TECHNICAL RESEARCH REPORT

A Parallel Virtual Queue Structure for Active Queue Management

by Jia-Shiang Jou, Xiaobo Tan, John S. Baras

CSHCN TR 2003-18
(ISR TR 2003-36)

The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site http://www.isr.umd.edu/CSHCN/

A Parallel Virtual Queue Structure for Active Queue Management

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

***Abstract*— The performance of the Adaptive RED scheme is susceptible to bursty web traffic. In this paper a parallel virtual queue structure is proposed for active queue management at the bottleneck router. Real time connections such as web and UDP, and non-real time connections such as FTP are served in two different virtual queues with drop-tail and Adaptive RED policies, respectively. Both queues share the same physical buffer memory. Simulation shows that file delivery for the web traffic is greatly improved due to shorter packet delay and lower packet loss rate, and that the queue length variation for the FTP traffic is small. To improve the goodput of the non-real time connections, a modified Adaptive RED scheme with dynamic queue length thresholds is also proposed. This scheme is able to keep the packet dropping probability within a desired small region. Stability of the queue length under this scheme is studied through analysis and numerical computation.**

***Index Terms*— Adaptive RED, Active Queue Management, Virtual Queue, Scheduling.**

I. Introduction

For small queuing delay, the buffer size in a router is in general not large. However, a router with small buffer size often has a high packet dropping rate since the Internet traffic is bursty. When packets are lost, the TCP protocol dramatically reduces the flow rate during the congestion avoidance phase [1]. Therefore, 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 increase link utilization, many active queue management schemes such as RED (Random Early Detection) [2] have been proposed and received increasing attention.

The basic idea of RED is to randomly drop packets to prevent buffer overflow and the global synchronization problem. The dropping probability is a non-decreasing function of the queue length. A TCP connection with a higher flow rate has a better chance to get packets dropped and reduce its rate more rapidly. By dropping packets actively, RED keeps the queue length within a desired region. However, some simulation and analysis results [3] [4] [5] have demonstrated that the performance of RED is very sensitive to parameter settings. Based on the original idea of RED, there have been some modifications such as Stabilized RED (SRED) [6], Flow RED (FRED) [7], Weighted RED [8], Random Early Marking (REM) [9], BLUE [10] and Adaptive RED [11] [12]. The Adaptive RED scheme dynamically updates the maximum dropping probability according to the exponentially weighted moving average (EWMA) of the queue length, and makes itself more robust with respect to the congestion level.

The Adaptive RED policy provides good rate control for TCP connections operating in the congestion avoidance phase [13] [12]. However, a great portion of Internet traffic is web and UDP traffic. Since most web connections involve transfer of several small files, these connections have a short life and are mostly operated in the TCP slow start phase with a small congestion window. Dropping web packets in this phase is not an effective way to control the traffic rate and alleviate the congestion at the bottleneck router. Furthermore, from the viewpoint of a web user, one or several packet losses in the slow start phase would lead to extra delay for retransmission or even TCP timeout. It would also force TCP to enter the congestion avoidance phase prematurely with a small congestion window and result in a low throughput. The delay and low throughput would severely degrade the performance of delivering short messages such as web pages, and web browsers experience long waiting times even with a high speed network. On the other hand, the Adaptive RED fails to maintain the queue length within the desired region due to the bursty nature of web traffic. To address

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

these problems, we propose a parallel virtual queue structure for active queue management in this paper. In this structure, real time (web, UDP) and non-real time (FTP) traffic are separated into two different virtual queues which share the same physical buffer memory. The drop-tail policy is applied at the first virtual queue to serve real time applications. 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 non-real time traffic is directed to an Adaptive RED virtual queue. Simulation shows that this parallel virtual queue structure not only has the advantages of Adaptive RED such as high link utilization and small delay, but also greatly reduces the total packet loss rate at the router. Despite that the bandwidth is shared with the bursty drop-tail virtual queue, the Adaptive RED queue has a small length variation.

The original Adaptive RED dynamically changes the maximum dropping probability P_{max} to keep the queue length within the thresholds. However, for some non-real time applications, high goodput (low packet dropping rate) is more important than short packet delay. Hence we explore a modified Adaptive RED policy for the non-real time applications at the second virtual queue, where the queue length thresholds are dynamically adjusted to maintain the dropping probability of Adaptive RED algorithm in a desired range.

The remainder of the paper is organized as follows. In Section II, we demonstrate the vulnerability of the Adaptive RED in the presence of web and UDP traffic. The parallel virtual queue structure is described in Section III. Comparison of this approach with the original Adaptive RED scheme is given through simulation in Section IV. In Section V, we present the modified Adaptive RED policy with dynamic queue length thresholds. Performance analysis is provided for both virtual queues (the drop-tail and the modified Adaptive RED queue) in Section VI. Finally we conclude in Section VII.

II. Vulnerability of Adaptive RED to Web-mice

In this section we consider 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 ns2 experiment has a simple dumbbell topology with the bottleneck link bandwidth C=3.0$Mbps$. One side of the bottleneck consists of 800 web clients. Each client sends a web request and has a think time of $Exponential$ distribution with mean 50s after the end of each session. The other side contains 800 web servers, running HTTP 1.1 protocol and having a $Pareto$ [14] file size distribution with parameters (K_p=2.3$Kbytes$, α=1.3) (mean 10$Kbytes$). The round-trip propagation delay of HTTP connections is uniformly distributed in (16, 240)ms. Note that the mean rate of the aggregate web traffic is around 1.2$Mbps$. There is one CBR traffic source which periodically generates a 1$Kbytes$ 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 the Adaptive RED works well with those FTP connections before the web traffic comes in. However, after the CBR source and web servers begin to share the bandwidth at time t=100s, the queue length of Adaptive RED deviates dramatically from the desired region. Since 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 the 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.

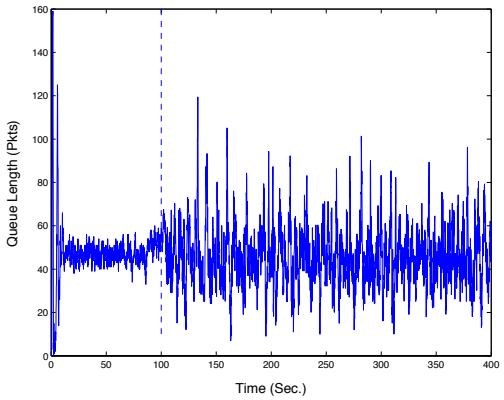

Fig. 1. Queue length of the Adaptive RED: 10 FTP starting at t=0 and 800 WEBs and 1 CBR coming in at t=100s.

Since most web pages contain one or several very small files, these TCP connections are mostly 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 the flow rate is low. Dropping packets in the slow start phase cannot efficiently alleviate the congestion level at the bottleneck router. In other words, any 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. This is further illustrated below.

From the viewpoint of a web browser, a short-life TCP session may only need several round-trip times (RTT) to finish the whole transmission. When the sender senses a packet loss, the slow start threshold $ssthresh$ will be reduced to $min(cwnd, rcv\_window)/2$ [1] and the new congestion window size $cwnd$ is also decreased depending on different TCP versions (For TCP Reno, the new $cwnd$ = $ssthresh$ and TCP enters the fast recovery phase. For TCP Tahoe, $cwnd$ = MSS (maximum segment size) and TCP begins a new slow start phase). Since original $cwnd$ is just beginning to increase its size from its initial value MSS in the 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 $ssthresh$. Therefore, losing one packet in the slow start phase takes TCP a long time to complete a short message. In addition, since the web traffic is short but bursty, these web connections usually experience a higher packet loss rate (see the web packet loss rates of the Adaptive RED and the drop-tail policies in Table III).

The default initial value of $ssthresh$ is $64KB$ and the packet size is $1KB$ in this paper. Assuming a typical packet dropping probability P_d=0.04 when using the Adaptive RED, the probability of losing one or more packets in the slow start phase is equal to $1-(1-P_d)^{64}$ = 0.9267 (assuming that packets are dropped independently). Therefore, most TCP connections have at least one packet dropped in their first slow start phase. For example, assuming that the 15^{th} packet is dropped by the Adaptive RED, $ssthresh$ decreases from $64KB$ to $4KB$ and the new congestion window $cwnd$ is decreased from $8KB$ to $1KB$ (Tahoe). The situation gets 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 to generate 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 also cannot be ignored. These events lead to a high probability of TCP timeout and connection reset.

For illustration we conduct simulation of transferring a small web file in a stand alone and one hop environment. There is no other traffic sharing the bandwidth and packets are dropped intentionally. Figure 2 shows the mean delivery delay $v.s.$ the dropping probability for file sizes $30KB$-$210KB$, and Table I lists the mean and standard deviation of the delay. For example, TCP takes about $4.81s$ to complete transmission of a $90KB$ file if $P_d = 0.04$; in comparison, in the loss free case, the file can be delivered in $1.18s$. Since most web pages have sizes in the above range, a web browser will experience a long response time when the dropping probability of the Adaptive RED is high.

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)

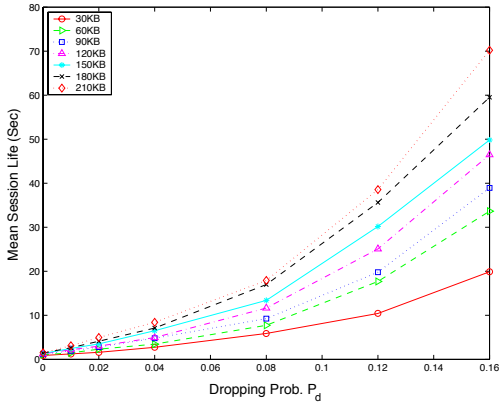

Fig. 2. Mean delivery delay of small file $v.s.$ dropping probability P_d with file sizes $30, 60, ..., 210Kbytes$, bandwidth $3Mbps$ and round-trip time $128ms$.

III. A Parallel Virtual Queues Structure

To solve the problem discussed in Section II, we propose a parallel virtual queue structure in the router. The first virtual queue deals with the short-life real-time traffic (web, UDP). Since dropping these packets cannot effectively alleviate the congestion level, but severely increases delivery delay, it would be good to keep them in the queue unless the total buffer (shared with the other queue) has overflowed. 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 serves long-life TCP connections such as FTP with large Ţle sizes, where the Adaptive RED is used. Although the available bandwidth of this queue is determined by $C_2(t)$=C-$C_1(t)$, the Adaptive RED scheme is expected (and will be veriŢed by simulation in Section IV) to keep its virtual queue length $q_2(t)$ in a desired region for the following reason. When there is a heavy workload at the drop-tail queue, $C_2(t)$ decreases quickly. FTP receivers experience slower packet arrival rates and send acknowledgement packets (ACK) back more slowly. Without increasing the dropping probability at the Adaptive RED queue, the slower ACK arrival rates from the receivers make FTP senders reduce ţow rates automatically without shrinking their congestion window sizes. On the other hand, when the congestion level is alleviated, the Adaptive RED queue receives more bandwidth. Since the congestion window sizes are still large in the FTP servers, the throughputs of FTP is quickly recovered by faster arrival rates of ACK packets from the receivers.

With this parallel virtual queue structure (which will be called RED+Tail policy in this paper), we can keep the beneŢts of Adaptive RED such as high (100%) link utilization. Furthermore, the packet loss rate of the short-life TCP and UDP connections is greatly reduced by the drop-tail policy and a shared buffer. The packet loss rate of long-life TCP traffic is also reduced due to the suppressed bandwidth, larger thresholds (longer RTT) and more stable average virtual queue length for the Adaptive RED queue.

We now discuss how to implement the RED+Tail policy. The Ţrst problem is how to split the long-life traffic from other short-life web traffic at the router. 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 arrives with a *new* source-destination pair which has not been seen by the router in the past T sec, 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, we assume that the Ţle size is large enough and this TCP connection has already 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 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 small Ţle sizes and short session times, there is no harm if the short-life connection packets are misdirected to the RED queue after time T.

Figure 3 shows the RED+Tail parallel queue structure in the router. Recall that $C_1(t)$ and $C_2(t)$ denote the service rates of the drop-tail queue and the Adaptive RED queue at time t respectively. In order to allocate bandwidth dynamically to both queues and assign a desired region of queue length for the Adaptive RED queue, we deŢne the maximum threshold $maxth_i$ and minimum 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 $0 < q_1 < minth_1$, 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 variable 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.

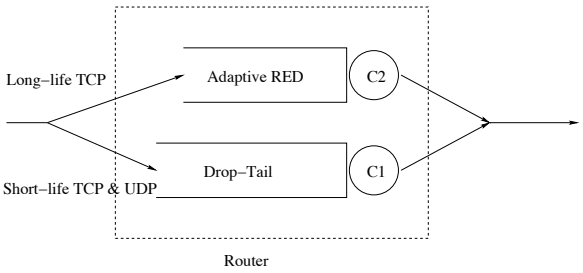

Fig. 3. The parallel virtual queue structure for active queue management.

IV. Simulation and Comparison

In this section, we compare the RED+Tail scheme with the Adaptive RED on typical TCP performance metrics. For the Adaptive RED, we use the parameter settings suggested by Floyd *et al* [12] (α and β of the AIMD algorithm). Both schemes were implemented in the ns2 simulator. The network topology and scenario are as described in Section II. Table II lists the parameters for the RED+Tail policy and the Adaptive RED policy. 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 meaning that the dropping probability between $(maxth_2, 2maxth_2)$ is linear in $(P_{max}, 1)$.

TABLE II
EXPERIMENT SETTINGS

Virtual Qu. 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 are more concerned about packet loss rate and delay for web (short-TCP) and CBR (UDP) connections, and more concerned about throughput for FTP (long-TCP). We replaced the Adaptive RED with RED+Tail scheme at the router and repeated the experiment of Section II. For comparison, an experiment with the drop-tail policy was also conducted. The random seed of the simulator was ţxed so that the processes of web requests and ţle sizes had the same sample paths in all experiments. Table III lists the performance metrics under RED+Tail, the Adaptive RED and the traditional drop-tail scheme respectively.

Figure 4 shows the queue lengths of the RED+Tail scheme, which demonstrates that the virtual queue length q_2 is quite stable and stays in the desired region even after the web and CBR trafţc begins to share the bandwidth at time t=100s. The actual dropping probability for the FTP trafţc is reduced from 4.15% to 2.75% by a longer queuing delay ($184ms$, see Table III). This scheme prevents the over-reaction behavior of RED in the original Adaptive RED case and keeps the mean queue length q_2 in a desired region (Compare to Figure 1).

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

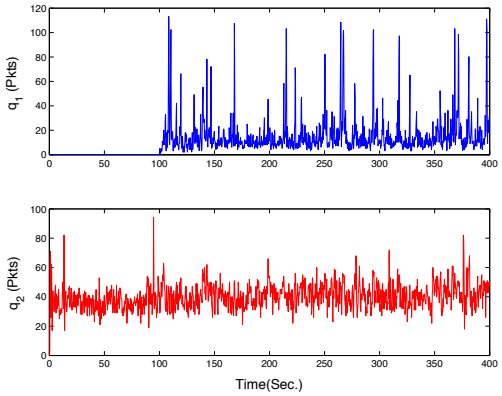

Fig. 4. Queue lengths of RED+Tail virtual queues: 10 FTPs starting at t=0 go to virtual queue 2, and 800 WEBs + 1 CBR starting at t=100 go to virtual queue 1.

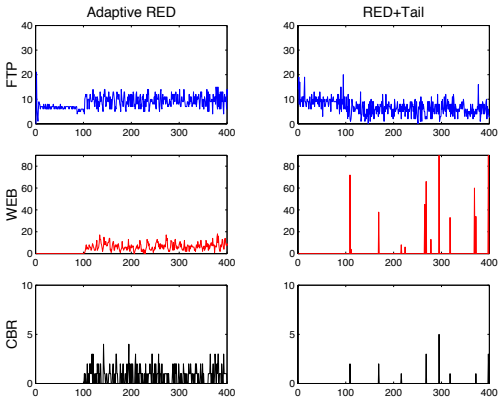

Fig. 5. Packet losses (packets/sec.) of Adaptive RED and RED+Tail.

Figure 5 shows the packet loss rates of FTP, web and CBR connections under the Adaptive RED and RED+Tail schemes. We see that RED+Tail provides great improvement in packet loss for web and CBR connections. The web packet loss rate is reduced from 4.51% to 1.28% and CBR packet loss rate is reduced from 3.95% to 0.30%.

Figure 6 compares the packet delays. The mean queuing delay of web and CBR packets in the RED+Tail scheme is shortened at the cost of the FTP packets delay. The web and CBR packet delay depends on how much bandwidth is allocated to the drop-tail queue. One can 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 estimate of mean delay at the drop-tail queue is about $80ms$. However, the service rate C_1 reaches its maximum C_{1max} when $q_1 > maxth_1$. The actual mean delay should be larger than expected. For our simulation the mean delay of web and CBR trafţc is around $110ms$ (refer to analysis in Section VI-A).

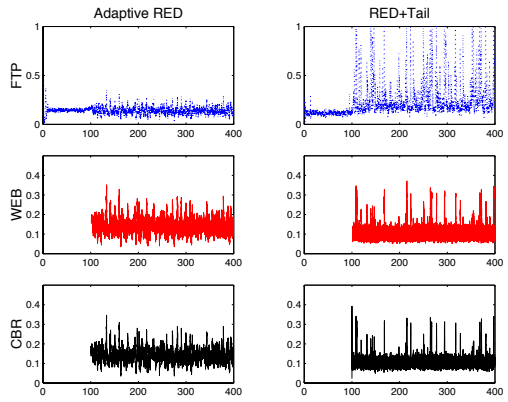

Fig. 6. Packet delays (sec.) of Adaptive RED and RED+Tail.

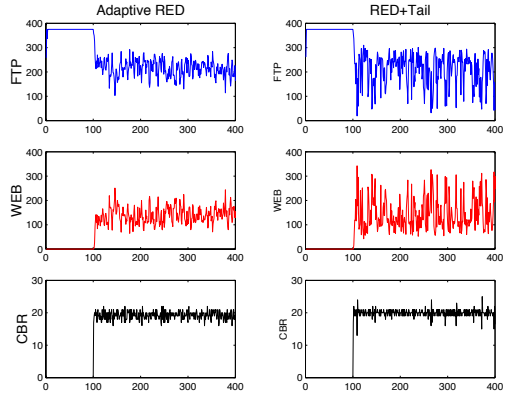

Fig. 7. Throughputs (KBytes/sec.) of Adaptive RED and RED+Tail.

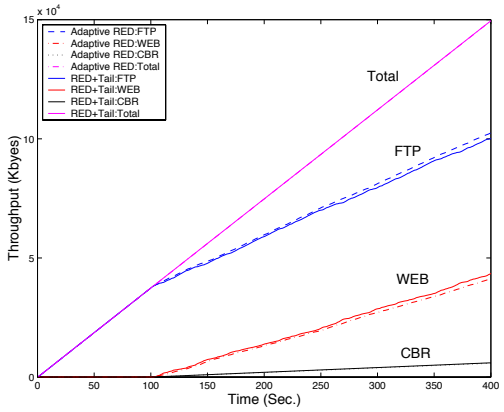

Fig. 8. Accumulated throughputs of Adaptive RED and RED+Tail.

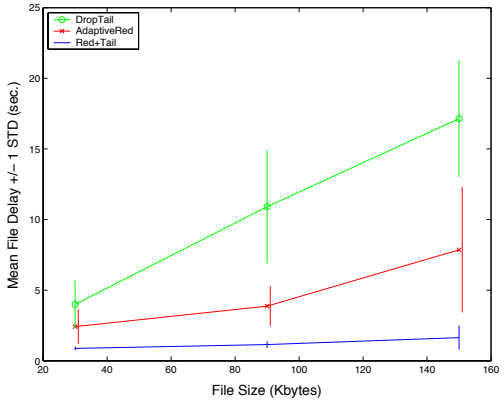

Fig. 9. Small Ṭle delivery delay: mean and standard deviation.

Figures 7 and 8 show the throughputs of FTP, web and CBR traṭc. Both schemes achieve 100% utilization of the link bandwidth. Due to the bandwidth allocation scheme 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.

Figure 9 compares the small web Ṭle 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%) as Adaptive RED for web packets. However, the Ṭle 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 $(0.340sec)$ of the drop-tail policy.

V. Dynamic Thresholds for Adaptive RED and Fairness

The Adaptive RED relies on adjusting the dropping probability to control the ṭow rates of TCP connections and keep the average queue length in a desired region. However, for those applications with large Ṭle sizes, the goodput is more important than the packet delay. The packet loss rate is a key factor in determining the connection goodput. Since the minimum and maximum thresholds of the Adaptive RED scheme are Ṭxed, the dropping probability of Adaptive RED is high when a congestion happens. This high dropping probability causes frequent retransmissions and low goodput.

To maintain a low packet loss rate, we propose the following modiṬed Adaptive RED scheme for the Adaptive RED virtual queue where $minth_2$ and $maxth_2$ are dynamically adjusted while D=$maxth_2 - minth_2$ is maintained constant.

- Pick $0 < \gamma < 1$ (γ=0.05 in this paper).
- 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 $minth_2 := minth_2(1-\gamma)$, $maxth_2 := minth_2 + D$,

where $\bar{P}_d$ is the average dropping probability obtained by the EWMA algorithm and (P_L, P_U) is the desired region of dropping probability. Note that if we set $P_U < P_{max}$, the ṭoating thresholds do not change the current slope of dropping probability function dramatically, since the

distance between the thresholds is Ţxed.

The rationale behind the above scheme is that, by increasing the thresholds (when $\bar{P}_d > P_U$), the queuing delay is increased and the ţow rates are reduced. Since the average TCP throughput is proportional to $\frac{1}{RTT\sqrt{P_d}}$, we achieve the same throughput without raising the packet loss rate. Figures 10 and 11 compare the Adaptive RED schemes with Ţxed and dynamic thresholds respectively. There are 20 persistent FTP servers sharing a $6Mbps$ bottleneck link. Another 20 FTP servers arrive at time $100s$ and leave at time $300s$. It can be seen that the Ţxed threshold scheme has a small queue length variation and a large dropping probability (0.05). In contrast, the dynamic threshold scheme has a much lower average dropping probability (0.014 with P_L=0.01, P_U=0.02), 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.

Dynamically varying the thresholds may also have implications in achieving throughput fairness among multiple Adaptive RED queues. Since the ţow rates of TCP connections are determined by the corresponding dropping probabilities and queuing delays at different queues, connections with shorter link propagation delays and higher throughputs can be suppressed by raising the queue length thresholds at the router.

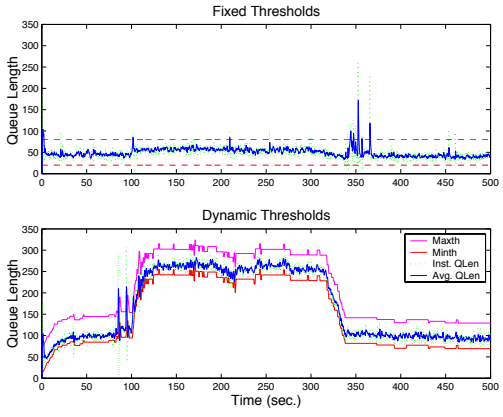

Fig. 10. Average queue length with Ţxed and dynamic thresholds: 20 FTP starting at t=0, and another 20 FTP starting at t=100s and leaving at t=300s, C=6Mbps, d_k=64ms.

The NS2 simulation in Section IV is conducted again with the *modified* Adaptive RED serving the second virtual queue. Parameters (except $minth_2$ and $maxth_2$, which are dynamically adjusted) used are as listed in Table II. The desired region of dropping probability for the

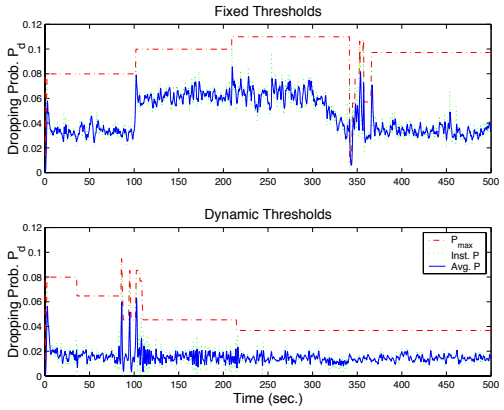

Fig. 11. Dropping probability with Ţxed and dynamic thresholds: 20 FTP starting at t=0, and another FTP 20 starting at t=100s and leaving at t=300s, C=6Mbps, d_k=64ms (Inst. P: instantaneous dropping probability; Avg. P: EWMA average of Inst. P).

TABLE IV

PERFORMANCE METRICS:RED+TAIL WITH DYNAMIC THRESHOLD SCHEME

Policy	Loss %	Delay Sec.	Rate KB/s
Dyn. Thres.:FTP	0.899	0.318	209.455
Dyn. Thres.:WEB	2.306	0.093	144.505
Dyn. Thres.:CBR	0.519	0.091	19.827

Adaptive RED queue is set to be (P_L,P_U)=(0.005,0.010). Figure 12 shows the lengths of both virtual queues and the dropping probability at the Adaptive RED queue. The dropping probability stays in the desired region most of the time as expected. Note that the ţow rate of FTP connections are reduced without increasing the queue length $q_2(t)$ and the dropping probability dramatically when the bursty web traffţc arrives at t=100. This is because that the available bandwidth for FTP connections is reduced and FTP senders see a longer round-trip time (longer packet delay at q_2, see Figure 14).

Figures 13 and 14 show the packet losses and delays for FTP, web and CBR connections respectively. Table IV collects the corresponding performance metrics. Comparing to Figure 5, 6 and Table III, The packet loss rate of FTP connection is reduced from 2.747% to 0.988% at the cost of packet delay (increased from $0.184s$ to $0.318s$). Since the average queue length at the Adaptive RED queue is about $80KBytes$ instead of $60KBytes$ in the Ţxed threshold scheme, web and UDP packets see a smaller shared buffer at the drop-tail queue and experience a higher loss rate from 1.278% to 2.306% and from 0.300% to 0.519%, respectively. However, the average delays of web and UDP packets are slightly shorter for a smaller shared buffer space at the drop-tail queue. The delay and loss at the drop-tail queue can be improved by increasing the buffer size

and using a more aggressive bandwidth allocation policy (smaller $maxth_1$). Finally, Tables III and IV also show that the throughputs for the Ţxed threshold scheme and the dynamic threshold scheme are almost the same.

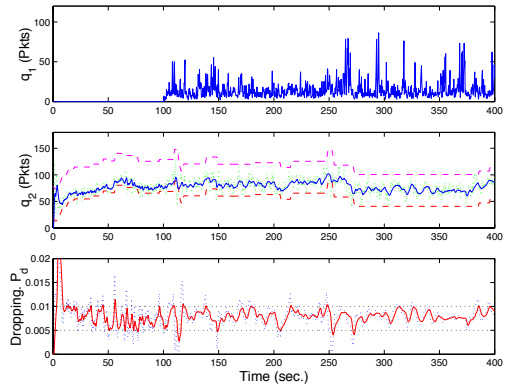

Fig. 12. Dynamic threshold scheme: Virtual queue lengths of RED+Tail and dropping probability of the Adaptive RED queue, 10 FTPs starting at t=0 and 800 WEBs + 1 CBR starting at t=100.

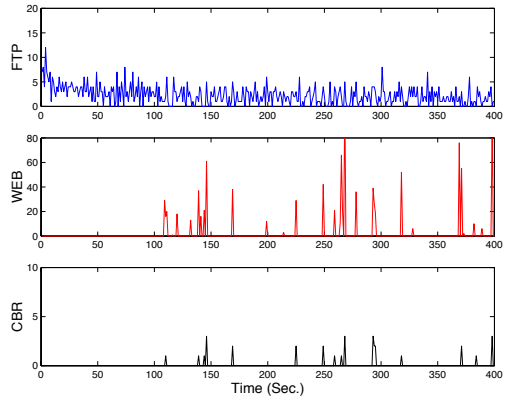

Fig. 13. Dynamic threshold scheme: Packet losses (packets/sec.) of RED+Tail.

VI. Performance Analysis for the Parallel Virtual Queues

A. Drop-Tail Queue with Adaptive Service Rate

First, we investigate the queuing delay of CBR and web traffic at the drop-tail queue. Note that the service rate $C_1(t)$ of this queue is a function of $minth_1$, $maxth_1$ and the current length of drop-tail queue $q_1(t)$ (Figure 15).

Without loss of generality, we let $C_{1min} = 0$ and $C_{1max} = C$ in the analysis. When a packet enters the drop-tail queue at time t, it sees an instant queue length $q_1(t)$ and a service rate

$$C_1(t) \triangleq \max(C_{1min}, \min(\frac{q_1(t)C}{maxth_1}, C_{1max})). \quad (1)$$

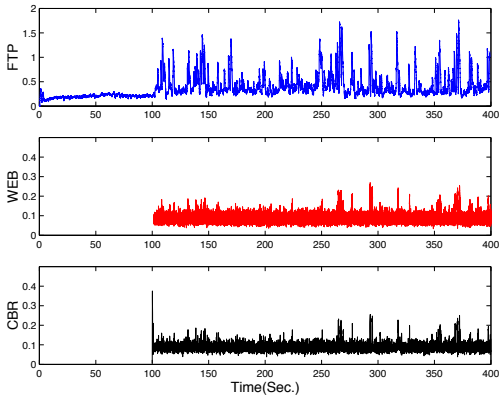

Fig. 14. Dynamic threshold scheme: Packet delays (sec.) of RED+Tail.

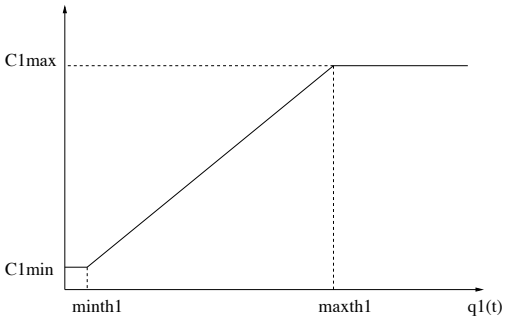

Fig. 15. Dynamic bandwidth allocation at the drop-tail queue: $C_1(t)$=max(C_{1min}, min($\frac{Cq_1(t)}{maxth_1}$, C_{1max})).

Since $C_1(t) \leq \frac{q_1(t)C}{maxth_1}$, $maxth_1/C$ is a lower bound of the average queuing delay at the drop-tail queue.

We redo the experiment in Section IV with parameters listed in Table II except $maxth_1$ being varied from $10KB$ to $80KB$. Figure 16 shows the mean packet delay of CBR and web traffic at the drop-tail queue and that of FTP traffic at the Adaptive RED queue as $maxth_1$ is varied. The simple lower bound of average queuing delay $maxth_1/C$ is seen to provide a good approximation. Note that the packet delay at the Adaptive RED queue with Ţxed thresholds is almost a constant even when $maxth_1$ is decreasing. That is because the Adaptive RED queue has Ţxed thresholds ($minth_2$, $maxth_2$) and the average queue length of RED queue is always around $\bar{q}_2$=($minth_2$+$maxth_2$)/2. Since the dynamic bandwidth allocation policy does not dramatically change the long-term average of bandwidth allocation $\bar{C}_2$, the mean delay at the Adaptive RED queue is around $\bar{q}_2/\bar{C}_2$.

For some real time applications such as video conference and voice, small delay jitter is very important for the connection quality. Figure 16 also shows that the drop-tail queue has a very small delay variance. Note that the delay variance at the Adaptive RED queue is slightly increased when a smaller value of $maxth_1$ at the

drop-tail queue is applied. According to these results, the mean packet delay requirements at both queues can be satisfied by properly designing the values of ($minth_1$, $maxth_1$) and ($minth_2$, $maxth_2$).

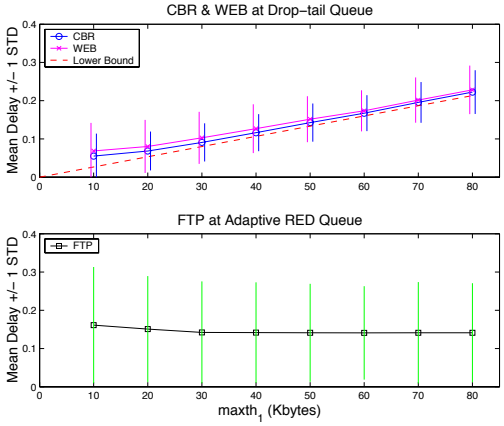

Fig. 16. Mean delays (sec.) of CBR and WEB packets at the drop-tail queue and mean delay of FTP packets at the Adaptive RED queue (with fixed thresholds) with $maxth_1$= 10, 20,...,80 (KBytes).

B. Adaptive RED Queue with Dynamic Thresholds

In Section V we proposed the modified Adaptive RED scheme with dynamic thresholds in the parallel queue structure for controlling the flow rate of non-real time applications. The maximum threshold $maxth_2$ and minimum threshold $minth_2$ are changed dynamically to keep the packet dropping probability P_d within a desired small region (P_L, P_U) at the cost of packet delay variation. In this subsection we analyze issues related to the stability of this virtual queue. For ease of analysis, it is assumed that the dropping probability P_d of the Adaptive RED at the bottleneck router is fixed so that the average flow rate of each TCP connection can be approximated by a simple function of its round-trip time. Note that this assumption is not very restrictive considering the interval (P_L, P_U) is small.

Consider N persistent TCP flows. Define T_k^n as the average flow rate of the k^{th} TCP connection during time slot n. Let d'_k be the link round-trip propagation delay of connection k. At the beginning of time slot n the k^{th} connection sees a round-trip time R_k^n, which is equal to the sum of link propagation delay and the average queuing delay in the forward direction q^n/C and in the backward direction q_b^n/C:

$$R_k^n = d'_k + \frac{q^n}{C} + \frac{q_b^n}{C}, \tag{2}$$

where C is the link bandwidth, q^n and q_b^n are the forward queue length and the backward queue length at the beginning of time slot n, respectively. We assume that congestion only happens in the forward direction and the queuing delay q_b^n/C in the backward direction is a constant. Hence we can write $R_k^n = d_k + q^n/C$ with $d_k = d'_k + q_b^n/C$.

Based on the assumption of fixed dropping probability at the router, each TCP connection experiences a fixed packet loss rate P_d and the corresponding average congestion window size is assumed to be a constant $\bar{W}$. Hence, the average flow rate T_k^n of the k^{th} TCP connection at slot n is

$$T_k^n = \frac{\bar{W}}{R_k^n} + E_k^n \tag{3}$$

where E_k^n is a white Gaussian process with zero mean and variance σ^2 modeling the flow rate perturbation of the k^{th} connection at slot n.

Given the arrival rate of each TCP connection, the dynamics of queue length q^n follows the $Lindley$ equation:

$$q^{n+1} = min\{B, max[0, q^n + (\sum_{k=1}^{N} T_k^n - C)S]\}, \tag{4}$$

where B is the buffer size and S is the duration of one time slot. Since the queue length of Adaptive RED is mostly operated in a region far from the boundary, we first ignore the max and min operations in (4) and have a simplified nonlinear dynamic system:

$$q^{n+1} = f(q^n) + \xi^n, \tag{5}$$

where

$$f(q^n) \triangleq q^n + S\{(\sum_{k=1}^{N} \frac{\bar{W}C}{q^n + d_k C}) - C\}, \tag{6}$$

and

$$\xi^n \triangleq S \sum_{k=1}^{N} E_k^n. \tag{7}$$

To avoid the trivial case $q \equiv 0$, we assume that the sum of possible peak rates of all connections is greater than the link bandwidth at the bottleneck router:

$$\sum_{k=1}^{N} \frac{\bar{W}}{d_k} \geq C. \tag{8}$$

Figure 17 shows the queue length dynamics (and the throughput of a persistent TCP connection) based on the model (5), where the flow rate deviations σ = 77021, 128490 (bits/s) for N=20, 40 are measured from the simulation in Section V, respectively. For both $N = 20$ and $N = 40$, Figure 17 shows consistent steady state behavior with simulation results in Figure 10. The

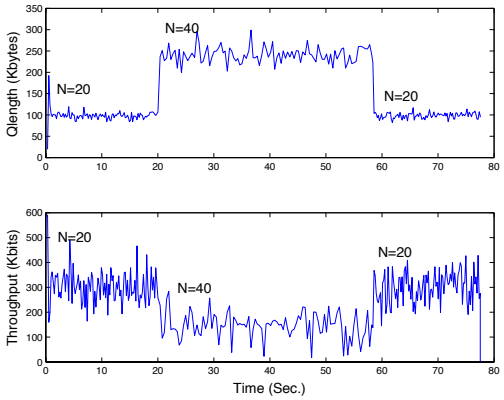

Fig. 17. Queue length and TCP throughput (of a single connection) with C=6Mbps, d_k=64ms, $\bar{W}$=6.02×10^4 bits. Compare with simulation in Fig.10.

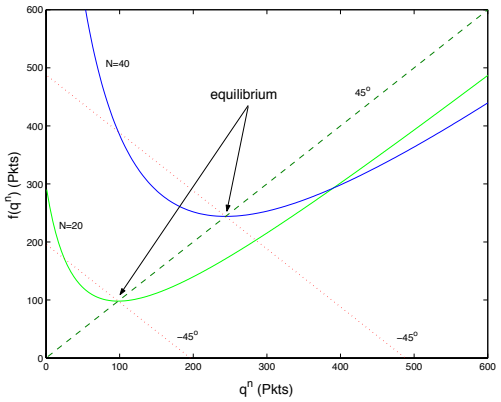

Fig. 18. Mapping functions and equilibrium points when N=20, 40 with S=RTT.

mapping $f(\cdot)$ is plotted in Figure 18 for $N = 20$ and $N = 40$.

We first analyze the stability of the equilibrium of the model (5) when there is no flow disturbance., *i.e.*, $E_k^n = 0$. An equilibrium q_e of $q^{n+1} = f(q^n)$ should satisfy

$$\sum_{k=1}^{N} \frac{\bar{W}}{d_k + q^n/C} = C. \tag{9}$$

Since $\sum_{k=1}^{N} \frac{\bar{W}}{d_k} \geq C$ by assumption, (9) has a unique solution q_e in $[0, \infty)$. q_e is located at the intersection of the graph of f with the 45^o line (see Figure 18).

It is well known that q_e is locally asymptotically stable if $|f'(q_e)| < 1$. In the following we give conditions for q_e to be *globally* asymptotically stable.

Proposition 6.1: If the rate update interval S satisfies

$$S < \frac{2C}{\bar{W}(\sum_{k=1}^{N} d_k^{-2})}, \tag{10}$$

the equilibrium q_e is globally asymptotically stable. Furthermore, $|q^n - q_e| < \rho^n |q^0 - q_e|$ for some $\rho \in (0, 1)$ dependent on q^0.

Proof: First we observe that the function f is convex since

$$f''(q) = \sum_{k=1}^{N} \frac{2S\bar{W}C}{(q + d_k C)^3} > 0, \quad \forall q \in [0, \infty). \tag{11}$$

For any B_0 such that $B_0 > q_e$ and

$$B_0 \geq f(0) = (\sum_{k=1}^{N} \frac{\bar{W}}{d_k} - C)S, \tag{12}$$

one can verify that f maps $[0, B_0]$ to $[0, B_0]$ due to convexity of f and

$$f'(q) = 1 - S\sum_{k=1}^{N} \frac{\bar{W}C}{(q + d_k C)^2} < 1, \quad \forall q \in [0, \infty). \tag{13}$$

When restricted to $[0, B_0]$, $f'(q) \leq \rho_1$ with $\rho_1 \in (0, 1)$. If (10) is satisfied, $f'(q) > -1, \forall q \in [0, \infty]$, and $f'(q) \geq -\rho_2, \forall q \in [0, B_0]$, with $\rho_2 \in (0, 1)$.

Hence $|f'(q)| \leq \rho \stackrel{\triangle}{=} \max(\rho_1, \rho_2) < 1 \ \forall q \in [0, B_0]$, which implies that f is a contraction mapping on $[0, B_0]$. By the Contraction Mapping Principle [15],

$$|q^n - q_e| < \rho^n |q^0 - q_e|, \quad \text{if } q^0 \in [0, B_0]. \tag{14}$$

Since B_0 can be arbitrarily large (as long as $B_0 < \infty$), q_e is globally asymptotically stable. Note that the contraction constant ρ depends on B_0 and thus on q^0. □

From Proposition 6.1, when rate update is frequent enough, the equilibrium will be asymptotically stable (the equilibrium itself does not depend on S). Another sufficient condition for asymptotic stability is the following:

Proposition 6.2: If $f'(q_e) \geq 0$, then q_e is a globally asymptotically stable.

Proof: As shown in the proof of Proposition 6.1, f is convex. If $f'(q_e) \geq 0$, graphical analysis reveals that

$$|q^{n+1} - q_e| \leq |q^n - q_e|,$$

where the equality holds if and only if $q^n = q_e$. The claim thus follows. □

For the homogeneous case $d_k = d$, we have $q_e = N\bar{W} - dC$. And the condition $f'(q_e) \geq 0$ is equivalent to $S \leq \frac{N\bar{W}}{C} = q_e/C + d$. In other words, q_e is asymptotically stable if the rate update interval S is no larger than the round-trip time (RTT). Figure 19 shows the mapping f and the equilibrium q_e for different S. Figure 20 shows the queue length dynamics (noise is included) for S=RTT and $2RTT$, respectively. We can see that in the case S=RTT, the queue length stays around q_e with small variation, while in the case S=$2RTT$, the queue length dynamics is much more chaotic.

For the heterogeneous case, a sufficient condition $S \leq (\frac{C}{W} - \frac{N(N-1)}{(q_e+D_m)^2})^{-1}$ for stability can be derived.

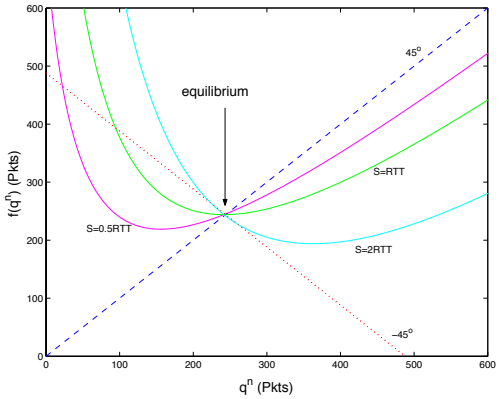

Fig. 19. Mapping function and equilibrium point when N=40 with S=0.5RTT, 1RTT and 2RTT.

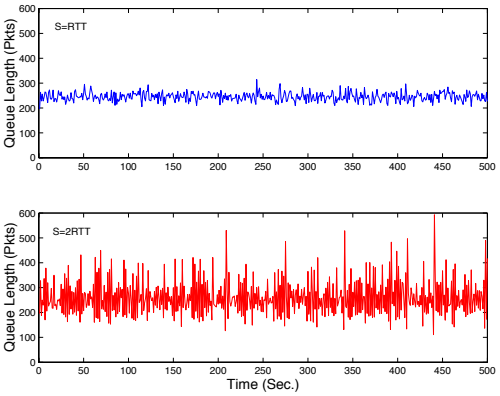

Fig. 20. Queue length with N=40, S=RTT and 2RTT.

Next we consider the *Lindley* equation with random perturbation $\xi^n = S\sum_{k=1}^{N} E_k^n$:

$$q^{n+1} \stackrel{\triangle}{=} g(q^n, \xi^n) = \min\{B, \max[0, f(q^n) + \xi^n]\} \quad (15)$$

Note that since $\{E_k^n\}$ is white and stationary, so is $\{\xi^n\}$. It turns out that stability of the equilibrium of the deterministic system $q^{n+1} = f(q^n)$ is closely related to stochastic stability of the system (15).

Proposition 6.3: The stochastic system (15) admits an invariant probability measure μ^* for the queue length q^n. Furthermore, if the condition (10) on Proposition 6.1 is satisfied, this system is weakly asymptotically stable, *i.e.*, the queue length distribution μ^n for q^n converges to μ^* weakly.

Sketch of Proof. Since f is continuous and $\{\xi^n\}$ is identically and independently distributed, the system (15) is a regular stochastic dynamic system [16].

Since $[0, B]$ is compact, the system admits an invariant probability measure μ^* by the *Krylov-Bogolubov* Theorem [16]. When condition (10) is satisfied, g is a contraction mapping with respect to its first argument, *i.e.*,

$$|g(x,\xi) - g(y,\xi)| < \rho|x-y|, \quad \forall x, y \in [0,B], \forall \xi, \quad (16)$$

where $\rho \in (0,1)$. Hence the system is weakly asymptotically stable by Theorem 12.6.1 of [16]. □

The invariant probability measure μ^* has probability masses at $q = 0$ and $q = B$, and has probability density on $(0, B)$. An approximation to μ^* can be obtained by numerically advancing the probability distribution μ^n for the queue length q^n. We have discretized the queue length and consequently obtained a Markov chain for the dynamics of the queue length distribution.

Let the packet size have a fixed length L (bits), z^n:=$ceil(q^n/L)$ be the number of packets in the queue at time n and $\pi^n = [Pr(z^n = 0), ..., Pr(z^n = B)]$ denote the corresponding probability vector. We have

$$\pi^{n+1} = \pi^n T, \quad (17)$$

$$\pi^* = \pi^* T, \quad (18)$$

where $\pi^* = \lim_{n\to\infty} \pi^n$ is the steady state distribution and $T(i,j) = Pr[z^{n+1} = j | z^n = i]$is the corresponding transition matrix of the Markov chain. The conditional probability $Pr[z^{n+1} = j | z^n = i]$ is obtained as

$$Pr[j \leq (min\{B, max[0, f(iL) + \xi\})/L < (j+1)]. \quad (19)$$

On the other hand, when the buffer size B is far greater than the equilibrium queue length and the perturbation magnitude is small, the transformation $g(q, \xi)$ can be linearized around the equilibrium point q_e. Let $Q^n \stackrel{\triangle}{=} q^n - q_e$. Then

$$Q^{n+1} \stackrel{\triangle}{=} f'(q_e)Q^n + \xi^n. \quad (20)$$

Since $\{\xi^n\}$ is white Gaussian process with zero mean and variance $NS\sigma^2$, $\{Q^n\}$ will be a Gaussian process with zero mean and normalized variance

$$Var[Q^n/S] = \frac{N\sigma^2}{1 - |f'(q_e)|^2}. \quad (21)$$

From (21) the normalized queue length variation will be minimal if $f'(q_e) = 0$, which corresponds to $S = RTT$ for the homogeneous case.

Figure 21 shows the queue length distributions obtained through empirical estimation from ns2 simulation, numerical computation based on (17), and linear approximation based on (21), repectively. Three distributions agree well, which verifies that our nonlinear model (15) captures the queue length dynamics under the Adaptive RED scheme with dynamic thresholds.

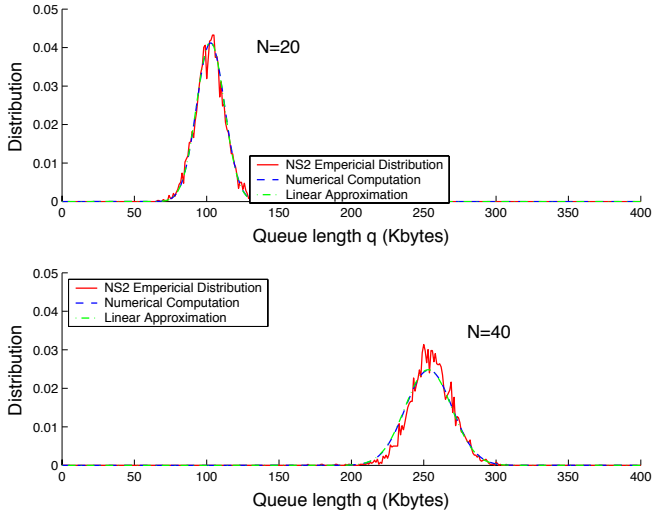

Fig. 21. Steady state queue length distributions for N=20 and 40, S=RTT.

VII. Conclusions

In this paper we have ṭrst demonstrated the vulnerability of Adaptive RED scheme to bursty web trafṭc, and then proposed a parallel virtual queue structure for active queue management at the router. 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 for short-life trafṭc can be greatly reduced by dynamic bandwidth allocation with this 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 virtual queue structure also offers more degrees of freedom for AQM due to its ṭexibility in accommodating variants of the Adaptive RED scheme and different dynamic bandwidth allocation algorithms. We have explored a modiṭed Adaptive RED scheme with sliding queue length thresholds. This scheme is able to maintain the dropping probability within a small interval and improve the goodput of non-real time connections. The queue length variation under this policy has been analyzed and conditions for it stability have been given. The dynamic threshold Adaptive RED might also be useful for achieving throughput fairness among multiple RED queues.

As to the dynamic bandwidth allocation policy for the drop-tail queue, we only used the current virtual queue length information. However, it is well-known that web trafṭc is strongly correlated and has a long range dependency property. Based on observations of the "recent past" trafṭc, the future bandwidth demand of the web trafṭc is predictable. In future work optimal bandwidth allocation based on prediction of the congestion level will be explored.

References

[1] W. R. Stevens, *TCP/IP Illustrated (Volume 1)*, Addison Wesley, 1994.

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

[3] M. Christiansen, K. Jaffay, D. Ott, and F.D. Smith, "Tuning RED for web trafṭc," in *Proceedings of SIGCOMM 00*, 2000, pp. 139–150.

[4] V. Misra, W. Gong, and D. F. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP ṭows with an application to RED," in *Proceedings of SIGCOMM 00*, 2000, pp. 151–160.

[5] P. Ranjan, E. Abed, and R. La, "Nonlinear instabilities in TCP-RED," in *Proceedings of INFOCOM 02*, 2002, pp. 249 –258.

[6] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," in *Proceedings of INFOCOM 99*, 1999, pp. 1346–1355.

[7] D. Lin and R. Morris, "Dynamics of random early detection," in *Proceedings of SIGCOMM 97*, Cannes, France, September 1997, pp. 127–137.

[8] D. Clark and W. Feng, "Explicit allocation of best-effort packet delivery service," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 362–373, August 1998.

[9] S. Athuraliya, S. Low, V. Li, and Q. Yin, "REM: Active queue management," *IEEE Network*, vol. 15, no. 3, pp. 48–53, 2001.

[10] W. Feng, D. Kandlur, D. Saha, and K. G. Shin, "BLUE: A new class of active queue management algorithms," Tech. Rep. U. Michigan EECS CSE-TR-387-99, 1999.

[11] W. Feng, D.D. Kandlur, D. Saha, and K.G. Shin, "A self-conṭguring RED gateway," in *Proceedings of INFOCOM 99*, 1999, pp. 1320–1328.

[12] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An algorithm for increasing the robustness of RED," available at http://www.icir.org/ṭoyd/papers/adaptiveRed.pdf, Aug. 2001.

[13] Van Jacobson, "Congestion avoidance and control," in *Proceedings of SIGCOMM 88*, 1988, pp. 314–329.

[14] M. Taqqu, W. Willinger, and R. Sherman, "Proof of a fundamental result in self-similar trafṭc modeling," *Comput. Comm. Rev.*, vol. 26, pp. 5–23, 1997.

[15] D. R. Smart, *Fixed Point Theorems*, Combridge University Press, London, New York, 1974.

[16] A. Lasota and M.C. Mackey, *Chaos, Fractals, and Noise: Stochastic Aspects of Dynamics*, Springer-Verlag, second edition, 1994.