RED Revisited

Tuan Anh Trinh and Sándor Molnár
High Speed Networks Laboratory
Department of Telecommunications and Telematics
Technical University of Budapest
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 RED. However, research results on RED performance are highly mixed, especially in the field of tuning its parameters. In this paper, we revisit some features in RED and study them in greater details. 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 PASTA and give 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 average queue-length, regardless of the weighting value w_q. 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 case of high work load, this mechanism can effectively reduce packet loss while maintaining high link utilization under different scenarios.**

I. 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 to 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 networks. Since routers are the common places that all flows share and go through, it is reasonable to detect and control congestion at these places, at least globally. Drop Tail buffer management scheme does little in this respect. To face this problem, Sally Floyd *et al* in [12] propose 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 [12] 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 [2]. Second, it has many parameters, and in consequence, is hard to tune [3].

Regarding literature on RED, we break them into two classes. The first class largely deals with analyzing and configuring RED, 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 class. In respect to analyzing RED, May *et al* [15] proposed a simple analytic model of RED and concluded (among others) that RED, in certain circumstances, provides no better performance than Tail Drop. Christiansen *et al* evaluated RED with pure web traffic and concluded that RED offers no clear advantage over Tail Drop, 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 [4],[5],[6]. In respect to new modification to RED, we would mention Self-Configuring RED in [14] and recently Adaptive RED in [13]. Basically, the authors propose adapting max_p as a function of average queue size to achieve specified target average queue size in a wide variety of traffic scenarios. Other modifications to RED can only be found in [7],[8],[9].

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 [12]. We use the generalization of Poisson Arrival See Time Average (PASTA) as suggested in [15] 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 Fuzzy Exponential Average instead of EWMA in RED to alleviate the inflexibility of fix weighting value to changing conditions of the system. We find that Fuzzy RED has more stable performance than standard RED when changing in congestion level is frequent.

The rest of the paper is organized as follows. In Section II, we give detailed analysis of proportional loss in RED. Section III shows the simulation topology. Section IV discusses 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.

II. Proportional Loss Revisited

Loosely speaking, proportional loss property means that the fraction of marked packets for each connection is proportional to that connection's share of bandwidth. RED is claimed to possess this property [12]. In addition, proportional loss is widely adopted in the fairness analysis of RED, [7], [16]. However, M. May *et al* in [15] suggested that the claim is true *only if* the arrival flows are Poisson arrivals. This is based on the PASTA (Poisson Arrival See Time Average) property of Poisson processes. We make one step further. First, notice that PASTA can be generalized to ASTA (Arrival See Time Average), [18]. Second, we will examine if TCP arrivals see time average or not.

Proposition 1: TCP arrivals do not see time average neither with RED, nor 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, the U is the event that N stays at that state. According to Theorem 1 in [19], 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. Denote 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 forward direction, in stable period, is approximately half of this value (since the other half are ACKs in 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 is *dependent* on the past of U. And so, the *LAA* fails. Consequently, TCP arrivals do not see time average neither with RED, nor with Drop Tail gateway, Q.E.D.

Remark 1: A more general condition of ASTA is LBA [18](Lack of Bias Assumption) which only requires that U and the conditional intensity, η_U, of N, given U, are pointwise uncorrelated. Certainly uncorrelated condition is weaker that independent condition. However, we can similarly show that this condition also fails.

Remark 2: ASTA, in the absence of Poisson flows, are all in the networks of quasi-reversible queues, in particular, for the M/M/1 queue with feedback. Burke in [17] has shown that the composite stream of exogenous Poisson arrivals and feedback customers is not Poisson even though this stream sees time average.

Remark 3: It is noteworthy, however, that for quasi-reversible queue *in isolation*, LBA implies Poisson arrivals [18]

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 average, but due to to duality of M/G/1 and G/M/1, we can explicitly express these two values by each others [20].

III. Simulation Topology

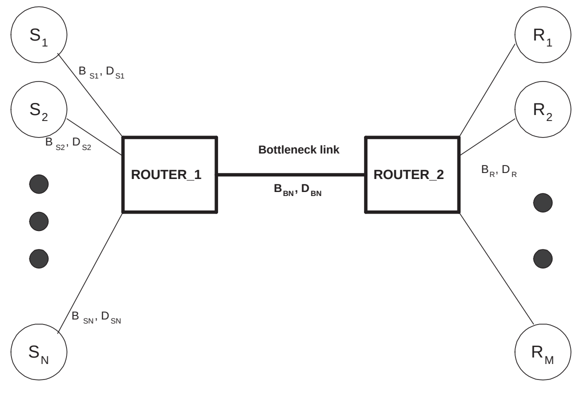

Fig. 1. Simulation Topology

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 receiver at the other end in case we want to generate backward traffic. However, if not further mentioned, we consider the bottleneck link is the only sink.

IV. Motivation for Fuzzy RED

A. Pitfalls in tuning RED parameters

One of the inherent weaknesses of RED is parameter sensitivity. Extensive research has been promoted to this issue and many publications have set light to 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 Drop-Tail gateway for end-user response time. M. May *et al* in [15] use a simple analytic model to evaluate RED performance in terms of *loss rates, link utilization, delay* and *delay variation*.

In this section, we use simulations to examine the performance of RED and Drop-Tail in various ways. We concentrate on three router-based metrics: *link utilization*, *link loss rate* and *average queuing delay*. We believe that these metrics clearly give us the 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 exists a parameter tuning of RED so that it can perform somewhat better than Drop-Tail. However, this parameter setting does 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,* as defined in [12] as the ration of throughput to delay.

For this section, the access links are all 100 Mb/s, with delays range from 10ms to $(10+N-1)$ ms, where N is the number of connections (nodes). 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. The simulation time is 30 seconds.

Connections are TCP connections with packet size of 1000 bytes.

We examine the impacts of tuning the w_q parameter when max_p is left unchanged and equal to the default value (0.1).

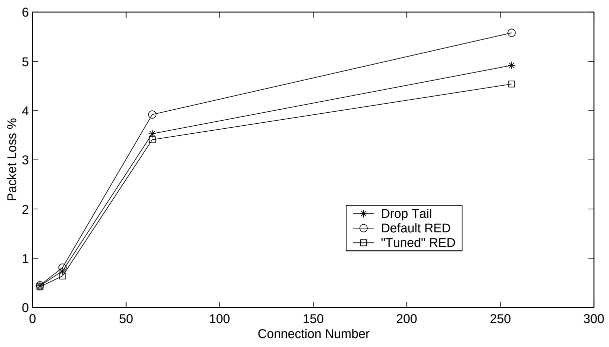

Fig. 2. Loss rates

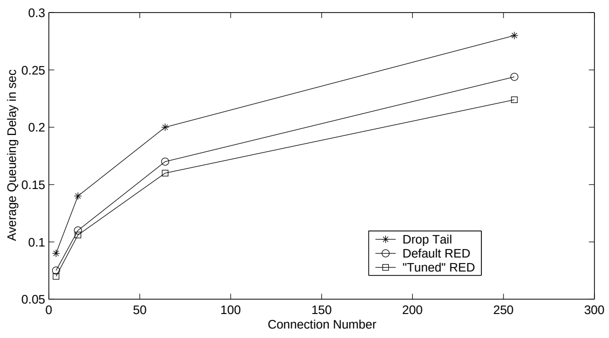

Fig. 3. Average Queueing Delay

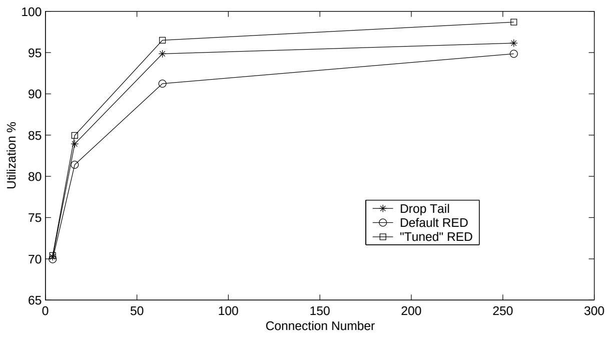

Fig. 4. Utilization

Figure 2 shows the impacts of tuning w_q on link loss rates. The x-axis shows the number of connections, the y-axis shows loss rates at the bottleneck link. The number of connections are 4, 16, 64, 256. Increasing the number of connections means increasing the workload feeding the router at the bottleneck link. We examine different w_q settings with RED: 0.002 (default) and 0.0005. As we can see in the Figure 2, except for $w_q = 0.0005$, Drop-Tail performs better than the default tuning of RED, at least in terms of link loss rates. However, $w_q = 0.0005$ is chosen deliberately according to the recommendation in [13]. We

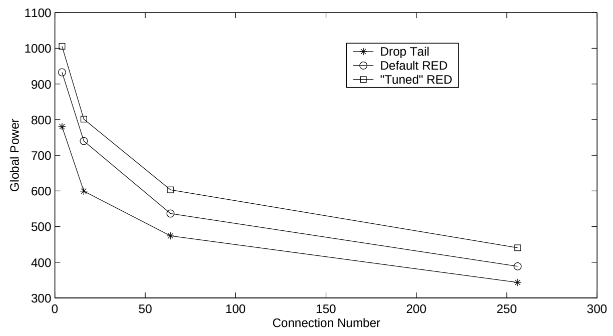

Fig. 5. Power

set

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

where B_{BN} is the bottleneck link capacity. We observe that as the number of connections are small and the round-trip times are relatively of the same range, then link loss rates are relatively the same. However, as the number of connections increases, the difference becomes significant. The simulation result reveals 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 4 shows performance of RED 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 workload is high (increased connection number).

As expected, the result is different with delay. Figure 3 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 examine this by the *global power* as defined in [12], but has not been examined in recent RED papers. The power= Link utilization/Link loss rates. We use power to judge the performance of Drop-Tail and different parameter settings of RED. Figure 5 shows that RED indeed perform better than Drop-Tail in terms of *global power*. However, this metrics is hardly observable by the end-user. What we can conclude here is that, fixed, default RED shows no clear advantage over Drop-Tail in a number of crucial performance metrics. However there exists a parameter tuning that improve RED performance. The problem remains here is that, as the conditions are changing, how to adapt the tuning properly to keep robust performance.

B. 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 of adapting only max_p other 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, what is changing, and the dynamics of which is yet to be estimated.

C. Reasons for Fuzzy Extension

C.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:

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

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

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

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:

$$D^2(\hat{m}_t) = \sigma^2 \frac{w_q - w_q(1 - w_q)^{2t-2}}{2 - w_q} + 2 \sum_{k=1}^{t-2} \varrho(k) \sum_{j=0}^{t-2-k} w_q^2 (1 - w_q)^{2j+k}$$

This proposition can be easily proved by using induction.

Remark 5: The coefficient of $\varrho(k) \to \frac{w_q}{2-w_q}(1 - w_q)^k$ as $t \to \infty$. 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, comparing with moving window), since it increases the variance of the estimator. In practice, empirical and simulation analysis in [11] show that the queue-length process is not only correlated, but even self-similar.

C.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 requires small buffer size for the storage of samples. It is, as proved in previous session, 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 for 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* [12] and *highly periodic* network traffic, especially TCP traffic. In this respect, the standard EWMA gives *fixed* weight to past history, thus ignoring transient phases in system dynamics. In [12], the authors proposed an analysis of bounds (or guild-lines) for the weight 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 adopt with the changing condition of the incoming traffic. To alleviate this problem, we propose the use of Fuzzy Exponential Averaging [21], [22], which automatically determines a ’good’ value of w_q, and is able to change this value on-line if the system behavior changes. Since RED dropping mechanism is based on the estimated average queue-length, with ”good value”, we mean that RED can better keep track with the queue-length variation, and in consequence, reduces the number of unnecessarily dropped packets at the router.

V. 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 is described in the original paper [21]. Now, we will discuss how Fuzzy EWMA works in our case.

A. Construction of Fuzzy Control System

Consider the 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 $q_k - \hat{q}_k$. The variance of system and observation noise is the problem. We need to construct a predictor that can adapt with the changing in system dynamics. We consider the Fuzzy EWMA for this purpose.

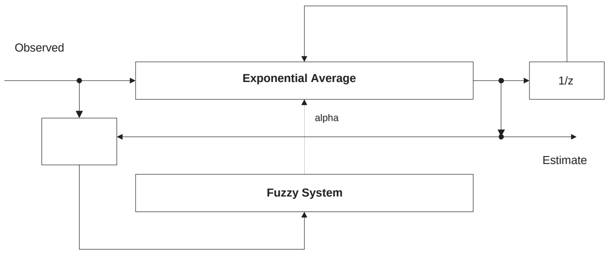

Fig. 6. Flow diagram

The question left is how to define fuzzy 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 future well (error is high). In this case, we set w_q low, so that the estimator can tracks the changes in the system. As the rules of thumb, the changes happen in linear manner with slope 1. Last, since we do not have a good grasp of state dynamics, we only define three gradations in the values of w_q and *error*. In addition, keeping the number of gradations minimal reduces 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

Finally, we need to define Error Membership function. For the shake of simplicity, we use the trapazoid form for these two fuzzy labels.

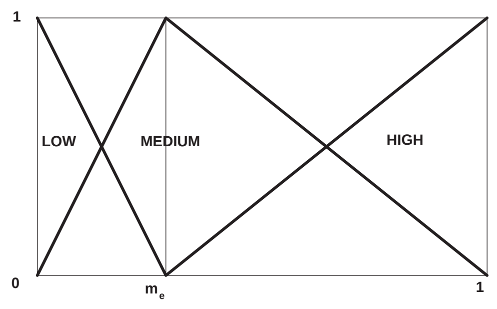

Fig. 7. Error Membership Function

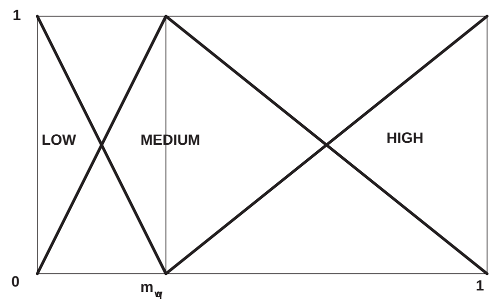

Fig. 8. Wq Membership Function

The question left is how to specify the membership functions.

A.1 How to Specify membership functions

This question is equivalent to specify m_{error} and m_{w_q}. Can be done on-line by neural network training algorithms (such as back propagation), but this is time consuming and lack of simplicity. So we do the training off-line to find these values. When it is good it can be fixed. Interestingly, the results for medium value are close to the value propose by Sally Floyd *et al* in [13].

It should be mentioned that we only apply the simplified version of Fuzzy EWMA proposed in [21] without smoothed proportional error because it is very time consuming and in consequence greatly affects the performance.

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

VI. Simulation Results

A. Stationary Performance

To examine the stationary behavior of Fuzzy RED, we first run the simulation with the same parameter as previous Sections. That is, the access links are all 100 Mb/s, with delays range from 10 ms to $(10 + N - 1)$ ms, where N is the number of connections (nodes). 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. The simulation time is 30 seconds. Connections are TCP connections with packet size of 1000 bytes. We compare our proposed Fuzzy RED not only with Drop Tail and default RED, but also with other Adaptive RED, such as Adaptive RED in [14] (we call it Adaptive RED-Feng), and Adaptive RED in [13] (we call it Adaptive RED-Sally).

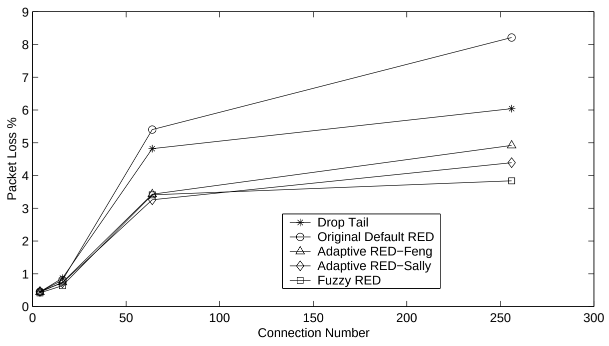

Fig. 9. Loss rates

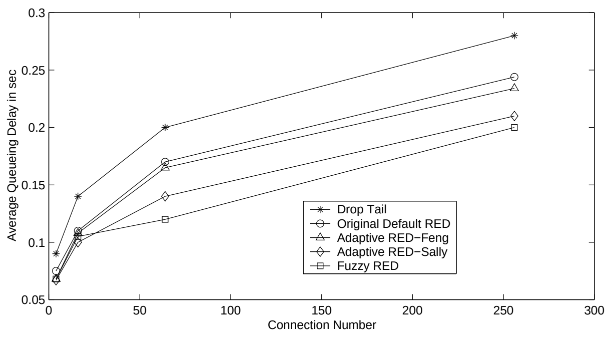

Fig. 10. Average Queueing Delay

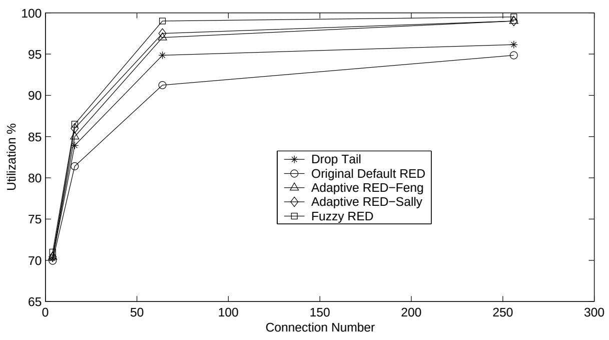

Fig. 11. Utilization

Figures 9.. 12 show 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 experience

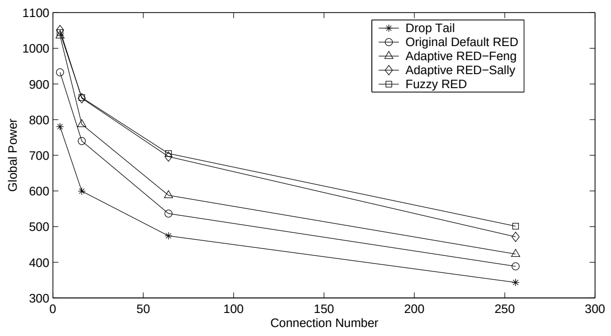

Fig. 12. Power

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 is more visible when the work load is high (ie. there are many TCP flows, sufficient training data for the Fuzzy Scheme) and the level of variation is high (different round-trip times of TCP connections). Original default RED suffers from 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 incoming packets. Packet loss rates with versions of Adaptive RED in 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 10 shows the comparative performance of the queueing management algorithms in terms of average queueing delay. We experience that 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 suffers in our simulations is utilization as shown in Figure 11. Interestingly, Figure 12 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 in low average queueing delay but also the *trade-off* between delay and utilization, at least in terms of global power as defined in [12].

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 multimedia. For these application, other performance metrics are also of importance. For example, for VoIP (Voice over IP) application, not only the average delay but delay variations (jitters) heavily affect the end-user performance. So in case both TCP flows and UDP flows sharing the router, queue-length variation should be kept low for the shake of the quality of service (QoS) of voice applications. We simulate RED and Fuzzy RED with 1000 flows (500 TCP flows and 500 UDP flows). TCP flows all have the same round-trip time of 100 ms. Duration time of the simulation is 180 seconds.

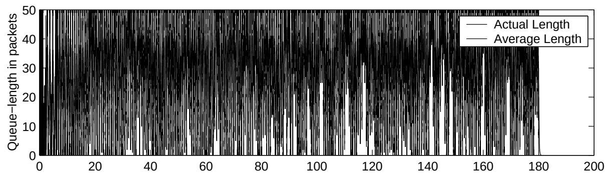

Fig. 13. Queue-length variation with RED

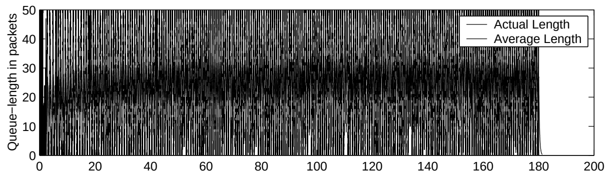

Fig. 14. Queue-length variation with Fuzzy RED

Figure 13 and Figure 14 show 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 variations (jitters), thus decreasing the quality of voice services.

A.1 Performance with Sudden Changes

To examine performance of Fuzzy RED with sudden changes, we run the simulation in three parts each with length of 10 seconds. First 10 TCP flows are active. After 10 seconds, addition 10 TCP flows enter. After 20 seconds, these 10 flows are terminated. All other parameters are the same as the simulation in stationary case.

Figure 15 and 16 show 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..50. Fuzzy RED adapt 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..35 packets. After the decrease in workload (10 flow leave), it takes around 2 seconds for both RED and Fuzzy RED to get back to normal

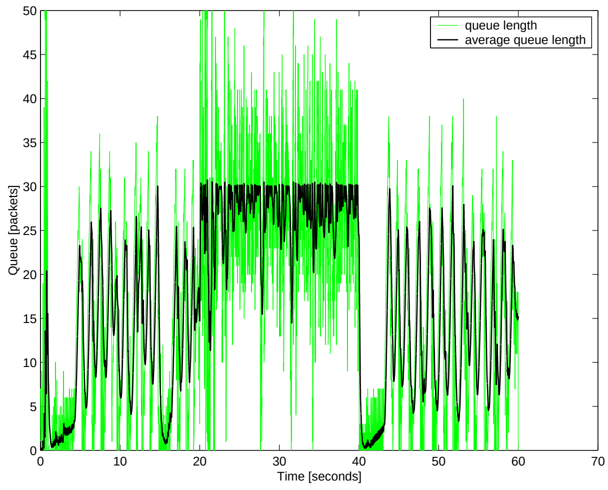

Fig. 15. RED

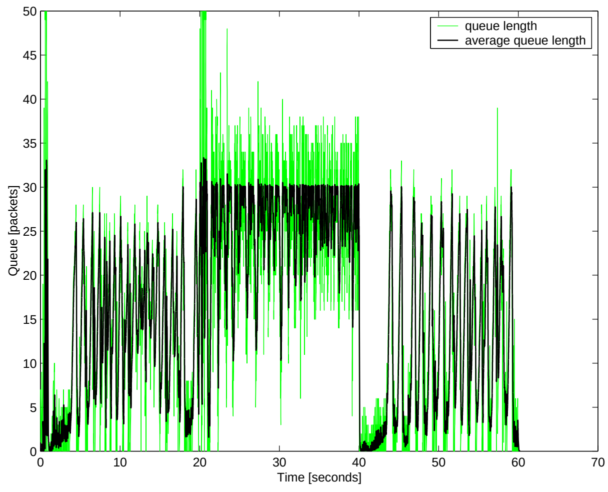

Fig. 16. Fuzzy RED

condition, but Fuzzy RED produce somewhat smaller both in average queue-length and queue-length variation.

VII. Conclusion

We have demonstrated that RED in general does not guarantee proportional loss to flows and gave proof for TCP cases. We also analytically evaluated the performance of 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 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 case of frequent changing congestion. Analytically evaluating Fuzzy RED is a subtle and difficult task, which is left as our future work.

VIII. Acknowledgement

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

References

[1] M. Yajnik, S. Moon, J. Kurose and D. Towsley, *Measurement and Modelling of the Temporal Dependence in Packet Loss* INFOCOM99, 1999.
[2] M. May, J. Bolot, C. Diot, and B. Lyles, *Reasons not to deploy RED* IWQoS'99, 1999.
[3] M. Christiansen, K Jeffay, D. Ott, F. D. Smith, *Tuning RED for Web traffic* ACM SIGCOMM'oo, Stockhom, 2000.
[4] T. Ziegler, S. Fdida, and Brandauer, *Stability Criteria for REd with TCP traffic* IEEE ICCCN'00
[5] 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.
[6] V. Firoiu and M. Borden, *A Study of Active Queue Management for Congestion Control* INFOCOM'00.
[7] D. Lin and R. Morris, *Dynamics of Random Early Detection* SIGCOMM'97
[8] T. Ott, T. Laksman, L. Gong, *SRED: Stabilized RED* INFOCOM'99.
[9] J. Aweya, M. Ouellette, D. Montuno, A. Chapman *A Control Theoretic Approach to Active Queue Management* Computer Networks, 36, 2001.
[10] C. V. Hollot, V. Misra, D. Towsley and W. Gong, *A Control Theoretic Analysis of RED* INFOCOM 2001, Alaska, April 22-26, 2001
[11] G. Vattay, A. Fekete Self-Similarity in Bottleneck Buffers Proc. of Globecom 2001.
[12] 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
[13] 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
[14] W. Feng, D. Kandlur, D. Saha, and K. Shin *A Self-Configuring RED Gateway* Infocom99, 1999.
[15] M. May, T. Bonald, and J. Bolot. *Analytic Evaluation of RED Performance* Proc. of INFOCOM'00, 2000.
[16] P. Hurley, J. Boudec, and P. Thiran, *A Note on the Fairness of Addictive Increase and Multiplicative Decrease* ITC 16, UK, 1999.
[17] 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.
[18] B. Melamed and D. Yao, *The ASTA Property* Frontiers in Queuing: Models, Methods, and Problems, CRC Press, 1995.
[19] R. Wolff, *Poisson Arrivals See Time Averages* Operations Research, vol. 30, no 2, 1982.
[20] L. Kleinrock, *Queueing Systems: Theory* John Wiley and Sons, 1975.
[21] S. Keshav, *A Control-theoretic Approach to Flow Control* Proc. ACM SIGCOMM 1991, Sept. 1991
[22] N. Hubele, I. Chang, *Adaptive Exponential Weighted Moving Average Schemes Using a Kalman Filter,* IIE Transactions, 22, 361-369.