改进的 RED 队列管理算法:RED-r*

姜文刚[1]，孙金生[2]，王执铨[2]
(1. 江苏科技大学 电子信息学院，江苏 镇江 212003；2. 南京理工大学 自动化学院，南京 210094)

摘　要：为了避免 RED 缺陷,提出一种改进的 RED 算法——RED-r。该算法采用二次圆函数来计算丢包概率,减少了 RED 的设置参数,实现了在网络大延时和小延时时的队列稳定,且在小延时能获得比 PID 队列更平滑的效果。NS2 仿真验证了 RED-r 算法的有效性。

关键词：RED；队列管理；网络拥塞控制；RED-r；PID
中图分类号：TP393　　**文献标志码**：A　　**文章编号**：1001-3695(2012)07-2632-03
doi:10.3969/j.issn.1001-3695.2012.07.063

New revised queue management algorithm of RED: RED-r

JIANG Wen-gang[1], SUN Jin-sheng[2], WANG Zhi-quan[2]
(1. *School of Electronic & Information, Jiangsu University of Science & Technology, Zhenjiang Jiangsu* 212003, *China*; 2. *School of Automation, Nanjing University of Science & Technology, Nanjing* 210094, *China*)

Abstract: In order to avoid the flaws of RED, this paper presented a new revised RED algorithm named RED-r. It used the quadratic round function to calculate the probability of dropping packets, then reduced the amount of RED parameters, the queue remained stable in great or small delay network, and achieved a more smooth effect with small delay than the PID queue. The NS2 simulations validate the effectiveness of the RED-r algorithm.

Key words: RED; queue management; network congestion control; RED-r; PID

0　引言

TCP/IP 拥塞控制主要包括两个方面,一是发送端的拥塞控制,称为源端拥塞控制,二是中间节点的拥塞控制,称为队列管理。TCP 拥塞控制机制由 Jacobson[1] 1988 年提出,包括慢启动和拥塞避免算法,就是和式增加积式减少(additive increase multiplicative decrease,AIMD)[1]。1990 年在 TCP Reno 版本中增加了快速重传、快速恢复,TCP NewReno 对快速恢复进行了改进。总体上 TCP 通过对这四个算法设置不同的参数来实现不同 TCP 拥塞控制[2]。

队列管理包括被动队列管理(passive queue management,PQM)和主动队列管理(active queue management,AQM)[3]。仅依靠 TCP 拥塞控制还不能解决网络拥塞问题,还必须结合网络中间节点的队列管理方法,两者一起来改善网络拥塞状况。队列管理通过丢弃或标记数据包来控制中间节点的队列长度。TCP 发送端通常通过检测数据包的丢弃或标记,来获得网络的拥塞状况,从而降低发送速度。队列管理算法是在拥塞的队列当中实施的,能够最先检测到网络的拥塞状态,并通过丢包或标记向 TCP 发送端发送拥塞信息。队列管理与 TCP 发送端共同作用,快速有效地实现网络拥塞控制。

被动队列管理在缓冲溢出时才丢弃或标记数据包,所以说是一种被动的队列管理方式。在实际网络中,中间节点中最常用的队列管理策略是弃尾(drop tail),即随着缓冲区的溢出而在队列尾部丢包,是一种被动队列管理机制。弃尾的缺陷包括数据流的全局同步、死锁(lock-out)、缓冲区队列长度振荡以及持续队满造成的延迟较大等[4,5]。

1993 年,Floyd 等人[6] 就提出了一种 AQM 算法——随机早期检测 RED 算法。1998 年,Braden 等人[3] 提出了主动队列管理的研究协议,而 RED 是当时唯一能实现其目标的主动队列管理算法,所以 RFC 2309 中将其推荐为 AQM 的唯一候选算法。1999 年,Floyd[7] 提出改进的 RED 算法 GentleRED,当平均队列长度大于 RED 设置的队列长度上限时,丢包/标记概率是线性增长到 1,而不是突变到 1。RED、GentleRED 算法的一个缺陷就是平均队列长度对拥塞程度和参数设置很敏感。如果拥塞不太严重或者最大丢包概率设置很大,则平均队列接近下限阈值;如果拥塞比较严重或最大丢包概率设置较小,则平均队列接近上限阈值。针对这个问题,Feng 等人[8] 设计了一种改进 RED 算法:ARED。其基本思想是:通过检查平均队长的变化,判断 RED 是应更激进还是更保守。但是 ARED 额外地增加了两个参数,也就是最大丢包概率的增加系数和减小系数,使系统的参数整定更加困难。

主动队列管理存在参数设置敏感,响应相对滞后等缺陷,并没有在网络上广泛使用[9]。目前,现代控制理论和智能控制算法虽然在网络拥塞控制上已经有较多的应用,并取得了一定的成果[10~12],但还存在一些问题。如基于神经网络的 AQM 算法[12],其离线或在线的学习方法,面对数据流量经常突变的网络,能不能快速收敛和及时作出调整还需进一步研究。另外,过于复杂的 AQM 算法会消耗中间节点的运算资源,导致其转发数据包变慢,反而加重网络拥塞。

收稿日期：2011-10-15；**修回日期**：2011-11-21　　**基金项目**：江苏高校优势学科建设工程资助项目;江苏省青蓝工程资助项目;国家自然科学基金资助项目(60974129)；江苏省自然科学基金资助项目(BK2009388)

作者简介:姜文刚(1973-),男,江苏丹阳人,副教授,博士研究生,主要研究方向为网络控制、伺服控制(a_1_2_3@163.com);孙金生(1967-),男,教授,博导,主要研究方向为网络拥塞控制、容错控制;王执铨(1939-),男,教授,博导,主要研究方向为复杂大系统理论、网络控制、容错控制.

1　RED 算法

RED 的基本思想是通过中间节点缓存中平滑后的队列长度来感知网络拥塞状况,并以此计算随机丢包概率。

RED 算法主要依赖如下变量:平均队列长度 avg,平均队列长度下阈值 $q_{\min}$,平均队列长度上阈值 $q_{\max}$,当前实际队列长度 q,丢包率过渡值 p_b,丢包率实际值 p_a,上次丢包后收到的数据包个数 count。

RED 算法主要分为两部分:

a)计算平均队列长度(指数加权滑动平均—exponential weighted moving average)。平均队列长度 $\text{avg}=(1-w_q)\times\text{avg}+w_q\times q$。使用平均队列长度的目的是将中间节点上的暂态过滤掉,平滑队列长度。w_q 相当于低通滤波器长度,决定了 q 变化时,avg 改变的快慢。通常 w_q 的值较小时,avg 与实际队列的变化相比有一定滞后,但可防止 RED 对短暂的队列干扰反应过激。

b)计算丢包概率。当 $q_{\min}\leqslant\text{avg}\leqslant q_{\max}$ 时,首先计算一个过渡的概率值 p_b,随着 avg 从 $q_{\min}$ 增大到 $q_{\max}$,p_b 从 0 线性增大到 max p。如图 1 所示。

RED 并不直接把 p_b 作为数据包丢弃的概率,RED 希望数据包丢弃相对均匀分布,对一个突发流不进行惩罚,所以引进一个变量 count,其用来保存上次丢包后收到的数据包个数,$p_a=p_b/(1-\text{count}\times p_b)$,$p_a$ 为实际丢包率,随着 count 的变大,到达数据包被丢弃的概率也在增加,这样做的目的是实现丢包间隔均匀,避免连续丢包,导致扼杀突发数据流和数据流同步。

GentleRED 改变了 p_b 的计算。当平均队长 avg 大于 $q_{\max}$ 时,p_b 的值并不突变到 1,而是在 $q_{\max}$ 和 $2\times q_{\max}$ 之间由 max p 线性增长到 1,如图 2 所示。

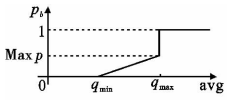

图1　RED概率计算方法

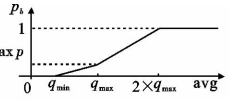

图2　GentleRED 概率计算方法

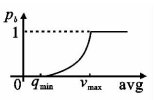

图3　RED-r概率计算方法

RED、GentleRED 算法的一个缺陷是平均队长对拥塞程度很敏感。如果拥塞不太严重或 max p 很大,则平均队列接近 $q_{\min}$;如果拥塞比较严重或 max p 较小,则平均队列接近 $q_{\max}$。

2　RED-r 算法

为了改善 RED 算法平均队长对拥塞程度很敏感的缺陷,减少 RED 参数的设置,避免采用线性化方程[13]经过复杂设计后仍难取得较理想的效果,本文提出一种改进的 RED 算法—RED-r。

RED-r 主要更改了 RED 的 p_b 计算方法,采用计算 p_b,使得其丢包概率变动更加平滑。二次圆函数的非线性特性,能够在平均队列长度增加过大时,计算得到比 RED 更大的丢包概率,平滑地遏制队列变长。

在平均队列长度较小时,二次圆函数计算得到的丢包概率比 RED 小,可以较好地稳定队列长度。RED-r 计算方法如图 3 和式(1)所示。

$$p_b=1.0-\sqrt{1-\left(\frac{\text{avg}-q_{\min}}{V_{\max}-q_{\min}}\right)^2},\ \text{avg}\leqslant\min(V_{\max},Q_{\max}) \qquad (1)$$

其中:$V_{\max}$是一个虚拟的上限值,该值的设置可以超过实际队列缓存的容量,$Q_{\max}$为队列物理缓冲的最大长度。RED-r 不需像 RED 那样要分别设置 max p 和 $q_{\max}$,只需设置 $V_{\max}$,因此比 RED 设置的参数要少。

采用二次圆函数,就是要避免 RED 在拥塞不太严重或 max p 很大,则平均队列接近 $q_{\min}$;如果拥塞比较严重或 max p 较小,则平均队列接近 $q_{\max}$。并使得丢包概率变化比较平滑,减少队列的抖动。RED-r 算法在 RED 的基础上增加了平方和开方运算,并没有增加过多的运算开销,比文献[12]等复杂的 AQM 算法计算量要小。

3　RED-r 算法的性能分析

在 NS2 下构建如图 4 仿真网络拓扑,$S_1\sim S_n$ 均为 TCP 发送端,分别向 $D_1\sim D_n$ 一一对应发送数据,即 S_i 发送数据给 D_i 接收。$S_1\sim S_n$ 均为持久性 FTP 业务源,到 R_0 的链路速率为 10 Mbps,延时 5 ms。R_0-R_1 为瓶颈链路,链路速率为 15 Mbps(3750 packets/s),延时 5 ms。R_1 到 D_1-D_n 链路速率为 45 Mbps,延时可以改变。各节点缓存队列长度均为 800 packets,数据包大小为 500 packets,接收端的接收窗口设置足够大,使得 TCP 的发送使得仅受拥塞窗口控制。

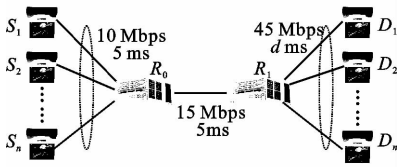

图4　仿真网络拓扑结构

3.1　网络延时较小时的性能

在图 4 中,$d=5$ ms,一开始启动 60 个 TCP 发送端,30 s 后再启动 60 个 TCP 发送端,再过 30 s 停止 90 个发送端,瓶颈链路队列管理分别采用 RED、RED-r 和 PID。RED 的参数设置:$q_{\max}=250$ packets,$q_{\min}=50$ packets,其余参数为 NS2 默认。RED-r 的参数设置:$q_{\min}=50$ packets,$V_{\max}=250$ packets。PID 控制器控制的期望队列设置为 150 packets,采样频率为 160 Hz,PID 参数分别采用文献[10,11]中的参数。仿真结果如图 5 ~ 8 所示。

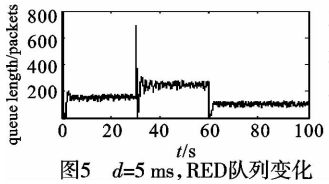

图5　d=5 ms,RED队列变化

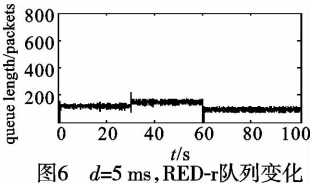

图6　d=5 ms,RED-r队列变化

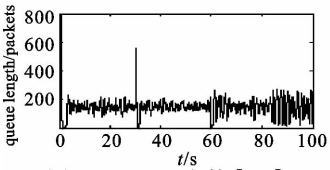

图7　d=5 ms,文献[10] PID队列变化

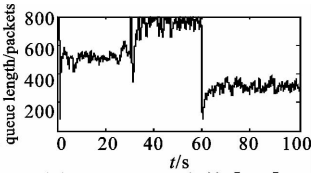

图8　d=5 ms,文献[11] PID队列变化

从图 5 可以看到,RED 控制的队列随着 TCP 发送端的数目变化,其队列平均长度变化较大,TCP 发送端从 60 个变化到 120 个时,过渡比较激烈,平均队列长度从 180 packets 变化到接近其 $q_{\max}=250$ packets;当 120 个发送端变为 30 个发送端时,其平均队列长度约为 100 packets,但除了过渡过程时间较长外,其队列能保持比较平稳。

图 6 采用 RED-r 队列管理,其过渡过程较短,TCP 发送端从 60 个增加到 120 个时,其平均队列长度从 170 packets 增加到 180 packets;TCP 发送端从 120 个减少到 30 个时,其平均队列长度为 165 packets,变化幅度明显小于 RED,说明 RED-r 能

改善 RED 控制的队列随着 TCP 发送端数目和丢包率变化而改变的缺陷。

图 7 中文献[10]的 PID 参数也不能使控制的队列达到期望值,TCP 发送端数目增加时,其平均队列长度能维持,同样由于线性化方程的缺陷,其队列波动较大,其波动范围,超过了 RED-r 的队列变化范围,当 TCP 发送端降为 30 时,队列不能稳定。

图 8 中文献[11]的 PID 参数并不能使队列达到期望值,TCP 发送端数目增加时,其平均队列长度也在增加,TCP 发送端为 120 时,其控制的队列处于满队列状态,PID 控制器不能稳定工作;TCP 发送端降为 30 个时,其控制的队列长度也变小。

文献[10,11]指出网络链路存在大时滞时,多种主动队列管理机制不能稳定,因此文献[10,11]PID 控制器设计时,加有相应的补偿算法,并经过优化。对于网络这样一个复杂系统,控制器要能保持大范围参数变动时的稳定性有很大难度。

3.2 网络延时较大时的性能

通常中间节点路由器在实施主动队列管理策略时,其链路的延时是可以通过测量大致得到,并以此设定主动队列管理的参数。

为了在大时滞时 RED-r 能维持队列的稳定,因为 RED 在大延时时不能稳定工作[10],在图 4 中采用 RED-r 算法和 PID 算法,RED-r 的 V_{max} 设置为 4 500 packets,链路时延 $d = 190$ ms,这样 TCP 发送端的 RTT 至少为 0.4 s。采用 60 个 TCP 发送端,仿真结果见图 9 ~ 11。

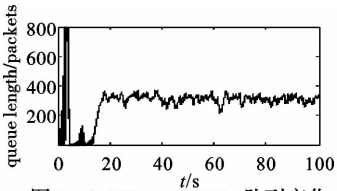

图9 d=190 ms, RED-r队列变化

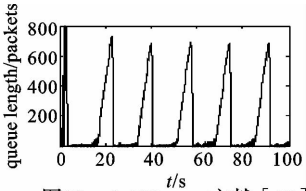

图10 d=190 ms, 文献[10] PID队列变化

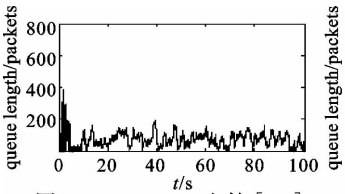

图11 d=190 ms, 文献[11] PID队列变化

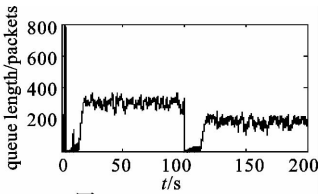

图12 d=250 ms, RED-r 队列变化

图 9 中,RED-r 队列能保持稳定;图 10 中,文献[10]的 PID 控制器不能稳定工作;图 11 中,文献[11]PID 控制的队列长度保持在较小的值,但经常为 0,传输效率下降,该 PID 参数在链路延时小时,其控制的队列保持在较大的长度。所以 PID 控制器很难在 RTT 变化较大时,将队列控制在期望值附近。

将链路时延改为 $d = 250$ ms,RTT 至少为 0.52 s,采用 60 个 TCP 发送端,100 s 后减为 30 个 TCP 发送端,仿真结果如图 12 所示。

图 12 可以看到 RED-r 在大时滞,和较少 TCP 发送端时,都能保持稳定。V_{max} 仍然设置为 4 500 packets,减少链路延时,设 $d = 5$ ms,仿真结果如图 13 所示,RED-r 还是保持稳定,但其维持的队列长度增加,在 500 packets 附近。如果按照经验法则[14]设置中间节点路由器的队列缓存,缓存大小为 $B = \text{RTT} \times C = 0.52 \times 3750 = 1950$ packets,RED-r 维持的队列长度为队列缓存的 1/4 左右。

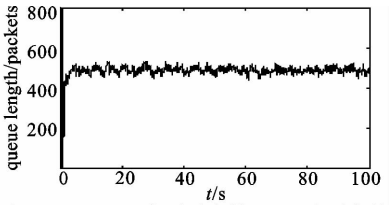

图13 d=5 ms, 大延时参数RED-r队列变化

仿真表明 RED-r 能有效改善 RED 的缺陷,简化其参数设置,不能在网络延时较大时保持队列长度的稳定。

4 结束语

为了改善 RED 算法平均队长对拥塞程度很敏感的缺陷,减少 RED 参数的设置,避免采用线性化方程经过复杂设计后仍难取得较理想的效果,本文提出一种改进的 RED 算法 RED-r,该算法采用二次圆函数来计算丢包概率,减少了 RED 的设置参数,能实现在网络大延时和小延时时保持队列稳定,在小延时时能获得比 PID 队列更平滑的效果。最后通过 NS2 仿真来验证。

参考文献:

[1] JACEBSON V. Congestion avoidance and control[J]. ACM Computer Communications Review,1988,18(4):314-329.

[2] KEVIN F, FLOYD S. Simulation-based comparisons of Tahoe, Reno, and SACK TCP[J]. ACM Computer Communication Review, 1996,26(3):5-21.

[3] BRADEN B, CLARK D, CROWCROFT J, *et al*. IETF RFC 2309, Recommendations on queue management and congestion avoidance in the Internet[S]. 1998.

[4] ZHENG Chang-yong, DAI Yue-hua, CHEN Jun-ning. Is current active queue management really necessary[C]//Proc of the 1st International Workshop on Education Technology and Computer Science. Washington DC: IEEE Compnter Society, 2009:538-541.

[5] STANOJEVI' C 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.

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

[7] FLOYD S. Recommendation on using the "Gentle_" variant of RED algorithm[EB/OL]. (2008-11-15). http://www. icir. org/floyd/red/gentle. html.

[8] FENG Wu-chang, KANDLUR D D, SAHA D, *et al*. A self-eonfiguring RED gateway[C]//Proc of IEEE INFOCOM. New York: IEEE Communications Society, 1999:1320-1328.

[9] DANA A, MALEKLOO A. Performance comparison between active and passive queue management[J]. International Journal of Computer Science Issues, 2010, 7(5):13-17.

[10] 任丰原,林闯,任勇,等. 大时滞网络中的拥塞控制算法[J]. 软件学报,2003,14(3):503-511.

[11] 陆锦军,王执铨. 基于一类新 PID 的网络拥塞控制算法[J]. 哈尔滨工业大学学报, 2008, 40(9):1457-1461.

[12] 张少博,李钢,康军. 基于神经网络监督控制的拥塞控制算法研究[J]. 计算机应用研究,2010,27(2):657-660.

[13] HOLLOT C V, MISRA V, OWSLEY T D, *et al*. A control theoretic analysis of RED[C]//Proc of IEEE INFOCOM. Piscataway, NJ: IEEE, 2001:1510-1519.

[14] VILLAMIZAR C, SONG C. High performance TCP in ANSNET[J]. ACM Computer Communication Review, 1994, 24(5):45-60.