一种选择性丢包机制的分析与设计*)

汤德佑[1,2] 骆嘉伟[1] 张大方[1] 黄元江[2] 张白妮[1]

(湖南大学计算机与通信学院 长沙410082)[1] (株洲工学院计算机系 株洲412008)[2]

摘 要 随机早期检测算法(RED)在产生丢包时简单地丢弃到达的数据包,由于Web流占据了网络上75%左右的带宽,而Web流的特点是数据包比较少,这就使得早期丢包的效果难以体现。本文提出一种选择性丢包机制,引进一个类似公平队列的虚队列,检测发送速率过大的流,筛选出丢包的候选链接,拥塞时丢弃候选链接在队列中的第一个连接的数据包。模拟实验结果表明,与队头丢包相比,采用选择性丢包的RED算法可进一步降低网关的丢包率,提高吞吐率,缩短Http的响应时间,提高队列的稳定性。

关键词 随机早期检测,拥塞控制,队列管理,选择性丢包

Analysis and Design of Selective Drop Policy

TANG De-You[1,2] LUO Jia-Wei[1] ZHANG Da-Fang[1] HUANG Yuan-Jiang[2] ZHANG Bai-Ni[1]

(Hunan University, Changsha 410082)[1] (Zhuzhou Institute of Technology, Zhuzhou 412008)[2]

Abstract Random Early Detection(RED) drops the arriving packet when it is necessary to drop a packet, but the Web traffic is the main traffic in the network and is has few packets, thus the effectiveness of the advance drop is not obviously. This paper proposes a selective dropping policy which uses a virtual queue resemble to the Fairness Queue to detect the flows sending more packet more than the fair share and drops the packets from those flows when congested. The simulation results have demonstrated that the RED algorithm with selective drop has decreased the dropping rate, increased the throughput of the gateway, cut down the response time of Http application and improved the stability of the RED queue.

Keywords Random early detection, Congestion control, Queue management, Selective drop

1 引言

目前,在IP层采用的拥塞控制策略主要是队尾丢弃(Drop-Tail),队尾丢弃网关实现简单,但它存在下面的缺陷[1]:(1)容易引起队列锁定,队列主要由某些连接占用,其他的连接无法公平地分享带宽;(2)产生满队列,造成连接的延时加大,甚至引起不必要的超时重传,加重拥塞;(3)没有考虑突发流量,进而引起同步问题,造成链路的利用率下降。解决队列锁定问题可以采用队头丢包或随机丢包,允许每个数据包进队,但是这些方法都无法解决满队列问题。因此,IRTF(Internet Research Task Force)在RFC2309中建议在路由器上实现主动队列管理(Active Queue Management, AQM),探测即将到达的拥塞,在路由器的缓冲区满以前主动丢包,以避免缓冲区的溢出和锁定,并推荐使用随机早期检测算法[2](Random Early Detection, RED)。RED通过计算每个数据包到达网关时的平均队列长度来判断是否会引起网络拥塞,并在平均队列长度到达一定的限度后以一定的概率随机地丢弃或者标记到达的数据包。RED算法能适应突发的流量和短暂的拥塞,在一定程度避免全局同步,减少数据包丢失率,提高公平性。但是RED还是存在两个主要缺陷:(1)对参数的依赖性很强,当参数配置不合理时,其性能甚至不如DropTail[3];(2)RED以较高的丢包率换取较小的平均排队延时[4]。为了解决RED的参数依赖性问题,产生了一些RED的派生算法,如gentle RED, WRED, Adaptive RED, Self-configuration RED,等等。这些算法主要解决的是RED的参数依赖性问题,并没有考虑丢包连接的选择问题。表1是对RED算法分别采用队头、队尾和随机丢包的丢包率对比。其中,队尾丢弃丢包率最高,队头丢包最低。在TCP拥塞控制中,丢包事件是端了解网络拥塞状况的唯一手段,要缩短TCP的拥塞响应时间就必须尽快将拥塞反馈到TCP端,队头丢弃比队尾丢弃提前将拥塞通知TCP端,因而效果最明显。另外,考虑一个拥塞窗口为32的TCP连接A和一个拥塞窗口为4的TCP连接B,连接在检测到拥塞以后进入拥塞避免,拥塞窗口减半。若选择连接A丢包,由于其拥塞窗口较大,则不但可以明显减小网关的负载压力,也有利于连接拥塞窗口的恢复。相反,若连接B丢包,网关只有继续丢包才能进一步降低负载压力,同时造成连接B很难公平地分享带宽。

本文设计了一个类似公平队列的虚队列,利用一种假轮转的方法检测发送速率过大的连接。同时在路由器中设一个丢包候选连接表,连接表由满足一定条件的发送速率较大的连接组成,拥塞时优先丢弃候选连接表中连接在队列头部的数据包,以尽快将拥塞信息反馈到能够有效减轻拥塞的发送端,从而缩短拥塞响应时间,降低网关的丢包率。

表1 RED中几种丢弃策略的对比

	队头丢弃	随机丢弃	队尾丢弃
丢包率	0.0977	0.0987	0.1061

2 RED算法

RED利用一个加权的平均队列长度(avg)来检测链路是否发生拥塞及拥塞的程度,在路由器队列满以前以一定的概

*)基金项目:国家自然科学基金项目(60273070)。汤德佑 硕士研究生,讲师,研究方向为拥塞控制、流控制。骆嘉伟 副教授,硕士生导师,研究方向为拥塞控制、网络数据挖掘。张大方 教授,博士生导师,主要研究方向为可信系统与网络、网络测试等。

率随机地提早丢弃或标记一些数据包以通告 TCP 发送者降低发送速率。RED 中有几个重要的参数：max_{th}，min_{th}，max_p，w_q。两个门槛值 max_{th} 和 min_{th} 用于检测拥塞和计算丢包的概率，max_p 决定了平均队列在 max_{th} 和 min_{th} 之间时线性丢包的上限，w_q 是一个权值，反映 avg 对即时队列的敏感程度。RED 算法的描述如下：

计算平均队列长度 avg

$avg=(1-w_q)*avg+w_q*q$

若 $avg \leqslant min_{th}$

数据包进队列

若 $min_{th}<avg<max_{th}$

计算随机丢包概率

$p_b=max_p*(avg-min_{th})/(max_{th}-min_{th})$

$p_b=p_b*PacketSize/MaximumPacketSize$

$p_a=p_b/(1-count*p_b)$

以概率 p_a 丢弃到达的数据包

若 $max_{th} \leqslant avg$

丢弃到达的数据包。

对 RED 算法的仿真实验表明，RED 算法利用平均队列长度检测拥塞，消除了 DropTail 算法对短暂的突发流量的偏见。也有实验表明，当加以较大的流量的时候，RED 算法尽管能够控制队列不会产生溢出，但是丢包率已经远远超过了 DropTail 算法[3]。在上述算法中，只要计算出的平均队列长度大于 max_{th}，RED 将丢弃达到的数据包，而不考虑当前的网络负载大小和负载变化情况，此时的 RED 算法实际上已经退化为缓冲区长度为 max_{th} 的 DropTail 算法，因而引起了更高的丢包率，并出现同步。

3 选择性丢包的设计与实现

对一个 TCP 友好的连接，存在下面的关系式[5]：

$$B_i=\frac{C\times MSS_i}{RTT_i\times\sqrt{P_i}} \tag{1}$$

其中 B_i 表示连接 i 所占用的带宽，MSS_i 是连接的最大段长度，RTT_i 是连接的往返时间，P_i 表示应该具有的丢包率，若 TCP 连接采用的是非延迟 ACK，则取 C=1.22，否则取 C=0.87。变换后可得：

$$P_i=\left(\frac{C\times MSS_i}{B_i\times RTT_i}\right)^2 \tag{2}$$

即连接的丢包率与所占用的带宽的平方成反比，与连接的 RTT_i 的平方成反比。在 RED 算法中，丢包的概率是利用平均队列直接计算的，并且当平均队列长度达到 max_{th} 后该数据包就将被丢弃，而不管该连接拥塞窗口的大小[6]，是否在最近丢包。因而容易造成在处于启动阶段的连接很难占有带宽，或者是造成某一个连接频繁丢包，直至该连接超时，拥塞窗口变为1。若大量的连接进入超时重传，那么网关在经历一次强制性丢包后，将出现队列排空，引起队列抖动。在当前的网络上，75%的流量是 Web 流，Web 流的一个主要特点是数据包较少。RED 算法中采用的丢包策略是随机的，与连接占用的带宽有关，因而 Web 流被选中丢包的概率最大。下图是采用随机丢包的 RED 丢包概率与连接的数据包分布图，结果表明连接的丢包概率并未随着数据包的增加而增加。相反，短连接具有了更高的丢包概率。

据统计，Web 流占据了75%的带宽，10%的流占据了96%的带宽，而在某一个时间段内(如5秒)，1%的流占据了50%的带宽，说明了只要限制部分的连接就可以实现其他的连接公平地分享带宽[7]。在选择性丢包中，连接总是处于以下两个阶段之一：启动阶段和稳定传输阶段。启动阶段拥塞窗口较小，稳定传输阶段连接的拥塞窗口较大或拥有比较大的发送速率。启动阶段和稳定传输阶段的区分是通过在一个虚队列上采用类似于公平队列的方法实现的，在实验中设计了一个带有2个双向循环链表 A、B 的哈希表，哈希表记录了每个连接的源地址、端口，目的地址、端口和连接的最近发送字节数，连接的数据包的平均排队时间，连接的近似 RTT 值，最近的丢包时间和最近的数据包进队时间。双向循环链表 A 按接收到的字节数逆序排列，每个连接在队列中的记录时间超过1个连接对应的 RTT 后进入轮转，当连接的字节数轮转到零，并在最近的一个 RTT 内没有接收到数据包以后，表明该连接不是一个活跃的连接，则删除该连接的记录信息。轮转的顺序由双向循环链表 B 确定，每一个新生成的连接插入到刚刚完成轮转的连接之前。在每一个连接完成轮转后，相应地重排该连接在表 A 中次序，保证表 A 有序。在表 A 中，若连接占有的带宽超过平均带宽，则该连接处于稳定传输阶段，否则为启动阶段。

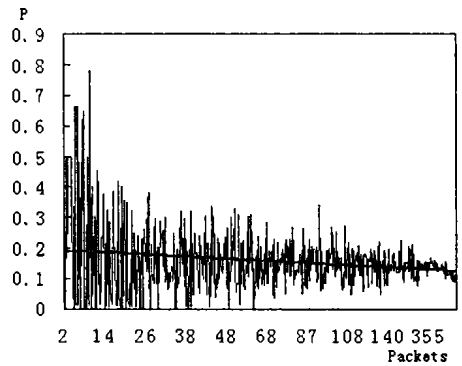


图1 丢包概率与连接数据包分布图

数据包进队时，首先查看是否是一个新的连接，若是一个新的连接，则记录下该连接的初始化时间，并以连接最初的两个数据包的到达时间差作为该连接的粗粒度的 RTT 时间。同时对每个数据包记录其排队时间，并对这些排队时间作加权移动平均，权值取为0.002，用这个加权值优化该连接的 RTT 值。引入 RTT 后，当到达的是非响应性连接时，计算出的 RTT 就是连接的发送数据包的间隔，若发送速率很大则丢包的比例也将加大。若连接已经存在，则修改接收到的字节数，并对链表 A 排序，同时修改该连接的 RTT 值。

同时，设一个候选连接表，表成员由发送速率大的连接组成。当 RED 产生丢包时，首先查看队列中是否有数据包对应连接属于候选连接，若有则丢弃该数据包，并从候选连接表中删除该连接。否则扫描链表 A 中的连接，若连接满足下列条件则认为该连接是一个候选连接：(Ⅰ)连接已经进入轮转；(Ⅱ)连接处于稳定传输阶段；(Ⅲ)连接在最近一个 RTT 内没有出现丢包；(Ⅳ)连接的丢包率低于利用公式(2)计算出的丢包率。若该连接在缓冲区中有数据包，则丢弃该连接在缓冲区中的第一个数据包，否则将其加入到候选连接表中。若上述条件均不满足，则将队头的数据包丢弃。下面是选择性丢包的伪码描述：

```
数据包进队：
初始化或者修改连接的状态信息；
更新链表 A、B；
每隔一个时间间隔 τ 触发：
按表 B 次序选择一个连接轮转；
若连非活跃则删除该连接的状态信息，并更新链表 A、B；
否则，更新链表 A；
丢包事件：
数据包属于候选连接则直接丢弃该数据包，并删除该从候选连
    接表中删除该连接；
否则，扫描链表 A，查找候选连接，若连接在缓冲区中有数据包，
    则丢弃该连接在缓冲区中的第一个数据包；
否则，将该连接加入到候选连接表；
若无满足条件连接，则丢弃队头数据包。
```

4 实验结果

在ns2.19ba中我们实现了这种选择性丢包机制，并与RED算法中的队头丢包作了比较，实验中采用的网络拓扑结构如图2所示。瓶颈链路上的延迟是20ms，带宽是1.5M，其他的链路的带宽是10M，延迟是3ms，R1的缓冲区大小为100Packet，运行RED网关，其余的均运行DropTail。RED网关中的 $Min_{th}=10$，$Max_{th}=30$，$w_q=0.002$，$Max_p=0.1$。为了验证选择性丢包，实验中采用了四种不同的实验流量。四组实验均运行65秒，实验中的http会话均含有10个页面，cbr的发送速率为0.4Mbps。

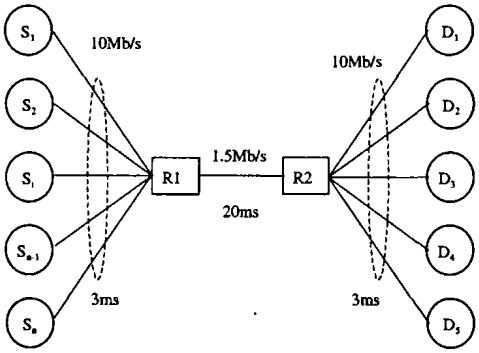


图2 模拟网络拓扑结构

实验一 混合流：60个http会话，8个ftp，1个cbr，每秒启动一个http会话，ftp在前10秒内随机启动，cbr在模拟实验开始时启动，表2是响应时间，丢包率，吞吐率，队列标准差对比。

表2 实验一结果对比

	http响应时间(s)	队列标准差(B)	丢包率	吞吐率
队头丢包	1.850	11116	0.0982	0.9411
选择性丢包	1.404	9006	0.0660	0.9433

实验二 混合流：120个http会话，8个ftp，1个cbr，每0.5秒启动一个http会话，ftp在前10秒内随机启动，cbr在模拟实验开始时启动。

表3 实验二结果对比

	http响应时间(s)	队列标准差(B)	丢包率	吞吐率
队头丢包	2.693	12962	0.1395	0.9424
选择性丢包	1.840	8588	0.0790	0.9443

实验三 纯Wed流：240个http会话，每0.25秒启动一个会话。

表4 实验三结果对比

	http响应时间(s)	队列标准差(B)	丢包率	吞吐率
队头丢包	1.764	12112	0.1176	0.8885
选择性丢包	1.675	10165	0.0819	0.8916

实验四 纯Web流：480个http会话，每0.125秒启动一个会话。

表5 实验四结果对比

	http响应时间(s)	队列标准差(B)	丢包率	吞吐率
队头丢包	3.674	13375	0.2204	0.9160
选择性丢包	3.341	7281	0.1912	0.9215

上述实验结果表明，采用选择性丢包，缩短了http的响应时间，提高了吞吐率，降低了丢包率。队列的标准差反映了队列的稳定性，进而影响端的RTT的计算，采用选择性丢包的RED队列的队列标准差较小，因而稳定性更好。表2和表4是负载较轻的情况，表3和表5是网络负载较重的情况，采用队头丢包，队列标准差随负载加大而加大，进一步验证了原RED在负载较重的情况下，队列稳定性较差的特点。而采用选择性丢包以后，队列的标准差却随负载增加而减小了，这是由于选择性丢包在负载较重的情况下并不是盲目地丢弃达到的数据包，而是选择能够有效减轻负载压力的拥塞窗口较大的连接丢包。在丢包率方面，当负载是混合流时效果尤其明显，图3是一个采用选择性丢包的RED丢包概率和连接数据包分布图。从图中可以看出，在选择性丢包中，连接的丢包概率随该连接占用的带宽的增加而加大。图4是实验二中两种丢包机制的队列对比，与队头丢包相比较，采用选择性丢包的RED队列抖动较小。

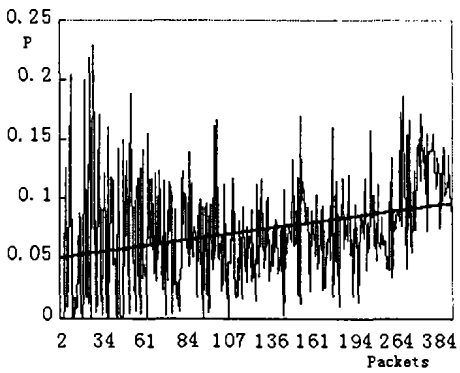
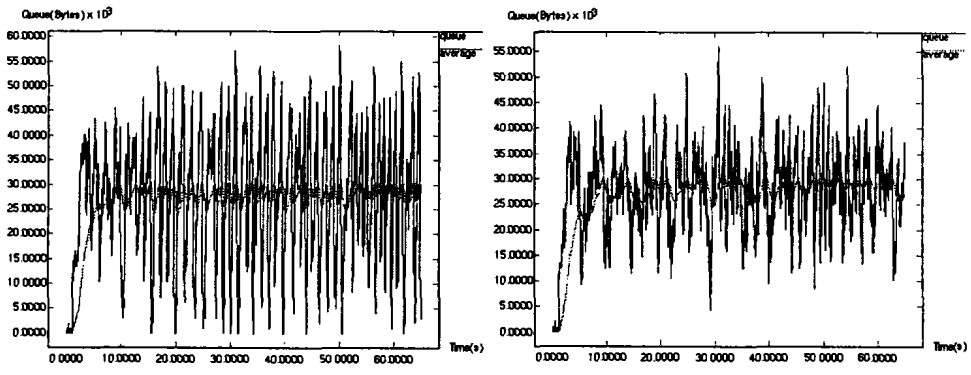


图3 采用选择性丢包的RED丢包概率与连接数据包分布图



(a) 队头丢包　　(b) 选择性丢包

图4 两种丢包策略的队列

(下转第162页)

们的初始值，然后进行下一个 λ 次的迭代，得到一个标准的 SOM 网络。

依次迭代下去，直至最后所有的神经元都满足全局控制的停止条件。

3 仿真实验及分析

为了充分展示新模型在模式识别中的应用，我们人工定义了如表1所示的输入向量矩阵，在这些输入向量之间有明显的分级结构，并且将输入向量标记为 A 到 Z。我们采用自增长型自组织映射网络对其进行分类，通过分级的网络能够直观地表示出这种结构。

表1 输入向量矩阵

属性	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A1	1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
A2	0	0	0	0	0	1	2	3	4	5	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
A3	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	3	3	3	3	3	3	3	3
A4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	1	2	3	4

含有4个神经元的中间层和含有70个神经元的最后层分类结果如下：

ABCDE	KLMNOPQR
FGHIJ	STUVWXYZ

4个神经元

BCDE * QR * YZ
A * * * * P * * X *
* F * NO * W * * *
* G * M * * * * * *
HJKL * TUV * *
* I * * * * S * * *

70个神经元

从实验结果可以看出，在中间层将输入分成了4大类，最终将它们按照彼此类似关系，一个个映射到不同的分类上去，越相似的神经元离得越近，而且从自增长过程中我们可以看到大类和小类之间的分级结构，并产生了一个合适规模的网络结构和神经元排列来分析输入的向量集。

结束语 本文分析了自组织映射网络的基本原理，提出了不同于传统模型的自增长型自组织映射网络模型，并详细阐述了自增长的训练过程及进行全局控制的数学标准。通过实验仿真结果，我们看到这种动态的结构模型不仅能够更有效地对高维数据进行模式识别，大大减少因不准确的预先估计带来的不必要计算，而且能将输入数据中的分级继承关系直观地展现出来。

参 考 文 献

1 Alahakoon D, et al, Dynamic self-organizing maps with controlled growth for knowledge discovery. IEEE Trans. Neural Networks, 2000, 11(1): 601～614

2 Kaski S, Kangas J, Kohonen T. Bibliography of self-organizing map (SOM) [J]. Neural Comput. Surveys, 1998, 1(3): 1～176

3 Suganthan P N. Hierarchical overlapped SOM's for pattern classification [J]. IEEE Transactions on Neural Networks, 1999, 10 (1): 193～196

4 Roussinov D G, Chen H. Information navigation on the web by clustering and summarizing query results [J]. Inform. Processing Manage, 2001, 37(1): 789～816

5 Jain A K, et al, Data clustering: A review [J]. ACM Compute. Surveys, 1999, 31(3): 264～323

(上接第43页)

结束语 主动队列管理是网络研究中的热点之一，拥塞程度的衡量和丢包策略的选择是实现有效的主动队列管理的关键技术。本文中提出的选择性丢包策略，通过记录连接的一些状态信息，将连接的传输阶段划分为启动阶段和稳定传输阶段，利用假轮转检测发送速率过大的连接，挑选出丢包的候选连接。当产生拥塞时，丢弃候选连接在缓冲区头部的数据包。实验结果表明，采用选择性丢包的 RED 算法，提高了队列的稳定性和网关的吞吐率，显著地降低了丢包率，缩短了 http 的响应时间。

参 考 文 献

1 Braden B, et al. Recommendations on queue management and congestion avoidance in the internet. RFC2309, 1998

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

3 Christiansen M, et al. Tuning RED for Web Traffic. In: Proc. of ACM SIGCOMM Conf. 2000

4 Iannaccone G, May M, Diot C. Aggregate Traffic Performance with active queue management and drop tail. 2001

5 Floyd S, Fall K. Promotion the use of end-to-end congestion control in the internet [J]. IEEE/ACM Transaction on Networking. Aug. 1999

6 Hartling M, Claypool M, Kinichi R. Active Queue Management for Web Traffic: [Technical Report]. May 2002

7 Mahajan M, Floyd S, et al. Controlling High-Bandwidth Flows at the Congested Router. In ICNP, Nov. 2001