ELectronics EXpress

IEICE Electronics Express, Vol.7, No.17, 1276–1282

An improvement over RED algorithm by using particle swarm optimization

Shahram Jamali[1a)] **and Seyed Reza Zahedi**[2a)]

[1] *University of Mohaghegh Ardabili, Ardabil, Iran*

[2] *Islamic Azad University-Khalkhal Branch, Khalkhal, Iran*

a) *jamali@iust.ac.ir*

b) *seyed.reza.zahedi@gmail.com*

Abstract: One of the most challenging issues in Random Early Detection (RED) algorithm is how to set its parameters to achieve high performance for the dynamic conditions of the network. While original RED uses fixed values for its parameters, this paper proposes a novel algorithm, in which particle swarm optimization (PSO) technique is used to dynamic tuning of RED's parameters. For this purpose, we formulate the active queue management issue as an optimization problem to be solved by PSO algorithm. Then, we employ PSO technique to direct the system to its optimum point. Simulation results show that the proposed algorithm behaves remarkably better than RED in terms of queue size, number of dropped packets, bottleneck utilization and global stability.

Keywords: congestion control, RED, particle swarm optimization (PSO)

Classification: Science and engineering for electronics

References

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

[2] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, 1993.

[3] W. Feng, D. Kandlur, D. Saha, and K. Shin, "A self-configuring RED gateway," *Infocom*, 1999.

[4] C. Hollot, V. Misra, D. Towlsey, and W. Gong, "A control theoretic analysis of RED," *IEEE INFOCOM*, 2001.

[5] W. Chen and S. Yang, "The mechanism of adapting RED parameters to TCP traffic," *J. Comput. Commu.*, vol. 32, 2009.

[6] S. H. Zahiria and S. A. Seyedin, "Swarm intelligence based classifiers," *J. Franklin Institute*, vol. 344, 2007.

[7] J. Kennedy and R. Eberhart, "Particle swarm optimization," *IEEE Neural Networks*, vol. 4, 1995.

[8] R. C. Eberhart and Y. Shi, "Particle swarm optimization: Development, applications and resources," *IEEE Congress on Evolutionary Computation,* 2001.

DOI: 10.1587/elex.7.1276
Received July 13, 2010
Accepted August 04, 2010
Published September 10, 2010

[9] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, 2002.
[10] T. Ye and S. Kalyanaraman, "Adaptive tuning of RED using on-line simulation," *GLOBECOM*, vol. 3, 2002.
[11] L. Rossides, A. Sekercioglu, A. Pitsillides, A. Vasilakos, S. Kohler, and P. Tran-Gia, "Fuzzy red: congestion control for TCP/IP diff-serv," *Electrotechnical Conf. MELECON 2000*, vol. 1, 2000.
[12] V. Vukadinoviæ and L. Trajkoviæ, "RED with dynamic thresholds for improved fairness," *School of Engineering Science Simon Fraser University*, 2000.
[13] M. Christiansen, K. Jeffay, D. Ott, and F. Donelson, "Tuning RED for web traffic," 2006. [Online] *http://www.Cs.Unc.Edu/Research/Dirt*
[14] C. Chien and W. Liao, "A self-configuring RED gateway for quality of service (QOS) networks," *IEEE ICME*, 2003.
[15] J. Orozco and D. Ros, "An adaptive rio (a-rio) queue anagement algorithm," *Lecture Notes in Computer Science*, Springer, 2003.
[16] M. Maowidzki, "Simulation-based study of ecn performance in red networks," *Military Communication Institute*, 2007.
[17] B. Prabhakar and K. Psounis, "Choke: a stateless active queue management scheme for approximating fair bandwidth allocation," *IEEE INFOCOM,* 2000.
[18] M. Jahanshahi and M. R. Meybodi, "An Adaptive Congestion Control Method for Guaranteeing Queuing Delay in RED-Based Queue Using Learning Automata," *Harbin*, China, 2007.

1 Introduction

The Internet research community regards active queue management (AQM) as an important mechanism to notify traffic sources about the early stages of congestion and thereby avoid the need for strong source reactions due to heavy overload [1]. The original Random Early Detection (RED) algorithm, defined in [2], can be considered to be the most popular AQM mechanism. Many researches have been devoted to the analysis of the dynamics of RED in interaction with TCP congestion control [3, 4, 5]. In [4] and [5] the authors have proposed a procedure to set parameters of RED based on control theory. However, the first one hasn't illustrated how the proposed method can be applied in networks with changing conditions. To solve this problem, Chen and his coworkers in [5] set parameters as function of network conditions, but their algorithm needs to know the number of passing flows and RTT of each flow. As we know original RED doesn't use these values, and hence their computation is an additional heavy load on RED routers. In this paper we aim at finding a dynamic approach for tuning of RED parameters to meet all requirements of an efficient AQM scheme, independently of the load range. For this purpose, this paper uses the PSO algorithm [6] for tuning RED parameters.

DOI: 10.1587/elex.7.1276
Received July 13, 2010
Accepted August 04, 2010
Published September 10, 2010

2 Preliminaries: RED and PSO algorithms

A router implementing RED accepts all packets until the queue reaches its minimum threshold min_{th}, after which it drops a packet with a linear probability distribution function. When the queue length reaches its maximum threshold max_{th}, all packets are dropped with a probability of one. The RED algorithm, therefore, includes two computational parts: computation of the average queue length and calculation of the drop probability. The average queue length at time t, is defined according to equation (1).

$$avg(t) = (1 - w)avg(t - 1) + wq(t) \tag{1}$$

The $q(t)$ is instantaneous queue length at time t, and w is a weight parameter for calculating $avg(t)$.

In this work we will propose a PSO-based procedure to dynamically adjustment of RED parameters, namely, max_{th} and min_{th}. PSO, which is tailored for optimizing difficult numerical functions and is based on metaphor of human social interaction, is capable of mimicking the ability of human societies to process knowledge [6]. It has roots in two main component methodologies: artificial life (such as bird flocking, fish schooling and swarming); and evolutionary computation. Its key concept is that potential solutions are flown through hyperspace and are accelerated toward better or more optimum solutions. Its paradigm can be implemented in simple form of computer codes and is computationally inexpensive in terms of both memory requirements and speed. It lies somewhere in between evolutionary programming and the genetic algorithms. As in evolutionary computation paradigms, the concept of fitness is employed and candidate solutions to the problem are termed particles, each of which adjusts its flying based on the flying experiences of both itself and its companion. The position and velocity of particle i at iteration k can be respectively expressed as

$$X_i(k)=[X_{i1}(k), X_{i2}(k), \ldots, X_{iN}(k)] \tag{2}$$

$$V_i(k)=[V_{i1}(k), V_{i2}(k), \ldots, V_{iN}(k)] \tag{3}$$

Particle i keeps track of its coordinates in the solution space which are associated with the best solution that has achieved so far by that particle. This value is called local best $Lbest_i$. Another best value that is tracked by the PSO is the best value obtained so far by any particle in the neighborhood of that particle. This value is called global best $Gbest$. The basic concept of PSO lies in accelerating each particle toward its local best and the global best locations. The velocity and position of particle i at iteration $k+1$ can be calculated according the following equations:

$$V_i(k+1) = wV_i(k) + c_1r_1\left(Lbest_i(k) - X_i(k)\right) + c_2r_2\left(Gbest(k) - X_i(k)\right) \tag{4}$$

$$X_i(k+1) = X_i(k) + V_i(k+1) \tag{5}$$

Where w is the inertia weight, c_1 and c_2 are constants which determine the influence of the local best position $Lbest_i(k)$ and the global best position $Gbest(k)$. Parameters r_1 and r_2 are random numbers uniformly distributed within [0,1].

DOI: 10.1587/elex.7.1276
Received July 13, 2010
Accepted August 04, 2010
Published September 10, 2010

3 Proposed method

Performance of RED algorithm, respect to the control of the queuing delay and link utilization, obviously depends on the optimal tuning of the RED's parameters. As we know, drop probability is function of max_{th} and min_{th}, beside other parameters. Hence, we can improve drop function dynamically by appropriate setting of max_{th} and min_{th}. It is well-known that using conventional method for optimal tuning of RED parameters may be tedious and time consuming. In order to overcome this problem this paper uses a PSO-based approach for tuning RED parameters. For this purpose, the problem of active queue management is formulated as an optimization problem and then PSO technique is used to solve this problem.

As the first step of our design process, an objective function must be defined to link the design requirements with the optimization algorithm. Our design is aimed to keep the queue length in its lowest level, while it considers constrains such as high utilization in bottleneck link. Considering all the above factors, the objective function is defined as follows:

$$\begin{aligned} &Minimize \;\; avg(t) \\ &Subject\ to: \;\; Bottleneck\,Utilization \;>\; Target\ Utilization \end{aligned} \tag{6}$$

Note that the $"Target\ Utilization"$ is the lowest acceptable level of utilization that is determined by the designer in range of (0, 1]. Although low values lead to smaller average queue length, its side effect is waste of the network resources. On the other hand, high values result in greater queue size along with efficient use of the network resources. In this work we set it to be 0.8. By using this objective function, we can employ PSO technique to design a novel AQM algorithm, in which, thresholds are set by considering the link utilization, the queue length and the number of dropped packets. In this way, the minimum threshold and maximum threshold will be calculated according to the following equations:

$$\begin{aligned} V_{min_{th}}(k+1) = {} & wV_{min_{th}}(k) + c_1 r_1 \left(Gbest_{min\,th} - min_{th}(k)\right) \\ & + c_2 r_2 \left(Lbest_{min\,th} - min_{th}(k)\right) \end{aligned} \tag{7}$$

$$min_{th}(k+1) = min_{th}(k) + V_{min_{th}}(k+1) \tag{8}$$

$$\begin{aligned} V_{max_{th}}(k+1) = {} & wV_{max_{th}}(k) + c_1 r_1 \left(Gbest_{max\,th} - max_{th}(k)\right) \\ & + c_2 r_2 \left(Lbest_{max\,th} - max_{th}(k)\right) \end{aligned} \tag{9}$$

$$max_{th}(k+1) = max_{th}(k) + V_{max_{th}}(k+1) \tag{10}$$

Where ($Lbest_{min\,th}$, $Lbest_{max\,th}$) is the local best position and ($Gbest_{min\,th}$, $Gbest_{max\,th}$), is the global best previous position. These best positions are selected according to objective function of (6).

The parameters c_1 and c_2 determine the relative pull of $Lbest$ and $Gbest$ and the parameters r_1 and r_2 lead to stochastically varying these pulls. These parameters should be selected sensitively for efficient performance of PSO-RED. The constants c_1 and c_2 represent the weighting of the stochastic acceleration terms that pull each particle toward $Lbest$ and $Gbest$ positions.

DOI: 10.1587/elex.7.1276
Received July 13, 2010
Accepted August 04, 2010
Published September 10, 2010

Low values allow particles to roam far from the target regions before being tugged back. On the other hand, high values result in abrupt movement toward, or past, target regions. Hence, the acceleration constants are set to be 0.5. Suitable selection of inertia weight, w, provides a balance between global and local explorations, thus requiring less iteration on average to find a sufficiently optimal solution. We set w.to be 0.6.

4 Implementation and simulation results

In order to study performance of the proposed model, we implement PSO-RED by making some modifications on RED module of ns-2 software package. Consider a network with a set of 6 source nodes and a set of 6 destination nodes. As can be found in Fig. 1, our network model consists of a bottleneck link from LAN to WAN, which its bandwidth is 10 Mbps. Other links have the same bandwidth of 100 Mbps. Network sources use TCP Reno for congestion control. We assume that all connections are long-lived, have same RTT of 200 ms and unlimited bulk data for communication. We simulate this network once under RED algorithm and then under $PSO-RED$ algorithm to compare their performance in terms of bottleneck utilization, queue length, number of dropped packets and smoothness.

Fig. 2 shows the simulation results. In order to reference to the results of these figure, we note that:

1. Utilization: According to this figure, after the startup transient, PSO-RED's utilization of bottleneck link remains more than RED's utilization. Also, we can see in this figure that RED's utilization is more oscillating and sometimes drops remarkably and then returns.

2. Queue Evolution: As can be found in Fig. 2(b), while the queue length of RED always remains over 2 packets, PSO-RED's queue is kept in lower level i.e. below 2 packets. This means that queuing delay of PSO-RED is negligible.

3. Fairness: As we can see in Fig. 2(c), congestion windows sizes of all the network sources are equal (we have brought 4 congestion windows from 6 windows). Therefore bandwidth is shared equally among them, despite their heterogeneous initial state (max-min fairness).

4. Stability and Speed of Convergence: As we can see in Fig. 2, bot-

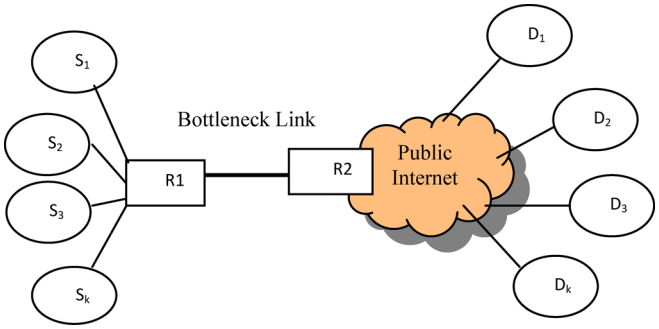
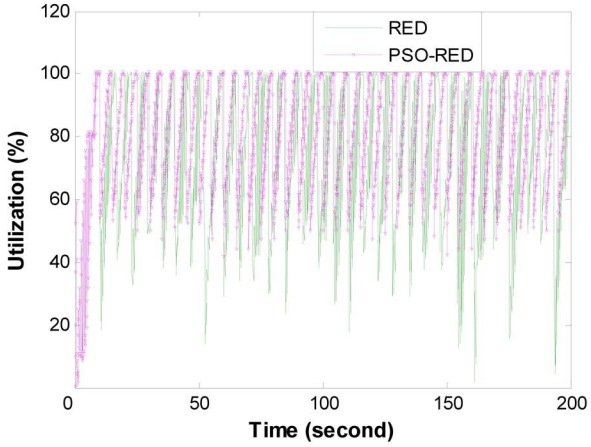
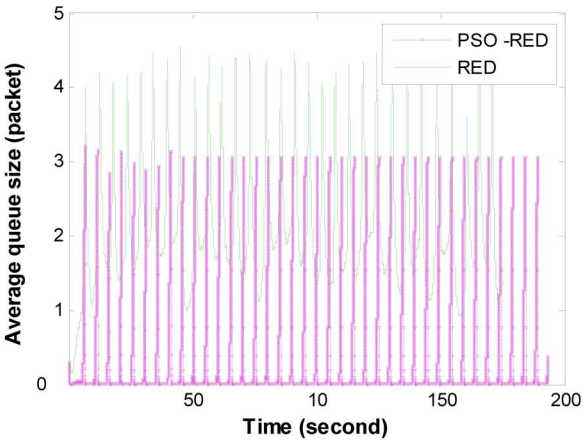


Fig. 1. General Network Model

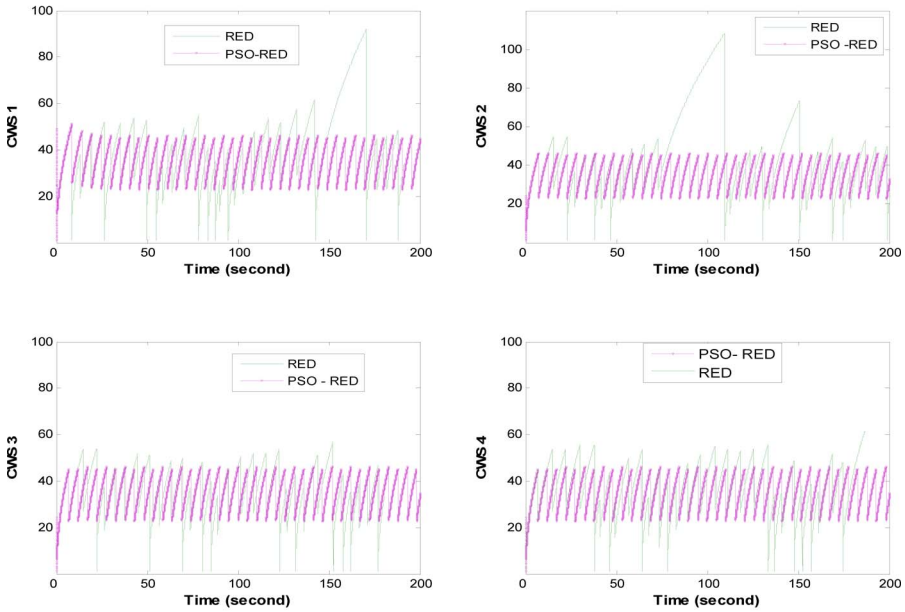
DOI: 10.1587/elex.7.1276
Received July 13, 2010
Accepted August 04, 2010
Published September 10, 2010



(a) Utilization



(b) Average queue length



(c) Stability

Fig. 2. Comparison of PSO-RED and RED algorithms in terms of (a) Utilization, (b) Queue length (c) Stability: PSO-RED behaves better than RED algorithm

DOI: 10.1587/elex.7.1276
Received July 13, 2010
Accepted August 04, 2010
Published September 10, 2010

Table I. Comparison of overall behavior of PSO-RED and RED algorithms

	Average Utilization	Average queue Size (bytes)	Drop Count
RED	73.01%	2095	329
PSO-RED	80.61%	354	273

tleneck utilization, queue length and congestion window sizes of PSO-RED have lower levels of oscillation in compare with RED.

You can find another view of comparison results between RED and PSO-RED in Tab. I. This table shows that average behavior of PSO-RED is better than RED in terms of utilization, drop rate and queue size. The highest improvement takes place in average queue size, where, it is 395 bytes for PSO-RED and 2095 bytes for RED. Taking into consideration the point that the size of packet is 1000 bytes, these average values for average queue sizes of RED and PSO-RED, match to instantaneous queue sizes, shown in Fig. 2(c). Note that according to Fig. 2(b), RED's queue size remains often around 2 packets and sometimes rises to 3 or 4 packets; hence, 2095 bytes are quite reasonable. In the same manner, 354 bytes is a reasonable average for queue size of PSO-RED that remains often near zero and rises sometimes to 2 or 3 packets.

5 Conclusion

In this paper we proposed PSO-RED algorithm as an improvement over RED active queue management scheme. The main feature of this algorithm is that it sets max_{th} and min_{th} parameters based on the network dynamic conditions. PSO-RED redefines the queue management issue as an optimization problem and tries to direct network to an optimum point in which queue length is low while bottleneck utilization remains over 0.8. For this purpose it uses PSO algorithm to solve the optimization problem. The simulating results indicate that the packet drop numbers and link utilization are better than RED algorithm.

DOI: 10.1587/elex.7.1276
Received July 13, 2010
Accepted August 04, 2010
Published September 10, 2010