PRED: Prediction-Enabled RED

Min Gyo Chung and Euinam Huh

Dept. of Computer Science, Seoul Women's University, Seoul, Korea
{mchung,huh}@swu.ac.kr

Abstract. This paper proposes a router congestion control mechanism called PRED (Prediction-enabled RED), a more adaptive and proactive version of RED (Random Early Detection). In essence, PRED predicts its queue length for an early detection of possible congestion alerts in the near future and operates adaptively to the predicted changes in traffic patterns. Typically, PRED does this by first making prediction about average queue length and then using the predicted average queue length to adjust three classic RED parameters max_{th}, min_{th}, and max_{p}. The incoming packets after the adjustment are now being dropped with the new probability defined by updated parameters. Due to its adaptability and proactive reaction to network traffic changes, PRED can be considered as a novel solution to dynamically configure RED. Extensive simulation results from NS-2 simulator are presented to verify the performance and characteristics of PRED.

1 Introduction

DT (Drop Tail) and RED (Random Early Detection) [1] are two well-known router congestion control algorithms. DT discards the packets when the queue becomes full, but RED drops the packets randomly before the queue is completely full. However, in order that RED operates at the maximal fairness without incurring major network disruption, RED should be properly configured. RED configuration is a problem to find out the optimal set of RED parameters given some dynamic traffic conditions such as number of active flows, connection bandwidth, congestion level, etc. The solution to the configuration problem can't be obtained easily because of the dynamic nature of the network conditions.

A number of approaches to this problem have appeared in the literature. The authors in [2,3] addressed the configuration problem and described many difficulties in the deployment of RED routers into the real world networks. The methods in [4,5] have been focused on the direct control of a packet drop probability, but Padhye et al. [6] shows that the fast and frequent fluctuation of the drop probability rather leads to lower overall throughput.

In this work, we propose PRED (Prediction-enabled RED), which is an adaptive and proactive version of RED (Random Early Detection) designed to alleviate the RED configuration difficulties. Basically, PRED keeps monitoring the current queue status for an early detection of possible congestion alerts in the near future and operates adaptively to the anticipated congestion, thereby leading to a more proactive reaction to changes in traffic patterns. Specifically, PRED

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3038, pp. 1193–1200, 2004.

does this by first making prediction about average queue length and then using the predicted queue size to adjust three classic RED parameters max_{th}, min_{th}, and max_{p}. The incoming packets after the adjustment are now dropped with the new probability defined by updated parameters. Due to its inherent characteristics of adaptability and proactive reaction, PRED can be a novel solution to the above configuration problem.

In addition, PRED has some other features. It still allows fair bandwidth sharing among TCP flows, and yields better network utilization than RED. As with standard RED, PRED can be also added to an existing FIFO-based router without any problems. To verify the performance and characteristics of PRED, we have conducted extensive simulation experiments using NS-2 simulator.

The rest of this paper is organized as follows. In Sec. 2, DT and RED congestion control algorithms are briefly introduced. In Sec. 3, we give a detailed description of PRED. In Sec. 4, detailed simulation results and analyses are presented to show the performance of PRED in many different aspects. Lastly, some concluding remarks are given in Sec. 5.

2 DT and RED

DT, a simplest form of congestion control over a router queue, accepts packets for the queue until the queue is full and then discards new incoming packets until the queue gets its room back again. However, DT occasionally allows a small number of flows to monopolize the queue space, leading to the disproportionate distribution of packet losses among flows. It also has a bias against bursty traffic and possibly causes a global synchronization problem.

RED is another congestion control scheme, specifically designed to eliminate the problems encountered in DT. RED reacts to a congestion signal proactively by dropping packets before the queue is completely full. Further, the packet drop is done at random so that all competing flows will get treated fairly in terms of packet loss rate. The drop probability used by RED, $P_{\text{d}}(\cdot)$, is a linearly increasing function of the average queue length and is defined as follows:

$$P_{\text{d}}(q) = \begin{cases} 0, & q < min_{\text{th}} \\ \frac{q - min_{\text{th}}}{max_{\text{th}} - min_{\text{th}}} max_{\text{p}}, & min_{\text{th}} \leq q \leq max_{\text{th}} \\ 1, & max_{\text{th}} < q, \end{cases}$$

where q denotes the (average) queue size, max_{th} a maximum queue length threshold, min_{th} a minimum queue length threshold, and max_{p} a maximum drop probability.

3 PRED

3.1 Motivation

RED has been used for congestion control in the Internet for many years, but it has one major drawback, namely RED configuration problem. If RED is not

configured properly, it can bring about traffic disruption and lower network utilization. The author in [7] showed that under the false configuration, the fairness of RED gets even worse than the fairness of DT as the number of TCP flows increases. Observing that RED yields the best result when the average queue length comes between max_{th} and min_{th}, Hasegawa et al. [8] proposed dt-RED. Although dt-RED exhibited some improvement over its predecessors, it is still difficult to find out max_{th} and min_{th} optimal for the current queue status. Overall, finding an optimal solution to RED configuration problem is tough, and requires many resources and efforts. PRED mainly aims to relieve such configuration difficulties.

3.2 Algorithm

As shown in Fig. 1, PRED consists of two functional modules: prediction module (PM) and congestion control module (CCM). PM continuously monitors the queue to collect its current and past statistical data into a database. Based on the accumulated information, it detects possible congestion signals ahead in the near future by analytically making predictions for some unknown variables such as average queue length. Using the predicted values generated by PM, CCM is responsible for updating three classic RED parameters max_{th}, min_{th}, and max_{p} to be optimal for the current network conditions. The packet drop probability is now redefined by updated parameters and subsequent incoming packets will be discarded with updated probability.

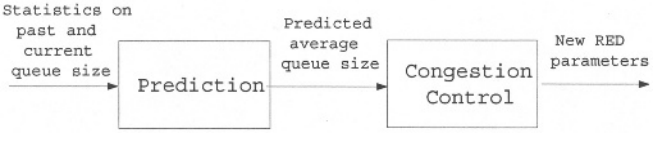

Fig. 1. PRED modules

With the average queue lengths in the current and past cycles, PRED is able to predict the variation of the average queue length in the next cycle. RED computes the average queue length in the current cycle to obtain the drop probability. On the other hand, PRED executes a built-in prediction algorithm to forecast the average queue length in the next cycle and compares the predicted average queue length with max_{th} and min_{th} to update those parameters with proper values. For example, if the predicted average queue length is greater than max_{th}, the threshold max_{th} is increased in advance so that the average queue length will not exceed max_{th} in the next cycle. Similarly, if the predicted average queue length is less than min_{th}, the threshold min_{th} is decreased before the next cycle so that the average queue length will come between max_{th} and min_{th}. In this way, PRED figures out the adequate values for thresholds max_{th} and min_{th}, adaptively to the dynamic changes in traffic patterns.

Prediction Module. Numerous prediction methods have already been available in the literature, including MACD (Moving Average Convergence/Divergence), AR (Autoregressive), and LR-Line (Linear Regression-Lines). The MACD model is based on two moving average series. When the shorter term moving average crosses over the longer term moving average, a rising tendency is predicted. Likewise, when the shorter term moving average falls below the longer term moving average, a declining signal is generated. The AR model, also known as IIR(Infinite Impulse Response) filter in engineering field, is one of linear prediction formulas that attempts to predict an output of a system using the current/past inputs and the past outputs. The LR-Lines is a statistical tool to make prediction about an unknown variable by discovering an equation for a line that most nearly fits the given data.

We tested the above three prediction approaches and discovered that all but the LR-Lines don't forecast the average queue length very well. The LR-Lines quite correctly predicts the average queue length when the queue is long enough, but as the queue gets shorter, it reports some discrepancies, that is, the predicted average queue length tends to be greater than the one measured. Nevertheless, it doesn't incur any big problems in applying the LR-Lines to PRED, because the throughput of RED-type routers doesn't get seriously deteriorated even if the average queue length goes below min_{th} due to the false prediction.

Congestion Control Module. Shortly after the predicted average queue length, denoted by avg_{p}, is available from PM, CCM attempts to modify three RED parameters max_{th}, min_{th} and max_{p} by referring to avg_{p}. Here, avg_{p} is calculated by $\frac{avg_{\text{p}_1}+avg_{\text{p}_2}+avg_{\text{p}_3}}{3}$, where avg_{p_i} is the predicted average queue length next i-cycles ahead of the current cycle. Depending on the relationship between avg_{p} and two thresholds (max_{th} and min_{th}), the modification can be done in three distinct ways. We will next discuss three possible situations and how CCM works in those situations.

First, consider the situation where avg_{p} is greater than max_{th}. This situation, for example, can take place when the amount of incoming traffic rises dramatically in a short period of time. In preparation for such a sudden traffic increase, CCM shifts max_{th} and min_{th} to the right such that avg_{p} is positioned between new max_{th} and min_{th} (See Fig. 2 (a)). More precisely, to calculate max_{th}, min_{th} and max_{p}, we use the following procedure:

$$\begin{aligned} max_{\text{th}} &= \min\{avg_{\text{p}}(1+i_{\max}), max_{\lim}\} \\ min_{\text{th}} &= \min\{min_{\text{th,old}}(1+i_{\min}), avg_{\text{p}}\} \\ max_{\text{p}} &= \frac{max_{\text{p,lim}}}{max_{\lim}-min_{\lim}}(max_{\text{th}}-min_{\lim}) \end{aligned}$$

where $max_{\lim}$ and $min_{\lim}$ represent maximum and minumum values, respectively, that the thresholds max_{th} or min_{th} can possibly take; $i_{\max}$ and $i_{\min}$ represent the rate of increase for max_{th} and min_{th}, respectively; $max_{\text{p,lim}}$ is a maximum possible value for max_{p}; and $min_{\text{th,old}}$ is the previous value of min_{th}. Note that the equation of the straight line connecting two points $(min_{\lim}, 0)$ and

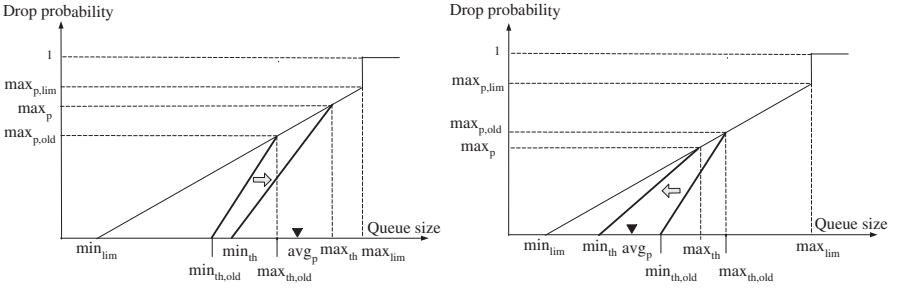

Fig. 2. PRED congestion control. (a) Situation where predicted average queue length is expected to increase, and (b) Situation where predicted average queue length is expected to decrease

$(max_{\text{lim}}, max_{\text{p,lim}})$ is $y = \frac{max_{\text{p,lim}}}{max_{\text{lim}} - min_{\text{lim}}}(x - min_{\text{lim}})$, and that max_{p} is derived from this equation by substituing $x = max_{\text{th}}$.

Second, consider the situation where avg_{p} is less than min_{th}, which is usually caused by persistent low traffic. Motivated by the fact that RED-type algorithms show the best result when avg_{p} is bounded by thresholds max_{th} and min_{th}, CCM will set up the new values for those thresholds by shifting them to the left. Detailed procedure to calculate max_{th}, min_{th} and max_{p} is shown below:

$$\begin{aligned} min_{\text{th}} &= \max\{avg_{\text{p}}(1 - d_{\text{min}}), min_{\text{lim}}\} \\ max_{\text{th}} &= \max\{max_{\text{th,old}}(1 - d_{\text{max}}), avg_{\text{p}}\} \\ max_{\text{p}} &= \frac{max_{\text{p,lim}}}{max_{\text{lim}} - min_{\text{lim}}}(max_{\text{th}} - min_{\text{lim}}) \end{aligned}$$

where d_{max} and d_{min} represent the rate of decrease for max_{th} and min_{th}, respectively; $max_{\text{th,old}}$ is the previous value of max_{th}; and other names carry the same meaning as the above first situation. Note that the formula to compute max_{p} is same.

The last situation can happen when $min_{\text{th}} \leq avg_{\text{p}} \leq max_{\text{th}}$. Generally, in this circumstance, each of incoming flows sends packets at its sustained rate, so the traffic does not show any remarkable fluctuations. Since avg_{p} is already bounded by thresholds max_{th} and min_{th}, CCM does not execute any particular operations to change max_{th}, min_{th} and max_{p}.

In brief, PRED gets a control over the packet drop probability in more proactive sense than RED by predicting the average queue length ahead of time and updating max_{th}, min_{th} and max_{p} based on the predicted average queue length. Ultimately, this proactive and adaptive behavior of RED to traffic dynamics helps to facilitate RED configuration process.

3.3 Features

In this section, we will describe some other notable features that PRED retains, especially compared with standard RED. The average queue size is used by RED

to determine its drop rate. Instead of instantaneous queue size, using the average queue size makes RED tolerant toward brief queue changes or bursts. However, the use of the average queue size can also bring undesirable detrimental effects to RED's overall throughput, because of its delayed reaction to dynamically changing network conditions. PRED is able to minimize this sort of detriment by reacting a few cycles earlier based on predicted quantities by LR-Lines.

In RED, the parameters max_{th}, min_{th} and max_{p} are fixed, but PRED dynamically adjusts their values according to the level of traffic congestion. Persistent high traffic will allow PRED to keep moving max_{th} and min_{th} toward max_{lim}, so the distance between max_{th} and min_{th} gets shorter and max_{p} gets bigger gradually. Similarly, as the traffic runs low, max_{th} and min_{th} come closer to min_{lim}. As a result, the distance between max_{th} and min_{th} as well as max_{p} becomes smaller and smaller.

In comparison with RED, PRED still allows fair bandwidth sharing among TCP flows and yields better network utilization. Further, as with standard RED, PRED can be added to an existing FIFO-based router without any big modification.

4 Simulation and Analyses

4.1 Simulation Environment

To demonstrate the performance and characteristics of PRED, we used NS-2 (Network Simulator Version 2) simulator and the topology of simulated network as shown in Fig. 3. NS is a discrete event simulator targeted at networking research [9]. The topology consists of N source nodes, one sink node and a single PRED router shared by the source and sink nodes. Each connection between a source node and the PRED router is a TCP link with 100 Mbps capacity and 10 ms propagation delay, while the connection between the sink node and the router is a TCP link with 1 Mbps and 10 ms propagation delay. TCP New-Reno was applied to those links.

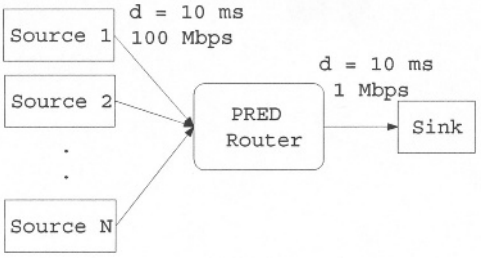

Fig. 3. Simulation Topology

4.2 Simulation Results

We consider the following simulation scenario: the number of source nodes, N, has some distinct values of 21, 100, 150, and 200, and each source node will send out packets at different bit rate every 10 second over a 70-second interval. This simulation scenario is applied to each of DT, RED, and PRED algorithms to measure the performance metrics such as fairness and the amount of dropped packets for congestion control.

The first simulation result as shown in Fig. 4 (a) supports that PRED performs well rather than the conventional TD and RED in terms of total delivered input from source nodes. This implies that PRED adjusts the drop probability to smaller than the RED as the declined traffic curve is forecasted. The second experiment as shown in Fig. 4 (b) is done under the increased traffic trend but the amount of packet delivered to the router is smaller than the peak transmission rate. PRED drops more packets than the RED as we expected. This implies the LR-Lines forecasts packet trend accurately. The fairness metric (as TCP is used in each node) is also measured from the third experiment as shown in Fig. 4 (c). PRED provides better fairness than others when the number of nodes is increased.

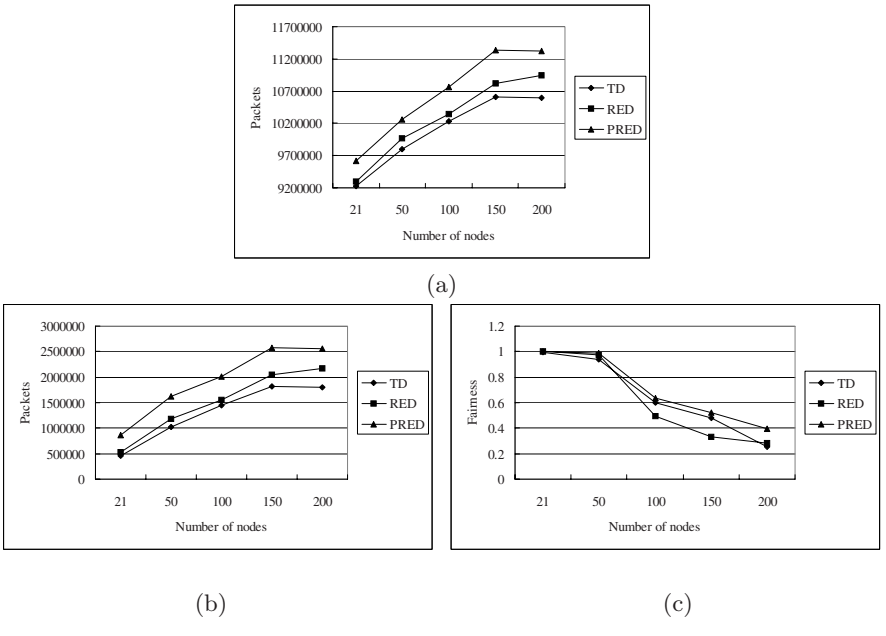

Fig. 4. Performance comparison: (a) Total amount of input data packets delivered over the link between the source nodes and the router, (b) Amount of data being dropped for congestion control when avg_{p} increases, and (c) Fairness index

5 Conclusion

This paper proposes a new congestion avoidance scheme called PRED, a more adaptive and proactive version of RED. PRED consists of two functional modules: prediction module (PM) and congestion control module (CCM). PM continuously examines the queue to collect its current and past statistical data into a database. Based on the accumulated information, it detects possible congestion signals ahead of time by making predictions about average queue length. Using the predicted values generated by PM, CCM is responsible for updating three classic RED parameters max_{th}, min_{th}, and max_{p} to be optimal for the current network conditions.

PRED provides more adaptability and proactive reaction to network traffic changes than RED, thus can be effectively used to dynamically configure RED parameters. Further, PRED allows fair bandwidth sharing among TCP flows, yields better network utilization, and can be added to an existing FIFO-based router without any big modification.

References

1. S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, vol. 1, pp. 397-413, Aug. 1993
2. Martin May, Jean Bolot, Christophe Diot, and Bryan Lyles, "Reasons not to deploy RED," in Proceedings of IWQoS '99, June 1999
3. Mikkel Christiansen, Kevin Jeffay, David Ott, F. Donelson Smith, "Tuning RED for web traffic," in Proceedings of ACM SIGCOMM 2000, August 2000
4. Haining Wang and Kang G. Shin, "Refined design of random early detection gateways," in Proceedings of Globecom '99, pp. 769-775, December 1999
5. Wu-chang Feng and Dilip D.Kandlur and Debanjan Saha and Kang G. Shin, "A self-configuring RED gateway," in Proceedings of IEEE INFOCOM '99, March 1999
6. J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: a simple model and itsempirical validation," in Proceedings of ACM SIGCOMM '98, pp. 303-314, Aug. 1998
7. Raj Jain, "Throughput fairness index: An explanation," ATM Forum Contribution 99-0045, February 1999
8. Go Hasegawa, Kouichi Tokuda and Masayuki Murata, "Analysis and Improvement of fairness among many TCP connections sharing Tail-Drop and RED Routers" in Proceedings of INET 2002
9. LBNL, LBNL Network Simulator-ns version 1, http://www-nrg.ee.lbl.gov/ns/