Modeling Random Early Detection in a Differentiated Services Network

Alhussein A. Abouzeid, Sumit Roy

Abstract

An analytical framework for modeling a network of Random Early Detection (RED) queues with mixed traffic types (e.g. TCP and UDP) is developed. Expressions for the steady state goodput for each flow and the average queuing delay at each queue are derived. The framework is extended to include a class of RED queues that provides differentiated services for flows with multiple classes. Finally, the analysis is validated against *ns* simulations for a variety of RED network configurations where it is shown that the analytical results match with those of the simulations within a mean error of 5%. Several new analytical results are obtained; TCP throughput formula for a RED queue; TCP timeout formula for a RED queue and the fairness index for RED and Tail-Drop.

Keywords

Random Early Detection, TCP, UDP, differentiated services, performance analysis, network modeling.

I. Introduction

The diverse and changing nature of service requirements among Internet applications mandates a network architecture that is both flexible and capable of differentiating between the needs of different applications. The traditional Internet architecture, however, offers best-effort service to all traffic. In an attempt to enrich this service model, the Internet Engineering Task Force (IETF) is considering a number of architectural extensions that permit service discrimination. While the resource reservation setup protocol (RSVP) [1], [2] and its associated service classes [3], [4] may provide a solid foundation for providing different service guarantees, previous efforts in this direction [5] show that it requires complex changes in the Internet architecture. That has led the IETF to consider simpler alternatives to service differentiation. A promising approach, known as Differentiated Services (DiffServ) [6], [7], [8] allows the packets to be classified (marked) by an appropriate class or type of service (ToS) [9] at the edges of the network, while the queues at the core simply support priority handling of packets based on their ToS.

Another Internet-related quality of service (QoS) issue is the queue's packet drop mechanism. Active queue management (AQM) has been recently proposed as a means to alleviate some congestion control problems as well as provide a notion of quality of service. A promising class is based on randomized packet drop or marking. In view of the IETF informational RFC [10], Random Early Detection (RED) [11] is expected to be widely deployed in the Internet. Two variants of RED, RIO [12] and WRED [13] have been proposed as a means to combine the congestion control features of RED with the notion of DiffServ.

This work aims to make a contribution towards analytical modeling of the interaction between multiple TCP flows (with different round-trip times), non-responsive flows (e.g. UDP) and active queue management (AQM) approaches (e.g. RED) in the context of a differentiated services architecture. A new framework for modeling AQM queues is introduced and applied to a network of RED queues with DiffServ capabilities termed DiffRED which is similar to the RIO and WRED algorithms. Our work complements previous research efforts in the area of

Manuscript first submitted December 7, 2000.

A. A. Abouzeid is with the Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, 110 8th St., Troy NY 12180, USA (e-mail: abouzeid@ecse.rpi.edu).

S. Roy is with the Department of Electrical Engineering, University of Washington, Box 352500, Seattle WA 98195-2500, USA (e-mail:roy@ee.washington.edu).

Part of the material in this paper (Section III.A,B) was presented at IEEE Global Communications Conference, San Francisco, November 2000, and is included in a revised form in this paper.

TCP modeling under AQM regimes. Specifically, this work has the following *unique* contributions:

C1. TCP timeout in AQM Networks: We derive an expression for the probability of a timeout that reduces the inaccuracy by as much as an order of magnitude vis-a-vis previous state-of-art [14], [15]

C2. Poisson Process Splitting (PPS) Based Analysis for TCP flows: We show by analysis and simulations that the PPS approximation holds for a population of heterogeneous (in terms of delay) TCP flows sharing a bottleneck RED queue.

C3. Fairness Results: We derive expressions for the fairness index of a population of heterogeneous TCP flows for the case of RED and compare it to Tail Drop.

C4. Network of AQM (RED) Routers: We extend the result to a network of AQM queues with TCP (and UDP) flows.

C5. Network of DiffRED Routers: Finally, the results are extended to a network of RED and DiffRED queues.

While the details of the above contributions are presented in the following sections, we first comment on the relation between the above contributions and previous/concurrent related work.

Literature Review

The steady-state throughput of a single TCP-Reno (or NewReno) flow with Tail Drop (TD) queueing policy where packet drops can be modelled as an i.i.d loss with probability p has been shown to have the well-known inverse-$\sqrt{p}$ dependance. This result was derived by several authors [16], [17], [18] using different analytic approaches; for example, [16] used a non-homogeneous Poisson process approximation of the TCP flow while [17], [18] use a fixed point approximation method, both yielding the same dependance to within a multiplicative constant. This result which we call *throughput formula1*, does not incorporate the effect of TCP timeout mechanism.

In [14], the above was extended to include timeouts for a Tail-Drop queue for scenarios where it is reasonable to assume that conditioned on a packet loss, all remaining packets in the same window of data are also lost (i.e. burst loss within the same window round). Based on this assumption, the authors derive a more accurate formula which we term *throughput formula2*, for the steady-state TCP throughput. Specifically, this derivation includes (as an intermediate step) an expression for the probability, given a packet loss, that this loss will be detected by a timeout - we term this the *the timeout formula*. An alternative timeout probability was derived in [15] with independent packet losses (i.e. without the assumption of burst loss within a loss window). However, the result is not available in closed form and requires the knowledge of the window size probability distribution function.

Building on the above two formulae for the throughput of an individual TCP flow, Bu et al. [19] extend the analysis to a *network* of RED queues by solving numerically coupled equations representing the throughput for each individual flow. A similar approach is adopted in [20], [21].

Our contribution **C4** listed above is similar to [19], [20], [21] insofar that we also seek to model a network of AQM queues with TCP flows. However, unlike [19], [20], [21], we do not use any of the formulae derived earlier. Specifically, we show that *the timeout formula* derived in [14] largely over-estimates the probability of a timeout detection for an AQM queue due to the assumption of burst loss within a loss window, via analysis that is supported by ns2 simulation. Thus in **C1**, we derive a new formula for the timeout probability and compare the accuracy of the new formula with those derived earlier.

In [11], the authors predict that the well known bias of TCP against connections with larger delay sharing a bottleneck Tail Drop queue will only be slightly affected by the introduction of RED without any definitive analytical support. Our results in **C2, C3** precisely quantify the amount of unfairness experienced by TCP flows over a bottleneck RED vis-a-vis a Tail Drop

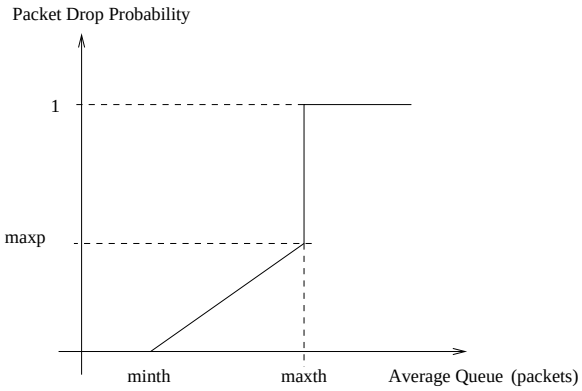

Fig. 1. RED packet drop probability as a function of the average queue size

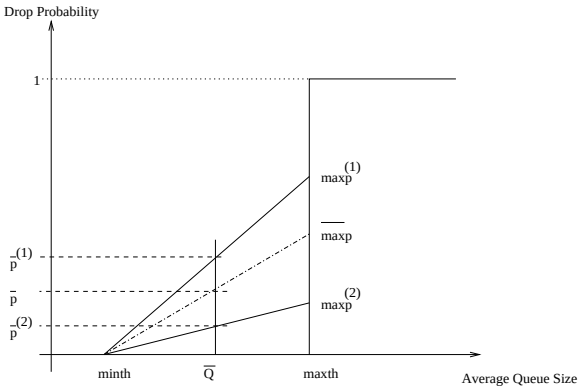

Fig. 2. Schematic of the DiffRED packet drop probability for two classes of service.

queue by using a PPS approximation (that is shown to hold within an acceptable error margin) and hence deriving a *fairness index* for a population of heterogeneous TCP flows for the two cases of RED and Tail Drop.

Paper outline

The paper is organized as follows. In Section II, we present the network model considered in this paper, describe the RED and DiffRED algorithms and outline our modeling approach. Section III considers a single congested RED queue, the PPS property is shown to hold for a population of heterogeneous flows, the analytical results for a single RED queue are validated against *ns* simulations and the fairness results of RED versus Tail-Drop are presented. Section IV presents a derivation of our *timeout formula* and a comparison between our results and previous results from [14] and [15] against *ns* simulations. Section V presents the analysis of a network of DiffRED[1] queues with TCP and UDP flows. The network model validation is presented in Section VI. Section VII concludes the paper outlining future extensions.

II. Modeling and Notation

A. The RED Algorithm

We briefly review the original RED algorithm (see [11] for further details). A RED router calculates a *time-averaged* average queue size using a low-pass filter (exponentially weighted moving average) or smoother over the sample queue lengths. The average queue size is compared to two thresholds: a minimum threshold ($minth$) and a maximum threshold ($maxth$). When the average queue size is less than $minth$ no packets are dropped; when the average queue size exceeds $maxth$, every arriving packet is dropped. Packets are dropped probabilistically when the time-averaged queue size exceeds $minth$ with a probability p that increases linearly until it reaches $maxp$ at average queue size $maxth$ as shown in Fig. 1. RED also has an option for marking packets instead of dropping them.

B. The DiffRED algorithm

We propose to investigate a DiffRED algorithm that is identical to the original RED algorithm except for the following important change - DiffRED differentiates between different packets by appropriately labelling each with the class it belongs to. If a packet is not labelled, it will be assigned the default (lowest) class. The RED queue associates a *priority coefficient*, denoted by $\beta^{(c)}$, for each class $1 \leq c \leq C$ where C is the total number of classes. The lowest priority class corresponds to $\beta^{(1)} = 1$, and the c-th (higher) priority class packets have coefficient $\beta^{(c)}$,

[1]It will be shown in Section II that RED is a special case of a DiffRED queue with no service differentiation.

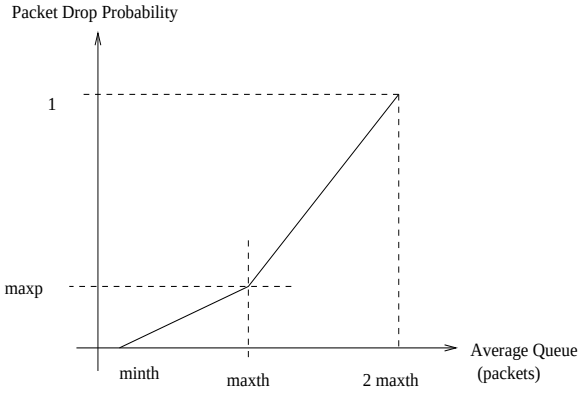

Fig. 3. Packet drop probability for the "Gentle" variant of RED.

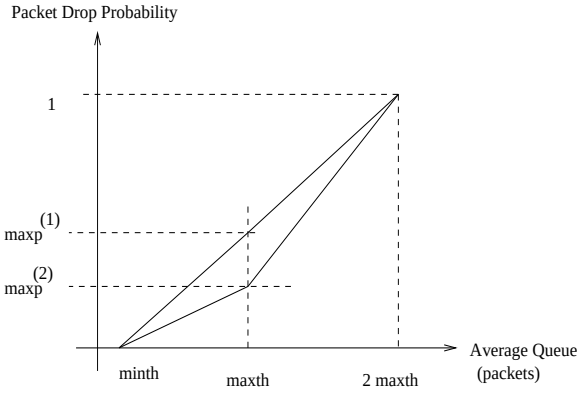

Fig. 4. Packet drop probability for the "Gentle" variant of DiffRED.

where $1 = \beta^{(1)} \geq \beta^{(2)} \geq \ldots \geq \beta^{(c)} \geq \ldots \geq \beta^{(C)}$. When a DiffRED queue receives a packet, it updates the average queue size and drops the packet with probability equal to $p\,\beta^{(c)}$ for a class c packet. Thus, for the same congestion level (i.e. average queue size), higher priority packets are dropped with a lower probability. A schematic diagram of the DiffRED dropping algorithm for the case of $C = 2$ is depicted in Figure 2. The plot shows that the packet drop probability function (as a function of the average queue size) has a lower slope for the higher priority class (for $c = 2$, $\beta^{(2)} = 0.3$), while the drop function for the lower priority class $c = 1$ has a higher slope.

C. *The* Gentle *Variant of RED*

In the recently recommended "gentle" [2] variant of RED [22], the packet-dropping probability varies from $maxp$ to 1 as the average queue size varies from $maxth$ to $2maxth$. Figures 3-4 depict the packet drop probability for RED and DiffRED with this option enabled. Our model applies equally well to both cases, but the model with the "gentle" option requires less restrictive assumptions and hence is applicable to a wider range of parameters as will be discussed in Section II-E.

D. *Network Model and Notations*

Assume a network of V queues that support M TCP flows and N UDP flows. Each TCP connection is a one way data flow between a source S_j and destination D_j, with reverse traffic consisting only of packet acknowledgments (ACKs) for each successfully received packet at the destination (i.e. no delayed-ACKs). Throughout the paper, we will assume that the TCP flows are indexed by $1 \leq j \leq M$ while UDP flows are indexed by $M < j \leq M + N$. (Figure 5 shows a network with $V = 2$, $M = 4$ and $N = 1$). Let T_j (sec.) denote the total round-trip propagation and transmission delays for the $j^{th}, 1 \leq j \leq M$ TCP flow (not including queuing delay) and let μ_v denote the v^{th} queue link capacity (packets/sec.).

Denote a zero-one $(M + N) \times V$ dimensional *order* matrix O with elements $o(j, v), 1 \leq j \leq M + N, 1 \leq v \leq V$ that specifies the order in which the j^{th} flow traverses the v^{th} queue. A 0 entry indicates that the corresponding flow does not pass through that queue. Since each flow passes through at least one queue, each row in O must have at least one entry equal to 1.

In this work, we consider TCP-NewReno and assume the reader is familiar with the key aspects of the window adaptation mechanism [23], [24] - i.e., the two modes of window increase (slow start and congestion avoidance) and the two modes of packet loss detection (reception of multiple ACKs with the same next expected packet number or timer expiry). Our analysis

[2]This option makes RED much more robust to the setting of the parameters $maxth$ and $maxp$.

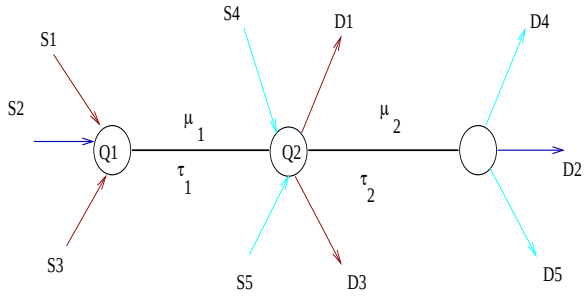

Fig. 5. Experimental network topology with two RED queues.

applies best to TCP-NewReno since we assume that multiple packets within the same loss window will trigger at most one duplicate ACK event (rather than one for each) [24].

E. Modeling Approach

The primary challenge facing any analytical approach is to capture the complex interplay of TCP congestion control algorithm and active queue management policies as RED. We adopt a mean value analysis (using fixed point approximation) to estimate (long-term) averages of various parameters as done in [17], [18], [19], [25], [26], while recognizing that such analysis does not allow for accurate prediction of higher order statistical information (e.g. variances).

We next state and discuss the following modeling assumptions:
A1: A congested queue is fully utilized (except possibly at some isolated instants).
A2: Each TCP flow passes through at least one congested queue.
A3: The probability of a forced packet loss is negligible.[3]

A1 is a well known result that has been observed in the literature through Internet measurements as well as simulations and was also confirmed in all the simulations we performed. **A2** is reasonable if the TCP flows window size are not limited by the receiver's advertised window sizes. **A3** implies that we model only 'unforced' (or probabilistic) RED packet losses and ignore forced losses which should be avoided when configuring RED queue parameters [4]. Note that it is much easier to configure a RED (or DiffRED) queue to avoid forced packet drops when the "gentle" option is used, as noted in [22] due to its avoidance of the abrupt increase in the drop probability from $maxp$ to 1. Nevertheless, our analysis results hold even if the "gentle" option is not used, as long as **A3** holds.

III. Single Queue

The aim of this section is two-fold: (i) we outline an analysis using the PPS approximation[5] and (ii) derive a result for the probability that a loss is detected by a timeout (TO) and not Duplicate Acknowledgment (DA). Both of these results will be used in the following section in a network (rather than a single queue) set-up.

A. PPS Based Analysis for TCP flows

Consider M Poisson processes, each with an arrival rate λ_j at a single queue that randomly drops packets with a probability p. Let $P_{i,j}$ denote the probability that the i^{th} packet loss event belongs to the j^{th} flow. From the PPS (Poisson Process Splitting) property,

$$P_{i,j} = \frac{\lambda_j}{\sum_{k=1}^{M} \lambda_k} \tag{1}$$

independent of i.

[3]Forced packet losses are those due to either buffer overflow or due to packet drop with probability one.
[4]This assumption has been adopted in many previous AQM-related research efforts (e.g. [27]).
[5]Part of this Section was presented in a preliminary form in [28].

Consider now M TCP flows - let X_i denote the i-th inter-loss duration at the queue as shown in Fig. 6 and $Q_{i,j}$ the queue size for session j packets at the end of epoch i. Then the net round-trip propagation and queuing delay for connection j at the end of the i^{th} epoch is $\gamma_{i,j} = T_j + Q_{i,j}/\mu$. Since all flows share the same buffer, $Q_{i,j} = Q_i \, \forall j$. Let $E[Q_i] = \overline{Q}$ denote the steady-state average queue size.

Let $W_{i,j}$ denote the window size of the j^{th} flow just before the end of the i^{th} epoch and $\overline{W_{av_j}}$ denote flow j time-averaged window size (as opposed to $E[W_{i,j}] = \overline{W_j}$ that represents the mean window size just prior to the end of the epoch).

We postulate that, for M TCP flows sharing a queue,

$$P_{i,j} = \frac{\frac{W_{i,j}}{\gamma_{i,j}}}{\sum_{k=1}^{M} \frac{W_{i,k}}{\gamma_{i,k}}} \tag{2}$$

Equation 2 is a postulate that the PPS approximation (1) holds for a population of TCP flows sharing a random drop queue. To justify the above we first consider the case of TCP flows that are perpetually in the congestion avoidance phase (additive increase - multiplicative decrease (AIMD) mode of window size) and use an analytical description of AIMD to derive expressions for the steady state average window and throughput. These expressions are then validated against *ns* simulations in support of (2). A modified PPS approximation for the case of timeouts is postponed to Sec. V.

The window evolution of the TCP-Reno sessions is governed by the following system of equations (refer to Figure 6),

$$W_{i+1,j} = \begin{cases} W_{i,j} + \frac{X_i}{\gamma_{i,j}} & \text{w.p.} \quad 1 - P_{i,j} \\ W_{i,j}/2 + \frac{X_i}{\gamma_{i,j}} & \text{w.p.} \quad P_{i,j} \end{cases} \tag{3}$$

Taking the expectation of both sides of (3)

$$\frac{\overline{W_j}\,\overline{P_j}}{2} = \frac{\overline{X}}{\overline{\gamma_j}} \quad j = 1, ...M \tag{4}$$

where we used the independence approximation $E[W_{i,j}\, P_{i,j}] = \overline{W}_j\, \overline{P}_j$ and a fixed point approximation $E[W_{i,j}/\gamma_{i,j}] = \overline{W_j}/\gamma_j$.

Similarly, denoting $\widetilde{W}_j = \frac{\overline{W_j}}{\overline{\gamma_j}}$, then, by fixed point approximation to (2),

$$\overline{P_j} = \frac{\widetilde{W}_j}{\sum_{k=1}^{M} \widetilde{W}_j}. \tag{5}$$

Substituting by (5) in (4),

$$\overline{W_j} = \sqrt{2\overline{X} \sum_{k=1}^{M} \widetilde{W}_k} \quad j = 1, \ldots, M \tag{6}$$

Equation (6) is a system of M quadratic equations in the M unknowns $\overline{W}_j$ (or $\widetilde{W}_j$). Denoting,

$$\alpha_j = \frac{\overline{X}}{\overline{\gamma_j}^2} \tag{7}$$

it can be readily verified by direct substitution that

$$\widetilde{W}_j = 2\sqrt{\alpha_j} \sum_{k=1}^{M} \sqrt{\alpha_k} \tag{8}$$

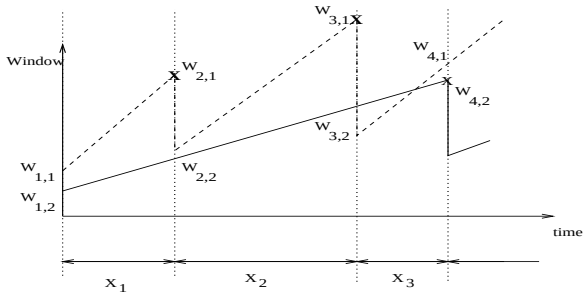

Fig. 6. A schematic showing the congestion window size for two (i.e. $M = 2$) TCP-Reno sessions with different round-trip delays sharing a bottleneck link with RED queue. Packet loss events are indicated by an X.

is an explicit solution for (6) and hence,

$$\overline{W_j} = 2\overline{X}\sum_{k=1}^{M}\frac{1}{\overline{\gamma_j}} \tag{9}$$

Remarkably, the above result implies that *the average steady-state window size is the same for all flows*; also, $\overline{W_j}$ implicitly depends on $\overline{Q}$ and $\overline{X}$ that need to be determined.

To relate the above result for $\overline{W_j}$ to $\overline{W}_{av_j}$, let $W'_{i,j}$ denote flow j window size just *after* the start of epoch i. It is straightforward to verify that

$$\overline{W}_{av_j} = \frac{\overline{W_j} + \overline{W'_j}}{2} \tag{10}$$

and from (3)

$$\overline{W'_j} = \overline{W_j}(1 - \frac{\overline{P_j}}{2}) \tag{11}$$

hence

$$\overline{W}_{av_j} = K_j\overline{W_j} \tag{12}$$

where

$$K_j = 1 - \frac{\overline{P_j}}{4} \tag{13}$$

Finally, let $\overline{r_j}$ denote flow j steady-state throughput (in packets/sec). Then,

$$\overline{r_j} = \frac{\overline{W_{av_j}}}{\overline{\gamma_j}} = K_j\frac{\overline{W_j}}{\overline{\gamma_j}} = \frac{2\overline{X}K_j}{\overline{\gamma_j}}\sum_{k=1}^{M}\frac{1}{\overline{\gamma_j}} \tag{14}$$

In case of full bottleneck link utilization,

$$\sum_{j=1}^{M}\overline{r_j} = \mu \tag{15}$$

which, by substituting from (14) and (9) yields,

$$\mu = \sum_{j=1}^{M}K_j\frac{\overline{W_j}}{\overline{\gamma_j}} \tag{16}$$

Finally, let $\overline{p}$ denote the average packet drop probability of the RED queue. Then,

$$\overline{p} = \frac{1}{\mu\overline{X}} \tag{17}$$

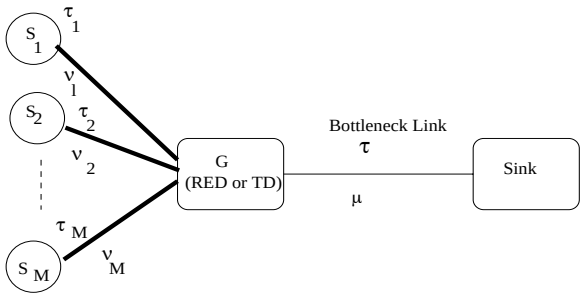

Fig. 7. Schematic of multiple TCP connections sharing a bottleneck link.

TABLE I
SINGLE RED GATEWAY PARAMETERS SETTING FOR EACH EXPERIMENT

Exp.	M	μ	τ	ν_j	τ_1	d	min_{th}	max_{th}	$1/max_p$	$\overline{Q}$ (an.)	$\overline{Q}$ (sim.)
1	16	1250	0.005	1250	0.001	1.5	10	100	16	41.4134	47.8644
2	32	1250	0.005	1250	0.001	1.2	10	100	16	77.6440	82.2968

Since the RED average packet drop probability also satisfies

$$\overline{p} = \frac{\overline{Q} - min_{th}}{max_{th} - min_{th}}\, max_p \tag{18}$$

a relationship between $\overline{Q}$ and $\overline{X}$ follows by equating (17) and (18).

B. Single Queue Validation

The analytic expressions for various parameters of interest for RED queues derived in the preceding section will be compared against results from *ns* simulations.

Figure 7 depicts the *ns* simulation model used for validating the above results. M TCP flows share a common bottleneck link with capacity μ packets/sec and (one-way) propagation delay τ. Forward TCP traffic is a one way flow between a source S_j and the sink, with reverse traffic consisting only of packet acknowledgments (ACKs) for each successfully received packet at the sink. We consider the long run (steady-state) behavior and assume that the sources always have packets to transmit (e.g. FTP of large files). The j-th flow encounters a one-way propagation delay τ_j, and is connected via a link with capacity ν_j packets/sec. to the bottleneck queue. The total round-trip time for propagation and transmission (but excluding queuing) for the j-th flow is thus

$$T_j = \frac{1}{\nu_j} + 2(\tau_j + \tau) + \frac{1}{\mu} \tag{19}$$

Table I lists the parameters used for two sets of simulation experiments with RED policy using the *ns* simulator [29] (more validation experiments are available in [30]); the link capacities μ and ν_j (packets/second), propagation delays τ and τ_j (seconds) for the topology in Fig. 7 as well as the number of competing flows used for each simulation. In each simulation, a value for the link delay τ_1 of the first flow and a factor $d > 1$ is chosen such that the remaining τ_j are specified according to $\tau_j = d * \tau_{j-1}$, i.e. the access link (from source to queue) propagation delay profile increases exponentially. The values of τ_1 and d for each simulation are listed in Table I. For each simulation we compare the analysis results to the simulations measurements of $\overline{Q}$ in Table I. All simulations use equal packet size of 1 KBytes. Figures 8-9 show the measured and the computed $\overline{r_j}$ for each competing flow, where the x-axis indicates the computed $\overline{\gamma_j}$. We choose to show the un-normalized $\overline{r_j}$ (packets/second) rather than the normalized

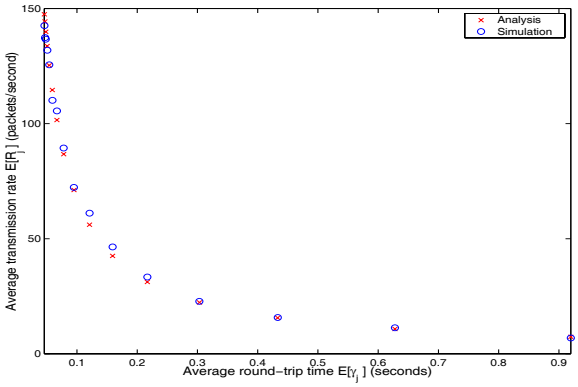

Fig. 8. Results for Exp.1.

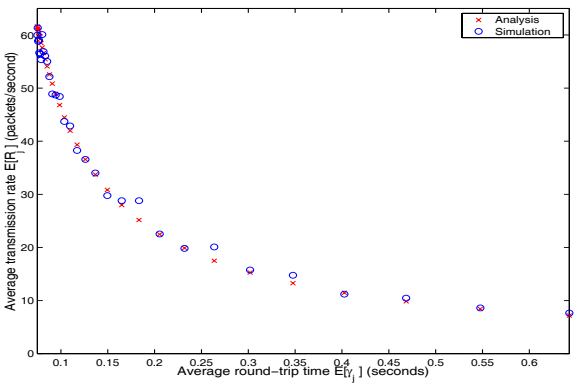

Fig. 9. Results for Exp.2.

throughput since the difference between the simulations measurement and the analysis results are indistinguishable if shown normalized to the link capacity.

C. *Fairness Properties of RED versus Tail-Drop*

TCP's inherent bias towards flows with shorter round-trip time as a result of its congestion window control mechanism is well-known - in [31], it has been shown that when multiple TCP flows with different round-trip times share a Tail Drop queue, the steady-state throughput of a flow is inversely proportional to the *square* of the average round-trip delay (see Appendix A for a simple derivation of this result). Our analysis/simulations results (specifically, (9) and (14)) of a single RED queue with multiple TCP flows and without timeouts (see comment at the end of this section) reveal a dependence of the throughput that differs from that of Tail Drop.

Consider parameter K_j defined by (13). For a population of heterogeneous flows, $\overline{P_j}$ is directly proportional to the TCP flow throughput and hence inversely proportional to the round-trip delay - K_j increases with round-trip delay. Since $0 \leq \overline{P_j} \leq 1$, $0.75 \leq K_j \leq 1$, implying mild variation. Hence, for a large number of flows, the dependence of flow throughput on K_j can be neglected [6]. Thus, the conclusions in the remainder of this section will be a lower bound to the improvement in fairness compared to Tail Drop.

With the above assumptions (i.e. neglecting the effect of K_j), the results of (9) and (14) show that *the average transmission rates of competing connections at a congested RED queue is inversely proportional to the average round-trip times and not to the square of the average round-trip times as in Tail Drop*. A fairness coefficient ρ may be defined as the ratio of the lowest to the highest average throughput among the competing flows sharing a queue (i.e. ideally, $\rho = 1$ is desirable). Let ρ_{TD} and ρ_{RED} denote the fairness coefficients of the equivalent[7] Tail Drop and RED queues. Then,

$$\rho_{RED} = \sqrt{\rho_{TD}}$$

implying that the RED queue achieves a higher fairness coefficient.

The previous definition of fairness only considers the flows with the maximum and minimum transmission rates; in the pioneering work of [32], the following index $\rho(\mathbf{r})$ was introduced to quantify fairness based on the rates of all flows:

$$\rho(\overline{\mathbf{r}}) = \frac{(\sum_{i=1}^{n} \overline{r_i})^2}{n \sum_{i=1}^{n} \overline{r_i}^2} \tag{20}$$

where $\overline{\mathbf{r}} = [\overline{r_1} \ldots \overline{r_n}]$. An ideally fair allocation ($\overline{r_i} = c \quad \forall i$) results in $\rho(.) = 1$; and a worst case allocation ($\overline{r_i} = 1, \overline{r_j} = 0 \; \forall j \neq i$) yields $\rho(.) = \frac{1}{n}$, which approaches zero for large n.

[6] In the limit as the number of flows tends to infinity, $K_j \rightarrow 1$ for all flows. Note that if the impact of K_j were to be included, it would help decrease TCP bias against larger propagation delay since it increases with the round-trip time.

[7] An *equivalent* queue is one that has identical TCP flows and the same average queue size ([11])

With this definition, the fairness index for a queue with n TCP flows yields

$$\rho = \frac{(\sum_{j=1}^{M} \overline{W_{av_j}}/\overline{\gamma_j})^2}{n \sum_{j=1}^{M} (\overline{W_{av_j}}/\overline{\gamma_j})^2} \tag{21}$$

As an illustrative example, consider TCP flows with an exponentially decaying (or increasing) delay profile with factor d. Then, assuming large n (and hence neglecting the effect of K_j for the case of RED), it follows after some simple algebraic manipulations that

$$\rho_{RED}(d) = \frac{1}{n}\left(\frac{1+d}{1+d^n}\right)\left(\frac{1-d^n}{1-d}\right) \tag{22}$$

and

$$\rho_{TD}(d) = \frac{1}{n}\left(\frac{1+d^2}{1+d^{2n}}\right)\left(\frac{1-d^{2n}}{1-d^2}\right) \ , \tag{23}$$

i.e. $\rho_{TD}(d) = \rho_{RED}(d^2)$.

An intuitive explanation for this decrease in bias (i.e. fairness coefficient closer to one) for RED is in order. TCP increases its window size by one every round-trip time - thus, lower delay links see a faster increase in their window size. In case of synchronized packet loss (Tail Drop queues), all windows are (typically) reduced at the same time, and hence the average window size is inversely proportional to the average round-trip time. But since the throughput of a flow is proportional to the window size divided by the round-trip time, the average rate of each TCP session is inversely proportional to the *square* of the average round-trip time. In case of RED, when a packet is dropped, the chances that the packet will belong to a certain connection is (on the average) proportional to that connection transmission rate- thus TCP sessions with lower delays are more likely to have their windows reduced. The analytical results show that this causes the *average window size* to be *equal* for all connections (6) and thus the throughput of a connection is only inversely proportional to the round-trip-delay. Thus while the basic RED algorithm does not achieve throughput fairness among the competing flows, it substantially reduces the bias as compared to Tail Drop.

Finally, the above fairness results do not take into account the effect of TCP timeouts. For reasons previously shown via simulations in the literature, the occurrence of timeouts will, in general, increase the unfairness in TCP against longer delay links.

IV. Timeout formula for a TCP flow with random packet loss

In this section, we first derive an expression for the probability that a packet loss event will be detected by a timeout (TO) (i.e. not a duplicate ACK detection (DA) event). Next, we compare our result and those in the literature against *ns* simulations.

A. Analysis

Consider an instant during a TCP-Reno session at which the window size is in the congestion avoidance phase. Let w denote the *current* congestion window size, in packets, during which at least one packet loss takes place; each packet is assumed lost independently with probability p. Let the current round be labelled as *round* i for reference (see Figure 10). Let $P(w,k)$ denote the probability, conditioned on at least one packet loss during the current window round, that packet k will be the first packet lost during the round. Then,

$$P(w,k) = \frac{p(1-p)^{k-1}}{1-(1-p)^w} \tag{24}$$

The $k-1$ packets transmitted before the first loss in the current round will cause the reception of $k-1$ ACKs in the *next* round ($i+1$) and hence the release of an equal number of packets.

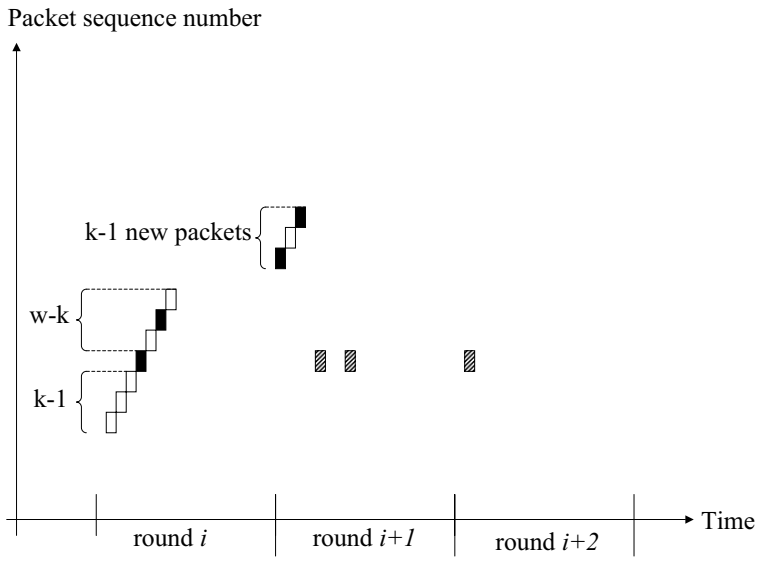

Fig. 10. Analysis of the probability of a TO vs. TD. Successful packets (i.e. not dropped), dropped packets, and ACK packets for packet k are represented by white-filled, completely shaded and partly shaded rectangles respectively. In this example, $w = 7$, $k = 4$, $N_1(w-k) = 2$ and $N_2(k-1) = 1$.

Let $N_2 = N(k-1)$ denote the number of duplicate ACKs received in round $i+2$, due to the successful transmission of N_2 out of the $k-1$ packets in round $i+1$. Similarly, consider the $w-k$ packets transmitted after the first loss during round i (the current round), and let $N_1 = N(w-k)$ denote the number of packets successfully transmitted out of the $w-k$ packets. Notice that, unlike the N_2 packets, the N_1 packets are transmitted after the first loss packet in the current round. Hence, N_1 duplicate ACKs will be generated immediately in round $i+1$.

Let $D(w,k)$ denote the probability that a DA detection takes place in round $i+1$ or $i+2$. Then,

$$D(w,k) = Pr(N_1 + N_2 \geq 3) = 1 - \sum_{m=0}^{2} \left(\begin{array}{c} w-1 \\ m \end{array} \right) (1-p)^m p^{w-m-1} \tag{25}$$

Notice that the above expression is independent of k. This is expected, and can be stated simply as the probability of at least 3 out of $w-1 = (k-1)+(w-k)$ successful packet transmissions.

Let $Q(w)$ denote the probability of a TO detection in rounds $i+1$ or $i+2$. Then

$$Q(w) = \sum_{k=1}^{w} P(w,k)(1 - D(w,k)) = \sum_{m=0}^{2} \left(\begin{array}{c} w-1 \\ m \end{array} \right) \tag{26}$$

For $p, pw << 1$, retaining only the most significant terms yields

$$Q(w) \approx \frac{(w-1)(w-2)}{2} p^{(w-3)} \tag{27}$$

Following a DA detection, the window is halved and TCP attempts to return to its normal window increase mode in congestion avoidance. However, one or more losses may take place in the immediate following round (i.e. before the window is increased). In this paper, consecutive congestion window rounds in which packet loss is detected by DA without any separating additive window increase will be treated as one loss window, that ultimately terminates via either a DA or a TO. Let $Z(w)$ denote the probability that a TO will take place (either immediately or following a sequence of consecutive DA's). Then,

$$Z(w) \;\; = \;\; Q(w) + [1 - Q(w)][Q(\frac{w}{2}) + [1 - Q(\frac{w}{2})][Q(\frac{w}{4}) + \ldots]$$

$$
\begin{aligned}
&= Q(w) + Q(\frac{w}{2})[1-Q(w)] + Q(\frac{w}{4})[1-Q(w)][1-Q(\frac{w}{2})] + \dots \\
&= Q(w) + \sum_{j=1}^{\lfloor log\frac{w}{3} \rfloor} \prod_{m=1}^{j} Q(\frac{w}{2^j})[1-Q(\frac{w}{2^{j-1}})] \qquad (28)
\end{aligned}
$$

where the logarithm is base 2.

Finally, recalling that W_i denotes the window size at the end of the i^{th} epoch, applying a fixed point approximation to (28) yields

$$E[Z(W_i)] = Z(\overline{W}) \qquad (29)$$

where $Z(\overline{W})$ is given by (28).

B. Timeout formula validation and comparison with previous results

In this section, previous results for $Z(w)$ are listed and compared with the results of a representative simulation using *ns* consisting of multiple TCP flows sharing a bottleneck RED queue.

In [14], the following expression for $Z(\overline{W})$ was derived

$$Z(\overline{W}) = \min\left(1, \frac{(1-(1-p)^3)(1+(1-p)^3(1-(1-p)^{\overline{W}-3}))}{1-(1-p)^{\overline{W}}}\right), \qquad (30)$$

assuming that, once a packet is lost, all remaining packets in the same congestion window are also lost (which is probably reasonable for a Tail Drop queue). In [19], the authors argue that this is still a good result when applied to an AQM like RED which drops packets independently. We show here that this is not true.

In [15], the following expression for $Z(w)$ was derived

$$Z(w) = \sum_{m=0}^{2} \left(\begin{array}{c} w-1 \\ m \end{array} \right) (1-p)^m p^{w-m-1} \qquad (31)$$

which is the same as our expression for $Q(w)$ in the previous section (26). However, the authors do not analyze the case where multiple consecutive DA's ultimately cause a TO. Also, the expression for $E[Z(w)]$ was not derived in [15], since the analysis relies on the knowledge of the congestion window probability distribution function. Nevertheless, we show here that applying a fixed point approximation to (31), yielding

$$Z(\overline{W}) = \sum_{m=0}^{2} \left(\begin{array}{c} \overline{W}-1 \\ m \end{array} \right) (1-p)^m p^{\overline{W}-m-1} \qquad (32)$$

would result in an under-estimation of the timeout probability.

In order to compare between the accuracy of our result (29) and the previous results (30) and (32) when applied to a RED queue, we construct the following simulation using *ns*. A single congested RED queue is considered with a number of identical TCP flows ranging from 5 to 40. The RED parameters are set as follows: $maxp = 0.1$, $maxth = 80$, $minth = 20$ and ECN disabled. The link capacity is 1 Mbps and two-way propagation delay (no including queuing) is 20 ms. The packet size is 1 KB. The buffer size is selected large enough to avoid buffer overflow. The trace functionality within *ns* was enhanced so as to record, for one of the flows, the time instant, congestion window size and the reason (TO or DA) for each window decrease. Post-processing scripts were used to compute the average window size just before a packet loss detection (i.e. $\overline{W}$), the average packet drop through the queue (p), the total number of packet loss detection events that are detected by TO and those detected by DA, and hence the

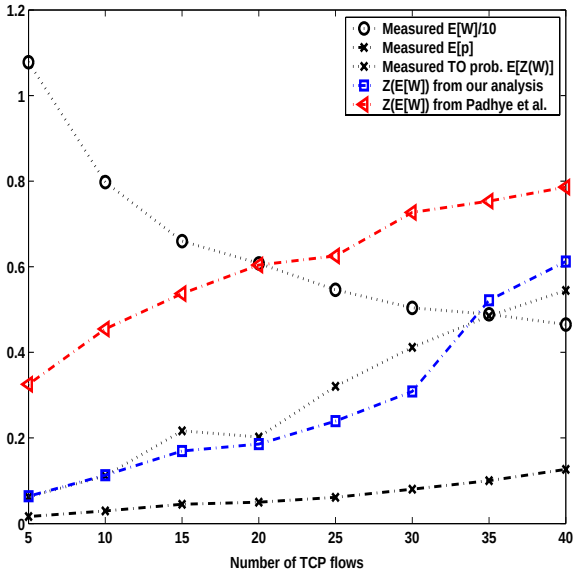

Fig. 11. Comparison between different *timeout formulae* against *ns* simulations for TCP-NewReno.

measured $E[Z(W)]$. Finally, the measured p and $\overline{W}$ from the simulations were substituted in (29), (30) and (32) to compute $Z(\overline{W})$. The results are shown in Figure 11.

Figure 11 does not show the results from (32) since the values produced are practically 0,i.e., (32) excessively under-estimates the probability of a timeout. Figure 11 shows that our result is much more accurate than that of [14] (as expected). The improvement accuracy is up to an order of magnitude for small number of competing flows (and hence small p). The reason is that, for small p, the assumption of conditional burst loss used in [14] becomes very unrealistic. As p increases, the RED queue operation approaches that of Tail Drop, since p approaches $maxp$ (and hence all arriving packets are dropped with higher probability, resulting in synchronous operation of the flows and thus burst losses). However, even for high p, our formula is more accurate than that of [14].

V. Network Analysis

Since a RED queue is a special case of a DiffRED queue with $\beta^{(c)} = 1 \, \forall \, c$, we present an analysis for the case of DiffRED queues. In the first subsection, we consider a network of TCP and UDP flows and we model the additive increase/multiplicative decrease dynamics of TCP only. Next, we extend the model results to capture the TCP timeout behavior.

A. Modeling without timeouts

Let $minth_v$, $maxth_v$ and ${maxp_v}^{(c)}$ denote the DiffRED parameters of the v^{th} queue. Let ${\beta_v}^{(c)}$ denote the priority coefficient of class c, where ${\beta_v}^{(1)} = 1$. Let $\overline{p_v}$ denote the average (steady state) packet drop probability in the v^{th} queue. Let $\overline{p_v}^{(c)}$ denote the average packet drop probability for class c at the v^{th} queue. Then for the case of the "gentle" variant,

$$\overline{p_v}^{(c)} = \begin{cases} 0 & ; 0 \leq \overline{q_v} < minth_v \\ \frac{\overline{q_v} - minth_v}{maxth_v - minth_v} {maxp_v}^{(c)} & ; minth_v \leq \overline{q_v} < maxth_v \\ {maxp_v}^{(c)} + \frac{\overline{q_v} - max_{th}}{max_{th}}(1 - {maxp_v}^{(c)}) & ; maxth_v \leq \overline{q_v} < 2maxth_v \\ 1 & ; 2maxth_v \leq \overline{q_v} \end{cases} \tag{33}$$

while for the case without the "gentle" variant,

$$\overline{p_v}^{(c)} = \begin{cases} 0 & ; 0 \leq \overline{q_v} < minth_v \\ \frac{\overline{q_v} - minth_v}{maxth_v - minth_v} {maxp_v}^{(c)} & ; minth_v \leq \overline{q_v} < maxth_v \\ 1 & ; maxth_v \leq \overline{q_v} \end{cases} \tag{34}$$

According to the description of the DiffRED dropping algorithm (Figure 2)

$$maxp_v{}^{(c)} = \beta^{(c)}\, maxp_v{}^{(1)} \tag{35}$$

where (35) assumes that all the DiffRED queues in a network use the same value of the class priority coefficients (i.e. $\beta^{(c)}$ is independent of v) [8].

Let $\beta(j)$ and $\overline{p_v(j)}$ denote the priority coefficient and average drop probability at the v^{th} queue for the class that flow j belongs to. For the case of the "gentle" variant, it follows from (33) and (35) that

$$\overline{p_v(j)} = \begin{cases} 0 & ;\ 0 \le \overline{q_v} < minth_v \\ \beta(j)\overline{p_v}^{(1)} & ;\ minth_v \le \overline{q_v} < maxth_v \\ \beta(j)maxp^{(1)} + \frac{\overline{p_v}^{(1)} - maxp^{(1)}}{1 - maxp^{(1)}}(1 - \beta(j)maxp^{(1)}) & ;\ maxth_v \le \overline{q_v} < 2maxth_v \\ 1 & ; 2maxth_v \le \overline{q_v} \end{cases} \tag{36}$$

while for the case without the "gentle" variant

$$\overline{p_v(j)} = \begin{cases} 0 & ;\ 0 \le \overline{q_v} < minth_v \\ \beta(j)\overline{p_v}^{(1)} & ;\ minth_v \le \overline{q_v} < maxth_v \\ 1 & ; maxth_v \le \overline{q_v} \end{cases} \tag{37}$$

Let $\overline{r_j}$ denote the steady state packet transmission rate of flow j. Let the *generator* matrix G denote the fraction of the j^{th} flow transmission rate at the *input* of the v^{th} queue. Then the elements of G are

$$g(j,v) = \prod_{k=1}^{V} h(o(j,v), o(j,k)) \tag{38}$$

where

$$h(o(j,v), o(j,k)) = \begin{cases} 0 & ; o(j,v) = 0 \\ (1 - \overline{p_k}) & ; o(j,v) > o(j,k) \\ 1 & ; o(j,v) \le o(j,k) \end{cases} \tag{39}$$

Let $\overline{B_v}$ denote the steady state probability that the dropped packet belongs to the v^{th} queue. Applying A4,

$$\overline{B_v} = \frac{\sum_{j=1}^{M} g(j,v)\overline{r_j}\overline{p_v(j)}}{\sum_{u=1}^{V}\sum_{k=1}^{M} g(k,u)\overline{r_k}\overline{p_u(k)}} \tag{40}$$

Let $\overline{P_{j|v}}$ denote the probability that, conditioned on a packet loss from the v^{th} queue, that packet loss belongs to the j^{th} flow, then, using A4,

$$\overline{P_{j|v}} = \frac{g(j,v)r_j\overline{p_v(j)}}{\sum_{k=1}^{M} g(k,v)r_k\overline{p_v(j)}} \tag{41}$$

Let $\overline{P_j}$ denote the probability that a packet loss belongs to the j^{th} flow. Then,

$$\overline{P_j} = \sum_{v=1}^{V} \overline{P_{j|v}}\,\overline{B_v} \tag{42}$$

which by substituting from (40) and (41) yields,

$$\overline{P_j} = \frac{\overline{r_j}\sum_{v=1}^{V}\overline{p_v(j)}\, g(j,v)}{\sum_{u=1}^{V}\sum_{k=1}^{M} g(k,u)\overline{r_k}\overline{p_u(k)}} \tag{43}$$

[8] It is possible to extend the analysis by assuming that each queue has a different set of priority coefficients by defining ${\beta_v}^{(c)}$ for each queue. This may be specifically valuable when modeling the interaction between DiffServ enabled and non-DiffServ queues. However, for initial model simplicity, we postpone this to future extensions.

where the above is simply the ratio of the sum of the packet drop rates of flow j at all the queues to the sum of the packet drop rates of all flows in the network.

Now, we can consider one of the TCP flows in the network and consider its window evolution. Let Y_i denote the time at which the i^{th} random packet loss (caused by any of the V queues and causing a packet loss from one of the $M+N$ flows) event takes place, and $X_i = Y_i - Y_{i-1}$ be the i-th inter-loss duration (*epoch*).

Let the j^{th} TCP session window size (in packets) at the beginning and end of the i^{th} epoch be denoted by $W_{i,j}$ and $W_{i+1,j}$ respectively. Let $q_{i,v}$ denote the queue size of queue v at the end of epoch i; then the net round-trip propagation and queuing delay for connection j at the end of the i^{th} epoch is $\gamma_{i,j} = T_j + \sum_{v=1}^{V} \left(\frac{q_{i,v}}{\mu_v} sgn(o(j,v)) \right)$ where

$$sgn(o(j,v)) = \begin{cases} 1 & ; o(j,v) > 0 \\ 0 & ; o(j,v) = 0 \end{cases}$$

Following the same steps as before, the window evolution of the TCP sessions is governed by the system of equations (4), where $E[\gamma_{i,j}] = \overline{\gamma_j} = T_j + \sum_{v=1}^{V} \left(\frac{\overline{q_v}}{\mu_v} sgn(o(j,v)) \right)$. This system of equations captures only the TCP additive increase/multiplicative decrease dynamics but not the impact of timeout - this is done in the next section.

From Sec. III-A,

$$\overline{X} = \frac{1}{\sum_{u=1}^{V} \sum_{k=1}^{M+N} g(k,u) \overline{r_k} \overline{p_u(k)}} \tag{44}$$

and

$$\overline{r_j} = K_j \frac{\overline{W_j}}{\overline{\gamma_j}} \, , \, j = 1, 2 \ldots M \tag{45}$$

where K_j is given by (12).

Substituting by $\overline{X}$ from (44), $\overline{P_j}$ from (43) and $\overline{r_j}$ from (45) in (4),

$$\overline{W_j} = \sqrt{\frac{2}{K_j \sum_{v=1}^{V} g(j,v) \overline{p_v(j)}}} \tag{46}$$

Finally, for congested links (A1),

$$\mu_v = \sum_{j=1}^{M+N} g(j,v) \overline{r_j} (1 - \overline{p_v(j)}) \quad ; \overline{q_v} > minth_v \tag{47}$$

while for uncongested (underutilized) links,

$$\mu_v < \sum_{j=1}^{M+N} g(j,v) \overline{r_j} (1 - \overline{p_v(j)}) \quad ; \overline{q_v} = 0 \tag{48}$$

The set of $M+V$ equations (46)-(48) in the $M+V$ unknowns ($W_j \, , \, j = 1, 2 \ldots M$ and $\overline{p_v}^{(1)} \, , \, v = 1, 2 \ldots V$) are then solved numerically as explained in Section IV.

B. Modeling TCP timeout mechanism

In this section, we show how the model can be extended to incorporate the effect of timeouts. Since it is well known that the slow start behavior has a negligible effect on the TCP steady-state throughput, we neglect slow start and assume that the window size after a timeout is set to 1 and that TCP continues in the congestion avoidance phase after some idle duration (discussed below). For simplicity, only single timeouts are considered since the difference in performance between a single timeout and multiple timeouts is insignificant (both result in very

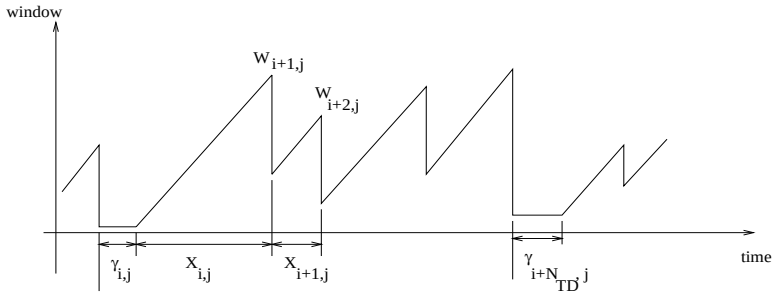

Fig. 12. Approximate timeout behavior. After timeout, slow-start is neglected, and TCP is assumed to resume in congestion avoidance mode with an initial window size value of one packet.

low throughput). Thus, our objective is to model the approximate timeout behavior depicted in Fig.12 , where a single timeout causes the window size to be reduced to one, and TCP refrains from transmission (i.e. is idle) for a period of one round-trip time ($\gamma_{i,j}$).

Let $Z(W_{i,j})$ denote the probability that, given a packet loss *belonging to flow* j, that loss will result in a timeout. Then, following the same steps as before (for $j = 1, 2 \ldots M$),

$$W_{i+1,j} = \begin{cases} W_{i,j} + \frac{X_i}{\gamma_{i,j}} & \text{w.p.} \quad 1 - P_{i,j} \\ W_{i,j}/2 + \frac{X_i}{\gamma_{i,j}} & \text{w.p.} \quad P_{i,j}\,(1 - Z(W_{i,j})) \\ 1 + \frac{(X_i - \gamma_{i,j})^+}{\gamma_{i,j}} & \text{w.p.} \quad P_{i,j}\,Z(W_{i,j}) \end{cases} \tag{49}$$

where

$$(x)^+ = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} \tag{50}$$

Taking the expectations of both sides and simplifying,

$$\frac{\overline{W_j P_j}}{2}(1 + Z(\overline{W_j})) = \frac{\overline{X}}{\overline{\gamma_j}} \tag{51}$$

Let $\overline{b_j}$ denote the steady-state proportion of time that TCP flow j is busy ($\overline{b_j} = 1\,, j > M$ for the non-responsive flows). Following the same derivation steps as before,

$$\overline{P_j} = \frac{\overline{r_j}\,\overline{b_j} \sum_{v=1}^{V} \overline{p_v(j)}\, g(j,v)}{\sum_{u=1}^{V} \sum_{k=1}^{M} g(k,u)\overline{r_k}\overline{p_u(k)}\,\overline{b_k}} \tag{52}$$

and

$$\overline{X} = \frac{1}{\sum_{u=1}^{V} \sum_{k=1}^{M+N} g(k,u)\overline{r_k}\overline{p_u(k)}\,\overline{b_k}} \tag{53}$$

An expression for $Z(\overline{W_j})$ was derived (and validated) in Sec. IV (equation 29). An expression for $\overline{b_j}$ is derived as follows.

Let $X_{i,j}$ denote the inter-loss times between packet losses *of TCP flow* j (as opposed to X_i which denotes the inter-loss times in the network) and consider the window size evolution of flow j in between two consecutive Duplicate ACK (DA) events (see Fig. 12) Then,

$$W_{i+1,j} = W_{i,j}/2 + \frac{X_{i,j}}{\gamma_{i,j}} \tag{54}$$

and hence

$$\overline{X_j} = \frac{\overline{W_j}\,\gamma_j}{2} \tag{55}$$

Let $N_{DA}(\overline{W_j})$ denote the average (from a fixed point approximation argument) expected number of consecutive DA's. Then $N_{DA}(\overline{W_j})$ follows a geometric distribution with parameter $Z(\overline{W_j})$,

$$N_{DA}(\overline{W_j}) = \frac{1 - Z(\overline{W_j})}{Z(\overline{W_j})} \tag{56}$$

and finally,

$$\overline{b_j} = 1 - \frac{\gamma_j}{\gamma_j + \overline{X_j}\, N_{DA}(\overline{W_j})}\, , j = 1, 2 \ldots M \tag{57}$$

Thus, a set of M equations in the $M+V$ unknowns is obtained by substituting by $\overline{b_j}$ from (57) and P_j from (52) in (51). The remaining V equations correspond to (47)-(48) with modification to include the timeout behavior yielding,

$$\mu_v = \sum_{j=1}^{M+N} g(j,v)\overline{r_j}(1 - \overline{p_v(j)})\overline{b_j} \quad ; \overline{q_v} < minth_v \tag{58}$$

for fully utilized links and

$$\mu_v < \sum_{j=1}^{M+N} g(j,v)\overline{r_j}(1 - \overline{p_v(j)})\overline{b_j} \quad ; \overline{q_v} = 0 \tag{59}$$

for underutilized links.

A solution for the $M+V$ unknowns is obtained by solving the $M+V$ equations numerically as explained in the following section.

VI. Network Model Results

In order to validate the analysis presented in the previous sections, we present a representative number of comparisons between the numerical results obtained from the solution of the $M+V$ non-linear equations and the simulation results from the *ns-2* simulations.

The simulations reported here use the "gentle_" option of RED in the *ns-2* simulator set to "true" (i.e. it uses the gentle variant) and the "setbit_" option set to false (i.e. packet drops instead of marking). Other experiments without the "gentle" option provide results with similar accuracy (not reported here due to space constraints).

We first describe the technique used for solving the non-linear equations. We then summarize the modifications to the RED algorithm in *ns-2* to simulate the DiffRED behavior. And finally, we present the validation experiments.

A. The network solver

In the analysis section, the $V+M$ unknowns ($W_j\,,\ j = 1, 2 \ldots M$ and $\overline{p_v}^{(1)}\,,\ v = 1, 2 \ldots V$) are related by a set of $V+M$ non-linear equations. In general, a set of non-linear equations can be solved using a suitable numerical technique. Two main problems encountered in solving non-linear equations are; (i) the uniqueness of the solution and (ii) the choice of an appropriate initial condition that guarantees the convergence towards a true solution. In this section, we describe the algorithm used to achieve both objectives.

Our algorithm is based on the observation that there exists a unique solution for (46)-(48) that satisfies $0 \leq \overline{q_v} \leq maxth_v$ and $\overline{W_j} > 0$. Thus, the network solver is composed of the following two steps (or modules):

Step I

The set of equations (46)-(48) is solved using any iterative numerical technique. We used a *modified* globally convergent Newton-Raphson technique [33] that operates as follows: (1) It chooses initial conditions randomly within the solution space; (2) Performs Newton-Raphson

technique while checking for convergence; (3) If the algorithm converges to a valid solution, the program terminates; else, the program repeats from step (1) again.

Note that the resulting solution does not take into account timeouts or non-responsive flows - hence, step II.

Step II

This step of our algorithm uses the solution provided in step I as initial conditions - it also uses a globally convergent Newton-Raphson technique applied this time to the extended modeling equations describing the timeout behavior in (51)-(52), (58)-(59) (and, if applicable, the UDP behavior).

Thus, the network solver is composed of two modules, each of which uses an iterative numerical method; the first module solves the simplified model and thus provides an approximate solution for the network which is then fed into the next module that refines the network solution providing a more accurate one. This technique has proved to converge to the correct solution in all experiments we tried, a representative of which (a total of 160 experiments) are reported in the following sections.

B. Modifying ns-2 *to model DiffRED*

The following modifications to *ns-2* were incorporated. In the packet common header, we added a *priority* field that contains an integer specifying the priority class of the packet. Similarly, an additional parameter for the TCP object specifying its class of service was added; when a TCP session is established, this parameter is set to the selected class of service for that session. The TCP mechanism is also modified to label each packet by the TCP session priority (by copying the TCP session class of service to the packet header of the packet being transmitted). Finally, the RED algorithm is modified so as to first check for the packet class (from the packet header) and compute the packet drop probability according to the mechanism outlined in Sections II.B (for the case without the "gentle" option) or II.C (for the "gentle" variant).

C. Experimental Topology

We use the topology of Figure 5. It consists of two RED/DiffRED queues Q_1 and Q_2. There are a total of five sets of flows going through the queues and each flow set is composed of four identical flows. In the figure, S denotes the source (origin) while D denotes the destination (sink) for those flows. Three sets of flows ($S1$, $S2$ and $S3$) arrive at Q_1, out of which only one ($S2$) traverses through Q_2. Two other sets of flows arrive at Q_2. The only two bottleneck links in the topology are those of Q_1 and Q_2, where the link speeds are μ_1 and μ_2 (packets/sec) and delays τ_1 and τ_2 (seconds).

In the experiments, we vary the priority class of $S2$, allowing it to have a higher priority in some of the experiments to compensate it for the higher loss rate it suffers since it has to traverse both queues.

Each experiment is composed of five simulation runs, at the end of each we compute the parameters of interest; average queue size for each queue, average window size for each TCP flow and the average goodput (number of packets received at the destination per second). The duration of each run is long enough to guarantee the operation of the network in the steady state for a long duration. We then compute the average, among the simulation runs, of each of these parameters and compare with the network solver results. In the figures, we plot one point (x, y) for each experiment, where x represents the simulation result and y represents the corresponding result obtained from applying our analytical network solver. Hence, ideally, all points should lie on a line with a slope of 45 degrees.

TABLE II

PARAMETERS SETTINGS FOR THE EXPERIMENTAL NETWORK TOPOLOGY OF FIG.5 AND FIG.16. LINK SPEEDS ARE IN MBPS, $minth$ AND $maxth$ IN PACKETS AND PACKET SIZE IN KBYTES. $\tau_1 = \tau_2 = 10$ MS.

μ_1	μ_2	$minth_1$	$maxth_1$	$maxp_1$	$minth_2$	$maxth_2$	$maxp_2$	pkt.
32	32	20	80	0.1	20	80	0.1	1
32	16	20	80	0.1	20	80	0.1	1
16	32	20	80	0.1	20	80	0.1	1
32	32	10	150	0.05	20	80	0.1	1
32	32	20	80	0.1	20	80	0.1	1.5

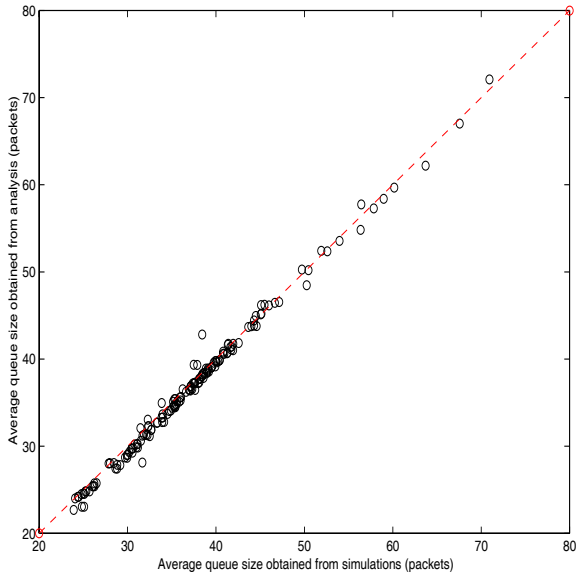

Fig. 13. Average queue size obtained from simulations and analysis for the experiments with TCP traffic only.

D. Experiments with TCP only

A total of eighty experiments have been performed with various parameters. The parameters of five of these experiments are shown in Table I. Three additional similar sets are performed but by reducing the bottleneck link speeds by half for each set (i.e. the first experiment in the last set would configure the bottleneck links at 4Mbps) thus forming twenty experiments. These twenty experiments are repeated four times, with the priority coefficient of the higher priority flow $S2$ varying from 0.1 to 1.0 (in steps of 0.3). The results are shown in Figures 13-15.

Note that this also validates the analysis for the RED case, since setting the priority coefficient to 1.0 corresponds to RED operation.

In Figure 13, each experiment results in two points, one for each queue. Hence, the figure contains a total of 160 simulation/analysis comparison points. On the other hand, in Figure 14 and Figure 15, each experiment results in three comparison points. The reason is that TCP flows $S1$ and $S3$ (on one hand) and $S4$ and $S5$ have identical goodput and congestion window size values. Thus, each experiment renders three window sizes (and three goodput values), one for $S1$ and $S3$, one for $S2$ and a third for $S4$ and $S5$ for a total of 240 comparison points for each of these figures.

E. Experiments with TCP and UDP

To validate the model results for mixed TCP and UDP traffic, we modify the topology of Figure 5 to Figure 16 by adding two sets of UDP (constant bit rate) flows, one that originates at

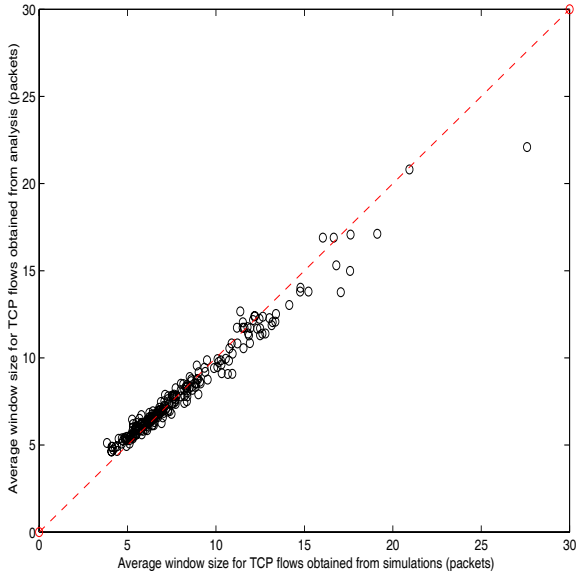

Fig. 14. average congestion window size from simulations and analysis for the experiments with TCP traffic only.

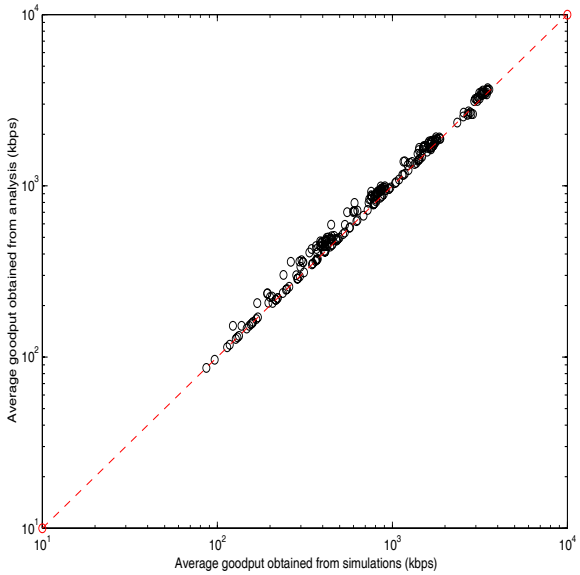

Fig. 15. The average goodput obtained from simulations and analysis for the experiments with TCP traffic only.

S6 and passes through both queues before exiting the network while the other originates at $S7$ and passes only through $Q2$. The transmission rates of the UDP flows are set such that the total transmission rates of $S1$ and $S2$ equals 10% of the link capacities μ_1 and μ_2. The results are shown in Figures 17-19.

Just like Figures 13-14, Figures 17-18 contains a total of 160 and 240 simulation/analysis comparison points (respectively). However, unlike Figure 15 which contains 240 comparison points Figure 19 contains an additional 160 comparison points accounting for the 2 UDP flows in each of the 80 experiments.

Finally, in Figure 20, we show the results of varying the priority coefficient for one of the experiments (the first experiment in Table I) separately. The figure shows how the priority coefficient is effective in boosting $S2$ goodput from its low value when $\beta(2) = 1.0$ to a much higher value (almost equal to the other TCP flows rate) when $\beta(2) = 0.1$.

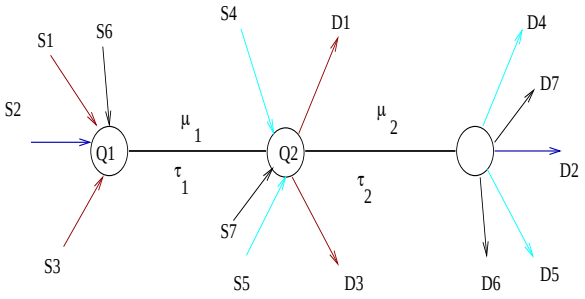

Fig. 16. Experimental network topology with two RED queues.

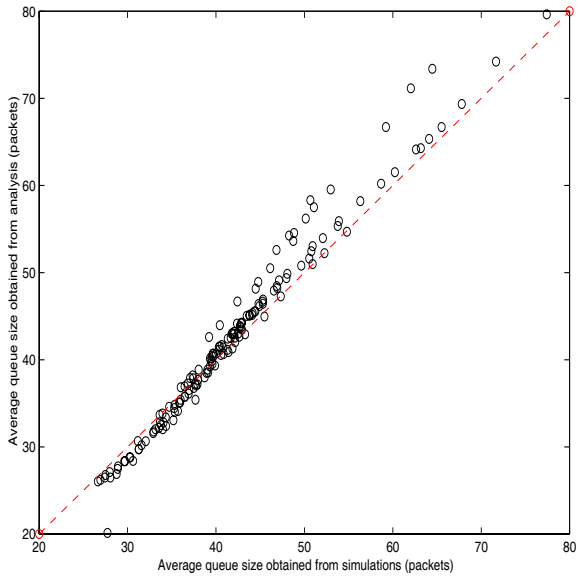

Fig. 17. Average queue size obtained from simulations and analysis for mixed TCP and UDP traffic.

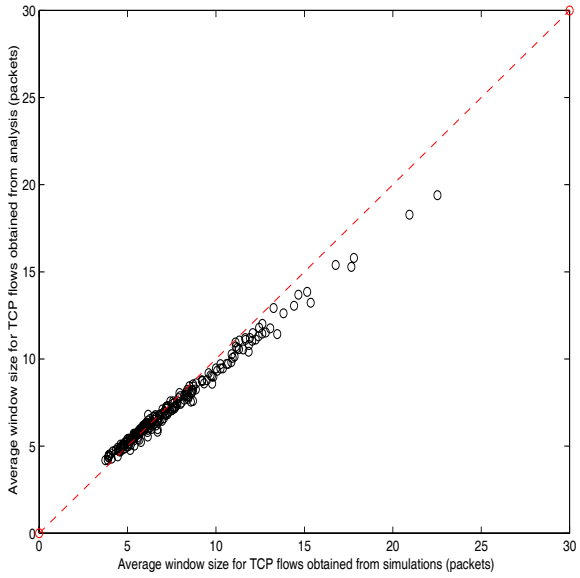

Fig. 18. Average congestion window size obtained from simulations and analysis for mixed TCP and UDP traffic.

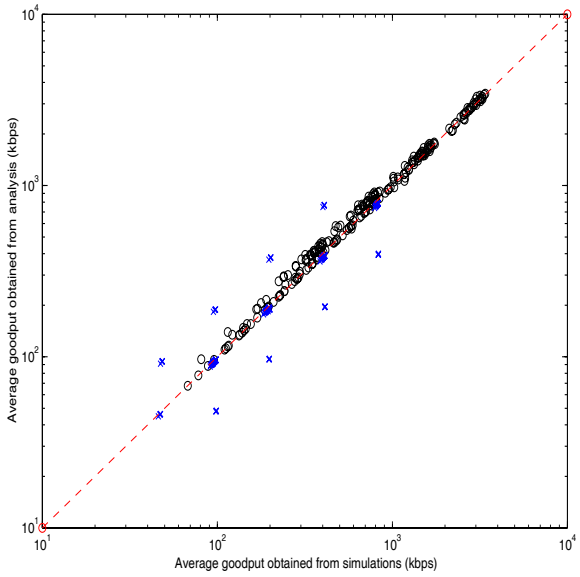

Fig. 19. Average goodput obtained from simulations and analysis for mixed TCP and UDP traffic.

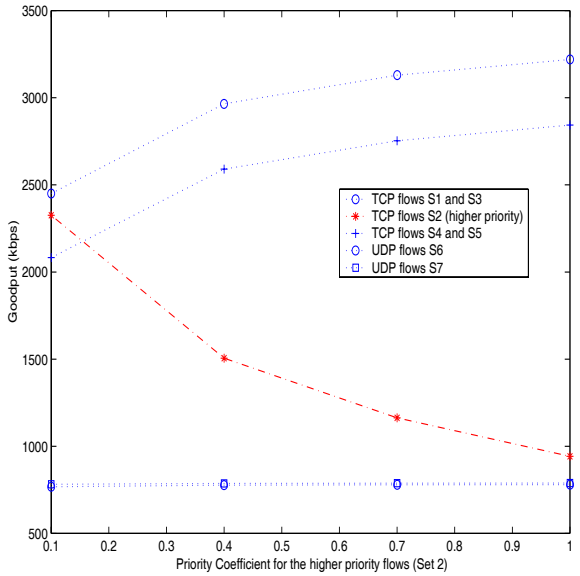

Fig. 20. Effect of varying the priority coefficient of $S2$ on its goodput. The experiment parameters are shown in the first entry of Table I.

VII. Conclusion

In this paper, an analytical model for a network of RED/DiffRED queues with multiple competing TCP and UDP flows was presented. Unlike previous work, our analysis is specifically targeted towards AQM schemes which are characterized by random packet drop (unlike Tail Drop queues which drop packets in bursts). Our main contributions are (i) An accurate timeout formula that is orders of magnitude more accurate than the best-known analytical formula, (ii) Closed form expressions for the relative fairness of RED and Tail-Drop towards heterogeneous TCP flows (iii) Analysis of RED queues in a traditional as well as differentiated services network. Our analysis has relied on a set of approximations to the timeout dynamics as well as to the loss rate process of each flow in the network. The analytical results were validated against *ns* simulations. These show that the results are accurate within a mean margin of error of 2% for the average TCP throughput, 5% for the average queue size and 4% for the average window size attributable to the approximations introduced in Section II. The model proposed should

be applicable to a variety of AQM schemes that rely on randomized (instead of deterministic) packet drops.

A number of avenues for future research remain. First, the model can be extended to the analysis of short-lived flows and the effects of limiting advertised congestion window. Also model extensions to other versions of TCP (e.g. Tahoe, SACK,..etc.) and to the case of ECN [34] may be considered. Finally, as noted in (8), especially with the expected coexistence of DiffServ networks with best effort networks, an extension of this model that captures the interaction between traditional RED queues and DiffRED would be valuable.

APPENDIX

In Sec. III-C, we have stated that the average window size of a TCP flow passing through a Tail Drop queue is inversely proportional to its round-trip delay.

Proof: We show below that, for a Tail-Drop queue, the steady state window size for any flow is inverse proportional to the round-trip time.

In Tail Drop queues, buffer overflow causes (typically) at least one packet loss from each flow passing through the queue ([35], [36]). Using the same notations as Sections II and III,

$$W_{i+1,j} = \frac{W_{i,j}}{2} + \frac{X_i}{\gamma_{i,j}} \tag{60}$$

Taking expectation of both sides of (60), and denoting

$$\overline{W_j} = 2\frac{\overline{X}}{\gamma_j} \tag{61}$$

proving the claim.

REFERENCES

[1] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP)-Version 1 functional specification. RFC 2205, IETF, September 1997.

[2] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A new resource ReSerVation protocol. *IEEE Network*, pages 8–18, September 1993.

[3] J. Wroclawski. Specification of controlled-load network element service. RFC 2211, IETF, September 1997.

[4] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. RFC 2212, IETF, September 1997.

[5] J. Wroclawski. The Use of RSVP with IETF Integrated Services. RFC Draft, IETF, September 1997.

[6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC 2475, IETF, December 1998.

[7] D. Clark and J. Wroclaski. An approach to service allocation in the Internet. RFC Draft, IETF, July 1997.

[8] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the Internet. RFC Draft, IETF, November 1997.

[9] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the differentiated services field (DS filed) in the IPv4 and IPv6 headers. RFC 2474, IETF, December 1997.

[10] B. Braden et al. Recommendations on queue management and congestion avoidance in the Internet. RFC 2309, IETF, April 1998.

[11] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[12] D. Clark and W. Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4):362–73, 1998.

[13] Technical Specification from Cisco. Distributed weighted random early detection. http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/wred.pdf.

[14] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Reno performance: A simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, 8(2):133–145, 2000.

[15] A. Kumar. Comparative performance analysis of versions of TCP in a local network with a lossy link. *IEEE/ACM Transactions on Networking*, 6(4):485–498, 1998.

[16] T. Ott, J. Kemperman, and M. Mathis. The stationary behavior of ideal TCP congestion avoidance. *preprint*, 1996. ftp://ftp.telecordia.com/pub/tjo/TCPWindow.ps.

[17] M. Mathis, J. Semke, J. Madhavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communications Review*, 27(3):67–82, 1997.

[18] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, 1999.

[19] T. Bu and D. Towsley. Fixed point approximations for TCP behavior in an AQM network. Technical Report 2000-43, University of Massachusetts, 2000.
[20] R. J. Gibbens, S. K. Sargood, C. Van Eijl, F. P. Kelly, H. Azmoodeh, R. N. Macfadyen, and N. W. Macfadyen. Fixed-point models for the end-to-end performance analysis of ip networks. In *Proceedings of 13th ITC Specialist Seminar: IP Traffic Measurement, Modeling and Management*, September 2000.
[21] V. Firoiu, I. Yeom, and X. Zhang. A framework for practical performance evaluation and traffic engineering in ip networks. In *Proceedings of IEEE ICT 2001*, June 2001.
[22] S. Floyd. Recommendation on using the gentle_ variant of RED. *URL http://www.aciri.org/floyd/red/gentle.html*, March 2000.
[23] W. Stevens. *TCP/IP Illustrated*, volume 1. Adison Wesley, 1994.
[24] S. Floyd and T. Henderson. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 2582, IETF, April 1999.
[25] F. P. Kelly. Blocking probabilities in large circuit-switched networks. *Advances in Applied Probability*, 18:473–505, 1986.
[26] T. V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking*, 5(3):336–50, 1997.
[27] V. Misra, W-B. Gong, and D. Towsley. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *Proceedings of SIGCOMM'2000*, 2000.
[28] A. Abouzeid and S. Roy. Analytic understanding of RED gateways with multiple competing TCP flows. In *Proceedings of GLOBECOM'2000*, 2000.
[29] S. McCanne and S. Floyd. ns-Network Simulator. *URL http://www-mash.cs.berkeley.edu/ns/*, 1995.
[30] A. Abou-Zeid. *Stochastic Models of Congestion Control in Heterogeneous Next Generation Packet Networks*. PhD thesis, Dept. of Electrical Engineering University of Washington, 2001.
[31] O. Ait-Hellal. *Flow Control in High-Speed Networks*. PhD thesis, Universite de Nice-Sophia Antiopolis, November 1998.
[32] D. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17:1–14, 1989.
[33] W. Press, S. Teukolsky, and B. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, second edition, 1997.
[34] K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ECN) to IP. RFC 2481, IETF, January 1999.
[35] L. Zhang and D. Clark. Oscillating behavior of network traffic: a case study simulation. *Internetworking: Research and Experience*, 1(2):101–12, 1990.
[36] V. Jacobson. Congestion avoidance and control. In *Proceedings of ACM SIGCOMM'88*, pages 314–329, 1988.