MODELING CONGESTED INTERNET CONNECTIONS

Ian Frommer,* Brian Hunt, Ryan Lance,
Edward Ott and James Yorke
University of Maryland
College Park, MD 20742, USA
email: orbit@glue.umd.edu

Eric Harder
US Department of Defense
Washington, DC, USA
email: ejh@tycho.ncsc.mil

ABSTRACT
In this paper, we introduce a continuous-time model aimed at capturing the dynamics of congested Internet connections. The model combines a system of differential equations with a sudden change in one of the state variables. Results from this model show good agreement with the well-known *ns* network simulator, better than the results of a previous, similar model. This is due in large part to the use of the sudden change to reflect the impact of lost data packets. We also discuss the potential use of this model in network traffic state estimation.

KEY WORDS
Simulations, Mathematical Modeling

1 Introduction

An Internet connection consists of the exchange of data packets between a source and destination through intermediate computers, known as routers. The transmission of data in the majority of Internet connections is controlled by the Transport Control Protocol (TCP) [1, 2]. TCP is responsible for initializing and completing connections, controlling the rate of flow of data packets during a connection, ensuring that lost packets are retransmitted, etc.

Congestion can occur when the rate of flow of the connection is limited by some link on the path from source to destination. This link is known as the *bottleneck* link. Routers have buffers in which to store packets in case one of their outgoing links reaches full capacity. If new packets arrive at a router whose buffer is full, that router must drop packets. There exist various strategies, known as active queue management (AQM) policies, for determining how and when to drop packets prior to the buffer filling. One commonly used AQM is Random Early Detection (RED) [3].

In this paper we model the interaction between TCP and RED for a simple network connection experiencing congestion. This scenario, or ones similar to it, have been modeled previously for the purposes of evaluating RED [4, 5, 6, 7], obtaining throughput expressions [8, 9], and obviating the need for time-consuming simulation [4, 10, 11, 12], among others. There are stochastic [9] and deterministic models [4, 5, 6], continuous-time [4] and discrete-time models [5, 6].

Our model is closest to that of Misra et al. [4]. That model successfully captures the *average* network state in a variety of situations. This allows the authors to analyze RED from a control theoretic basis. Our aim is to develop a model of greater accuracy that will be useful in estimation not only of the average network state, but of additional quantities. For example, a model capable of capturing the mean queue length allows one to estimate the mean round-trip time. But a model that can capture the range, and better still, the variance of the queue length, will allow one to estimate the range and variance of the round-trip time. While this level of model accuracy can be useful, it is necessary in a model that is to be used for our ultimate goal of network traffic state estimation. Given some knowledge of the state of the network, an accurate model may be combined with a filter-based state-estimation scheme (which we discuss briefly in Sec. 4), in order to estimate the full network state at the current time. Network state estimation can be useful from the perspectives of both security and performance.

In this paper, we focus on network connections in which the senders always have data to transmit for the duration of the connection. (This is known as a *bulk transfer*.) We also assume that the path is fixed, the transfer is one-way (the data packets travel in only one direction), any cross-traffic is negligible, and the path contains one bottleneck link whose capacity is less than that of all other links in the path. Most of these assumptions reflect typical network scenarios with the possible exceptions of the cross-traffic and bottleneck assumptions.

In an actual network, data traveling from the sender to the receiver is likely to pass through several intervening routers. However, based on our assumption that there is one bottleneck link, we need only consider the router whose outgoing link is this bottleneck link. All other routers simply forward the data along the path, and have unoccupied buffers. Thus the network we model can be represented as in Fig. 1.

2 The Model

We assume the reader is familiar with the workings of TCP and RED; a more thorough discussion of both along

*This work was supported by AFOSR grant # FA95500410319.

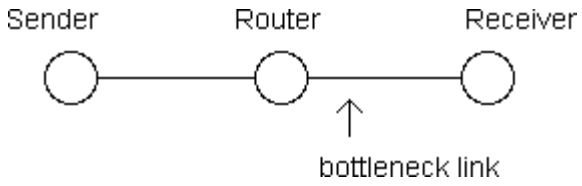

Figure 1. Network configuration

with a more detailed exposition of our model and a similar discrete-time model may be found in [13]. Previous attempts to deterministically model the TCP/RED interaction in the bottleneck/bulk situation we consider have made use of either discrete-time maps [5, 6] or differential equations [4]. Continuous-time models employing delay differential equations may do well in capturing the aspects of network behavior that evolve gradually on small-time scales, such as flow rate increases. However, they are likely to smooth out the effects of sudden large changes in state (such as flow rate reductions). Furthermore, systems of delay differential equations can be cumbersome to express and solve due to the delay.

We propose here a continuous-time model that combines a system of differential equations to capture the evolution of the small-time scale changes with a discrete impulse (sudden change in one of the state variables) to represent sudden large changes. In addition, it utilizes a packet-based frame of reference for the state, which simplifies handling of the delay in the system, making the model easier to express and execute.

To describe the system, we consider the state variables, all of which have units of "packets" but are allowed to take non-integer values:

- W – the congestion window size of the sender;
- q – the length of the queue at the router;
- x – the exponentially-weighted average of the queue length at the router (used by RED).

In our packet-based frame of reference, $W(t)$ represents the congestion window in effect for a packet *arriving at the queue* at time t. This was the sender's congestion window at an earlier time, when the packet left the sender.

In developing the model we will often refer to the round-trip time, R. The round-trip time is the time between the departure of a packet from the sender and the return to the sender of the acknowledgment for that packet. It consists of a fixed propagation delay, a, and a queuing delay. For a packet encountering a single congested router with queue length q, and outgoing link capacity c (measured in packets per unit time), the queuing delay is q/c. Thus,

$$R(q) = a + \frac{q}{c}. \tag{1}$$

As indicated by its name, RED has a stochastic component. Initially we will show how a deterministic model that does not reflect the random component of the dynamics can capture the system behavior under some network settings. As the network complexity grows, the impact of the random component increases. To address this, we add a stochastic component to our model in Sec. 3.

The system of differential equations in our model is similar to the one found in [4]. It is used to model the additive-increase of the window size, and both instantaneous and averaged queue lengths. The impulse in our model, in which the window size state variables are instantaneously reset, is used to model the multiplicative decrease in the window size caused by a dropped packet. In the interest of model simplicity, we do not model slow start or time-outs, and we neglect some details of packet retransmissions.

2.1 Modeling the Additive Increase

Although network data is transmitted in discrete packets, during the period of additive-increase and in the absence of dropped packets, quantities appear to vary smoothly when viewed over the time-scales we are interested in (on the order of 1 to 100 seconds) for our network settings. Hence we model the dynamics continuously as summarized by equations (2), (3), and (4):

$$\frac{dW(t)}{dt} = \frac{1}{R(q(t))} \tag{2}$$

$$\frac{dq(t)}{dt} = \frac{W(t)}{R(q(t))} - c \tag{3}$$

$$\frac{dx(t)}{dt} = wc(q(t) - x(t)). \tag{4}$$

Equation (2) reflects the approximation that a sender's TCP window increases at the rate of one packet per round-trip time. The rate of change of the queue length at a given time is equal to the difference between the flows into and out of the router. Because of our bulk-transfer and bottleneck assumptions (see Sec. 1), the network is usually congested to the point of saturation. We assume this is always the case, and consequently in equation (3) the flow rate out of the router always equals the capacity c of the outgoing link. The exponentially-weighted average queue is determined by RED upon each packet arrival as follows:

$$x_{n+1} = wq_{n+1} + (1 - w)x_n, \tag{5}$$

where w is the exponential-weighting parameter, $0 \leq w \leq 1$, and the subscripts denote successive measurements of q and x (which occur at packet arrivals) for a given router. This equation describes a low-pass filter, and, making use of the fact that w is small in practice, we approximate it by the differential equation (4). Here we have also simplified matters by replacing the flow rate into the queue with its time average value, which must equal the outgoing flow rate c.

2.2 Modeling the Multiplicative Decrease

In a real network situation, a router using RED will drop packets randomly when the exponentially averaged queue x exceeds a threshold q_{min}. In the interest of keeping our model simple, we initially tried having the model assume a drop occurred *as soon as* RED turned on (i.e., when x exceeded q_{min}). In reality, there tends to be a delay between the time RED turns on and a packet drop occurs. Under the network settings we used, this delay could be up to one second. By estimating this lag time as described below, the model results improved significantly compared to using a lag time of 0. Note that the model remains deterministic under this change.

In most descriptions of RED, the nominal drop probability p per packet is given as:

$$p(x) = \begin{cases} 0 & \text{when } x < q_{min}, \\ \frac{x - q_{min}}{q_{max} - q_{min}} p_{max} & \text{when } q_{min} \le x \le q_{max}, \\ 1 & \text{when } x > q_{max}, \end{cases} \tag{6}$$

where p_{max} and q_{max} are additional RED parameters. In practice, RED has an added layer of complexity. Namely, it has been designed so that effectively, the drop time is chosen from a uniform distribution on a time interval of length $1/p$, given a constant RED drop probability p. As implemented in the ns simulator [14], an additional *wait* parameter is used, which causes this time interval to lie between $1/p$ and $2/p$ from the time of the last drop. (This spacing also applies to the time between RED turning on and the first drop.) As a result, the expected time between drops is $3/(2p)$.

More precisely, when x exceeds q_{min} and RED turns on, it begins counting packets in order to determine when a packet drop should occur. The actual drop probability used by RED, which we denote p_{drop}, is given by:

$$p_{drop}(p, k) = \begin{cases} 0 & \text{when } k < 1/p, \\ p/(2 - kp) & \text{when } 1/p \le k < 2/p, \\ 1 & \text{when } k \ge 2/p, \end{cases} \tag{7}$$

where k is the number of packets that have arrived since RED turned on, or since the previous packet drop. Let K be the random variable representing the number of packets that arrive between drops, or between RED turning on and the first drop. If x is constant, then p is constant, in which case it can be shown that K is uniformly distributed between $1/p$ and $2/p$. This means $E[K] = 3/2p$.

Of course x is generally not constant, and in our model we make the rough approximation that $E[K]$ is the solution to the equation $k_{drop} = 3/[2p(x_{k_{drop}})]$, where x_k is the value of x when the packet counter reaches k. We further approximate x as a linear function of k between packet drops. Rewriting (5) as

$$x_k = x_{k-1} + w(q_k - x_{k-1}), \tag{8}$$

we replace $q_k - x_{k-1}$ by $q_0 - x_0$:

$$x_k = x_0 + kw(q_0 - x_0) \tag{9}$$

Assuming that x_k remains between q_{min} and q_{max},

$$p(k) = a_1 k + a_2 \tag{10}$$

where a_1 and a_2 are constants determined by equations (6) and (9).

Since we want to find k_{drop} such that $k_{drop} = 3/[2p(x_{k_{drop}})]$, we substitute (10) into this equation. Solving the resulting quadratic produces the following solution:

$$k_{drop} = \frac{-a_2 + \sqrt{a_2^2 + 6a_1}}{2a_1} \tag{11}$$

If $a_2^2 + 6a_1 < 0$, then since $a_1 < 0$, $p(k)$ is a decreasing function of k, and we assume that it becomes zero before another drop occurs.

When RED turns on, we compute k_{drop}, the expected time until the next packet drop. A timer variable counts down this amount and when it expires, we consider the drop to have occurred. We then continue to evolve the continuous system for one round-trip time to reflect the delay in notification of the sender that a packet has been dropped. After this delay, we let $M = W/2$, and hold $W = 0$ for M/c seconds, continuing to evolve q and x according to (3) and (4). This reflects the fact that after the window has been cut in half, the sender will not transmit data until M acknowledgments have been received; due to saturation, acknowledgments return every $1/c$ seconds. Once M/c seconds have elapsed, we set $W = M$ and hold the window state variable constant for one round-trip time (as measured from the time of the drop notification). This is done in order to represent the Fast Recovery/Fast Retransmit(FRFR) algorithm in the NewReno[1] version of TCP. Once this round-trip time has elapsed, the model resumes continuous evolution as described in Section 2.1.

2.3 Modeling Multiple Senders

The model can be easily generalized to handle more than one sender. As in the case of one sender, we assume that each sender is upstream from the same bottleneck link and is engaged in bulk transfer, sending data as fast as allowed by TCP. Following [4], denote the window size of the i^{th} sender by W_i. We keep track of each sender's window separately. The queue equation (3) is replaced by:

$$\frac{dq(t)}{dt} = \sum_{i=1}^{N} \frac{W_i(t)}{R_i(q(t))} - c, \tag{12}$$

where N is the number of senders. The connections need not all share the same fixed propagation delay, as the R_i terms indicate. We use a separate copy of (2) for each window W_i, and separate timer variables for each connection. When a drop occurs, we assign it to the sender with the largest window. Finally, to reflect the no-send period following a drop we wait for $\sum_{i=1}^{N} W_i/(2c)$ seconds before

[1] Recent indications are that NewReno is the most widely used variant of TCP [1].

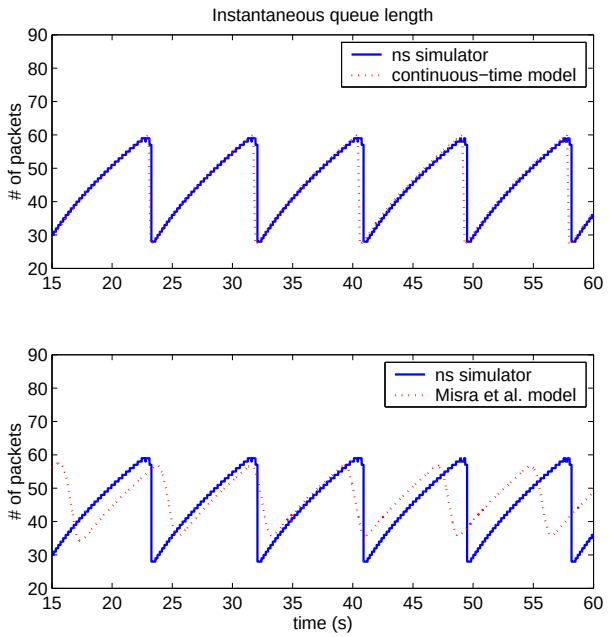

Figure 2. Comparison of models with ns simulator for the one-sender scenario described in Sec. 2.4.

retransmitting to reflect the fact that the packet bursts from each sender are interspersed.

2.4 Results

In this section we compare results obtained from applying our model to the network set-up mentioned above with those obtained using the ns simulator on an equivalent network. We used the following settings (refer to Sec. 2 for variable definitions): $a = .01$ s, $c = 1.5$ Mbps, $q_{min} = 50$, $q_{max} = 100$, $p_{max} = .1$, $w = .003$, and packet size $= 1000$ bytes.

We implemented the model in MATLAB. Since we do not model the slow-start behavior of TCP, we begin all of our comparisons between the simulator and model after an initial transient has ended. Fig. 2 shows comparisons of the queue length in the continuous-time model and the Misra et al. model [4] with the ns network simulator [14]. Despite the lack of a stochastic component as well as several of the details of the TCP implementation, our model is able to very closely reproduce the behavior of the simulator. We believe that the model has captured the essential behavior of this network under these flow conditions. The use of an impulse helps the model account for the sharp declines in the queue length caused by drop events. Lacking this feature, the fluid model of Misra et al. does not perform as well in this case of one sender.

However, the accuracy of this approach diminishes as the number of senders increases. For larger numbers of senders the model of Misra et al. shows better agreement with the simulator than our model does. Thus our model in this form is advantageous mainly in cases when the bottleneck link congestion is due to a small number of senders. To improve our model's performance, we introduce a more accurate mechanism for determining dropped packets in the following section. In particular, we fully implement RED as it is executed in the ns simulator.

3 Extension - Full Red Implementation

Model simplicity is important if we are interested in being able to concisely describe a system, and for amenability to mathematical analysis. On the other hand, for the purpose of traffic state estimation, the key attributes we desire in a model are accuracy and computational efficiency. With this in mind, we propose an extension to the model to improve its accuracy while still keeping the model simpler than a full-fledged packet-level simulator. The primary modification we consider is to implement a more realistic packet dropping mechanism within the model, one that more closely resembles RED.

In order to fully model RED, we begin by following the procedure laid out in the previous section up to equation (7). At that point, a uniform random number between 0 and 1 is generated and if this number is less than p_{drop}, a drop occurs. In the case of multiple senders, we let the probability of a given sender experiencing the drop be proportional to that sender's share $(\frac{W_i}{R_i})/(\sum_{j=1}^{N} \frac{W_j}{R_j})$ of the overall flow. This introduces a greater degree of complexity and a stochastic component to the model.

We applied this model to the network described earlier, but now with four senders at varying fixed propagation delays of 5, 20, 50 and 110 ms from the receiver. Fig. 3 shows that there is good qualitative agreement between the model and the simulator. The additional jitter in the simulator's queue is due to packet level activity. The model shows good statistical agreement with the simulator as well. Fig. 4 shows a comparison of queue histograms. Note that the model correctly captures the range in queue values, which can be used to calculate the range of round-trip times using equation (1).

In order to evaluate the model's responsiveness to network changes, we tested it on a network with two classes of flows, one of which turns off for part of the run. The classes both consist of five bulk transfer senders, but the fixed propagation delay is different for each class: 20 ms for class 1 senders, 35 ms for class 2 senders. The class 2 senders turn off from 75 to 125 seconds after the start of the run. In practice, this effect could be caused by the sender-side application not having data to send for that stretch of time. Another possibility could be that the class 2 senders are temporarily rerouted.

The results, shown in Fig. 5, indicate that the model does a good job of capturing the changes in the network state caused by the senders turning off. This includes the sudden drop in the queue just after the senders turn off. The model also reproduces the overall decreased level of

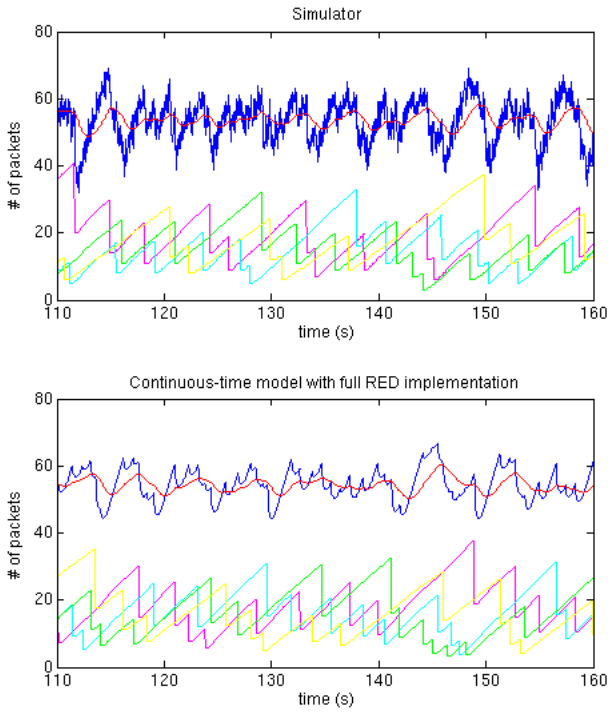

Figure 3. Comparison for four senders between the ns simulator and a realization of the continuous-time model using the full RED implementation. In both plots, the upper two curves represent the instantaneous and exponentially averaged queues. (The exponentially averaged queue is the curve showing less variation.) The lower four curves in both plots represent the window sizes of the four senders.

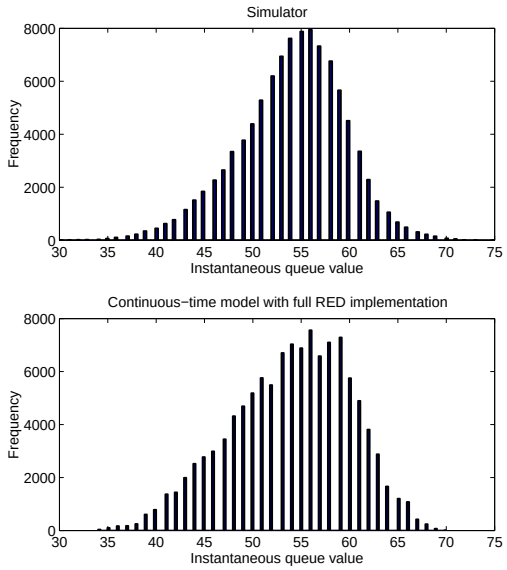

Figure 4. Comparison of instantaneous queue histograms for the four-sender network. The model is using the full RED implementation. Model queue values, which may be fractional, have been rounded for the purposes of comparison.

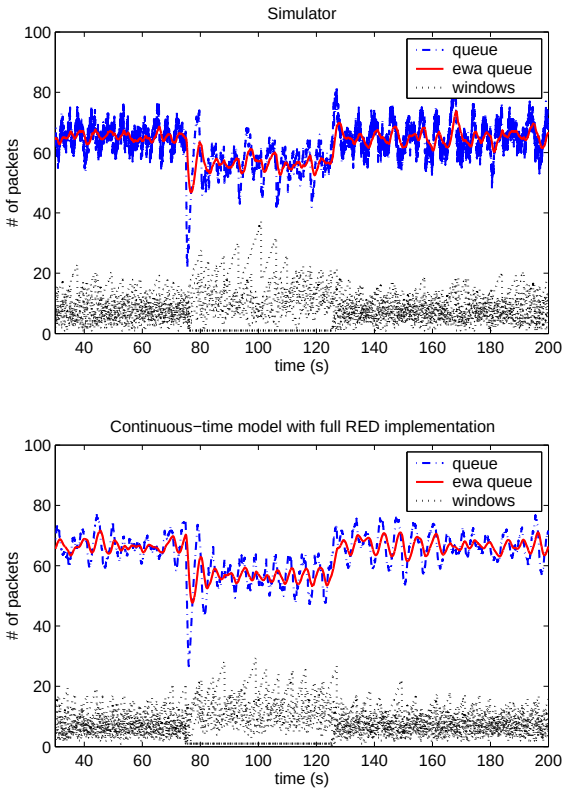

Figure 5. Comparison for two classes with five senders per class. Class 2 senders are off from 75 to 125 seconds.

the queue and increased level of the class 1 senders' windows during the class 2 off period. Since both simulator and model have stochastic components, each plot represents a realization rather than the expected performance. Nevertheless, based on our observations of many realizations for both model and simulator, we expect the average performance to show good correspondence as well. For the time being, we provide typical statistics from sample realizations in Table 1, indicating the close statistical match between model and simulator. Note that the model is able to estimate the variance of the queue to within 10%. This implies that by using the model in conjunction with the round-trip time equation (1), a reasonable estimate of the variance of the round-trip time can be obtained.

Table 1. Statistical Comparison for Two-Class Scenario

Variable	Quantity	Simulator	Model
All Flows On			
q	mean	65.5	66.7
q	st.dev.	4.8	4.4
Half of the Flows Off			
q	mean	56.0	56.0
q	st.dev.	7.0	6.5

4 Towards Network State Estimation

Ultimately, we plan to use the model in a network traffic state estimation scheme. The estimation problem can be posed as follows: Consider a network of N senders and one bottleneck router, whose state can be characterized by the sender window sizes and the router queue and exponentially averaged queue lengths. Given only a series of observations of some subset of the full network state, such as the window sizes and round-trip times of a small number of senders, what is our best estimate of the state of the system at the current time? That is, we must estimate all of the sender window sizes, using only the past history of some observed portion of the state of the network.

The estimation scheme we have in mind can be described as a particle filter or, more generally, a sequential Monte Carlo method [15, 16]. The basic idea is related to that of a Kalman filter. But unlike the Kalman filter, the particle filter makes no assumptions about the linearity of the model, or about the probability distributions being Gaussian. Thus it is a better fit for our model and system.

The particle filter starts off with a random ensemble of states. It uses the observation to filter and re-weight the ensemble members based on how consistent they are with the observation. Then the ensemble members are advanced in time using the model, and the filter and re-weighting process repeats. We are currently testing this procedure in a variety of network scenarios.

5 Conclusion

We have developed a deterministic, continuous-time model with a discrete impulse that successfully captures the network behavior of TCP/RED in simple cases. Extending the model by adding a stochastic dropping mechanism consistent with RED, the model shows good correspondence in more complicated network situations, including those with larger numbers of senders and flows turning on and off.

Aside from network traffic state estimation described in Sec. 4, other works in progress include modeling networks with multiple bottlenecks and cross-traffic, consideration of the physical layer, and comparisons of the model to real networks.[2]

References

[1] A. Medina, M. Allman and S. Floyd, Measuring the Evolution of Transport Protocols in the Internet, preprint, available from http://www.icir.org/tbit.

[2] M. Fomenkov, K. Keys, D. Moore and k claffy, Longitudinal Study of Internet Traffic in 1998-2003, Technical report, Cooperative Association for Internet Data Analysis (CAIDA), 2003.

[3] S. Floyd and V. Jacobson, Random Early Detection Gateways for Congestion Avoidance, *IEEE Trans. on Networking*, 1(7), 1993, 397-413.

[4] V. Misra, W. Gong, D. Towsley, Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED, *Proc. of SIGCOMM 2000*, Stockholm, Sweden, 2000, 151-160.

[5] V. Firoiu and M. Borden, Study of Active Queue Management for Congestion Control, *Proc. of Infocom 2000*, Tel Aviv, Israel, 2000, 1435-1444.

[6] P. Ranjan, E. Abed and R. La, Nonlinear Instabilities in TCP-RED, *Proc. of Infocom 2002*, New York, USA, 2002, 249-258.

[7] S.H. Low, F. Paganini, J. Wang, S. Adlakha and J. Doyle, Dynamics of TCP/RED and a Scalable Control, *Proc. of Infocom 2002*, New York, USA 2002, 239-248.

[8] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm, *Computer Communications Review*, 27(3), 1997, 67-82.

[9] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," *Proc. of SIGCOMM 1998*.

[10] J. Hespanha, S. Bohacek, K. Obraczka and J. Lee, Hybrid Modeling of TCP Congestion Control, *Lecture Notes in Computer Science*, no. 2034, 2001, 291-304.

[11] S. Bohacek, J. Hespanha, J. Lee and K. Obraczka, Hybrid Systems Modeling Framework for Fast and Accurate Simulation of Data Communication Networks, *Proc. of SIGMETRICS 2003*, San Diego, USA, 2003, 58-69.

[12] Y. Liu, F. Lo Presti, V. Misra, D. Towsley and Y. Gu, Fluid Models and Solutions for Large-Scale IP Networks, *Proc. of SIGMETRICS 2003*, San Diego, USA, 2003, 91-101.

[13] I. Frommer, E. Harder, B. Hunt, R. Lance, E. Ott and J. Yorke, Two Models for the Study of Congested Internet Connections, arXiv:cs.NI/0409022.

[14] http://www.isi.edu/nsnam/ns/

[15] N. Gordon, D. Salmond and A. Smith, Novel approach to nonlinear/non-Gaussian Bayesian state estimation, *IEE Proceedings-F*, 140(2), 1993, 107-113.

[16] A. Doucet, N. de Freitas, N. Gordon, eds., *Sequential Monte Carlo methods in practice* (New York: Springer, 2001).

[2] This work is part of the first author's Ph.D. dissertation.