Adaptive Tuning of RED Using On-line Simulation

Tao Ye, Shivkumar Kalyanaraman
Department of Electrical, Computer and System Engineering
Rensselaer Polytechnic Institute
Troy, New York 12180
yet3@networks.ecse.rpi.edu, shivkuma@ecse.rpi.edu

Abstract—**Random Early Detection (RED) is an active queue management mechanism designed to provide better performance than traditional DropTail. However, its parameter setting has proved to be very sensitive to network scenarios and needs constant tuning to achieve ideal performance under varying network conditions. In view of the fact that RED has not been understood well enough for an analytical approach, this paper takes advantage of network simulation techniques and formulates the optimal configuration of RED as a black-box optimization problem. An optimization objective is designed to effectively reflect the tradeoff between utilization and queueing delay. Based on the proposed RED optimization scheme, a general automatic network management system, i.e., on-line simulation system[1], has been used for the on-line tuning of RED under changing network conditions. The proposed approach is empirically validated with simulations and real network experiments. The simulation results show that RED controlled with on-line simulation system is able to stabilize around the expected equilibrium status under varying conditions and maintain high utilization.**

I. Introduction

Congestion control in the current Internet is accomplished by end-to-end congestion avoidance together with queue management mechanism. Traditional DropTail queue management could not effectively prevent the occurrence of serious congestion and often suffer from long queueing delays. Furthermore, the global synchronization may occur during the period of congestion, i.e., a large number of TCP connections experience packet drops and hence back off their sending rate at the same time, resulting in underutilization and large oscillation of queueing delay. Random Early Detection (RED) has been proposed [2] to address these problems. The basic idea of RED is to detect the inception of congestion and notify traffic sources early to avoid serious congestion. It has been demonstrated to be able to avoid global synchronization problem, maintain low average queueing delay and provide better utilization than DropTail[2]. Therefore, IETF has recommended RED as the single active queue management for wide deployment in the Internet[3]. However, the setting of RED parameters has proved to be highly sensitive to network scenarios and the performance of misconfigured RED may suffer significantly [4], [5], [6]. In addition, since network is a dynamic system, RED needs constant tuning to adapt to current network conditions. In view of this, it has been debated whether or not RED can achieve its claimed advantages[6], [7], [8].

This paper attempts to address the following questions:

- Given a network scenario, how to optimally configure RED parameters?
- Given varying network conditions, how to dynamically tune RED to the optimal setting?
- When optimally configured, can RED effectively control congestion and achieve its design objective?

Currently the interaction between RED and TCP is not yet clearly understood. Based on simplified models, some general guidelines for setting RED parameters have been proposed[2], [5], [9]. Intuitive modifications on RED have also been proposed to automate the tuning of RED under varying network conditions by adjusting one of the parameters[4], [10]. However, the effectiveness of these methods in complex network scenarios is still under investigation. Rather than relying on simplified models or intuition, this paper exploits the advantage of network simulation technique and formulates the optimal configuration of RED as a black-box optimization problem. In this approach, RED is considered as a black-box and network simulation is used to evaluate its performance in a specific parameter setting. With this empirical mapping between performance metric and parameter setting, a black-box optimization algorithm, such as genetic algorithm, can be employed to obtain the optimal RED setting.

This paper adopts a general network management scheme, i.e., on-line simulation system[1], to adjust RED configuration to varying network conditions. The on-line simulation scheme monitors network conditions and constantly tunes RED with the proposed RED optimization scheme when network conditions change. The underlying assumption of the on-line tuning scheme is that network conditions are *quasi-stationary*, i.e., they do not change too fast. Note that the on-line simulation system does not attempt to change network protocols in any way but only applies *"second-order" control* over network by tuning network protocol parameters. As a result, it is highly flexible and can be applied to any network protocol.

The paper first analyzes the mechanism of RED and then proposes an appropriate performance metric to effectively reflect the tradeoff between utilization and queueing delay. The features of the RED optimization problem are also examined and an efficient black-box optimization algorithm is selected accordingly. Simulations and real network experiments have been performed to validate the proposed approach. The results demonstrate that RED is capable of effectively controlling congestion and achieving its design objective when dynamically tuned.

The rest of the paper is organized as follows: Section II analyzes RED mechanism and the sensitivity of its parameters to

network conditions, and proposes a performance metric for use in RED optimization. Section III empirically validates the online tuning approach with simulations and experiments in real network. Section IV concludes this paper and presents further research directions.

II. Formulation of RED Optimization Problem

A. Parameter Sensitivity of RED

RED uses the average queue size $\bar{q}$ as an indicator of the congestion extent and determines the packet drop rate accordingly. Fig 1 illustrates the working mechanism of RED. As shown in

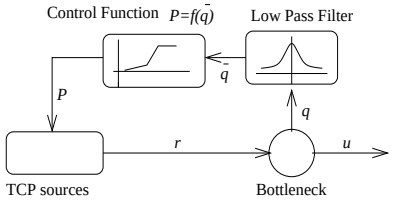

Fig. 1. RED working mechanism

the figure, the instantaneous queue size q is sampled at every packet arrival and then passed through a low-pass filter to remove transient noises. Based on the smoothed average queue size $\bar{q}$, the drop probability P is calculated with a control function $P = f(\bar{q})$. The arrived packets are randomly dropped (or marked) according to this probability P. Traffic sources react to these drops and adjust offered load r accordingly. Therefore, RED is mainly designed to work with TCP traffic sources which are responsive to packet drops and it will not work well in the cases like UDP traffic or short-life HTTP traffic.

A queue will build up and keep increasing if the offered load is larger than the bottleneck capacity; therefore, the objective of a queue management is to stabilize the offered load around the bottleneck capacity. Basically, TCP sources increase their sending rate every round trip time; on the other hand, the packet drops cause TCP sources to lower their sending rates. In the equilibrium status, the increase rate of TCP traffic should be approximately equal to its decrease rate caused by packet drops and thus the offered load will stabilize around a certain level. If this equilibrium status is achieved while maintaining a certain queue size, the link utilization will be close to 1, i.e., the offered load will stabilize around the bottleneck capacity. The rationale of RED is to search for an appropriate packet drop rate by varying the average queue size to counteract the increase of offered load.

There are four parameters in RED. Among them, the moving average weight w_q determines the cut-off frequency of the low-pass filter, and the other three parameters, i.e., minimum threshold min_{th}, maximum threshold max_{th} and maximum drop probability max_p, determine the control function $P = f(\bar{q})$. In the standard version of RED, the control function is determined by the parameters as illustrated in Fig 2. With this function, the drop probability can be calculated according to the

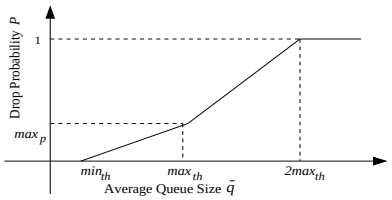

Fig. 2. RED control function $P = f(\bar{q})$

average queue size. The equilibrium drop probability depends on two factors, the offered load increase rate and the granularity of congestion notification, i.e., the load decrement caused by one packet drop. With TCP fast recovery and fast retransmission mechanism, each drop will cause a TCP source to decrease its sending rate by half. Therefore, the granularity of the congestion notification is determined by the average TCP sending rate. When the average sending rate is large, for example, a small number of TCPs share a bottleneck, each packet drop will cause a large decrease in offered load, and *vice versa*. In different scenarios, the increase rate of offered load is also different. For example, the increase rate will be large when there are many TCP flows or the round trip time is short. As a result, the drop probability should be adjusted according to network scenarios to maintain a stable equilibrium point. If the control function remains unchanged, the average queue size has to be varied to obtain the new equilibrium drop probability. Therefore, to keep the average queue size stable around a certain level in varying conditions, the control function has to be adjusted accordingly, i.e., the three parameter which determines $f(\bar{q})$ should be dynamically tuned.

w_q controls the cut-off frequency of the low-pass filter. The cut-off frequency should be high enough to detect manageable traffic variations, while low enough to filter out transient traffic oscillations which can not be effectively controlled by RED. For example, the oscillation within one round trip time rtt should be removed. Therefore, the optimal w_q is usually related to rtt. In addition, since the average queue size is calculated at every packet arrival instead of a constant interval, different link speeds will result in different packet arrival intervals and hence affect the cut-off frequency of the low-pass filter. Consequently, the optimal w_q is also dependent on the link speed.

B. Optimization Objective

For a queue management mechanism, there are basically two performance metrics, i.e., link utilization and average queue size. The main objective of RED is to *maintain a high utilization while keeping a low average queue size*[2]. However, optimizing one of the performance metrics may compromise the other. For example, a high link utilization can always be obtained by increasing min_{th} or decreasing max_p, hence virtually increasing the average queue size. On the other hand, a low average queue size can be obtained by decreasing max_{th}

or increasing max_p. However, this obviously will cause underutilization of the link. Therefore, an appropriate tradeoff has to be made to reflect the requirement of network operators. This is essentially a multi-objective optimization problem and corresponding techniques should be employed to convert it into a tractable single objective problem.

One classic multi-objective optimization technique is to optimize the weighted average of the performance metrics. The weights for different metrics reflect the quantitative tradeoff among them and are essential to the effectiveness of optimization results. However, the weights are normally difficult to determine. Another common technique is to define the lower limits for less significant metrics, and only optimize the most important one with the restriction that the other metrics are not below their limits. In this paper, instead of using traditional multi-objective optimization techniques to directly work on link utilization and queueing delay, we have proposed a performance metric whose optimization will cause RED to settle in a equilibrium status and hence achieve high utilization and low queueing delay.

As mentioned above, in the equilibrium status, the average queue size of RED stabilizes around a certain level. When traffic pattern changes, the equilibrium point may also shift which makes the average queue size move around. This is an undesired behavior since end users normally expect a predictable delay and constant changes in delay are unacceptable for delay-sensitive applications. Furthermore, when the average queue size drifts beyond the control of RED, RED will become unstable, i.e., the queue status oscillates between full and empty[4], [5]. This not only causes end users to experience significant delay jitters, but also results in link underutilization. Therefore, it is important to keep the average queue size of RED stable at a target level, such as the middle between min_{th} and max_{th} as proposed in[10]. In consideration of this, we define the performance metric to be optimized as:

$$m = \frac{\sum_{i=1}^{N}(\bar{q_i} - q_0)^2}{N} \tag{1}$$

where q_0 is the expected average queue size predefined by network operators, $\bar{q}_i$ is the periodic sample of the average queue size and N is the number of samples. This metric essentially calculates the variance of the average queue size relative to q_0 over a certain period of time. When the equilibrium level of RED is far from the expected level, m will be large. Or when RED is misconfigured and hence the equilibrium cannot be reached, the queue size will oscillate greatly, also resulting in a large m. Therefore, minimizing m will cause RED to avoid both situations and always maintain an equilibrium around q_0. Thus, high link utilization and stable queueing delay can both be achieved.

C. Choice of Optimization Algorithm

Based on the above analysis, the optimization of RED can be formulated as: given a parameter space D specifying the ranges of parameters, minimize the following objective function

$$m = f(w_q, min_{th}, max_{th}, max_p) \tag{2}$$

Here, m is the performance metric to be optimized, i.e., the variance of RED queue size relative to the expected level. $f(\cdot)$ is a scalar function mapping a set of RED parameters to the performance metric. $f(\cdot)$ is determined by specific network scenario and is analytically unknown, which is the basic feature of black-box optimization. For a set of RED parameters, the value of m can be empirically evaluated with network simulation based on Equation (1).

Due to the strong demand in various areas of science and engineering, a large number of black-box optimization algorithms have been proposed, such as genetic algorithm, simulated annealing and tabu search. These algorithms have met with much success, but no one seems to consistently outperform the others. In fact, the so-called *No Free Lunch Theorem*[11] has theoretically demonstrated that any optimization algorithm can only achieve the same average performance over all classes of problems. However, for one specific class of problems, it is possible for one optimization algorithm to outperform the others when its search techniques match properties of the underlying problem.

The desired optimization algorithm for on-line tuning of RED is required to have the following characteristics:
High efficiency , i.e., quickly find better parameters with a minimum number of network simulations. This is because network simulation is often time-consuming. As a result, the emphasis of the algorithm should not be on seeking the strictly global optimum, but on finding a better solution before current network conditions change.
Scalability to high-dimensional problems This is because of the existence of a large number of protocol parameters to be tuned in a network.
Robustness to noise Network simulation only provides an approximate evaluation of the objective function, and furthermore, inaccuracies in network modeling may also introduce noises into the objective function. For example, Fig 3 shows an empirical objective function obtained with network simulations, where the performance metric, average drop rate of RED, is plotted as a function of two parameters, w_q and max_p. The

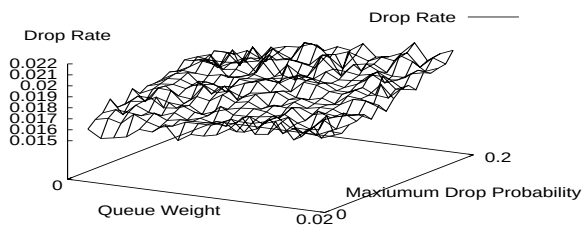

Fig. 3. Shape of empirical objective function obtained through network simulation

figure demonstrates that there are many small oscillations imposed on the overall structure of the objective function. Note that using packet drop rate as the performance metric is only for the purpose of illustrating the property, and any other metric should expect similar effects.

Taking the above features into account, a Recursive Random Search algorithm (RRS) has been proposed in [12] for the efficient optimization of network protocols. The RRS is based on the high-efficiency feature of random sampling at initial steps. The idea is to use the first part of high-efficiency random samples to identify promising areas and then start recursive random sampling processes in these areas which shrink and re-align the sample space to local optima. Readers can refer to [12] for more details. RRS will be used in this paper for the tuning of RED. Simulations have proved that the algorithm is able to generate good solutions very quickly.

III. Test Results

A. Simulation for Optimization of Single RED

Extensive simulations have been performed to validate the effectiveness of the proposed approach. Due to the space limit, the paper only presents two of these simulations which deal with varying traffic load and round trip time, two major factors affecting RED performance.

The network topology used in the simulations is shown in Fig 4. We have adopted the widely used ns[13] as the simulation tool. Infinite FTP traffic between TCP sources and sinks is used to build up a queue at router $r1$. RED is configured on $r1$ to manage a 100-packet buffer. Each simulation runs for 80 seconds and at half of the simulation time, network conditions are changed. We will compare the performance of standard RED and RED controlled with the on-line simulation scheme.

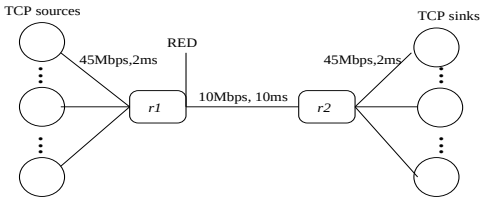

Fig. 4. Network topology for simulation

We define an expected average queue size of 30 packets and the objective is to maintain the equilibrium status of RED around this level. According to the common guideline of RED parameter setting, we use $min_{th} = 15, max_{th} = 45, max_p = 0.1, w_q = 0.002$ for standard RED. We also assume that the on-line simulation scheme can promptly detect the change in network conditions and trigger the optimization process of RED parameters. In reality, this can be achieved by monitoring the change in performance metrics or analyzing traffic statistics directly.

First we test the tuning of RED to varying traffic load. The simulation starts with 16 TCP flows and then increases the offered load to 64 flows after around 20 seconds. The instantaneous queue sizes of standard RED and RED with on-line simulation control are shown in Fig 5.

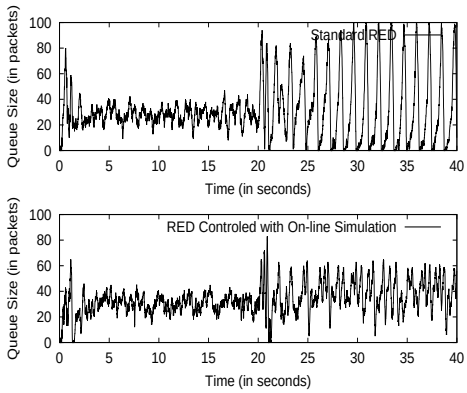

Fig. 5. Comparison of standard RED (upper graph) and RED controlled by on-line simulation (lower graph) under varying traffic load

Then we test the tuning of RED to varying round trip time. The simulation starts with 16 TCP flows and each with a round trip time of 28ms (not including queueing delay). In the middle of the simulation, these flows are gradually replaced by the ones with a rtt of 180ms. The instantaneous queue sizes of standard RED and RED with on-line simulation control are shown in Fig 6.

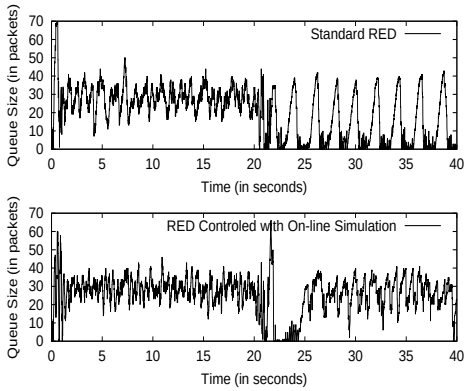

Fig. 6. Comparison of standard RED (upper graph) and RED controlled by on-line simulation (lower graph) under varying round trip time

The above simulations indicate that when dynamically tuned with the proposed approach, RED is able to maintain an equilibrium status in changing network conditions. The queue size is very stable and the utilization is close to 100%. For standard RED, when the configuration does not match for network conditions, the equilibrium status might not be reached and large oscillations in the queue size could happen as shown in the figures. This also causes underutilization. For example, in the simulations, the average utilization of standard RED can only reach around 93% or lower.

B. Real Network Experiment for Optimization of Multiple RED Queues

The effectiveness of our approach is also tested with experiments in real network situations. This section presents one such experiment. A Linux-based testbed shown in Fig 7 is used and ns is adopted for network simulation in the on-line simulation system. There are 4 Linux routers in the network and

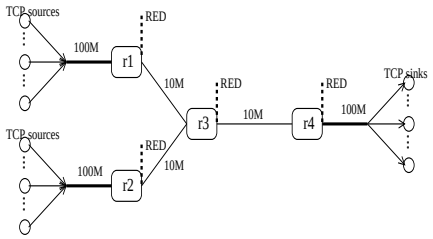

Fig. 7. Linux-based testbed topology with multiple RED queues

each of them is configured with a RED queue which is monitored and controlled by the on-line simulation system through SNMP. Again, infinite FTP sources are used to generate network traffic. Note that in this test we will try to tune the parameters for all four RED concurrently. Since optimizing each RED individually may compromise the performance of the others, we have taken all RED queues as a single black-box system with a total of 16 parameters. Consequently, a global performance metric has to be defined based on the objective of network operators. If using ISP-based metrics, such as utilization and queueing delay, a certain multi-objective technique has to be employed to combine the metrics from every RED router. Instead, we have selected an end user performance metric, i.e., the coefficient of variation ($\frac{\sigma}{\mu}$) of goodputs for TCP connections, which measures the variability of TCP goodputs. This choice is somewhat arbitrary, only to demonstrate the effectiveness of our approach. In addition, choosing such a metric is also to demonstrate the flexibility of the approach, i.e., rather than being restricted to a few metrics like utilization and delay, RED can be tuned according to any performance metric defined by network operators though the mechanism of how RED affects this performance metric may be completely unknown.

During the experiment, a number of TCP flows are generated from one side to the other. The goodputs of these TCP flows are collected periodically from TCP sinks. The COV of the goodputs is calculated and plotted as a function of time as shown in Fig 8. In the beginning, the parameters of these RED queues are set to random values to present a misconfigured system, which results in a large unfairness between TCP flows, i.e., a high average COV value and large oscillations. At 325 second, the on-line simulator starts and detects the misconfiguration of REDs. Soon the good configuration with a performance better than a predefined threshold is found and the network is reconfigured. This results in an immediate performance improvement as shown in the plot: the average of COV drops to a very low value and the instantaneous COV curve becomes stable over time.

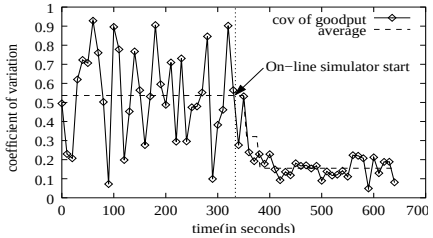

Fig. 8. Tuning multiple RED queues for optimizing coefficient of variation of goodputs

IV. Conclusion

To achieve the performance advantages claimed in RED, the parameters of RED have to be dynamically configured according to current network conditions. This paper formulates the optimal configuration of RED as a black-box optimization problem and then uses an automatic network management scheme proposed in [1] to perform on-line tuning of RED. Appropriate formulation of an optimization problem needs insight into the underlying problem. The control mechanism of RED and the sensitivity of its parameters are investigated. Based on the analysis, a performance metric is established to best reflect the tradeoff between the two major optimization objectives of utilization and queueing delay. Simulation results have demonstrated that when managed by the proposed approach, RED can effectively control congestion under varying network conditions and achieve its design objective.

The approach for RED tuning presented in this paper is a highly flexible technique. It can be easily applied to other network protocols, such as OSPF and BGP. Its flexibility also lies in the fact that it can be easily adapted to achieve various optimization objectives, such as optimzing RED for minimum packet loss. This black-box approach is especially advantageous in the cases where the mechanism of how the underlying network protocol affects the concerned performance metric is not well understood. The success in applying the proposed scheme to real network relies on two key factors: the sensitivity of network protocol parameters to network conditions, and the accuracy of network monitoring and modeling which is a very active subject in current networking research.

Optimization techniques can also be used for the study of network protocols. The obtained empirical knowledge can help with better understanding of network protocols. For example, we can optimize RED for different network scenarios and obtain the correlations between RED parameters and network conditions, such as the correlation between w_q and round trip time. Further investigation of RED with optimization techniques is one of our future research directions.

References

[1] Tao Ye and et al. Traffic management and network control using collaborative on-line simulation. In *Proc. of IEEE ICC'01*, Helsinki, Finland, 2001.

[2] S. Floyd and V. Jacobson. Random early detection gateways for congetsion avoidance. *IEEE/ACM Transactions on Networking*, 1:397–413, August 1993.

[3] B. Braden and et al. Recommendations on queue management and congestion avoidance in the internet. RFC 2309, 1998.

[4] Wu chang Feng, Dilip D. Kandlur, Debanjan Saha, and Kang G. Shi. A self-configuring RED gateway. *Proceedings of IEEE Infocom 1999*, March 1999.

[5] Victor Firoiu and Marty Borden. A study of active queue management for congestion control. In *INFOCOM (3)*, pages 1435–1444, 2000.

[6] M. May, J. Bolot, C. Diot, and B. Lyles. Reasons not to deploy RED. Technical report, INRIA Sophia-Antipolis, France, 1999.

[7] M. Christiansen, K. Jeffay, D. Ott, and F.D. Smith. Tuning RED for web traffic. In *Proceeding of ACM SIGCOMM*, 2000.

[8] Thomas Bonald, Martin May, and Jean-Chrysostome Bolot. Analytic evaluation of RED performance. *IEEE Infocom 2000*, pages 1415–1444, 2000.

[9] C.V. Hollot, Vishal Misra, Don Towsley, and Wei-Bo Gong. A control theoretic analysis of red. In *Proceedings of IEEE Infocom 2001*, 2001.

[10] Sally Floyd, Ramakrishna Gummadi, and Scott Shenker. Adaptive RED: An algorithm for increasing the robustness of RED's active queue management. unpubished, 2001.

[11] D. h. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transaction on Evolutionary Computing*, 1:67–82, 1997.

[12] Tao Ye and Shivkumar Kalyanaraman. A recursive random search for optimizing network protocol parameters. Technical report, ECSE Department, Rensslaer Polytechnique Institute, Dec 2001.

[13] NS. network simulator. http://www-mash.cs.berkeley.edu/ns.