基于 ARED 的主动队列管理改进算法

饶 刚,周井泉
(南京邮电大学 电子科学与工程学院,江苏 南京 210003)

摘 要:随机早期检测(Random Early Detection,RED)是 IETF 推荐部署的主动队列管理(Active Queue Management,AQM)算法。RED 存在参数难以配置、无法适应动态网络环境的缺点。ARED(Adaptive RED)是 RED 的自适应版本,通过平均队列长度来动态调整最大丢弃概率,从而达到稳定平均队列长度的目的,但是存在瞬时队列长度振荡的问题。文中研究了拥塞控制中的主动队列管理,对 ARED 算法进行了改进,优化丢弃概率计算函数,提出 TTS-ARED 算法,实现在动态网络环境下队列长度的稳定以及丢包率降低。NS2 的仿真结果表明,TTS-ARED 算法显著地降低了丢包率,队列长度稳定性比 ARED 算法更优越。

关键词:拥塞控制;主动队列管理;ARED
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2014)05-0027-04
doi:10.3969/j.issn.1673-629X.2014.05.007

Improved Active Queue Management Algorithm Based on ARED

RAO Gang,ZHOU Jing-quan
(College of Electronic Science and Engineering,Nanjing University of Posts & Telecommunications,Nanjing 210003,China)

Abstract:Random Early Detection (RED) is the Active Queue Management (AQM) algorithm recommended by IETF. RED is difficult to configure its parameters,and can't adapt to dynamic network. ARED is an improved adaptive RED,by measuring average queue size to dynamically adjust maximum drop rate,achieving stable average queue size. But it may cause instantaneous queue size oscillation. It researched the active queue management in congestion control,and some improvements are made based on ARED algorithm. It optimized the calculation of dropping probability and a new algorithm TTS-ARED is proposed,low drop rate and the stable queue size in dynamic network are realized. The simulation results indicate that the drop rate can be decreased significantly by using TTS-ARED,and in the area of stabilizing queue size,TTS-ARED algorithm is superior to RED algorithm.

Key words:congestion control;active queue management;adaptive random early detection

0 引 言

随着互联网的规模扩大以及网络数据流量的激增,仅仅依靠端系统来提供拥塞控制是远远达不到要求的。首先,互联网是复杂、状态多变的系统,在这样的系统中,期望所有的网络应用都能实现端到端的拥塞控制机制是不现实的;另外,端到端的拥塞控制具有一定的滞后性,发送端检测到拥塞和拥塞发生之间存在着时间间隔,在这时间间隔内,网络会一直处于拥塞状态;所以,让网络参与到拥塞控制中成了不二的选择。当前的网络节点中使用的拥塞控制机制主要分为两类:队列管理和队列调度,其中队列管理又分为被动队列管理(Passive Queue Management,PQM)和主动队列管理(Active Queue Management,AQM)[1-2]。

被动队列管路算法中最典型的是“弃尾算法”(Drop Tail)[3],算法的优点是简单、容易实施,缺点是持续满队列、全局同步[4]和死锁。最典型的 AQM 算法是由 Floyd 等在 1993 年提出的随机早期检测(Random Early Detection,RED)算法[5-6]。能够让路由器的缓冲队列在没有达到极限值之前就开始丢包,并向源端发送拥塞反馈信息,来达到缓解网络拥塞的目的。

文中对 RED 的改进算法 ARED 进行优化,针对 ARED 在队列长度稳定性方面的缺陷,利用三次方函

收稿日期:2013-07-12 **修回日期**:2013-10-18 **网络出版时间**:2014-02-11
基金项目:江苏省自然科学基金项目(CXLX12_0471)
作者简介:饶 刚(1989-),男,江苏南京人,硕士研究生,研究方向为复杂网络与系统;周井泉,博士,教授,硕士生导师,研究方向为通信网络的信息管理和控制。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20140211.1453.025.html

数(Three Times Square,TTS)优化丢弃概率计算函数,提出了 TTS-ARED 算法。通过 NS2 仿真实验表明:该算法能够在队列稳定性方面表现更好,同时降低了丢包率。

1 典型主动队列管理算法原理

1.1 RED 算法原理

RED 算法通过使用指数加权平均算法来计算瞬时队列长度,得到平均队列长度 avg(t),以此来表征队列的拥塞和拥塞的程度。通过 avg(t)与预先设定的两个门限阈值 min_th 和 max_th 进行对比,当 avg(t)< min_th 时,接收到达的分组;当 avg(t)>max_th 时,丢弃或标记所有到达的分组;当 min_th<avg(t)<max_th 时,按照丢弃概率 Pa 计算新到达分组的丢弃、标记概率。RED 算法主要分为两个部分:

(1)计算平均队列长度。

$$\mathrm{avg}(t+1)=(1-w_q)*\mathrm{avg}(t)+w_q*q \quad (1)$$

其中,w_q 为队列权值,它表征 RED 算法对数据流输入的敏感程度;q 为实时队列长度。

(2)计算丢弃概率函数:

$$\mathrm{Pb}=\begin{cases}0 & \mathrm{avg}(t)<\mathrm{min\_th}\\ \dfrac{\mathrm{avg}(t)-\mathrm{min\_th}}{\mathrm{max\_th}-\mathrm{min\_th}}\max_p & \mathrm{min\_th}<\mathrm{avg}(t)<\mathrm{max\_th}\\ 1 & \mathrm{avg}(t)>\mathrm{max\_th}\end{cases} \quad (2)$$

$$\mathrm{Pa}=\frac{\mathrm{Pb}}{1-\mathrm{count}*\mathrm{Pb}} \quad (3)$$

其中,$\max_p$ 为最大丢弃概率;count 表示 avg(t)在最大、最小阈值之间时,进入到队列中的数据包的个数。使用 Pa 来丢包是因为这样可以使得丢包随着时间均匀分布,避免出现连续丢包的现象,采用随机丢包,而并不是通知所有的源端,可以避免产生全局同步现象。

RED 算法最大的缺点在于对参数设置的敏感,细微的参数变动都会对网络的性能造成很大的影响,并不能很好地适应动态的网络环境。

1.2 ARED 算法原理

ARED 算法[7-8]是研究者提出的一种根据平均队列长队的变化自适应调节最大丢弃概率 $\max_p$ 的算法。考虑到在 RED 算法中 $\max_p$ 是人为设置的,如果 $\max_p$ 设置过大,会因为早期拥塞检测过于严格而频繁丢包。相反,如果 $\max_p$ 设置过小,会出现路由器队列长期处于接近 max_th 的满队列状态。ARED 算法在 RED 算法的基础上,利用 RED 算法计算平均队列长度和丢弃概率计算方法。通过检查平均队列长度的变化来动态调整 $\max_p$ 的值,在对 $\max_p$ 进行更新时采用和式增加积式减小(MIMD)的方式,从而有效的进行拥塞控制。

具体算法如下

every interval seconds

if(avg(t) > target && $\max_p \leqslant 0.5$)

$\max_p = \max_p + \beta$;($\beta = \min(0.01, \max_p/4)$)

else if(avg(t) < target && $\max_p \geqslant 0.01$)

$\max_p = \max_p * \alpha$;($\alpha = 0.9$)

其中,interval 为丢弃概率改变的时间间隔,一般取 0.5 s;target 来表示平均队列的理想区间范围,取 min_th+0.4*(max_th-min_th),min_th+0.6(max_th-min_th)之间;α、β 分别表示 $\max_p$ 减小、增大的值。

2 TTS-ARED 算法原理

ARED 虽然解决了 RED 中的参数设置问题,但它本身又引入三个参数,产生了新的参数设置问题;并且在动态调整 $\max_p$ 的过程中对队列的稳定度造成了很大的影响。文中提出的 TTS-ARED(Three Times Square-ARED)算法就是在修改丢弃概率计算函数的基础上达到对队列长度稳定性的巩固。

TTS-ARED 算法主要通过对丢弃函数 Pb 的计算方法的改进,采用三次方函数来计算 Pb。当平均队列长度在 ξ($\xi = 0.5 * (\mathrm{min\_th} + \mathrm{max\_th})$)附近时,$\max_p$ 变化率比较平缓,而当平均队列长度接近 min_th 或者 max_th 时,$\max_p$ 的变化比较剧烈,这样能够很好地将队列长度稳定在目标区间内,同时 $\max_p$ 的变化相对平缓,对瞬时队列长度的影响降低,可以有效地稳定队列长度。TTS-ARED 算法的丢弃概率曲线和 Pb 计算公式分别如图 1 和式(4)所示。

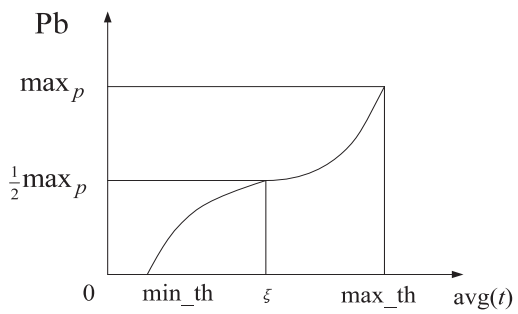

图 1 TTS-ARED 丢弃概率计算方法

$$\mathrm{Pb}=\begin{cases}0 & \mathrm{avg}(t)<\mathrm{min\_th}\\ \dfrac{1}{2}\max_p+\dfrac{1}{2}\max_p*\dfrac{(\mathrm{avg}(t)-\mathrm{min\_th})^3}{(\mathrm{max\_th}-\mathrm{min\_th})^3} & \mathrm{min\_th}<\mathrm{avg}(t)<\mathrm{max\_th}\\ 1 & \mathrm{avg}(t)>\mathrm{max\_th}\end{cases} \quad (4)$$

3　TTS-ARED 算法的仿真和分析

利用 NS2 软件对 TTS-ARED、ARED 算法进行仿真分析[9-11]。仿真的网络拓扑结构如图 2 所示。

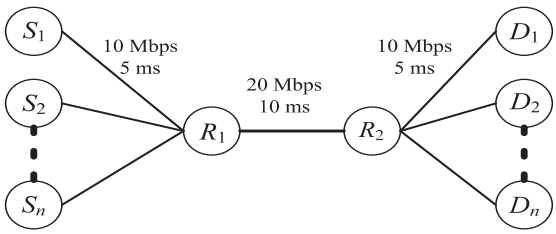

图 2　仿真使用的网络拓扑结构

路由器 R_1 和 R_2 之间为瓶颈链路,容量 20 Mbps,时延为 10 ms。$S_1,\cdots,S_n$ 为发送源端,全部为 FTP 流,源端到路由器 R_1 的链路容量为 20 Mbps,时延为 5 ms。$D_1,\cdots,D_n$ 为接收端节点,到路由器 R_2 的链路容量为 10 Mbps,时延为 5 ms。路由器 R_1 的队列使用 TTS-ARED、ARED 算法,其余队列均使用 DropTail 算法,所有网络节点的缓冲区大小为 300 packets。所有仿真实验 TTS-ARED、ARED 算法参数选择如下:Wp = 0.02,$\max_p = 0.1$,min_th = 100,max_th = 200[12]。

3.1　队列长度的稳定性比较

如图 2 的网络拓扑结构,开始启动 80 个 TCP 源,持续时间为 100 s,对瓶颈链路队列长度仿真结果如图 3、4 所示。

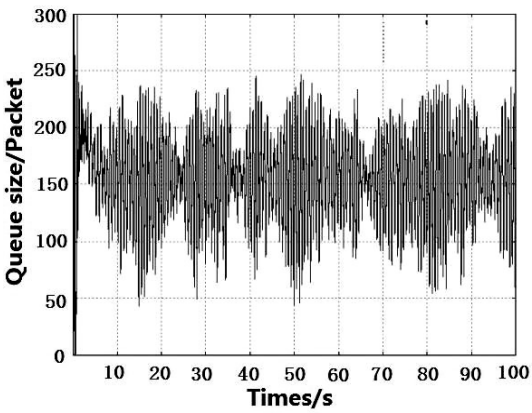

图 3　ARED 队列长度(1)

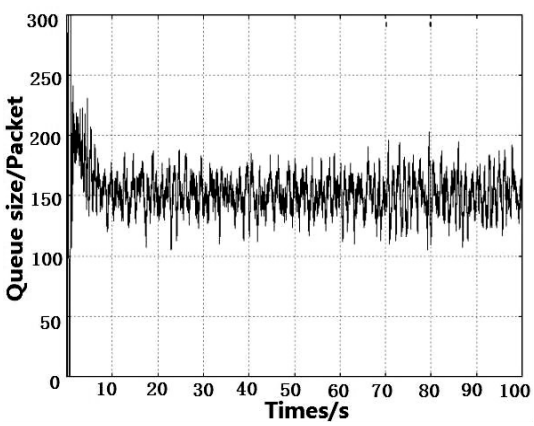

图 4　TTS-ARED 队列长度(1)

本组实验来比较 ARED 算法和 TTS-ARED 算法在队列长度稳定性方面的性能。从图中的对比可以明显看出,在 80 个 TCP 连接的情况下,TTS-ARED 在队列长度稳定方面明显好于 ARED,队列长度稳定在 150 左右,波动范围较小。而 ARED 实时队列长度波动范围为[100,200],波动较大。所以 TTS-ARED 算法在稳定队列长度方面更有优势。

3.2　有突发流存在的情况下队列长度的稳定性

如图 2 的网络拓扑结构,开始时启动 80 个 TCP 流,在 30 s 时增加 40 个 TCP 流,在 60 s 时减少 40 个 TCP 流,恢复为 80 个 TCP 流,时间持续 100 s,如图 5、6 所示。从图中可以看出,ARED 算法的队列长度波动性比较大,队列长度出现明显振荡,TTS-ARED 算法表现相对稳定,只在 30 s 和 60 s 附近出现短暂的较大波动,但是过渡时间很短,能够迅速稳定。可以看出 TTS-ARED 算法在稳定队列长度方面优于 ARED 算法,具有更好的鲁棒性。

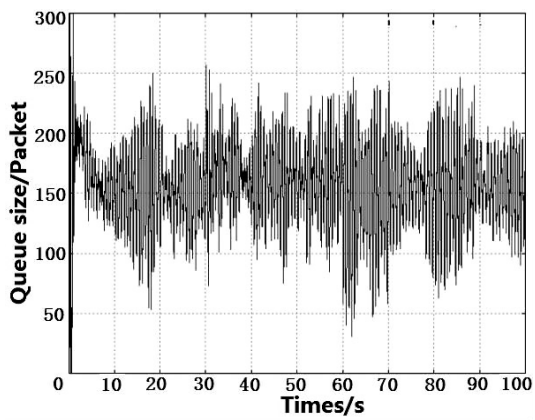

图 5　ARED 队列长度(2)

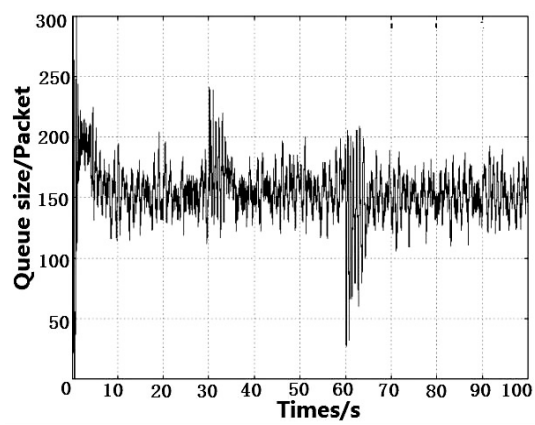

图 6　TTS-ARED 队列长度(2)

3.3　算法丢包率比较

文中选择使用丢包率来评价算法的性能。通过仿真对 ARED、TTS-ARED 算法进行了分析比较,具体结果如表 1 所示。

从表中可以看出,在相同 TCP 连接数的情况下,TTS-ARED 的丢包率要小于 ARED 算法,明显降低了

丢包率。TTS-ARED 算法表现明显优于 ARED 算法。

表 1 ARED、TTS-ARED 算法丢包率对比

TCP 连接数	ARED		TTS-ARED	
	丢包数/发送数	丢包率/%	丢包数/发送数	丢包率/%
80	21 018/442 313	4.75	19 347/440 644	4.39
140	41 341/463 427	8.92	39 819/461 888	8.62

4 结束语

主动队列管理技术,作为一种重要的拥塞控制机制,对于网络服务质量的提高起到了很大的作用。文中鉴于 ARED 算法的不足,在此基础上提出了改进的 ARED 算法 TTS-ARED。通过改进算法的丢弃计算函数,实现了队列长度的稳定,并且提高了算法在复杂网络环境下队列长度的稳定性,增强了算法的鲁棒性。通过利用 NS2 软件对算法进行了仿真分析,实验结果表明改进的 TTS-ARED 算法在队列稳定性和丢包率等方面都明显优于 ARED。

改进算法仍需要进一步研究,算法本身的参数设置是根据经验值设置的,并不一定能够适应普遍动态的网络环境,对参数设置以及自适应方面的研究将是接下来的研究重点。

参考文献:

[1] Braden B, Clark K, Crowcroft J, et al. Recommendations on queue management and congestion avoidance in the Internet [EB/OL]. 1998. http://www.rfc.net/rfc2309.html.

[2] Stanojevic R, Shorten R N, Kellett C M. Adaptive tuning of drop-tail buffers for reducing queueing delays [J]. IEEE Communications Letters, 2006, 10(7): 570-572.

[3] Jacobson V. Congestion avoidance and control [C]//Proc of SIGCOMM'88 symposium on communications architectures and protocols. New York, NY, USA: ACM, 1988: 314-329.

[4] Hashem H. Analysis of random drop for gateway congestion control [R]. Cambridge, MA: Laboratory for Computer Science, MIT, 1989.

[5] Floyd S, Jacobson V. Random early detection gateways for congestion avoidance [J]. ACM/IEEE Trans on Networking, 1993, 1(4): 397-413.

[6] Guan L, Awan I U, Woodward M E. Discrete-time performance analysis of a congestion control mechanism based on RED under multi-class bursty and correlated traffic [J]. Journal of Systems and Software, 2007, 80(10): 1716-1725.

[7] Floyd S, Gummadi R, Shenker S. Adaptive RED: An algorithm for increasing the robustness of RED's active queue management [EB/OL]. 2001. http://www.cs.berkely.edu/.

[8] Kim Tae-Hoon, Lee Kee-Hyun. Refined adaptive RED in TCP/IP network [C]//Proc of IEEE ICASE. [s.l.]: [s.n.], 2006: 3722-3725.

[9] 王秀利. AQM 算法在 NS2 中的实现及其性能评价 [J]. 计算机科学, 2009, 36(5): 60-64.

[10] 方路平, 刘世华. NS2 网络模拟基础与应用 [M]. 长沙: 国防科技大学出版社, 2008.

[11] 柯志亨, 程荣祥. NS2 仿真实验-多媒体和无线网络通信 [M]. 北京: 电子工业出版社, 2009.

[12] 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.

(上接第 26 页)

[5] 张荣华, 田 泽, 韩 炜. AFDX 网络端系统芯片架构的研究与设计 [J]. 计算机技术与发展, 2011, 21(8): 165-168.

[6] 张 志, 翟正军, 姚方圆. 基于 FPGA 的 AFDX 端系统协议芯片的设计与实现 [J]. 计算机测量与控制, 2010, 18(2): 422-424.

[7] Grieu J. Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques [D]. France: INP-ENSEEIHT, 2004.

[8] 张奇智, 张 彬, 张卫东. 基于网络演算计算交换式工业以太网中的最大时延 [J]. 控制与决策, 2005, 20(1): 117-120.

[9] Bauer H, Scharbarg J, Fraboul C. Improving the Worst-Case Delay Analysis of an AFDX network using an optimized trajectory approach [J]. IEEE Transactions on Industrial Informatics, 2010, 6(4): 521-533.

[10] Charara H, Scharbarg J L, Ermont J, et al. Methods for bounding end-to-end delays on an AFDX network [C]//Proceedings of the 18th Euromicro conference on real-time systems. Washington, DC, USA: IEEE Computer Society, 2006.

[11] Bauer H, Scharbarg J, Fraboul C. Applying and optimizing trajectory approach for performance evaluation of AFDX avionics network [C]//Proc of IEEE conference on emerging technologies & factory automation. Mallorca: [s.n.], 2009: 1-8.

[12] Boyer M, Fraboul C. Tightening end to end delay upper bound for AFDX network calculus with rate latency FIFO servers using network calculus [C]//Proc of 2008 IEEE international workshop on factory communication systems. Dresden: [s.n.], 2008: 11-20.

基于AQED的主动队列管理改进算法

作者： 饶刚， 周井泉， RAO Gang， ZHOU Jing-quan
作者单位： 南京邮电大学 电子科学与工程学院,江苏 南京,210003
刊名： 计算机技术与发展 ISTIC
英文刊名： Computer Technology and Development
年，卷(期)： 2014(5)