基于 NS2 的 RED 算法研究与仿真分析①

RED Algorithm and Simulation Analysis Based on NS2

汪华斌　(惠州学院 计算机科学系 广东 惠州 516007)

摘　要： 主动队列管理(Active Queue Management)算法是近几年网络研究的重点。详细讨论了 RED 主动队列管理算法的关键问题,研究了近年来对 RED 算法的几种改进算法,采用 NS2 对其改进算法仿真分析,通过大量仿真实验,结论表明其自适应 ARED 和改进的 ARED(NewARED)算法减少了排队时延,提高系统稳定性和可靠性。

关键词： 网络仿真 随机早期丢弃 主动队列管理 拥塞控制

1 引言

主动队列管理(Active Queue Management, AQM)[1]作为 Internet 端到端拥塞控制的一项增强手段,有效地克服了传统 DropTail 缓冲区队列管理策略下的满队列、对业务流的死锁以及全局同步问题。在现有的 AQM 方案中,链路的拥塞都是通过队列长度、注入网络分组的速率、缓冲区队列溢出或空白等网络特性或者是这些特性的结合来衡量的。

1999 年 S. Floyd 和 V. Jacobson 等提出来的 RED[2]算法是最典型的 AQM 拥塞控制机制算法,能够提供比 DropTail 弃尾算法更佳的网络性能,目前已成为 IETF 推荐的唯一的 AQM 候选策略。RED 算法能避免全局同步现象有效地提高链路带宽利用率和减小队列平均长度。但其仍具有参数设置复杂和不能有效估计拥塞的严重性等问题。随着 RED 应用的增多,研究者提出了许多形式的 RED 算法的改进策略,目前,Internet 上广泛使用的主动队列管理方案主要有 RED[2]、gentle – RED[3]、ARED[4]和 NewARED[5]等。本文在分析 RED 算法原理的基础上,结合其它几种改进算法,通过大量的仿真实验,总结几种算法的优缺点。

2 OriginalRED 算法原理

S. Floyd 等提出的随机早期检测 RED 算法,其基本思想是:其基本思想是通过检测路由器输出端口的队列平均长度来判断是否会引起网络拥塞,并在队列平均长度达到一定阀值时,随机的选择部分新到达的分组进行丢弃或者标记,并通知源端减小拥塞窗口,降低分组发送速率,从而缓解网络拥塞。RED 算法中缓冲区队列是一个 FIFO 的分组队列。缓冲区队列的利用率通过队列的平均长度来反映,队列长度一般以分组的个数为单位。RED 算法有以下几个重要参数,平均队列长度 Qavg,队列长度两个门阀值 Qmin、Qmax,系统设定的分组丢弃的最大概率为 Pmax 和计算平均队列长度权值 wq。

当新分组到达路由器时,RED 算法先按低通滤波器计算队列平均长度,如公式(1):

$$Qavg(new) = (1 - wq) * Qavg(old) + wq * Qcur \qquad (1)$$

其中 Qcur 为瞬间队列长度, wq 为计算平均队列的权值,其大小决定 Qcur 对 Qavg 结果的影响程度。

如果平均队列长度 Qavg 小于最小门阈值 Qmin,则该分组进入队列;如果平均队列长度位于最小阈值 Qmin 和最大阈值 Qmax 之间,则通过下列公式计算丢弃概率 Pa,并以 Pa 概率丢弃该分组,如果平均队列大于最大门阀值 Qmax,则丢弃该分组。计算分组丢弃概率 Pa 的公式(2)(3)如下:

$$pb = \begin{cases} 0 & Qavg \le Q\min \\ P\max * \dfrac{Qavg - Q\min}{Q\max - Q\min} & Q\min < Qavg < Q\max \\ 1 & Qavg \ge Q\max \end{cases} \qquad (2)$$

$$Pa = \frac{pb}{1 - count * pb} \qquad (3)$$

① 基金项目:惠州学院 2008 年自然科学研究项目 ($C_2$08・0204)

算法描述如下:

For each packet arrival

Calculate the average queue size Qavg

If Qmin ≤ Qavg < Qmax

Calculate probability pa

With probability pa:

Mark the arriving packet

Else if Qmax ≤ Qavg

Mark the arriving packet

从公式(2)**可以看出,如果** Pmax **设置得太大或者网络流量负载较轻,则平均队列长度会靠近** Qmin;**如果** Pmax **设置得太小或者网络流量负载较重,则平均队列长度会在** Qmax **附近。**RED **算法依赖队列平均长度来判断是否会发生网络拥塞,然而队列平均长度受网络拥塞和控制参数的影响非常巨大,再者概率** P **不仅和** Qavg **有关,而且还和上次丢弃分组开始到现在连续进入队列的分组数量** Count **有关,如公式**(3)**所示随着** Count **的增加,下一个分组被丢弃的可能性也在缓慢增加,这主要时为了均匀间隔的丢弃分组,以消除对突发流的偏见和全局同步现象。**

3 gentelRED 算法

改进的 RED **算法**(Gentle – RED)**继承了** RED **算法的主要思想,当平均队列小于最大门阀值时,其丢弃概率的计算与** RED **算法完全一致,而当平均队列** Qavg **大于** 2Qmax **时,则不象** RED **算法一样丢弃概率由** Pmax **到**1 **的跳跃,而是以另一种线性关系计算** Pb **的概率,然后以** Pa **概率来丢弃新到达路由器的分组。其丢弃概率** Pa **函数表示为公式**(4)(5):

$$pb = \begin{cases} 0 & Q\text{avg} \le Q\min \\ \text{Pmax} * \dfrac{Q\text{avg} - Q\min}{Q\max - Q\min} & Q\min < Q\text{avg} < Q\max \\ (1 - P\max) * \dfrac{Qavg - Q\max}{Q\max} + P\max & \text{Qmax} \le \text{Qavg} < 2\text{Qmax} \\ 1 & 2Q\max \le Qavg \le Buffersize \end{cases} \quad (4)$$

$$Pa = \frac{pb}{1 - count * pb} \quad (5)$$

4 Self – configuring RED 算法

在拥塞严重的网络中,路由器必须将足够多的拥塞信息通知到源端,以充分降低负载,避免由于队列溢出而丢包。另一方面,路由器也要防止拥塞信息过多地传给源端,从而造成瓶颈链路利用率的下降。因此,进行拥塞通知时应充分考虑到瓶颈链路上流的数量。而 RED **并没有考虑到这一点。为此** ARED(A Self – configuring RED)**提出了一种自动配置机制,根据流量的变化来配置适当的参数。**

RED **中,拥塞指示的发送速度是由参数** Pmax **来体现的。如果** Pmax **太大,那么丢包主要就是由于早期拥塞检测中产生的丢包造成的;如果** Pmax **太小,丢包主要就是由于队列溢出造成的。**RED **的一个弱点是平均队长对拥塞程度和参数设置很敏感。如果拥塞不太严重或者** Pmax **很大,则平均队长接近** Qmin;**如果拥塞很严重或者** Pmax **很小,则平均队长接近或大于** Qmax**。结果造成平均排队时延对流量负荷和参数设置很敏感。**

ARED **的基本思想就是通过检查平均队长的变化来感知** RED **是应更激进**(aggressive)**还是更保守**(conservative)**。如果平均队长是在** Qmin **附近振荡,那么拥塞控制就太激进了;如果在** Qmax **附近振荡,那么拥塞控制就太保守了。根据所观察到的平均队长,**ARED **动态地** Pmax **调整的值。其算法如下**:

Every QavgUpdate:

if(Qmin < Qavg && Qavg < Qmax)

status = between;

else if(Qavg < Qmin && status != below)

status = below

Pmax = Pmax/α

else if(Qavg > Qmax && status != above)

status = above

Pmax = Pmax * β

各参数变量含义:

status:**平均队长状态**

between: Qmin **和** Qmax **之间**

below:**小于** Qmin

above:**大于** Qmax

α:Pmax **减少量**

β：Pmax 增加量

ARED 算法的思想很简单，就是根据平均队长是否在 Qmin 和 Qmax 之间，对 Pmax 采用乘法增加和乘法减少（Multiplicative Inerease Multiplicative Decrease，MIMD）从而尽量保持平均队长在 Qmin 和 Qmax 之间。

ARED 是对 RED 改动很小的一种算法，它保留了 RED 的基本结构，只需调节参数 Pmax 从而保持平均队长在 Qmin 和 Qmax 之间，消除了 RED 的队列延时问题和参数敏感性问题。

5 NewARED 算法

为了提高 ARED 的鲁棒性，Sally Floyd 等人提出了一种新的改进 ARED 算法 NewARED（又称为 Adaptive RED）。其基本思想和 ARED 一样，都是采用自适应的 Pmax 以保持平均队长在 Qmin，Qmax 之间。不同之处在于 NewARED 保持平均队长在 Qmin 和 Qmax 的一半左右；Pmax 不是每来一个包都改变，而是有一定时间间隔；Pmax 不采用乘法增加和乘法减少，而是加法增加和乘法减少（Additive Inerease Multiplieative Deerease，AIMD）；Pmax 限制在[0.01，0.5]。具体算法如下：

Every interval seeonds：

if（Qavg > target && Pmax≤0.5）

Increase Pmax：

Pmax = Pmax + α

elseif（Qavg < target&&Pmax≥0.01）

deerease Pmax：

Pmax = Pmax * β

各参变量含义：

interval：时间间隔（通常取 0.05）

target：Qavg 的目标范围【Qmin + 0.4 *（Qmax – Qmin），Qmin + 0.6 *（Qmax – Qmin）】

NewARED 的鲁棒性主要来自于 Pmax 不是很频繁的变化。但如果拥塞程度急剧变化，则 Pmax 需要过一段时间才能适应。为了保证 ARED 在这段时间里性能不会过度下降，因此将 Pmax 限制在[0.01，0.5]之间。这样，即使这段时间内平均队长不在目标范围内，平均延时和吞吐量也不会下降太多。

6 仿真及网络性能比较分析

6.1 仿真模型

为了验证几种 RED 算法，作者进行了大量的仿真实验，并就此进行了性能比较和评价。本文采用的仿真环境为 Winxp + Cygwin + Ns2.31[6]。仿真采用的拓扑结构如图1的哑铃模型。在 R1～R2 路由器之间的瓶颈链路，为各业务流共同竞争的资源，带宽和延迟分别为 10Mb 和 10ms，路由器 R1 缓冲区为 50 个分组，其他链路带宽和延迟均为 20Mb 和 1ms。TCP1～TCPn 作为 TCP 源端，TCPsink1～TCPsinkn 作为 TCP 接收端，UDP1～UDPn 作为 UDP 源端，NULL1～NULLn 作为 UDP 接收端，仿真时间为 40s。

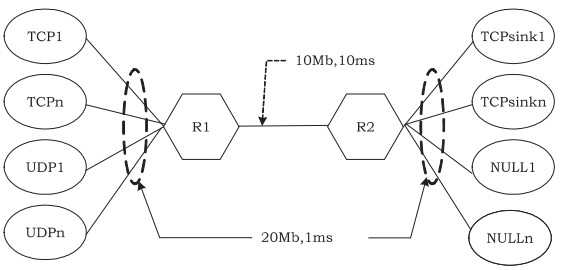

图1　仿真实验拓扑图

6.2 仿真结果及分析

由于当前 Internet 中约 90% ～95% 以上的数据为 TCP 流[7]，考虑到 UDP 和 TCP 流的不同机制，采用一个 UDP 流作为背景，多个 TCP 流参与竞争的模式。为了使实验更好进行，且数据更真实有效，仿真使用突发流进行实验，在实验中几种算法均采用最小门阀值 Qmin =0.2 * ql 和最大门阀值 Qmax =0.6 * ql，ql 为路由器缓冲队列，权重 Wq = 0.002，最大丢弃概率 Pmax = 0.1[8]。

在实验中根据实验拓扑图（如图1所示）设计由多个 TCP 连接突发流和一个 UDP 连接作为背景流的情况，仿真时间为 40S。利用该模型对四种算法分别进行仿真，仿真结果如表1（以 40 个 FTP 连接为例）：

表 1　四种算法的数据对比

	original RED		gentelRED	
	到达包	**丢弃包**	**到达包**	**丢弃包**
FTP **统计**	51134	7148	51847	7759
CBR **统计**	33334	4486	33334	5163
合计	84468	11634	85181	12922
排队时延	0.0243657		0.0186381	
	ARED		NewARED	
	到达包	**丢弃包**	**到达包**	**丢弃包**
FTP **统计**	52304	8248	51868	7862
CBR **统计**	33334	5522	33334	4900
合计	85638	13770	85202	12762
排队时延	0.0112923	0.0150973		

表 1 的数据表明，原始的 originalRED **吞吐量较大，但在路由器队列中排队时延较大。改进的** RED **算法** gentelRED **和** NewARED **次之，而** ARED **算法的吞吐量虽然较小，但其排队时延比其它三种算法小得多，这对于对时延要求较高的业务流而言是值得的。**

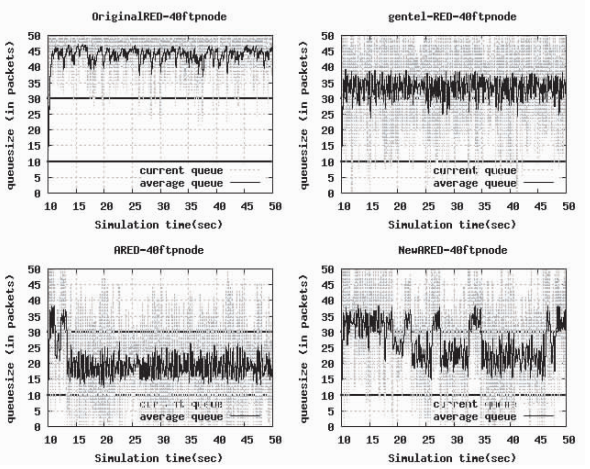

图 2　40 个 TCP 连接下队列长度变化

图 2 队列长度变化表明四种算法在轻度拥塞和中度拥塞情况下，gentelRED **和** ARED **算法队列长度变化较稳定，振荡较弱，稳定性较好。**

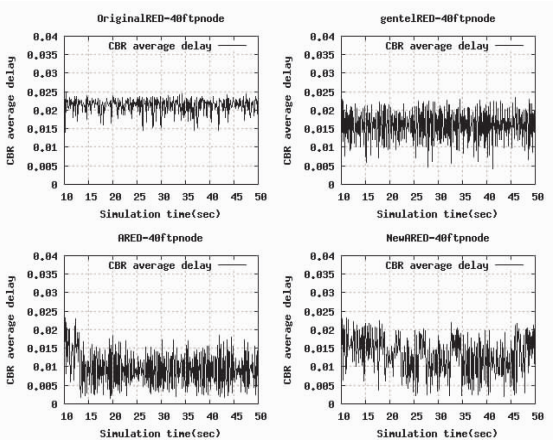

图 3　40 个 TCP 连接下 CBR 流时延变化

图 3 中 CBR **流时延变化图表明** ARED **和** NewARED **算法在传播时延上较小，而** OriginalRED **和** gentelRED **较大，**ARED **和** NewARED **算法更符合对时延要求较高的业务流。**

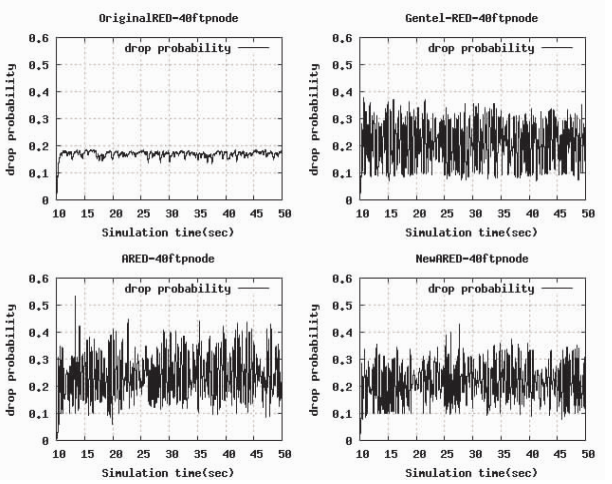

图 4　40 个 TCP 连接下丢包率变化

图 4 中 originalRED **算法丢包率一直维持在** 17% **左右，当队列长度超过** Qmax **时采用强制性丢包进行拥塞控制，而** gentelRED **则采用新的丢弃概率计算修正，**ARED **和** NewARED **算法采用** α、β **参数自适应对丢弃概率进行修正路由器队列的最大丢弃概率** Pmax。

7　结论

RED **路由器中，当队列长度超过** Qmax **后将引起**

强制性的丢包,频繁的强制性的丢包将引起网络的有效利用率下降,同时增加数据包的排队延时。自适应地修正 Pmax 值,可在保持高的吞吐率的基础上,使队列更趋于稳定,减少路由器的丢包,缩短数据包的排队时间,从而减小网络延时。实验结果表明,通过监视队列的变化,自适应地调整 Pmax 的值,改进的 RED 算法显著地降低了丢包率,提高了网络的链路利用率,在解决网络业务流突发的问题上,ARED 算法和 NewARED 算法更稳定和可靠。

参考文献

1 Braden B, Clark D, et al. Recommendation on queue management and congestion avoidance in the internet. Request for Comments (RFC) 2309. http://www.ietf.org/rfc,2003 - 02 - 15.

2 Floyd S, Jacobson V. Random early detection gateways for congestion avoidance. IEEE/ACM Transaction on Networking. August 1993,1(4):397 - 413.

3 Floyd S. Recommendation on using the "gentle_" variant of RED. http://www.icir.org/floyd/red/gentle.html.

4 Feng W, Kandlur D, et al. A Self - Configuring RED gateway. Proceedings Infocom 1999, New - York, March 1999,1320 - 1328.

5 Floyd S, Gummadi R, Shenker S. Adaptive RED: An Algorithm for Increasing the Robustness of RED. Technical Report,2001.

6 UCN/LBL/VINT. Network Simulator - NS2. http://www - mash.cs.berkeley.edu/ns

7 Thompson K, Miller GJ, Wilder R. Wide Area Internet Traffic Patterns and Characteristics. IEEE Network, 1997,11(6):10 - 23

8 Floyd S. RED: Discussions of Setting Parameters. http://www.icir.org/floyd/REDparameters.txt, November 1997.