An average queue weight parameterization in a network supporting TCP flows with RED

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


Additional Information:

- This is a conference paper [© IEEE]. It is also available from: http://ieeexplore.ieee.org. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Metadata Record: https://dspace.lboro.ac.uk/2134/4141

Version: Published

Publisher: © IEEE

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/
An Average Queue Weight Parameterization in a Network Supporting TCP Flows with RED

Wu CHEN, Yunqiu LI, Shuang-Hua YANG
Computer Science Department, Loughborough University

Abstract—In this paper we use a previously developed RED (Random Early Detection) model to analyze and develop a quantitative approach of defining one of the RED parameters, average queue weight, while the network load level varies. First, we introduce the linear control system and the pre-developed RED model. Based on this model, we next develop a proposition aiming at the RED parameter, average queue weight, and the load level only, to ensure the system stay stable. Our research is intended to provide a quantitative basis for parameterizing RED while load level varies. We present ns simulations to support our analysis.

Index Terms— Parameterization, AQM, RED, Congestion Control

I. INTRODUCTION

The Internet is undergoing a major change in size, diversity, and reach, and impacting all aspects of our lives. It is now the largest man-made interconnected system in the world. Congestion control plays a very important role in the Internet to ensure good network performance. Internet congestion control comprises two main parts: TCP (Transmission control protocol) and AQM (Active Queue Management). Although many AQM algorithms have been proposed over the last decade, such as RED (Random Early Detection) [3], BLUE [4], GREEN (Generalized Random Early Evasion Network) [5], and REM (Random Exponential Marking) [6], only RED AQM has been implemented in major commercial routers. However, parameterizing RED to obtain good performance under variable congestion scenarios is very difficult. Such difficulties discourage network administrators from turning on RED in their routers. Instead, they simply use the default and primitive queue management function, Drop tail, which is not efficient and leads to higher than necessary packet loss. Certainly, there are many parameter tuning techniques for RED proposed in the literature, but they are only applicable under restrictive assumptions, e.g., constant number of greedy TCP connections, etc.

The performance of RED is highly sensitive to its parameter settings. It has even been debated whether RED can deliver good performance in real network scenarios in [7], [8], and [9]. In particular, one of the goals of RED is to stabilize the queue lengths in the routers, but the current version of RED does not succeed in this goal because the steady-state queue length strongly depends on the number of active TCP connections. On the other hand, the system response time is also an aspect to measure the RED stability, as the quicker the queue gets to a stable length, the better the system would perform.

Parameter setting remains a critical open problem today [10] in other AQM or AQM-based approaches, such as Dynamic-RED (DRED), Stabilized-RED (SRED), BLUE and adaptive virtual queue (AVQ). Based on extensive simulations and empirical investigations Floyd et al. [11] have provided a guideline of parameter settings as well as some recommended parameters values for RED; However, the parameters are still tightly coupled with each other and they need to be selected very carefully; otherwise, the performance of RED will be degraded. Therefore, to analyze the parameterization of RED from a quantitative point of view would be a consequential research to take place.

The rest of the paper is organized as follows. In Section II, we analyze the pre-developed model and develop our stable RED proposition. In Section III, we develop our algorithm aiming at the RED parameter, average queue weight, and the load level only. We next present simulation results obtained by using Network Simulator 2 (NS2) that verify our analysis in Section IV. Finally, we present our conclusions in Section V.

II. STABILITY PROPOSITION

A. Model

According to a dynamic model of TCP congestion control [1], a feedback control system of RED shown in Fig. 1 was presented in [2].

![Feedback control system of RED](image)

where
\[
\begin{align*}
\delta W &= W - W_0; \\
\delta p &= p - p_0; \\
\delta q &= q - q_0; \\
W &= \text{expected TCP window size (packets)}; \\
p &= \text{probability of packet mark/drop}; \\
q &= \text{expected queue length (packets)}; \\
(W, q, p) &= \text{the operating point defined by } W = 0 \text{ and } q = 0.
\end{align*}
\]

In Fig. 1, \( C_{\text{red}}(s) \) denotes RED control strategy, \( e^{-s\delta} \) denotes the delay term, \( P_{\text{eq}}(s) \) denotes the linearized TCP dynamics and \( P_{\text{queue}}(s) \) denotes the queue dynamics. They were given in the following equations:

\[
\begin{align*}
P_{\text{eq}}(s) &= \frac{R_0}{s + \frac{1}{R_0}} \\
P_{\text{queue}}(s) &= \frac{R_0}{s + \frac{1}{R_0}} \\
C_{\text{red}}(s) &= \frac{L_{\text{red}}}{s/K + 1}
\end{align*}
\]

where

\[
\begin{align*}
L_{\text{red}} &= \max_{\text{th}} - \min_{\text{th}} \\
K &= \frac{-\log_a(1 - \alpha)}{\delta} \\
\max_{\text{th}} &= \text{maximum threshold} \\
\min_{\text{th}} &= \text{minimum threshold} \\
\alpha &= \text{Average queue weight} \\
\delta &= \text{Sample time} \\
R &= \text{Round-trip time} \\
N &= \text{Load factor (number of TCP sessions)} \\
C &= \text{Link capacity (packet/sec)}
\end{align*}
\]

We assume \( N(t) \equiv N \) and \( R(t) \equiv R_0 \) as constants.

Then we have the following plant transfer function describing how packet-marking probability dynamically affects the queue length.

\[
P(s) = P_{\text{eq}}(s)P_{\text{queue}}(s)e^{-s\delta} = \frac{\left( \frac{C^2}{2N} \right) e^{-s\delta}}{\left( s + \frac{2N}{R_0} \right) \left( s + \frac{1}{R_0} \right)}
\]  

The high-frequency plant gain of \( P(s) \) is \( \frac{C^2}{2N} \). The variation of TCP load \( N \) should have a direct impact on stability, transient response and steady-state performance. A small TCP load \( N \) increases the high-frequency gain leading to decreased stability margins and increased oscillatory response. Conversely, large TCP loads will tend to damp the closed-loop transient response.

\section*{B. Stability Proposition}

According to [2], we derive a simplified version of stability proposition in this section. We assume a scale for the number of TCP sessions, say all \( N \geq N^- \) and all \( R_0 \leq R^+ \).

\textbf{Stability Proposition:} Let \( L \) and \( K \) satisfy:

\[
\frac{L_{\text{red}}(R'C)^3}{(2N^-)^3} \leq \frac{\omega_z}{K} \]

where

\[
\omega_z = 0.1 \min \left\{ \frac{2N^-}{(R^+)C}, 1 \right\}
\]

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

\textbf{Proof:} Consider the frequency response of the compensated loop transfer function, we have

\[
L(j\omega) = C_{\text{red}}(j\omega)P(j\omega)
\]

\[
= \frac{L_{\text{red}}(R_0 C)^3}{(2N)^3} e^{-j\omega R_0} \\
\quad \cdot \left( \frac{j\omega}{K} + 1 \right) \left( \frac{j\omega}{2N^{-}} + 1 \right) \left( \frac{j\omega}{r_0^{-}} + 1 \right)
\]

\[
\leq \frac{L_{\text{red}}(R_0 C)^3}{(2N)^3} e^{-j\omega R_0} \cdot \frac{1}{(R_0 C)^3} \frac{1}{(R_0 C)^3}
\]

\[
= \omega_z \leq \frac{\omega_z}{K} \quad \forall \omega \in [0, \omega_z]
\]

Given any \( N \geq N^- \) and \( R_0 \leq R^+ \),

\[
|L(j\omega_z)| \leq \frac{\omega_z}{K} + 1 < \frac{\omega_z}{K}
\]

From this and (3) it follows

\[
|L(j\omega_z)| < 1
\]

We again use (4) to obtain

\[
L_{\text{red}}(R'C)^3 \left( \frac{2N^-}{(R^+)C} \right)^3 \leq \frac{\omega_z}{K} \quad \forall \omega \in [0, \omega_z]
\]

(7)

According to Nyquist stability criterion, (6) and (7) indicate that the closed-loop system is asymptotically stable.

\section*{III. ALGORITHMPROPPOSITION}

In this section we develop an algorithm for tuning RED
parameters from control theoretic standpoint. Let’s consider a single bottlenecks router in which all flows have the same round-trip time and packet size. Given initial TCP load \( N_o \), we have the initial average queue weight \( \alpha_o \), and other RED parameters from (3) to stabilize the system. Then the following algorithm describes how we change \( \alpha \) according to the observed load \( N^- \).

\[
\alpha = \left( \frac{N^-}{N_o} \right)^3 \alpha_o \tag{8}
\]

where
\[
N_o, N^- \leq RJC/2
\]
\[R(t) = R_o\]
\[C(t) = C_c\]

or
\[
\alpha = \left( \frac{N^-}{N_o} \right)^2 \alpha_o \tag{9}
\]

where
\[
N_o, N^- \geq RJC/2
\]
\[R(t) = R_o\]
\[C(t) = C_c\]

Proof:
From (1) we have
\[
\alpha = 1 - e^{-\lambda t} = 1 - (1 - K\delta) = K\delta
\]
\[K = \frac{\alpha}{\delta} \tag{10}
\]

From (10) and (3), it follows
\[
\frac{\alpha}{\delta} I_{m} \leq \left( \frac{2N^-}{RJC} \right)^3 \omega_s 
\]

(11)

Given any \( N_o, N^- \leq RJC/2 \), from (11) and (4) we have
\[
\frac{\alpha}{\delta} I_{m} \leq \left( \frac{N^-}{RJC} \right)^3 \frac{0.8}{C} \tag{12}
\]

Thus, when \( N^- \) changes, we have the following algorithm of \( \alpha \) that can satisfy (12)

\[
\alpha = \left( \frac{N^-}{N_o} \right)^3 \alpha_o \tag{13}
\]

where
\[
R(t) = R_o\]
\[C(t) = C_c\]

Similarly, given any \( N_o, N^- \geq RJC/2 \), we have
\[
\frac{\alpha}{\delta} I_{m} \leq \left( \frac{N^-}{RJC} \right)^3 \frac{0.1}{C}
\]

and
\[
\alpha = \left( \frac{N^-}{N_o} \right)^2 \alpha_o
\]

where
\[
R(t) = R_o\]

From (5), it is clear that a larger \( K \) means faster response time and vice versa. So we choose the average queue weight \( \alpha \) in \( K \) to tune according to (8) and (9). When TCP load \( N \) becomes larger, a larger \( \alpha \) can improve transient response while keep system stable. On the other hand, when \( N \) becomes smaller, although a smaller \( \alpha \) could damp transient response, it makes system retain enough stability margin.

IV. SIMULATIONS

We verify our propositions via a set of ns simulations. However, our proposition is based on a liberalized model while ns simulation is actually non-linear in nature.

Our aim is to analysis how the change of load level would affect the RED performance and how tuning average queue weight correspondingly could help. Our experiment includes the following steps:

1. We define a set of parameters in Mode 1, which make the network perform stably, compared with network with default RED parameters.

2. We increase the load level, and remain the default parameters to analysis the load level’s influence. After that, retaining the high load level, we tune the average queue weight according to our proposition.

3. We define another set of parameters to make equation (3) only satisfied with a large load level in Mode 2, and then reduce the load level to see its effect to the queue stability. After that, we reduce the average queue weight correspondingly to see the output.

We look at a single bottle-link router with RED application. The topology of the network is shown in Fig. 2.

A number of HTTP flows is also included in this simulation. The short-lived effects of HTTP flows could be considered as the noise to the RED queue.
**Mode 1:** Set parameters: \( N = 300, \ R^+ = 200 \text{ ms}, \) packet size = 500 Bytes, so \( C = 15 \text{ Mb/s} = 3750 \text{ packets/s}, \ p_{\text{max}} = 0.1. \)

From (4), we have

\[
\omega_g = 0.1 \min \left\{ \frac{2N^-}{(R^+)^2 C}, \frac{1}{R^+} \right\} = 0.4
\]

For \( K = 0.01, \) from (3), we have

\[
L_{\text{red}} \leq \frac{\omega_g (2N^-)^2}{K (R^+ C)^3} = 3.41e - 2
\]

In this case, we set \( L_{\text{red}} \) to be 0.0002.

From (1), we have the minimum dynamic range of the threshold \((\max_{\text{th}} - \min_{\text{th}})\) should be 500 packets.

**A. Experiment 1**

In the first experiment, we look at a queue with 300 ftp (greedy) and 150 http flows. The bottle link bandwidth is 15Mb/s, and the propagation delays for the flows range uniformly between 160 and 240 ms. The minimum threshold and maximum threshold are set as 150 and 250 packets, with average packet size being 500 Bytes. The queue has an 800 packets buffer, and all of the other parameters are set to be default.

We set the RED parameter, gentle\_as, as “TURE” to make the average queue change smoothly. The current queue length and average queue length are shown in Fig. 3.

As we can see, the average queue length is oscillating around 250, and the current queue length frequently goes down to zero. It certainly leads to the under-utilization of the bottle-link, and brings considerable jitter to the round trip time.

**B. Experiment 2**

Now we use our model as derived before to re-set the parameters. The threshold is set as 150-650. We set \( p_{\text{max}} \) to 0.1, and \( \alpha \) to 2.66e-6.

The system gets stable in 400 seconds, which is almost as twice long as it took in Experiment 2. This long unstable duration could be the major disturbance which is certainly not expected to be encountered in the operation.

**C. Experiment 3**

In this experiment, we add another 100 FTP and 50 HTTP flows into the load level. The \( p_{\text{max}} \) and \( \alpha \) remain to the same as in Experiment 2.

The results are plotted in Fig. 5.

**D. Experiment 4**
Now we increase the average queue weight to maintain a constant $\omega/(N)^3$ ratio in the scale $N_0, R^C/2$ and a constant $\omega/(N)^2$ ratio when $N$ is larger than $R^C/2$. The results are plotted in Fig. 6.

Compared with previous outputs, the system response time indeed goes down. The system performance gets stable in less than 200 seconds.

**E. Experiment 5**

To analysis how a small load level could affect the system, we let the load level vary during the simulation to make the compare more obvious. This simulation starts with 300 FTP and 150 HTTP. We set the duration to be 1200 seconds. After one third of the duration, three forth of the data sources stop, and re-start after another 600 seconds.

Based on Mode 1, we now define Mode 2 that only satisfy equation (3) when the load level is heavy.

**Mode 2:** Firstly, when $N^- = 300$, $R^+ = 200$ ms, $C = 3750$ packets/s, $p_{max} = 0.1$, $(\max_{th} - \min_{th}) = 500$, from (1), we have

$$L_{red} = \frac{p_{max}}{\max_{th} - \min_{th}} = 0.0002$$

From (3) and (4), we have

$$K \leq \frac{\omega_s (2N^-)^2}{(R^+ C)^3 L_{red}} = 1.71$$

So we set $K=1.5$, from (10), we have

$$\alpha = K \delta = 3.99e-4$$

Based on our proposition, this average queue weight could only guarantee a stable RED supporting large $N^-$, because if we set $N^-$ down to 100, from (4), we have

$$\omega_s = 0.1 \min \left\{ \frac{2N^-}{(R^+)^2 C} + \frac{1}{R^+} \right\} = 0.133$$

From (14), we have

$$K \leq \frac{\omega_s (2N^-)^2}{(R^+ C)^3 L_{red}} = 0.63$$

So, from (10), we have

$$\alpha = K \delta \leq 1.69e-5$$

Obviously, $\alpha = 3.99e-4$ is not in this scale.

The results are plotted in Fig. 7. It is observed that the response time is extremely small, because of the large $\alpha$. However, when there are only 100 FTP flows, the current queue length fluctuates between a much larger scale about the average queue. It shows that a lower load level brings larger oscillation range to the queue length. This is because when $N$ reduces, the right side of the equation (14) gets smaller as well, that directly make the equation unsatisfied.

**F. Experiment 6**

Based on Experiment 5, we now decrease the average queue weight to retain a constant $\omega/(N)^3$ ratio. As $\alpha = 3.99e-4$, $N_0 = 300$, $N^- = 100$, from (13), we have

$$\alpha = \left( \frac{N^-}{N_0} \right)^3 \alpha_0 = 1.48e-5$$

The results are plotted in Fig. 8.

We observe the system is settled to stable in 80 seconds, which is a longer duration than it was in Experiment 5. However, the oscillation around the average queue while system is stable obviously gets smaller, which leads a better system performance while the load level reduces.

594
V. CONCLUSION

We have developed a simple, scalable rule of parameter tuning for RED routers by performing a control-theoretic analysis of TCP/RED system. Instead of tuning many parameters of RED, we adapt only average queue weight to the changing network loads while the TCP/RED system keep stable and achieves desirable transient performance. The simulation results verified our analysis. This is a starting point for decoupling RED parameters. In practice, there are more complex and realistic network conditions under which we need to explore. We are currently studying the self-tuning algorithm for RED to enhance its performance in practical networks involving flows with variable round-trip time and packet size.

REFERENCES