Dynamic Configuration of Red Parameters

Tigist Alemu, Alain Jean-Marie

Dynamic Configuration of RED Parameters

Tigist Alemu
LIRMM - University of Montpellier II
161, Rue Ada
34392 Montpellier Cedex 5, France
Email: tigist@lirmm.fr

Alain Jean-Marie
LIRMM - University of Montpellier II
161, Rue Ada
34392 Montpellier Cedex 5, France
Email: ajm@lirmm.fr

***Abstract*— Our work focuses on an adaptive approach of RED namely ARED (Adaptive RED) that performs a constant tuning of RED parameters according to the traffic load. ARED requires no hypothesis on the type of traffic, which diminishes its dependency on the scenario parameters such as the bandwidth, the round-trip time and the number of active connections. Our goal is to find a simple extension to ARED in order to improve the predictability of performance measures like queueing delay and delay jitter without sacrificing the loss rate. To achieve this goal, we propose a new algorithm that sets the RED parameters and evaluate it by extensive simulations. Our results show that compared to the original ARED, our algorithm can stabilize the queue size, keep it away from buffer overflow and underflow, and achieves a more predictable average queue size without substantially increasing the loss rate.**

Keywords : Adaptive RED, Simulations, Quality of Service.

I. Introduction

In order to prevent the Internet congestion collapse, the Internet Engineering Task Force (IETF) has recommended the use of RED [1], [2], an active queue management scheme which is able to achieve high throughput and low average delay (for TCP traffic) by spreading randomly packets drops between flows.

The most important drawback of RED is the difficulty of the tuning of its parameters Max_p, Min_{th} and Max_{th}. This is the topic of the present paper. Works investigating this issue are numerous : [3]–[7] showed that RED performances are sensitive to the level of congestion and to the settings of its parameters. This lead Feng et al. in [8] to the conclusion that there is no single set of RED parameters that works under different traffic scenarios and to the claim that the correct tuning of RED implies a "global" parameterization. Consequently, efforts have been made in order to reduce the sensitivity of RED towards its parameters.

The first approach is to build quantitative models as in [7], [9], [10] allowing the prediction of RED parameters based on network variables like the round trip time, the number of flows and link bandwidth. This approach provides a better understanding of the dependency of RED on traffic conditions. However, it is still difficult to retrieve or infer accurate informations about network variables from local observations. In practice, network load characteristics are not known from RED nodes and practical implementation of schemes like the PI controller [10], [11], and other works as [11]–[14] rely on values set a priori. Therefore, several works like [6], [8], [15] have advocated an adaptive approach named ARED (Adaptive RED) which can be deployed in the current Internet since it does not rely on hypotheses on the type of traffic because it infers the traffic load from measurements of the queue size.

This paper adopts this point of view. It differs however from previous works in the way parameters are adjusted. Our goal is to achieve a performance improvement over adaptive RED for the queueing delay and the delay jitter without sacrificing the loss rate. ARED adapts the value of Max_p, but uses fixed adjustment factors that do not reflect the effective change rate of the traffic load. Note that if the traffic load shows a slight change there is no need of applying a sharp change on Max_p. Clearly, using more elaborate dynamic adjustments and allowing for modifications of all the three parameters should offer the possibility of performance improvements on ARED. To this end, we propose a scheme which adapts RED parameters by using target-oriented adjustments which are a function of the distance to the performance objective. Simulations results show that our scheme can stabilize the queue size, keep it away from buffer overflow and buffer underflow independently of the number of connections and can also achieve a more predictable and lower average queue size without substantially increasing the loss rate.

We describe in Section II a new algorithm that adjusts dynamically RED parameters. We evaluate this algorithm in Section III by extensive simulations under `ns` [16]. We conclude and propose future works in Section IV.

II. Adapting RED Parameters

We present an adaptive algorithm (named PSAND) that aims at *minimizing the variance* of the instantaneous queue size and improve the stability of ARED under the following constraints:

- achieve a specified target average queue size chosen *a priori* by the network operator as a trade-off between link utilization and delay.
- avoid an excessive increase of the packet loss rate as compared to the original RED and ARED.

To achieve this goal, we adapt Max_p with a multiplicative factor which is computed dynamically as a function of the changes in the average queue length making it more or less aggressive. Our approach differs from the algorithms of [6], [8] in the following points:

- An increase (resp. a decrease) in the average queue size by a certain factor is directly translated into an increase (resp. a decrease) of Max_p by the same multiplicative factor. This multiplicative factor is the ratio of the value of the weighted average queue size in the current interval (denoted by $\hat{K}_{cur}$) over the value of the weighted average queue size in the previous interval (denoted by $\hat{K}_{prev}$). We denote by $change\_rate$ this ratio that measures the queue length change:

$$change\_rate \;=\; \hat{K}_{cur} \;/\; \hat{K}_{prev} \;.$$

- In addition, we follow the approach of Floyd *et al.* in [6] that tends to bring the average queue size to a specified target value. Our aim is to bring it closer to its target value more quickly. For this purpose, we measure the gap between the current average queue size and its target value and increase or decrease Max_p proportionally so as to reduce this gap. Hence Max_p is adapted by a multiplicative factor which is a ratio of the weighted average queue size in the current interval ($\hat{K}_{cur}$) over the target average queue size (denoted by $\hat{K}_T$). We denote by $prox\_rate$ this ratio that measures the proximity of $\hat{K}_{cur}$ as compared to $\hat{K}_T$:

$$prox\_rate \;=\; \hat{K}_{cur} \;/\; \hat{K}_T \;.$$

  An average queue size lower (resp. larger) than its target value triggers a decrease (resp. an increase) of Max_p.
- We adapt Max_p using a rate which is a function of these two ratios representing a trade-off between the two increasing/decreasing trends. We denote this rate by r and compute it as follows:

$$r \;=\; prox\_rate \times change\_rate \;=\; \hat{K}_{cur}^2/(\hat{K}_T \times \hat{K}_{prev}) \;.$$

As illustrated in Fig. 1 these two ratios are measured once every time interval at which Max_p is adapted (0.5 seconds). $change\_rate$ measures the queue length change by comparing the weighted average queue size in the current interval ($\hat{K}_{cur}$) to its value in the previous interval ($\hat{K}_{prev}$). $prox\_rate$ measures the gap between $\hat{K}_{cur}$ and $\hat{K}_T$ (10 packets).

According to the results of our experiments, if we used directly r as a multiplicative factor of Max_p, our mechanism would not be aggressive enough in case of a severe congestion. In order to make the change rate more or less aggressive and perfect our model, we modified the influence of r by using a simple functional form. We set:

$$\beta \;=\; coef \times r^{\gamma} \;,$$

and let the adaptive scheme be:

$$Max_p \;=\; Max_p \times \beta \;. \qquad (1)$$

Indeed, the queue length does not allow alone to determine the severity of congestion that is the number of flows [17]. For instance a single highly active source can cause a persistent queue as well as large number of flows can. Hence, since the mechanism that uses directly r, measures the queue length change, it can not be able to detect efficiently the degree of severity of the congestion and can not therefore respond aggressively enough. Moreover, as stated in [8], congestion notification does not directly depend on the number of connections multiplexed across the link. Nevertheless, congestion notification should be given at a rate which is high enough to avoid packet loss due to buffer overflow and low enough to avoid underutilization of the link. The functional form (1) should be able to adapt Max_p with the aim of sending congestion notifications at a rate that reflects better the change of the congestion level.

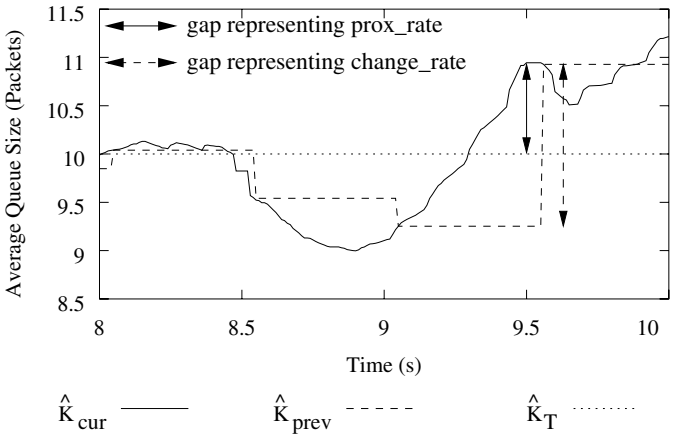


Fig. 1. Measuring the Traffic Change.

Algorithm 1 summarizes our scheme. The setting of the parameters Min_{th}, Max_{th}, $coef$ and γ used by this algorithm is discussed briefly in Section III-C.

Algorithm 1 PSAND Algorithm.

Set $\hat{K}_T$ as a function of the delay target.
$INTERVAL \leftarrow 0.5$ seconds.
/* Adaptation of Max_p. */
for each INTERVAL **do**
  prox_rate $\leftarrow \hat{K}_{cur} \;/\; \hat{K}_T$.
  change_rate $\leftarrow \hat{K}_{cur} \;/\; \hat{K}_{prev}$.
  $\beta \leftarrow coef \times (\text{prox\_rate} \times \text{change\_rate})^{\gamma}$.
  $Max_p \leftarrow \max(\underline{Max_p}, \min(Max_p \times \beta, \overline{Max_p}))$.
  $\hat{K}_{prev} \leftarrow \hat{K}_{cur}$.
end for
/* Upper Max_p : $\overline{Max_p} = [0.5, 1]$. */
/* Lower Max_p : $\underline{Max_p} = 0.01$. */

III. Stability and Performance Measures

A. Simulation Settings and Metrics

For a better comparison with [6], [8] respectively denoted $ARED_{Floyd}$ and $ARED_{Feng}$, we used the network topology they proposed in order to evaluate our scheme through simulations. Refer to Fig. 2 for the topology and link settings. The

TABLE I

SETTING OF THE SIMULATIONS PARAMETERS.

	Min_{th}	Max_{th}	$InitialMax_p$	K_T	α	β	$coef$	γ	Interval
ARED$_{Feng}$	5	15	0.02	–	3	2	–	–	–
ARED$_{Floyd}$	5	15	0.1	10	0.01	0.9	–	–	$0.5s$
PSAND	0	20	0.1	10	–	–	1.75	1.5	$0.5s$
RED	5	15	0.1	–	–	–	–	–	–

Fig. 2. Network Topology.

configuration uses TCPSack connections that generate long-lived FTP traffic from node S_1 to S_3 at time 0 second, from node S_3 to S_1 at time 1 second and from node S_2 to S_3 at time 3 seconds. The TCP congestion window size is 15 packets and the packet size is 1600 bytes. We added a number of connections from node S_1 to S_3 according to the simulation. For these additional connections, we set the TCP window size to 20 and the packet size to 1000 bytes. For ARED$_{Feng}$ and ARED$_{Floyd}$, we took the values of the simulation's parameters they recommended in [6], [8]. The upper bound of Max_p for PSAND is set to 0.75 and for ARED$_{Floyd}$ it is set to 0.5. For all simulations, we follow the approach of [6], [7], [18] to set the average queue size weight (ω_q) as a function of the link bandwidth, according to the formula $\omega_q = 1 - \exp(-1/C)$, where C is the link capacity in packets/second. Refer to Table I for the other parameters settings.

B. Performance Results

1) Qualitative observations: The experiments conducted in this section use one simulation lasting 100 seconds where 20, 70 and 100 additional flows started every 0.1 seconds from node S_1 at time 50 seconds. We make a visual and qualitative comparison of the evolution of the instantaneous and weighted average queue size through Fig. 3 and 4. These figures allow us to observe not only the behavior of each mechanism in presence of a high and a low traffic load but also how each mechanism reacts in response to a change of the congestion level.

We can observe in Fig. 3(c) that our mechanism responds more quickly to a rapid change in the congestion increase and shows a better adaptation to this change. Indeed, after a sharp change at time 50 seconds, it needs less than 5 seconds to bring the average queue size back down to its target value (10 packets) whereas ARED$_{Feng}$ in Fig. 3(a) takes roughly 15 seconds and ARED$_{Floyd}$ in Fig. 3(b) takes 10 seconds. For a higher traffic load increase (70 additional flows) observed in Fig. 4, ARED$_{Feng}$ and ARED$_{Floyd}$ are unable to control the average queue size. Moreover, it displays a reduced amplitude of queue size oscillations. It also shows less frequent occurrences of an empty queue, and also less frequent occurrences of a queue size close to the maximum buffer capacity.

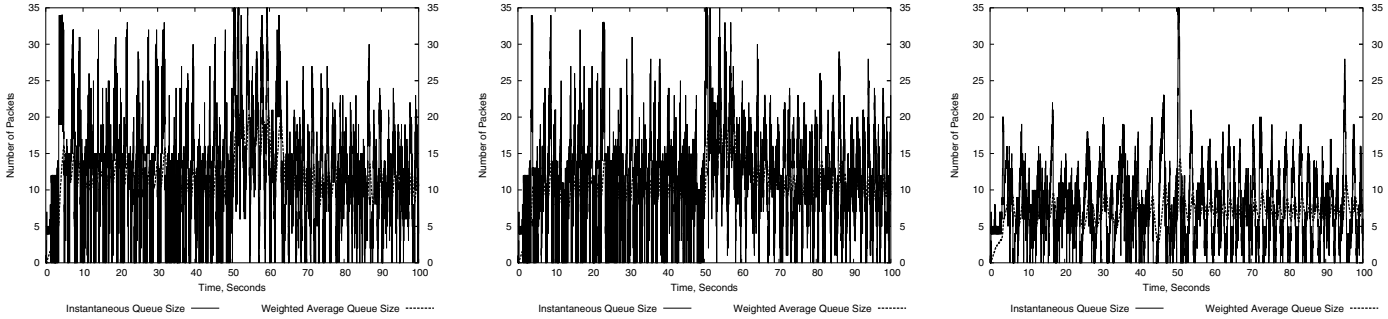
2) Statistical observations: To consolidate all these observations, we conducted more rigorous statistics by measuring the following metrics:

- the delay jitter represented by the variance of the instantaneous queue size (Fig. 5(a)).
- the average queueing delay inferred by the true average queue size, *i.e.* the average of the instantaneous queue size (Fig. 5(b)).
- the packet loss rate (Fig. 5(c)).
- the distribution of the instantaneous queue size (the probability that the queue is empty, full or very close to the target queue size (Fig. 6)).

One point on each curve of Fig. 5 and 6 is the average of 100 independent simulations where additional flows start at time 3.5 seconds every 0.1 seconds from node S_1. For instance, for 20 additional flows, the first additional flow starts at time 3.5 seconds and the 20th at 5.4 seconds.

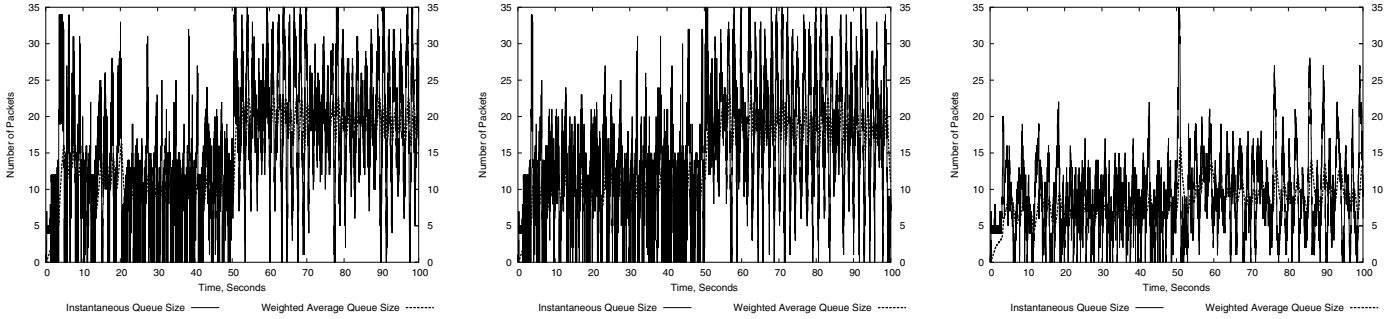
As shown by Fig. 5(a) and Fig. 5(b), PSAND reduces the queueing delay and the queue size variance, which increases the stability of the queue size, independently of the number of flows as compared to RED and the other versions of ARED. We also observe in Fig. 5(c) that PSAND did not sacrifice the loss rate for this performance gain. We rather observe a slight reduction of the packet loss rate as the number of flows increases. Moreover, as illustrated by Fig. 6(a) and 6(b), as the number of flows increases, the probability that the queue might be empty ($P(X = 0)$) and also that the probability that the queue might be full ($P(25 < X \leq 35)$) is reduced as compared to the other mechanisms. Since the link utilization can be computed as $1 - P(X = 0)$, our scheme increases the link utilization rate. In Fig. 6(c), we also observe that under PSAND, the probability that the queue size is close to its target value is generally larger. For a small load, this is not true, but then the average queueing delay is less than the target delay, which is more interesting than having a delay close to the target one as long as this is not payed in return by an increase of the packet loss rate.

Finally, the qualitative and statistical analysis of all the results show that our approach offers a desirable overall improvement of ARED performances and achieves our goal



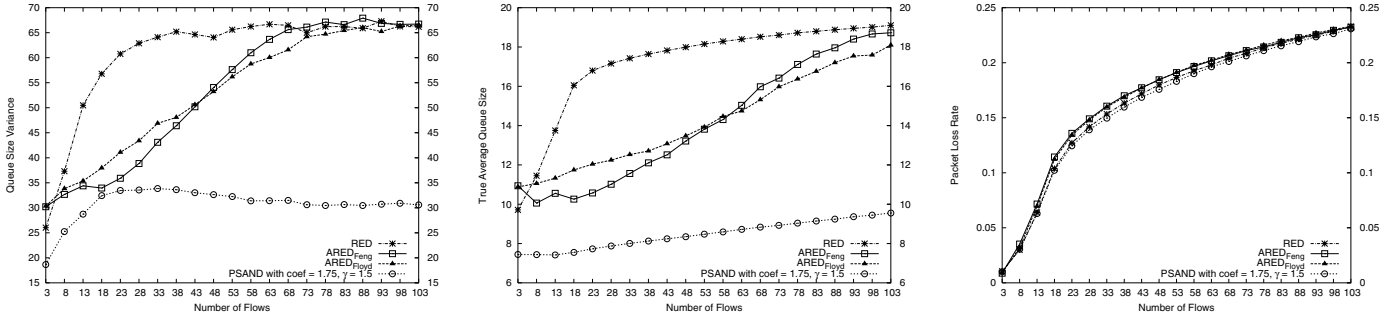
(a) ARED$_{Feng}$. (b) ARED$_{Floyd}$. (c) PSAND.

Fig. 3. Evolution of the Instantaneous and the Weighted Average Queue Size (for 20 additional flows).



(a) ARED$_{Feng}$ (70 additional flows). (b) ARED$_{Floyd}$ (70 additional flows). (c) PSAND (100 additional flows).

Fig. 4. Evolution of the Instantaneous and the Weighted Average Queue Size.



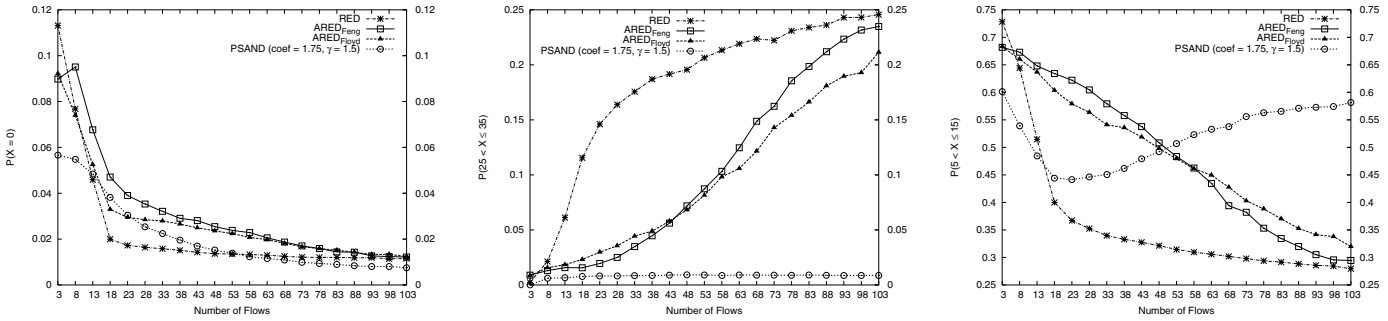
(a) Queue Size Variance. (b) True Average Queue Size. (c) Packet Loss Rate.

Fig. 5. Comparison of Performances for Different Numbers of Flows.

of reducing the queue size variance while keeping the average queue size close to its target value without sacrificing the packet loss rate.

C. Configuration of Min_{th}, Max_{th}, $coef$ and γ

We investigated the influence of Min_{th}, Max_{th}, $coef$ and γ on our adaptive scheme through the experiments fully reported in [19]. The results suggest to choose a range of values of $coef$ and γ which is appropriate to small and large number of flows. Hence, based on extensive simulations, we set $coef = 1.75$ and $\gamma = 1.5$ as this setting brings about reasonable overall improvement on ARED. The results showed also that by using an automatic configuration of $coef$ and γ according to the traffic load, the performance of our adaptive scheme can be further improved. In addition, the results showed also that by starting the random drop earlier (very small Min_{th}) at a rate which reflects closely the evolution of the queue length, our scheme scatters the packets drops over time and maintains a slower and gradual build up of the queue



(a) $P(X = 0)$. (b) $P(25 < X \leq 35)$. (c) $P(5 < X \leq 15)$.

Fig. 6. Distribution of the Instantaneous Queue Size.

reducing the oscillations of the queue length as well as the average queue size. The results showed also that large values of Max_{th} increases highly the queueing delay and overall good performances are obtained for $Max_{th} = 2 * \hat{K}_T$. Based on these results, we configured Min_{th} and Max_{th} so as they remain symmetrical about the target queue size ($\hat{K}_T$) while being as far apart as feasible depending on the value of $\hat{K}_T$ as compared to the buffer capacity. For B representing the buffer capacity, we set these two parameters as follows:

- If $\hat{K}_T > B/2$ then $Min_{th} = 2\hat{K}_T - B$ and $Max_{th} = B$.
- If $\hat{K}_T \leq B/2$ then $Min_{th} = 0$ and $Max_{th} = 2\hat{K}_T$.

IV. Conclusion and Future Work

This paper has described a mechanism for RED parameters setting in order to enhance the effectiveness of RED. Our work is based on the adaptive approach of RED described in [6], [8] since this approach does not require any hypothesis on the type of traffic and therefore reduces the dependency of RED on traffic scenario input parameters. We do not claim to present an optimal setting of RED parameters but rather show among other possible alternatives, a way to improve the performances of Adaptive RED. Unlike in [6], [8] where a constant factor is used, our mechanism adapts Max_p with a dynamic rate which is a function of the change in the average queue size and the distance of the average queue size to the specified target queue size. In addition we conclude from simulations that Min_{th} should be set as small as possible, whereas Max_{th} should be set as large as feasible. Indeed, the results showed a reduction of the variance of the instantaneous queue size as well as the average queue size independently of the number of flows, without increasing and even in some cases (for large number of flows for instance) while decreasing the loss rate. Moreover, our mechanism keeps the queue size away from buffer overflow and buffer underflow.

We are currently investigating a way to choose dynamically some of the parameters of our algorithm for a further improvement. In particular, selecting a value of $coef$ and γ that gives a best performance is an important issue. In addition, we are comparing our scheme with other mechanisms such as [11]–[14] and also studying its stability by using the models proposed in these papers.

References

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

[2] B. Braden and al., *Recommendations on Queue Management and Congestion Avoidance in the Internet (RFC 2309)*, IETF, April 1998.

[3] M. May, T. Bonald, and J. Bolot, "Analytic evaluation of RED performance," in *Proc. IEEE INFOCOM'00*, 2000.

[4] T. Ott, T. Lakshman, and L. Wong, "SRED: Stabilized RED," in *Proc. IEEE INFOCOM'99*, 1999.

[5] V. Misra, W. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," in *Proc. SIGCOMM'00*, 2000.

[6] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: an algorithm for increasing the robustness of RED's active queue management," August 2001, available at http://www.icir.org/˜floyd.

[7] T. Ziegler, S. Fdida, C. Brandauer, and B. Hechenleitner, "Stability of RED with two-way TCP traffic," in *IEEE ICCN*, October 2000.

[8] W. Feng, D. Kandlur, D. Saha, and K. Shin, "A self-configuring RED gateway," in *Proc. IEEE INFOCOM'99*, March 1999.

[9] V. Firoiu and M. Borden, "A study of active queue management for congestion control," in *Proc. IEEE INFOCOM'00*, Tel Aviv, Israel, 2000.

[10] C. Hollot, V. Misra, D. Towsley, and W. Gong, "A control theoretic analysis of RED," in *Proc. IEEE INFOCOM'01*, 2001.

[11] ——, "On designing improved controllers for AQM routers supporting TCP flows," in *Proc. INFOCOM'01*, Anchorage, Alaska, April 2001.

[12] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management," in *Proc. SIGCOMM'01*, San Diego, CA, August 2001, pp. 123–134.

[13] S. Athuraliya, V. Li, S. Low, and Q. Yin, "REM: Active queue management," in *IEEE Network*, vol. 15, May/June 2001, pp. 48–53.

[14] C. Wang, B. Li, Y. Hou, K. Sohraby, and Y. Lin, "LRED: a robust active queue management scheme based on packet loss ratio," in *Proc. INFOCOM'04*, Hong Kong, March 2004.

[15] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Techniques for eliminating packet loss in congested TCP/IP networks," University of Michigan, Tech. Rep. CSE-TR-349-97, November 1997.

[16] "Ns simulator homepage," 2003, http://www.isi.edu/nsnam/ns.

[17] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Blue: a new class of active queue management algorithms," University of Michigan, Tech. Rep. UM CSE-TR-387-99, 1999.

[18] V. Jacobson, K. Nichols, and K. Poduri, "RED in a different light," Cisco system, Tech. Rep., 1999.

[19] T. Alemu and A. Jean-Marie, "Dynamic configuration of RED parameters," LIRMM, University of Montpellier II, Tech. Rep. 03-042, December 2003, http://www.lirmm.fr/˜tigist/papers/RR01-LIRMM.pdf.