数据流的活动队列管理算法: MBLUE

徐 建, 李善平
(浙江大学计算机学院, 浙江杭州 310027)

摘 要: MBLUE(Modified BLUE) 是一种面向数据流的活动队列管理算法. 它不是使用平均队列长度指示缓冲区拥塞状态, 而是使用数据报丢弃的频率和队列空闲程度来管理网络拥塞. 探测瓶颈连接早期的拥塞信息, 通过数据报的丢弃和标记避免拥塞. 它只维护一个先进先出队列, 以较少的数据流状态信息, 在不同流之间公平的分配网络带宽. 能够适应瞬时的猝发流, 能合理控制非 TCP 数据流, 又能够保持较短的平均队列长度, 从而控制、减轻网络拥塞. 通过 TCP/IP 网络的模拟, 证实算法在公平的分配网络带宽和降低数据报的丢失率上具有较好的鲁棒性.

关键词: 拥塞控制; 数据流; 队列管理
中图分类号: TP393 文献标识码: A 文章编号: 0372-2112 (2002) 11-1732-05

Flow Active Queue Management: Modified BLUE

XU Jian, LI Shan-ping
(*College of Computer Science, Zhejiang University, Hangzhou, Zhejiang 310027, China*)

Abstract: An active queue management algorithm for flows called MBUE is proposed. It uses packet loss and link utilization history to indicate the congestion, and not relies on average queue length as an estimator of congestion. MBLUE detects incipient congestion, and notifies connections either by dropping packets or explicit congestion notification (ECN). It makes no assumptions about queuing architecture and will work with a FIFO queue. There is small amount of each flow maintained to achieve fair bandwidth allocation among flows sharing the bottle-neck connection. This algorithm removes biases against bursty sources, and gives protection to fragile flows. It can manage non-adaptive flows also. Using simulation and experiments of TCP and UDP traffic, MBLUE is shown to perform better than other algorithms on both fairness and packet loss rate.

Key words: congestion control; flow; queue management

1 引言

今天, 在网络上正出现一系列新的应用, 如音频和视频, 它们要求对每个数据流提供能够保证的吞吐率[1]. 效率和公平已成为网络上带宽共享的两个主要目标. 理想状态下, 高效意味着低的延迟: 包括传输延迟和排队延迟, 并且没有带宽被浪费. 公平意味着在共享拥塞连接的用户中每一位都受到平等的对待. 但是在现实情况下, 数据流是波动的, 数据源倾向于占有更多的带宽. 所以, 必须有有效地机制来维护网络的效率和公平. 目前, 人们已经有了许多拥塞控制的包调度算法如 Hahne E 等人对数据流的 Round Robin[2], Floyd S 等的 RED[3], W Feng 等的 BLUE[4] 等.

RED(Random Early Drop)[3] 是包交换网络中的一个推荐的包调度算法. 它通过计算平均队列长度决定该节点是否存在拥塞, 通过丢弃数据报或使用 ECN(Explicit Congestion Notification)[5,6] 通知数据源、目的, 使数据源减小数据发送速率, 避免拥塞. 当平均队列长度超过预先设置的最小队列长度时, 到达队列的每个数据报将按照相应的概率丢弃或标记 ECN 标志位, 并且丢弃或标记的概率是平均队列长度的函数. 虽然 RED 比先前的一些队列管理算法如 ERD[7] 更加有效和公平, 但由于是依靠平均队列长度指示拥塞, RED 的设计目标如拥塞避免、全局同步和对猝发流的保护不是在任何情况下都能够达到的. 在拥塞十分严重的情况下, 平均队列长度并没有正确的指示拥塞. RED 能够达到理想的运行状态当且仅当是正确的配置并拥有足够的缓存. W Feng 等[4] 提出了一种不同的、使用数据报丢弃率和队列空闲程度指示拥塞的简单算法 BLUE. 它维护一个关于数据报丢弃或标记的概率, 当缓存溢出、队列连续丢弃数据报达到一定的时间长度就增加丢弃概率, 当队列连续空闲就减少丢弃概率. 然后根据该概率丢弃或标记后续到来的数据报, 通知数据源和目的有关该结点的拥塞状态, 使数据源调整数据发送的速率. W Feng 等通过模拟和真实的试验证实了算法即使在只有少量缓存的情况下也工作的相当好.

公平性是另外一个重要的目标. 例如适应流和非适应流之间的公平性在许多文章中都有讨论. Floyd S 等[2] 提出了一种与 RED 相应的识别不端用户(misbehaving user) 的方法.

收稿日期: 2002-02-05; 修回日期: 2002-07-06

W Feng 等[8]提出了 SFB 来测试和限制非适应性流的速率. 还有许多其它的关于带宽共享公平性的讨论[2,9,10]. 要获得带宽的公平共享, 需要一种基于流的包调度机制. Dong Lin 等[11]提出了 FRED, 一种对数据流实施 RED 的算法. 但一个显而易见的问题是算法要维护每一个流的状态信息或者为每一个数据流维护一个单独的队列, 将导致实现的复杂性增加. 所以又有人提出了一种结构: 核心无状态(core-stateless)[12], 这种结构只在网络边缘靠近数据源的地方维护每个流的状态, 而在网络的核心不针对每个数据流作调度。相应的区分边缘路由和核心路由, 在边缘路由依据某种策略进行数据流的分类、标记, 而在核心路由简单的维护一个 FIFO 队列. Z Cao 等[13]提出了 RFQ(Rainbow Fair Queuing), 在边缘路由将数据流按照速率分类, 然后在网络的核心进行调度. 还有 Antoine Clerget[4]提出的 TUF(Tag-based Unified Fairness) 等基本上都是属于这种边缘-核心分类思想的解决方法.

综上所述, 象 RED、BLUE 等算法根据数据流占有的网络带宽比例丢弃数据报, 将导致不公平的带宽分配, 而 RED 和其他的依靠平均队列长度指示网络拥塞的算法在拥塞严重的情况下不能正确的表示拥塞程度. 核心无状态结构在边缘路由上依然需要对每个数据流进行分类、维护每个流的状态, 基于数据流进行包调度. 所以, 如果能够减少网络必须维护的数据流状态的复杂性, 那么对数据流的包调度算法对网络拥塞控制依然是一种值得研究的方案. 在本文中, 提出了一种活动队列管理算法称为改进了的 BLUE, 使用类似 BLUE 的拥塞控制方法, 它保证每个共享瓶颈连接的数据流都公平的享有网络带宽, 不管是否是适应流或非适应流, 猝发流或健壮流, 并且只维护在缓存中有数据报的数据流的状态信息, 减小了算法实现的复杂性.

文章接下来的内容如下安排: 第 2 部分描述了算法的设计目标、算法的简单实现, 第 3 部分使用该算法对不同的数据流进行了分析和模拟, 第 4 部分是总结.

在本文中, 将区分连接和数据流这两个概念. 连接指两个网络节点之间的物理连接. 数据流指某个数据源到一个相同的目的地的多个数据报. 使用 IP 分组头中的多个域: 如源和目的地址、源和目的端口号、协议标识域来进行数据流的分类.

2　算法及其简单的描述

本部分介绍算法设计的指导目标和算法的简单实现. 首先是通过观察数据报的丢失率和队列的空闲程度进而引导数据源调整数据发送速率以避免拥塞. 第二个目标是在网络拥塞的情况下, 通过丢弃或标记占有超过应得份额的数据流的数据报, 使瓶颈连接的带宽在共享用户之间公平的分配. 第三个目标是允许和保护短时间的猝发流. 第四个目标是提高连接的使用率、降低排队延迟. 前三点主要涉及公平性, 第四点强调的是效率.

算法的拥塞避免机制主要是探测早期的拥塞. 类似 BLUE, MBLUE 使用单个关于丢弃或标记的概率. 但与 BLUE 不同的是, MBLUE 不但在缓存连续溢出和数据报连续被丢弃时提高丢弃概率, 而且在队列的长度超过一定上限时提高丢弃概率; 不是在队列为空时, 而是在队列长度低于一定下限时, 降低丢弃概率. 当丢弃概率较高时, 算法认为网络中有可能发生拥塞, 将丢弃或标记随后到来的数据报. 这样适应性流的数据源将探测到数据报的丢失, 从而减少数据发送速率, 避免了可能发生的拥塞.

算法尽可能维护较少的数据流信息, 只使用一个先进先出队列, 只维护在缓存中保留有数据报的数据流状态; 仅使用数据流标识、字节数、数据报数等状态信息. 依照所维护的数据流信息, 判断一个流占有的带宽是否超过它应得的份额, 如果此时网络处于拥塞状态, 即使用反映拥塞状况的丢弃概率丢弃或标记数据报. 这样的一种机制保证了瓶颈连接的带宽能够公平的分配给共享该连接的所有用户. 在这里, 由于只标记特定的数据流, 所以避免了全局的同步.

算法中使用另外一个参数 freeze_time 控制丢弃概率的改变过程. 它定义了丢弃概率两次连续的改变之间的最小间隔. 这样只要缓存足够大, 就可以允许短时间的猝发流发生. 算法思想非常简单, 但它几乎给予了我们对网络中数据流完全的控制. 在后面的模拟试验中, 可以看到它的实际工作性能. 下面是对算法的简单描述:

```
Upon packet loss(or Q_len > L_max) event:
  If((now - last update) > freeze_time) then {
    p_marking = p_marking + d_i
    last_update = now
    }
Upon link idle(or Q_len < L_min) event:
  If((now - last_update) > freeze_time) then {
    p_marking = p_marking - d_d
    last_update = now
    }
Upon packet arriving event:
  If(flow_in_queue[i] > max_allow[i]) then
    Drop or mark packet by p_marking
```

算法中出现的参数描述如下:

Q_{len}:	队列长度
L_{max}:	设置的队列长度上限
L_{min}:	设置的队列长度下限
$p_{marking}$:	丢弃或标记的概率
last_update:	$p_{marking}$最后更新时间
freeze_time:	$p_{marking}$两次连续更新间隔时间
d_i:	$p_{marking}$每次增加的单位
d_d:	$p_{marking}$每次减少的单位
flow_in_queue:	数据流在队列中的数据报数或字节数
max_allow:	数据流被允许的最大的数据报数或字节数

在网络中通过连接的数据流数目非常大, 数据流的数目可能超过节点缓存中可以容纳的数据报数目. 在这种情况下, 算法将工作在大量数据流的处理模式下. MBLUE 尽可能的分配缓存空间给每一个数据流, 每个流在缓存中允许保存的数

据报不超过两个, 也即把 max_ allow 设为 2. 保证在数据流的数目非常大的情况下, 各个流也能够得到公平的带宽分配.

3 模拟分析

为了评估算法的性能, 我们使用事件驱动的面向对象的网络模拟器 ns[15] 进行了以下试验. 在试验中, 每个 IP 数据报的长度都设置为 1kb, 并且假设每个数据源都有无限的数据可供发送.

3.1 MBLUE 与 RED、BLUE 算法的比较

在如图 1 的拓扑结构的网络上, 设置瓶颈连接 $G0$ 到 $G1$ 的带宽为 10Mb, 延迟为 10ms. S 系列节点到 $G0$ 的带宽为 100Mb, 延迟从 5, 6, 7, …, (5+ n) 不等. $G1$ 节点到 R 系列节点的带宽也是 100Mb, 延迟是 5ms. 这样各个连接总的的传输延迟就分布于 20ms 到(20+ n) ms 之间. S 系列节点到 $G0$ 的连接和 $G1$ 到 R 系列的连接使用 Drop tail 队列管理算法. $G0$ 与 $G1$ 的连接分别采用 MBLUE、RED、BLUE 队列管理算法进行试验. 在这次试验中, 取 n 为 25, 即有 25 个连接通过瓶颈连接 $G0$ 到 $G1$. 在 25 个连接中, 4/5 的连接使用 TCP/ Reno, 1/5 的连接使用 UDP. 使用 Pareto on/ off 数据源, on 时间为 3s, off 时间为 2s. 25 个数据源全部在试验的第一秒内随机的启动开始发送数据. 每次的模拟时间都为 100s, 然后统计数据报的丢失率. 试验中平均队列长度的计算统一使用 RED 算法中的方法.

RED 的配置是 $\min_{th}$设置为 1/5 的最大队列长度, $\max_{th}$设置为 3/5 的最大队列长度. 其他的参数使用 ns 中的默认配置. RED 算法采用 ns 中实现的代码.

BLUE 的配置是 freeze_ time 为 10ms, d_i 为 0.02, d_d 为 0.002. BLUE 算法采用 W. Feng 等提供的代码.

MBLUE 算法的 L_{max}取 4/5 的最大队列长度, L_{min}取 1/5 的最大队列长度, max_ allow 取(最大队列长度/act_ flow), act_ flow 是缓冲区中缓存有数据报的数据流数目, 其它的参数同 BLUE.

图 2 是试验中统计的结果. 从结果中我们证实 BLUE 的数据报丢失率确实低于 RED, 但试验中没有出现象 W. Feng 等[4]那样显著的差别. 随着缓冲区的增大, 瓶颈节点能够容纳更多的数据报, 三个算法的数据报丢失率都呈现下降趋势. 我们注意到 MBLUE 的数据报丢失率接近 BLUE, 这显示 MBLUE 在降低数据报丢失率方面的性能接近 BLUE 或略好于 BLUE, 因为多次试验重复了同样的结果.

图 3 是最大队列长度为 50 时, 分别使用三种算法模拟获得的平均队列长度的变化情况. 由于 MBLUE 使用了队列的上限 L_{max}和下限 L_{min}, 在拥塞发生即缓冲区溢出之前就开始增加丢弃概率 $p_{marking}$, 使算法能够较早开始察觉到拥塞, 有目的地丢弃或标记数据报, 使数据源减小数据发送速率, 避免拥塞的发生. 相反, 在队列长度下降到其下限时, 开始减小丢弃概率 $p_{marking}$, 避免使瓶颈连接的利用率降低至零. 两个方面的约束, 使队列长度维持在一个稳定水平, 即避免了拥塞的发生, 也使连接的利用率得到了最大化, 提高了网络效率. 在图中不难看出 MBLUE 与 RED 的平均队列长度有相似变化趋势, 这反映了它们基本上都可以维持较高的连接使用率, 也具有相似地排队延迟, 而 BLUE 的平均队列长度变化波动特别的大.

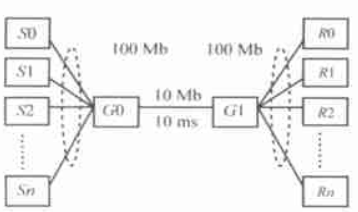

图 1 网络拓扑结构

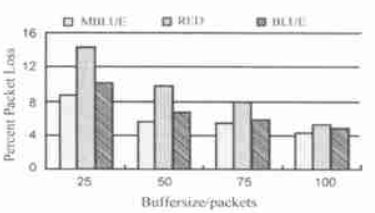

图 2 25 个数据流的数据报丢失率比较

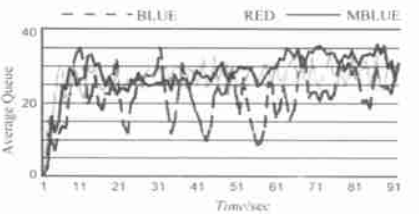

图 3 平均队列长度变化比较

3.2 多个数据流的公平性分析

这一小节分析算法对通过它的各个数据流带宽分配的公平性. 当多个用户共享多个资源的时候, 经常采用的是最大-最小公平标准. 根据这个标准, 在某个特定的瓶颈连接的多个数据源, 应该享有公平的带宽. 即对所有的 i, j , 应有 $b_i(t) = b_j(t)$. 如果各个数据源享有的带宽不相等, 那这种分配是不公平的. 假设对 N 个数据流有一种带宽分配 $B = \{b_i\}_{i<N}$, 如果它满足容量限制

$$\sum_{i<N} b_i \leqslant c_l$$

其中 C_l 是连接的带宽, 那么就是可行的. 对于不同的分配, 我们使用下式来衡量它的公平性[16]:

$$F = \frac{\left(\sum_{k=1}^{N} b_i\right)^2}{N\sum_{k=1}^{N} b_i^2}$$

其中 N 是数据流的数目, b_i 是第 i 个数据流占有的带宽. 公平性的取值显然在 0- 1 之间, 一种完全的不公平分配 F 的取值将是 $1/N$, 对于完全的公平分配, F 的取值将是 1. F 的计算结果与带宽描述使用的单位无关, 并且是一个连续函数.

试验的网络还是图 1 中的网络. 但是使用不同的数据源, 在试验中 4/5 的连接使用 FTP 数据源, 1/5 的连接使用非适应的 CBR 数据源. 另外, 试验中为了使 $G0$ 和 $G1$ 连接成为瓶颈连接, 将 $G0$ 到 $G1$ 的带宽改为 2Ms, 传输延迟时间不改变. 这样, $G0$ 和 $G1$ 之间 2Ms 的带宽将在多个数据流之间分配. 试验分三次: 25 个连接, 50 个连接, 75 个连接. $G0$ 和 $G1$ 之间的连接使用 MBLUE 算法进行队列管理, 队列最大长度取 100. 不断的改变 CBR 数据源的大小, 进行 100ms 的模拟, 然后分别测试计算 F 值. 根据得到的数据绘制成图 4, 图中横坐标是各个 CBR 数据源总的 CBR 速率, 纵坐标是 F 值. (注: 各个数据源总的发送速率可以大于瓶颈连接的带宽, 但是各个数据源/数据流实际能够获得的总带宽一定小于瓶颈带宽. 这里为了说明 MBLUE 在不同的数据源/数据流之间分配瓶颈带宽的公平

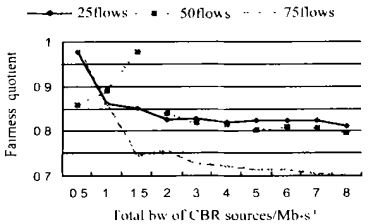

图 4　多个数据流的公平性

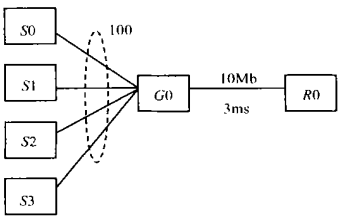

图 5　网络拓扑结构 2

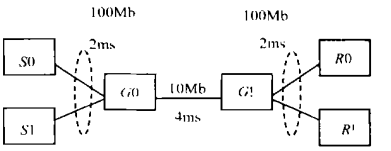

图 6　网络拓扑结构 3

性, 没有给出各个数据源/数据流实际获得的带宽, 而是给出了计算后的 F 值.)

从图 4 中可以看到 MBLUE 能够取得的公平性保持在 0.7 以上, 如果算法不是工作在大量数据流的处理模式下(如图中的 75 个数据流的曲线), 公平性可以保持在 0.8 以上. 当 CBR 数据源增加数据发送总量时, 公平性缓慢下降, 但总保持在一个稳定的水平. 当每个 CBR 数据源的发送速率取值都接近它应有的带宽份额时, 总的公平性值可以达到 0.95 以上, 如 25、75 个数据流曲线在 0.5 的取值和 50 个数据流曲线在 1.5 的取值.

3.3　对长延迟数据流的保护

在这个试验中使用图 5 的网络结构.

S 系列到 $R0$ 节点都使用 TCP 连接, 类型为 TCP/Reno. $S0$、$S1$、$S2$ 到 $G0$ 使用 1ms 的延迟, $S3$ 到 $G0$ 使用 15ms 的延迟. 结果是使 $S0$、$S1$、$S2$ 到 $R0$ 总的传输延迟是 4ms, $S3$ 到 $R0$ 总的传输延迟是 18ms. 18ms 的延迟使 $S0$ 到 $R0$ 连接的 RTT 时间大于 36ms, 相对于其他延迟较短的几个连接, 就形成一个非健壮的长延迟数据流. $G0$ 到 R0 的队列分别使用 MBLUE、RED、BLUE, 队列长度统一为 50. 数据源使用 FTP. 几个数据源在试验开始的 0.1 秒的时间里随机启动, 试验重复 10 次, 统计得到表 1 和表 2.

TCP0– TCP3 分别表示 $S0$、$S1$、$S2$、$S3$ 到 $R0$ 连接上的数据流, 从表中可以看到 MBLUE 算法下, 几个数据流比较公平的共享 $G0$ 到 $R0$ 的瓶颈连接, 与 25% 的公平份额差距最大的是 2.29 个百分点. 而 RED、BLUE 算法中出现的差别是 12.16、13.34 个百分点, 两者相差一个数量级. 再看数据报的丢失率, 在 MBLUE 算法中, 长延迟的数据流的丢包率是较低的. 所以 MBLUE 很好的保护了有较长传输延迟的数据流, 对延迟短的健壮的数据流进行了限制, 从而提高了公平性.

表 1　数据流占有的带宽份额(Percent)

	TCP0	TCP1	TCP2	TCP3
MBLUE	24.20	24.42	24.09	27.29
RED	28.73	28.76	29.67	12.84
BLUE	29.10	29.39	29.85	11.66

表 2　数据流的数据报丢失率(Percent)

	TCP0	TCP1	TCP2	TCP3
MBLUE	1.49	1.49	1.43	0.00
RED	1.60	1.66	1.51	1.48
BLUE	1.49	1.44	1.25	1.96

3.4　对非适应流的控制

最后一个试验是针对适应性流和非适应性流之间的公平性, 如 TCP 和 UDP 流之间的公平性. 试验使用图 6 中类似哑铃的拓扑结构, 各种配置如图. 在 $G0$ 和 $G1$ 的连接上使用 MBLUE、RED、BLUE 几种不同的队列管理算法, 其他的连接使用 Drop tail 算法, 最长队列长度取值 50. $S0$ 和 $R0$ 之间使用 TCP/Reno, 数据源是 FTP, $S1$ 和 $R1$ 之间使用 UDP, 数据源使用 CBR. 试验持续 10ms 的模拟时间, 两个数据源在试验开始的 0.1 随机启动, 试验重复三次, 然后根据所得数据计算两种数据流在不同模拟时间占有的瓶颈连接 $G0$ 和 $G1$ 带宽的百分数.

表 3　TCP/ UDP 数据流在几种算法中的带宽份额(Percent)

Algorithm	MBLUE		RED		BLUE	
Time (sec)	UDP	TCP	UDP	TCP	UDP	TCP
1	80	20	26	74	85	15
2	66	34	87	13	84	16
3	57	43	80	20	76	24
4	56	44	81	19	81	19
5	57	43	80	20	78	22
6	56	44	80	20	82	18
7	55	45	85	15	77	23
8	57	43	82	18	76	24
9	60	40	79	21	83	17
10	56	44	80	20	77	23

表 3 中的数据显示了 TCP、UDP 两种数据流在 RED 和 BLUE 算法里不平等的带宽分配, 比例大致是 8: 2, 而在 MBLUE 中只是在开始的 1m 钟时间里 UDP 占有较大的份额, 以后的比例就慢慢地接近 5: 5. 试验证实本文提出的算法能很好的控制非适应性数据流如 UDP, 而保护了 TCP 这种适应性流享有带宽资源的公平性.

4　结论

在本文中, 针对当前使用队列平均长度来指示拥塞程度的算法的不足, 对按照数据流使用瓶颈连接的带宽大小随机的选择丢弃或标记数据报将导致不公平的分配这一现象, 提出了一种有选择的丢弃或标记数据流的算法, 从而保护那些所使用的网络带宽小于它应享有份额的数据流, 限制健壮的数据流过多的不公平的占用网络带宽资源.

MBLUE 算法, 使用数据报丢弃的频率和队列空闲程度来

指示网络拥塞,只对缓存中有数据报的数据流维护最低限度的流状态信息,降低了算法实现的复杂性. 它只使用一个先进先出队列结构,在多个共享瓶颈连接的数据流之间实现了带宽分配的公平性. 模拟试验证实它比 RED、BLUE 具有更好的公平性,并且在降低数据报丢失率方面也有很好的性能.

参考文献:

[1] Marjory S Blumenthal, David D Clark. Rethinking the design of the Internet: the end-to-end argument vs. brave new world[J]. ACM Transactions on Internet Technology, 2001, 1(1): 70-109.

[2] Hahne E, Gallager R. Round Robin scheduling for fair flow control in data communications networks[D]. Cambridge: MIT, December, 1986.

[3] S Floyd, V Jacobson. Random early detection gateways for congestion avoidance[J]. IEEE/ACM Transactions on Networking, 1993.

[4] W Feng, D Kandlur, D Saha, K Shin. Blue: a new class of active queue management algorithms [DB/OL]. UM CSE-TR-387-99, http://thefengs.com/wuchang/blue/, 1999-04.

[5] S Floyd. TCP and explicit congestion notification[J]. Computer Communication Review, 1994, 24(5): 10-23.

[6] K Ramakrishnan, S Floyd. A proposal to add explicit congestion notification (ECN) to IP[DB/OL]. RFC2481, http://www.ietf.org/, 1999-01.

[7] Hashem E. Analysis of random drop for gateway congestion control[R]. Report LCS TR-465, MIT, Cambridge, MA: Laboratory for Computer Science, 1989.

[8] W Feng, D Kandlur, D Saha, K Shin. Stochastic fair blue: a queue management algorithm for enforcing fairness[A]. In Proceedings of INFOCOM 2001[C]. Alaska: April 2001.

[9] D Stiliadis, A Varma. Efficient fair queuing algorithms for packet-switched Networks[J]. IEEE/ACM Trans. Networking, 1998, 6(2): 175-185.

[10] T Bonald, L Massoulié. Impact of fairness on Internet performance[DB/OL]. http://research.microsoft.com/.

[11] Dong Lin, Robert Morris. Dynamics of random early detection[A]. In Proceedings of ACM SIGCOMM'97[C]. September 1997.

[12] I Stoica, S Shenker, H Zhang. Core-stateless fair queuing: achieving approximately fair bandwidth allocations in high speed networks[A]. In Proceedings of ACM SIGCOMM'98[C]. Vancouver: September 1998.

[13] Z Cao, Z Wang, E Zegura. Rainbow fair queueing: fair bandwidth sharing without per-flow state[A]. In Proceedings of INFOCOM'99[C]. Tel-Aviv, Israel: March 2000.

[14] Antoine Clerget, Walid Dabbous. TUF: tag-based unified fairness[A]. In Proceedings of INFOCOM 2001[C]. Alaska: April 2001.

[15] ns-2[CP/OL]. http://www.isi.edu/nsnam/ns/.

[16] D Chiu, R Jain. Analysis of the increase and decrease algorithms for congestions avoidance in computer networks[J]. Computer Networks and ISDN Systems, 1989, 17(5): 1-14.

作者简介:

徐 建 男,1975 年 2 月出生于浙江缙云,计算机科学与技术专业博士研究生,主要研究方向为操作系统,计算机网络等.

李善平 男,1963 年 8 月出生于浙江宁波,计算机应用专业博士,教授,博士研究生导师,主要研究领域为嵌入式操作系统(特别是 Linux 操作系统),信息集成技术等.