A Study of Active Queue Management for Congestion Control

Victor Firoiu
vfiroiu@nortelnetworks.com
Nortel Networks
600 Tech Park
Billerica, MA 01821 USA

Marty Borden[1]
mborden@tollbridgetech.com
TollBridge Technologies
872 Hermosa Dr.
Sunnyvale, CA 94086 USA

Abstract - **In this work, we investigate mechanisms for Internet congestion control in general, and Random Early Detection (RED) in particular. We first study the current proposals for RED implementation and identify several structural problems such as producing large traffic oscillations and introducing unnecessary overhead in the fast path forwarding.**

We model RED as a feedback control system and we discover fundamental laws governing the traffic dynamics in TCP/IP networks. Based on this understanding, we derive a set of recommendations for the architecture and implementation of congestion control modules in routers, such as RED.

I. Introduction

Congestion control for IP networks has been a recurring problem for many years. The problem of congestion collapse encountered by early TCP/IP protocols has prompted the study of end-to-end congestion control algorithms in the late 80's and proposals such as [4], which forms the basis for the TCP congestion control in current implementations. The essence of this congestion control scheme is that a TCP sender adjusts its sending rate according to the rate (probability) of packets being dropped in the network (which is considered a measure of network congestion). This algorithm is relatively well understood and several models have been proposed and verified with increasing degrees of accuracy through simulation and Internet measurements [6], [8].

In traditional implementations of router queue management, the packets are dropped when a buffer becomes full (in which case the mechanism is called Drop-Tail). More recently, other queue management mechanisms have been proposed (the most popular being Random Early Discard (RED), [3]). RED has the potential to overcome some of the problems discovered in Drop-Tail such as synchronization of TCP flows and correlation of the drop events (multiple packets being dropped in sequence) within a TCP flow. In RED, packets are randomly dropped before the buffer is completely full, and the drop probability increases with the average queue size.

RED is a powerful mechanism for controlling traffic. It can provide better network utilization than Drop-Tail if properly used, but can induce network instability and major traffic disruption if not properly configured. RED configuration has been a problem since its first proposal, and many studies [2], [1], [5] have tried to address this topic. Unfortunately, most of the studies propose RED configurations ("optimal" sets of RED parameters) based on heuristics and simulations, and not on a systematic approach. Their common problem is that each proposed configuration is only good for the particular traffic conditions studied, but may have detrimental effects if used in other conditions.

In this study, we propose a general method for configuring RED congestion control modules, based on a model of RED as a feedback control system with TCP flows. We use this model and requirements for stability and efficiency to derive a set of RED configuration parameters good for a given range of traffic characteristics and line speed. Our algorithm for computing RED parameters is implemented in a configuration program that can be used by network managers for router configuration.

The rest of the paper is organized as follows. In Section II we construct a model for RED as a feedback control system, we verify it by simulation, and we use it to derive its stability conditions. In Section III we use these conditions to configure the RED control function. In Section IV we study the estimator of the average queue size and make recommendations for its configuration. Section V concludes the paper and gives some directions for future work.

II. Queue-size-based Congestion Control as a Feedback Control System

In this section, we analyze the dynamics of TCP congestion control in the presence of a queue-size-based congestion control module. First (in Section II-A), we develop a model of average queue size when TCP flows pass through a queue system with fixed drop probability. In Section II-B we verify this model through simulations. In Section II-C, we combine this model with RED's control element and derive the steady state behavior of the resulting feedback control system. In Section II-D we analyze the stability of the RED control system.

A. A model of average queue size as a function of average packet drop probability

In this section we construct a model of TCP congestion control coupled with a queuing system that drops packets. For the

[1] Most work done while at Nortel Networks.

purpose of constructing an approximate but tractable model, we make several simplifying assumptions.

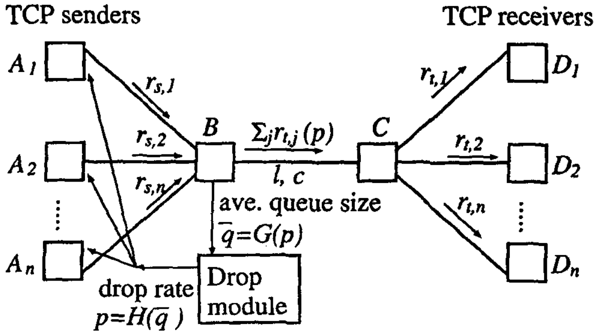

Fig. 1. An n-flow feedback control system

We consider a system of n TCP flows passing through a common link l with capacity c as in Figure 1. TCP flow f_i, $1 \leq i \leq n$, is established between hosts A_i and D_i and transports data in one direction, from A_i to D_i. The traffic in opposite direction consists only of acknowledgement packets (ACKs).

We assume that all access links, $A_i - B$ and $C - D_i$, have enough capacity so that $B - C$ is the only bottleneck link for any flow f_i, i.e., the only link where incoming traffic rate can surpass the link's capacity, and thus where packets may be discarded. We assume that the number of flows n remains constant for a long period of time, and that all flows have long duration (i.e., have data available to send for a long time). We assume a TCP Reno implementation (currently the most commonly deployed), where the congestion control window increases linearly over time during periods of no packet discard. If a packet discard is detected, the congestion control window is halved or a time-out waiting period begins. [10] has a detailed description of TCP Reno. In [8] and [9] we have derived a model for TCP Reno congestion control, which we will use in the following.

Each flow f_i sends with rate $r_{s,i}$. The sending rates of all n flows combine at the buffer of link l and generate a queue of size q. The discard module at link l drops packets with probability p that is a function of average queue size $\bar{q}$. For each flow f_i, the packets that are not discarded are sent on link l with rate $r_{t,i}$ (smaller than $r_{s,i}$ by the number of packets dropped from f_i per second). Each TCP sender adjusts its sending rate according to the drop probability p.

We can view this system as a feedback control system with the controlled systems being the TCP senders, the controlling element being the drop module, the feedback signal - the drop probability, and the controlled variables - the TCP sending rates.

This control system is different from a classical control system in that there are several simultaneously controlled elements instead of one. Also, the number of controlled elements (TCP flows) can vary over time. The purpose of the controlling element is to bring and keep the cumulative throughput (of all flows) below (or equal to) the link's capacity c:

$$\sum_{j=1}^{n} r_{t,j} \leq c \qquad (1)$$

Since we have assumed that the TCP flows have long duration and that their number does not change, the throughput of each TCP flow follows the steady state model that we have derived in [9]: [1]

$$r_{t,i}(p, R_i) = T(p, R_i)$$

where $T(p,R) =$

$$\begin{cases} M\dfrac{\frac{1-p}{p}+\frac{W(p)}{2}+Q(p,W(p))}{R(\frac{b}{2}W(p)+1)+\frac{Q(p,W(p))F(p)T_0}{1-p}} & \text{if } W(p) < W_{max} \\ M\dfrac{\frac{1-p}{p}+\frac{W_{max}}{2}+Q(p,W_{max})}{R(\frac{b}{8}W_{max}+\frac{1-p}{pW_{max}}+2)+\frac{Q(p,W_{max})F(p)T_0}{1-p}} & \text{otherwise} \end{cases} \qquad (2)$$

T is the throughput of a TCP flow (in bits/second) and depends on the packet drop probability p, the average round trip time R, the average packet size M (in bits), the average number of packets acknowledged by an ACK b (usually 2), the maximum congestion window size advertised by the flow's TCP receiver W_{max} (in packets) and the duration of a basic (non-backed-off) TCP timeout T_0 (which is typically $5R$). Also, W, Q and F have the following expressions:

$$W(p) = \tfrac{2+b}{3b}+\sqrt{\tfrac{8(1-p)}{3bp}+\left(\tfrac{2+b}{3b}\right)^2} \qquad (3)$$

$$Q(p,w) = \min\left(1, \frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{w-3}))}{1-(1-p)^w}\right) \qquad (4)$$

$$F(p) = 1+p+2p^2+4p^3+8p^4+16p^5+32p^6 \qquad (5)$$

In the following, we construct a simple model [2] of the n-flow feedback system by assuming that all flows have the same average round trip time, $R_i = R$, $1 \leq i \leq n$, the same average packet size, $M_i = M$ and that $W_{max,i}$ are sufficiently large to not influence $T(p, R)$. We have

$$r_{t,i}(p,R) = r_{t,j}(p,R), \quad 1 \leq i,j \leq n$$

and (1) becomes

$$r_{t,i}(p,R) \leq c/n, \quad 1 \leq i \leq n$$

We can now reduce the n-flow feedback system to a single-flow feedback system, as in Figure 2 (where we dropped the index i).

To determine the steady state of this feedback system (i.e., the average values of r_t, $\bar{q}$ and p when the system is in equilibrium), we need to determine the queue function (or queue "law") $\bar{q} = G(p)$ and the control function $p = H(\bar{q})$. The control function H is given by the architecture of the drop module, such as Drop-Tail or RED.

[1] Observe that the model in [9] assumes correlated drops due to Drop-Tail, whereas here we assume uncorrelated, random drops. We will see in Section II-B that our use of the model in [9] is a good approximation.

[2] The general model given in (1)..(5) and having heterogeneous RTTs, packet sizes and finite congestion windows is also valid, but does not have a closed-form solution. We report the study of the general model elsewhere.

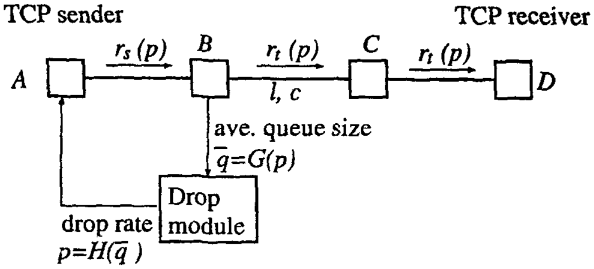

Fig. 2. A single-flow feedback control system

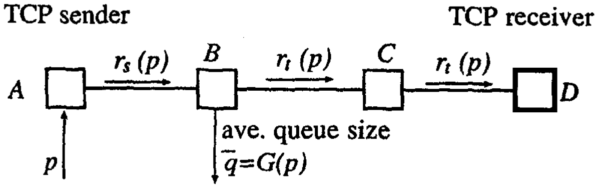

Fig. 3. An open control system with one TCP flow

To determine $\overline{q} = G(p)$ let us examine the open-loop (non-feedback) system in Figure 3, where p is an independent parameter. Since we have assumed that l is the only bottleneck link for any TCP flow, we have that the average round trip time of a packet is the sum of the average waiting time in the queue of link l, and R_o, the propagation and transmission time on the rest of its round trip. Assuming a FIFO queuing discipline at l, we have that the average waiting time in queue l is $\overline{q}/c$, and thus

$$R = R_o + \overline{q}/c$$

Depending on the value of p, the system can be in one of two states. For $p > p_0$, the line's bandwidth is underutilized

$$r_t(p, R) < c/n,$$

Then, the average queue size is negligible, thus $R = R_o$ and the utilization of the link is

$$u(p) = \frac{r_t}{c/n} = \frac{T(p, R_o)}{c/n}, \quad p > p_0$$

For $p \leq p_0$, the link's bandwidth is fully utilized, $u(p) = 1$, and the average queue size can be derived from the condition $r_t(p, R_o + \overline{q}/c) = c/n$:

$$\overline{q}(p) = c(T_R^{-1}(p, c/n) - R_o)$$

where $T_R^{-1}(p, y)$ is the inverse of $T(p, R)$ in R, i.e., $T_R^{-1}(p, T(p, R)) = R$, the function $T(p, R)$ being defined in (2). If the probability of random drop p is small enough so that $c(T_R^{-1}(p, c/n) - R_o) > B$ (B is the buffer size), then some additional packets are dropped due to buffer overflow. Obviously, the average queue cannot be larger that the buffer size B, thus,

$$\overline{q}(p) = \max(B, c(T_R^{-1}(p, c/n) - R_o)), \quad p \leq p_0$$

We can also determine p_0, the value of drop probability at which the link regime changes between under-utilized and fully-utilized. Since for $p < p_0$, $R = R_o$, we have that p_0 is given by

$$r_t(p_0, R_o) = c/n$$

Denoting by $T_p^{-1}(x, R)$ the inverse of $T(p, R)$ in p, i.e., $T_p^{-1}(T(p, R), R) = p$, we have that

$$p_0 = T_p^{-1}(c/n, R_o) \tag{6}$$

In conclusion we have the following expressions for the average queue size and link utilization as a function of the drop probability p:

$$\overline{q}(p) = \begin{cases} \max(B, c(T_R^{-1}(p, c/n) - R_o)), & p \leq p_0 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$u(p) = \begin{cases} 1, & p \leq p_0 \\ \frac{T(p, R_o)}{c/n}, & \text{otherwise} \end{cases} \tag{8}$$

In the next section, we verify the above model of the n-flow non-feedback system through simulation.

B. *Verification through simulation*

We have conducted extensive simulation experiments using the *ns* simulator [7]. The topology of the simulated network is as in Figure 1.

We have performed three simulation sets with link l of capacity $c = 1.5Mb/s$, $45Mb/s$ and $150Mb/s$. In all cases, all other links have significantly higher speed and buffer capacity than l such that there is no packet drop due to buffer overflow. For each line speed, we perform several experiments with different numbers of flows n. For each link speed c, we vary the number of flows n such that the average throughput per flow, $\tau = c/n$, lies between $\tau_{min} = 12.5Kb/s$ and $\tau_{max} = 750Kb/s$. Each TCP flow is generated by an "infinite" FTP application, i.e., that is active throughout the duration of the simulation. All TCP flows have RTT $R_o = 100ms$ (which does not include waiting time at queue l) and average packet size $500B$. The buffer at link l has a size $B = 2cR_o$. In all our experiments, this buffer size is large enough to avoid packet drops due to buffer overflow.

At the output port of link l at router B, we installed a module that randomly drops packets with a probability p. p is a configurable parameter that is fixed for the duration of a simulation experiment. For each experiment, we record a trace of the instantaneous queue size (sampled at each packet arrival and departure) and compute the average queue size, $\overline{q}_m$, over the duration of the experiment.

We ran several experiments with different values for p, and computed the measured average queue size $\overline{q}_m(p)$. In Figures 4-5 we compare $\overline{q}_m(p)$ with the predicted average queue size $\overline{q}_p(p)$ computed using (7)).

In Figure 4, we plot the measured and predicted average queue size as a function of drop probability for line speed $c = 1.5Mb/s$ and $n = 2, 20$ flows. The queue size is scaled

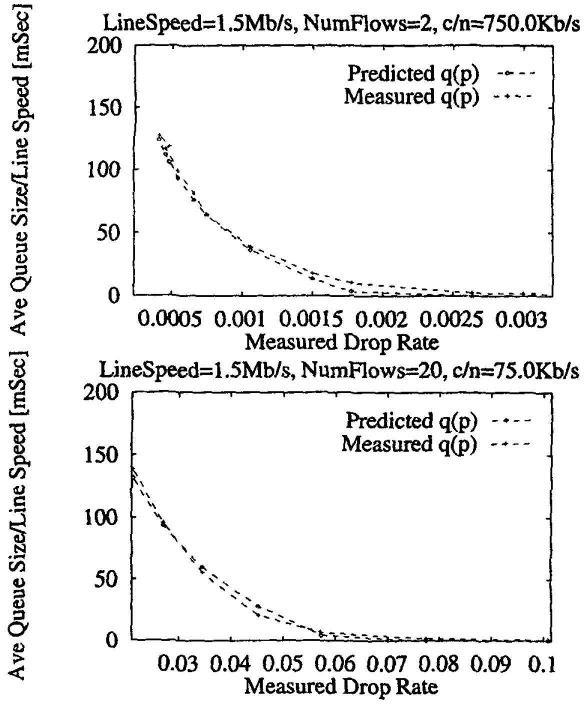

Fig. 4. Measured and predicted scaled average queue size, Line speed = 1.5Mb/s

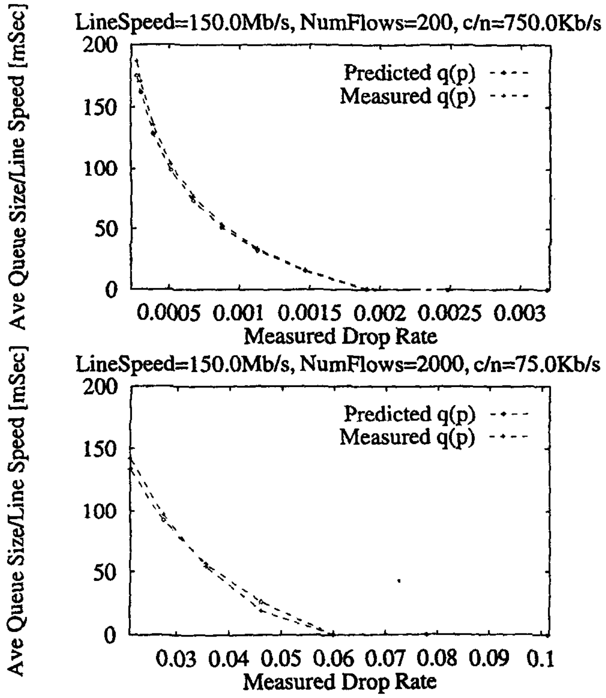

Fig. 5. Measured and predicted scaled average queue size, Line speed = 150Mb/s

with the line speed for the purpose of comparison between operation of links with different line speeds. We plot similar graphs in Figure 5 for a line speed $c = 150Mb/s$ and $n = 200, 2000$ flows.

First, we observe that the predictions are close to the measurements in all the cases, and for all values of p. A second observation is that the predictions are equally valid for all link speeds. This leads us to the conclusion that the queue dynamic is independent of the link speed. Moreover, we observe that the scaled average queue size $\overline{q}(p)/c$ does not depend on the link speed but just on $\tau = c/n$, the average throughput per flow. We conclude here that τ is a measure of level of congestion, and henceforth we take it as our definition for the level of congestion at a given link.

In all of the above experiments we also recorded the state of the link over time, i.e., the beginning and end of the busy periods (when the link is busy sending packets). At the end of each experiment, for a given drop probability p, we compute the average utilization of the link, $u_m(p)$ (computed as the time average of the link state trace).

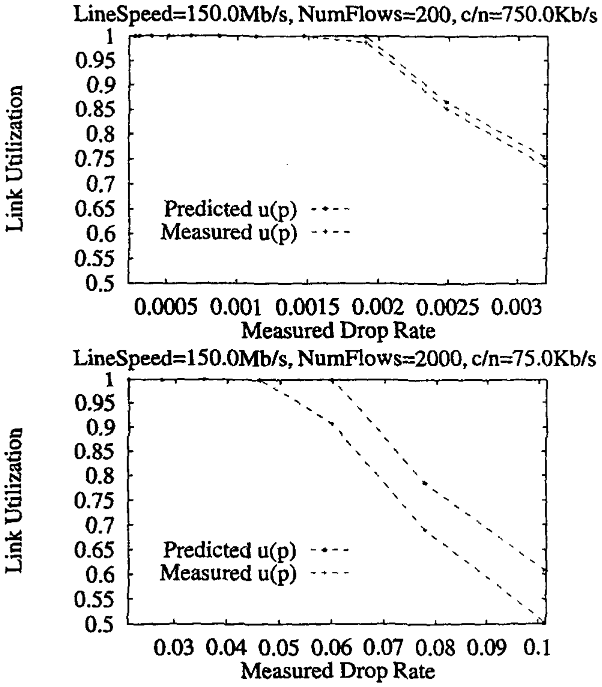

Fig. 6. Measured and predicted average link utilization, Line speed = 150Mb/s

In Figure 6, we plot the measured and predicted average link utilization as a function of drop probability for a line speed $c = 150Mb/s$ and $n = 200, 2000$ flows. We obtain similar results for line speeds of $1.5Mb/s$ and $45Mb/s$. In all our experiments we observe that the predictions are close to the measurements, although not as close as in the case of average queue size.

Given all the above results, we conclude that our model of queue dynamics is well confirmed through simulation. In the following sections we use this model to analyze the dynamics of RED as a feedback control system.

C. Steady-state operation of RED congestion control

Let us return to the feedback control system in Figure 2. In Section II-A we have derived an expression for the long-term (steady-state) average queue size as a function of packet drop probability denoted by $\overline{q}(p) = G(p)$, as in (7). If we assume that the drop module has a feedback control function denoted by $p = H(\overline{q}_e)$, where $\overline{q}_e$ is an estimate of the long-term average of the queue size, and if the following system of equations

$$\begin{cases} \overline{q} = G(p) \\ p = H(\overline{q}) \end{cases} \tag{9}$$

has a unique solution $(p_s, \overline{q}_s)$, then the feedback system in Figure 2 has an equilibrium state $(p_s, \overline{q}_s)$. Moreover, the system operates on average at $(p_s, \overline{q}_s)$, i.e., its long-term average of packet drop probability is p_s and average queue size is $\overline{q}_s$.

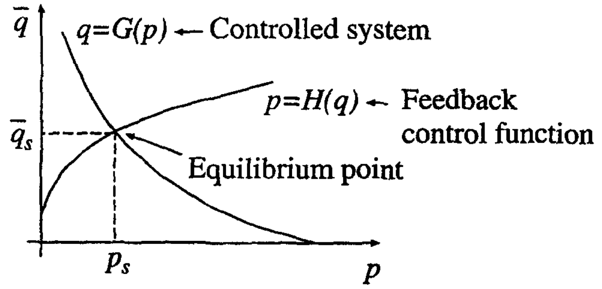
$\overline{q}$
q=G(p) ← Controlled system
p=H(q) ← Feedback control function
$\overline{q}_s$
Equilibrium point
p_s
p

Fig. 7. Equilibrium point for a feedback system

Figure 7 illustrates this result: the equilibrium point $(p_s, \overline{q}_s)$ is at the intersection of the curves $\overline{q} = G(p)$ and $p = H(\overline{q})$. A justification for the above result is that the system is constrained on one hand by the queue size law $\overline{q} = G(p)$ (equation (7) derived in Section II-A), and on the other hand by the control module through its control function $p = H(\overline{q})$. We also observe that $(p_s, \overline{q}_s)$ is the equilibrium point for the dynamic system defined in (9). We emphasize here that the system resides in state $(p_s, \overline{q}_s)$ **on average**, and that it does not necessarily stay in this state at all times. Indeed, the system can converge to this state or permanently oscillate around this state. We will consider the transient behavior of the system in Section II-D.

We can apply the above result to determine the long-term average operation of a system where the control module follows the RED algorithm [3]. In this case, the control function is:

$$p = H(\overline{q}_e) = \begin{cases} 0, & 0 \leq \overline{q}_e < q_{min} \\ \frac{\overline{q}_e - q_{min}}{q_{max} - q_{min}} p_{max}, & q_{min} \leq \overline{q}_e < q_{max} \\ 1, & q_{max} \leq \overline{q}_e \leq B \end{cases} \tag{10}$$

where $\overline{q}_e$ is the exponential weighted moving average of queue size, q_{min}, q_{max}, p_{max} are configurable RED parameters, and B is the buffer size.

We have run simulation experiments with the RED control module, recorded traces of queue size and packet drops and computed the average queue size $\overline{q}_m$ and packet drop probability p_m. We have also computed the predicted operating point $(p_s, \overline{q}_s)$ as the solution of the system of equations (10) and (7). In all cases, the predicted operating points were close to the measurements.

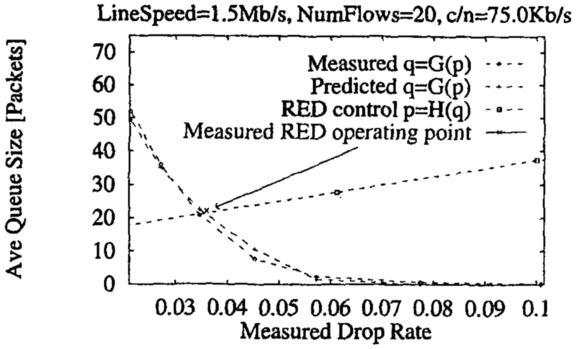

Fig. 8. RED average operating point: measured and predicted

For example, in Figure 8 we plot the theoretical average queue size given by (7), and the RED control function given by (10) with parameters $p_{max} = 0.1$, $q_{min} = 12.5 packets$, $q_{max} = 37.5 packets$, and buffer size $B = 75 packets$, as recommended in [2], and packet size $M = 500B$. Their intersection point represents the predicted operating point. We also plot the point $(p_m, \overline{q}_m)$ resulting from simulation. We observe that the average RED operating point is close to the intersection of the queue law and RED control curves, which confirms our model for the steady-state of the RED control system.

D. Transient operation of RED congestion control

In (9) we have defined a dynamic system having the average queue size and average packet drop rate as state parameters. This system may or may not be stable around the equilibrium point, depending on the functions H and G. Moreover, we are interested in the evolution of the instantaneous queue size in time, not only in its average. Therefore, in the following we describe more precisely the RED dynamic system.

We start from the observation that a TCP sender adjusts its congestion window (and thus its sending rate) depending on whether it has sensed that packets are dropped or not. If a packet is dropped at link l, this event is typically sensed at the TCP sender (which changes its rate accordingly) approximately one RTT after the packet has been dropped. Therefore, the feedback system we are about to model has a time lag of about one RTT between the moment a signal is sent by the control module (by dropping or not a packet) and the moment the controlled system (TCP sender) reacts to this signal. The increase or decrease in the TCP sending rate produces an increase or decrease in the instantaneous queue length at bottleneck link l, which prompts the RED module to again change its drop rate, and the process repeats.

In the following we model the RED feedback system as a discrete-time dynamic system with the time step of one RTT R. Assume that at time t_k the drop probability is p_k. At time $t_{k+1} = t_k + R$, the TCP senders react to p_k, and on average they adjust their sending rate to r_{k+1}. The result is that the queue size at t_{k+1} is $q_{k+1} = G(p_k)$, following the "queue law" (7). The RED module computes a new estimate of queue size $\overline{q}_{e,k+1} = A(\overline{q}_{e,k}, q_{k+1})$, following the exponential weighted moving average

$$A(\overline{q}_{e,k}, q_{k+1}) = (1-w)\overline{q}_{e,k} + wq_{k+1}$$

RED then changes its drop rate to $p_{k+1} = H(\overline{q}_{e,k+1})$, according to its "control law" (10). In summary, we have identified a discrete-time dynamic system defined by the following system of recurrence equations: [3]

$$\begin{aligned} q_{k+1} &= G(p_k) \\ \overline{q}_{e,k+1} &= A(\overline{q}_{e,k}, q_{k+1}) \\ p_{k+1} &= H(\overline{q}_{e,k+1}) \end{aligned}$$

A quantitative study of the transient evolution of this dynamic system can be quite complex, and we leave it for future work. In the following we give a qualitative investigation, backed by some experimental examples.

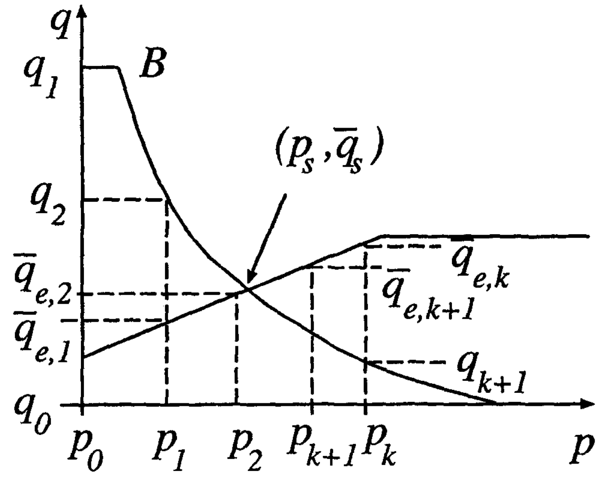

Fig. 9. RED operating point converges

In a first scenario, the "queue law" and "control law" are as in Figure 8, also sketched in Figure 9. Suppose the initial state is $q_0 = 0$, $\overline{q}_{e,0} = 0$ and $p_0 = 0$. At time t_1, $q_1 = B$ (since the drop rate is zero, the sending rate is high, and the buffer is full) and $\overline{q}_{e,1} = wB$ (w is a small value, say 0.002, as recommended in [3]). So the average queue size increases a bit, and the new drop rate increases a bit too, to p_1 (see Figure 9). The process repeats, and the system approaches the equilibrium point $(p_s, \overline{q}_s)$ (defined in Section II-C). On the other hand, if at some time t_k, we have $p_k > p_s$, then $p_{k+1} < p_k$ since (follow the Figure 9) $q_{k+1} < \overline{q}_{e,k}$, and thus $\overline{q}_{e,k+1} < \overline{q}_{e,k}$. We conclude that the equilibrium point $(p_s, \overline{q}_s)$ is an attractor for all states around it and that, once the system reaches the equilibrium state, it will stay there with only small statistical fluctuations, given that the number of flows n does not change.

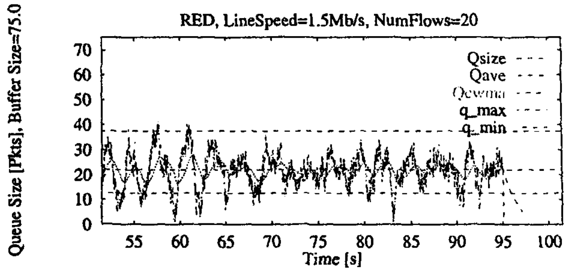

Fig. 10. Instantaneous and average queue size in time, converging case

In Figure 10 we show the evolution of the instantaneous queue size and e.w.m. average queue size in time, and observe that they have indeed a moderate variation around the equilibrium point. This queue trace corresponds to the system whose steady state was plotted in Figure 8.

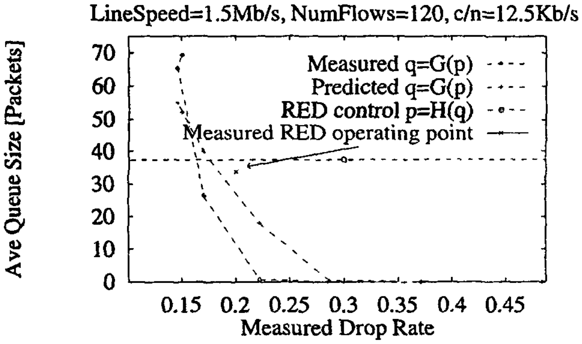

Fig. 11. RED average operating point situated beyond $p_{max} = 0.1$

In a second scenario, the queue and control laws are as in Figure 11. In this case, the equilibrium point is situated beyond $p_{max} = 0.1$, i.e., on the horizontal line of the control law where the drop rate has a jump from 0.1 to 1, as given by (10). We clearly see that the system, although attracted by this point, cannot stay there, since the value of p is not defined.

More precisely (see Figure 12), once $\overline{q}_{e,k} > q_{max}$, $p_{k+1} = 1$, i.e., all packets are dropped. Then all TCP rates drop to zero, and thus $q_{k+1} = 0$ and the average queue size begins to decrease, $\overline{q}_{e,k+1} < \overline{q}_{e,k}$. After the average queue size $\overline{q}_e$ drops below q_{max}, the drop rate becomes less than 1 (in fact less than 0.1), and the queue grows again. The effect is that the queue

[3] Note that this is not an accurate model of a single TCP flow on a small time scale such as one round trip time. Nevertheless, it proves to be a good model for a set of TCP flows aggergated on a link.

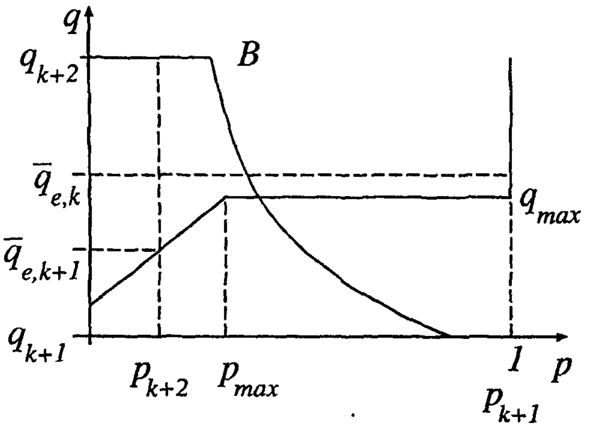

Fig. 12. RED operating point oscillates

length oscillates widely between 0 and full buffer B. These large oscillations can propagate through the network, making it unstable and resulting in a very harmful, erratic behavior.

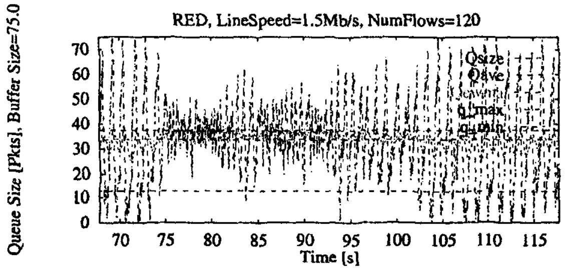

Fig. 13. Instantaneous and average queue size in time, oscillating case

In Figure 13 we plot the trace of queue size for the RED system whose steady state is in Figure 11. We observe that indeed the queue size oscillates between 0 and full buffer size B, and that the oscillations do not decrease over time. Definitely, this kind of situation is harmful and should be avoided by a proper configuration of the RED control law. Such a configuration would avoid an operation close to the discontinuity of the RED control law or would eliminate any such discontinuity. We specify such configuration details in Section III.

Other characteristics of the RED control function, such as the slope of the segment between q_{min} and q_{max}, may influence the stability of the feedback system and/or its rate of convergence to the equilibrium point. For example, it is likely that a small slope of RED control function, $\alpha = \partial q/\partial p$ would result in a faster converging system than a control function with large slope, but the system would be less stable. We leave this study for future work.

In the next section, we give a set of recommendations for configuring RED, that take into consideration the above constraints.

III. Recommendations for configuring RED control function

In this section we derive a set of recommendations for configuring the RED control function. In general, there are many different functions that can be good candidates for RED control. Even if the space of possible functions is limited by the constraints from Section II-D, the number of functions remains infinite.

To reduce the complexity of choosing such a function, we group them in a few categories, and restrict their shape to being piece-wise linear. In Section II-D we have uncovered the "queue law" $\overline{q} = G(p)$, as shown in Figure 14. The steady state of the system is determined by the intersection of the control law with the queue law. For example, a control law H_1 with a high slope results in a state with low drop rate but large average queue size. Conversely, H_2 results in a lower average queue size, but larger drop rate.

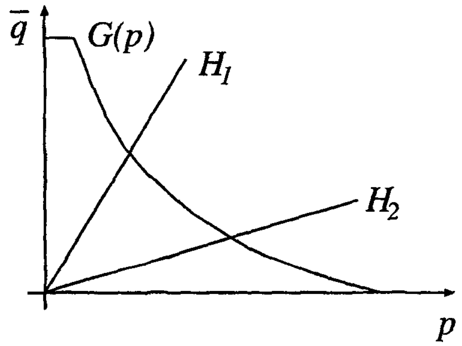

Fig. 14. Two control laws implementing different policies

In the following, we define two policies for RED control and we give recommendation for configuring each of them:

- **drop-conservative policy:** low p, high $\overline{q}$
- **delay-conservative policy:** low $\overline{q}$, high p

When configuring the RED control at a specific line (output interface) of a router or switch, we start from several estimates of averages or bounds for traffic conditions:

- The line speed c.
- The minimum and maximum throughput per flow, τ_{min}, τ_{max}, or alternatively, the minimum and maximum number of simultaneous flows, n_{min} and n_{max}. For example, $\tau_{min} = 28.8Kb/s$ and $\tau_{max} = 56Kb/s$ for a WAN with predominantly dial-up traffic, or $\tau_{min} = 100Kb/s$ and $\tau_{max} = 1Mb/s$ for LAN or enterprise environment.
- The minimum and maximum data packet size (non-ACK), M_{min} and M_{max}, or average, M_{ave}. For example, $M_{min} = 256B$, $M_{max} = 1500B$, or $M_{ave} = 512B$.
- The minimum and maximum round trip time (excluding time in the queue of the line under configuration), $R_{o,min}$ and $R_{o,max}$, or average, $R_{o,ave}$. For example, in a LAN,

$R_{o,min} = 2ms$, $R_{o,max} = 15ms$ and $R_{o,ave} = 10ms$ and in a WAN, $R_{o,min} = 50ms$, $R_{o,max} = 200 - 500ms$ and $R_{o,ave} = 100ms$.

Each combination of n, M, and R produces a different queue law $G(p)$. We are interested in applying the conditions from Section II-D. We observe that, since the conditions should hold for any queue law corresponding to any combination of n, M and R within a specified range, it is sufficient to verify the conditions for functions $G(p)$ corresponding to extreme values of n, M and R.

We define the "maximum queue law" $G_{max}(p)$ to be the function $G(p)$ defined in (7) with parameters n_{max}, M_{max} (or M_{ave} if M_{max} is not defined) and $R_{o,min}$ (or $R_{o,ave}$ if $R_{o,min}$ is not defined). It is easy to show that $G_{max}(p) \geq G(p)$ for all p, where $G(p)$ has parameter values n, M and R within their specified ranges. Similarly, we define the "minimum queue law" $G_{min}(p)$ to be the function $G(p)$ defined in (7) with parameters n_{min}, M_{min} (or M_{ave} if M_{min} is not defined) and $R_{o,max}$ (or $R_{o,ave}$ if $R_{o,max}$ is not defined). It is easy to show that $G_{min}(p) \leq G(p)$ for all p, where $G(p)$ has parameter values n, M and R within their specified ranges. Thus, we have that the queue law can be anywhere between G_{max} and G_{min} as in Figure 15.

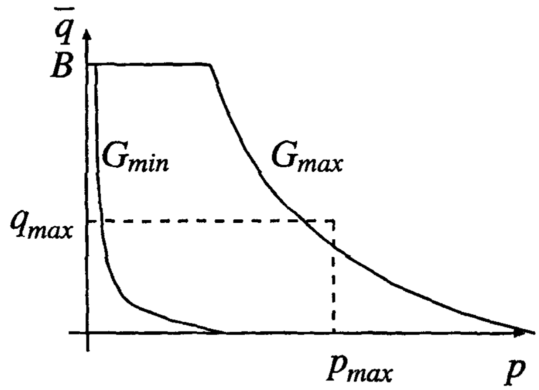

Fig. 15. Range of queue laws for a given queue system

In Section II-D we have seen that a condition for stability is that (p_{max}, q_{max}) of the RED control function should be "above" the queue law G. In the context defined above, this condition becomes "(p_{max}, q_{max}) above G_{max}".

For the drop-conservative policy, we are given p_{max}. Then, the condition becomes $q_{max} > G_{max}(p_{max})$. We can choose for example $q_{max} = 1.2G_{max}(p_{max})$ to provide space for statistical fluctuations.

For the delay-conservative policy, we are given d_{max}, the maximum queueing delay at the link. For example $d_{max} = 0.2R_{o,ave}$ in order to not add much to the average round trip time. Then $q_{max} = d_{max}c$, and the condition becomes $p_{max} > G^{-1}_{max}(q_{max})$. We can choose for example $p_{max} = 1.2G^{-1}_{max}(q_{max})$ to provide space for statistical fluctuations.

Although our traffic estimations (reflected in τ_{min} (or n_{max}), M_{max} and $R_{o,min}$) imply that the system will not operate "beyond" the G_{max} curve, we need to define the control function for that region as well, in order to cover any exceptional operation. Given the instability issues uncovered in Section II-D, we strongly recommend against any discontinuity in (p_{max}, q_{max}) such as a jump of p from p_{max} to 1 as suggested in [3]. We recommend a linear segment from (p_{max}, q_{max}) to $(1, B)$, where B is the buffer size. We recommend dimensioning the buffer size $B = 2q_{max}$, to allow for queue fluctuations on small time scales.

The set of recommendations put forth in this section involve some computations that may be complicated analytically, but have simple numerical solutions. We have incorporated all these computations into a configuration program which inputs the estimated parameters such as the ranges of RTT and number of flows along with the desired policy, and outputs the recommended parameters for the RED control function.

IV. Configuring the estimator of average queue size

A. Queue averaging algorithms

A queue-averaging algorithm is essentially a low-pass filter on the instantaneous queue size. It is intended to filter out brief queue changes or bursts, and to estimate longer-term queue average. The estimate is used by the RED drop module to adjust its drop rate according to a control function, studied in the previous sections.

The estimate is a moving average: at any given time, the average is computed over the set of samples taken in the previous I seconds, and we define I to be the averaging interval. The exponentially weighted moving average $\overline{q}$ is computed recursively based on previous average $\overline{q}_k$ and a new sample q_k:

$$\overline{q}_{k+1} = wq_k + (1 - w)\overline{q}_k$$

where $0 < w < 1$ is a weight. It follows that the average expressed only in terms of samples is

$$\overline{q}_{k+1} = w\sum_{i=0}^{k}(1 - w)^i q_{k-i}$$

If we assume that samples are taken at fixed time intervals δ, then a sample taken at time $t - m\delta$ contributes to the average computed at time t with weight $(1 - w)^m$. In other words, the contribution of a sample decreases exponentially with its age. If we consider that a weight smaller than a threshold a, $0 < a < 1$ (for example $a = 0.1$) makes a sample's contribution to the average negligible, then we can determine the number of samples m that are significant:

$$m = \frac{\ln a}{\ln(1 - w)}$$

It follows that the time interval I (where sample contributions are significant) is $I = m\delta$, or:

$$I = \delta \frac{\ln a}{\ln(1-w)} \tag{11}$$

In the next section, based on traffic considerations, we will develop recommendations for values for the averaging interval I and sampling interval δ. Then, the averaging weight follows from (11):

$$w = 1 - a^{\delta/I} \tag{12}$$

B. Averaging interval

A first condition on the queue averaging algorithm (exponentially weighted moving average) is to provide a good approximation of the long-term average in a system with a constant number of flows. In other words, if traffic conditions (number of flows, round trip times) do not change, the queue length estimate should be quasi-constant over time and close to the average queue length computed over a very long time interval. For example, the moving average should not be influenced by the linear increase and multiplicative decrease of flow rates produced by the TCP congestion control algorithm. We observe that the longer the averaging interval, the closer the moving average is to the long-term average.

A second and opposing condition on the queue averaging algorithm is to change its value as fast as possible to the new long-term average after a change in traffic conditions (such as number of flows or round trip times). We observe that the shorter the averaging interval, the faster the moving average reflect the new conditions.

In the following we derive an averaging interval that provides a good compromise between the two conditions above. Let us consider a system with n TCP flows having the same average round trip time R. Following the arguments in Section II, each TCP flow has on average a throughput of $r = c/n$ and a drop rate p. According to the TCP Reno congestion control algorithm, the sending rate is increasing until a loss indication occurs. If the indication is "triple-duplicate ACK," (TD) the window is reduced to half, and the window increase resumes (see Figure 16). If the indication is "time-out" (TO), there is a period of potentially multiple time-out intervals (when no packets are sent) alternating with small transmissions, after which the TCP window resumes its increase from one (see Figure 17). For the detailed description and analysis please refer to [8] and [9]. In both cases denote the period of this function by P. The variation in sending rate is reflected in similar variation of queue size, and thus the queue size has the same periodicity P.

We observe that if we take the averaging interval of our moving average to be equal to the period, $I = P$, then the moving average with interval I is close to the long-term average.

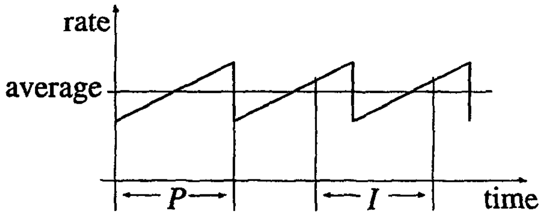

Fig. 16. Period and average interval of TCP sending rate, trip-dup (TD) case

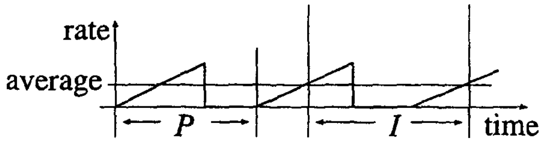

Fig. 17. Period and average interval of TCP sending rate, time-out (TO) case

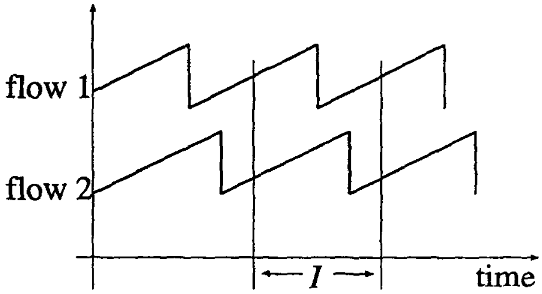

Fig. 18. Averaging two TCP flows

For an interval smaller than the average TCP period, $I < P$, the average follows closer the instantaneous queue size. For $I > P$, the moving average has small variations, but converges rapidly to the long-term average as I increases. We also observe that $I = P$ is a good averaging interval for a superposition of any number of functions having period P, as we can see in Figure 18. We conclude that $I = P$ is the value of choice for averaging interval. From the models in [8] and [9] and given the average RTT R and the drop probability p, we can compute P as the average TCP period $E[S]$:

$$\begin{cases} R(\frac{b}{2}W(p)+1) + \frac{Q(p,W(p))F(p)T_0}{1-p} & \text{if } W(p) < W_{max} \\ R(\frac{b}{8}W_{max} + \frac{1-p}{pW_{max}} + 2) + \frac{Q(p,W_{max})F(p)T_0}{1-p} & \text{otherwise} \end{cases} \tag{13}$$

where all the notation was introduced in connection to equation (2).

C. Sampling the queue size

To determine the frequency at which the queue size should be sampled, let us consider again the process to be sampled, which is the instantaneous queue size. We have seen in Section IV-B that the throughput of each TCP flow has a quasi-periodical variation, where the duration between loss indications, although statistical, has a defined mean. In steady state (i.e., when the drop rate p exhibits only small variations) TCP throughput has a periodic increase and decrease, as in Figure 16. More precisely, the packets in a TCP round are sent in a burst, and thus, the variation in TCP throughput is close to a step function that increases every b RTTs, where b is typically 2, as in Figure 19.

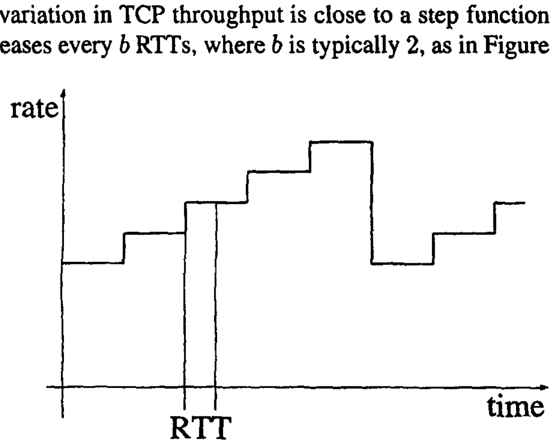

Fig. 19. TCP throughput is a step function

The queue size then behaves as a step function, changing at every RTT, since the queue size is synchronized with the TCP rate variation. It follows that the "ideal" sampling rate should be "once every RTT", since this would capture each change of value. More frequent sampling (at the same averaging interval) would not bring significantly more accuracy to the average, but would not decrease the accuracy either. Less frequent sampling, on the other hand, would miss significant changes, would diminish the accuracy of the average, and thus is not recommended.

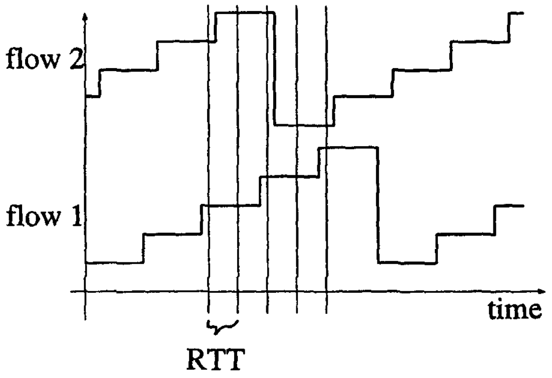

Fig. 20. Sampling two TCP flows

Observe also that the same sampling interval is also good for a superposition of TCP flows having the same RTT, as Figure 20 shows. If the flows have different RTTs, then, following the above logic that "larger is worse, smaller may be better or equal", we recommend the sampling interval to be equal to the minimum RTT, $\delta = R_{o,min}$.

V. Conclusion

In this study, we have investigated the Random Early Detection (RED) mechanism for congestion control. First, we have surveyed the existing work on RED configuration and implementation and identified potential problems. Then, we have modeled the queue-based congestion control (including RED) as a feedback control system. We have identified the "queue law" governing the queue size of a link transited by a number of TCP flows which, in conjunction with a given control law, determines the equilibrium state of the feedback system.

We have validated this model through simulation experiments. Using this model, we have identified potential problems of instability of the feedback system. To avoid such problems, we recommend a configuration of the RED control law.

Also, based on our understanding of TCP traffic dynamics, we have derived a set of recommendations for configuration of the RED queue size estimator: the frequency of queue sampling and the averaging weight.

There are several other aspects of RED that we intend to study in the future such as: the process of queue sampling (deterministic or stochastic), the process of randomly choosing packets to drop, and validating our results in heterogeneous TCP traffic such as flows with different RTTs and mix of short and long lived flows. Also, our study has focussed only on TCP traffic in general and its Reno implementation in particular. We plan to consider other TCP implementations such as TCP SACK, and other traffic types such as open-loop (uncontrolled) UDP traffic (such as voice) and mixes of TCP and UDP traffic.

VI. Acknowledgements

The authors would like to thank Don Towsley for many useful discussions.

References

[1] W. C. Feng, D. Kandlur, D. Saha, and K. Shin. A Self-configuring RED Gateway. In *Infocom'99*, 1999.

[2] S. Floyd. Notes on RED in the end-to-end-interest mail list. 1998.

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

[4] V. Jacobson and M. J. Karels. Congestion Avoidance and Control. In *SIGCOMM'88*, 1988.

[5] D. Lin and R. Morris. Dynamics of Random Early Detection. In *SIGCOMM'97*, 1997.

[6] M. Mathis, J. Semske, J. Mahdavi, and T. Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *Computer Communication Review*, 27(3), July 1997.

[7] S. MCanne and S. Flyod. ns-LBL Network Simulator, 1997. Obtain via http://www-nrg.ee.lbnl.gov/ns/.

[8] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *ACM SIGCOMM*, 1998.

[9] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. A Stchastic Model of TCP Reno Congestion Avoidance and Control. Technical Report CMPSCI TR 99-02, Univ. of Massachusetts, Amherst, 1999.

[10] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC2001, Jan 1997.