The mechanism of adapting RED parameters to TCP traffic

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Citation: CHEN, W. and YANG, S., 2009. The mechanism of adapting RED parameters to TCP traffic. Computer Communications, 32 (13-14), pp.1525-1530.

Additional Information:

• This article was published in the journal Computer Communications [© Elsevier] and the definitive version is available at: www.elsevier.com/locate/comcom

Metadata Record: https://dspace.lboro.ac.uk/2134/5552

Version: Accepted for publication

Publisher: © Elsevier

Please cite the published version.
This item was submitted to Loughborough’s Institutional Repository (https://dspace.lboro.ac.uk/) by the author and is made available under the following Creative Commons Licence conditions.

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/
The Mechanism of Adapting RED Parameters to TCP Traffic

Wu CHEN, Shuang-Hua YANG
Department of Computer Science
Loughborough University, Loughborough, LE11 3TU, UK
Emails: {w.chen2, s.h.yang}@lboro.ac.uk

Abstract— Random Early Detection (RED) can stabilize the queue within a given target range and thus achieve high throughput in the routers. However, the average queue size is quite sensitive to network scenarios and it is difficult to adapt RED parameters to changing network traffic. In this paper we use a previously developed dynamic model of TCP behavior and linear feedback model of TCP/RED to analyze and design a mechanism for RED parameter tuning in response to changing network conditions like traffic load, link capacity and round-trip time. Even though the values of four key RED parameters are determined by varying network conditions, they can be tuned independently without consideration of the interactions among these RED parameters. Simulation results show that this mechanism can stabilize the queue and maintain high link utilization in a wide variety of network conditions.

Keywords-RED; Congestion Control; Parameter tuning; Control theory

I. INTRODUCTION

Congestion control plays an important role in ensuring good network performance of the Internet. Internet congestion control comprises two main parts: TCP and Active Queue Management (AQM). Among many AQM algorithms proposed over the last decade, RED has been widely implemented in major commercial routers, especially in edge routers [1][2]. It is seldom deployed in core routers because they do not typically run congested links [3]. One of the main goals of RED is to stabilize the queue size within a given target while maintaining high link utilization [4]. The average queue size varies with traffic load as well as round-trip time and link capacity, and consequently parameterizing RED to obtain good performance under variable congestion scenarios can be very difficult. Such difficulties discourage network administrators from activating RED in their routers [5].

Certainly, there are many parameter tuning techniques for RED proposed in the literature [6]-[10], but they were either developed on the basis of empirical investigations and analysis, or are only applicable under certain assumptions. In particular, Adaptive RED (ARED) [6] provided a simple guideline for adjusting only one of the RED parameters in response to the changing traffic load. So it is applicable only under a narrow range of round-trip times and link capacities. As demonstrated in other AQM or AQM-based approaches, such as the Proportional Integrator (PI) controller [9], Random Exponential Marking (REM) [11] and BLUE [12], parameter setting still remains a critical unsolved problem. In addition, the Web performance of these AQMs is not as good as ARED when dropped packets are used as indications of congestion [13], and their transient response would become much slower when network scenarios change dynamically [14].

In [10] the authors proposed a method to set initial RED parameters from a Control Theoretic perspective. However, they did not illustrate how to adapt these RED parameters to changing network scenarios. By improving their Control Theoretic analysis of TCP/RED systems, we have developed an Auto-Parameterization RED (AP-RED) to provide a simple, scalable and systematic algorithm for parameter tuning as a function of network traffic conditions such as link capacity, round-trip time, and the number of TCP flows. Theoretic analysis and nonlinear simulations using a ns-2 simulator [15] have demonstrated that it is robust, adaptive to TCP dynamics, and produces desirable transient performance.

The rest of the paper is organized as follows. Section II illustrates a pre-developed nonlinear dynamic model of TCP and a linear feedback model of TCP/RED system. Section III describes AP-RED and presents stability analysis. Section IV introduces the calculation method for network parameters. Simulation results to validate the algorithm are presented in Section V. Finally, we conclude in Section VI.

II. MODEL

In [16], a nonlinear dynamic model of TCP behavior was developed using fluid-flow and stochastic differential equation analysis. By ignoring timeout mechanism, [10] uses a simplified version to describe the model by the following differential equations:

\[ W(t) = \frac{1}{R(t)} - \frac{W(t)}{2R(t)} p(t - R(t)) \]

\[ q(t) = \frac{W(t)}{R(t)} N(t) - C \]  

(1)
where \( \dot{x} \) denotes the time-derivative of \( x \) and
\[
W = \text{expected TCP window size (packets)};
q = \text{expected queue size (packets)};
R = \text{round-trip time (seconds)};
C = \text{Link capacity (packets/second)};
N = \text{Number of TCP flows};
p = \text{probability of packet mark/drop};
\]

According to these equations, [10] further describes the behavior of \( \delta W \), \( \delta q \), and \( \delta p \) in a linear feedback control model of TCP/RED by linearizing variables \( W, q, p \) at its equilibrium point \( (W_0, q_0, p_0) \). The linear control model is depicted in Fig. 1.

![Fig. 1. Feedback control model of TCP/RED system](image)

In the above model \( P_{tcp}(s) \) denotes the linearized TCP dynamics, \( P_{queue}(s) \) denotes the queue dynamics, \( e^{-\alpha s} \) denotes the delay term and \( C_{red}(s) \) denotes RED control strategy. They were given by the following equations:

\[
P_{tcp}(s) = \frac{RC^2}{s + \frac{2N}{R^2C}};
\]

\[
P_{queue}(s) = \frac{N}{s + \frac{1}{R}}
\]

\[
C_{red}(s) = \frac{L_{red}}{s + K + 1}
\]

where

\[
L_{red} = \frac{p_{max}}{\max_a - \min_a};
K = -\log_e(1 - \alpha) / T_s;
\]

\[
p_{max} = \text{Maximum drop probability};
\max_a = \text{Maximum threshold};
\min_a = \text{Minimum threshold};
\alpha = \text{Average queue weight};
T_s = \text{Sampling interval}.
\]

III. AP-RED ALGORITHM AND STABILITY ANALYSIS

A. The Algorithm

The objective of AP-RED is to maintain the average queue size within the limits of \( \max_a \) and \( \min_a \), and to provide high link utilization in widely varying network scenarios. The rationale behind setting \( \max_a \) and \( \min_a \) as a fraction of bandwidth-delay product is to follow the theoretical analysis and rule-of-thumb in accommodating bursty traffic [6][7][17].

The rational behind setting \( \alpha \) is to achieve a desirable equilibrium point based on theoretical analysis. The rationale behind setting \( \alpha \) is to maintain the stability of the system from a Control Theoretic perspective.

Thus, consider the initial RED parameters \( \alpha_0, p_{max0}, \max_a, \) and \( \min_a \) that stabilize the queue under the initial network scenario of TCP load \( N_0 \), round-trip time \( R_0 \) and link capacity \( C_0 \). When the network scenario varies we present the following algorithm for adjusting RED parameters.

\[
\max_a = k_a \max_a; \quad \min_a = k_a \min_a; \quad p_{max} = \left( \frac{k_a}{k_c} \right)^2 p_{max_0}
\]

\[
\alpha = \begin{cases} 
\frac{1}{k_c} \alpha_0 & N \leq RC / 2 \\
\frac{k_c}{(k_c)^2} \alpha_0 & N > RC / 2 
\end{cases}
\]

where

\[
k_a = \frac{R}{R_0};
\]

\[
k_c = \frac{C}{C_0};
\]

\[
k = \frac{N}{N_0};
\]

and constraint \( p_{max} \) within the range \([0.01, 0.5]\)

i.e. if \( p_{max} > 0.5 \) \( p_{max} = 0.5 \)

if \( p_{max} < 0.01 \) \( p_{max} = 0.01 \)

Compared with the automatic setting of queue weight \( \alpha \) based on link capacity in [6], our tuning algorithm of \( \alpha \) is based not only on link capacity, but also on round-trip time and traffic load. As \( \alpha \) is adapted to congestion scenarios, a larger \( \alpha \) can improve transient response while a smaller \( \alpha \) provides a sufficient stability margin [10]. When \( \max_a \) and \( \min_a \) are reduced according to the changing network parameters, the queuing delay is decreased correspondingly. In contrast a larger \( \max_a \) and \( \min_a \) can still stabilize the queue within a given target range when round-trip time increases or link capacity increases.
B. Determining the Equilibrium Point

The purpose of adjusting \( p_{\text{max}} \) is to keep the queue at a desirable equilibrium point. At a new equilibrium point where \( W = 0 \) and \( \dot{q} = 0 \), from (1) we have

\[
W' - W = \frac{RC}{N}
\]

Then we obtain

\[
p = \frac{q - \min_a - \min_a}{\max_a - \min_a} p_{\text{max}} = 2\left(\frac{N}{RC}\right)^2 \tag{7}
\]

Hence at the equilibrium point the drop probability is determined by network scenario parameters rather than other RED parameters.

Similarly, at the initial equilibrium point \( q_0 \) we obtain

\[
p_{\text{eq}} = \frac{q_0 - \min_a - \min_a}{\max_a - \min_a} p_{\text{max}} = 2\left(\frac{N}{RC}\right)^2 \tag{8}
\]

Since we tune \( p_{\text{max}} \) in terms of (5), from (7) and (8) we have

\[
q = k, k, q_a
\]

Thus the equilibrium point \( q \) moves in proportion to the max, and \( \min_a \). It still stays between max, and \( \min_a \) when network condition changes.

C. Stability Analysis

1) Stability Proposition

In this section we derive a simplified version of stability proposition based on the analysis given in [10].

**Stability Proposition:** Let \( L_{\text{rad}} \) and \( \alpha \) satisfy:

\[
L_{\text{rad}}(RC)^3 \leq \frac{0.8N^3}{(2N)^2} \quad N \leq RC / 2
\]

\[
L_{\text{rad}}(RC)^3 \leq \frac{0.4N^2}{(2N)^2} \quad N > RC / 2
\]

where

\[
\alpha \ll 1
\]

Then the linear feedback control system in Fig. 1 is stable.

**Proof:**

Consider the frequency response of the compensated loop transfer function

\[
|L(j\omega)| = \frac{L_{\text{rad}}(RC)^3 (2N)^2 e^{-j\omega}}{(j\omega)^2 + \frac{1}{RC} + 1}
\]

where

\[
\omega_x = 0.1 \min\left[\frac{2N}{RC}, \frac{1}{RC} \right]
\]

Thus we obtain

\[
|L(j\omega)| \leq \frac{\omega_x^2}{\omega_x^2 + 1} \leq 1
\]

where

\[
L_{\text{rad}}(RC)^3 \leq \frac{\omega_x^2}{K^2 + 1}
\]

We again use (10) to obtain

\[
\angle|L(j\omega)| \approx \angle \frac{j\omega_x}{\omega_x + 1} - \omega_x R \geq -90^\circ - 0.1 \frac{180^\circ}{\pi} > -180^\circ
\]

This and (12) indicate that the closed-loop system is asymptotically stable according to the Nyquist stability criterion [18].

Given \( \alpha \ll 1 \), we have

\[
L_{\text{rad}}(1 - \alpha) \approx -\alpha
\]

For a stable congested queue we have

\[
T_s = \frac{1}{C}
\]

Thus, from (1), (14) and (15) we have

\[
K = \alpha C
\]

Consequently, given any \( N \leq RC / 2 \), from (13) we obtain

\[
\frac{L_{\text{rad}}(RC)^3}{(2N)^2} \leq \frac{0.2N}{\alpha C}
\]

where

\[
L_{\text{rad}} \leq \frac{0.8N^3}{(2N)^2}
\]

Similarly, given any \( N > RC / 2 \), we obtain

\[
L_{\text{rad}} \leq \frac{0.4N^2}{(2N)^2}
\]

This completes the proof.

2) Stability analysis

For initial RED parameters \( \alpha_0 \), \( p_{\text{max}} \), \( \max_a \), \( \min_a \) that satisfy (9), given \( N_0 > R_0 C_0 / 2 \), we have

\[
\frac{p_{\text{max}}}{\max_a - \min_a} \leq \alpha_0 \leq \frac{0.4N_0^2}{(C_0)^4} < \frac{0.8N_0^3}{(C_0)^3}
\]

and given \( N_0 \leq R_0 C_0 / 2 \), we still have
for ARED the same as the corresponding methods described in the literature to obtain the network and the IPMON system [20]. It is feasible for us to use the loads. Although there is some difference for the calculated hand we can use a shorter time interval to calculate the traffic observed, and the flow ends when no packets with the same 5-tuple information is first time when a packet with new 5-tuple information is classified as the same flow. The start time of a flow is the source port, destination address, destination port, and protocol)

\[
\frac{p_{\text{max},a}}{\max_{a_0} \alpha_a} \leq \left(\frac{0.8N_t}{(R,C_t)}\right)^{1/3} \\
\frac{L_{\text{red}}}{\alpha} \leq \frac{p_{\text{max},a}}{\max_{a_0} \alpha_a} \leq \left(\frac{0.8N_t}{(R,C_t)}\right)^{1/3} < \frac{0.4N_t(RC)^{3/8}}{(RC)^{3/8}}
\]

Similarly, Given \( N > RC/2 \) we have

So the stability proposition in (9) can always be satisfied when we tune RED parameters according to equations (3)-(6). This provides a useful guideline for tuning RED parameters whilst maintaining the stability.

The derivation of our stability proposition is similar to that proposed in [10]. So we have an intuitive sense that our algorithm can provide a similar performance to that in [10]. However, the RED parameters in [10] are tightly coupled to each other and a complex calculation is needed to obtain suitable RED parameters for every changed network scenario. In contrast RED parameter tuning becomes much easier with the algorithm as shown in (3)-(6).

IV. NETWORK TRAFFIC MEASUREMENT

The online statistic characteristics of heterogeneous network scenarios used in AP-RED include round-trip time \( R \), number of TCP flows \( N \) and link capacity \( C \). Based on packet-level traces and flow-level statistics we know they are measurable and do not change dramatically over a relatively short time. For example, the characteristics of \( R, N, C \) and average package size etc have been observed in OC3MON [19] and the IPMON system [20]. It is feasible for us to use the methods described in the literature to obtain the network parameters.

In [20] the number of flows is calculated per minute. Those packets with the same 5-tuple information (source address, source port, destination address, destination port, and protocol) are classified as the same flow. The start time of a flow is the first time when a packet with new 5-tuple information is observed, and the flow ends when no packets with the same 5-tuple information are seen for a time interval 60s. On the other hand we can use a shorter time interval to calculate the traffic loads. Although there is some difference for the calculated value of traffic loads with different time intervals, in terms of [21] the variation of proportionality factor \( k_a \) is small with the new five.

In [20][22] round-trip time is measured as the time elapsed between a SYN packet and the first ACK packet that completes the three-way handshake. This is a rough estimate because we only compute the round-trip times for flows of which we observe the SYN/ACK pair. Fortunately, the deviation of the estimate is minimized because we use a proportionality factor \( k_a \). In addition, RED also has its own robustness under changed scenarios [23].

In [24] link capacity can be sampled by calculating the ratio of departed TCP packets to the router’s busy time. Then this capacity is smoothed by a low pass filter.

The measurement methods can be deployed in routers. For example, Cisco NetFlow is configured on most Cisco routers to provide a highly condensed and detailed view of all real time network traffic [25]. Because the statistic characteristics of network parameters are not changing dramatically we do not need to predict the network scenario parameters \( R, N \) and \( C \) in advance and could measure these parameters less frequently. In practice we need to set a suitable interval time to obtain these network parameters on a trial-and-error basis. Because it is easy to obtain these network parameters in edge routers, the feasibility of our RED tuning algorithm is ensured, especially in edge routers.

V. SIMULATIONS

Although AP-RED is based on the analysis of a linearized model, we verify the algorithm by using a ns-2 simulator to capture the stochastic, nonlinear nature of the network dynamic. As the prominent algorithm for RED parameter tuning [13], ARED is compared with AP-RED in the simulations. These simulations use a simple dumbbell topology with a single bottleneck link shared by persistent FTP flows, short HTTP sessions and reverse-path traffic. Each HTTP sessions repeatedly make short file transfers. Between two consecutive transfers, there is a ‘think time’ that starts after the last byte of the first file has been acknowledged. The transfer size is exponentially distributed with a mean of twelve 1 KB-packets. The think time is exponentially distributed with a mean of 500ms. The presence of these short-lived flows introduces noise into the queue. The number of HTTP flows in our simulations is set to twice that of FTP flows. The presence of reverse-path traffic introduces ack (acknowledgement) compression and the loss of ack packets, and thus increases the burstiness of the forward-path traffic. In all simulations the reverse-path traffic consists of 20 persistent FTP flows. The average packet size is set to 500 bytes, and the buffer size is set to 500 packets.

A. Experiment 1

In the first experiment we have \( N_a = 50 \) FTP flows with round-trip propagation delay \( T_{\text{prop}} = 120 \) ms. The link capacity is set to 10 Mbits/second, that is, \( C_a = 2500 \) packets/second. The corresponding parameters of AP-RED are \( p_{\text{max},a} = 0.05 \), \( \min_{a_0} = 50 \) packets, \( \max_{a_0} = 150 \) packets and \( \alpha_a = 10^{-4} \), which satisfy the stability proposition (9). We choose \( \min_{a_0} \), \( \max_{a_0} \) and \( \alpha_a \) for ARED the same as the corresponding parameters for AP-RED above, and keep these ARED parameters unchanged in the following experiments. Fig. 2(a) plots the instantaneous queue size of ARED. Observe a stable queue with small fluctuations around the equilibrium point of 100 packets. Fig. 2(b) shows that the instantaneous queue size of AP-RED is also stable. Since both ARED and AP-RED are
essentially RED, we observe quite similar performance of their queues in Fig. 2.

![Fig. 2. Comparison of ARED and AP-RED under network scenarios: N= 50, C= 2500 packets/second and T_p = 120ms.](image)

(a) ARED  
(b) AP-RED

**B. Experiment 2**

In the second experiment the round-trip propagation delay decreases to T_p = 100 ms and the link capacity decreases to C=1250 packets/second. The number of FTP flows decreases to N=30. The plot in Fig. 3(a) shows that the queue oscillations of ARED become smaller than that in Fig. 2(a). This is because RED becomes more stable when round-trip time or link capacity decreases [23]. When we use T_p / T_p to approximate k_e in AP-RED algorithm, min_a / C and max_a / C are in proportion to T_p. So are the queuing delay and the round-trip time. With the same network scenario, AP-RED parameters are changed to p_{max} = 0.1 , min_a = 20 packets, max_a = 60 packets and α = 3.5 * 10^{-4} , which are derived from the tuning algorithm (3)-(6). We can see that min_a and max_a decrease when R and C decrease. Fig. 3(b) shows that the queue size of AP-RED reduces to around 40 packets, compared to 100 packets in Fig. 3(a), whilst the queue can be stabilized at the new equilibrium point between the updated min_a and max_a. Consequently, a lower queuing delay is obtained. At the same time, the large queue weight used in AP-RED should improve the transient response. As we observe, the queue settles to its equilibrium point after 5 seconds, compared to 20 seconds in ARED.

![Fig. 3. Comparison of ARED and AP-RED under network scenarios: N= 30, C= 1250 packets/second and T_p = 100ms.](image)

(a) ARED  
(b) AP-RED

**C. Experiment 3**

In the third experiment the round-trip propagation delay increases to T_p = 250 ms and the link capacity increases to C=3000 packets/second. The number of FTP flows increases to N=100. The increase in the round-trip propagation delay and the link capacity should reduce the stability of the system or even make the system unstable [23]. The plot in Fig. 4(a) displays deterministic oscillations of ARED, which lead to large delay jitter and low link utilization. With the same network scenario, AP-RED parameters are adjusted to p_{max} = 0.032 , min_a = 125 packets, max_a = 375 packets and α = 3.2 * 10^{-5} . These parameters are derived from the tuning algorithm (3)-(6) and still satisfy the stability proposition (9). We can see that min_a and max_a increase when R and C increase. Fig. 4(b) displays that the queue of AP-RED can be stabilized around a new equilibrium point between the updated min_a and max_a. Although the queue size becomes large, it remains greater than zero and smaller than the total buffer size as desired. Thus high link utilization is maintained.
In this paper we developed AP-RED to provide a scalable and systematic mechanism for RED parameter tuning. The advantages of AP-RED are that it provides a simple mechanism, from a theoretic standpoint, for constantly tuning four key RED parameters instead of tuning only one of them, like \( P_{\text{max}} \), under varying network conditions, such as traffic load, round-trip time and link capacity. It is unnecessary to consider the interaction among these RED parameters when they are changed in response to changed network conditions. Stability analysis and nonlinear simulations showed that AP-RED could stabilize the queue and maintain high link utilization in dynamic network scenarios. Multi-service situations involving networks with multiple bottlenecks and connections have not been considered in this study. More research is required in order to extend AP-RED to the multi-service. We took a single bottleneck link in a dumbbell topology in our simulations in order to best compare the proposed algorithm with ARED. For a more realistic topology, further work is required.

**REFERENCES**


