Congestion Avoidance Using Adaptive Random Marking

Marissa Borrego, Na Li, Gustavo de Veciana, San-qi Li
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, Texas
(marissa, gustavo, sanqi)@ece.utexas.edu
lina.li@santera.com

Abstract— **Recent work on congestion control in TCP/IP networks combines improved end-user transmission mechanisms with active queue management schemes. In this paper we propose MARS, an adaptive early packet marking active queue management mechanism. Unlike previous proposals, MARS adapts the marking probability to drive the average queue length towards a target size. By doing so, MARS implicitly reacts to changes in the network dynamics without requiring per-flow state or estimates thereof. Our simulations confirm MARS's flexibility and effectiveness in reducing packet loss, controlling queuing delay, and improving utilization and flow goodput.**

Keywords—**RED, TCP, congestion control, active queue management**

I. Introduction

The success of the Internet is in part due to its simple connectionless architecture. However, its stability relies on proper flow and congestion control mechanisms. These mechanisms, such as those found in TCP, play a critical role in reducing packet losses and avoiding congestion collapse [8]. The basic congestion control principle underlying TCP is for a flow to increase its transmission rate until it observes packet loss or high delays. It then aggressively backs-off to alleviate possible congestion [1]. In this manner, TCP attempts to gauge bandwidth availability and avoid network instability without requiring implicit feedback from the network.

While the flow and congestion control mechanisms of TCP and its variants have been successful, there is increased recognition for the need to improve upon them [15]. In particular, more proactive steps are needed to not only recover from periods of congestion but to attempt to avoid congestion and further improve network utilization [9]. This has prompted the IETF to recommend use of both active queue management within network routers [2] in conjunction with enhancing TCP flows using Explicit Congestion Notification (ECN) [16].

ECN allows for explicit congestion notification using packet marking instead of packet loss. Active queue management attempts to improve performance by monitoring the network router queues. Its goals include avoiding congestion, reducing packet loss, improving utilization and controlling queuing delay. Several schemes have been proposed based on probabilistic packet dropping/marking including Random Early Detection (RED) [6], Stabilized-RED (SRED) [13], Flow-Proportional Queuing (FPQ) [12], and BLUE [3].

This paper presents MARS, Marking based on an Adaptive Random Scheme. Unlike previous active queue management mechanisms, MARS intends to control the aggregate sending rate by bring the *average* queue length to a target size. It adapts the marking probability by observing queue deviations from the target. This allows MARS to *implicitly* gauge the sensitivity of user transmissions to the random marking, without requiring per-flow state, or estimates thereof.

MARS only requires the configuration of a few simple parameters and, unlike other schemes, its performance is highly insensitive to their specific values. Its overhead is small, based on simple periodic updates. The simulation results demonstrate MARS effectively stabilizes the queue performance, increases throughput, reduces loss, and controls queuing delays. It outperforms both RED and SCRED and results in more consistent behavior. Overall, MARS provides a flexible and effective counterpart to the end system congestion control mechanisms.

The paper is organized as follows: §II describes the MARS adaptation algorithm and discusses several design and implementation considerations. The simulation results and comparisons to RED and SCRED are found in §III. §IV describes related works. Finally, §V concludes the paper and discusses future work. For simplicity, the remainder of the paper discusses packet marking and assumes support for ECN. We also assume the reader is familiar with TCP and ECN congestion control mechanisms [1], [16], as well as RED and its variants [6], [4].

II. MARS Adaptation

This section describes MARS, an adaptive random packet marking scheme. We begin by motivating our approach, then present the adaptation algorithm followed by various design and implementation considerations. The goal for MARS is to achieve full utilization of the bottleneck link while avoiding packet loss and controlling queuing delay. It adapts the marking probability $p(t)$ at time t to drive the *average* queue length towards a given target q^*. The key advantage of using a random marking scheme is simplicity. It is based on the notion of "fairly" distributing congestion indications among an unknown set of ongoing transmissions, while avoiding flow synchronization, and without requiring routers to keep per flow state.

However, using a *random* marking scheme, in conjunction with an end-user mechanism such as TCP/ECN, results in unpredictability[1]. This, in conjunction with other dynamic factors such as number of flows, multiple bottlenecks, network delays, traffic burstiness, and capacity changes, results in unavoidable queue fluctuations.

It is reasonable to model the queue length as a random process whose characteristics depend on the marking probability. Thus, while it makes no sense to consider driving the *instantaneous* random queue length to the target, one can nevertheless consider adapting the marking probability so that the queue fluctuations occur about the target q^*. As we have found, this is in fact a highly effective approach for managing the queue in order to control the fluctuations to avoid queue underflow and overflow. Based on this idea, MARS is able to improve utilization, reduce packet loss, and control queuing delay.

A. MARS Algorithm

The goal for MARS is to bring the *average* queue to a target value, q^*. This is accomplished by periodically adapting the packet marking probability $p(t)$ based on the deviation of the *instantaneous* queue $q(t)$ from the target q^*. The rationale for selecting the instantaneous queue is discussed in §II-D. The algorithm is based on the following commonsense guidelines:

1. if $q(t) > q^*$ increase $p(t)$; if $q(t) < q^*$, decrease $p(t)$;
2. update $p(t)$ in a manner such that "small" (normal) deviations between $q(t)$ and q^* are inconsequential while "larger" deviations lead to aggressive updates.

The rationale for the first point should be obvious from the observation that the arrival rate, and thus queue length, will decrease with an increase in packet marking probability and vice versa. The second point follows from the observation that normal queue fluctuations will occur

[1] The effect of a marked packet depends upon the user's transmission "rate". Marking a packet of a user with a very large rate will have a greater impact than that of a user with a small rate.

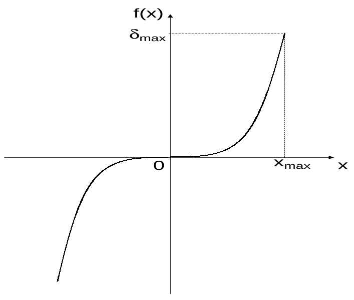

Fig. 1. Delta Update Function - f(x)

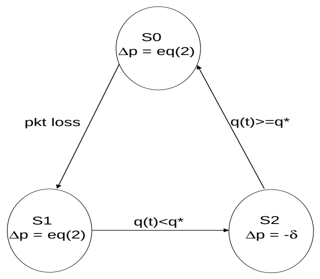

Fig. 2. Packet Loss State Machine

due to dynamic nature of the network. Thus, small queue deviations from the target value should be tolerated. However, larger deviations should lead to the proper adaptation of $p(n)$. The larger the deviation the more aggressive the change.

MARS adapts the packet marking probability $p(t)$ periodically using the following equations. Selection of an update period is discussed in §II-D:

$$\begin{aligned} p(t) &= p(t-1) + \Delta p(t), & (1)\\ \Delta p(t) &= s(t) \times f(q(t) - q^*), & (2)\\ s(t) &= \max[p(t-1), \delta] & (3) \end{aligned}$$

where the function $f(x)$ determines the magnitude of the update and is asymmetric, increasing, and convex on the positive orthant. The function $s(t)$ scales the update to the proper range, with the minimum scaling value $\delta > 0$. This choice for $s(t)$, as discussed in §II-B, can achieve better "stability" for the mechanism around equilibrium.

We have considered various candidate functions $f(x)$ and found that the proposed mechanism is quite robust to this choice. For convenience we use the following function, shown in Fig.1:

$$f(x) \quad = \quad \mathrm{a} \times sgn(x) \times (e^{b|x|} - 1) \tag{4}$$

where $a, b > 0$ are constants. The selection of a and b determine δ_{max}. The value for δ_{max} has practical implications and is discussed in §II-D. This function satisfies our requirements. When the deviation $|q(t) - q^*|$ is small, the update $\Delta p(t)$ will be relatively small. Yet, a large deviation from q^* will lead to an exponentially larger update. This helps MARS react aggressively to significant change while tolerating normal queue fluctuations.

MARS is simple to implement. The computing overhead is small since the adaptation is performed periodically. At each update interval, only a few simple operations are required. $f(t)$ can be quantized, precomputed and stored in table format, and thus only entails a table lookup at run-time. Next, in §II-B we discuss the proposed scaling factor for the update, followed by an additional improvement to render the mechanism robust to the influence of packet losses due to overflow, in §II-C.

B. On the update scaling factor $s(t)$

In this section we assume the packet marking probabilities are small, i.e., the average time between marks is large relative to the flow round trip times. We further assume that in this regime the dynamics of TCP with and without support for ECN are similar. Consider the basic model relating the average throughput r_i of an active TCP/ECN flow i, sharing a router buffer with a small packet dropping/marking probability p, see [11]:

$$r_i = \frac{\alpha}{\sqrt{p} \times \mathrm{RTT}_i} \tag{5}$$

where α is a constant and RTT$_i$ denotes its round trip time. Taking the derivative one can show that

$$dr_i = -\frac{1}{2} \times r_i \times \frac{dp}{p}. \tag{6}$$

Furthermore, letting r denote the sum of the average throughput of flows sharing the buffer, then based on Eq.(6) one can assess the sensitivity of the overall average throughput to a change in the marking probability:

$$dr = -\frac{1}{2} \times r \times \frac{dp}{p}. \tag{7}$$

Rearranging to express in terms of dp:

$$dp = -2 \times p \times \frac{dr}{r}. \tag{8}$$

Now suppose the MARS mechanism has roughly converged, meaning that the average queue length is on target, the aggregate arrival rate is approximately equal to the link capacity c and the marking probability is stable:

$$p(t) \approx p \ \ , \bar{q}(t) - q^* \approx 0, \ \ r \approx c.$$

Further suppose there is a small exogenous positive disturbance to the arrival rate Δr. Based on Eq.(8), in order to keep the arrival rate ($\Delta r - dr = 0$) and queue on target, the marking probability should be increased to reduce the arrivals from TCP connections by Δr. The corresponding change Δp to p is roughly

$$\Delta p \ = \ 2 \times p \times \frac{\Delta r}{c} \ \propto \ p \times \Delta r. \tag{9}$$

Assuming the queue is not idle, the disturbance in arrival rate translates to a proportional deviation from the target, $\Delta q \propto \Delta r$. By linearizing Eq.(2) for small queue distrubances, we see that this deviation results in a marking probability update given by

$$\Delta p \ \approx \ ab \times p \times \Delta q \ \propto \ p \times \Delta r.$$

Thus, by introducing the scaling factor $s(t)$ in Eq.(2) one ensures an update scales as suggested by Eq.(9). Intuitively, when the nominal value of p is small, a few users share the link capacity, and a small change in p, i.e. Δp, may result in a big change in throughput. In this regime updates should be made cautiously, i.e. scaled by the small value of p. Note that if the scaling factor did not include the maximum with δ then the marking probability would remain at 0 if ever $p(t-1) = 0$. The proposed scaling factor precludes this pathology.

C. Controlling impact of packet loss

The issue of accounting for the influence of packet loss due to queue overflow is discussed in this section. Queue overflow can occur when the marking probability is too low given the current network dynamics, e.g., due to a drastic increase in number of active flows. With MARS,

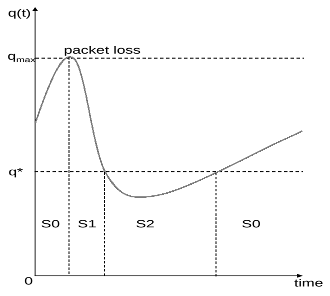

Fig. 3. State Transition Queue Behavior

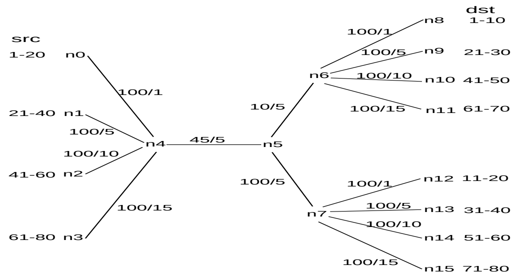

Fig. 4. Topology: 2 Bottleneck Links, RTT from 24ms-80ms

the updates are based on the assumption that on average changes in the queue reflect previous adaptations of $p(t)$.

With overflows, however, a dramatic decrease in queue size is probably the result of the packet losses and not of the marking probability. In all likelihood, $p(t)$ is indeed too low. However, under these circumstances MARS may incorrectly decrease $p(n)$ when the queue drops below q^*. This can lead to subsequent queue overflow, new losses and recurrent underflow. During such oscillations MARS may fail to set the marking probability to a stable value. In order to control this effect, we propose a state machine shown in Fig.2. The idea is to reduce the impact of the changes in the queue triggered by packet loss on the adaptation of $p(t)$.

Referring to Fig.2, S0 is the state of normal operation, wherein updates are based on Eq.(2). Upon detecting packet losses the system transits to state S1. It continues updating $p(t)$ in the normal fashion until the queue decreases below the target q^*, at which point the system transits to S2. In S2 the update equation is modified to $\Delta p(t) = -\delta$. The system returns to state S0 from S2 once the queue length exceeds q^*. Fig.3 depicts the corresponding state transitions.

Using state S2 to control the decrease of $p(t)$ serves to dampen the effect of the reduction in queue size caused by packet loss. By decreasing $p(t)$ we insure the transition back to S0, even in the unlikely event that a large number of flows suddenly depart. The simulation results confirm the effectiveness of this simple mechanism in controlling the impact of queue overflow and eliminating oscillation. In general, packet loss oscillations are an important consideration for other adaptive marking schemes. Our results and those in [3] demonstrate that both SCRED and BLUE are susceptible to this type of oscillation.

D. Additional design considerations

This section presents several important design and implementation choices. First, use of the instantaneous queue and p averaging are addressed. Next, selection of the update interval is considered. Finally, we discuss the ratio of the minimum to maximum update size.

As previously mentioned, MARS must deal with normal queue fluctuations. One obvious choice for filtering the noise is to use the average queue $\bar{q}(t)$. The problem with using $\bar{q}(t)$ with the proposed adaptation is that $\bar{q}(t)$ is unable to reflect the impact of changes caused by control decisions in a timely manner. This may lead control in the wrong direction. For example, by increasing the marking probability, the queue size starts to drop below the target, but the average queue size is still above the target. This causes further updates in the wrong direction. As our experiments confirmed, waiting for the average queue to properly reflect the current state reduces the algorithm's ability to properly react and significantly compromises its performance.

Instead, MARS uses two mechanisms to help filter the impact of normal queue fluctuations from the control. First, as previously mentions, the delta function $f(t)$ is designed to be less sensitive to change close to the target. Second, averaging $p(n)$ over a short time interval, rather than $q(n)$, further reduces the noise without compromising performance. These two mechanisms lead to a highly effect solution for reducing the impact of normal queue fluctuations.

Next, we discuss setting the update interval. Recall MARS periodically updates the marking probability using Eq.(2). The effect of an update impacts the queue sometime in the future, on the order of the flow RTTs. Therefore, it is reasonable to set the update interval in this range. Updating much more frequently does not allow the algorithm to accurately sense the impact of previous change(s). Conversely, updating much slower effects convergence. Our simulations indicate MARS is fairly insensitive to the specific update interval value, as long as it is in a reasonable range. Given flows with multiple RTTs, we recommend setting it close to the average RTT.

Finally, we discuss setting δ_{max} for function $f(t)$ in Eq.(4). This determines the magnitude of maximum update. If this value is very small convergence to the proper range may be lengthy. However, an extremely large value may result in overshooting followed by backtracking, and thus also exhibit poor convergence. Through experimentation we found performance was best with a reasonable value, in the 0.1-0.3 range.

III. Simulation Results

We studied the effectiveness of MARS and compared its performance to RED [6] and SCRED [4]. ns version 2 [17] was used to run the simulations. A variety of scenarios were investigated including single and multiple bottleneck links, similar and different RTTs, and dynamically changing flow number. Here we present the results for a dynamic case where the number of active flows abruptly changes from 80 to 40, using different RTTs and two bottleneck links.

Note that MARS' performance was consistent for all cases, without the need for any parameter reconfiguration. This was not true for RED and SCRED. Their performance was dependent on the specific network dynamics and highly sensitive to their parameter settings. It was necessary to adjust the parameter values according to the different scenarios. Here we present the best results that were obtained for RED and SCRED.

The simulation topology is shown in fig. 4. Links are full duplex with equal bandwidth in both directions. All are 100Mbps links except for the two bottlenecks, link [n4-n5] at 45Mbs and link [n5-n6] at 10Mbs. The link propagation delays vary from 1ms to 15ms, in increments of 5ms. The values for each are labeled in fig. 4. The maximum queue size is 150 packets.

There are four source nodes, n0-n4, each with 20 TCP sources. There are two sets of destination nodes. To reach the first set, n8-n11, flows must traverse both bottleneck links. The second set, n12-n15, only entails a single bottleneck link, [n4-n5]. Each source node divides its TCP flows evenly between the two sets of destination nodes. The TCP flows are all infinite ftp sources that adapt their sending rate according to Reno [1].

The simulation is run for 160 seconds. 40 flows run for the entire duration. The other 40 flows are turned on and off every 40 seconds. All

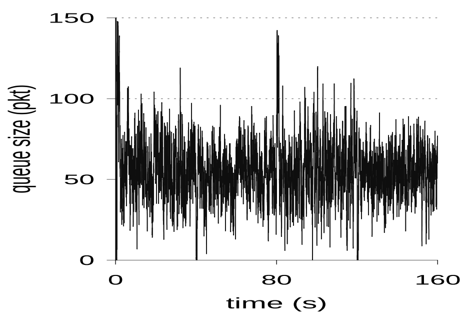

Fig. 5. MARS Queue Performance (q^*=50, interval=70ms

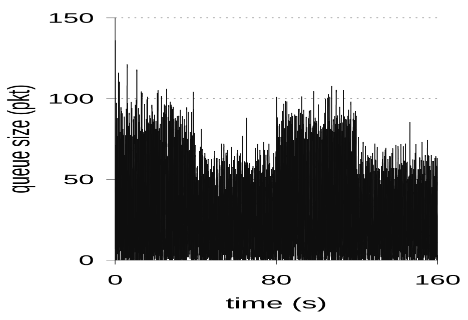

Fig. 6. RED Queue Performance (maxp=0.1, minth=20, maxth=80)

80 flows are started initially. We tested MARS, RED, and SCRED using the following parameter settings. For MARS, the target queue size q^*=50 and the update-interval=70ms. For RED, the maximum marking probability $maxp$=0.1, the minimum threshold $minth$=20, and the maximum threshold $maxth$=80. Finally for SCRED, the minimum threshold $minth$=20, the maximum threshold $maxth$=80, a=3, and b=2. A high-level description of RED and SCRED is found in §IV.

The queue performance results for MARS, RED, and SCRED at the first bottleneck link [n4-n5] are shown in figs. 5, 6 and 7, respectively. For brevity, we omit results for the second bottleneck link, as they present similar performance.

MARS delivers significantly more consistent and controlled behavior than RED or SCRED. It is able to stabilize the queue fluctuations around the target. It reduces incidences of queue overflow and underflow. Performance is consistent regardless of the number of flows.

RED's performance is less predictable and depends on the number of active flows. The queue fluctuations are less controlled and result in more underflow. As we shall see, RED must turn to packet loss to control the queue size in the case of 80 flows.

The performance for SCRED is even less consistent than RED. The queue grows larger and oscillates more irregularly in the case of 80 flows. Here, SCRED experience significant packet dropping and underflow. However, the results for 40 flows are much better, with performance similar to that of MARS. In this case, SCRED is able to stabilize the oscillations and reduce both packet loss and underflow.

Comparing the marking probabilities helps explain the differences in queue performance. In fig. 8, we see MARS achieves precise tuning of the marking probability. It is able to adapt its magnitude to the correct range and to stabilize the value. By matching the marking probability to the flow dynamics, MARS can control the aggregate sending rate in a stable manner.

RED's marking probability fluctuates over the entire range, from $0..maxp$, as seen in fig.9. There is very little difference between the values used for 80 and 40 flows. Yet, as shown using MARS, the marking probabilities should operate in different ranges. In addition, RED must rely on packet discard to control the queue in the case of 80 flows. Dropped packets were logged with a marking probability of 1.0. In the graphs, these can be seen as the vertical lines that shoot to the top. Overall, RED introduces packet loss and fails to match the marking probability with the flow dynamics.

SCRED is better than RED at matching the marking probability. Although it's value still fluctuates between $0..maxp$, by adapting $maxp$, it is able to bring the oscillations within the proper range. In the case of 40 flows, the marking probability fluctuates in a range similar to MARS. Yet, performance is much worse with 80 flows. Here, it fails to stabilize $maxp$ and introduces packet discard. The instability can be attributed to the fact that SCRED fails to account for the impact of packet loss on performance. This type of behavior is described in §II-C.

We logged the average goodput for the 80 flows over the first 35

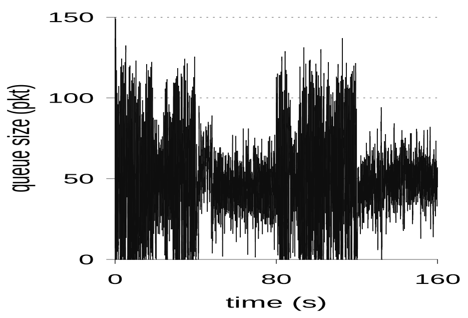

Fig. 7. SCRED Queue Performance (minth=20, maxth=80)

seconds of the simulation. The total average goodput plus bottleneck link utilization and loss rate at n4-n5 for MARS, RED and SCRED are summarized in table I. MARS outperformed the other schemes. It achieved the highest goodput and utilization with the smallest loss rate. Especially notable is its significantly smaller loss rate.

Overall, MARS delivered the most consistent and stable behavior. By adapting the marking probability to stabilize the queue performance, it was able to reduce both overflow and underflow. Unlike the other schemes, MARS was able to control the queue size without relying on packet discard. Hence, it delivered the highest goodput and utilization with the lowest loss rate.

IV. Related Work

This section provides a brief overview of several active queue management schemes designed to regulate TCP flows based on random packet marking/dropping. Random Early Detection (RED) is the most prominent and widely studied scheme today [6]. RED detects incoming congestion based on monitoring the *average* queue length and uses random packet marking which linearly increases in this average. However, it uses a static linear function that limits the dynamic range over which RED can adapt the marking probability. Therefore, it may need to rely on packet discard to control the queue size. It is difficult to configure RED to perform well under different scenarios. Its performance varies depending upon the specific parameter settings and network dynamics.

Self-Configuring RED (SCRED) [4] is also based on a linear relation between *average* queue length and marking probability, but adap-

TABLE I
Bottleneck Link Performance (n4-n5)

	loss rate	utilization	goodput(Mb/s)
MARS	2.904×10^{-4}	0.9982	44.72
RED	1.064×10^{-2}	0.9631	41.93
SCRED	1.021×10^{-2}	0.9817	43.28

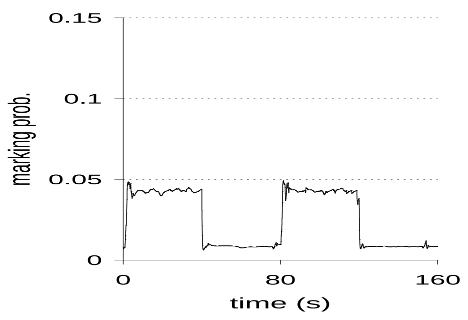

Fig. 8. MARS Marking Probability (q^*=50, interval=70ms)

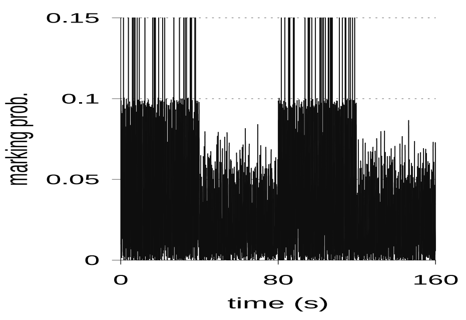

Fig. 9. RED Marking Probability (maxp=0.1, minth=20, maxth=80)

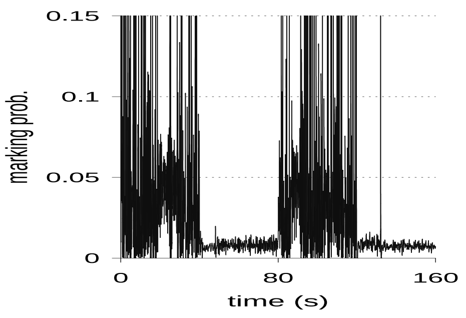

Fig. 10. SCRED Marking Probability (minth=20, maxth=80)

tively tunes its slope by adjusting the maximum marking probability $maxp$. $maxp$ is adjusted whenever the estimated average queue size is larger or smaller than the maximum or minimum threshold, respectively. However, SCRED can only adjust the parameter with a coarse granularity. Its performance is also sensitive to the other parameter settings. Furthermore, SCRED can oscillate since its adaptation does not account for the influence of packet loss, as evident in our simulation results.

An alternate scheme, Stabilized RED (SRED) [13], calculates the marking probability based on an estimate of the current number of active flows using hit probability. However, the flow number estimation is based on a noisy observation and is not always accurate, especially when faced with a large number of active flows. As the number of flows increases, the hit probability decreases and it is vulnerable to measurement error. Further, its performance is based on a somewhat arbitrary static three-step function and a maximum drop probability. SRED uses the *instantaneous* queue. Simulations show that while SRED is able to control the maximum queue size it experiences significant underflow[13].

A final scheme, BLUE [3], proposes increasing the marking probability $p(t)$ by a constant when experiencing queue overflow, and decreasing $p(t)$ by another constant when the queue is empty. BLUE is the closest to our scheme in the sense that both are based on adaptively tuning $p(t)$ by observing the *instantaneous* queue. However, BLUE is unable to stabilize queue fluctuations around a target. Further, it does not properly scale the updates according to $p(t)$. This presents a problem once $p(t)$ is close to the proper range. Convergence is also an issue since BLUE only updating $p(t)$ when the queue reaches an extreme. The simulation results in [3] show that with BLUE, $p(t)$ takes an extremely long time to converge and the queue experiences periods of oscillation between underflow and overflow, as described in §II-C.

V. Conclusion and Further Considerations

This paper proposes MARS, a novel adaptive random marking active queue management scheme. It uses a simple mechanism based on a target queue size to adaptively tune the marking probability $p(t)$ by monitoring the instantaneous queue. The design includes proper scaling of the updates to $p(t)$. It further accounts for the influence of packet loss on the adaptation scheme.

Through simulation, we demonstrated MARS' ability to control the queue performance and dynamically adjust to the network dynamics, including number of active flows and multiple bottleneck links. MARS effectively stabilized queue fluctuations around the target value and reduced both queue overflow and underflow. It outperformed RED and SCRED in terms of utilization, loss rate and goodput, and delivered the most consistent behavior.

MARS has several additional benefits. Its computing overhead is small, requiring only a few simple operations performed periodically. MARS is easy to configure, only involving the setting of a few parameters. It is fairly insensitive to their specific value as long as configured in a reasonable range.

Several issues for further consideration remain. We are still investigating a proper criteria for selecting a target queue size, q^*. It may be possible to design a mechanism to adaptively tune q^* based on queue variance. The variance might also be an effect mechanism in designing a function $f(x)$ that further ignores normal queue fluctuations. Finally, while use of the instantaneous queue provides an effective mechanism for adapting the marking probability, it may be possible to design an alternate, equally as effective, scheme based on more stable measures, such as $\bar{q}(t)$.

References

[1] M. Allman, V. Paxson, "TCP Congestion Control," IETF RFC 2581, April 1999.
[2] B. Braden, et al, "Recommendations on Queue Management and Congestion Avoidance in the Internet," IETF RFC 2309, April 1998.
[3] W. Feng, D. Kandlur, D. Saha, K. Shin, "BLUE: A New Class of Active Queue Management Algorithms," University of Michigan CSE-TR-387-99, April 1999.
[4] W. Feng, D. Kandlur, D. Saha, K. Shin, "A Self-Configuring RED Gateway," IEEE INFOCOM'99.
[5] S. Floyd, "Connections with Multiple Congested Gateways I Packet-Switched Networks Part 1: One-way Traffic," Computer Communication Review, vol. 21, no. 5, October 1991.
[6] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Transactions on Networking, August 1993.
[7] S. Floyd, "TCP and Explicit Congestion Notification," Computer Communication Review, vol. 24, no. 5, October 1994.
[8] V. Jacobson, "Congestion Avoidance and Control," In Proceedings of ACM SIGCOMM, August 1988.
[9] C. Lefelhocz, B. Lyles, S. Shenker, L. Zhang, "Congestion Control for Best-Effort Service: Why We Need a New Paradigm," IEEE Network, Juauary/February 1996.
[10] D. Lin, R. Morris, "Dynamics of Random Early Detection," In Proceedings of ACM SIGCOMM, September 1997.
[11] M. Mathis, J. Semke, J. Mahdavi, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithms," Computer Communication Review, 27(3), July 1997.
[12] R. Morris, "Scalable TCP Congestion Control," IEEE INFOCOM 2000.
[13] T. Ott, T. Lakshman, L. Wong, "SRED: Stabilized RED," IEEE INFOCOM'99.
[14] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," in Proceedings of ACM SIGCOMM, 1998.
[15] V. Paxson, "End-to-end Internet Packet Dynamics," in Proc. Of ACM SIGCOMM, September 1997.
[16] K. Ramakrishnan, S. Floyd, "A Proposal to Add Explicit Congestion Notification (ECN) to IP," IETF RFC 2481, January 1999.
[17] "Network Simulator - ns version 2," http://www-mash.cs.berkeley.edu/ns/, 1999.