source code tree. The simulation network topology as multiple cross bottlenecks is our choice because of its realistic smart grid IP networks (Figure 3). The simulation configuration is presented in Table 2.

5.2. Simulation Results. We evaluate our proposed algorithm performance according to three main characteristics of the bottleneck link: bandwidth (Mbps), propagation round-trip time (ms), and packet error rate ratio. Each experiment is compared between AQM algorithms (in Table 2) with respect to mean queue length, link utilization, and packet drop rate. We expect that mean queue length is as low as possible, link utilization is about 80 → 95%, and packet error rate should be low. These requirements of three criteria, however, cannot be achieved at the same time due to trade-off. Therefore, our algorithm searches for a balance point among three of them.

Table 2: Simulation configuration.

(a)

Parameter	Value
Simulation time	600 sec
FTP packet size	(Bulk transmission) 500 bytes
VoIP (G.711)	On: 1 sec; idle: 1.35 sec
Video (MyEvalVid)	Sample foreman; packet size: 1024 bytes

(b)

AQMs	Configuration
RED	Max average queue size threshold: 20 kbytes
FB-OCQ [6]	Target queue length: 25 kbytes
PI	Target queue length: 25 kbytes
CoDel	Target delay: 10 msec
sfqCoDel	Target delay: 10 msec; 32 subflows
EDDC	Target delay: 10 msec; a_O, b_O, c_O, and a_I

5.2.1. Experiment 1: Varied Link Bandwidth. In the first experiment, we vary the bottleneck link bandwidth from 1 Mbps to 1000 Mbps. This variation represents different types of links in heterogeneous smart grid networks. For example, a wired link such as Ethernet has a bandwidth of 100 Mbps while a wireless link from wireless adapter has

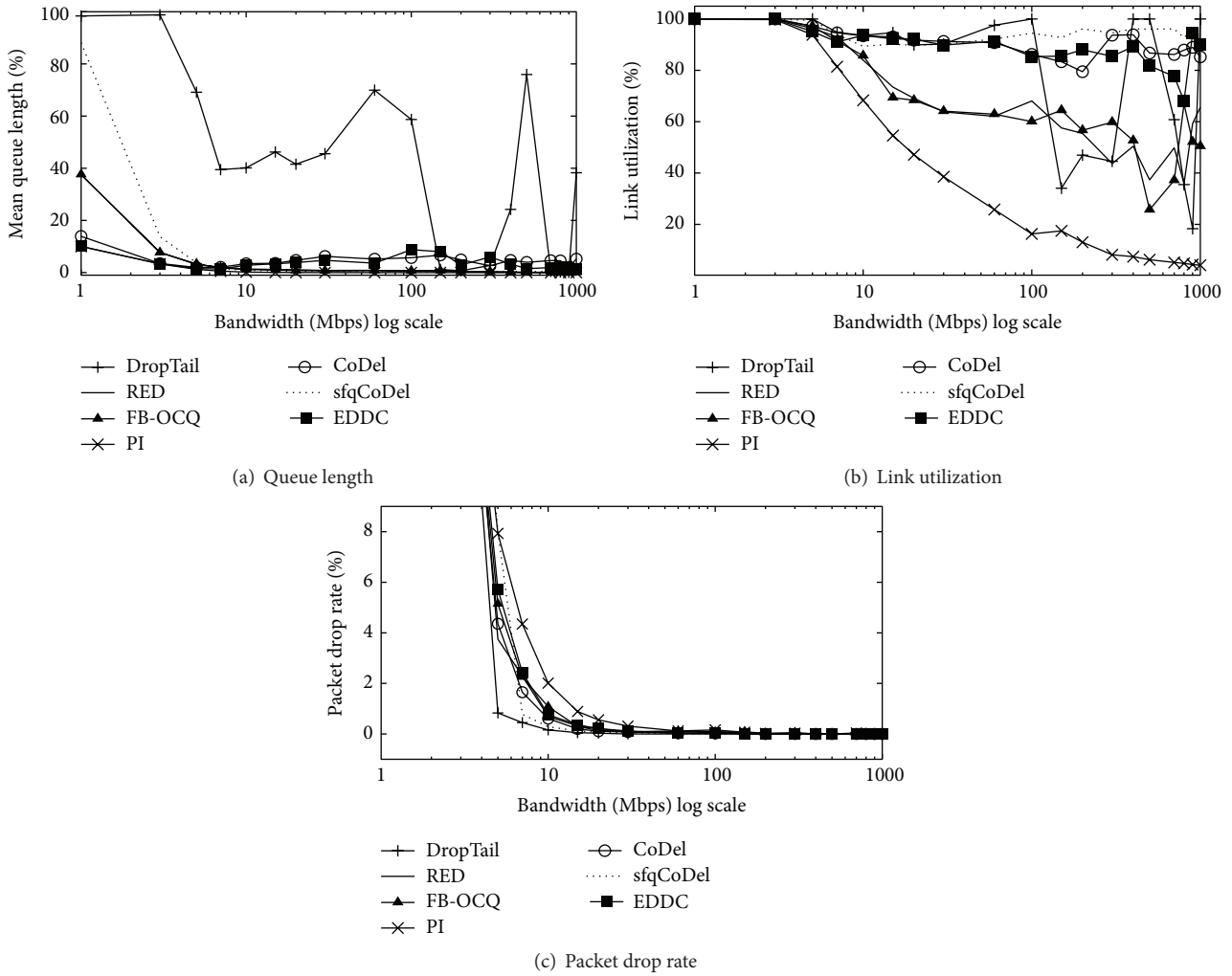


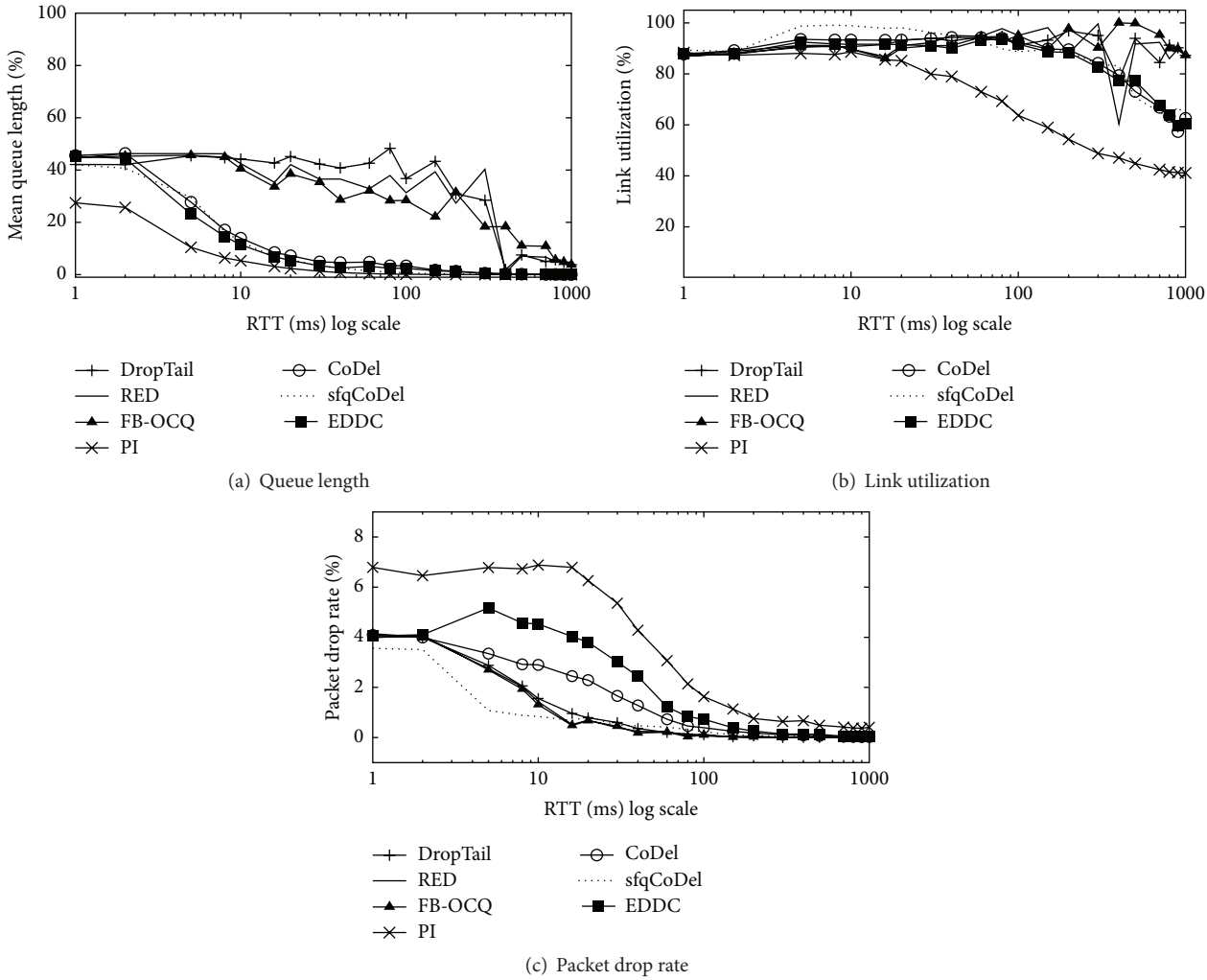
Figure 4: Experiment 1: varied link bandwidth.

varied bandwidth because of wireless signal attenuation and fading.

Firstly, Figure 4 presents our comparison results when we change the value of link bandwidth from 1 Mbps to 1000 Mbps. Our algorithm achieves very low and stable mean queue length (≈10%); even bandwidth is varied in a wide range. Meanwhile, the link utilization of EDDC is about 95%, which is higher than the default DropTail or RED and nearly approaches the modern sfqCoDel algorithm (96%). The packet drop rate performance of EDDC, however, is almost the same as the others. It makes sense that when we use a low-bandwidth link, the packet drop rate is high due to full buffer phenomenon. We conclude that our EDDC algorithm promisingly maintain the queue length of substation router, while link utilization or the busy level of link is acceptable under different kinds of links.

5.2.2. Experiment 2: Varied Link Round-Trip Time. Figure 5 shows our comparison of AQM algorithms performance in case of varied link propagation round-trip time (RTT). This criterion represents the latency of packets when using a loss-based TCP congestion control scheme, for example, TCP Reno. As mentioned before, one consequence of a large buffer (bufferbloat) is that TCP operation might be damaged because of untimely congestion information feedback of acknowledgment (ACK) packets. The higher the propagation RTT is, the higher chance of incorrect ACK feedback to the sources is. We observe the same results of EDDC performance for the mean queue length and link utilization. In Figure 5(c), the packet drop rate results of EDDC, however, are higher than the other AQM algorithms in the RTT range from 1 ms to 100 ms.

5.2.3. Experiment 3: Varied Link Packet Error Rate. Lastly, we change the packet error rate (PER) of the congested link. A wired link, for example, Ethernet cable, usually has a PER value of zero. A wireless link, however, has the wide range of PER values because of wireless signal characteristics such as



(a) Queue length

(b) Link utilization

(c) Packet drop rate

Figure 5: Experiment 2: varied link propagation round-trip time.

attenuation or fast fading. In order to simulate them, we vary PER from 0 to 0.9; for instance, a link with PER = 0.9 means that if we transmit 10 packets, only 1 packet is successfully delivered to destination.

In Figure 6(a), we see that EDDC keeps the router queue length at 2% which is lower and better than DropTail, RED, FB-OCQ, and CoDel algorithms. The modern sfqCoDel using the stochastic fair method seems to manage queue length to be too short (nearly 0%) which might increase the number of packet drop. It is, in fact, confirmed from Figure 6(c) that sfqCoDel has the highest packet drop rate (≥2%), while our algorithm EDDC only drops packets smartly at 1% ratio.

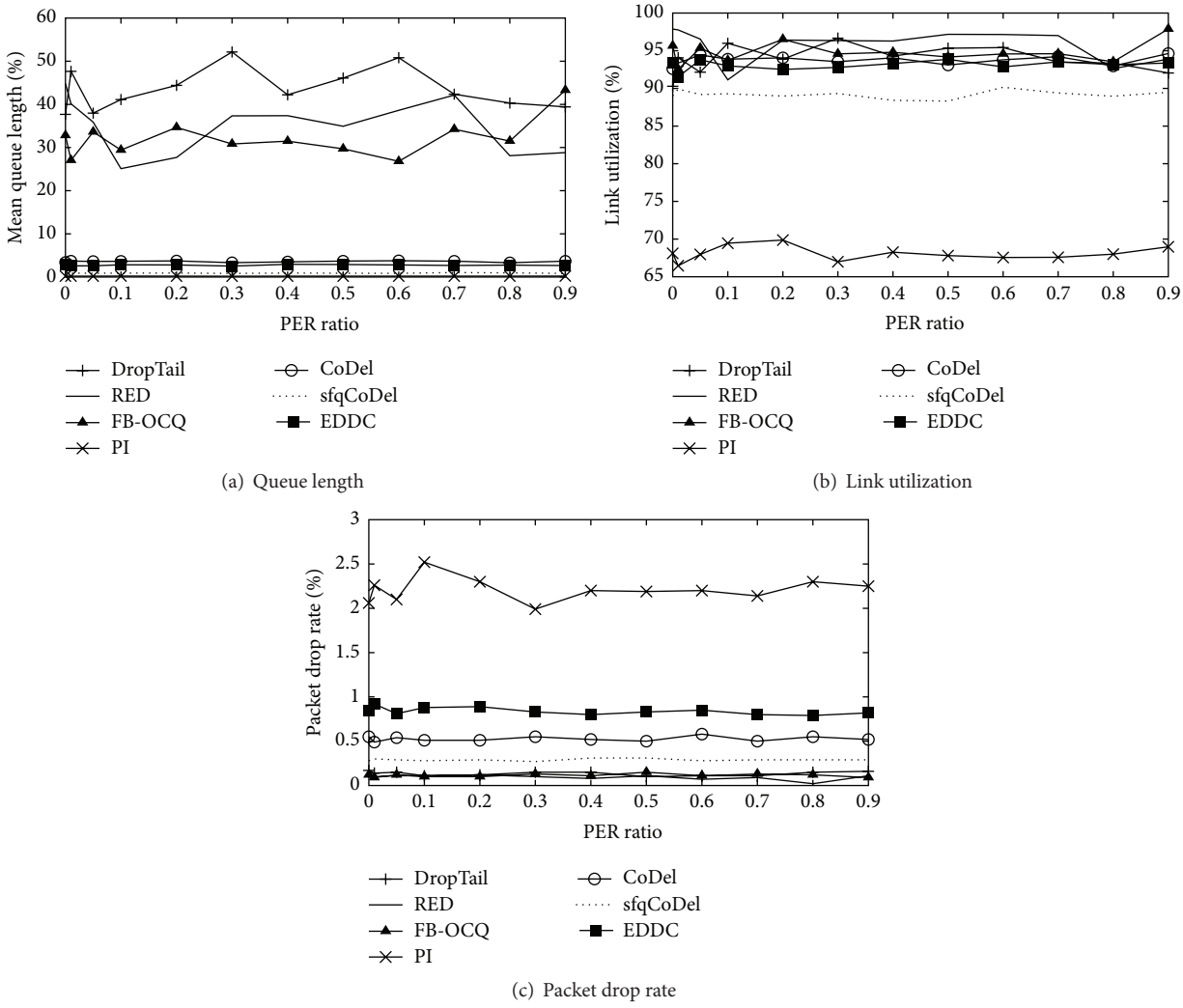
The above experiments support us to conclude some important advantages of our queue management algorithm EDDC. By smartly dropping packets with EDDC scheme, the mean queue length is kept at a stable and low level (2–5%) comapred with the other popular algorithms. Moreover, link utilization value is reasonable (80–95%) while the packet drop rate of EDDC is lower than the newest and modern algorithm (sfqCoDel).

6. Conclusions

An algorithm for better queue management of heterogeneous smart grid traffic was proposed. The algorithm, based on cascade control theory, smartly dropped packets waiting in the queue based on information of inflow smart grid traffic and current state of input queue. The algorithm was shown to effectively manage the queue length or the number of packets that are waiting in queue at a stable and low level (2–5%) while the link utilization does not reduce too much.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.



(a) Queue length

(b) Link utilization

(c) Packet drop rate

Figure 6: Experiment 3: varied link packet error rate.

Acknowledgments

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the Grand Information Technology Research Center Support Program (IITP-2016-R71181610050001002) supervised by the IITP (Institute for Information & Communications Technology Promotion). This work was also supported by a 2-year research grant of Pusan National University.

References

[1] F. Baker and D. Meyer, "Internet protocols for the smart grid," Tech. Rep., 2011.

[2] P. Kansal and A. Bose, "Bandwidth and latency requirements for smart transmission grid applications," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1344–1352, 2012.

[3] J. G. Deshpande, E. Kim, and M. Thottan, "Differentiated services QoS in smart grid communication networks," *Bell Labs Technical Journal*, vol. 16, no. 3, pp. 61–81, 2011.

[4] O.-H. Borlaug, "Real-time applications in smart grids using internet communications:: latency issues and mitigations," 2014.

[5] The Virtual Access GW2027 Industrial Router, Virtual Access, 4, 2016.

[6] A. Radwan, H.-L. To, and W.-J. Hwang, "Optimal control for bufferbloat queue management using indirect method with parametric optimization," *Scientific Programming*, vol. 2016, Article ID 4180817, 10 pages, 2016.

[7] J. Gettys and K. Nichols, "Bufferbloat: dark buffers in the internet," *Queue*, vol. 9, no. 11, p. 40, 2011.

[8] D. Wischik and N. McKeown, "Part I: buffer sizes for core routers," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 3, pp. 75–78, 2005.

[9] D. Taht and J. Gettys, "Best practices for benchmarking codel and fq codel," Wiki Page on bufferbloat.net Web Site, 2012.

[10] V. G. Cerf, "Bufferbloat and other internet challenges," *IEEE Internet Computing*, vol. 18, no. 5, p. 80, 2014.

[11] K. Nichols and V. Jacobson, "Controlling queue delay," *Communications of the ACM*, vol. 55, no. 7, pp. 42–50, 2012.

[12] V. P. Rao, M. P. Tahiliani, and U. K. K. Shenoy, "Analysis of sfqCoDel for active queue management," in *Proceedings of the 5th International Conference on the Applications of Digital Information and Web Technologies (ICADIWT '14)*, pp. 262–267, Bengaluru, India, February 2014.

[13] R. Pan, P. Natarajan, C. Piglione et al., "PIE: a lightweight control scheme to address the bufferbloat problem," in *Proceedings of the IEEE 14th International Conference on High Performance Switching and Routing (HPSR '13)*, pp. 148–155, IEEE, Taipei, Taiwan, July 2013.

[14] C. V. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "A control theoretic analysis of RED," in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '01)*, vol. 3, pp. 1510–1519, April 2001.

[15] G. Raina, D. Towsley, and D. Wischik, "Part ii: control theory for buffer sizing," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 3, pp. 79–82, 2005.

[16] W. K. Ho, K. W. Lim, and W. Xu, "Optimal gain and phase margin tuning for PID controllers," *Automatica*, vol. 34, no. 8, pp. 1009–1014, 1998.

[17] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[18] M. May, J. Bolot, C. Diot, and B. Lyles, "Reasons not to deploy red," in *Proceedings of the IEEE 7th International Workshop on Quality of Service (IWQoS '99)*, pp. 260–262, London, UK, June 1999.

[19] T. Høiland-Jørgensen, "Technical description of fq codel," 2014.

[20] C.-K. Chen, H.-H. Kuo, J.-J. Yan, and T.-L. Liao, "GA-based PID active queue management control design for a class of TCP communication networks," *Expert Systems with Applications*, vol. 36, no. 2, pp. 1903–1913, 2009.

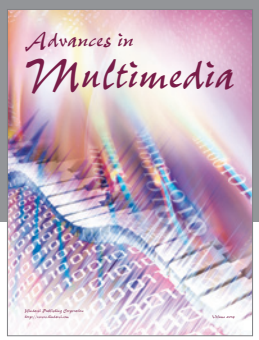
[21] K. B. Kim, "Design of feedback controls supporting TCP based on the state-space approach," *IEEE Transactions on Automatic Control*, vol. 51, no. 7, pp. 1086–1099, 2006.

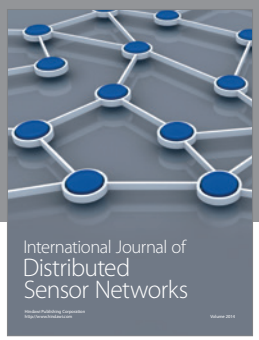
[22] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, no. 1, pp. 1–14, 1989.

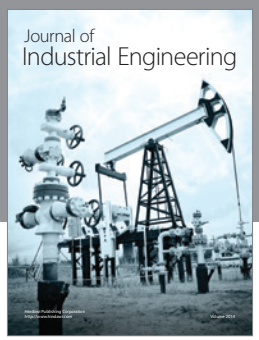
[23] G. Patil and G. Raina, "Some guidelines for queue management in high-speed networks," in *Proceedings of the 3rd Asian Himalayas International Conference on Internet (AH-ICI '12)*, pp. 1–6, IEEE, Kathmandu, Nepal, November 2012.

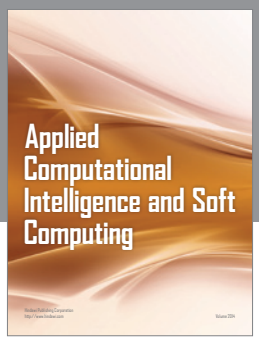
[24] M. Chiang, A. Sutivong, and S. Boyd, "Efficient nonlinear optimizations of queuing systems," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '02)*, vol. 3, pp. 2425–2429, IEEE, Taipei, Taiwan, November 2002.

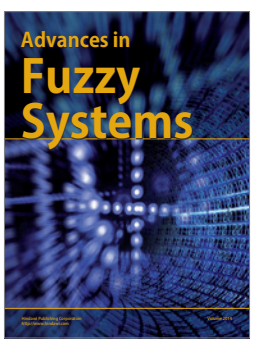
[25] G. Wang, Y. Xia, and D. Harrison, "An NS2 TCP evaluation tool," Tech. Rep., Internet Engineering Task Force, 2007.

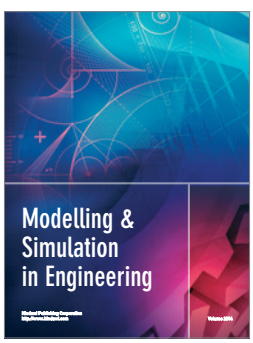


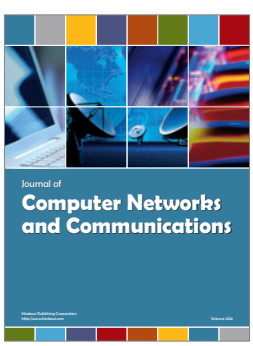












Hindawi
Submit your manuscripts at
http://www.hindawi.com