Minimizing Queue Variance Using Randomized Deterministic Marking

Na Li
Ashley Laurent Inc.
Gustavo de Veciana, Sangkyu Park, Marissa Borrego, San-qi Li
Department of Electrical and Computer Engineering
University of Texas at Austin

Abstract— Recent work on congestion control in TCP/IP networks combines improved end-user transmission mechanisms with active queue management schemes at network routers. An active queue management scheme consists of two stages. In order to stabilize the queues at a router, one must first determine an appropriate packet marking probability given the current degree of congestion. Second, in order to realize the desired marking probability, an effective packet marking algorithm needs to be implemented to decide which packets should be marked. Recently, researchers have increasingly focused on the first stage, namely determining the fraction of packets to mark, overlooking the fact that for a given marking probability, various possible marking algorithms result in different queue variance, and thus loss, delay, and jitter. In this paper we propose a marking algorithm DREAM. DREAM decouples the functions of reducing queue variance and randomizing the phases of flows. Compared to existing schemes, it significantly reduces queue variance while avoiding flow synchronization. Based on a simple Markov chain model we explain why our scheme is superior. Our simulation results confirm its effectiveness. Furthermore, DREAM is simple to implement and has much lower overhead as compared with existing mechanisms.

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

I. Introduction

TCP plays a critical role in reducing packet losses and avoiding congestion collapse [7] in the Internet. However, there are still some performance issues [12]. For example, flow synchronization can occur when congested routers drop all incoming packets with no more buffer avaiable. By synchronization, we mean that a large number of flows concurrently increase and decrease their congestion windows, and thus their sending rates. This results in periods of queue overflow followed by periods of queue underflow. When such oscillation persists, it may reduce network utilization. In other cases it simply adds to the queue variance and thus jitter experienced by packets in the network.

It is recognized that more proactive steps are needed to not only recover from periods of congestion, but also avoid congestion and achieve high network resource utilization [9]. This has prompted the IETF to recommend the use of active queue management [2] in conjunction with Explicit Congestion Notification (ECN) [6].

Active queue management at network routers attempts to improve performance by monitoring the queue size. When the queue is building up, it marks or drops some incoming packets to indicate the onset of congestion. Our goal is to spread out congestion notifications over time, rather than sending them out in a burst as dropping packets from the tail of a queue. In this way, TCP flows can gradually reduce their total sending rates before severe congestion occurs. The frequency and duration of congestion are reduced. This may reduce packet loss, improving utilization and reduce queuing delays and jitter. Randomness is added in this mechanism to eliminate biases against some flows, which could occur in a deterministic system [4]. Furthermore, randomness is highly desirable to avoid flow synchronization, as is discussed in the next section.

An active queue management scheme consists of two stages. To stabilize the queue size, one determines an appropriate marking probability given the current congestion state. Second, one realizes this target marking probability via a marking algorithm. Recent research work has increasingly focused on the first stage, namely finding an appropriate marking probability. Previous proposals include Random Early Detection (RED) [5], and Stabilized-RED (SRED) [11]. As a new observation, we have found that for a given marking probability different marking schemes may result in very different marking intervals (the number of packets between adjacent marks) of individual flows, as well as queue variance at routers. The larger the queue variance, the more buffer is required to support TCP traffic. For a given queue size, a queue with larger queue variance is likely to see a higher packet loss ratio, and results in lower bandwidth utilization and higher delay jitter. Therefore, we believe it is also critical to minimize the queue variance via an appropriate marking scheme.

Packet marking at routers can be applied to either aggregate traffic or individual flows. Within a Differentiated Service (Diffserv) network [3], aggregated marking is most likely used at the core routers [2], and per-flow marking is appropriate for the traffic conditioner at the edge routers [1]. Aggregated marking has low overhead since packets of different flows are stored in the same queue. Roughly speaking, more evenly distributed marks within an aggregate flow leads to more even marks within individual flows. However, it is difficult to enforce that marks be uniformly distributed among and within flows. Per-flow marking requires per-flow queuing, but enables control over the marking intervals of individual flows. The following marking schemes can be used for both cases.

The rest of the paper is organized as follows: Section II discusses the origins of queue variance to get an idea on how to tackle this problem. Related work is presented and some comments are given in Section III. The DREAM algorithm is detailed in Section IV. The analysis is omitted in this paper, but available at [10]. The simulation results and comparisons to existing algorithms are found in Section V. Finally, Section VI concludes the paper.

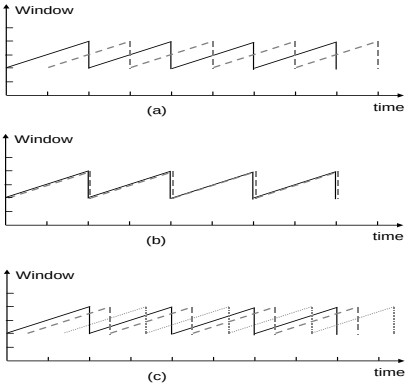

Fig. 1. The relative phases of TCP flows.

II. Queue Variance Analysis

The queue variance at a network router is determined by the variability of individual TCP congestion window processes and the relative phases of the flows passing through the router. Intuitively, the smoother the individual windows are, the smoother the total arriving rate at a network router will be, as well as the queue variance. As to the relative phases of flows, there are three major scenarios: out of phase, in phase, and random phase. The most favorable case is that where all flows are coordinated and are uniformly "out of phase". In the case of two flows, the peaks of their congestion windows are perfectly interleaved, as shown in Fig. 1(a). It is possible for this situation to occur naturally, but difficult to guarantee it will occur consistently.

The least favorable case is that where all or most of the flows are in phase. They increase and decrease their windows almost at the same time. This scenario is illustrated in Fig. 1(b). This situation could also happen naturally, and could persist in a deterministic system. The system must avoid being locked in this state since the aggregate variance is large. A more desirable scenario is one where flows have roughly random phases (Fig. 1(c)).

We assume the ideal saw-like congestion control window pattern of TCP Reno [14] during congestion avoidance, and want to justify the performance differences among the out of phase, in phase, and random phase scenarios. Consider M flows with the same Round Trip Times (RTTs) and passing through a single bottleneck. The window size of an individual TCP flow is a random process over time. We denote it as W, and its average as $\overline{W}$. The throughput of a flow T is also a random process, and its average is $\overline{T}$. The capacity of the bottleneck router is constant and denoted by C. To simplify the derivation, we assume that the buffer size of the router is infinite and never empty. The relations between the W, T, $\overline{W}$, $\overline{T}$, and C are approximately

$$\begin{aligned} T &= \frac{W}{RTT}, \\ \overline{T} &= \frac{C}{M}, \\ \overline{W} &= \frac{C \times RTT}{M}. \end{aligned} \tag{1}$$

The variance of the window size W can be easily derived as W varies uniformly distributed on the interval $[\frac{2}{3}\overline{W}, \frac{4}{3}\overline{W}]$, giving

$$\text{VAR}(W) = \frac{1}{27}\overline{W}^2. \tag{2}$$

From the above equations, the variance of T is derived as

$$\begin{aligned} \text{VAR}(T) &= \frac{1}{RTT^2}\text{VAR}(W) \\ &= \frac{1}{27RTT^2}\overline{W}^2 \\ &= \frac{C^2}{27M^2}. \end{aligned} \tag{3}$$

When all flows are perfectly out of phase, the aggregated window size of all flows is uniformly distributed in $[M \times \overline{W} - \frac{1}{3}\overline{W}, M \times \overline{W} + \frac{1}{3}\overline{W}]$. Therefore, the variance of the aggregated window size is $\frac{1}{27}\overline{W}^2$, and the variance of the total arrival rate is still $\frac{C^2}{27M^2}$. When all flows are in phase, the aggregated window size of all flows is uniformly distributed in $[\frac{2}{3}M \times \overline{W}, \frac{4}{3}M \times \overline{W}]$. Therefore, the variance of the aggregated window size is $\frac{1}{27}(M \times \overline{W})^2$, and the variance of the total arrival rate is $\frac{C^2}{27}$. If the phases of all flows are identically but independently distributed variables, the variance of the total arrival rate is the summation of individual variances, and is equal to $\frac{C^2}{27M}$. The results are summarized in Table I.

We now consider how these scenarios scale. Generally, the required buffer at a router is proportional to its queue standard deviation given a packet loss ratio. Suppose the link bandwidth increases linearly with the number of flows M supported by the system ($C \propto M$), i.e., the bandwidth of each flow is a constant. When all flows are out of phase, the variance of the arrival rate at the router is independent on the flow number. Theoretically, the same buffer size is enough to support an increasing number of flows with the same performance. The system is highly scalable. When all flows have totally random phases, the variance increases linearly with the flow number, i.e., the buffer

TABLE I
Variance of total arrival rate vs. flow phase characteristics

In phase	Random phase	Out phase
$\frac{C^2}{27}$	$\frac{C^2}{27M}$	$\frac{C^2}{27M^2}$

allocated to each flow decreases as the square root of the flow number M. Therefore, it is beneficial to aggregate more flows through a larger pipe to benefit from the statistical multiplexing gains. When all flows are in phase, the buffer per flow is the same when the capacity scales up, i.e., there is no statistical multiplexing gain and the system is not scalable. As a compromise among these scenarios we prefer to ensure, the random phase scenario is in force. Therefore, our control goal is to reduce the individual window variance, and randomize the flow phases.

Two major factors contribute to the individual window variance. The first one is the linear increase and exponential decrease of the TCP congestion window. Unless the TCP congestion control mechanism is modified, such variance cannot be reduced. TCP congestion control algorithm sets the minimum possible range on which its congestion window varies. Given an average window size $\overline{W}$, the smallest range on which the window size W varies is $[\frac{2\overline{W}}{3}, \frac{4\overline{W}}{3}]$. Secondly, the packet number between two marks of a flow also affects the individual window variance. If consecutively marked packets of a flow are too close, TCP congestion window can drop below $\frac{2\overline{W}}{3}$. On the other hand, if the intervals between marked packets are too large, the window can increase beyond $\frac{4\overline{W}}{3}$. Both cases result in larger window variance, consequently, larger queue variance. According to our simulation results, the more variation in the number of packets between two marks for a given flow, the larger the queue variance is. Therefore, a good marking scheme should distribute marks as uniformly as possible among and within flows, resulting in a reduction of the queue variance.

III. Related Work

In this section, we describe current marking algorithms given a desired marking probability p. Floyd originally discussed the question of how to mark packets given a marking probability [5]. The goal was to randomly mark packets but at fairly regular intervals.

A natural solution is to mark incoming packets with probability p, which we call Random Marking (RM) in this paper. However, the marking interval, i.e., number of unmarked packets between two that are marked, is geometrically distributed. The memoryless nature of this approach cannot guarantee a regular distribution of marks.

Floyd then proposed to introduce correlation in packet marking [5], and this approach is referred as Uniform Random Marking (URM). A counter n is used to keep the number of unmarked packets that have arrived since the last marked packet. After each marking, the counter n is reset to be zero. Given the constant marking parameter p, the actual packet-marking probability $p_a(n)$ increases quickly with the counter increment:

$$p_a(n) = \frac{p}{1 - n \times p}. \tag{4}$$

It is proved that the number of arriving packets between marked packets is a uniform random variable in $[1, \frac{1}{p}]$. Simulation results also show that URM distributes marks more evenly than RM does. Note that RM can be expressed as

$$p_a(n) = p. \tag{5}$$

URM attempts to reduce the interval variation while keeping the randomness of marking. However, our simulations show that in some scenarios, the queue variance of URM is even higher than RM since URM increases $p_a(n)$ aggressively, particularly when p is large.

A more mild approach, referred as Wait Uniform Random Marking (WURM) is implemented in the NS-2 simulator. It also introduces correlation in packet marking [15]. In contrast to URM, WURM waits for a while after a marked packet, and increases $p_a(n)$ more slowly than URM does. The actual packet-marking probability $p_a(n)$ is given by

$$p_a(n) = \begin{cases} 0 & \text{if } n \times p < 1, \\ \frac{p}{2 - n \times p} & \text{if } 1 \le n \times p < 2, \\ 1 & \text{otherwise.} \end{cases} \tag{6}$$

The problem is that when p takes certain values, $p_a(n)$ may see a big jump as a function of n, i.e., is not very "smooth". Table II shows the numerical values of $p_a(n)$ versus n when p is below 0.5 and above 0.5. The average final marking probability $\overline{p}_a$ is $\frac{1}{3}$ for $p = 0.5^-$, and $\frac{5}{12}$ for $p = 0.5$. As a result, under some scenarios, we cannot achieve the desired marking probability. A lower actual marking probability $\overline{p}_a$ has drop-tail like behavior, and larger value results in persistent queue underflow.

A further modification to WURM is

$$p_a(n) = \begin{cases} p & \text{if } n \times p < 1, \\ \frac{p}{2 - n \times p} & \text{if } 1 \le n \times p < 2, \\ 1 & \text{otherwise.} \end{cases} \tag{7}$$

We call this variant as Slow Random Marking (SRM). It is less aggressive than URM and avoids the discontinuity problem of WURM.

IV. DREAM: Deterministic Marking with Random Alignment

From the above discussion of related work, we observe that introducing correlation makes the marking more predictable and smoothes out the marking interval. It helps to reduce queue

TABLE II
Discontinuity of WURM around $p = 0.5$

$p_a(n)$	p	
n	0.5^-	0.5
1	0	0
2	0	0.5
3	1^-	1

p : marking probability
U : uniform random variable in [0,1)
n : number of unmarked packets since the last mark

On receiving an incoming packet
if($n < \frac{1}{p}$)
  $n++$
else
  mark the packet
  generate a value of U
  if($U < 0.5$)
    $n = n - \frac{1}{p} - 1$
  else
    $n = n - \frac{1}{p} + 1$

Fig. 2. DREAM Algorithm

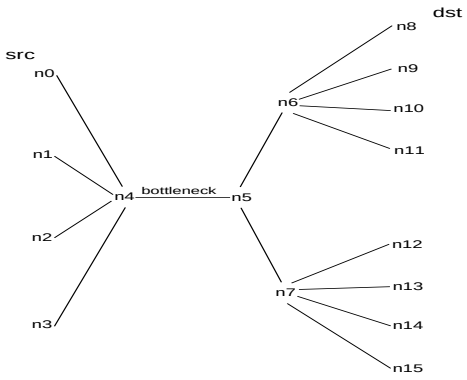

Fig. 3. Simulation topology.

variance by reducing the individual flow variance. In fact, Deterministic Marking (DM), i.e., marking a packet every $\frac{1}{p}$ packets, results in constant marking intervals, and therefore should have the smallest queue variance. Unfortunately, synchronization among flows is a concern with deterministic marking. The previous approaches all add randomness by randomly marking packets, based on a state-dependent probability. As more and more correlation among packets is introduced, less and less randomness is left. Reducing per-flow variance and randomizing the flow phases appear to be two conflicting goals.

Instead of continuing in the above direction and introducing more delicate mechanism, we take a different path towards solving this problem and propose DREAM, a Deterministic with Random Ergodic Alignment Marking scheme. DREAM decouples the functions of reducing the making interval variance and randomizing the phase of flows. Deterministic marking can guarantee that marks are evenly spaced, therefore keeps the queue variance small. On the other hand, adding a small random shift to the marking interval randomizes the phases of flows, and therefore reduces the likelihood of synchronization.

A. Algorithm

The actual packet-marking probability $p_a(n)$ of DREAM is given by

$$p_a(n) = \begin{cases} 0 & \text{if } n \times p < 1, \\ 1 & \text{otherwise.} \end{cases} \quad (8)$$

The pseudo code for the algorithm is exhibited in Fig. 2. When the number of unmarked packet n is below $\frac{1}{p}$, the counter n is increased by one. Otherwise, the incoming packet is marked. Such counting and marking form a deterministic process. At the same time, the phase of the flow, i.e., the state of the embedded random walk process [8], is updated by adding an independent random variable with equal probability of being one or minus one at the start of each interval. The state space of the random walk is $[0, \frac{1}{p})$. The state wraps around back to zero when it exceeds $\frac{1}{p}$. The formal definition of a flow phase and theoretical analysis are given in [10].

Note that $\frac{1}{p}$ may be a parameter with double precision. The counter n can be implemented as an integer variable, or a variable with double precision. Making n a double variable can keep the residual amount of the last marking and realizes the marking probability precisely.

Besides its good performance, DREAM has very low implementation overhead. For DREAM, one counter update is performed at each packet arrival. A random number is generated only when $n \geq \frac{1}{p}$. Also note that DREAM does not require using floating point calculations. On the other hand, all the other approaches (URM, WURM, and SRM), have to generate a random number and perform a floating point calculation of $p_a(n)$ at every packet arrival, as well as performing counter updates. Therefore, they involve much higher overhead than DREAM does.

V. Simulation Results

The simulator NS-2 [15] was used to asses the proposed scheme. ECN was enabled. We shall present results for various scenarios with different numbers of flows ranging from 40 to 2000 flows. All flows have the same RTTs, go through the same bottleneck link, and are synchronized at the beginning. All simulations last for 1000 seconds.

The simulation topology is shown in Fig. 3. All links have bandwidth of 600Mb/s except the bottleneck link [n4-n5]. To investigate the mechanism's scalability, the bandwidth at the bottleneck increases linearly with the number of flows, adding 0.25 Mb/s per flow. The maximum queue size of each link is 1800 packets. The marking probability is about 0.07, and changes slightly with different marking algorithms to bring the queue size within [0, 1800]. In another word, the utilization of the bottleneck link is always 100%, and there is not packet loss.

The queue standard deviation versus flow number with per-flow marking at the bottleneck link is shown in Fig. 4. DM results in a very large queue variance. The reason is that most flows are synchronized initially given the topology setup and DM has no way to reduce the synchronization. RM re-

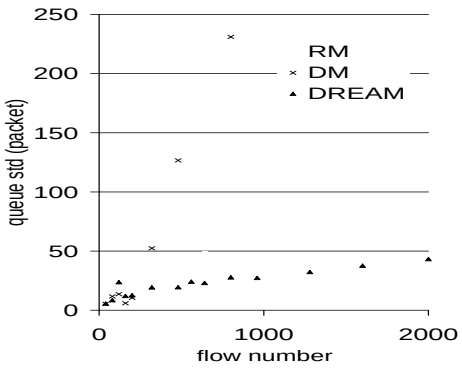

Fig. 4. Queue standard deviation with per-flow marking.

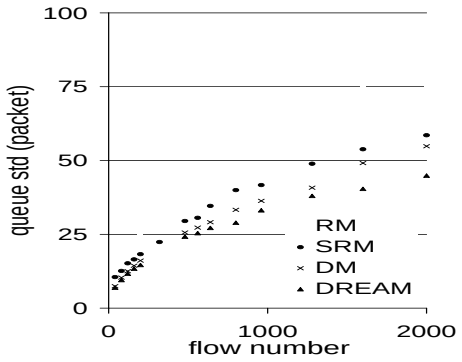

Fig. 5. Queue standard deviation with aggregated marking.

duces flow synchronization, and thus achieves substantially less queue variance compared with DM. By combining deterministic marking with phase randomization, DREAM achieves the lowest queue variance. The queue standard deviation of DREAM is almost half of the value of RM for all tested scenarios. When SRM is used for per-flow marking, many TCP flows stop sending packets, therefore, its result is not reliable and not listed here.

Fig. 5 shows the queue performance with aggregated marking. RM has the largest queue variance, and DREAM has the lowest queue variance. RM, SRM, and DREAM are all able to randomize the flow phases. By reducing the variance of marking intervals within the aggregated flow to the minimum, DREAM manages to get the smallest queue variance. It is important to notice that the queue variance of DM is consistently larger than that of DREAM for all tested scenarios. A plausible explanation is that DM exhibits biases among flows by marking more packets of some flows while favoring some others. On the other hand, DREAM randomizes the chance a flow gets marked. Therefore, on average, all flows show similar behaviors. Further investigation is necessary to explain the result.

VI. Conclusion

In this paper, we propose a Deterministic Marking scheme with Random Ergodic Alignment (DREAM). DREAM reduces queue variance significantly compared with existing algorithms while avoiding flow synchronization. Unlike previous proposals, DREAM decouples the functions of reducing variance of marking intervals and randomizing the phases of flows. A simple analysis helps explain why our scheme is superior to other schemes and simulation results confirm the effectiveness of our scheme. Furthermore, DREAM is simple to implement and has much lower overhead than the others.

There are two observations regarding to this solution. Randomness is desirable in various distributed systems to reduce the global synchronization or avoid biases against certain parties. The manner in which randomization is introduced may affect the system performance significantly. Furthermore, it is usually easier to solve a problem by identifying its causes and decomposing the required functionalities into distinctive modules.

References

[1] Y. Bernet, S. Blake, D. Grossman, A. Smith, “An Informal Management Model for Diffserv Routers,” IETF draft draft-ietf-diffserv-model-05.txt, November 2000.

[2] B. Braden, et al, “Recommendations on Queue Management and Congestion Avoidance in the Internet,” IETF RFC 2309, April 1998.

[3] M. Carlson, W. Weiss, S. Blake, Z. Wang, D. Black, and E. Davies, “An Architecture for Differentiated Services,” IETF RFC 2475, December 1998.

[4] S. Floyd, V. Jacobson, “On Traffic Phase Effects in Packet-Switched Gateways,” Internetworking: Research and Experience, V.3 N.3, September 1992, p.115-156.

[5] S. Floyd, V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance,” IEEE/ACM Transactions on Networking, August 1993.

[6] S. Floyd, “TCP and Explicit Congestion Notification,” Computer Communication Review, vol. 24, no. 5, October 1994.

[7] V. Jacobson, “Congestion Avoidance and Control,” In Proceedings of ACM SIGCOMM, August 1988.

[8] L. Kleinrock, “Queueing Systems volume 1: Theory,” Wiley-Interscience Publication, New York, 1975, page 23.

[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] N. Li, G. de Veciana, S. Park, M. Berrego, S. Li, “Minimizing Queue Variance Using Randomized Deterministic Marking,” full version at http://64.245.57.34/papers.htm.

[11] T. Ott, T. Lakshman, L. Wong, “SRED: Stabilized RED,” IEEE INFOCOM’99.

[12] V. Paxson, “End-to-end Internet Packet Dynamics,” in Proc. Of ACM SIGCOMM, September 1997.

[13] K. Ramakrishnan, S. Floyd, “A Proposal to Add Explicit Congestion Notification (ECN) to IP,” IETF RFC 2481, January 1999.

[14] W. Stevens, “TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms,” IETF RFC2001.

[15] “Network Simulator - ns version 2,” http://www-mash.cs.berkeley.edu/ns/, 1999.