

# Modeling TCP/RED: a dynamical approach

Hui Zhang, Mingjian Liu, Vladimir Vukadinović, and Ljiljana Trajković

Simon Fraser University, Burnaby, BC, V5A 1S6 Canada {hzhange, jliu1, vladimir, ljilja}@cs.sfu.ca

**Summary.** Interaction between Transmission Control Protocol (TCP) and Random Early Detection (RED) gateways can be captured using dynamical models. The communication network is viewed as a discrete-time feedback control system where TCP adjusts its window size depending on whether or not it has detected a packet loss during the previous round trip time (RTT) interval. In this article, we describe a discrete-time nonlinear dynamical model for interaction between TCP and RED gateways. The model, constructed using an iterative map, captures a detailed dynamical behavior of TCP/RED, including slow start, fast retransmit, and timeout events common in TCP. Model performance for various RED parameters is evaluated using ns-2 network simulator.

## 1 Introduction

Today's Internet applications, such as World Wide Web, file transfer, Usenet news, and remote login, are delivered via Transmission Control Protocol (TCP). With an increasing number and variety of Internet applications, congestion control becomes a key issue. Active Queue Management (AQM) interacts with TCP congestion control mechanisms and plays an important role in meeting today's increasing demand for performance of Internet applications. Random Early Detection (RED), a widely deployed AQM algorithm, is a gateway-based congestion control mechanism. An accurate model of TCP with RED may help understand and predict the dynamical behavior of the network. In addition, the model may help analyze the stability margins of the system and provide design guidelines for selecting network parameters. The design guidelines are important to network designers who aim to improve network robustness. Therefore, modeling TCP with RED is an important step toward improving the service provided to Internet users and the network efficiency.

Modeling TCP performance has gained increased attention during the last few years, due to the benefits that TCP models offer to the networking community. Analytical TCP models enable researchers to closely examine the existing congestion control algorithms, address their shortcomings, and propose methods for improvement. They may also be used to compare various TCP flavors and implementations, and to determine their performance under

various operating conditions. Moreover, these models help examine the interactions between TCP and the queuing algorithms implemented in network routers. Hence, they help improve the existing and design better algorithms, such as AQM techniques. Finally, such models offer the possibility of defining TCP-friendly behavior in terms of throughput for non-TCP flows that coexist with TCP connections in the same network.

The goal of TCP modeling is to investigate the nonlinear phenomena in a TCP/RED system. We use an iterative map to model the system. We derive a second-order discrete-time model to capture the interactions of TCP congestion control algorithm with the RED mechanism. We use the concepts proposed in [31] and construct a nonlinear dynamical model of TCP/RED that employs two state variables: the window size and the average queue size. The change of window size reflects the dynamics of TCP congestion control, while the average queue size captures the queue dynamics in RED gateway. The novelty of the proposed model is in capturing detailed dynamical behavior of TCP/RED. The proposed model considers slow start phase and takes into account timeout events common in TCP.

This article is organized as follows: In Section 2, we briefly describe TCP congestion control and the RED algorithm. In Section 3 we survey related work. A nonlinear second-order discrete-time model named S-model is introduced in Section 4. In Section 5, we compare its performance to the ns-2 simulation results and to an existing model. Conclusions are given in Section 6.

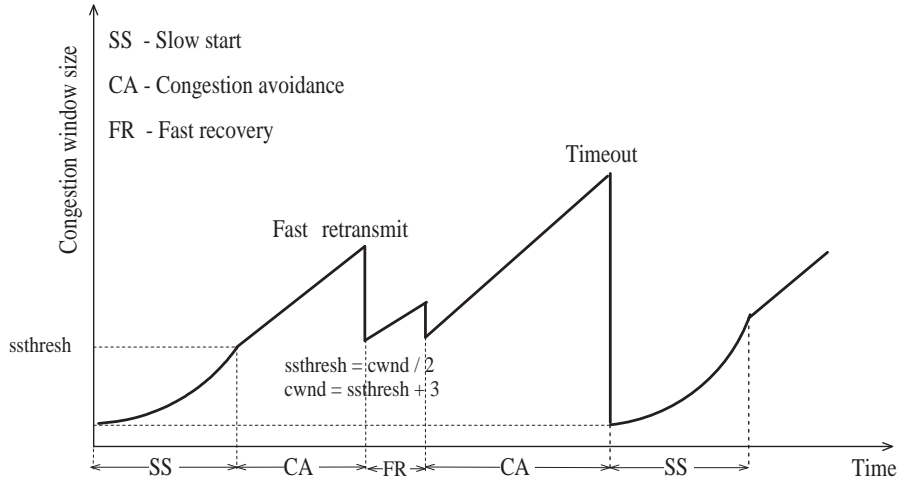
## 2 TCP and RED Algorithms

In this section, we describe TCP congestion control mechanisms and the RED algorithm.

### 2.1 TCP Congestion Control Algorithms

To adjust the window size, the TCP congestion control mechanism employs four algorithms: slow start, congestion avoidance, fast retransmit, and fast recovery, as shown in Fig. 1. They were introduced by Jacobson [18], [19] and are described in RFCs 1122 [4] and 2581 [1]. We briefly describe here their basic characteristics.

In order to avoid congesting the network with large bursts of data, an established TCP connection first employs the slow start algorithm to detect the available bandwidth in the network. Typically, a TCP sender initializes its congestion window (*cwnd*) to one or two segments, depending on the TCP implementation. Upon receipt of each acknowledgment (ACK) that acknowledges receipt of new data by the receiver, TCP increments *cwnd* by one segment size.



**Fig. 1.** Evolution of the window size in TCP Reno. It consists of slow start, congestion avoidance, fast retransmit, and fast recovery phase.

When  $cwnd$  exceeds a threshold ( $ssthresh$ ), the sender's mechanism leaves the slow start and enters the congestion avoidance phase. During the congestion avoidance,  $cwnd$  is incremented by one segment size per round trip time (RTT). A timer is set every time a sender sends a packet. A packet loss is detected by the timeout mechanism if the timer expires before the receipt of the packet has been acknowledged. If a packet loss is detected by the timeout mechanism, the TCP sender adjusts its  $ssthresh$  and switches back to the slow start.

The fast retransmit algorithm is used for recovery from losses detected by triple duplicate ACKs. Whenever a TCP receiver receives an out-of-order segment, it immediately sends a duplicate ACK, which informs the sender of the sequence number of the packet that the receiver expects. The receipt of triple duplicate ACKs (four consecutive ACKs acknowledging the same packet) is used as an indication of packet loss. The TCP sender reacts to the packet loss by halving  $cwnd$  and re-transmitting the lost packet, without waiting for the retransmission timer to expire.

The fast recovery algorithm is used to control data transmission after fast retransmission of the lost packet. During this phase, the TCP sender increases its  $cwnd$  for each duplicate ACK received. The fast recovery algorithm recognizes each duplicate ACK as an indication that one packet has left the channel and has reached the destination. Since the number of outstanding packets has decreased by one, TCP sender is allowed to increment its  $cwnd$ . When a non-duplicate ACK is received, TCP switches from the fast recovery to the congestion avoidance phase.

## 2.2 TCP Implementations

Older TCP implementation, released in the early 1980s, employed a simple window-based congestion control specified in RFC 793 [34]. TCP Tahoe, released in the late 1980s, employed the slow start, congestion avoidance, and fast retransmit algorithms. TCP Reno, introduced in the early 1990s, added the fast recovery algorithm.

Using ns-2 simulations, Fall and Floyd [11] demonstrated that TCP Reno exhibits poor performance in terms of link utilization whenever multiple packets are dropped from a single window of data. To alleviate this problem, they introduced two modifications to TCP Reno: TCP New-Reno and TCP SACK [23]. A large number of Internet Web servers still use TCP Reno and its variants [32].

Other TCP implementations, such as TCP Vegas [5] and TCP Westwood [7], use various techniques to avoid congestion. They adjust the congestion window size based on estimates of the throughput at the bottleneck.

## 2.3 RED Algorithm

A traditional DropTail queue management mechanism discards the packets that arrive when the buffer is full. However, this method has two drawbacks. First, it may allow few connections to monopolize the queue space so that other flows are starved. Second, DropTail allows queues to be full for a long period of time. During that period, incoming packets are dropped in bursts. This causes severe reduction in throughput of TCP flows. One solution, recommended in RFC 2309 [2], is to deploy active queue management (AQM) algorithms. The purpose of AQM is to react on incipient congestion, before the buffer overflows. Active queue management allows responsive flows, such as TCP flows, to react timely and reduce their sending rates in order to prevent congestion and severe packet losses.

The most popular active queue management algorithm is Random Early Detection (RED), proposed by Floyd and Jacobson [14]. The RED mechanism calculates exponentially weighted moving average of the queue size. Let  $w_q$  be the weight factor and  $q_{k+1}$  be the current queue size. At every packet arrival, RED gateway updates the average queue size as:

$$\bar{q}_{k+1} = (1 - w_q)\bar{q}_k + w_q \cdot q_{k+1}. \quad (1)$$

The average queue size is compared to two parameters: minimum queue threshold  $q_{min}$  and maximum queue threshold  $q_{max}$ . If the average queue size is smaller than  $q_{min}$ , the packet is admitted to the queue. If it exceeds  $q_{max}$ , the packet is marked or dropped. If the average queue size is between  $q_{min}$  and  $q_{max}$ , the packet is dropped with a drop probability  $p$  that is a function of the average queue size:

$$p_{k+1} = \begin{cases} 0 & \text{if } \bar{q}_{k+1} \leq q_{min} \\ 1 & \text{if } \bar{q}_{k+1} \geq q_{max} \\ \frac{\bar{q}_{k+1} - q_{min}}{q_{max} - q_{min}} p_{max} & \text{otherwise} \end{cases}, \quad (2)$$

where  $p_{max}$  is the maximum packet drop probability. The relationship between the drop probability and the average queue size is shown in Fig. 2.

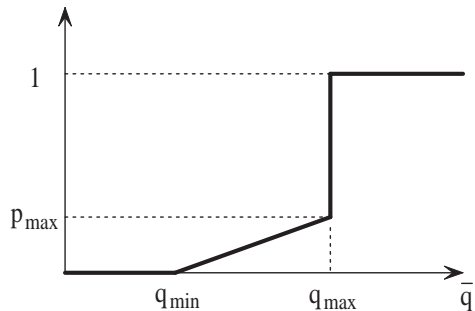


Fig. 2. RED drop probability as a function of the average queue size.

### 3 Modeling Methodologies

From the viewpoint of flow characteristics, analytical TCP models can be classified in three categories based on the duration of the TCP flows, which determines the dominant TCP congestion control algorithms to be modeled, and the aspects of TCP performance that can be captured by the model [20]. The first category models short-lived flows, where TCP performance is strongly affected by the connection establishment and slow start phases [27]. These models typically approximate average latency or completion time, i.e., the time it takes to transfer a certain amount of data. The second category models long-lived flows that characterize the steady-state performance of bulk TCP transfers during the congestion avoidance phase [26], [29], [31], [37]. These models approximate aspects such as the average throughput and window size evolution. The final category includes models for flows of arbitrary duration, i.e., those that can accommodate both short and long-lived flows [6], [8].

From the control theoretic point of view, the developed models of TCP and TCP/RED [16], [17], [21], [22], [28], [35], [36] can be classified into two types: averaged models and iterative map models. An averaged model is described by a set of continuous differential equations. It neglects the detailed dynamics and only captures the “low frequency characteristics” of the system. It can be used to analyze the steady-state performance and to predict

low frequency slow-scale bifurcation behavior, such as Hopf bifurcations. Examples of such models are given in [16], [17], [21]. In contrast, an iterative map model has a discrete-time form and employs a set of difference equations. It provides relatively complete dynamical information. Iterative maps are adequate to explore nonlinear phenomena, such as period-doubling and saddle-node bifurcations, which may appear across a wide spectrum of frequencies and cause the existence of solutions in the high frequency range. Examples of iterative maps are given in [22], [35], [36].

### 3.1 Survey of Related TCP/RED Models

Several models have been proposed recently in order to analyze and understand the performance of packet networks. A simple steady-state model of TCP Reno, introduced in [31], models the steady-state sending rate as a function of the loss rate and the round trip time (RTT) of a bulk data transfer TCP flow. The model not only captures the essence of TCP's fast retransmit and congestion avoidance, but it also takes into account the effect of timeout mechanism, which is important from a modeling perspective. Measurements demonstrated that the proposed model was adequate over a range of loss rates.

A simplified first-order discrete-time nonlinear dynamical model was developed for TCP with RED control in [22], [35], [36]. An exponentially weighted average queue size has been used as the state variable. The model describes the system dynamics over large parameter variations, and employs sampling the buffer occupancy at certain time instances. This dynamical model was used to investigate the stability, bifurcation, and routes to chaos in a network for various system parameters. Based on the developed model, the authors demonstrated that nonlinear phenomena, such as bifurcation and chaos, might occur if the system parameters were not properly selected. However, this discrete-time model neglects the dynamics of TCP. The derived map is:

$$q_{k+1}^{ave} = \begin{cases} (1-w)q_k^{ave} & \text{if } q_k^{ave} \geq q_u^{ave} \\ (1-w)q_k^{ave} + w \cdot B & \text{if } q_k^{ave} \leq q_l^{ave} \\ (1-w)q_k^{ave} + w(\frac{N \cdot K}{\sqrt{p_k}} - \frac{C \cdot d}{M}) & \text{otherwise} \end{cases}, \quad (3)$$

where

- $q_{k+1}^{ave} \doteq$  average queue size in round  $k+1$
- $q_k^{ave} \doteq$  average queue size in round  $k$
- $q_u^{ave} \doteq$  upper bound of average queue size
- $q_l^{ave} \doteq$  lower bound of average queue size
- $p_k \doteq$  drop probability at round  $k$
- $w \doteq$  queue weight in RED algorithm
- $B \doteq$  buffer size
- $N \doteq$  number of TCP connections

$K \doteq$  constant  $[1, \sqrt{8/3}]$   
 $C \doteq$  link capacity  
 $d \doteq$  round trip propagation delay  
 $M \doteq$  packet size.

In [28], a second-order nonlinear dynamical model was developed to examine the interactions of a set of TCP flows with RED routers. The model employs fluid-flow and stochastic differential equations. Window size and average queue length are used as state variables. From a set of coupled ordinary differential equations, the authors develop a numerical algorithm to obtain the transient behavior of average queue length, round trip time, and throughput of TCP flows. This model is described by nonlinear differential equations that employ the average values of network variables:

$$\begin{aligned}
 \dot{W}(t) &= \frac{1}{R(t)} - \frac{W(t)W(t-R(t))}{2R(t-R(t))}p(t-R(t)) \\
 \dot{q}(t) &= \frac{N(t)}{R(t)}W(t) - C
 \end{aligned} \quad , \quad (4)$$

where

$W(t) \doteq$  expectation of TCP window size  
 $q(t) \doteq$  expectation of queue length  
 $R(t) \doteq$  round trip time  
 $N(t) \doteq$  load factor (number of TCP sessions)  
 $p(t) \doteq$  probability of packet mark/drop  
 $C \doteq$  link capacity.

A third-order dynamical model that describes the interaction of TCP flows with an RED-controlled queue was developed in [21]. The state variables of the model are average queue length, instantaneous queue length, and throughput. TCP sources are idealized to operate only in congestion avoidance phase where congestion window follows the rule of linear increase and multiplicative decrease. This dynamical model is used to explore various parameter settings and observe transient and equilibrium behavior of the system. The validity of the model is verified by comparison with simulation results. The interaction between TCP and RED is modeled as:

$$\begin{aligned}
 \frac{d}{dt}\bar{s}(t) &= \bar{\lambda}(t-R/2)\beta(\bar{q}(t) - \bar{s}(t)) \\
 \frac{d}{dt}\bar{q}(t) &= \bar{\lambda}(t-R/2)(1 - \pi_K(\bar{q}(t)))(1 - p(\bar{s}(t))) - \mu(1 - \pi_0(\bar{q}(t))) \\
 \frac{d}{dt}\bar{\lambda}(t) &= -\frac{P_L(t-R/2)}{2m}\bar{\lambda}(t)\bar{\lambda}(t-R) + (1 - P_L(t-R/2))\frac{m}{R^2}\frac{\bar{\lambda}(t-R)}{\bar{\lambda}(t)} \\
 P_L(t) &= p(\bar{s}(t)) + \pi_K(\bar{q}(t)) - p(\bar{s}(t))\pi_K(\bar{q}(t))
 \end{aligned} \quad , \quad (5)$$

where

$\bar{s}(t) \doteq$  expectation of the exponentially averaged queue length  
 $\bar{q}(t) \doteq$  expectation instantaneous queue length  
 $\bar{\lambda}(t) \doteq$  expectation of TCP sending rate  
 $P_L(t) \doteq$  loss probability in the queue at time  $t$

- $\pi_K(\bar{q}(t)) \doteq$  steady-state probability for the queue to be full
- $\pi_0(\bar{q}(t)) \doteq$  steady-state probability for the queue to be empty
- $p \doteq$  drop probability
- $R \doteq$  round trip time
- $\beta \doteq$  queue weight in RED algorithm
- $m \doteq$  number of identical TCP sources.

In [16], [17], the authors obtained a second-order linear model for TCP and RED by linearizing a fluid-based nonlinear TCP model. Window size and average queue length are used as state variables of the system. The authors performed analysis of TCP interactions with RED from a control theoretic viewpoint. They presented design guidelines for choosing RED parameters that lead to local stability of the system. In addition, they proposed two alternative controllers to improve the transient behavior and stability of the system: a proportional (P) controller that possesses good transient response but suffers from steady-state errors in queue regulation, and a proportional-integral (PI) controller that exhibits zero steady-state regulation error and acceptable transient behavior. An important contribution of this paper is a good example how to use classical control theory to solve problems in complex communication systems. The model linearized around the operating point is described as:

$$\begin{aligned} \delta\dot{W}(t) &= -\frac{N}{R_0^2 C}(\delta W(t) + \delta W(t - R_0)) - \frac{1}{R_0^2 C}(\delta q(t) - \delta q(t - R_0)) \\ &\quad - \frac{R_0 \cdot C^2}{2N^2} \delta p(t - C) \\ \delta\dot{q}(t) &= \frac{N}{R_0} \delta W(t) - \frac{1}{R_0} \delta q(t) \end{aligned}, \quad (6)$$

where

- $\delta W \doteq W - W_0$
- $\delta q \doteq q - q_0$
- $\delta p \doteq p - p_0$
- $W_0, q_0, p_0 \doteq$  the set of operating points
- $W \doteq$  expectation of TCP window size
- $q \doteq$  expectation of queue length
- $R_0 \doteq$  round trip time
- $C \doteq$  link capacity
- $T_p \doteq$  propagation delay
- $N \doteq$  load factor (number of TCP sessions)
- $p \doteq$  probability of packet mark/drop.

A multi-link multi-source model [25] was used to study the stability of a general TCP/AQM system. A local stability condition were derived for the case of a single link with heterogeneous sources and the stability region of TCP/RED. The state variables of this model are window size, instantaneous queue length, and average queue length. large link capacities. Finally, they devised a new distributed congestion control algorithm that maintains



local stability for arbitrary delay, capacity, and traffic load. They provided preliminary simulation results to illustrate the model's behavior.

## 4 Discrete-time Dynamical Model of TCP/RED

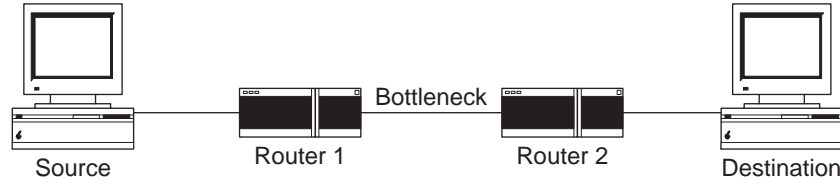
The basic idea behind RED is to sense impending congestion before it occurs and to try to provide feedback to senders by either dropping or marking packets. Hence, from the control theoretic point of view, the network may be considered as a complex feedback control system. TCP adjusts its sending rate depending on whether or not it has detected a packet drop in the previous RTT interval. The drop probability of RED can be considered as a control law of the network system. Its discontinuity is the main reason for oscillations and chaos in the system. Hence, it is natural to model the network system as a discrete-time model. In this article, we model the TCP/RED system using a “stroboscopic map”, which is the most widely used type of discrete-time maps for modeling power converters [3], [9], [10], [39]. This map is obtained by periodically sampling the system state. In our study, the sampling period is one RTT. Since window size and queue size behave as step functions of RTT, one RTT is the sampling period that captures their changes [13]. Higher sampling rate would not significantly improve the accuracy of the model. On the other hand, lower sampling rate would ignore the changes and affect the model accuracy.

State variables employed in the proposed S-model are window size and average queue size. These state variables capture the detailed behavior of TCP/RED. Variations of the window size reflect the dynamics of TCP congestion control. The window size increases exponentially and linearly in slow start and congestion avoidance phases, respectively. It multiplicatively decreases when loss occurs. The average queue size captures the queue dynamics in RED because it is updated upon every packet arrival. In our study, we do not consider instantaneous queue size as an independent state variable.

### 4.1 TCP/RED Model

We consider a simple network shown in Fig. 3. It consists of one TCP source, two routers, and a destination. RED is employed in Router 1. A TCP Reno connection is established between the source and the destination. Data packets are sent from the source to the destination, while traffic in the opposite direction consists of ACK packets only.

We made several assumptions in order to construct an approximate model. We assume that ACK packets are never lost. The connection is long-lived and the source always has sufficient data to send. Round trip propagation delay  $d$  between the source and destination and the data packet size  $M$  are kept constant. The link that connects the two routers is the only bottleneck in the network. We also assume that the timeout is caused only by packet loss and



**Fig. 3.** Topology of the modeled and simulated network.

that the duration of the timeout period is 5 RTTs [13]. The state variables of the system are sampled at the end of every RTT period. We assume that the queue size is constant during each sampling period. The model includes three cases, depending on the number of packets lost in the previous RTT period: no loss, single loss, and multiple packet losses.

#### 4.2 Case 1: No Loss

Let  $W_k$ ,  $q_k$ , and  $\bar{q}$  be the window size, queue size, and average queue size at the end of the sampling period  $k$ . If no packet is dropped during the last RTT period, TCP Reno increases its window size. The window size is increased exponentially in the slow start phase and linearly in the congestion avoidance phase:

$$W_{k+1} = \begin{cases} \min(2W_k, ssthresh) & \text{if } W_k < ssthresh \\ \min(W_k + 1, rwnd) & \text{if } W_k \geq ssthresh \end{cases}, \quad (7)$$

where  $rwnd$  is the receiver's advertised window size, i.e., the largest window size that the receiver could accept in one round. Usually,  $rwnd$  is greater than window size. In this case,  $rwnd$  does not affect the variations of window size. In case when window size increases linearly and reaches the value  $rwnd$ , window size is kept at  $rwnd$  until loss occurs in the network.

In order to calculate the average queue size given by Eq. (1), we need to find the queue size at the sampling period  $k + 1$ . This queue size depends on the queue size in the previous period, the current window size, and the number of packets that have left the queue during the previous sampling period. Therefore, the current queue size is:

$$\begin{aligned} q_{k+1} &= q_k + W_{k+1} - \frac{C \cdot RTT_{k+1}}{M} \\ q_{k+1} &= q_k + W_{k+1} - \frac{C}{M} \left( d + \frac{q_k \cdot M}{C} \right), \\ q_{k+1} &= W_{k+1} - \frac{C \cdot d}{M} \end{aligned} \quad (8)$$

where

- $q_{k+1} \doteq$  instantaneous queue size in round  $k + 1$
- $q_k \doteq$  instantaneous queue size in round  $k$
- $W_{k+1} \doteq$  current TCP window size in round  $k + 1$
- $RTT_{k+1} \doteq$  round trip time in round  $k + 1$

$C \doteq$  link capacity  
 $M \doteq$  packet size  
 $d \doteq$  round trip propagation delay.

Substituting  $q_{k+1}$  in (1) gives the average queue size:

$$\bar{q}_{k+1} = (1 - w_q)\bar{q}_k + w_q \cdot \max(W_{k+1} - \frac{C \cdot d}{M}, 0). \quad (9)$$

RED updates the average queue size at every packet arrival. Hence,  $\bar{q}$  is updated  $W_{k+1}$  times during the current sampling period. We assume that the queue size is constant during each period and that  $\bar{q}$  is given as:

$$\bar{q}_{k+1} = (1 - w_q)^{W_{k+1}}\bar{q}_k + (1 - (1 - w_q)^{W_{k+1}}) \cdot \max(W_{k+1} - \frac{C \cdot d}{M}, 0). \quad (10)$$

Finally, if  $p_k W_k < 0.5$ , which implies that no packet loss occurred in the previous sampling period, the state variables of the model are:

$$\begin{aligned} W_{k+1} &= \begin{cases} \min(2W_k, ssthresh) & \text{if } W_k < ssthresh \\ \min(W_k + 1, rwnd) & \text{if } W_k \geq ssthresh \end{cases} \\ \bar{q}_{k+1} &= (1 - w_q)^{W_{k+1}}\bar{q}_k + (1 - (1 - w_q)^{W_{k+1}}) \cdot \max(W_{k+1} - \frac{C \cdot d}{M}, 0). \end{aligned} \quad (11)$$

### 4.3 Case 2: One Packet Loss

If  $0.5 \leq p_k W_k < 1.5$ , which implies that one packet loss occurred in the previous RTT period, the congestion control mechanism of TCP Reno halves the window size in the current sampling period:

$$W_{k+1} = \frac{1}{2}W_k. \quad (12)$$

The average queue size is updated in a manner similar to Case 1:

$$\begin{aligned} W_{k+1} &= \frac{1}{2}W_k \\ \bar{q}_{k+1} &= (1 - w_q)^{W_{k+1}}\bar{q}_k + (1 - (1 - w_q)^{W_{k+1}}) \cdot \max(W_{k+1} - \frac{C \cdot d}{M}, 0). \end{aligned} \quad (13)$$

### 4.4 Case 3: At Least Two Packet Losses

In this case  $p_k W_k \geq 1.5$ , which implies that at least two packets are lost in the previous RTT period. When multiple packets are lost from the same window, TCP Reno may not be able to send a sufficient number of new packets in order to receive three duplicate ACKs for each packet lost. TCP source will often have to wait for the timeout before retransmitting the lost packet [11]. During the timeout period, the source does not send packets into the network. In S-model, window size is equivalent to the number of packets

that are sent by the source during one RTT period. Hence, we assume that the window size is zero during the timeout period.

RED mechanism updates the average queue size for each packet arrival. However, during timeout period there are no packet arrivals. Average queue size is not updated and has the same value as in the previous RTT period. RED takes this “idle time” period into account when it updates the average queue size upon the next packet arrival. However, S-model does not take into account the “idle time”. TCP/RED system during the timeout period is modeled as:

$$\begin{aligned} W_{k+1} &= 0 \\ \bar{q}_{k+1} &= \bar{q}_k \end{aligned} \quad (14)$$

The pseudo code describing the S-model is shown in Algorithm 1.

---

**Algorithm 1** S-model
 

---

```

Initialization:
 $\bar{q}_0 \leftarrow 0$ 
 $q_0 \leftarrow 0$ 
 $p_0 \leftarrow 0$ 
for every round
  calculate the product of  $p_k W_k$ 
  if  $p_k W_k < 0.5$  then
    compare  $W_k$  with ssthresh
    if  $W_k < \textit{ssthresh}$  then
       $W_{k+1} \leftarrow \min(2W_k, \textit{ssthresh})$ 
       $\bar{q}_{k+1} \leftarrow (1 - w_q)^{W_{k+1}} \bar{q}_k + (1 - (1 - w_q)^{W_{k+1}}) \max(W_{k+1} - \frac{Cd}{M}, 0)$ 
    else
       $W_{k+1} \leftarrow \min(W_k + 1, \textit{rwnd})$ 
       $\bar{q}_{k+1} \leftarrow (1 - w_q)^{W_{k+1}} \bar{q}_k + (1 - (1 - w_q)^{W_{k+1}}) \max(W_{k+1} - \frac{Cd}{M}, 0)$ 
    end if
    calculate the drop probability using Eq. (2)
  else if  $0.5 \leq p_k W_k < 1.5$  then
     $W_{k+1} \leftarrow \frac{1}{2} W_k$ 
     $\bar{q}_{k+1} \leftarrow (1 - w_q)^{W_{k+1}} \bar{q}_k + (1 - (1 - w_q)^{W_{k+1}}) \max(W_{k+1} - \frac{Cd}{M}, 0)$ 
    calculate the drop probability using Eq. (2)
  else
     $W_{k+1} \leftarrow 0$ 
     $\bar{q}_{k+1} \leftarrow \bar{q}_k$ 
  end if

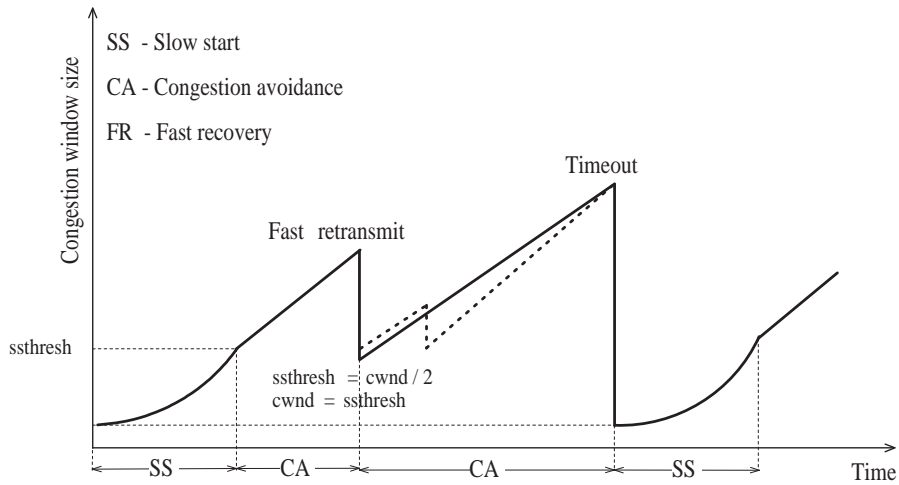
```

---

#### 4.5 Properties of the S-model

The proposed second-order discrete model captures interactions between TCP Reno congestion control algorithm and RED mechanism. It models

the dynamics of the TCP/RED system. Unlike past models [21], [22], [31], [35], [36], the S-model includes the slow start phase. It also takes into account timeout, common in TCP [31], which models [21], [22], [35], [36] ignore. However, the S-model does not capture all details of the fast recovery phase. Congestion window size in the S-model is not increased for each duplicate ACK received after retransmitting the lost packet. Instead of “inflating” the window we assume that TCP sender switches to the congestion avoidance phase without performing slow start. Evolution of the window size in the S-model is shown in Fig. 4.



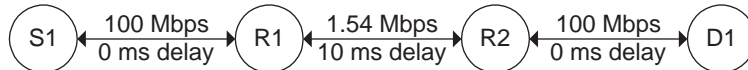
**Fig. 4.** Evolution of the window size in the proposed model. The fast recovery phase has been simplified.

The S-model captures the most important characteristics of the RED algorithm. The average queue size is updated after each packet arrival, i.e.,  $W_{k+1}$  times in the sampling period  $k+1$ . In contrast, models presented in [22], [35], [36] update the average queue size only once in every RTT period. However, in deriving the new model, we have also made simplifications in the RED algorithm: we ignored the counter that counts the number of packet arrivals since the last packet drop. RED uses this counter to modify drop probability (2). We have also ignored the “idle time” period, since it has no significant impact on the dynamics of the system.

## 5 S-Model Validation and Comparison

In order to verify the accuracy of the S-model, we compare its performance with the ns-2 simulation results. The topology of the simulated network is

shown in Fig. 5. It consists of one source, one sink, and two routers: R1 and R2. RED is employed in router R1. The link between R1 and R2 is the only bottleneck in the network. Its capacity is 1.54 Mbps and propagation delay is 10 ms. The capacity of the links between the source and R1 and between R2 and the sink is 100 Mbps. This is sufficient to guarantee no congestion in these links. Their propagation delay is 0 ms. We compared window size and average queue size in the S-model and the ns-2 simulation results.



**Fig. 5.** Topology of the simulated network.

The model validation is divided into four stages. First, we used default ns-2 RED parameters. Second, we choose various queue weights  $w_q$  while keeping other system parameters constant. In the third scenario, we varied the maximum drop probability  $p_{max}$ . Finally, we varied the minimum and maximum queue thresholds  $q_{min}$  and  $q_{max}$  simultaneously, while keeping their ratio  $q_{max}/q_{min} = 3$ . In each simulation scenario, we observed system behavior and measured average RTT, sending rate, and drop probability.

We also compared the proposed S-model with a discrete-time nonlinear dynamical model of TCP Reno with RED gateway proposed in [22], [35], [36], named here M-model. The M-model is a first-order discrete-time dynamical model with the average queue size as the state variable.

The default RED parameters are shown in Table 1. Other system parameters for the proposed S-model and the M-model are shown in Table 2.

**Table 1.** Default ns-2 RED parameters.

Packet size M (bytes)	500
Maximum drop probability ( $p_{max}$ )	0.1
Minimum queue threshold ( $q_{min}$ ) (packets)	5
Maximum queue threshold ( $q_{max}$ ) (packets)	15
Queue weight ( $w_q$ )	0.002

### 5.1 Default RED Parameters

In order to verify that the S-model can capture the detailed information of the system behavior, we evaluated the waveforms of the two state variables: window size and average queue size.

**Table 2.** System parameters.

	S-model	M-model
Link capacity C (bit/s)	1.54e+6	1.54e+6
Packet size M (bytes)	500	500
Round trip propagation delay d (ms)	22.8	22.8
Buffer size B (packets)	-	100
Slow start threshold size ssthresh (packets)	20	-
Number of TCP connection N	1	1
Constant K	-	$\sqrt{3/2}$

The waveforms of the window size for various time scales are shown in Fig. 6. The S-model and ns-2 simulation results are quite similar, especially during the steady-state response.

The waveforms of the average queue size for various time scales are shown in Fig. 7. The average queue size using the S-model is approximately one packet size larger than the average queue size obtained by the ns-2 simulations. This difference is due to introduced simplifications. The new model employs packet-marking/drop probability calculated by Eq. (2), while RED algorithm adopts a smooth packet-drop probability  $p_a$  that increases slowly as the *count* increases:

$$p_a \leftarrow p_b / (1 - \text{count} \cdot p_b), \quad (15)$$

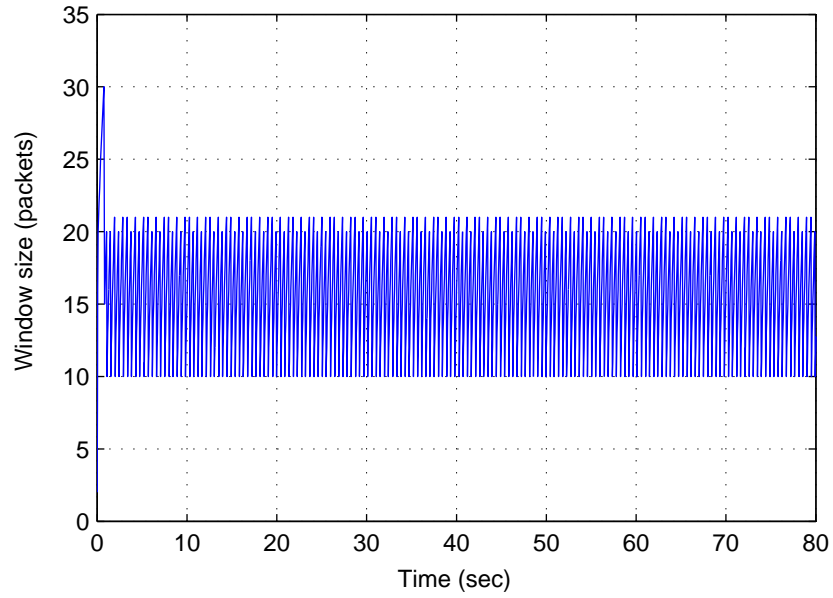
where  $p_b$  is the drop probability given by Eq. (2) and *count* measures the number of packets that have arrived since the last dropped packet. In the S-model,  $p_b$  is used as the final drop probability and the counter is ignored. Since  $p_b < p_a$ , the average queue size of the S-model is larger than that obtained via ns-2 simulations.

The statistics for the two state variables (window size and average queue size) and comparison with ns-2 simulation results are shown in Table 3.

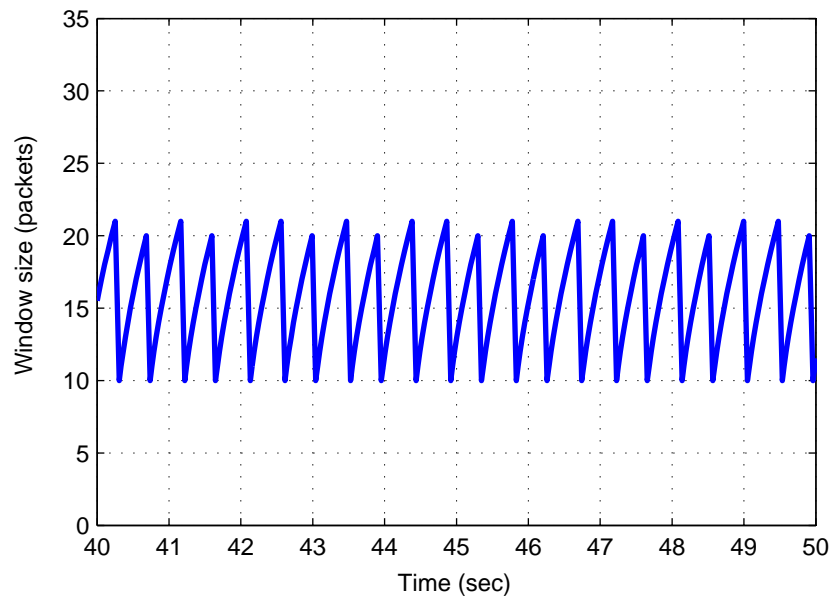
The M-model matches closely the ns-2 results shown in Figs. 7(c) and (d). The time waveform of the average queue size for the M-model is shown in Fig. 8. After the transient state, the average queue size reaches a constant value (5.71 packets), while the average queue size in ns-2 simulations varies around its fixed point. To the contrary, the proposed S-model captures the dynamical characteristic of the average queue size.

## 5.2 Queue Weight $w_q$

The S-model was also verified for various queue weight  $w_q$ . The average queue size during the steady-state for various values of  $w_q$  is shown in Fig. 9. When the queue weight increases, the average queue size slightly decreases in both the S-model and in ns-2 simulation results.

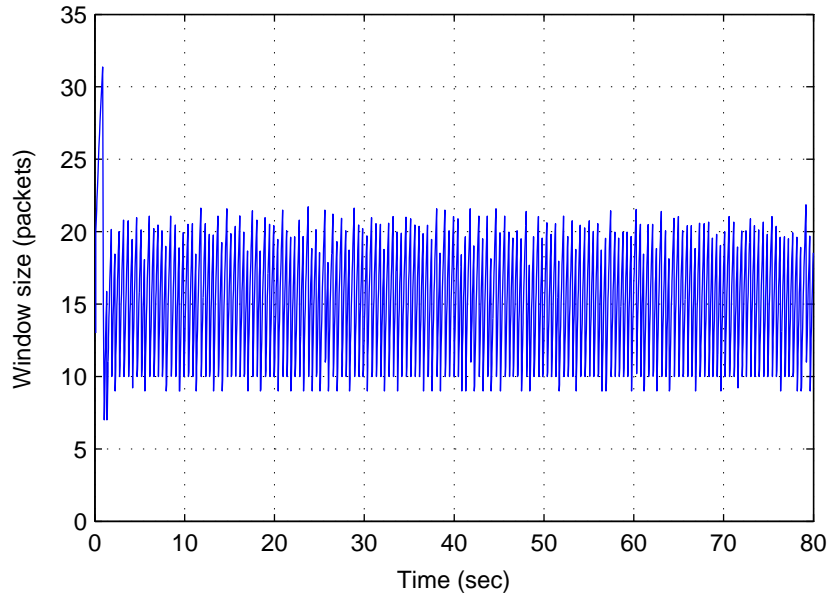


(a)

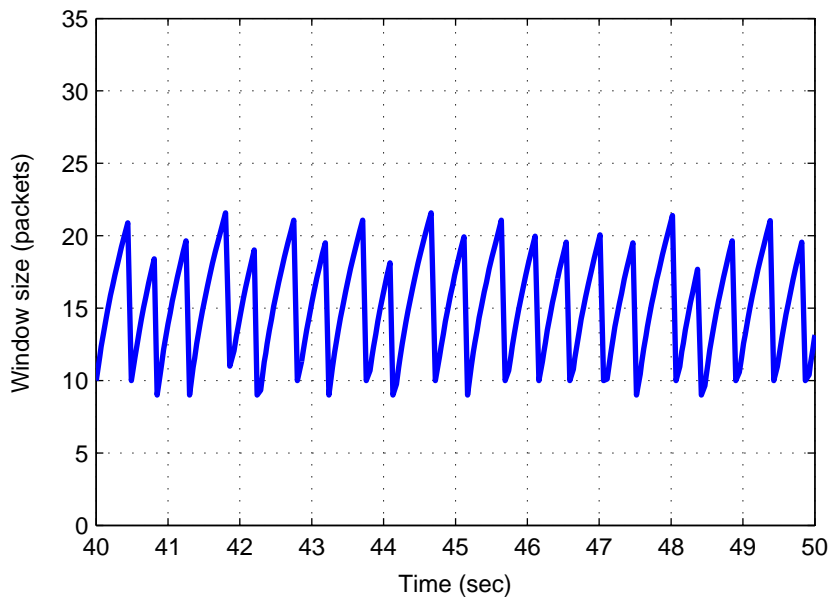


(b)



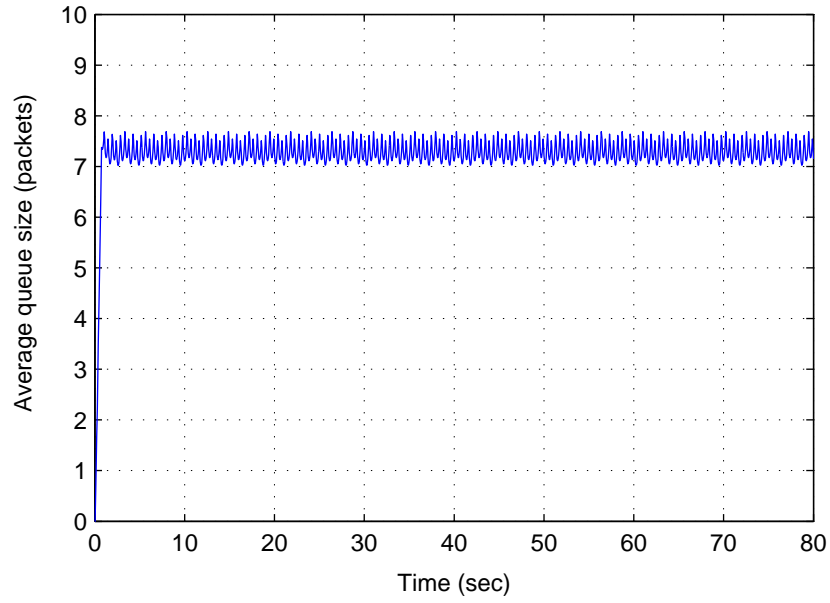


(c)

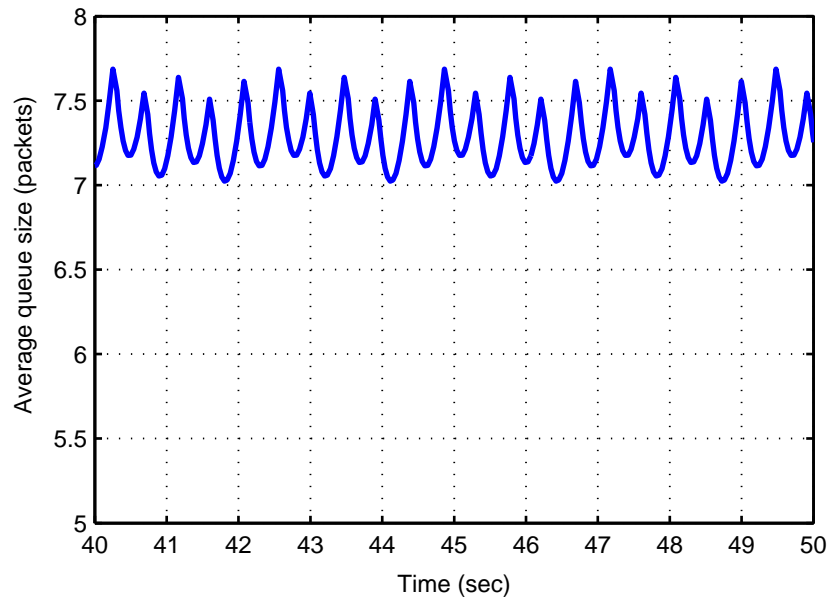


(d)

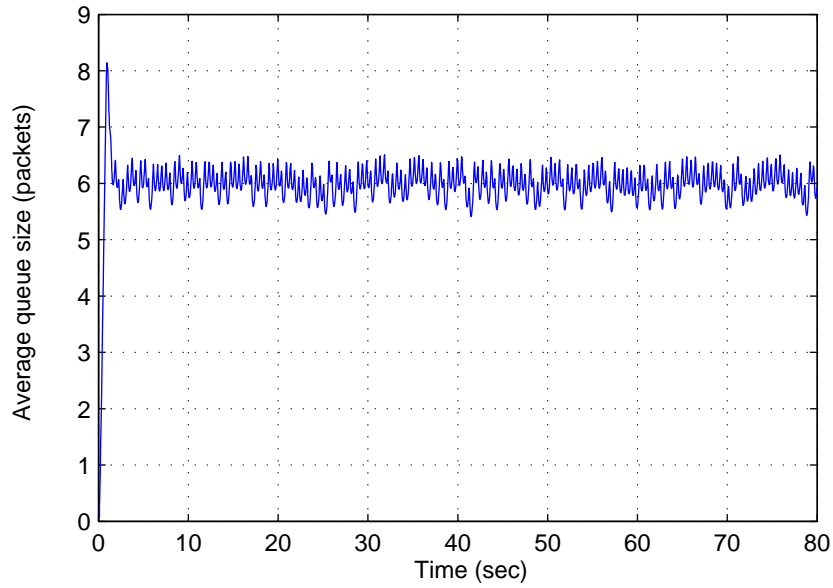
**Fig. 6.** Evolution of the window size with default RED parameters: (a) S-model and (b) zoom-in, (c) ns-2 simulation results and (d) zoom-in. Waveforms show good match between the S-model and the ns-2 simulation results.



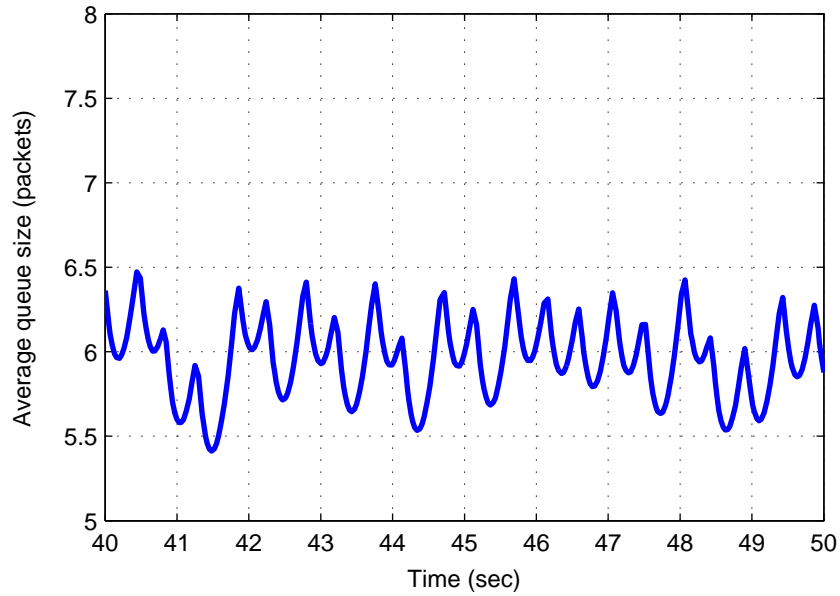
(a)



(b)



(c)



(d)

**Fig. 7.** Evolution of the average queue size with default RED parameters: (a) S-model and (b) zoom-in, (c) ns-2 simulation result and (d) zoom-in. The average queue size obtained using the S-model is higher than the average queue size obtained using ns-2 simulations.

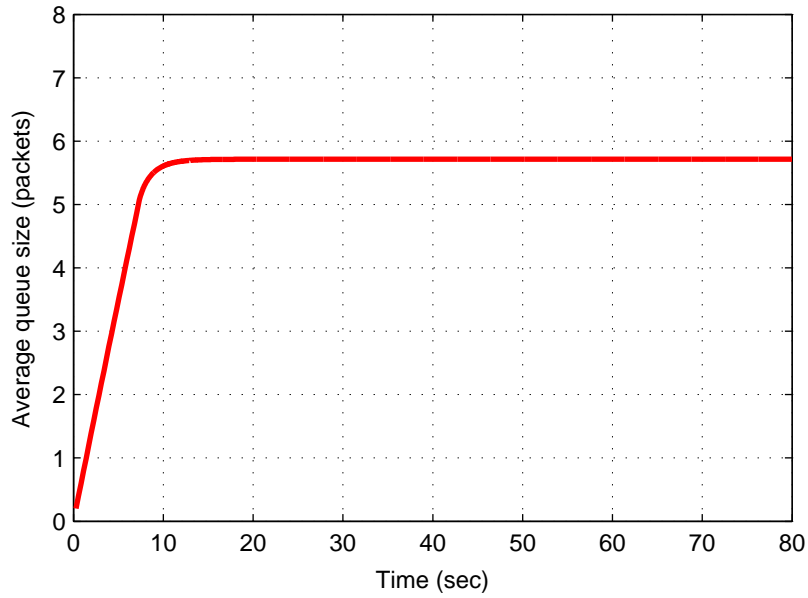


Fig. 8. Average queue size in the M-model with default RED parameters.

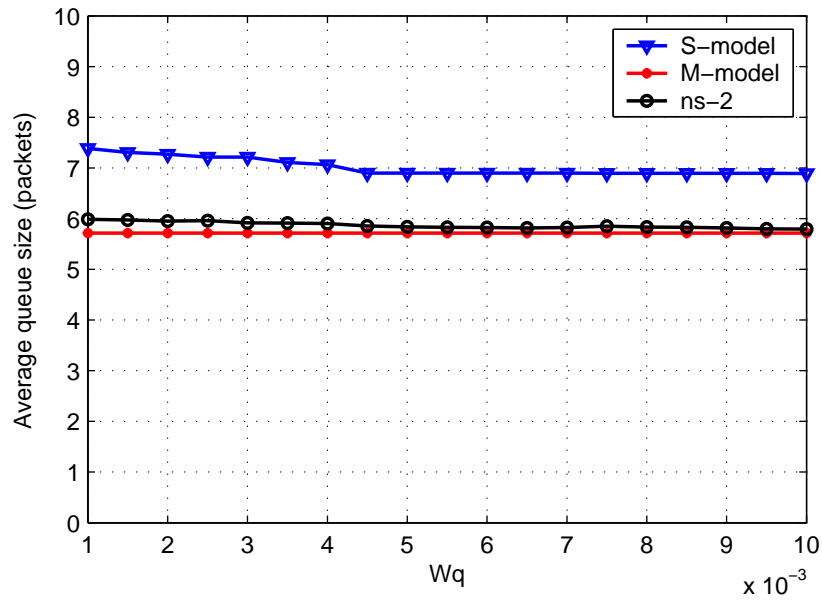


Fig. 9. Comparison of the average queue size for various  $w_q$ .

**Table 3.** State variables: S-model and ns-2 simulation results.

	Window size (packets)			Average queue size (packets)		
	S-model	ns-2	$\Delta$ (%)	S-model	ns-2	$\Delta$ (%)
Average	15.35	15.01	2.25	7.24	5.95	21.63
Max	30	31.38	-1.20	7.69	8.14	-5.60
Min	2	2	0	0.12	0.05	77.45
Max (steady-state)	21	21.86	-3.93	7.69	6.51	18.16
Min (steady-state)	10	9	11.11	7.02	5.41	29.79

The S-model was also validated for average RTT, sending rate, and packet loss rate. Results are summarized in Table 4. Values obtained using the S-model and ns-2 are quite similar. Small variations in queue weight have significant influence on RTT and packet loss rate.

As shown in Table 4, we also evaluated the M-model for the same system variables: average RTT, sending rate, and packet drop rate. Except for  $w_q = 0.001$ , when S-model performs better, the discrepancy in predicting RTT for two models are similar. In all cases, the S-model more accurately predicts the sending and drop rates.

### 5.3 Drop Probability $p_{max}$

We also evaluated the S-model for various  $p_{max}$ . When the maximum drop probability is set to a very small value, RED algorithm has small influence on the system behavior. In this case, the system behaves as TCP with DropTail, which leads to bursty packet losses and longer queuing delays. However, if the value of  $p_{max}$  is close to one, high drop rate will cause the system to become unstable. Simulation results are shown in Fig. 10. The average queue size decreases as the maximum drop rate increases. Results obtained from the S-model and ns-2 simulation results show the same trend.

Validation results for the average RTT, sending rate, and drop rate are listed in Table 5. They show that system variables in S-model and in ns-2 simulations change in a similar manner. As expected, when the maximum drop probability increases, the actual drop rate increases. At the same time, the average RTT decreases indicating a lower queuing delay.

The average RTT, sending rate, and drop rate for the M-model are also summarized in Table 5. Under various drop probabilities  $p_{max}$ , the S-model better estimates the average RTT, the sending rate, and the drop rate.

### 5.4 Thresholds $q_{min}$ and $q_{max}$

The S-model is also evaluated for various queue thresholds. Values of  $q_{min}$  and  $q_{max}$  are varied simultaneously, while maintaining the ratio  $q_{max}/q_{min}=3$ , as

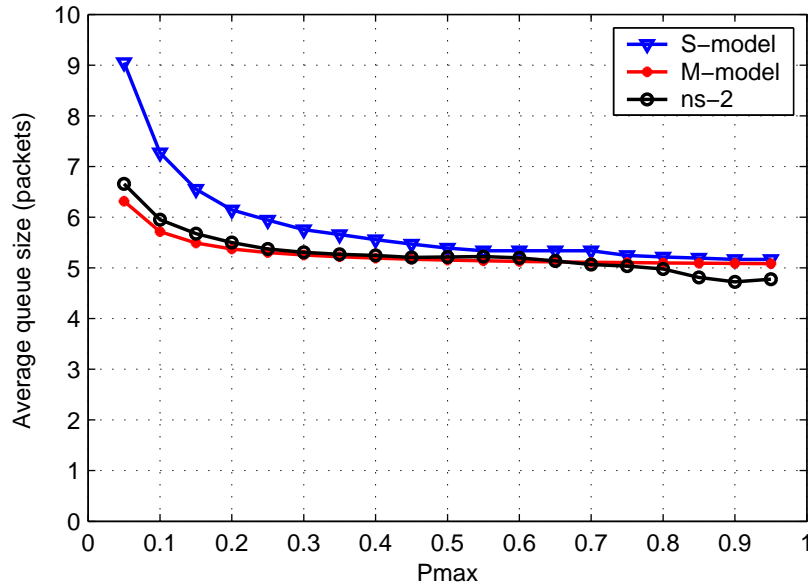


Fig. 10. Comparison of the average queue size for various  $p_{max}$ .

recommended in [2]. Simulation results for the average queue size are shown in Fig. 11.

The average queue size increases when thresholds  $q_{max}$  and  $q_{min}$  increase. At the same time, the average RTT increases and the drop rate decreases, as shown in Table 6.

A comparison with the M-model suggests that the proposed S-model is more accurate in predicting average RTT, sending rate, and drop rate.

### 5.5 Validation Summary

The S-model was evaluated by comparing the waveforms of its two state variables (window size and average queue size) to the ns-2 simulation results. While the window sizes match well, the steady-state values of the average queue size differ. Nevertheless, the average queue size of the S-model and ns-2 results have similar trend as the system parameters  $w_q$ ,  $p_{max}$ ,  $q_{min}$ , and  $q_{max}$  vary.

The difference in average queue size between the S-model and ns-2 is due to simplifications to the RED's packet discarding algorithm: S-model employs probability  $p_b$  (Eq. 2) as the final drop probability, while RED in ns-2 uses  $p_a$  (Eq. 15). If a modified drop probability  $p_a = \alpha p_b$  is used, the window size and the average queue size would evolve as shown in Figs. 12(a) and (b), respectively. Comparison shows that the average queue size matches well the ns-2 simulation results for modified drop probabilities with  $\alpha = 1.8$ .

**Table 4.** System variables for various  $w_q$ .

Parameters Average RTT (ms)					
weight ( $w_q$ )	S-model	$\Delta$ (%)	M-model	$\Delta$ (%)	ns-2
0.001	40.3	11.63	45.8	26.87	36.1
0.002	39.9	10.83	41.3	14.72	36.0
0.004	39.4	8.80	39.4	8.84	36.2
0.006	39.0	8.93	38.8	8.38	35.8
0.008	39.0	8.90	38.5	7.54	35.8
0.01	38.9	8.96	38.3	7.28	35.7
Sending rate (packets/s)					
weight ( $w_q$ )	S-model	$\Delta$ (%)	M-model	$\Delta$ (%)	ns-2
0.001	384.99	0.07	325.89	-15.29	384.71
0.002	384.98	0.06	355.26	-7.67	384.77
0.004	385.11	0.08	370.04	-3.83	384.79
0.006	385.08	0.09	374.90	-2.55	384.73
0.008	385.10	0.11	377.25	-1.93	384.68
0.01	385.02	0.08	378.37	-1.65	384.70
Drop rate (%)					
weight ( $w_q$ )	S-model	$\Delta$ (%)	M-model	$\Delta$ (%)	ns-2
0.001	0.55	1.29	0.67	23.39	0.54
0.002	0.56	2.56	0.70	28.21	0.55
0.004	0.59	6.12	0.71	27.70	0.56
0.006	0.60	7.91	0.71	27.70	0.56
0.008	0.61	11.11	0.71	29.33	0.55
0.01	0.61	11.72	0.71	30.04	0.55

## 6 Conclusions

TCP/RED system can be viewed as a complex feedback control system where TCP adjusts its sending rate depending on the packet loss probability determined by RED. In this article, we have introduced a second-order discrete model for interaction between TCP Reno and RED algorithms. We used an iterative map to construct the discrete-time model of the system. The S-model captures the dynamical behavior of TCP/RED and may be used to study its nonlinear behavior. Unlike other models, it takes into account the TCP slow start and timeout events. We evaluated the model by comparing its performance to the ns-2 simulation results and an existing TCP/RED model. Validation of the proposed model illustrates the performance of the model for various RED parameters.

**Table 5.** System variables for various  $p_{max}$ .

Parameters Average RTT (ms)					
$p_{max}$	S-model $\Delta$ (%)		M-model $\Delta$ (%)		ns-2
0.05	44.3	16.27	43.4	13.91	38.1
0.1	39.9	10.83	41.3	14.72	36.0
0.25	36.5	5.80	39.9	15.65	34.5
0.5	35.3	3.80	39.4	15.88	34.0
0.75	34.8	-0.85	39.2	11.68	35.1
Sending rate (packets/s)					
$p_{max}$	S-model $\Delta$ (%)		M-model $\Delta$ (%)		ns-2
0.05	385.13	0.11	354.23	-7.92	384.70
0.1	384.98	0.06	355.26	-7.67	384.77
0.25	384.93	0.05	356.02	-7.46	384.73
0.5	384.98	1.48	356.27	-6.09	379.37
0.75	384.63	7.60	356.33	-0.34	357.55
Drop rate (%)					
$p_{max}$	S-model $\Delta$ (%)		M-model $\Delta$ (%)		ns-2
0.05	0.45	-11.76	0.63	23.53	0.51
0.1	0.56	2.56	0.70	28.21	0.55
0.25	0.65	11.28	0.74	26.50	0.59
0.5	0.73	19.09	0.76	23.98	0.61
0.75	0.74	14.37	0.77	19.01	0.65

## 7 Acknowledgment

The authors thank I. Khalifa, W. G. Zeng, N. Cackov, B. Vujičić, S. Vujičić, Q. Shao, and J. Chen for comments, discussions, and suggestions that helped improve the content of the article.

## References

1. M. Allman, V. Paxson, and W. Steven, “TCP Congestion Control,” *Request for Comment (RFC) 2581*, Apr. 1999.
2. B. Barden et al., “Recommendations on queue management and congestion avoidance in the Internet,” *Request for Comments (RFC) 2309*, Apr. 1998.
3. M. di Bernardo and C. K. Tse, “Chaos in power electronics: an overview,” *Chaos in Circuits and Systems*, New York: World Scientific, pp. 317–340, 2002.
4. R. Braden, “Requirements for Internet hosts–communication layers,” *Request for Comment (RFC) 1122*, Oct. 1989.



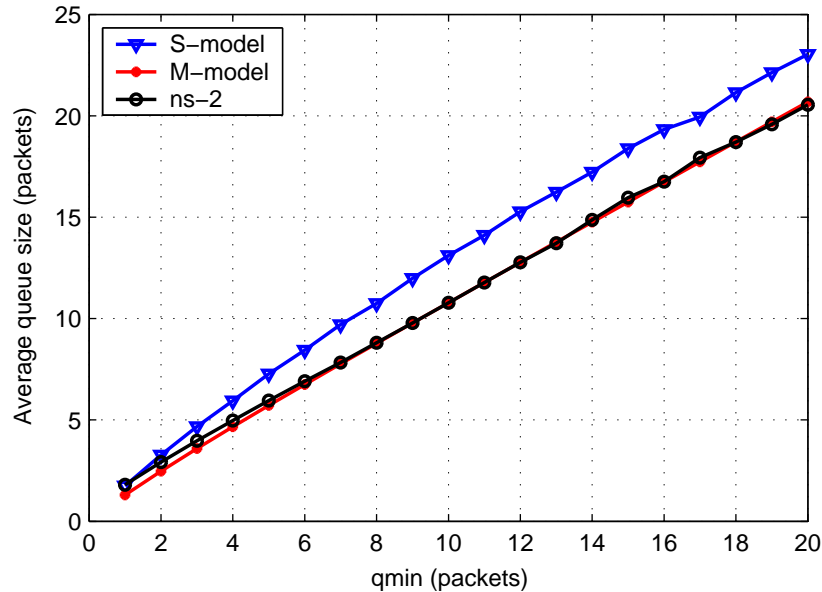
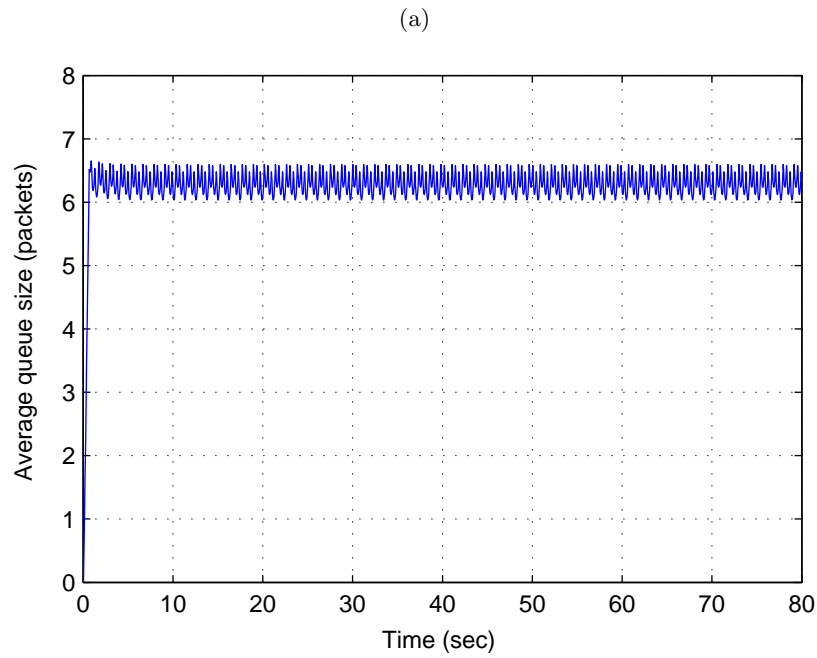
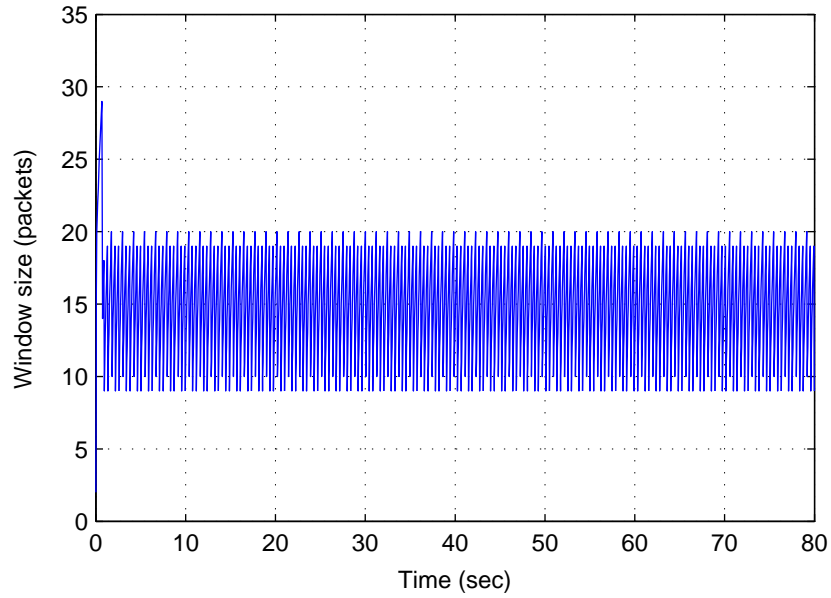


Fig. 11. Comparison of the average queue size for various  $q_{min}$  and  $q_{max}$ .

5. L. Brakmo and L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communication*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.
6. N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP latency," in *Proc. IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000, vol. 3, pp. 1742–1751.
7. C. Casetti, M. Gerla, S. Lee, S. Mascolo, and M. Sanadidi, "TCP with faster recovery," in *Proc. MILCOM 2000*, Los Angeles, CA, USA, Oct. 2000, vol. 1, pp. 320–324.
8. C. Casetti and M. Meo, "A new approach to model the stationary behavior of TCP connections," in *Proc. INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000, vol. 1, pp. 367–375.
9. W. C. Y. Chan and C. K. Tse, "Study of bifurcation in current-programmed DC/DC boost converters: from quasi-periodicity to period-doubling," *IEEE Trans. Circuit and Systems I*, vol. 44, no. 12, pp. 1129–1142, Dec. 1997.
10. K. W. E. Cheng, M. Liu, and J. Wu, "Chaos study and parameter-space analysis of the DC-DC buck-boost converter," *IEE Proceedings-Electric Power Applications*, vol. 150, pp. 126–138, Mar. 2003.
11. K. Fall and S. Floyd, "Simulation-based comparison of Tahoe, Reno, and SACK TCP," *ACM Communication Review*, vol. 26, no. 3, pp. 5–21, July 1996.
12. K. Fall and K. Varadhan, "The ns Manual," UC Berkeley, LBL, USC/ISI, and Xerox PARC, June 2003: [http://www.isi.edu/nsnam/ns/doc/ns\\_doc.pdf](http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf).
13. V. Firoiu and M. Borden, "A study of active queue management for congestion control," in *Proc. IEEE INFOCOM 2000*, Tel-Aviv, Israel, Mar. 2000, vol. 3, pp. 1435–1444.



**Fig. 12.** Evolution of window size and average queue size under modified drop probability.

**Table 6.** System variables for various  $q_{min}$  and  $q_{max}$ .

Parameters	Average RTT (ms)				
$q_{min}$ (packets)	S-model	$\Delta$ (%)	M-model	$\Delta$ (%)	ns-2
3	33.4	7.40	34.1	9.65	31.1
5	39.9	10.83	41.3	14.72	36.0
10	54.7	13.72	60.8	26.40	48.1
15	67.7	12.27	83.1	37.81	60.3
20	79.1	8.36	109.1	49.45	73.0
Sending rate (packets/s)					
$q_{min}$ (packets)	S-model	$\Delta$ (%)	M-model	$\Delta$ (%)	ns-2
3	383.22	0.20	366.13	-4.26	382.44
5	384.98	0.06	355.26	-7.76	384.77
10	385.10	0.06	330.94	-14.01	384.85
15	385.06	0.06	311.01	-19.19	384.83
20	385.30	0.09	296.27	-23.04	384.95
Drop rate (%)					
$q_{min}$ (packets)	S-model	$\Delta$ (%)	M-model	$\Delta$ (%)	ns-2
3	0.78	10.01	0.96	35.40	0.71
5	0.56	2.56	0.70	28.21	0.55
10	0.31	-6.34	0.37	11.78	0.33
15	0.20	-10.71	0.22	-1.79	0.22
20	0.15	-5.66	0.14	-11.95	0.16

14. S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.
15. S. Floyd, "RED: discussions of setting parameters," Nov. 1997: <http://www.icir.org/floyd/REDparameters.txt>.
16. C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, "A control theoretic analysis of RED," in *Proc. IEEE INFOCOM 2001*, Anchorage, AK, Apr. 2001, vol. 3, pp. 1510–1519.
17. C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, "Analysis and design of controllers for AQM routers supporting TCP flows," *IEEE Trans. on Automatic Control*, vol. 47, no. 6, pp. 945–959, June 2002.
18. V. Jacobson, "Congestion avoidance and control," *ACM Computer Communication Review*, vol. 18, no. 4, pp. 314–329, Aug. 1988.
19. V. Jacobson, "Modified TCP congestion avoidance algorithm," <ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail>, Apr. 1990.
20. I. Khalifa and Lj. Trajković, "An overview and comparison of analytical TCP models," in *Proc. IEEE International Symposium on Circuits and Systems*, Vancouver, BC, Canada, May 2004, vol. V, pp. 469–472.

21. P. Kuusela, P. Lassila, J. Virtamo, and P. Key, "Modeling RED with idealized TCP sources," *9th IFIP Conference on Performance Modeling and Evaluation of ATM and IP Networks 2001*, Budapest, Hungary, June 2001, pp. 155–166.
22. R. J. La, P. Ranjan, and E. H. Abed, "Nonlinearity of TCP and instability with RED," in *Proc. SPIE ITCOM*, Boston, MA, USA, July 2002, pp. 283–294.
23. Y. Lai and C. Yao, "TCP congestion control algorithms and a performance comparison," in *Proc. 10th International Conference on Computer Communications and Networks*, Scottsdale, AZ, USA., Oct. 2001, pp. 523–526.
24. S. H. Low and D. E. Lapsley, "Optimization flow control—I: basic algorithm and convergence," *IEEE/ACM Trans. Networking*, vol. 7, no. 6, pp. 861–874, Dec. 1999.
25. S. H. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle, "Dynamics of TCP/RED and a scalable control," *Proc. IEEE INFOCOM 2002*, New York, NY, USA., June 2002, vol. 1, pp. 239–248.
26. M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm," *ACM Computer Communication Review*, vol. 27, no. 3, pp. 67–82, July 1997.
27. M. Mellia, I. Stoica, and H. Zhang, "TCP model for short lived flows," *IEEE Communications Letters*, vol. 6, no. 2, pp. 85–87, Feb. 2002.
28. V. Misra, W. B. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proc. ACM SIGCOMM 2000*, Stockholm, Sweden, Aug. 2000, pp. 151–160.
29. A. Misra, J. Baras, and T. Ott, "Generalized TCP congestion avoidance and its effect on bandwidth sharing and variability," in *Proc. IEEE GLOBECOM*, San Francisco, CA, USA., Nov. 2000, vol. 1, pp. 329–337.
30. ns-2 Network Simulator: <http://www.isi.edu/nsnam/ns>.
31. J. Padhye, V. Firoiu, and D. F. Towsley, "Modeling TCP Reno performance: a simple model and its empirical validation," *IEEE/ACM Trans. Networking*, vol. 8, no. 2, pp. 133–145, Apr. 2000.
32. J. Padhye and S. Floyd, "On inferring TCP behavior", in *Proc. ACM SIGCOMM 2001*, San Diego, CA, USA, Aug. 2001, pp. 287–298.
33. V. Paxson and M. Allman, "Computing TCP's Retransmission Timer," *Request for Comment (RFC) 2988*, Nov. 2000.
34. J. Postel, "Transmission control protocol," *Request for Comment (RFC) 793*, Sept. 1981.
35. P. Ranjan, E. H. Abed, and R. J. La, "Nonlinear instabilities in TCP-RED," in *Proc. IEEE INFOCOM 2002*, New York, NY, USA, June 2002, vol. 1, pp. 249–258.
36. P. Ranjan, R. J. La, and E. H. Abed, "Bifurcations of TCP and UDP traffic under RED," in *Proc. 10<sup>th</sup> Mediterranean Conference on Control and Automation (MED) 2002*, Lisbon, Portugal, July 2002.
37. R. Roy, R. C. Mudumbai, and S. S. Panwar, "Analysis of TCP congestion control using a fluid model," in *Proc. IEEE ICC 2001*, Helsinki, Finland, June 2001, vol. 8, pp. 2396–2403.
38. W. R. Stevens, *TCP/IP Illustrated, Volume 1: The protocols*. New York: Addison-Wesley, 1994.
39. C. K. Tse and M. D. Bernardo, "Complex behavior in switching power converters," *Proceedings of the IEEE*, vol. 90, no. 5, pp. 768–781, May 2002.