doi: 10. 3772 / j. issn. 1002-0470. 2019. 05. 005

基于 S 型函数动态自适应改进的 RED 算法①

任金霞② 蒋梦倩③ 温春晖

(江西理工大学电气工程与自动化学院 赣州 341000)

摘 要 针对随机早期检测(RED)算法所存在的参数敏感性问题,本文提出了一种基于参数自适应动态调整的 RED 算法。针对 RED 算法的丢弃概率函数是线性的这一问题,利用 S 型升半哥西分布函数对传统 RED 算法的丢包率函数进行非线性处理,同时借鉴自适应 RED 算法调整最大丢弃概率的思想,并利用目标队长的范围和平均队列长度的关系引入参数自适应动态调整策略对最大丢包率进行改进。仿真结果表明改进算法在性能方面有较好的改善效果。

关键词 拥塞控制,随机早期检测(RED)算法, 参数敏感, 自适应

0 引 言

随着互联网、物联网、大数据和人工智能的兴起,网络拥塞问题成为提高网络服务质量的一大障碍。网络拥塞控制的研究浪潮也愈发高涨,大批研究者提出了各式各样的拥塞控制算法,其中主动队列管理算法中的随机早期检测(random early detection,RED) 算法是国际互联网任务组推荐的一种算法,有着较好的性能效果[1]。但是 RED 算法存在参数敏感性以及全局同步现象等缺陷,许多研究人员对 RED 算法进行了一系列的改进,成就了一系列 RED 的衍生算法,其中比较经典的衍生算法有自适应随机早期检测(adaptive random early detection, ARED) [2] 算法。在一定程度上,ARED 算法针对 RED 算法的参数敏感性做出了相应的改善,网络性能也得到较好的提升,但是由于算法引入了自身的两个参数 α 和 β, 且这两个参数都是静态且固定设置的,所以在改善 RED 算法的参数敏感性问题上,ARED 算法具有一定的局限性。因此,本文通过借鉴 ARED 算法参数自适应调整的思想,同时引入二倍最大阈值参数($2\max\_th$) 的思想,设计了一种新的参数自适应动态调整的 RED 改进算法。

1 RED 算法分析

1.1 传统 RED 算法分析

RED 算法是最早也是最著名的一个有效缓解网络拥塞的算法,RED 算法通过网络链路中的平均队列长度 $Qavg$ 大小来确定网络运行状态以及判断有无网络拥塞的出现,根据一定策略来求得该分组数据的丢包率[3]。RED 算法的原理图如图 1 所示。

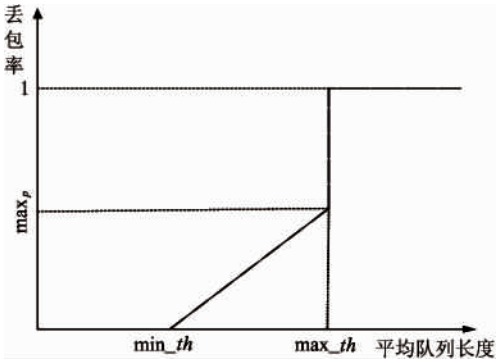

图 1 RED 算法原理图

① 江西省教育厅科学技术研究(GJJ150679) 资助项目。
② 女,1970 年生,硕士,副教授; 研究方向: 智能控制,机器学习; E-mail: dxzghyqq@ sina. com
③ 通信作者,E-mail: jmqwyc@ sina. com
(收稿日期: 2018-07-10)

其中 max _ *th* 和 min _ *th* 分别为 RED 算法的最大阈值上限和最小阈值下限，RED 算法的丢包策略主要通过比较平均队列长度和阈值上下限的关系进行丢包。当 *Qavg* < min _ *th* 时，属于无拥塞状况，此刻的分组数据无需丢包，可直接传输到路由队列中；当 *Qavg* 处在 min _ *th* 至max _ *th* 的范围内时，此时链路为拥塞状态，需要按一定的策略求得丢包率 *P*，并按丢包率 *P* 进行随机分组数据的丢弃；当 *Qavg* > max _ *th* 时，此时链路基本上处于完全拥塞状态，丢包率会直接上升到 1，即在下一刻到达的分组数据会全部丢弃。拥塞状态下，丢包率 *P* 表达式如式(1)所示：

$$P = \max_p \frac{(Qavg - \min\_th)}{(\max\_th - \min\_th)} \tag{1}$$

1.2 ARED 算法分析

ARED 算法的主要思想是根据网络负载情况自适应调整参数最大丢弃概率 $\max_p$。算法的基本原理是算法设定两个拥塞控制策略，分别为激进型和保守型[4]。选择哪一种拥塞策略模式主要看队列长度的变化来决定，当平均队列长度 *Qavg* 长时间在阈值下限 min _ *th* 附近上下浮动时，此时说明网络处于较好的网络状态中，通过激进的拥塞控制，减小 $\max_p$ 的值，此做法对队列的稳定更有帮助；当 *Qavg* 长时间在阈值上限 max _ *th* 处上下浮动时，此时由于链路环境处于拥塞的状况，通过保守型的控制策略，增大 $\max_p$ 的值，此做法对稳定队长更有帮助。所以 ARED 算法是根据 *Qavg* 的大小来对 $\max_p$ 进行自适应的调整，实现 *Qavg* 值稳定在阈值上下限范围之内[5]。ARED 算法的原理图如图 2 所示。

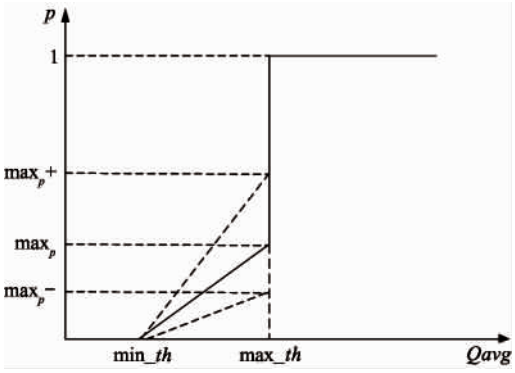

图 2 ARED 算法原理图

其中 $\max_p +$ 和 $\max_p -$ 分别表示激进增大后得到的最大丢弃概率和保守减小后得到的最大丢弃概率，表示如下：

$$\begin{cases} \max_p + = \max_p + \alpha \\ \max_p - = \max_p \times \beta \end{cases} \tag{2}$$

其中，α、β 分别表示增加因子和减少因子。

对整体来说，ARED 算法虽然针对 RED 算法的参数敏感性问题做出了一定的改善，但该算法引入了自身的参数 α 和 β，但 α 和 β 的取值问题同样还是算法本身的参数设置问题，其次在平均队列长度和阈值上下限的比较中，阈值 max _ *th* 和 min _ *th* 都是人为设定的，也存在着很多的不确定性。

2 RED 算法的改进

2.1 RED 算法非线性处理

由于 RED 算法中 *Qavg* 和丢包率是线性关系，这会造成 *Qavg* 值在阈值上下限 min _ *th* 和 max _ *th* 的附近时不能得到准确的丢弃概率，而且 *Qavg* 达到最大阈值的附近时，RED 算法的思想是将丢弃概率直接变为 1，这种情况下会造成队列的震荡，影响网络传输的稳定[6,7]。要使算法满足 *Qavg* 值在 min _ *th* 附近时采取缓慢增加 $\max_p$，*Qavg* 值在 max _ *th* 附近时迅速增加 $\max_p$ 的思想，就必须对原 RED 的丢弃概率线性函数进行平滑处理[8]。

本文采用 S 型升半哥西分布隶属函数对 RED 算法的 $\max_p$ 函数作非线性处理，它可以将 RED 算法的线性 $\max_p$ 函数变得更平滑。其中 S 型升半哥西分布函数如式(3)所示：

$$\mu_A(u) = \begin{cases} 0 & u \leqslant \alpha \\ a(u-\alpha)^{\beta}/(1+(u-\alpha)^{\beta}) & u \geqslant \alpha,\ a > 0,\ \beta > 0 \end{cases} \tag{3}$$

式中，$\mu_A(u)$ 表示函数隶属度，α 和 β 为参数，a 值依据参数 α 和 β 的值确定；将上式结合 RED 丢包率函数，令 u 为平均队列长度 *Qavg*，α 为阈值下限 min _ *th*，β 值为 3，则可得到：

$$P_b = \begin{cases} 0 & Qavg \leqslant \min\_th \\ a(Qavg - \min\_th)^{\beta}/(1 + a(Qavg - \min\_th)^{\beta}) & \min\_th < Qavg < 2\max\_th \\ 1 & Qavg \geqslant 2\max\_th \end{cases} \quad (4)$$

由式(4) 可知,当 RED 算法中平均队列长度 $Qavg$ 等于最大阈值 $\max\_th$ 时,则可以求得 a 的值,如式(5) 所示。

$$a = \max_p/((1 - \max_p)(\max\_th - \min\_th)^{\beta}) \quad (5)$$

把式(5) 中的 a 重新返回式(4) 中,可得到升半哥西分布函数处理后的 RED 丢弃概率函数,如式(6) 所示。

$$P_b = \begin{cases} 0 & Qavg \min\_th \\ \dfrac{\max_p(Qavg - \min\_th)^{\beta}}{(1 - \max_p)(2\max\_th - \min\_th)^{\beta} + \max_p(Qavg - \max\_th)^{\beta}} & \min\_th < Qavg < 2\max\_th \\ 1 & Qavg \geqslant 2\max\_th \end{cases} \quad (6)$$

另外,通过对 RED 作非线性处理,可以得到改进 RED 算法的原理图,如图 3 所示。

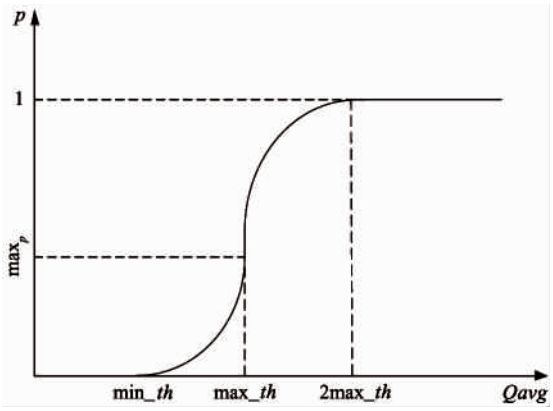

图 3　改进 RED 算法的原理图

2.2　RED 算法动态自适应改进

在 ARED 算法中,虽然也实现了对参数 $\max_p$ 的自适应调整,但是 ARED 算法引进了两个新的参数,而且这些参数都是人为设定的,所以从根本上说,ARED 算法还是属于静态固定参数的设置,并没有实现动态的自适应改进[9,10]。由于动态自适应改进是依据 ARED 算法的思想,并利用目标队列长度范围和平均队列长度的关系对 RED 算法的 $\max_p$ 进行动态调整[11]。ARED 算法的目标队列长度范围为式(7)。

$$target = [\min\_th + 0.4(\max\_th - \min\_th), \min\_th + 0.6(\max\_th - \min\_th)] \quad (7)$$

令式(7) 中的 $\min\_th + 0.4(\max\_th - \min\_th)$ 用参数 A 表示,$\min\_th + 0.6(\max\_th - \min\_th)$ 用参数 B 表示,则目标队列长度的范围则为 $[A,B]$。ARED 算法中对 $\max_p$ 取值所采取的方法是和式减少积式增加法,则 $\max_p$ 可表示为式(8)。

$$\max_p = \begin{cases} \max_p + (Qavg - B)/(2\max\_th - \min\_th) & Qavg > B \\ \max_p & A \leqslant Qavg \leqslant B \\ \max_p \times (1 - (A - Qavg)/(2\max\_th - \min\_th)) & Qavg < A \end{cases} \quad (8)$$

根据 ARED 算法的自适应调整丢包率公式可知,ARED 算法引入的参数 α 和 β 就相当于式(8) 中的 $(Qavg - B)/(2\max\_th - \min\_th)$ 和 $(1 - (A - Qavg)/(2\max\_th - \min\_th))$,所以实现了对参数 $\max_p$ 的动态调整。

3　算法仿真实例分析

针对本文改进的 RED 算法,利用网络仿真工具 NS2 设计实验,并对其进行有效性验证,本文对传统 RED 算法和 ARED 算法的各性能比较结果,并逐一分析。实验主要从平均队列长度、吞吐量、端到端时延、时延抖动和丢包率 5 方面进行分析。

网络的拓扑结构模型采用常用的哑铃状结构,如图 4 所示。

发送端 S1,S2 和接收端 D1,D2 节点的带宽为 12 Mbps,时延为 2 m/s。网络中的路由节点为 R0 和 R1,R0 与 R1 之间的带宽为 1.5 Mbps,时延为 20 m/s。R0 和 R1 采取的链路队列管理算法分别为

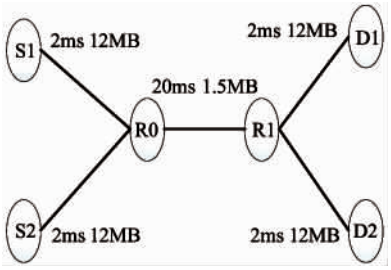

图 4　网络拓扑结构

RED 算法和 ARED 算法。其余节点采取的是 *DropTail* 算法，链路最大容量都为 10。发送端节点 S1、S2 采取的数据流类型均为 TCP。仿真时间为 50 s。

各算法参数设置如下：

RED 算法：$\min\_th = 10$，$\max\_th = 30$，$Wq = 0.02$，$\max_p = 0.1$，其余参数为 NS2 默认参数设置。

ARED 算法：$\alpha = 0.01$，$\beta = 0.9$，其余参数与 RED 算法设置相同。

改进 RED 算法与 RED 算法的参数设置相同。

（1）平均队列长度的比较

平均队列长度及平均队列长度的变化是评价一个网络性能的重要的指标，它可反映网络服务质量的好坏以及链路使用率的高低。从图 5 明显看出，对比 RED 算法和 ARED 算法，改进的 RED 算法的平均队长的波动幅度非常低，图中显示改进的 RED 算法的平均队列长度在 15 s 附近时逐渐趋于一条直线，这说明平均队长的波动幅度小，进而系统链路的利用率高，即性能更稳定，从而可以推出本文的改进 RED 算法具有更好的控制效果。

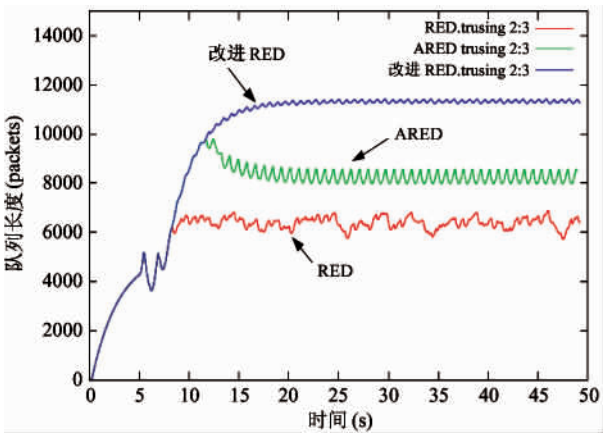

图 5　3 种算法的平均队列长度的比较

（2）吞吐量的比较

由图 6 可知，在一定的时间范围内，3 种算法都有着较高的吞吐量，并且，随着仿真时间的增加，大概在7 s左右，3 种算法的曲线有所下降并趋于稳定值，但 ARED 算法较 RED 算法保持有微高的吞吐量，相比于这两种算法，改进 RED 算法具有明显高的吞吐量。较高的吞吐量可以使网络的流畅性更好，从而改善网络的性能。

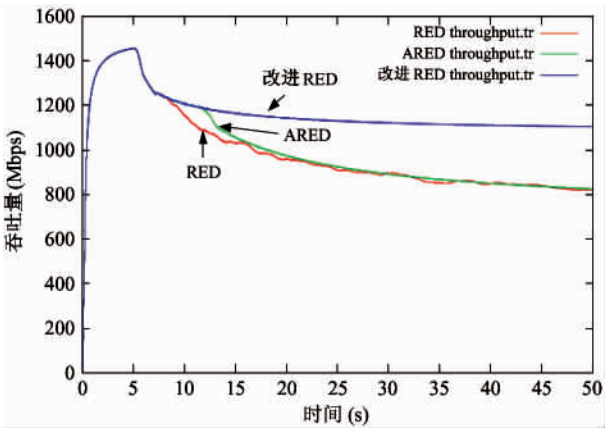

图 6　3 种算法的吞吐量比较

（3）端到端时延的比较

从图 7、图 8 和图 9 中可以看出，ARED 算法的端到端时延最大，改进的 RED 算法次之，RED 算法的端到端时延最小。改进的 RED 算法比 RED 算法时延大主要是由于 RED 算法的全局同步现象导致的，因为改进的 RED 算法的平均队列长度波动更小，易造成全局同步现象，而全局同步现象会造成时延增大。时延包括传输时延和排队时延，所以从 3 种算法的时延比较中可知，在传输时延一定的情况下，数据包在队列瓶颈路由缓冲区中 RED 算法的排

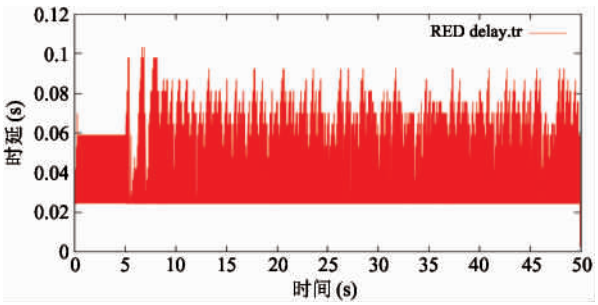

图 7　RED 算法的端到端时延

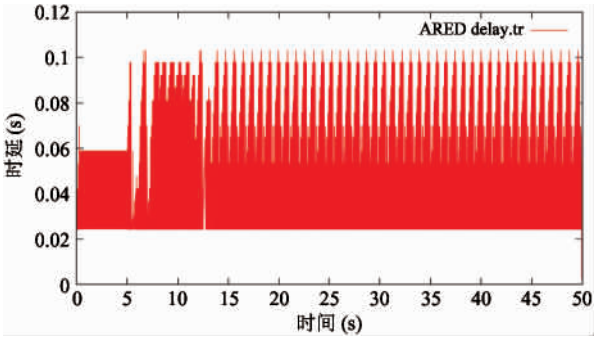

图 8　ARED 算法的端到端时延

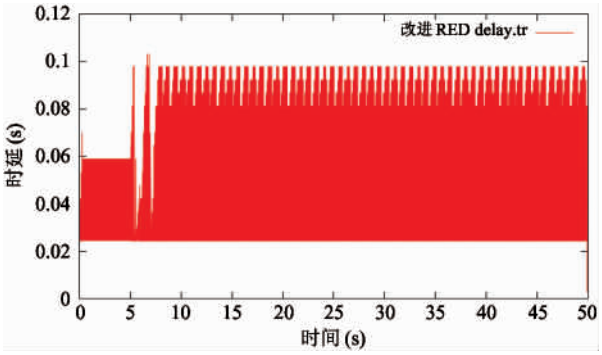

图 9　改进 RED 算法的端到端时延

队时延比 ARED 算法和改进的 RED 算法的时延更小。

(4) 时延抖动的比较

从图 10、图 11 和图 12 中可看出,RED 算法的时延抖动最大,其次是 ARED 算法,改进的 RED 算法的时延抖动最小。时延抖动是数据包时延的变化量,时延抖动的大小,反映了一个算法在系统网络中的稳定性,时延抖动越小,网络稳定性越好,反之,时延抖动越大,网络稳定性越差。通过以上分析并结合图中数据可得出,本文所采用的改进 RED 算法在 3 种算法中网络稳定性最好。

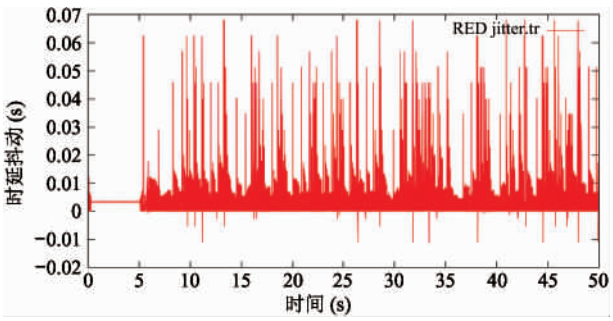

图 10　RED 算法的时延抖动

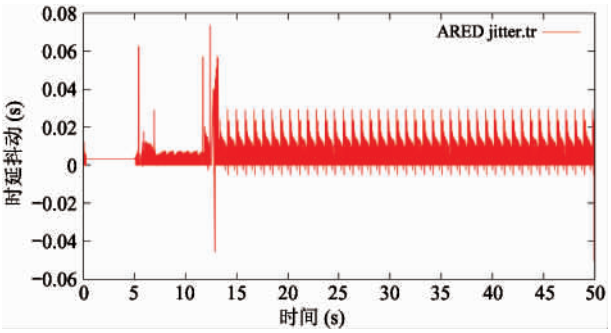

图 11　ARED 算法的时延抖动

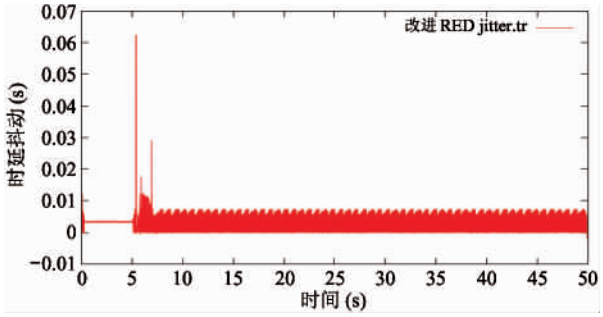

图 12　改进 RED 算法的时延抖动

(5) 丢包率比较

表 1 数据表明,相比 RED 算法,ARED 算法有较低的丢包率,而改进的 RED 算法丢包率最低。丢包率是拥塞控制性能评价指标中一个重要的指标,对用户而言,更低的丢包率可说明数据资源丢失有所减少,一般在发生拥塞时,链路中的拥塞控制算法都会对数据进行丢包处理,来解决网络拥塞。而丢包率的降低,意味着拥塞得到了更有效的解决,从侧面也反映出了拥塞的处理情况。本文改进的 RED 算法的丢包率比 RED 算法和 ARED 算法都有较明显的减小,表明了本文改进 RED 算法的有效性。

表 1　丢包率的比较

	RED	ARED	改进 RED
丢包数	62	49	3
到达包数	4 940	4 956	6 637
丢包率(%)	1.25	0.98	0.05

4　结　论

本文提出的基于 S 型函数动态自适应改进的

RED 算法在丢包率、吞吐量、时延抖动和平均队列长度等指标方面均比 RED 算法和 ARED 算法有较大的提高,改进的 RED 算法有较好的网络性能。

参考文献

[1] Floyd S, Jacobson V. Random early detectio-n gateways for congestion avoidance. *IEEE/ACM Transaction on Networking*. 1993, 1(4): 397-413

[2] Feng W, Kandlur D D, Saha D, et al. A se-lf-configuring RED gateway[C]. In: Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies, New York, USA, 1999. 1320-1328

[3] 黄迎春, 李向丽, 邱保志. 一种改进的 RED 算法[J]. 计算机工程,2007, 33(1): 117-121

[4] 饶刚, 周井泉. 基于 ARED 的主动队列管理改进算法[J]. 计算机技术与发展, 2014,24(05): 27-30

[5] Chen J Y, Hu C Y, Zhen J. An improved RED algorithm for congestion control of network transmission[J]. *Mathematical Problems in Engineering*, 2010, article ID 329035

[6] Abbasoy, Babek, Korukoglu, et al. Effective RED: An algorithm to improve RED's performance by reducing packet loss rate[J]. *Journal of Network and Computer Applications*, 2009, 32(3): 703-709

[7] Liu Z, Zhang Y, Philip Chen C L. Adaptive mechanism-based congestion control for networked systems[J]. *International Journal of Systems Science*, 2013, 44(3): 533-544

[8] Shubhangi R, Hira Z. Comparative analysis of queuing mechanisms: Droptail, RED and NLRED[J]. *Social Network Analysis and Mining*, 2016, 6(1): 123-232

[9] 任丰原, 林闯, 王福豹. RED 算法的稳定性: 基于非线性控制理论的分析[J]. 计算机学报, 2002, 25(12): 1302-1307

[10] 贾永库. 基于非线性自适应 RED 算法的网络拥塞控制研究[D]. 西安: 西北大学信息科学与技术学院, 2010

[11] Zhang J J, Xu W L, Li G W. An improved active queue management algorithm based on nonlinear smoothing[J]. *Advanced Materials Research*, 2011, 1336(295): 1823-1828

An improved random early detection algorithm based on parameter adaptive dynamic adjustment

Ren Jinxia, Jiang Mengqian, Wen Chunhui

(School of Electrical Engineering and Automation, Jiangxi University of Science and Technology, Ganzhou 341000)

Abstract

Aiming at the parameter sensitivity problem of RED algorithm, this paper proposes a random early detection (RED) algorithm based on parameter adaptive dynamic adjustment. In view of the problem that the discarding probability function of the RED algorithm is linear, it uses the S type ascending semi Cosi distribution function for the nonlinear processing of the packet loss rate function of the traditional RED algorithm, and uses the adaptive RED algorithm to adjust the maximum discarding probability, and introduces the parameter adaptation by using the relation of the range of the target queue and the length of the queue. The dynamic adjustment strategy improves the maximum packet loss rate. Simulation results show that the improved algorithm has better performance in terms of performance.

Key words: congestion control, random early detection (RED) algorithm, parameter sensitivity, adaptive