Technical Research Report

A Parallel Virtual Queue Structure for Active Queue Management

by Jia-Shiang Jou, John. S. Baras

CSHCN TR 2003-4
(ISR TR 2003-9)

A Parallel Virtual Queue Structure for Active Queue Management

Jia-Shiang Jou and John S. Baras
Department of Electrical and Computer Engineering
and the Institute for Systems Research
University of Maryland, College Park, MD 20742 USA
{jsjou, baras}@isr.umd.edu

Abstract*— The Adaptive RED proposed by Feng $et\ al.$ [5] is shown to have small packet delay and queue length variation for long-life TCP traffic such as FTP connection with a large file size. However, a great portion of Internet traffic is short-life web and UDP traffic. Most web traffic has a small file size and its TCP session is mainly operated in the slow start phase with a small congestion window size. Since the file size is small, dropping short-life TCP (and UDP) packets is not very effective in alleviating congestion level at a bottleneck router. From the viewpoint of TCP, one or several packet losses in its slow start phase lead to extra delay for retransmission and even cause TCP timeout. This delay severely degrades the performance of delivering short messages such as web pages and web browsers experience a long waiting time even with a high speed network. We first show that the Adaptive RED is vulnerable to these short-life TCP traffic and propose a virtual parallel queue structure as a new active queue management scheme (AQM). The idea is to separate the long-life and short-life (including UDP) traffic into two different virtual queues. The first queue is to run the drop-tail policy and work for the short-life TCP and UDP packets. In order to have a small mean delay, the service rate of this drop-tail queue is dynamically determined by its virtual queue length. The remaining long-life traffic is directed to an Adaptive RED virtual queue. Even the available bandwidth is shared with the drop-tail queue, the simulation results show that the queue length variation of the RED queue is still located in a desired region. Note that both virtual queues share the same physical buffer memory. Those packets in the drop-tail queue will not be dropped unless the shared buffer is overflowed. This parallel virtual queue structure not only keeps the benefits of RED such as high utilization and small delay, but also greatly reduces the packet loss rate at the router.

***Index Terms*— Adaptive RED, AQM, Scheduling.**

I. Introduction

THE RED (Random Early Detection) scheme has been attracting a lot of attention for active queue management in the network router. In order to have a small queuing delay, it is desired to have a short mean queue length so that the total buffer size in a drop-tail queue can not be large. However, the Internet traffic is quite bursty so that small buffer routers encounter a high packet loss rate. Moreover, the TCP protocol dramatically reduces its flow rate in the congestion avoidance phase when packets are lost. After a buffer overflow event in a drop-tail queue, all connections sense packet loss and slow down the transfer rate simultaneously. In order to prevent this global synchronization phenomenon and low link utilization, many active queue management schemes such as RED are proposed. The basic idea of RED is randomly dropping packets to prevent buffer overflow. The dropping probability is a non-decreasing function of the queue length so that more packets will be dropped when the queue length is larger. The connection at a high flow rate has a higher chance to get more packets dropped and reduces its rate rapidly. This scheme controls the flow rate of TCP connections and keeps the queue length in a desired region. However, some simulation results [3] have demonstrated that the performance of RED is pretty sensitive to parameter settings. Based on the original idea of RED, there are some modifications such as Stabilized RED [8], Random Early Marking (REM) [1], Blue [4] and Adaptive RED [5] [6]. The Adaptive RED scheme dynamically updates the maximal dropping probability according to the exponentially weighted moving average (EWMA) of queue length, and makes itself more robust with respect to congestion level. Based on [5] [6], we are motivated to consider UDP and short-life TCP connections such as web traffic. Dropping these kinds of packets is not only useless for controlling the flow rate, but also wastes network resources and causes unnecessary delay and retransmissions. In addition, the performance of Adaptive RED is severely degraded by these short but bursty connections. We first demonstrate the vulnerability of Adaptive RED in this situation and propose a parallel virtual queue structure for active queue management.

Research partially supported by DARPA through SPAWAR, San Diego under contract No. N66001-00-C-8063.

II. Vulnerability to Web-mice

In this section, we describe a scenario containing short-life TCP (WEB), UDP (CBR) and long-life TCP (FTP) traffic. The purpose is to demonstrate that the performance of the Adaptive RED scheme is severely degraded by the short-life web traffic. The network in our experiment has a simple dumbbell topology with the bottleneck link bandwidth C=$3.0Mbps$. One side of the bottleneck consists of 800 web clients. Each client sends a web request and has an $Exponential$ distribution of think times with mean $50s$ after the end of each session. The other side contains 800 web servers, running HTTP 1.1 protocol and having a $Pareto$ file size distribution with parameters (K=$2.3Kbyte$, α=1.3) (mean $10Kbytes$). The round-trip propagation delay of HTTP connections is uniformly distributed between $(16,\ 240)ms$.

Note that the mean rate of the aggregate web traffic is around $1.2Mbps$. There is one CBR traffic source which periodically generates a $1Kbytes$ UDP packet every $50ms$. Besides these short web connections and UDP traffic, there are 10 persistent FTP connections sharing the bottleneck link with round-trip propagation delay of $64ms$. Figure. 1 shows that Adaptive RED works well with those FTP connections before the web traffic comes in. At time t=$100s$, the CBR source and web servers begin to share the bandwidth. Since the aggregate web traffic is very bursty, the queue length of Adaptive RED deviates dramatically from the desired region.

Since most web pages contain one or several very small files, those TCP connections are almost operated in their slow start phase during the session life. According to the TCP protocol, the congestion control window is just beginning to increase its size from the initial value and has a low transmission speed. Dropping packets in the slow start phase can not efficiently alleviate congestion level at the bottleneck router. In other words, any blind random dropping/marking policy such as RED is unable to effectively control the congestion level without considering short-life TCP (and UDP) traffic. Furthermore, losing one or two packets in the slow start phase not only causes a very low throughput and extra delay, but also leads to a high probability of connection timeout. We desire to avoid this phenomenon especially for those short and time-constrained messages. In addition, the Adaptive RED scheme relies on average queue length to determine the dropping probability and control the TCP flow rate. The extra queue length perturbation contributed by the bursty web traffic makes Adaptive RED increase/decrease its dropping probability rapidly. This over-reaction causes a great queue length variation and poor performance in packet delay and loss.

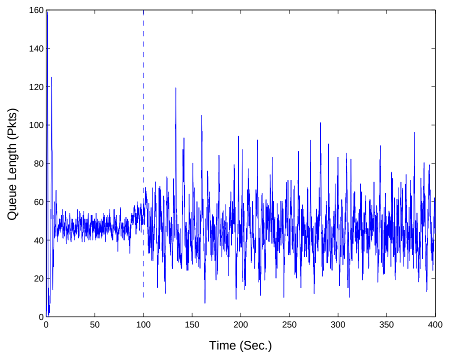

Fig. 1. Queue length of Adaptive RED: 10 FTP starting at t=0 and 800 Webs and 1 CBR starting at t=100s.

III. Two Virtual Queues Structure: RED and Tail Policy

A. Motivation

Dropping short-life TCP and UDP packets in a RED queue can not effectively alleviate the congestion, wastes network resources, and causes unnecessary retransmissions and delay. From the viewpoint of a web browser, a short-life TCP session may only need several round-trip times to finish the whole transmission. One packet loss in its slow start phase not only causes extra round-trip time to retransmit the dropped packet, but also immediately enforces TCP to leave its slow start phase with a very small slow start threshold ($ssthresh$). When the sender senses a packet loss, $ssthresh$ will be reduced to $min(cwnd, rcv\_window)/2$ and the new $cwnd$ is also decreased depending on different TCP versions. (For TCP Reno, the new window size $cwnd$ = $ssthresh$ and TCP enters the Fast Recovery phase. For TCP Tahoe, $cwnd$ = MSS (maximum segment size) and TCP begins a new slow start phase.) Since the original congestion window is just beginning to increase its size from MSS in its first slow start phase, one packet loss during the initial several round-trip times leads TCP to enter the congestion avoidance phase with a very small $ssthresh$ and $cwnd$. In the congestion avoidance phase, TCP slowly increases $cwnd$ (the increment is about one MSS per round-trip time) from the current $ssthresh$. Thus, losing one packet in the slow start phase causes TCP to take a long time to complete a short message. In addition, since the web traffic is assumed to be small but bursty, these web connections usually experience a higher packet loss rate (see the web packet loss rate of RED and drop-tail queue in Table III). Note that the default initial value of $ssthresh$ is $64KB$ and packet size is $1KB$ in this paper. At a high packet loss rate P_d=0.04 contributed by RED, the probability of losing one or more packets in the slow start phase is equal to $1-(1-P_d)^{64}$ = 0.9267 (assume packets are dropped independently). Therefore, most TCP connections have at least one packet loss in their first slow start phase. For example, assuming that the 15^{th} packet is dropped by RED, the $ssthresh$ decreases from $64KB$ to $4KB$ and the new congestion window $cwnd$ is decreased from $8KB$ to $1KB$ (Tahoe). The situation is getting worse if one packet is dropped earlier (in the first 3 round-trip times). The congestion window at this moment is so small that the sender may not have enough data packets to trigger the receiver into generating three duplicate acknowledgements. If packets cannot be recovered by this fast recovery scheme, TCP has to depend on the protocol timer for error recovery. The default value of the protocol timer is usually large and the delivery delay could be increased dramatically by timeout events. Moreover, the probability of losing two or more packets of the same congestion window in the slow start phase still can not be ignored. These events also lead to a high probability of TCP timeout and connection reset. To demonstrate this phenomenon, we consider a small web page which has 90 packets to be transferred in a stand alone environment. There is no other traffic sharing the bandwidth and packets are dropped artificially. The simulation results in Figure 2 show that TCP takes about $4.81s$ to complete the whole session if P_b =0.04. In comparison, in the loss free case, this $90Kbytes$ file can be delivered within $1.18s$. Table I lists the mean and standard deviation of this delivery delay (in $sec.$) with file size from $30K$ to $210K$ bytes. Since most web pages have their size located in this range, the web browser will experience a long response time when the packet loss rate is large. As mentioned previously, the performance of the Adaptive RED scheme is also severely degraded by this short but bursty traffic. It is natural to have a separate queue

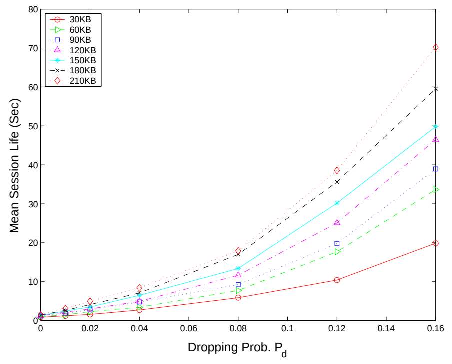

Fig. 2. Mean delivery delay of small file $v.s.$ dropping probability P_d with file size $30, 60, ..., 210Kbytes$, bandwidth $3Mbps$ and round-trip time $128ms$.

TABLE I

DELIVERY DELAY OF SMALL FILE: MEAN AND STANDARD DEVIATION

P_d	0.00	0.02	0.04	0.08
$30KB$	0.88(.0006)	1.60(1.88)	2.74(4.27)	5.88(7.79)
$90KB$	1.18(.0008)	2.79(2.39)	4.81(3.91)	9.24(6.30)
$150KB$	1.34 (0.0008)	3.51(1.90)	6.51(4.60)	13.38(8.87)

deal with the short-life TCP and UDP traffic. Since dropping these packets can not effectively alleviate congestion level, but severely increases delivery delay, it would be better to keep them in the queue unless the total buffer (shared with the other queue) is overflown. Hence, the queuing policy of the first virtual queue is chosen to be drop-tail to minimize the packet loss rate. In order to have a short delivery delay for web browsers and UDP connections, the service rate $C_1(t)$ is changed dynamically according to its virtual queue length $q_1(t)$.

The second virtual queue is running the Adaptive RED policy with the long-life TCP connections such as FTP with a large file. Even if the available bandwidth of the second queue is determined by $C_2(t)$=C-$C_1(t)$, the Adaptive RED scheme still works well in keeping its virtual queue length $q_2(t)$ in a desired region. Note that the available bandwidth at the RED queue is suppressed and controlled by the drop-tail queue. When there is a heavy workload at the drop-tail queue, $C_2(t)$ decreases quickly. FTP receivers experience a slower packet arrival rate and send acknowledgement packets (ACK) back slowly. Without increasing the dropping probability at the RED queue, the slower ACK arrival rate from the receivers causes FTP senders to reduce their flow rate automatically without shrinking their control window size. On the other hand, when the congestion level is alleviated, the RED queue receives more bandwidth. Since the congestion window size is still large in the FTP server, the throughput of FTP is quickly recovered by a faster arrival rate of ACK packets from the receiver. Since the TCP average throughput [7] [2] is proportional to $\frac{1}{RTT\sqrt{P_d}}$, and the RTT is equal to the sum of the round-trip propagation delay and the queuing delay. Given the available bandwidth of $C_2(t)$, there is a trade off between dropping probability and queuing delay. We are able to reduce the dropping probability and increase queuing delay by dynamically adjusting the thresholds of Adaptive RED.

With this approach, we can keep the benefits of Adaptive RED such as high (100%) link utilization. Furthermore, the packet loss rate of the short-life TCP and UDP connections is greatly eliminated by the drop-tail policy and a shared buffer. The packet loss rate of long-life TCP traffic is also reduced by the suppressed bandwidth, larger thresholds (longer RTT) and a more stable average virtual queue length in the RED queue.

B. Description

By following the basic idea of this RED+Tail policy, the first problem is how to split the long-life traffic from other short-life web traffic at the routers. To this end, the router has to know the age or elapsed time of each TCP connection. Unfortunately, this information is hidden in the TCP header which is not available to the IP router. However, one may roughly estimate the elapsed time by using the following approach:

- When a packet from a new source-destination pair, which is not seen by the router in the past T sec arrives, we treat it as a new TCP connection and identify this connection as a short-life connection.
- Send the new connection packets to the drop-tail queue.
- Set a counter for the number of packets of this connection.
- If the cumulative packets number is greater than a threshold N, then we believe that the file size is large enough and this TCP connection already has left its slow start phase. We redirect the subsequent packets of this connection to the Adaptive RED queue.
- Remove the counter if there is no packet arrival in the last T sec.

Preliminary simulation results show that this approach successfully prevents the small web traffic from entering the RED queue. The probability of false alarm is less than 0.02 in our scenario. Since the web traffic has a small file size and a short session time, there is no harm if the short-life connection packets were misdirected to the RED queue after time T.

Figure 3 shows the RED+Tail parallel queue structure in the router. Let $C_1(t)$ and $C_2(t)$ denote the service rates of the drop-tail queue and the Adaptive RED queue respectively. In order to allocate bandwidth dynamically to both queues and assign a desired region of queue length in the Adaptive RED queue, we define the maximal threshold $maxth_i$ and minimal threshold $minth_i$ for $i = 1, 2$. The service rates $C_1(t)$ and $C_2(t)$ are given by the following algorithm:

- if $q_1 = 0$, then $C_1(t) = 0$.
- if $0 \leq q_1 < minth_1$, then $C_1(t)$:=C_{1min}.
- if $minth_1 \leq q_1$, then $C_1(t)$:=$min(C\frac{q_1}{maxth_1}, C_{1max})$.
- $C_2(t) := C - C_1(t)$,

where C is the link bandwidth. The parameter q_1 denotes the queue length of the drop-tail queue. The constant C_{1max} preserves the minimum available bandwidth $C - C_{1max}$ for the RED queue to prevent FTP connections from timeout.

IV. SIMULATION AND COMPARISON

In this section, we compare the Adaptive RED and RED+Tail schemes with typical TCP performance metrics. For

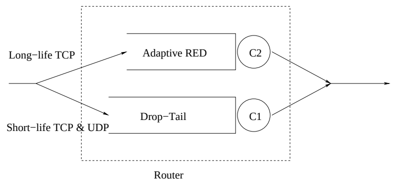

Fig. 3. Parallel Virtual Queue Structure for Active Queue Management.

the parameters of Adaptive RED, we employ the parameter settings suggested by Floyd [6] (α and β of the AIMD algorithm). We also implemented the RED+Tail scheme in the ns2 simulator. The network topology and scenario were already described in Section II. Table II lists the parameter settings of the RED+Tail parallel queues and Adaptive RED queue. Note that the virtual queues of the RED+Tail scheme share the total physical buffer size, $i.e.$ the packets in the drop-tail virtual queue will not be dropped unless the physical memory is full. The Adaptive RED is set in a "$gentle$" mode indicating that the dropping probability between $(maxth_2, 2maxth_2)$ is linear in $(P_{max}, 1)$.

TABLE II
EXPERIMENT SETTINGS

Queue i	Buffer Size	$minth_i$	$maxth_i$	α	β
$i = 1$	160KB	2KB	30KB	-	-
$i = 2$	shared	20KB	80KB	0.01	0.9
Adapt. RED	160KB	20KB	80KB	0.01	0.9

The performance for a connection is evaluated by the packet loss rate, delay and throughput. However, we focus on packet loss rate and delay for web (short-TCP) and CBR (UDP) connections, and place more concern on throughput for FTP (long-TCP). We replace the Adaptive RED with RED+Tail scheme at the router and redo the experiment of Section II. The random seed of the simulator is fixed so that the processes of web request and file size have the same sample path in both experiments. Figure 4 shows the queue length of the RED+Tail scheme, which demonstrates that the virtual queue length q_2 is quite stable and located in the desired region even after the web and CBR traffic begin to share the bandwidth at time t=$100s$. When a web burst arrives at the drop-tail queue, the queue length q_1 increases rapidly and most bandwidth is allocated to serve the drop-tail queue for a short time. Meanwhile, the available bandwidth is reduced at the RED queue so that FTP clients experience a long packet interarrival time. As a result, FTP servers receive acknowledgement packets slowly and the data rate is automatically reduced without increasing the dropping probability in the RED queue. Since the TCP average throughput is proportional to $\frac{1}{RTT\sqrt{P_d}}$, the actual dropping probability at the RED queue is also reduced from 4.15% to 2.75% by a longer queuing delay ($0.184ms$). This scheme prevents the over-reaction behavior of RED in the original Adaptive RED case and keeps the mean queue length q_2 still in a desired region. Figure 5 shows the packet loss rates of FTP, web and CBR connection with Adaptive RED and RED+Tail schemes. It is obvious that RED+Tail provides great improvement in packet loss for web and CBR connections. The

TABLE III
PERFORMANCE METRICS

Policy	Loss %	Delay Sec.	Rate KB/s
RED+Tail:FTP	2.747	0.184	209.465
RED+Tail:WEB	1.278	0.114	144.455
RED+Tail:CBR	0.300	0.109	19.867
AdaptRED:FTP	4.149	0.143	217.531
AdaptRED:WEB	4.514	0.143	137.124
AdaptRED:CBR	3.950	0.141	19.140
DropTail:FTP	1.916	0.349	215.243
DropTail:WEB	4.234	0.340	138.983
DropTail:CBR	1.550	0.342	19.601

TABLE IV
SMALL FILE DELIVERY TIME: MEAN AND STANDARD DEVIATION

File Size	30KB	90KB	150KB
RED+Tail:WEB	0.88(0.13)	1.15(0.24)	1.64(0.85)
AdaptRED:WEB	2.42(1.21)	3.87(1.41)	7.86(4.44)
DropTail:WEB	4.00(1.69)	10.91(4.01)	17.15(4.15)

web loss rate is reduced from 4.51% to 1.28% and CBR loss rate is reduced from 3.95% to 0.30%.

Figure 6 compares the packet delay. The mean queuing delay of web and CBR packets in the RED+Tail scheme is shortened by increasing the FTP packet delay. The web and CBR packet delay depends on how much bandwidth is allocated to the drop-tail queue. We are able to satisfy a mean delay requirement for the web and CBR connections by properly adjusting the parameter $maxth_1$. For example, the $maxth_1$ of the RED+Tail scheme is set to be $30Kbytes$ so that the expectation of mean delay at the drop-tail queue is about $80ms$. However, the service rate C_1 reaches its maximum C_{1max} when $q_1 > minth_1$. The actual mean delay should be larger than expected. Our simulation results show that the mean delay of web and CBR traffic is around $110ms$.

Figures 7 and 8 show the mean receive rate and throughput of FTP, web and CBR traffic. Both schemes achieve 100% utilization of the link bandwidth. Due to the unfair bandwidth allocation in the RED+Tail scheme, FTP has a slightly smaller throughput. However, the saved bandwidth allows web burst to pass through the bottleneck link faster. Table III lists the performance metrics of RED+Tail, Adaptive RED and the traditional drop-tail scheme respectively.

Table IV compares the small web file delivery time under different schemes. Since the RED+Tail policy has a small packet loss rate, its delivery time is almost equal to the loss free case in Table I. On the other hand, the Adaptive RED has a loss rate 4.5%, its delivery time is three times longer. Note that the drop-tail queue has a similar loss rate (4.2%) with Adaptive RED for web packets. However, the file delivery time of the drop-tail scheme is about 2.5 times longer than Adaptive RED's. This is mainly due to the long queuing delay ($0.34sec$) of the drop-tail policy.

V. DYNAMIC THRESHOLDS FOR ADAPTIVE RED

The Adaptive RED relies on the dropping probability to control the flow rates of TCP connections and keep the average queue length in a desired region. However, for those applications which have a large file to transfer, the goodput

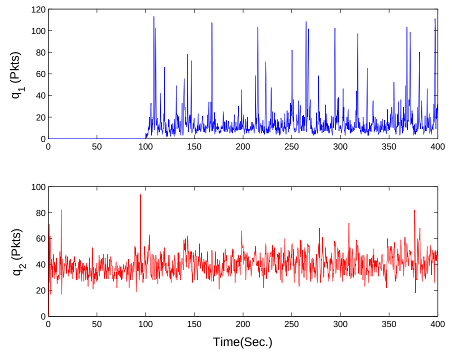

Fig. 4. Queue length of RED+Tail virtual queues: 10 FTP starting at t=0 go to virtual queue 2, and 800 Webs + 1 CBR starting at t=100 go to virtual queue 1.

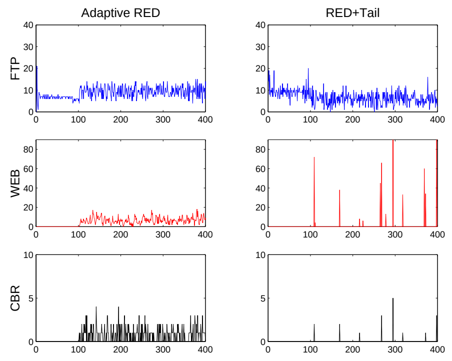

Fig. 5. Packet loss rate of Adaptive RED and RED+Tail: 10 FTP starting at t=0, and 800 Webs + 1 CBR starting at t=100s.

is more important than the packet delay. The packet loss rate is a key factor of connection goodput. Since the minimal and maximal thresholds of the Adaptive RED scheme are fixed, the dropping probability of RED must reach a high value to restrict the flow rates. This high dropping probability causes a great portion of retransmission and low goodput.

In order to control the flow rate and to keep a low packet loss rate, we propose a dynamic threshold algorithm for Adaptive RED scheme to guarantee the average packet loss rate in the RED queue.

- $D := maxth_2 - minth_2$ and $0 < \gamma < 1$.
- if $\bar{P}_d > P_U$, then $minth_2 := minth_2(1+\gamma), maxth_2 := minth_2 + D$.
- if $\bar{P}_d < P_L$, then $minth_2 := minth_2(1-\gamma), maxth_2 := minth_2 + D$,

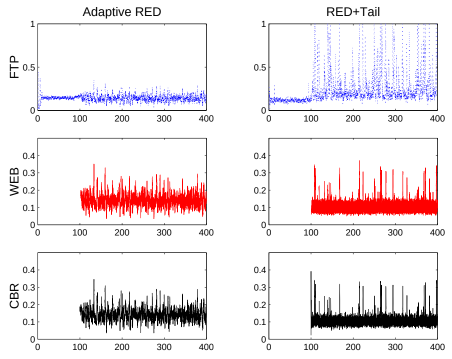

Fig. 6. Packet delay of Adaptive RED and RED+Tail: 10 FTP starting at t=0, and 800 Webs + 1 CBR starting at t=100s.

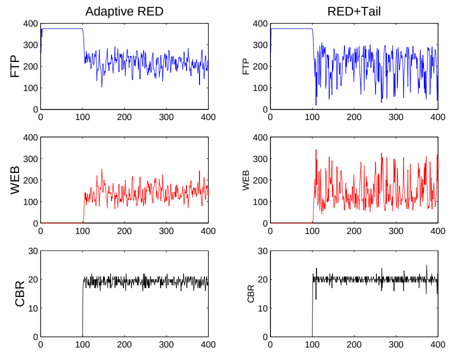

Fig. 7. Mean receive rate of Adaptive RED and RED+Tail: 10 FTP starting at t=0, and 800 Webs + 1 CBR starting at t=100s.

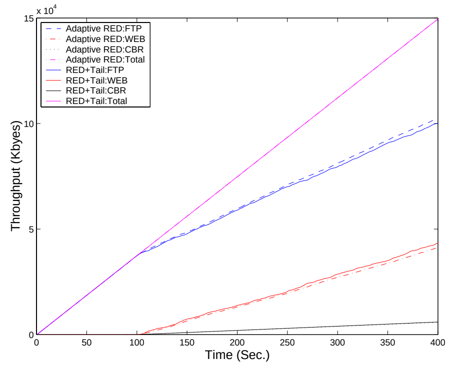

Fig. 8. Throughput of Adaptive RED and RED+Tail: 10 FTP starting at t=0, and 800 Webs + 1 CBR starting at t=100s.

where $\bar{P}_d$ is the average dropping probability obtained by EWMA algorithm and (P_U, P_L) are the desired region of dropping probability. Note that the distance $D := maxth_2 - minth_2$ is a constant and these floating thresholds do not change the current slope of dropping probability function.

Since the average TCP throughput is proportional to $\frac{1}{RTT\sqrt{P_d}}$, increasing the thresholds also increases the queuing delay and causes TCP to reduce its flow rate without a high packet loss rate. Figure 9 is a comparison of the queue length behavior between fixed and dynamic threshold schemes. There are 20 persistent FTP servers sharing a $6Mbps$ bottleneck link. Another 20 FTP servers arrive at time $100s$ and leave at time $300s$. Figure 9 shows that the fixed threshold scheme has a small varying queue length and a large dropping probability (0.05). In comparison with the previous, the dynamic threshold scheme has a much lower average dropping probability (0.014 with P_U=0.02, P_L=0.01), but a higher packet delay. Note that both schemes achieve 100% link utilization so that each FTP connection has the same throughput. However, with a much lower packet loss rate, the dynamic threshold scheme achieves a higher goodput. This dynamic threshold scheme allows us to consider the trade off between packet loss and queuing delay in an Adaptive RED queue.

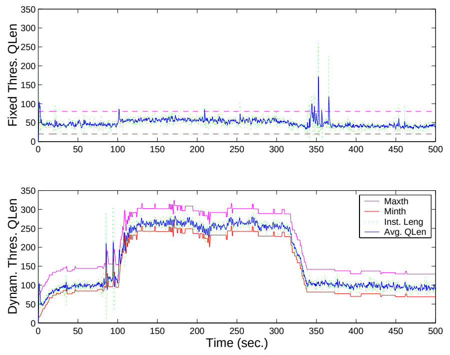

Fig. 9. Average queue length of fixed and dynamic thresholds: 20 FTP starting at t=0, and 20 starting at t=100s, C=6Mbps.

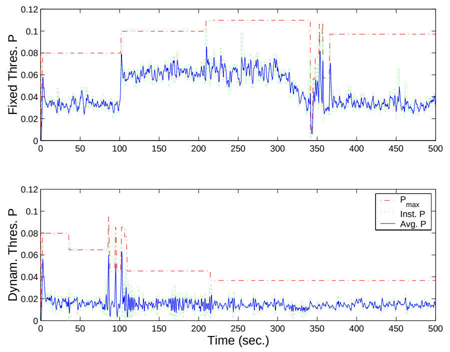

Fig. 10. Dropping probability of fixed and dynamic thresholds: 20 FTP starting at t=0, and 20 starting at t=100s, C=6Mbps.

VI. Conclusions

In this paper, we first demonstrated the vulnerability of Adaptive RED scheme to bursty traffic and then proposed a parallel virtual queue structure to eliminate unnecessary packet loss. A simple detection algorithm is employed to separate the short-life and long-life TCP connections into different virtual queues. The packet loss rate and mean delay can be greatly reduced by dynamic bandwidth allocation and active queue management with a parallel queue structure. This scheme combines the advantages of drop-tail and Adaptive RED policies. The simulation results in the study show that this scheme achieves a shorter mean delay for real time applications and keeps a high throughput for the best effort connections as well as greatly reduces the packet loss rate in both queues.

This parallel queue structure also provides more degree of freedom to control the router by considering different bandwidth allocation policies and dynamic thresholds for Adaptive RED. Here, the bandwidth allocation policy is a simple function of the current virtual queue length. However, it is well-known that web traffic is strongly correlated and has a long range dependency property. Based on observations of the ”recent past” traffic, the future bandwidth demand of the web traffic was shown to be predictable. In future work, we will consider the optimal bandwidth allocation policy based on the prediction of congestion level.

References

[1] S. Athuraliya, S. Low, V. Li, and Q. Yin. REM: Active queue management. *IEEE Network*, 15(3):48–53, 2001.

[2] T. Bu and D. Towsley. Fixed point approximations for TCP behavior in an AQM network. *ACM SIGMETRICS Performance Evaluation Review*, 29(1):216–225, 2001.

[3] M. Christiansen, E. Jaffay, D. Otu, and F.D. Smith. Tuning RED for web traffic. In *SIGCOMM*, pages 139–150, 2000.

[4] W. Feng, D. Kandlur, D. Saha, and Kang G. Shin. BLUE: A new class of active queue management algorithms. Technical Report U. Michigan EECS CSE-TR-387-99, 15, 1999.

[5] W. Feng, D.D. Kandlur, D. Saha, and K.G. Shin. In *Proceedings of INFOCOM 99*, volume 3, pages 1312–1328, 1399.

[6] S. Floyd, R. Gummadi, and S. Shenker. Adaptive RED: An algorithm for increasing the robustness of RED, 2001.

[7] T. Ott, J. Kemperman, and M. Mathis. The stationary behavior of ideal tcp congestion avoidance ftp://ftp.bellcore.com/pub/tjo/tcpwindow.ps.

[8] Teunis J. Ott, T. V. Lakshman, and Larry H. Wong. SRED: Stabilized RED. In *Proceedings of INFOCOM*, volume 3, pages 1346–1355, 1999.