A Comprehensive Performance Analysis of Random Early Detection Mechanism

Tuan Anh Trinh and Sándor Molnár

High Speed Networks Laboratory

Department of Telecommunications and Telematics

Budapest University of Technology and Economics

H-1117, Magyar tudósok körútja 2., Budapest, Hungary

E-mail: {tuan,molnar}@ttt-atm.ttt.bme.hu

Abstract

One of the most promising active queue management schemes being proposed for deployment in the Internet is the Random Early Detection (RED) scheme. However, research results on RED performance are highly mixed, especially in the field of tuning its parameters. In this paper, a comprehensive performance analysis of RED is presented. We revisit some features in RED and study them in greater detail. We point out that RED, in general, does not possess proportional loss between flows as claimed and widely adopted in previous research. We suggest the generalization of the PASTA property and give a proof for TCP flows. We also evaluate the performance of the Exponential Weighted Moving Average (EWMA) algorithm in RED. We find that EWMA in RED is an unbiased estimator of the average queue-length, regardless of the weighting value w_q. We also point out the theoretical and practical limits of EWMA in RED. Finally, we propose the use of Fuzzy EWMA to RED (Fuzzy RED) to alleviate the inflexibility of RED tuning. We use simulations to evaluate the performance of Fuzzy RED and compare it with other versions of RED. Our simulations show that, in the case of a high workload and a high level of variation, Fuzzy RED, by tracking system variation in an on-line manner, improves RED performance in a number of important router-based metrics like packet loss rate, average queueing delay, link utilization, and global power.

1 Introduction

Traffic in the Internet is composed of flows with different nature and different characteristics, as more and more new IP-based applications are brought into existence. Some of them are

congestion-aware and some are not. As a consequence, end-to-end congestion control algorithms such as those in TCP are not enough to prevent congestion in the Internet, and they must be supplemented by control mechanisms inside the network. Since routers are the common points for all flows, it is reasonable to detect and control congestion at these places, at least globally. The Drop Tail buffer management scheme does little in this respect. To face this problem, Sally Floyd *et al* in [6] proposed the Random Early Detection (RED) scheme that can efficiently manage the buffer at the router to avoid congestion. Basically, RED provides congestion avoidance by controlling the average queue size and dropping incoming packets at random before the buffer gets full. The average queue size should be kept low, while fluctuations in the actual queue size should be allowed to accommodate bursty traffic and transient congestion. RED was claimed in [6] to provide: congestion avoidance, appropriate time scales, no global synchronization, maximizing global power and fairness. However, RED has some problems to face. First, it is not a thoroughly understood scheme [13]. Second, it has many parameters, and consequently, it is hard to tune [3].

Regarding the literature on RED, we devide it into two classes. The first class largely deals with analyzing and configuring RED, while keeping the algorithm intact. The second class considers how to change the original RED to have better performance. In fact, there is no distinct border between the two classes. In respect to analyzing RED, May *et al* [14] proposed a simple analytic model of RED and concluded (among others) that RED, in certain circumstances, provides no better performance than Drop Tail. Christiansen *et al* evaluated RED with pure web traffic and concluded that RED offers no clear advantage over Drop Tail, at least in terms of delay. The paper also reports that performance is quite sensitive to the setting of RED parameters. Problems with tuning and configuring RED parameters can also be found in [24],[16],[5]. In respect to new modification to RED, we would mention Self-Configuring RED in [4] and recently Adaptive RED in [7]. Basically, the authors propose adapting the dropping probability max_p as a function of average queue size to achieve the specified target average queue size in a wide variety of traffic scenarios. Other modifications to RED can be found in [23], [18], [12],[17],[1], [21].

In this paper, we first reexamine some features and performance of the RED mechanism. The main observation is that RED does not, in general, guarantee proportional loss to flows as claimed in [6]. We use the generalization of Poisson Arrivals See Time Averages (PASTA) as suggested in [14] to study this property for TCP arrivals. Regarding RED performance, we find that although choosing the right value for the weighting parameter (w_q) is difficult and sensitive, the Exponential Weighted Moving Average algorithm in RED is an *unbiased* estimator of the

average queue-length, regardless of the value w_q. Furthermore, we propose the use of the Fuzzy Exponential Average instead of EWMA in RED to alleviate the inflexibility of a fixed weighting value to changing system conditions. We find that Fuzzy RED has a more stable performance than standard RED when changes to the congestion level are frequent.

The rest of the paper is organized as follows. In Section II, we give a detailed analysis of proportional loss in RED. Section III shows the simulation topology. Section IV discusses the motivation for Fuzzy RED. Section V describes the Fuzzy RED Mechanism. Section VI evaluates the performance of Fuzzy RED. Finally, Section VII concludes the paper.

2 Proportional Loss Revisited

Loosely speaking, the proportional loss property means that the fraction of marked packets for each connection is proportional to that connection's share of the bandwidth. RED is claimed to possess this property [6]. In addition, proportional loss is widely adopted in the fairness analysis of RED, [12], [9]. However, M. May *et al* in [14] suggested that the claim is true *only if* the arrival flows are Poisson arrivals. This is based on the PASTA (Poisson Arrivals See Time Averages) property of Poisson processes. We take one step further. Notice that PASTA can be generalized to ASTA (Arrivals See Time Averages), [15], and Burke in [2] has shown that the composite stream of exogenous Poisson arrivals and feedback customers is *not* Poisson even though this stream sees a time average. Since TCP flows account for a large portion of Internet traffic, TCP arrivals are mainly of interest. The question that arises is then: Do TCP arrivals see time averages or not?

Proposition 1 *TCP arrivals do not see time averages either with RED, or with Drop Tail.*

Proof. Let $N \equiv \{N(t), t \geq 0\}$ be the queue length process and $A \equiv \{A(t), t \geq 0\}$ be the arrival process. For an arbitrary set B in the value space of N, define

$$U(t) = \begin{cases} 1 & \text{if } N(t) \in B \\ 0 & \text{otherwise} \end{cases}$$

If B is the stationary queue-length, then U is the event that N remains in that state. According to Theorem 1 in [20], the future increments of A should not depend on the past of U. Formally, it is the *Lack of Anticipation Assumption (LAA)*. That is, for each $t \geq 0, \{A(t+u) - A(t), u \geq 0\}$ and $\{U(s), 0 \leq s \geq t\}$ are independent. Now, let us consider the mechanism of TCP. For the sake of simplicity, we take TCP Reno for our analysis. Let W be the congestion window size and W_{th} the threshold value. Notice that if the sender always has data to send then the congestion

window is approximately the number of packets that were sent but not yet acknowledged. The number of packets going in a forward direction, in a stable period, is approximately half of this value (since the other half are ACKs in the backward direction). Consequently, the dynamics of the congestion window reflect the dynamics of the packet flows feeding a router.

1. After every nonrepeated acknowledgment: if $W < W_{th}$, set $W = W + 1$; *Slow Start Phase* else set $W = W + 1/W$; *Congestion Avoidance Phase*

2. **When the duplicate acknowledgments exceed a threshold, retransmit next expected packet; set** $W_{th} = W/2$**, then set** $W = W_{th}$ **and enter** *Fast Recovery Phase*

3. Upon timer expiration, the algorithm goes into slow start: set $W_{th} = W/2$ set $W = 1$.

Let's consider Phase 2, when the congestion window is halved after sensing duplicate acknowledgements. Duplicate ACKs imply dropping of packets at the buffer and that the buffer at the router is full (for Drop Tail) or potentially full (for RED). That is, the future increments of A in this phase are *dependent* on the past of U. And so, the LAA fails. Consequently, TCP arrivals do not see time average either with RED, or with Drop Tail gateway. Q.E.D.

Corollary 1 *We cannot achieve proportional loss between TCP flows either with RED or with Drop-Tail.*

Remark 1 *A more general condition of ASTA is LBA [15](Lack of Bias Assumption) which only requires that U and the conditional intensity, η_U, of N, given U, are point-wise uncorrelated. Certainly the uncorrelated condition is weaker than the independence condition. However, we can similarly show that this condition also fails.*

Remark 2 *ASTA, in the absence of Poisson flows, are all in the category of networks of quasi-reversible queues; in particular, for the M/M/1 queue with feedback. Once again, Burke in [2] has shown that the composite stream of exogenous Poisson arrivals and feedback customers is not Poisson even though this stream sees a time average.*

Remark 3 *It is noteworthy, however, that for quasi-reversible queues* **in isolation**, *LBA implies Poisson arrivals [15].*

Remark 4 *Let us assume that the service time at the router is exponentially distributed (Markovian service). In this case, consider the G/M/1 queue. We allow the arrival process to be general. Certainly, the arrivals generally (except Poisson ones) do not see time averages, but due to the duality of M/G/1 and G/M/1, we can explicitly express these two values by each other [11].*

3 Simulation Topology

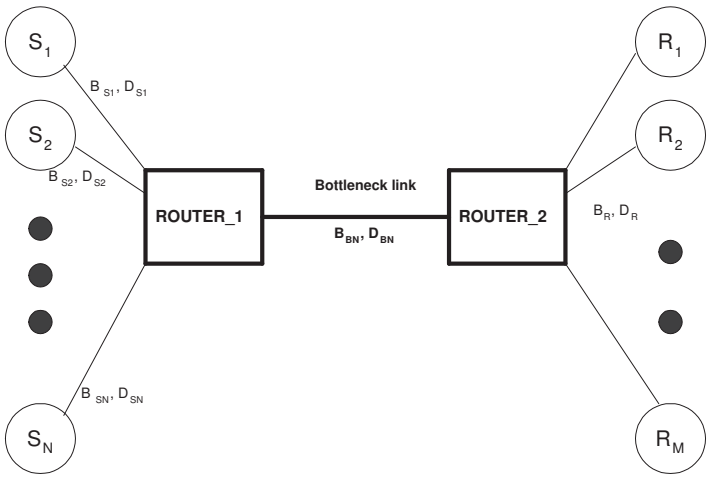


Figure 1.: Simulation Topology

Figure 1. shows the topology template for all of our simulations throughout this paper. We consider the general topology of N senders $S_1, S_2, .., S_N$ and N access links. The i-th access link is specified by bandwidth B_{Si} and delay D_{Si}. Router1 is the access router. We suppose the link between Router1 and Router2 is a bottleneck link with bandwidth B_{BN} and delay D_{BN}. We also add M receivers at the other end in case we want to generate backward traffic. However, unless otherwise stated, we consider the bottleneck link is the only sink.

4 Motivation for Fuzzy RED

4.1 Pitfalls in Tuning RED Parameters

One of the inherent weaknesses of RED is parameter sensitivity. Extensive research has been devoted to this issue and many publications have highlighted various aspects of this issue. However, the question of how to configure the parameters of RED for optimal performance is still open. Christiansen *et al* in [3] examined the impacts of tuning RED's parameters on end-user *delay*, and concluded that for links carrying only web traffic, RED queue management appears to provide no clear advantage over the Drop-Tail gateway for end-user response times. M. May *et al* in [14] use a simple analytic model to evaluate RED performance in terms of *loss rate, link utilization, delay* and *delay variation*.

In this section, we use simulations to examine the impacts of tuning different RED parameters and compare their performance with Drop-Tail. We concentrate on three router-based metrics: *link utilization*, *link loss rate* and *average queuing delay*. We believe that these metrics clearly

provide insight into the performance of queueing management algorithms at routers because end-user metrics of interest (such as end-user delay) are mainly dependent on these metrics. Our simulations reveal two main points. First, RED with fixed, default parameters is no better than Drop-Tail, at least in terms of the examined metrics. Second, there exist parameter tunings of RED so that they can perform somewhat better than Drop-Tail. However, these parameter settings do not increase RED performance both in link utilization and average queuing delay simultaneously. Rather, in this case, RED performs better than Drop-Tail in terms of *global power* defined in [6] as the ratio of throughput to delay.

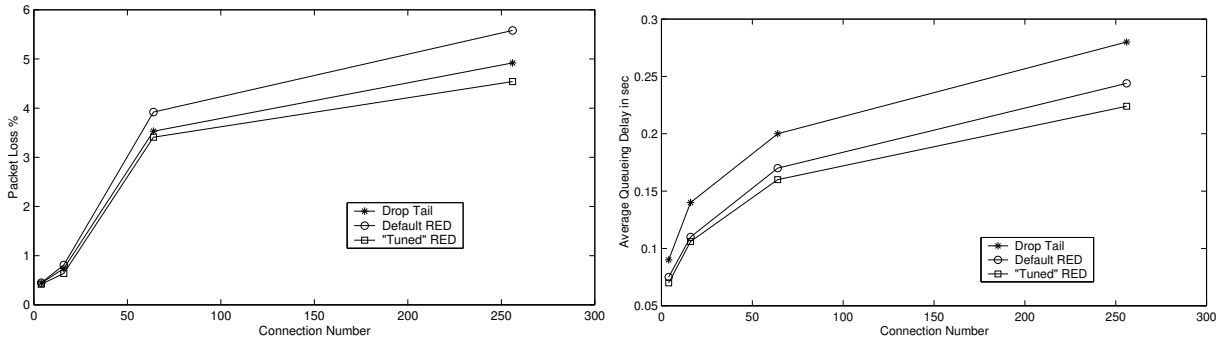
Impact of weighting parameter w_q. We examine the impact of tuning the weighting parameter w_q when other parameters are left unchanged and equal to the default values (max_p=0.1, min_{th}=10 packets, max_{th}=30 packets according to the buffer size of 50 packets). The bottleneck link bandwidth is 15 Mb/s, with delay 50 ms. The access links are all 100 Mb/s. All connections are TCP connections with a packet size of 1000 bytes. To simulate the impact of w_q on different workloads, we examine it with an increasing number of connections (4, 16, 64, 256, accordingly). Increasing the number of connections means increasing the workload feeding the router at the bottleneck link. To simulate high level of variation of incoming TCP traffic, we set the access link delays in a range from 10ms to $(10+N-1)$ ms, where N is the number of connections (nodes). The simulation time used is 30 seconds. We experience a large fluctuation in queue length dynamics in the first few seconds (typically around 5 seconds in our case) due to TCP's first slow starts. So, the simulation time should not be too short. We find that 30 seconds simulation time is adequate to ensure statistical accuracy and to match TCP session time details.

RED is tuned according to the recommendation found in [7]. We set

$$w_q = 1 - exp(-1/B_{BN}) \tag{1}$$

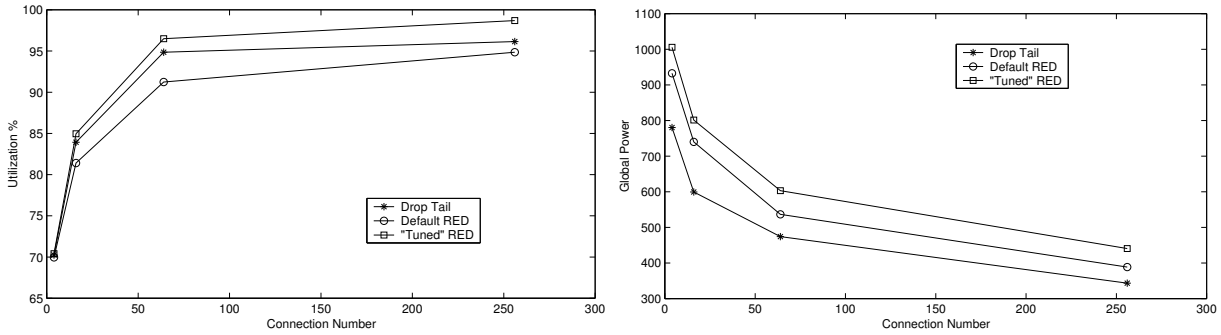
where B_{BN} is the bottleneck link capacity. In our simulation, B_{BN} is set to 15 Mb/s, so w_q is set to 0.005, accordingly.

Figure 2.(a) shows the impact of tuning w_q on link loss rates. As we can see, Drop-Tail performs better than the default tuning of RED, at least in terms of link loss rates. We observe that as the number of connections is small and the round-trip times are relatively in the same range, link loss rates are relatively similar. However, as the number of connections increases, the difference becomes significant. The simulation results reveal to us that there exists a parameter tuning for RED that produces better performance than Drop-Tail. However, Drop-Tail seems to be more robust than a number of cases with RED, especially when max_{th} is far from buffer size and max_p is high (aggressive early detection). Figure 2.(c) shows the performance of RED



(a) Loss rates

(b) Average Queueing Delay



(c) Utilization

(d) Power

Figure 2.: Impact of weighting parameter on router-based performance metrics: RED vs. Drop-Tail

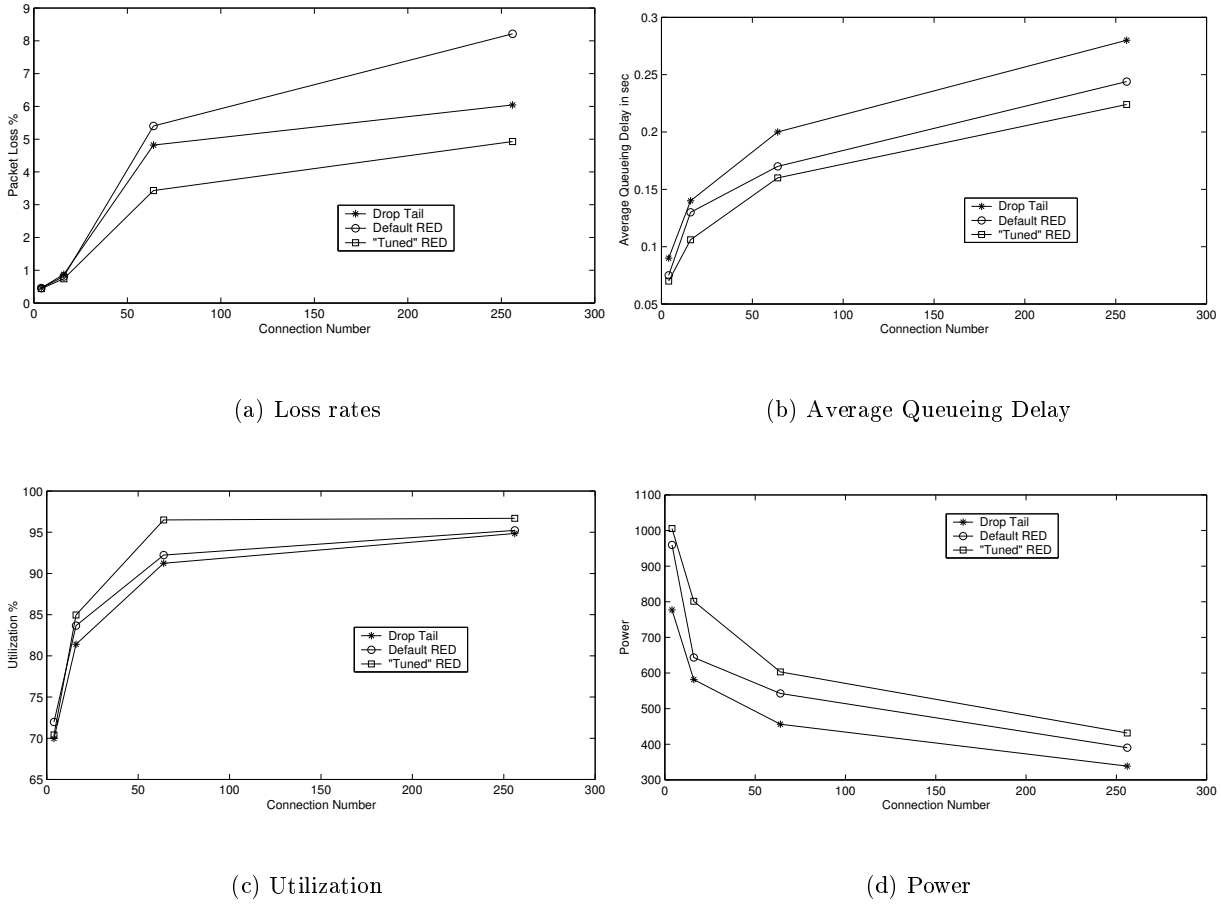
and Drop-Tail in terms of link utilization. We observe that default RED is rather aggressive in detection, thus reducing the utilization of link capacity, especially when the workload is high (increased connection number).

As expected, the results are different with delay. Figure 2.(b) shows that both versions of RED (default and tuned) have smaller average queueing delay than Drop-Tail. However, we have to find the trade-off between average queueing delay and link utilization. We use global power to judge the trade-off performance of Drop-Tail and different parameter settings of RED. Figure 2.(d) shows that RED indeed performs better than Drop-Tail in terms of global power. However, this metrics is hardly observable by the end-user.

Impact of dropping parameter max_p. We examine the impact of tuning the dropping parameter max_p when other parameters are left unchanged and equal to the default value(w_q=0.02, min_{th}=10 packets, max_{th}=30 packets according to the buffer size of 50 packets). The simulation topology is the same as the simulation with w_q. The bottleneck link bandwidth is 15 Mb/s, with a delay of 50 ms. The access links are all 100 Mb/s. All connections are TCP connections

with a packet size of 1000 bytes. To simulate the impact of max_p on different workloads, we examine it with an increasing number of connections (4, 16, 64, 256, accordingly). To simulate a high level of variation of incoming TCP traffic, we set access link delays ranging from 10ms to $(10 + N - 1)$ ms, where N is the number of connections (nodes). The simulation time is 30 seconds.

RED is tuned according to the recommendation in [4], with α and β are set to 3 and 2, respectively.



(a) Loss rates

(b) Average Queueing Delay

(c) Utilization

(d) Power

Figure 3.: Impact of dropping parameter on router-based performance metrics: RED vs. Drop-Tail

As we can see in Figure 3.(a) and 3.(b), we have similar results in terms of loss rates and delay as with the w_q tuning simulations. However, Figure 3.(c) shows that tuning max_p according to Adaptive RED in [4], unlike the w_q tuning simulations, also improve link utilization. Power, as a ratio of throughput to delay, is thus certainly improved.

What we can conclude here is that, fixed, default RED shows no clear advantage over Drop-Tail in a number of crucial router-based performance metrics. However, there exists a parameter tuning that can improve RED performance. The problem remains here is that, as the conditions

are changing, how to adapt the tuning properly to maintain robust performance.

4.2 Adaptive RED

Consider the dropping function in RED. We observe that the adaptation of any parameter will affect the overall system performance. We see no clear justification for adapting only max_p rather than min_{th} and max_{th}, as long as $min_{th} < max_{th} < K$.

In Sally's Adaptive RED, the authors proposed the tuning of w_q based on link capacity. However, what we really consider here is *available* capacity, which is changing, and the dynamics of which is yet to be estimated.

4.3 Reasons for Fuzzy Extension

4.3.1 Theoretical Limits of EWMA in RED

For any fixed $w_q \in [0, 1]$, let $a\hat{v}g_t$ be the estimator of the average queue length by:

$$a\hat{v}g_t = w_q q_t + (1 - w_q) a\hat{v}g_{t-1} \tag{2}$$

where q_t is the instantaneous queue length at time t.

Lemma 1 *If $\{q_t\}$ is stationary with $E(q_t) = \mu_q$ then $a\hat{v}g_t$ is an* unbiased *estimator of μ_q, regardless of the weighting value w_q.*

We consider this as a well-known fact in statistics.

Now, let's consider the variance of this estimator. Without losing generality, we can suppose that $q_1 = 0$, that is the queue starts from empty. Let σ^2 be the variance of $\{q_t\}$.

Lemma 2 *[22] If $q_1, q_2, ...$ are independent (uncorrelated) then the variance of the estimator can be calculated as:*

$$D^2(a\hat{v}g_t) = \sigma^2 \frac{w_q - w_q(1 - w_q)^{2t-2}}{2 - w_q} \tag{3}$$

From Equation 3, if w_q is small ($w_q \approx 0$) then $D^2(a\hat{v}g_t) \approx \sigma^2 \frac{w_q}{2}$ as $t \to \infty$. Now consider the case when $q_1, q_2, ..$ are correlated. Denote $\gamma(k) = E\Big[(q_{t+k} - \mu_q)(q_t - \mu_q)\Big]$ the covariance function of $\{q_t\}$ at lag k and $\varrho(k) = \gamma(k)/\gamma(0)$ the correlation function of $\{q_t\}$ at lag k.

Proposition 2 *The variance of the estimator can be calculated as:*

$$D^2(a\hat{v}g_t) = \sigma^2 \frac{w_q - w_q(1 - w_q)^{2t-2}}{2 - w_q} + 2 \sum_{k=1}^{t-2} \varrho(k) \sum_{j=0}^{t-2-k} w_q^2 (1 - w_q)^{2j+k} \tag{4}$$

This proposition is actually the corollary of Lemma 1 and Lemma 2.

Remark 5 *The coefficient* $\varrho(k) \rightarrow \frac{w_q}{2-w_q}(1-w_q)^k$ *as* $t \rightarrow \infty$. *Interestingly, the correlation function* $\varrho(k)$ *in the expression is also "exponentially weighted" with the weighting parameter* $1 - w_q$.

Remark 6 *The additional term contributes to the variance of the estimator. This makes the estimator worse (it is not so good already, compared with a moving window), since it increases the variance of the estimator. In practice, empirical and simulation analysis in [19] show that the queue-length process is not only correlated, but exhibits fractal properties, e.g. self-similarity.*

4.3.2 Practical Limits of EWMA in RED

The standard Exponential Weighted Moving Average applied in RED possesses a number of good properties. It is easy to be implemented and it requires only a small buffer size for the storage of samples. It is, as mentioned in the previous section, also an unbiased estimator of the mean. However, it is inflexible in some points. First, when we average the queue-length, we are implicitly choosing a *time scale* over which to average it. The problem is then "What should that time scale be?". Intuitively, it should match the round-trip time of a typical TCP connection through the RED buffer. In practice, however, TCP connections can have round-trip times which vary by several orders of magnitude. Furthermore, TCP is self-clocking and so already has its own averaging mechanism built-in which automatically averages over a round-trip time. So why should we try to average something that is already doing its own averaging and when it's simply impossible to get the time scale right anyway? Second, RED was basically designed to face with *transient congestion* [6] and *highly periodic* network traffic, especially TCP traffic. In this respect, the standard EWMA gives a *fixed* weight to past history, thus ignoring transient phases in system dynamics. In [6], the authors proposed an analysis of bounds (or guide-lines) for the weighting value w_q. The analysis in that paper is only for a *given* burst size and buffer size. In other words, we need to know these parameters *a priori* in order to find an appropriate w_q to meet our performance target. A fixed w_q is inflexible in the sense that the EWMA algorithm cannot adapt to the changing condition of the incoming traffic. To alleviate this problem, we propose the use of Fuzzy Exponential Averaging [10], which automatically determines a 'good' value of w_q, and is able to change this value on-line if the system behavior changes. Since the RED dropping mechanism is based on the estimated average queue-length, with "good value", we mean that RED can better keep track with queue-length variations, and consequently, reduces the number of unnecessarily dropped packets at the router.

5 Fuzzy RED Mechanism

We basically keep the RED mechanism intact and only modify the weighting parameter w_q. When estimating the average queue-length at the router, instead of using a fixed weighting parameter, we apply Fuzzy EWMA. Details about Fuzzy EWMA are described in the original paper [10]. Now, we shall discuss how Fuzzy EWMA works in our case.

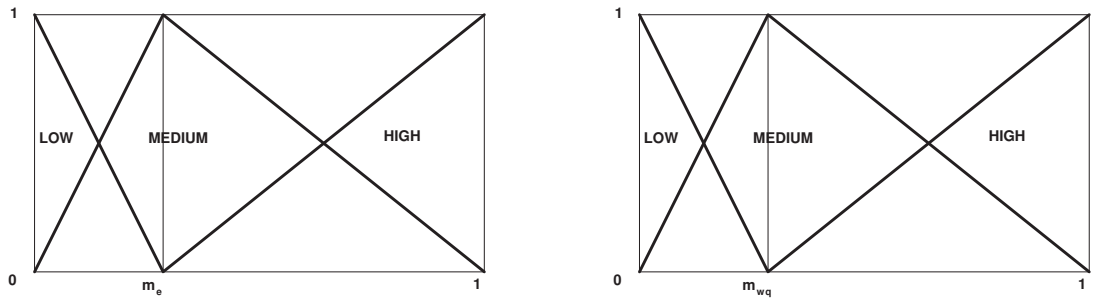
5.1 Construction of Fuzzy RED Mechanism

Consider a discrete time system with q_k, the queue length at the buffer at time k, as the state variable. The system can span a spectrum varying from 'steady' (stationary) to 'noisy' (non-stationary). Let $\hat{q}_k$ be the estimate of q_k, then observation noise (error) is $q_k - \hat{q}_k$. To see the relation between error and the predictor, we define scaled error as $|q_k - \hat{q}_k|/\hat{q}_k$. From now on, if not further mentioned, we deal with this error, because it gives us insight into how the error is related to the estimated queue-length. The variance of system and observation noise is the problem. We need to construct a predictor that can adapt to the changing of system dynamics. We consider the Fuzzy EWMA for this purpose.

The first question we need to deal with is how to define the control rules. We assume that when the queue stays in its stationary (stable) state, the *estimation error* is small. That is, if the dynamics of queue-length in the buffer has little perturbation, then the exponential averaging technique will produce a predictor that is usually close to the actual system state (error is small). In this case, w_q should be large. In contrast, when there is a large variation in queue-length, past history cannot predict the future well (the error is high). In this case, we set w_q low, so that the estimator can track changes in the system. Finally, since we do not have a good grasp of the state dynamics, we only define three gradations in the values of w_q and *error*. In addition, keeping the number of gradations minimal reduces the overhead computing time for the algorithm. Thus, we adopt the following control rules:

- IF *error* is HIGH THEN w_q is LOW
- IF *error* is MEDIUM THEN w_q is MEDIUM
- IF *error* is LOW THEN w_q is HIGH

Secondly, we need to answer the question: HIGH, MEDIUM, LOW are related to what? The answer for this question is equivalent to defining the membership functions for *error* and w_q. For the sake of simplicity, we use the trapezoid form (the conventional and simplest form) for these two variables. The question that remains is how to specify the membership functions.



(a) Error Membership Function

(b) Weighting Value Membership Function

Figure 4.: Membership Functions

5.1.1 How to Specify Membership Functions

This question is equivalent to specifying m_e and m_{w_q}, as shown in Figure 4.. It can be done on-line by neural network training algorithms (such as back propagation), but this is time consuming and lacks simplicity. So we do the training off-line to find appropriate values for these parameters. When it is good, it can be fixed. The outcome of the training, for the topology of our simulations, have m_{w_q} in [0.002..0.05] range and m_e in [0.06..0.2] range. Interestingly, the results for medium value of w_q are close to the value proposed by Sally Floyd *et al* in [7]. At this point, it seems that we arrive at the point where we started. That is, we still need to train the system for some *a priori* knowledge. The only difference here, and also the intuitive force behind our approach, is that once a good trained parameter is chosen, it can be fixed, and from that point, the system will adapt to the changing condition of the incoming traffic.

It should be mentioned that we only apply the simplified version of Fuzzy EWMA proposed in [10] without a smoothed proportional error because we find that it is very time consuming and in consequence greatly affects the performance.

We implemented the proposed algorithm in ns-2. Except for the EWMA algorithm part, all other features in RED are kept intact.

6 Simulation Results

6.1 Stationary Performance

To examine the stationary behavior of Fuzzy RED, we first run the simulation with the same parameters as in previous Sections. That is, the access links are all 100 Mb/s. The bottleneck link bandwidth is 15 Mb/s, with delay 50 ms. Buffer size at router-1 is set to 50 packets, min_{th} is set to 10 packets, max_{th} is set to 30 packets. Connections are TCP connections with a packet

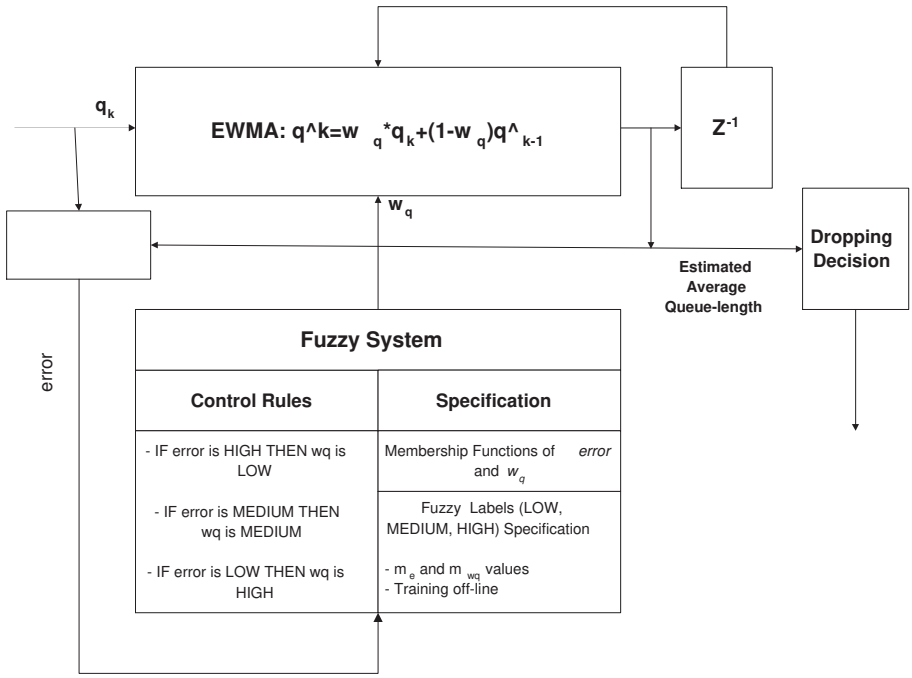
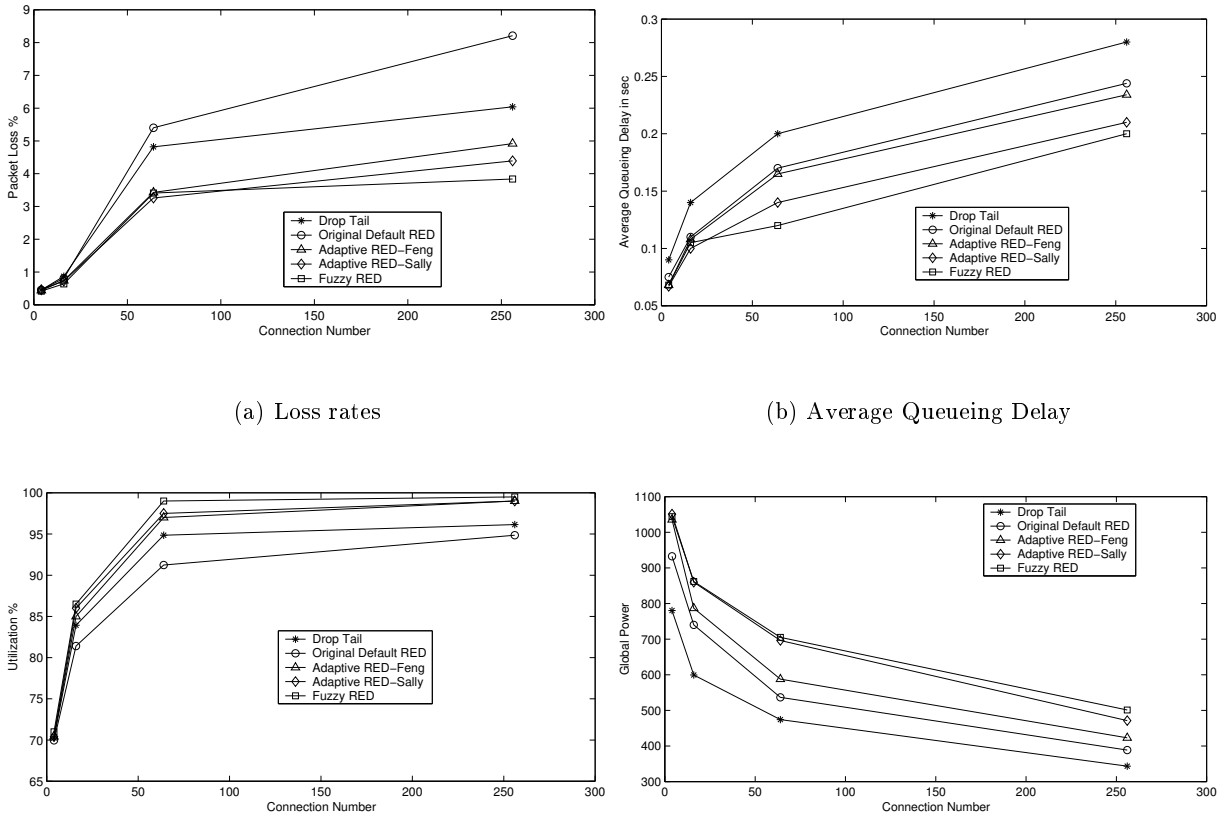


Figure 5.: Flow diagram of Fuzzy RED

size of 1000 bytes. To simulate the impact of different workloads on performance of versions of RED and Drop-Tail, we examine them with an increasing number of connections (4, 16, 64, 256, accordingly). The simulation time is 30 seconds. We compare our proposed Fuzzy RED not only with Drop Tail and default RED, but also with other Adaptive RED versions, such as Adaptive RED in [4] (we call it Adaptive RED-Feng), and Adaptive RED in [7] (we call it Adaptive RED-Sally).

Scenario 1. *TCP incoming traffic with different RTTs.* To simulate high level of variation of incoming TCP traffic, we set access link delays range from 10ms to $(10 + N - 1)$ ms, where N is the number of connections (nodes).

Figure 6. shows comparative performance of Fuzzy RED against other versions of RED and Drop Tail. As mentioned and explained in previous sections, we concentrate on router-based performance metrics. We learned from our simulations that under light-weight load (few number of connections), there is no significant difference between versions of RED and Drop Tail, and the orders are changing from simulation to simulation. But the situation is different with heavily loaded incoming traffic (eg. 256 connections). In most of our simulations, three versions of Adaptive RED perform closely together in all examined performance metrics. The benefits of Fuzzy RED are more visible when the workload is high (ie. there are many TCP flows, sufficient training data for the Fuzzy Scheme) and the level of variation of burstiness is high (different round-trip times of TCP connections as in this scenario). Original default RED



(a) Loss rates

(b) Average Queueing Delay

(c) Utilization

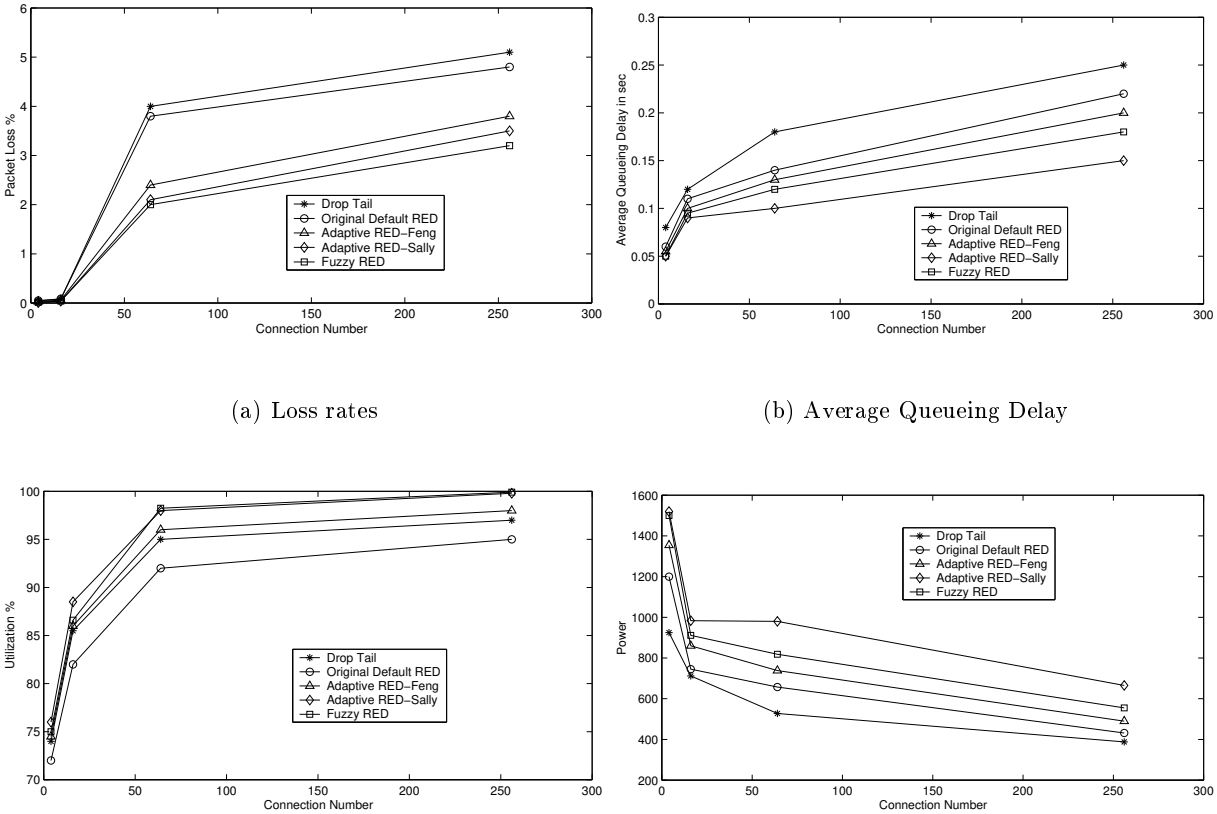
(d) Power

Figure 6.: Router-based performance metrics- different RTTs: Fuzzy RED vs. Adaptive RED versions and Drop-Tail

suffers from a high loss rate because of fixed parameter setting. These fixed default parameters seem to be too aggressive. In terms of loss rate, RED with fixed default parameters, in our case, perform even worse than Drop-Tail. We believe that, this happens because RED, in this case, unnecessarily and too early dropped the incoming packets. Packet loss rates with versions of Adaptive RED in the case of heavy load (256 TCP flows, with different round-trip time setting) oscillate around 5 percent whereas it is far above for Drop Tail and Default RED (6-10 percent). Figure 6.(b) shows the comparative performance of the queueing management algorithms in terms of average queueing delay. We experience the situation where Drop Tail performs worst because Drop Tail only drops packets when the queue is full thus keeping the queue potentially full all of the time. One more thing to mention is that RED with fixed default parameters has low utilization as shown in Figure 6.(c). Interestingly, Figure 6.(d) reveals that all versions of RED (default RED included) perform better than Drop-Tail in terms of global power as mentioned in previous sections. This means that what we really benefit from RED is not only

a low average queueing delay but also the *trade-off* between delay and utilization, at least in terms of global power as defined in [6].

Scenario 2. *TCP incoming traffic with the same RTTs.* To simulate a low level of variation of burstiness, we set access link delays all equal to 10ms.



(a) Loss rates

(b) Average Queueing Delay

(c) Utilization

(d) Power

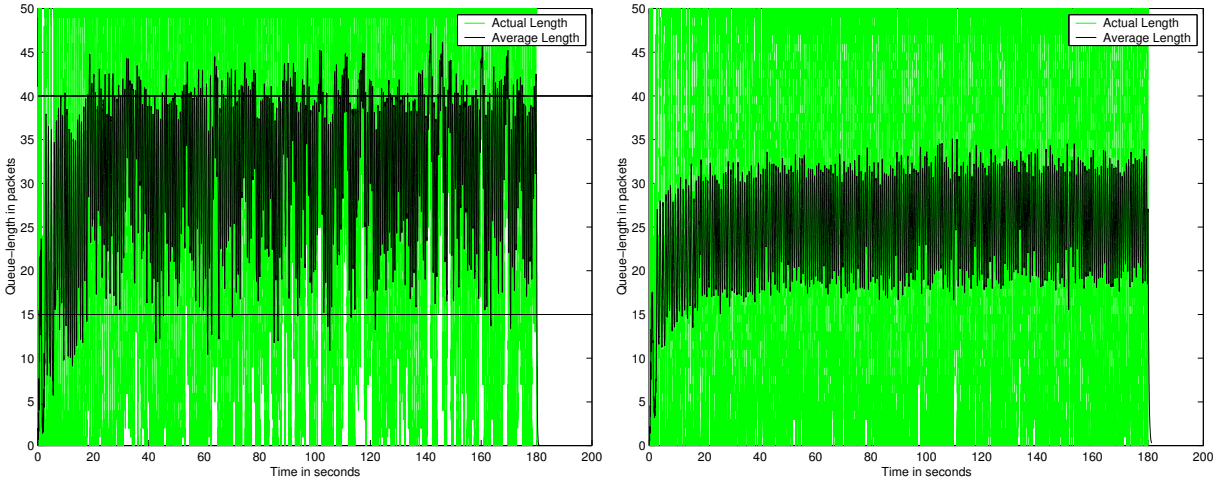
Figure 7.: Router-based performance metrics- Same RTTs: Fuzzy RED vs. Adaptive RED versions and Drop-Tail

Figure 7. shows comparative performance of Fuzzy RED against other versions of RED and Drop Tail. As we can see in Figure 7.(a), packet loss rates with versions of Adaptive RED in the case of heavy load (256 TCP flows) oscillate around 3 percent whereas it is much higher for Drop Tail and Default RED (4-5 percent), which are significantly smaller than simulation with different RTTs. The situation is similar with delay and link utilization (nearly 100% with Adaptive RED-Sally and Fuzzy RED, 256 connections) in the way that all the versions perform somewhat better with the same RTTs than with different RTTs, as shown in Figure 7.(b) and Figure 7.(c). Global power, as a result and shown in Figure 7.(d), is certainly improved in all cases. An important observation to be noticed here is that, for this scenario, Adaptive RED-

Sally out-performs all other versions of RED and Drop-Tail. This outcome shows the benefit of simplicity in Adaptive RED-Sally compared to Fuzzy RED. However, as we mentioned earlier, we basically design Fuzzy RED to deal with high workload and high level of variations of burstiness which is, we believe, a realistic condition of today's Internet.

What we have been discussing so far is only for pure TCP traffic. However, as RED routers are also responsible for directing and managing other flows of different traffic such as voice and video traffic. For these applications, other performance metrics are also of importance. For example, for VoIP (Voice over IP) applications, not only the average delay but delay variation (jitter) heavily affects the end-user view of performance. So in case both TCP flows and UDP flows sharing the router, queue-length variation should be kept low for the sake of the Quality of Service (QoS) for voice applications. We simulate RED and Fuzzy RED with 1000 flows (500 TCP flows and 500 UDP flows). Since the number of connections in this significantly increased, we also increase the duration time of the simulation to 180 seconds in order to ensure statistical accuracy of our results.

Scenario 3. *Queue-length variation with the same RTTs.* To simulate low level of variation of burstiness, we set access link delays all equal to 100ms.



(a) Original Default RED

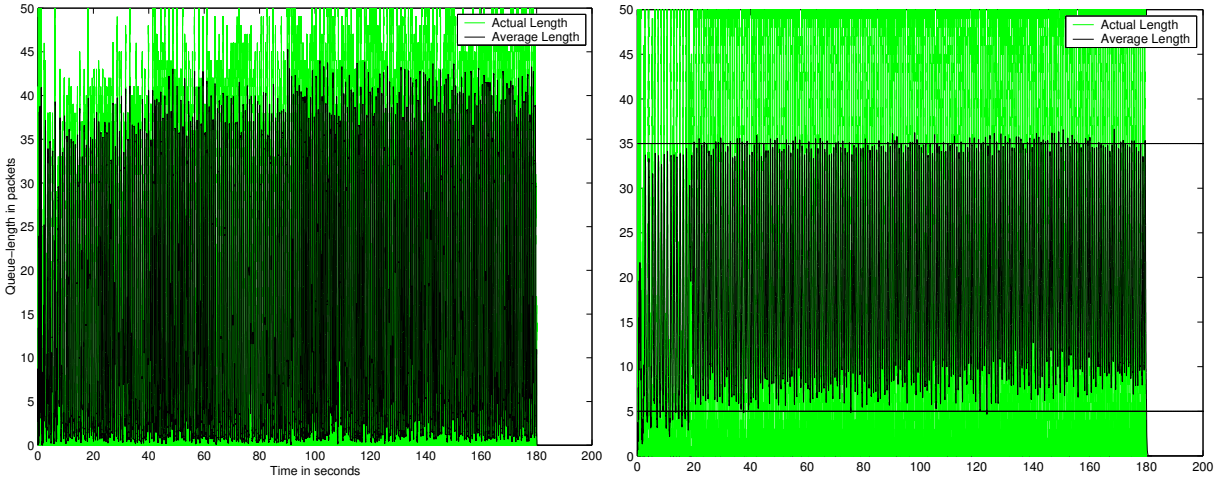
(b) Fuzzy RED

Figure 8.: Queue-length variations: Fuzzy RED vs. RED

Figure 8. shows that although the overall long run average queue-length is quite similar for RED and Fuzzy RED (around 25 packets), the variation in queue length of RED is significantly higher with RED than with Fuzzy RED. High variation in queue-length results in high delay variation (jitter), thus decreasing the quality of voice services.

Scenario 4. *Queue-length variation with different RTTs.* To simulate a high level of vari-

ation of incoming TCP traffic, we set access link delays to range from 100ms to $(100 + N - 1)$ ms, where N is the number of connections (nodes).



(a) Original Default RED

(b) Fuzzy RED

Figure 9.: Queue-length variations: Fuzzy RED vs. RED

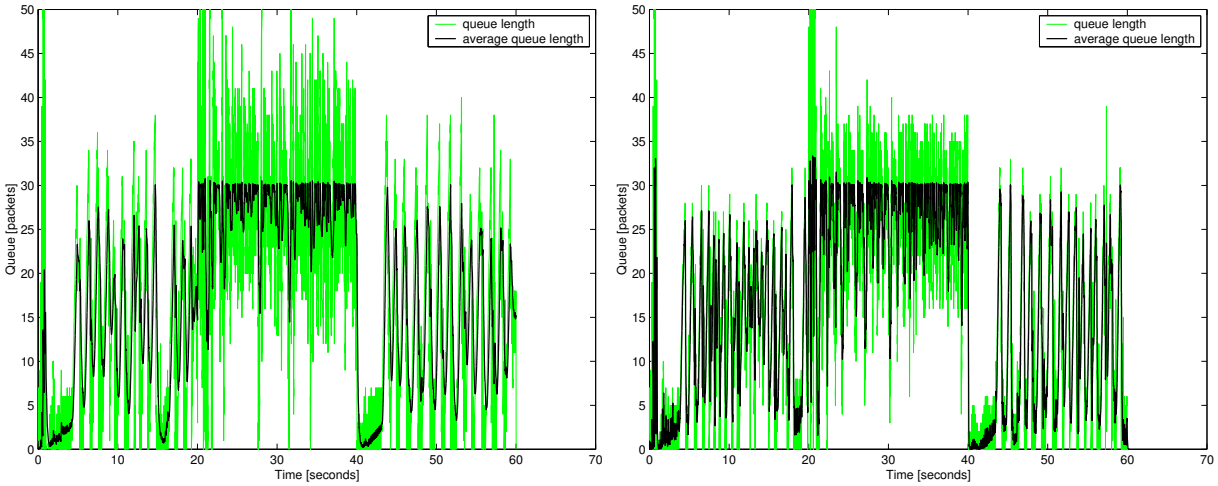
As we can see in Figure 9., queue-length variation with RED is significantly higher with RED than with Fuzzy RED. Moreover, Figure 8. and Figure 9. clearly show that performance of both RED and Fuzzy RED, respectively, decrease when we simulate with different RTTs.

6.1.1 Performance with Non-Stationarities

We examine performance of Fuzzy RED with level-shifts, which are the most common non-stationarity effects observed. We run the simulation in three parts each with length of 20 seconds. First, 10 TCP flows are active. After 20 seconds, an addition of 10 TCP flows enter. After 20 seconds, these 10 flows are terminated. All other parameters are the same as the simulation for the stationary case.

Scenario 5. *Performance without background traffic*

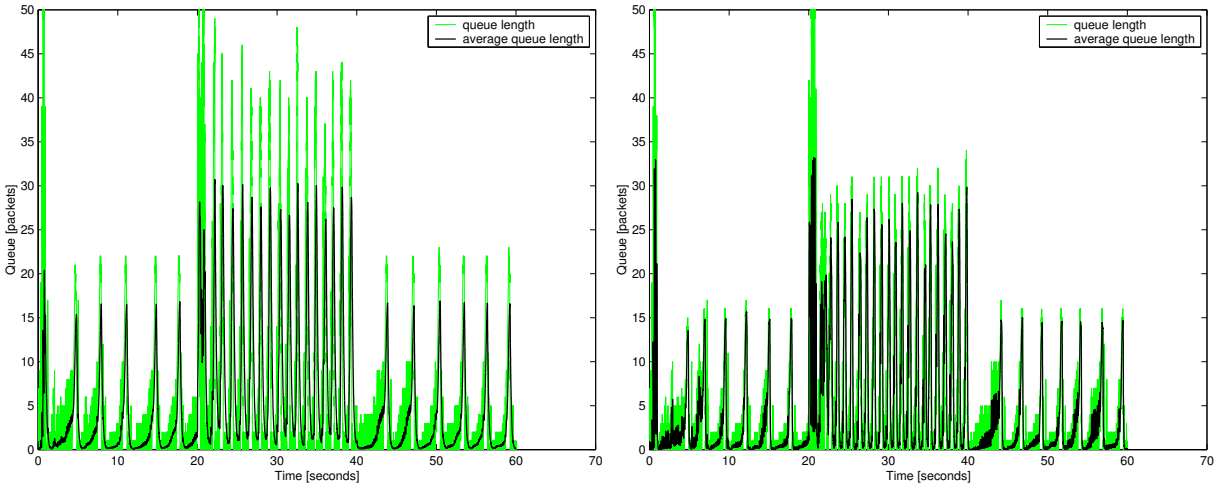
Figure 10. shows the dynamic of queue-length with RED and Fuzzy RED. After the increase in workload (additional 10 flows enter), actual queue-length with RED vary widely in the full range between 1 and 50. Fuzzy RED adapts to the sudden change in condition, and do not allow the queue-length to change quickly, keeping the actual queue-length in the target of 15 to 35 packets. After the decrease in workload (10 flows leave), it takes around 2 seconds for both RED and Fuzzy RED to get back to a normal condition, but Fuzzy RED produce somewhat smaller values for average queue-length and queue-length variation.



(a) Original Default RED (b) Fuzzy RED

Figure 10.: Performance with level-shifts: RED vs. Fuzzy RED

Scenario 6. *Performance with background traffic.* We run the simulation with some background web (http) traffic by adding short http sources to the examined long FTP connections. Each http source sends a request (a packet) to its destination, which replies with a file of size that is exponentially distributed with a mean of 125 KB-packets (the file size distribution can also be modeled by the Weibull distribution, but here, we use exponential distribution, for the sake of simplicity). The waiting time for another request is also exponentially distributed with a mean of 1 second.



(a) Original Default RED (b) Fuzzy RED

Figure 11.: Performance with Non-Stationarities: RED vs. Fuzzy RED

We observe in Figure 11. periodicity with both RED and Fuzzy RED. Periodicity is well-known in the dynamics of queue-length in a buffer and it seems that Fuzzy RED does not filter out periodicity, but it adapts to changes somewhat quicker. More importantly, although the average queue-lengths both for Default Original RED and Fuzzy RED are similar, the actual queue-length is higher with RED than with Fuzzy RED.

7 Conclusion

We have demonstrated that RED in general does not guarantee proportional loss to flows and gave a proof for the TCP case. We also analytically evaluated the performance of the EWMA algorithm in RED. We found that the EWMA algorithm in RED is an unbiased estimator of average queue-length, regardless of the weighting value w_q. We also pointed out the theoretical and practical limits of the EWMA in RED. We proposed the use of Fuzzy EWMA to RED (Fuzzy RED). Simulation results show that our proposed Fuzzy RED has some advantages over the original RED in the case of frequently changing congestion. Analytically evaluating Fuzzy RED is a subtle and difficult task, which is left as our future work.

8 Acknowledgement

The idea of applying Fuzzy EWMA to RED was suggested by Prof. S. Keshav. The first author would like to thank Dr. Polly Huang at ETH, Zurich, for valuable comments on ns-2. We also thank colleagues at Ericsson Research at Budapest for their feedback and fruitful discussions.

References

[1] J. Aweya, M. Ouellette, D. Montuno, A. Chapman *A Control Theoretic Approach to Active Queue Management* Computer Networks, 36, 2001.

[2] P. Burke, *Proof of a Conjecture on the Interarival-Time Distribution in M/M/1 Queue with Feedback* IEEE Trans. on Communication, vol. 24, 1976.

[3] M. Christiansen, K Jeffay, D. Ott, F. D. Smith, *Tuning RED for Web traffic* ACM SIGCOMM'00, Stockhom, 2000.

[4] W. Feng, D. Kandlur, D. Saha, and K. Shin *A Self-Configuring RED Gateway* Infocom99, 1999.

[5] V. Firoiu and M. Borden, *A Study of Active Queue Management for Congestion Control* INFOCOM'00.

[6] Sally Floyd and Van Jacobson *Random Early Detection Gateways for Congestion Avoidance* IEEE/ACM Transactions on Networking, vol. 1, no. 4, August 1993, pp. 397-413

[7] Sally Floyd, Ramakrishna Gummadi, and Scott Shenker *Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management,* ACIRI Technical Report, 2001

[8] C. V. Hollot, V. Misra, D. Towsley and W. Gong, *A Control Theoretic Analysis of RED* INFOCOM 2001, Alaska, April 22-26, 2001

[9] P. Hurley, J. Boudec, and P. Thiran, *A Note on the Fairness of Addictive Increase and Multiplicative Decrease* ITC 16, UK, 1999.

[10] S. Keshav, *A Control-theoretic Approach to Flow Control* Proc. ACM SIGCOMM 1991, Sept. 1991

[11] L. Kleinrock, *Queueing Systems: Theory* John Wiley and Sons, 1975.

[12] D. Lin and R. Morris, *Dynamics of Random Early Detection* SIGCOMM'97

[13] M. May, J. Bolot, C. Diot, and B. Lyles, *Reasons not to deploy RED* IWQoS'99, 1999.

[14] M. May, T. Bonald, and J. Bolot. *Analytic Evaluation of RED Performance* Proc. of INFOCOM'00, 2000.

[15] B. Melamed and D. Yao, *The ASTA Property* Frontiers in Queuing: Models, Methods, and Problems, CRC Press, 1995.

[16] V. Misra, W. Gong, D. Towsley, *Fluid-based Analysis of a Network of AQM Routers supporting TCP flows with an Application to RED* SIGCOMM'00.

[17] T. Ott, T. Laksman, L. Gong, *SRED: Stabilized RED* INFOCOM'99.

[18] T. A. Trinh, S. Molnár, *RED Revisited.* The 10th International Conference on Telecommunication Systems Modeling and Analysis, October 3-6, 2002, CA, USA.

[19] G. Vattay, A. Fekete *Self-Similarity in Bottleneck Buffers* Proc. of Globecom 2001.

[20] R. Wolff, *Poisson Arrivals See Time Averages* Operations Research, vol. 30, no 2, 1982.

[21] B. Wydrowski, M. Zuckerman, *GREEN: An Active Queue Management Algorithm for a Self Managed Internet* Proc. ICC 2002, New York, vol. 4, pp 2368-2372, 2002

[22] M. Yajnik, S. Moon, J. Kurose and D. Towsley, *Measurement and Modelling of the Temporal Dependence in Packet Loss* INFOCOM99, 1999.

[23] B. Zheng, M. Atiquzzaman, *DSRED: Improving Performance of Active Queue Management over Heterogenous Networks.* The 25th Annual IEEE Conference on Local Computer Networks, November 9-10, 2000, Tampa, Florida, pp. 242-251.

[24] T. Ziegler, S. Fdida, and Brandauer, *Stability Criteria for RED with TCP traffic* IEEE ICCCN'00