Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED *

Vishal Misra
Department of Computer Science
University of Massachusetts
Amherst MA 01003
misra@cs.umass.edu

Wei-Bo Gong
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst MA 01003
gong@ecs.umass.edu

Don Towsley
Department of Computer Science
University of Massachusetts
Amherst MA 01003
towsley@cs.umass.edu

ABSTRACT

In this paper we use jump process driven Stochastic Differential Equations to model the interactions of a set of TCP flows and Active Queue Management routers in a network setting. We show how the SDEs can be transformed into a set of Ordinary Differential Equations which can be easily solved numerically. Our solution methodology scales well to a large number of flows. As an application, we model and solve a system where RED is the AQM policy. Our results show excellent agreement with those of similar networks simulated using the well known `ns` simulator. Our model enables us to get an in-depth understanding of the RED algorithm. Using the tools developed in this paper, we present a critical analysis of the RED algorithm. We explain the role played by the RED configuration parameters on the behavior of the algorithm in a network. We point out a flaw in the RED averaging mechanism which we believe is a cause of tuning problems for RED. We believe this modeling/solution methodology has a great potential in analyzing and understanding various network congestion control algorithms.

1. INTRODUCTION

Active queue management techniques have recently been proposed [8], [3] to to both alleviate some congestion control problems for IP networks as well as provide some notion of quality of service. Modeling and analysis of such networks is important to understand their dynamics. While traditional discrete event simulations work well in general, even the most efficiently coded simulators suffer from the problem of scaling. In this paper, we exploit fluid modeling to present a general methodology for the analysis of a network of routers supporting active queue management with TCP flows. We model the data traffic as a fluid and specifically use Poisson Counter Driven Stochastic Differential Equations to model sample path description of TCP traffic. We also derive a set of differential equations that describe the AQM policy and the router queueing process. Next, we develop a numerical scheme for obtaining the transient average behavior of a number of metrics including queue length, round trip time and TCP flow throughput from a set of coupled ordinary differential equations that result from our analysis. Given an AQM policy, we are able to get (expected) transient behavior of networks from our solution. We are able to handle large flows without a significant increase in computational complexity.

In order to illustrate the advantages of our technique, we consider RED [8], one of the most popular AQM schemes to address the congestion control issues. We show that our solution technique yields predictions that match well with those obtained with the `ns` simulator. We are able to make some critical comments about RED. Our modeling and solution methodology lead to a straightforward discovery of a critical problem with the RED averaging mechanism, one which we believe has not been addressed elsewhere. Our scheme has similar aims, to obtain transient behavior by numerical solution of a system of equations, to a heuristic approach proposed in [9]. It is however not clear how and why the heuristics work in cases reported.

The rest of this paper is organized as follows. In Section 2, we develop our analytical model and describe the solution technique. In Section 3, we consider an application to our techniques in a setting with RED as the AQM policy. We compare our results with those obtained using the well known `ns` simulator and make some observations about RED behavior. Finally, we present our conclusions in Section 4.

2. MODEL AND ANALYSIS

In [11], we modeled the behavior of TCP using jump process driven Stochastic Differential Equations [2]. The results showed a good match between predictions from this model and measurements reported in [13]. A deficiency of the model however was that the packet loss process was independent of the data flow. Our model remedies this deficiency by modeling a *complete* system, in which losses and TCP sending rates are closely coupled. Thus we have a closed loop control system [1] giving rise to a set of coupled differential equations. We will begin with a single router in Section 2.1 and then show how the model and analysis are extended to a network in Section 2.2. In Section 2.3 we extend the techniques to include TCP timeouts. Last, we describe some optimizations in the case of identical TCP flows in Section 2.4.

*This work is supported in part by the National Science Foundation under Grants ANI-9809332, by EPRI/DoD under contract W08333-03, and by DARPA under Contract DOD F30602-00-0554.

SIGCOMM'00, Stockholm, Sweden.

[1] Such a system was alluded to in [6].

We first consider a system in which there is a single congested router with a transmission capacity of C. Associated with this router is an active queue management (AQM) policy that is characterized by a packet discard function $p(x)$ that takes as its argument an *estimate* of the average queue length at the router. The queue length of the router is denoted by $q(t)$, $t \geq 0$. The classical example of an AQM policy is RED [8] for which $p(x)$ takes the form

$$p(x) = \begin{cases} 0, & 0 \leq x < t^{min} \\ \frac{x - t^{min}}{t^{max} - t^{min}} p^{max}, & t^{min} \leq x \leq t^{max} \\ 1, & t^{max} < x \end{cases} \quad (1)$$

where t^{min}, t^{max}, and p^{max} are configurable parameters. The drop function is depicted in Figure 1, showing the discontinuity at t^{max}.

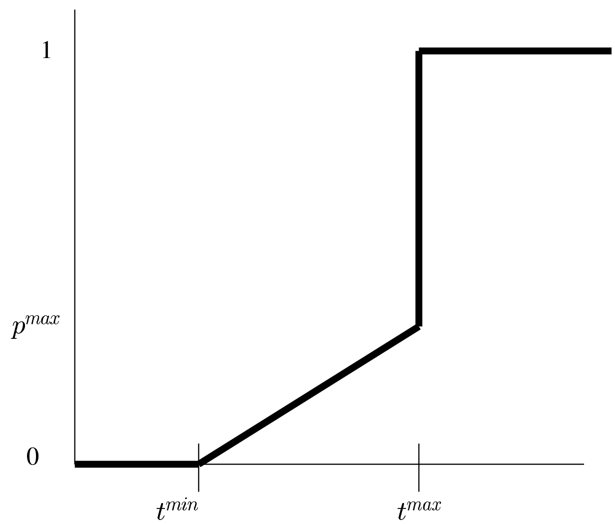


Figure 1: RED drop function

2.1 A single congested router

Let N TCP flows labeled $i = 1, \ldots N$ traverse the router. Let $W_i(t)$ and $R_i(t)$ denote the TCP window size and round trip time at time $t \geq 0$, of flow i, $i = 1, \ldots, N$. We assume that $R_i(t)$ takes the form

$$R_i(t) = a_i + q(t)/C, \quad t \geq 0; i = 1, \ldots, N \quad (2)$$

Where a_i is a fixed propagation delay and $q(t)/C$ models the queueing delay.

Last, let $B_i(t)$ denote the instantaneous throughput of TCP-flow i at time $t \geq 0$.

Our model is based on the assumption that packet losses to flow i are described by a Poisson process $\{N_i(t)\}$ with time varying rate $\lambda_i(t)$. The time varying nature of $\lambda_i(t)$ is able to model the independent marking schemes commonly found in AQM. Here $N_i(t)$ denotes the number of losses suffered by flow i. Note that t here denotes the point in time when the flow detects losses, which is different from when the actual dropping at the queue occurs. Henceforth, when there is no ambiguity, we will omit the argument t. A short description of Poisson Counter Driven Stochastic Differential equations is included in an Appendix for completeness. We have the following equation describing the behavior of the window size $W_i(t)$,

$$dW_i(t) = \frac{dt}{R_i(q(t))} - \frac{W_i(t)}{2} dN_i(t) \quad (3)$$

This models the additive-increase multiplicative-decrease behavior of TCP. The first term corresponds to the additive increase part, which says that the window size will increase by one every round trip time. The second term corresponds to the multiplicative decrease part, which halves the window size at the instant of the arrival of a loss ($dN_i(t) = 1$). We model the traffic as a fluid, making the instantaneous throughput $B_i(t)$ equal to $W_i(t)/R_i(t)$. Taking the expectation[2] of each side of the above equation yields

$$\begin{aligned} E[dW_i(t)] &= E\left[\frac{dt}{R_i(q(t))}\right] - \frac{E[W_i(t) dN_i(t)]}{2} \\ dE[W_i] &\approx E\left[\frac{dt}{R_i(q)}\right] - \frac{E[W_i]\lambda_i(t)}{2} dt \end{aligned}$$

The above equation is approximate because we have broken down $E[W_i(t)dN_i(t)]$ as $E[W_i(t)]E[dN_i(t)]$, which assumes independence between the two which is not true, especially in proportional marking schemes. However, this approximation does not change the fundamental nature of the multiplicative decrease mechanism, and we are able to capture TCP dynamics. Now $\lambda(t)$ is the loss indication received by the source. It reaches the source approximately one round trip delay (τ) after a packet has been marked/dropped at the queue. In [14] an exact stochastic differential equation for TCP with feedback delay has been studied, in the context of fairness. Here we model the delay τ as the solution to the following equations

$$\begin{aligned} t &= R(q(t')) + t' \\ t` &= t - \tau \end{aligned}$$

In the proportional marking schemes employed in AQMs, marking/dropping is implemented to distribute the losses in proportion to a flows bandwidth share. Thus, if the throughput of a flow is $B(t-\tau)$ at the time $t-\tau$, the rate of loss indications it receives at time t is $p(\bar{x}(t-\tau)B(t-\tau)$ ($\lambda(t)$. We denote the expected value of it by $p(\bar{x}(t-\tau)E[W_i(t-\tau)/R_i(q(t-\tau)$.

Now, returning to our earlier differential equation equation describing the expected window size, we have

$$\begin{aligned} dE[W_i] &\approx \frac{dt}{R_i(\bar{q})} - \frac{\bar{W}_i}{2} p(\bar{x}(t-\tau)) \frac{\bar{W}_i(t-\tau)}{R_i(\bar{q}(t-\tau))} dt \\ &= \frac{dt}{R_i(\bar{q})} - \frac{\bar{W}_i \bar{W}_i(t-\tau)}{2R_i(\bar{q}(t-\tau))} p(\bar{x}(t-\tau)) dt \end{aligned}$$

The approximation that we made in the step above involved making the approximation $E[f(x)] \approx f(E[x])$. We'll comment more on this in a later section. Thus, we have,

$$\frac{d\bar{W}_i}{dt} = \frac{1}{R_i(\bar{q})} - \frac{\bar{W}_i \bar{W}_i(t-\tau)}{2R_i(\bar{q}(t-\tau))} p(\bar{x}(t-\tau)) \quad (4)$$

We assume that the estimate of the average queue length is an exponentially weighted moving average based on samples taken every δ

[2]Throughout this section we represent the expected value of any variable ν by $\bar{\nu}$

seconds Using a weight α, $0 < \alpha < 1$,

$$x((k+1)\delta) = (1-\alpha)x(k\delta) + (\alpha)q(k\delta) \tag{5}$$

It is useful to convert this equation into a differential equation. Given the form (5), the natural candidate is

$$\frac{dx}{dt} = Ax(t) + Bq(t)$$

Then, in a sampled data system [1], $x(t_{k+1})$ is given by

$$x(t_{k+1}) = e^{A(t_{k+1}-t_k)}x(t_k) - \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\tau)}Bd\tau q(t_k) \tag{6}$$

Note that the differential equation matches the discrete time system *exactly* at the sample points, and there is no accumulation of error as k increases. Comparing the coefficients in (6) and (5), we obtain

$$1-\alpha = e^{A\delta}$$

or

$$A = \frac{\log_e(1-\alpha)}{\delta} = -B$$

Thus, we have the following equation describing the behavior of x,

$$\frac{dx}{dt} = \frac{\log_e(1-\alpha)}{\delta}x(t) - \frac{\log_e(1-\alpha)}{\delta}q(t)$$

As this is a linear equation, taking expectation of both sides yields

$$\frac{d\bar{x}}{dt} = \frac{\log_e(1-\alpha)}{\delta}\bar{x}(t) - \frac{\log_e(1-\alpha)}{\delta}\bar{q}(t) \tag{7}$$

Finally, we have the following equation describing the behavior of q, which is the differential version of the Lindley equation

$$\frac{dq(t)}{dt} = -1_{q(t)}C + \sum_{i=1}^{N}\frac{W_i}{R_i(q)} \tag{8}$$

Here, the first term models the decrease in the queue length, when it is greater than zero, due to the servicing of packets. The second term corresponds to the increase in the queue length due to the arrival of packets from the N TCP flows. Again, taking expectations, we obtain

$$\begin{aligned}\frac{d\bar{q}(t)}{dt} &= E\left[-1_{q(t)}\right]C + \sum_{i=1}^{N} E\left[\frac{W_i}{R_i(q)}\right] \\ &\approx E\left[-1_{q(t)}\right]C + \sum_{i=1}^{N}\frac{\bar{W}_i}{R_i(\bar{q})}\end{aligned}$$

Now for a bottlenecked queue, we'll have $q(t) > 0$ with probability close to 1. Thus, we can approximate$E\left[1_{q(t)}\right]$ to obtain

$$\frac{d\bar{q}(t)}{dt} \approx -C + \sum_{i=1}^{N}\frac{\bar{W}_i}{R_i(\bar{q})} \tag{9}$$

We have $N+2$ coupled equations (4), (7), and (9) and $N+2$ unknowns, $(\bar{x}, \bar{q}, \bar{W}_i)$ which can be solved numerically. The solution provides an estimate of the average transient behavior of the system. We get the queue length, queue estimate and window size evolution directly, and those values in turn yield average round trip delay, average loss rate etc.

2.2 Extension to a network

The extension to the network case is straightforward. Let V be a collection of communication routers. Each router $v \in V$ has a transmission capacity of C_v bits per second. Router v has an AQM policy characterized by a probability discard function, $p_v(x_v)$, which takes as its argument x_v, the estimated average queue length of v. The queue length for router v is $q_v(t)$. Last, let $\mathbf{x}(t)$ and $\mathbf{q}(t)$ be vectors whose components are the estimated average queue lengths and queue lengths, respectively, at time $t \geq 0$. Associated with each router is the sampling period δ_v and averaging weight α_v.

Consider a workload of N TCP flows labelled $i = 1, \ldots, N$. Let $V_i = (j_{i,1}, j_{i,2}, \ldots, j_{i,n_i})$ be the ordered set of links (i.e., path) taken by packets from flow i, where $j_{i,m} \in V$, $m = 1, \ldots, n_i$ and n_i is the path length. Generalizing (2), the average round trip time of session i is approximated by:

$$R_i(\mathbf{q}, t) = a_i + \sum_{v \in V_i} q_v(t)/C_v \tag{10}$$

The routes can be represented by a binary matrix $\mathbf{A}$ where the rows represent the different flows and the columns represent the different routers (queues). In other words, $\mathbf{A}_{i,j} = 1$ iff $j \in V_i$. We modify equation (4) to account for losses arriving from each router in the path. If $\mathbf{P}(\mathbf{x})$ is the vector of loss probabilities at each node, then we define a matrix $\mathbf{AP}$ where we multiply every column of the $\mathbf{A}$ matrix by the corresponding element of the $\mathbf{P}$ vector. The combined loss seen by a particular flow i is, then, given by $1 - \prod(1 - \mathbf{AP}(x)_i)$ Thus, (4) is modified to

$$\begin{aligned}\frac{d\bar{W}_i}{dt} &= \frac{1}{R_i(\bar{q}(t-\tau))} \text{-} \left(\frac{\bar{W}_i\bar{W}_i(t-\tau)}{2R_i(\bar{q}(t-\tau))}\right) \cdot \\ &\quad (1 - (\prod(1 - \mathbf{AP}(\mathbf{x})_i))) \\ &= \frac{1}{R_i(\bar{q}(t-\tau))} \text{-} \left(\frac{\bar{W}_i\bar{W}_i(t-\tau)}{2R_i(\bar{q}(t-\tau))}\right) \cdot \hat{p}_i(\bar{x}(t-\tau))\end{aligned}$$

where $\hat{p}_i(\bar{x})$ is the probability of packet loss experienced by flow i on its path. The equation for the estimated average queue length and the queue length at each node remain unchanged. Define F_ν as the set of flows through queue v. Then

$$\frac{d\bar{q}_v(t)}{dt} = -1_{\bar{q}_v(t)}c + \sum_{i \in F_v}\frac{\bar{W}_i}{R_i(\bar{q}_v)} \tag{11}$$

In the networked case, for any queue which is a bottleneck, $q_\nu(t) > 0$ with probability close to 1, and for a non-bottlenecked queue, $q_\nu(t) > 0$ with probability close to 0. This is the basis for the approximation that $E[1_{q_v(t)}] \approx 1_{\bar{q}_v}$. We end up with a system of equations with $N + 2|V|$ unknowns that can be solved numerically to yield the transient behavior of the network.

2.3 Modeling Timeouts

To account for timeouts, we need to quantify the division of losses into the two types, viz. timeout (TO) and triple duplicate ack (TD). Let $Q(w)$ denote the the probability that the loss is a TO loss, given that the window size at the time of a loss is w. We use the simplified function $Q(w) = \min(1, 3/w)$ derived in [13], for this purpose. Intuitively, this expression for Q is based on the assumption that all packets in a particular round are equally likely to be dropped, with at most one drop per round. In that case, any one of the last 3 packets in a round can cause a timeout if dropped, hence the function $\min(1, 3/w)$. Equation (4) can be modified to account for TO losses resulting in

$$\frac{d\bar{W}_i}{dt} \approx \frac{1}{R_i(q)} + (1 - Q(\bar{W}_i))(-\frac{W_i \bar{W_i(t-\tau)}}{2R_i(\bar{q}(t-\tau))})\bar{p}_i(\bar{x}(t-\tau)) + (1-\bar{W}_i)\frac{\bar{W}_i(t-\tau)}{R_i(\bar{q}(t-\tau))}Q(\bar{W}_i)\bar{p}_i(\bar{x}(t-\tau))$$

2.4 Aggregation of identical flows

The system of equations describing the transient behavior of a network of TCP flows can be simplified in the presence of identical flows, i.e., those with the same route and round trip time. This is done by representing identical flows by a single class. Let there be F classes of flows, where the j-th class contains n_j identical flows with route V_j and round trip time $R_j(\mathbf{q})$. All flows in the same class will have the same average behavior. Thus, we can represent the behavior of the average window size for each flow using equation (9). We suitably modify the equation for the average queue size in the following manner

$$\frac{d\bar{q}_\nu(t)}{dt} = -1_{\bar{q}_\nu(t)}C_i + \sum_{j \in F_\nu} n_j \frac{\bar{W}_j}{R_j(\bar{q}_v)} \qquad (12)$$

This formulation leads to a considerable savings in computation time when we have a large number of identical flows to solve for. The number of equations and unknowns reduces from $\sum_{i=1}^{F} n_i + 2|V|$ to $F + 2|V|$.

It is also worth noting that, as the number of flows in a class increase, the law of large numbers comes into play and the expected behavior begins to approach the aggregate sample path behavior. Thus, the stochastic differential equations start converging to ordinary differential equations.

3. AN APPLICATION TO THE RED ACTIVE QUEUE MANAGEMENT POLICY

We now present an application of the system, taking RED as the AQM policy RED has been shown to outperform Drop-Tail queues under certain scenarios. RED is a powerful mechanism to control traffic, potentially solving problems like flow synchronization, correlation of drop events while providing consistently high link utilizations. However, numerous problems have been cited with RED [10]. RED works well in certain scenarios, whereas it does very poorly, even worse than Drop-Tail, in other cases [6]. There is no clear understanding on how to tune various RED parameters that work well in all scenarios. Consequently, there is considerable nervousness in the community regarding deployment of RED, and numerous variations of RED have been proposed [5], [12], [4]. Some of the schemes have self-tuning parameters, while others maintain per-flow state. It is clear that there is a great need for thoroughly understanding the behavior of RED. We believe our techniques can help in that effort.

Throughout this section we will compare the results obtained from our model to those obtained from simulating an equivalent system using the well known `ns` simulator. To avoid confusion between the process of averaging by RED, and our operation of obtaining expected values using differential equations, we denote the average queue length (as it is more commonly referred) as calculated by the `ns` implementation of RED as Queue estimate (x) and the result obtained by our differential equation solution ($\bar{x}$) as average Queue estimate. Similarly, $\bar{q}$ is referred to as the average Instantaneous Queue length and q as reported by `ns` as instantaneous queue length. Our differential equation solver was implemented as a MATLAB program (a simple implementation with 42 total lines of code) which takes the matrix **A** and a link capacity vector **C** as input. We did not incorporate slow-start in our program. All routers are assumed to have the same RED parameters . The propagation (non-queueing) delay for each class of flows is kept at 200 ms. The buffers are sized so that all losses are RED-related, i.e. no drop-tail losses occur on the network.The t^{min} is 150 packets and the t^{max} is 200 packets. δ is a parameter in our solver which is not specified in RED. Instead RED updates the queue size estimate on the arrival of each packet. We can account for it in two different ways

- `ns` updates the queue at every packet arrival. Thus, we can choose δ_v to be $1/b_v$, where b_v is the instantaneous arrival rate at a queue measured in packets. δ_ν thus becomes a function of time.

- We can choose a fixed value of δ_v. If the queue is stable, then the steady state arrival rate = service rate. Then, for each (bottlenecked) queue, δ_v is simply $1/C_v$ where C_v is the capacity of link v in terms of (average) sized packets per second.

Note that both techniques are approximate, in our implementation we use $1/C_v$ as an estimate of δ_v.

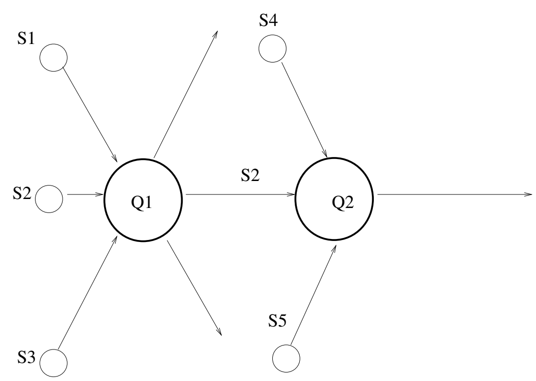


Figure 2: Simple network topology

3.1 Experiment topology

We use the simple topology shown in Figure 3. It consists of two RED queues $Q1$ and $Q2$. Both links have three sets of flows going through them. $S2$ goes through both the queues, whereas $S1$, $S3$ and $S4$, $S5$ go through only $Q1$ and $Q2$ respectively. The only bottleneck links are the queues $Q1$ and $Q2$. We'll show results from 5 different experiments performed using `ns` and our DE solver. We tabulate the parameter choices in the various experiments in the following table for reference

Parameter sets for various experiments

Exp. No.	α	Q1 cap.	Q2 cap.	Pkt size	p_{max}
1	0.0001	5 Mb/s	5 Mb/s	500 Bytes	0.1
2	0.0001	5 Mb/s	2.5 Mb/s	500 Bytes	0.1
3	0.0001	15 Mb/s	15 Mb/s	500 Bytes	0.1
4	0.0001	15 Mb/s	15 Mb/s	1500 Bytes	0.1
5	0.0001	15 Mb/s	15 Mb/s	500 Bytes	1

3.2 Experiment 1

We first consider a symmetric case, where both RED queues have similar bandwidth capacity of 5Mb/s. α is kept at 0.0001, $p^{max} = 0.1$. Each class of flows consists of 40 individual flows which start at $t = 0$ (200 flows in all). At time $t = 75$, three fourths of the flows in each class drop out (so there are only 10 flows in each class). At time $t = 150$, those flows restart. We plot the Queue estimate and instantaneous queue length for Queue 1 and 2 in Figures 3, 4 respectively along with our model predictions of the expected values for the same. Note that we are plotting the results of *one* `ns` simulation along with our solution which gives the *expected* results. As we can see, the differential equation solution tracks the simulations pretty well, tracking both the average queue estimate and the instantaneous queue length well. Our method adapts to the changing nature of the load as well. Note that our solution differs initially from the simulation in all cases, because we didn't implement the effect of slow start. We'll return to this discrepancy in a later section.

3.3 Experiment 2

Now we repeat the experiment in an asymmetric setting. We reduce the link capacity of the second queue to be 2.5Mb/s from 5Mb/s. Again we show the average queue estimate and instantaneous queue length along with our DE estimates for both the queues in Figure 5 and our results match well with `ns` simulations for both the queues. Notice that the average queue estimate stays higher for Queue 2 which is correctly reflected in our DE solution.

3.4 The importance of δ

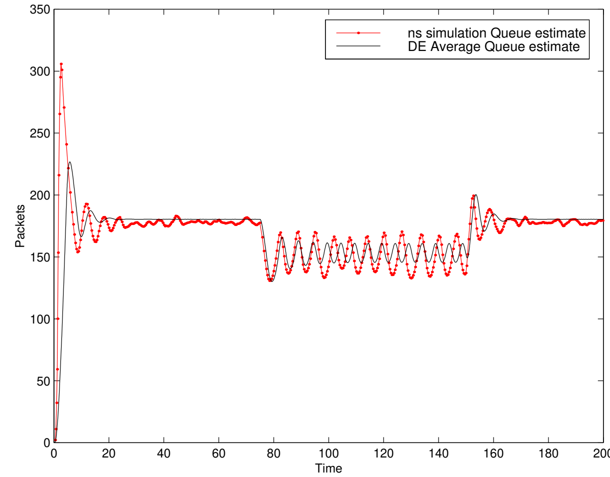
If we focus on in the middle portion of Figure 4 and 5 (when only a fourth of the flows are active), we observe that Queue 1 exhibits more oscillations than Queue 2. The oscillations are not good for the network as they may result in unacceptably large queue lengths and hence a large variability in delays for the flows going through. Even if the mean delay may turn out to be the same, these oscillations add considerable jitter to the delays. If the buffer is not large enough, then the effect of the oscillations will be to cause buffer overflows. They also cause periodically high RED loss rates and affect the throughput adversely. The question is: why? The larger bandwidth capacity of Queue 1 certainly plays a part, reducing stability margins, however we would like to point out another, hidden, cause. This is done by revisiting equation (7):

$$\frac{dx}{dt} = \frac{\log_e(1-\alpha)}{\delta}x(t) - \frac{\log_e(1-\alpha)}{\delta}q(t)$$

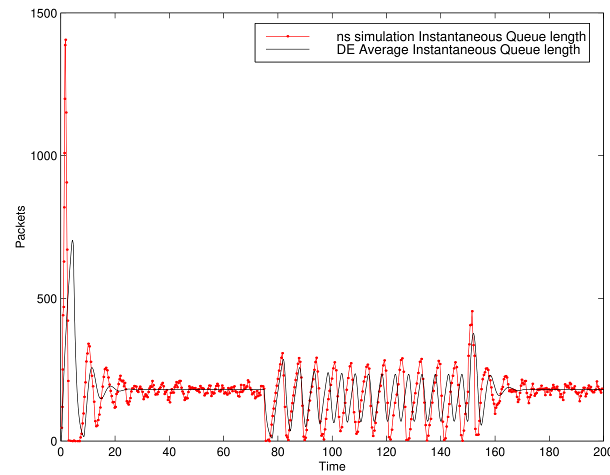
Denote $\log_e(1-\alpha)/\delta$ by $-K$ (since $\log_e(1-\alpha)$ is a negative quantity). Then we have

$$\frac{dx}{dt} = -Kx(t) + Kq(t)$$

Taking the Laplace transform of both sides, we get the transfer function of the queue averaging module as $\frac{K}{K+s}$. This is the first order low pass filter which was the original design goal of RED, to track the average queue size (low frequency signal), and to filter out bursts (high frequency signal). The input to this filter is the instantaneous queue length, and the output is the average queue estimate. Asymptotically, the frequency response of this filter is described by the magnitude Bode plot shown in Figure 7. It allows frequencies smaller than K to pass through, while damping inputs at a frequency higher than K. In simple terms, K determines the responsiveness of the filter. The higher the value of K, the faster it will respond to a sudden change. If we maintain a high value of K, then the AQM function starts tracking the instantaneous queue length closely resulting in sustained oscillations. Let's perform experiment 3, with all settings unchanged except with both link capacities set at 15 Mb/s, and Figure 8 illustrates the Queue estimates and Instantaneous Queue Lengths for Queue 1 in the time interval [0 75] (i.e. when all flows are active). We observe the presence of large oscillations. In the scenario where the link capacity was 5Mb/s and the packet size 500 Bytes, the effective sampling period δ was 8×10^{-4} (the link capacity is 1250 packets of 500 bytes per

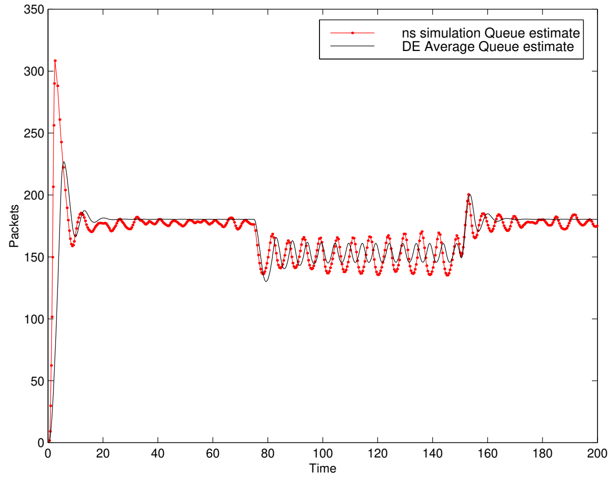


(a) Queue estimate

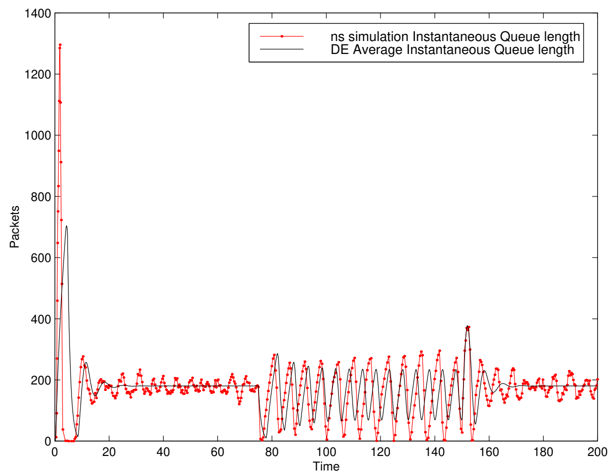


(b) Instantaneous Queue Length

Figure 3: Symmetric case, Plots for Queue 1, Experiment 1

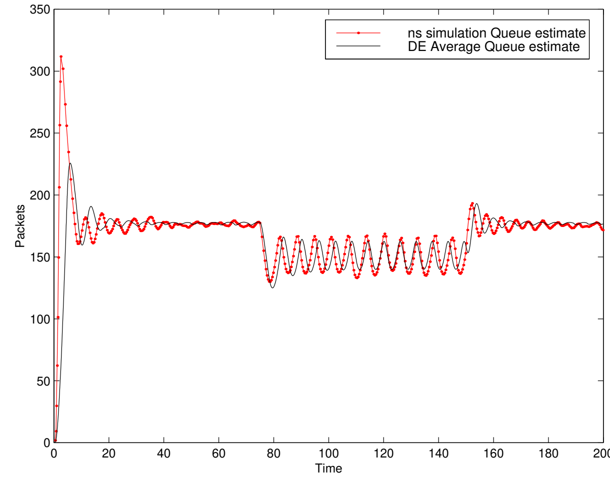


(a) Queue estimate

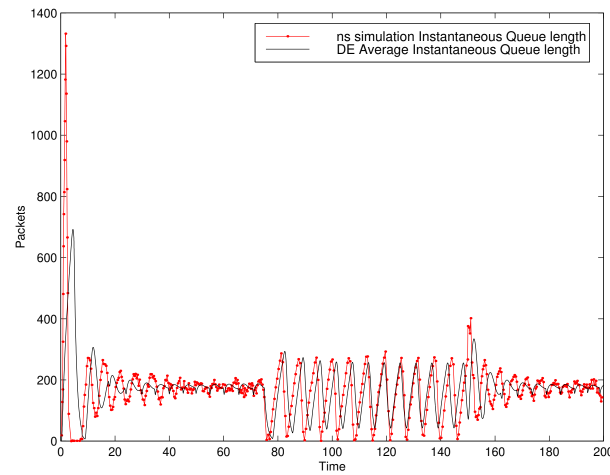


(b) Instantaneous Queue Length

Figure 4: Symmetric case, Plots for Queue 2, Experiment 1



(a) Queue estimate



(b) Instantaneous Queue Length

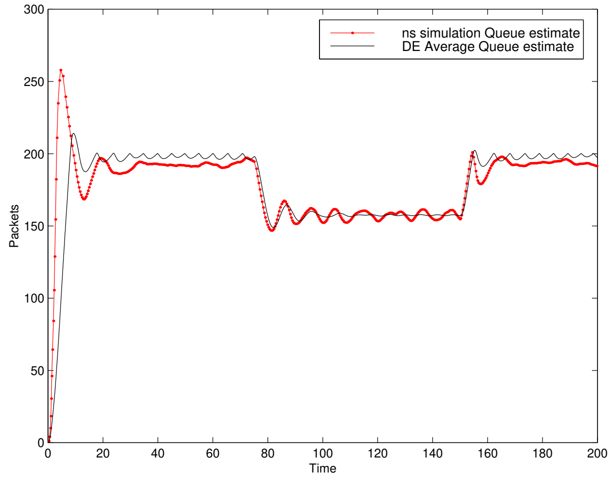
Figure 5: Asymmetric case, Plots for Queue 1, Experiment 2

second, $1/C$ gives δ). With $\alpha = 0.0001$,

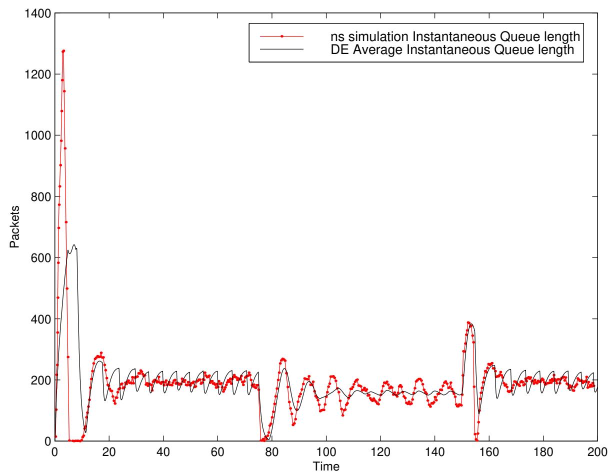
$$\begin{aligned} K &= -\frac{\log_e(1-\alpha)}{\delta} \\ &= -\frac{\log_e(1-0.0001)}{8 \times 10^{-4}} \\ &= 0.1250 \end{aligned}$$

With the increase in link capacity to 15Mb/s, the value of δ reduced to 2.66×10^{-4} and K becomes .3750 (3 times the earlier value). As the link capacity increases, the RED average queue estimate tracks the instantaneous queue length more closely, essentially resulting in sustained oscillations. This hidden artifact of the RED algorithm, the *adaptive* nature of δ, is, in our opinion, a significant cause of the "tuning problem" with RED. We can modify our DE solution system by using a static value of δ, larger than $1/C$ with the hope of reducing K and thereby reducing oscillations. Figure 9 illustrates the behavior of the Queue Estimate and Instantaneous Queue Length for different values of δ. We keep α constant at 0.001, the value we have used in all the experiments so far. As we observe, increasing the value of δ results in increasing stability, with both the average queue estimate as well as average instantaneous queue length settling down . However, care should be taken that δ not be kept too large, as increasing values of δ result in increasing rise times of the average queue estimate and increasing initial overshoot of the average instantaneous queue length.

Returning to the problem of tuning RED parameters, not only does the performance of the mechanism depend on link bandwidth, but also on the average packet size of the flows. We now illustrate this via experiment 4. Consider the two queue setting with the link capacities set to 15Mb. This time we increase the packet size from 500 Bytes to 1500 Bytes. This results in a δ that is approximately the same as the one where link capacity was 5 Mb/s with 500 Byte



(a) Queue estimate



(b) Instantaneous Queue Length

Figure 6: Asymmetric case, Plots for Queue 2, Experiment 2

packets (the number of packets that are being processed remains a constant). Looking at Figure 10, we indeed observe that the system is stabilized.

Since the average packet size is not something network designers can control, the only hope to stabilize the RED algorithm is to make δ a parameter whose value is independent of C and the packet size. Implementing an algorithm whose stability is influenced by external factors (user packet sizes) is also not good from a security point of view. A malicious user could conceivably influence RED behavior by sending very small or very large packets.

It is not sufficient to make α, the forgetting factor, small to reduce the effects of transient spikes in incoming traffic. If δ changes in response to the input packet rate, then a changing δ means a changing forgetting rate. In fact, because of the very nature of the spikes, at the point of their arrival RED starts sampling the queue

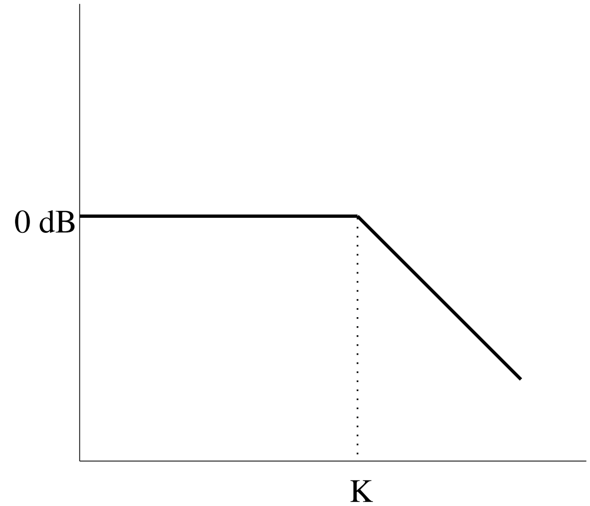


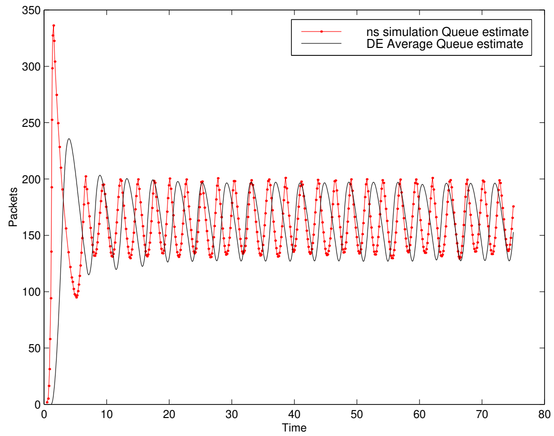
Figure 7: Magnitude Bode Plot of the first order averaging filter

size much more frequently, thereby "forgetting" history that much more quickly and starts tracking the spike closer.

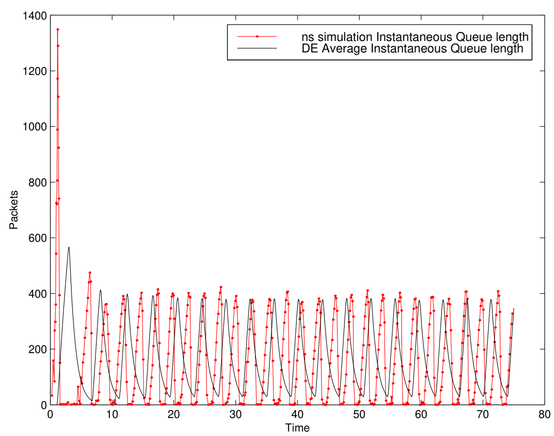
In [6] Firoiu et. al. suggest that δ should be made equal to the smallest round trip time of the flows going through the link. They also suggested that the δ thus selected is "good enough" and finer sampling won't improve things. The authors propose some guidelines for choosing RED parameters. We are currently investigating "tuning" RED parameter values via a different, control theoretic viewpoint.

Further, in [6] the authors also suggested that the discontinuity in the RED drop function (the jump from p^{max} to 1 at t^{max}) is a cause of oscillations in RED. It has been suggested that making the drop function continuous via the *gentle_* [7] should improve things in that regard. Our experiments show that the discontinuity is *not* the *only* cause of oscillations and that a simple fix by making the drop function continuous won't remove them. To illustrate that, we repeat the previous experiment with $p^{max} = 1$, thereby removing the discontinuity. We perform experiment 5, and the results are illustrated in Figure 11, showing that the oscillations *persist*. A quick investigation with our differential equation tool reveals that $\alpha = 1 \times 10^{-5}$ stabilizes the system, keeping $\delta = 1/C$. However, on the downside the system becomes very sluggish in it's response time to changes in the load. Thus, this tradeoff between responsiveness and stability unfortunately cannot be avoided with the RED control mechanism. Thus, it is a combination of the link bandwidth C, average packet size, α, δ and load levels which make the system stable. Summarizing, our main observations with RED are

- The adaptive nature of the sampling interval is harmful and can lead to oscillations
- The averaging algorithm needs to be modified, to make the sampling period a static value independent of packet sizes or arrival rates
- The presence of oscillations depends on many factors including packet size, link bandwidth and load levels

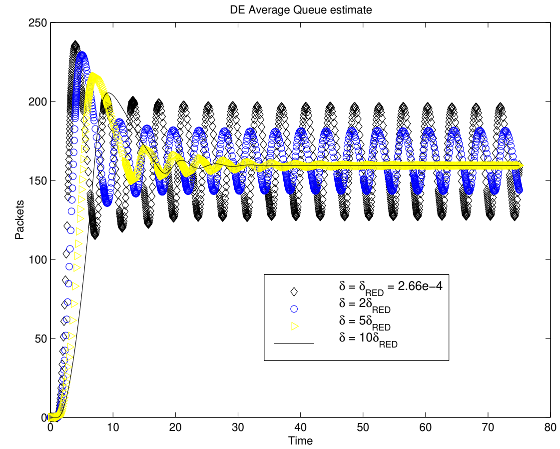


(a) Average Queue estimate

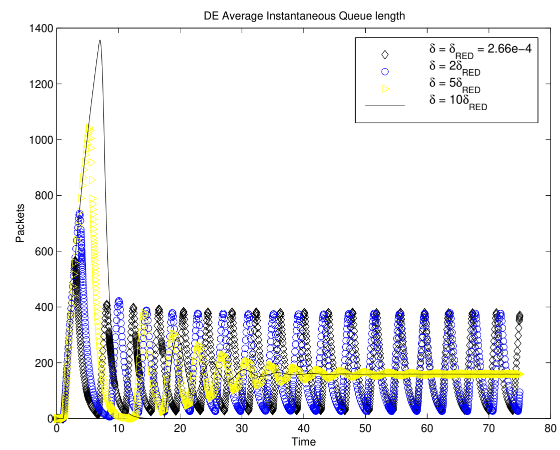


(b) Instantaneous Queue Length

Figure 8: Link speed 15 Mb/s, Packet size 500 Bytes, Experiment 3



(a) Average Queue estimate



(b) Instantaneous Queue Length

Figure 9: Link speed 15 Mb/s, Packet size 500 Bytes

3.5 Modeling different variations of TCP

We have described a generic TCP model, i.e., we have not considered the different variations Reno, Sack, Fack etc. One of the major differences between the variations of TCP is how they infer timeouts. Sack and Fack yield a much better behavior with respect to timeouts We model the effects of timeouts by the function $Q(w)$, where $Q(w)$ gives the split between timeout and triple duplicate ack losses given that a loss occurred when the window size was w. The $Q(w)$ that we used was derived in [13], which seems valid for Reno. For Sack, a $Q(w)$ lower than the one derived for Reno should be used. Our observation has been that a $Q(w)$ of $\min(1, 1/w)$ seems to work for Sack while Reno needs a $Q(w)$ of $\min(1, 3/w)$. This was validated by experiments not reported here.

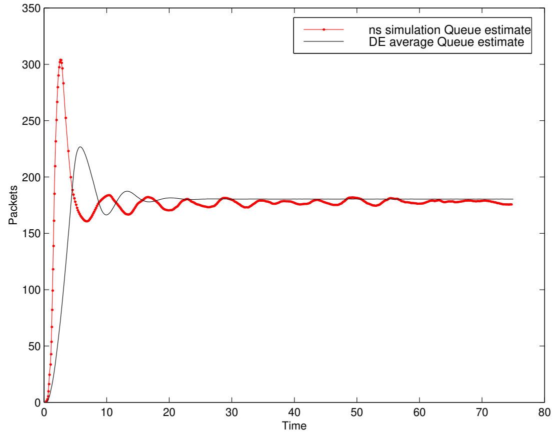
3.6 Control systems mapping

A very important by-product of our modeling and formulation is that we can map the differential equation based TCP+AQM system into a classical control systems model. We can then use standard techniques to analyze various mechanisms and propose improvements to algorithms as well as analysis-backed guidelines for choosing parameters of the algorithms. We are currently preparing a manuscript describing our work on the analysis of such systems from a control systems perspective.

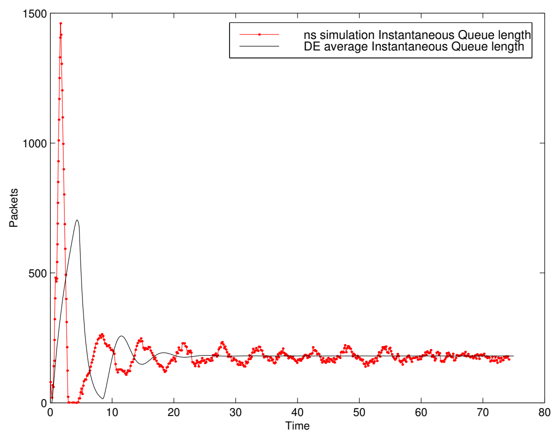
3.7 Some comments on modeling and simulation

The lack of slow-start in the model used to predict performance only affects initial start up of the experiments. Once the system nears the stable point, the DE solver is able to track changes in the network well. Hence, for computational simplicity, we did not incorporate slow-start.

At a number of places in our derivation, we had to make the approximation $E[f(x)] \approx f(E[x])$. This is strictly not correct and should cause errors. However our system seems to capture the dynamics of TCP reasonably well. For the particular case of $E[1/R(q)] \approx 1/R(\bar{q})$ we explain why the approximation may not be so bad: If we divide $R(q)$ into (a propagation delay + queueing delay), the



(a) Average Queue estimate

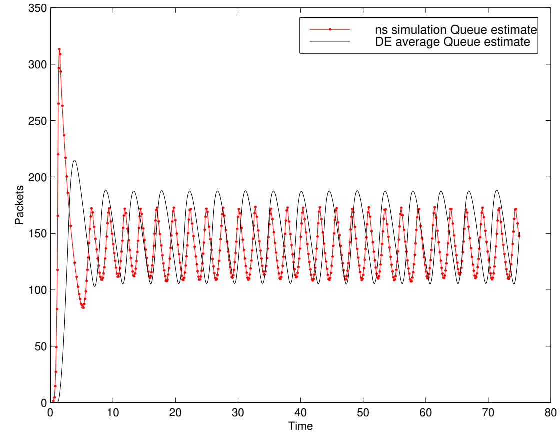


(b) Instantaneous Queue Length

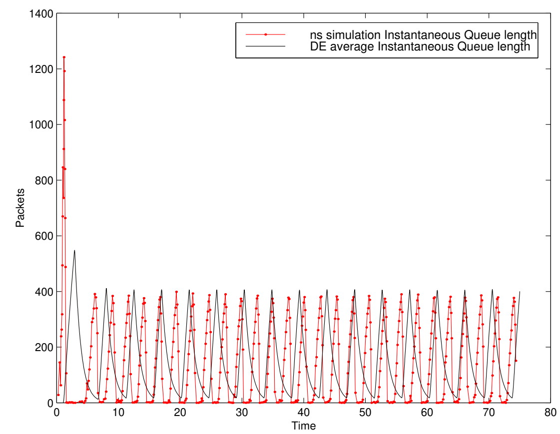
Figure 10: Link speed 15 Mb/s, Packet size 1500 Bytes, Experiment 4

randomness comes from the queueing delay. If we look at the plots from the experiments, for most cases we see that the queue length varies periodically. Thus, we can further break it down to a deterministic, periodic part and a random part. As the number of flows starts to increase, the random part of the round trip time makes up a smaller and smaller contribution, it is dominated by the deterministic part. In which case the approximation is not bad, since, if we write $R(q)$ as $R_d + R_r$, denoting the deterministic and random part respectively, we get

$$\begin{aligned} E\left[\frac{1}{R_d + R_r}\right] &= E\left[\frac{1/R_d}{1 + R_r/R_d}\right] \\ &= \frac{1}{R_d} E\left[\frac{1}{1 + R_r/R_d}\right] \\ &\approx \frac{1}{R_d} E[1 - R_r/R_d] \\ &= \frac{1}{R_d}(1 - \frac{E[R_r]}{R_d}) \end{aligned}$$



(a) Average Queue estimate



(b) Instantaneous Queue Length

Figure 11: Link speed 15 Mb/s, Packet size 500 bytes, $p_{max} = 1$, Experiment 5

We have used our methods to solve for thousands of flows in high bandwidth networks, however due to our inability to validate our results with simulation we do not present those results. While it is not right to compare the computational complexity of our technique with `ns`, as `ns` provides a lot more functionality, to get an idea of the speeds involved we mention that none of the experiments presented in the paper took more than 10 seconds to solve within MATLAB, while some of the `ns` simulations took several minutes to run.

4. CONCLUSIONS

We have developed a methodology to model and obtain (numerically) expected transient behavior of networks with Active Queue Management routers supporting TCP flows. We applied our techniques to analyze networks where the AQM policy was RED. Our results match well with simulation results, and are able to scale up well to large flows. We are able to get a qualitative understanding of the behavior of such networks quickly with our tool. Our modeling technique enables us to spot a possible problem with the RED averaging mechanism, which we verify via simulations. The technique that we presented in this paper is quite general purpose and can be easily extended to model and analyze other AQM mechanisms. There are several avenues of future work which we intend exploring. We are currently studying the control-dynamics of the system of RED routers and flows using the analytical formulation that we obtained. In terms of improvements to the methodology, finer grain modeling of TCP, accurate modeling of drop-tail behavior and a better handling of the stochastic nature of queueing delays are issues to be explored.

5. ACKNOWLEDGEMENTS

We thank C.V. Hollot for many useful discussions. We also thank the anonymous reviewers and Victor Firoiu for useful comments and suggestions on earlier drafts of this paper.

APPENDIX

A. POISSON COUNTER DRIVEN STOCHASTIC DIFFERENTIAL EQUATIONS

Consider a stochastic integral equation

$$x(t) = x(0) + \int_0^t f(x(\tau), \tau)d\tau + \int_0^t g(x(\tau), \tau)dN(\tau) \quad (13)$$

where N_τ is a Poisson Counter. The solution of equation (13) is defined as follows.

Definition: $x(\cdot)$ is a solution of (13) in the Itô sense if, on an interval where N is constant, x satisfies $\dot{x} = f(x, t)$ and if, when N jumps at t_1, x changes according to

$$\lim_{t \to t_1^+} x(t) = g(\lim_{t \to t_1^-} x(t), t_1) + \lim_{t \to t_1^-} x(t)$$

and $x(\cdot)$ is taken to be continuous from the left. Equation (13) is often written as

$$dx(t) = f(x, t)dt + g(x)dN(t)$$

and is called the Poisson Counter Driven Stochastic Differential Equation (PCSDE) or Jump Process Driven Stochastic Differential Equation.

We list some direct consequences of the above definition. Consider a stochastic differential equation driven by n independent Poisson Counters $N_1, ..., N_n$:

$$dx = f(x)dt + \sum_{i=1}^{n} g_i(x)dN_i(t), \quad x \in R^n.$$

We have the following "Itô rule". If $\psi : R^n \to R$ is a differentiable function, then

$$d\psi(t) = \langle \frac{\partial \psi}{\partial x}, f(x) \rangle dt + \sum_{i=1}^{n} [\psi(x(t) + g_i(x(t))) - \psi(x(t))]dN_i(t).$$

Since $x(t)$ is continuous from the left and the Poisson counter is taken to be continuous from the right, we have

$$\frac{d}{dt}E[x(t)] = E[f(x(t))] + \sum_{i=1}^{m} (E[g_i(x(t), t))]\lambda_i(t)$$

where $\lambda_i(t)$ is the rate for $N_i(t)$.

2. REFERENCES

[1] K. J. Åström and B. Wittenmark. *Computer Controlled Systems: Theory and Design*. Prentice-Hall, 1984.

[2] R. Brockett. Stochastic control. Lecture Notes, Harvard University.

[3] D. D. Clark and W. Fang. Explicit allocation of best effort packet delivery service. *ACM Transations on Networking*, August 1998.

[4] W. Feng, D. Kandlur, D. Saha, and K. Shin. Blue: A New Class of Active Queue Management Algorithms. Technical report, UM CSE-TR-387-99, 1999.

[5] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin. A Self-Configuring RED Gateway. In *Proceedings of Infocom 1999*.

[6] V. Firoiu and M. Borden. A study of active queue management for congestion control. In *Proceedings of Infocom 2000*.

[7] S. Floyd. Recommendation on using the "gentle_" variant of RED. http://www.aciri.org/floyd/red/gentle.html, March 2000.

[8] S. Floyd and V. Jacobson. Random Early Detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4), August 1997.

[9] I. Matta and A. U. Shankar. Z-iteration: Efficient estimation of instantaneous measures in time-dependent multi-class systems. In *Proceedings of SIGMETRICS/Performance '95*. 1995.

[10] M. May, T. Bonald, and J.-C. Bolot. Analytic Evaluation of RED Performance. In *Proceedings of Infocom 2000*.

[11] V. Misra, W. B. Gong, and D. Towsley. Stochastic Differential Equation Modeling and Analysis of TCP Windowsize Behavior. Technical Report ECE-TR-CCS-99-10-01, 1999. Presented at Performance 99, October 1999 Istanbul. Available at ftp://gaia.cs.umass.edu/pub/Misra99-TCP-Stochastic.ps.gz.

[12] T. J. Ott, T. V. Lakshman, and L. H. Wong. SRED: Stabilized RED. In *Proceedings of Infocom 1999*.

[13] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of ACM/SIGCOMM*, 1998.

[14] M. Vojnovic, J. Y. Le Boudec, and C. Boutremans. Global fairness of additive-increase and multiplicative-decrease with heterogenous round-trip times. In *Proceedings of IEEE INFOCOM 2000*, March 2000.