基于排队时延和丢包率的拥塞控制

谢 钧[①] 俞 璐[②] 金凤林[①]
[①](解放军理工大学指挥自动化学院 南京 210007)
[②](解放军理工大学通信工程学院 南京 210007)

摘 要：作为拥塞度量，排队时延具有很多优点，但仅利用排队时延并不能完全避免丢包，而在链路缓存不足出现丢包时，排队时延已不能有效反应网络拥塞情况。该文提出了一种基于排队时延和丢包率的拥塞控制模型，该模型采用双模控制的方法。在瓶颈链路上有足够缓存时，模型利用排队时延作为拥塞度量，使各流获得稳定的动态性和成比例公平性。当瓶颈路由器上没有足够缓存不可避免要丢包时，模型利用丢包率作为拥塞度量，使各流仍能获得与不丢包情况下相近的流特性。模型在两种模式的切换中保持稳定，实现平滑过渡。

关键词：网络拥塞控制；拥塞度量；排队时延；丢包率
中图分类号：TP393 **文献标识码**： A **文章编号**：1009-5896(2010)09-2058-07
DOI: 10.3724/SP.J.1146.2009.01228

Congestion Control Based on Queuing Delay and Packet Loss Probability

Xie Jun[①] Yu Lu[②] Jin Feng-lin[①]
[①](*Institute of Command Automation, PLA University of Science and Technology, Nanjing* 210007, *China*)
[②](*Institute of Communications Engineering, PLA University of Science and Technology, Nanjing* 210007, *China*)

Abstract: As a congestion measure, queuing delay has many advantages. But packet loss can not be completely avoided if only queuing delay is used as a congestion measure. When the buffer is not large enough and some packets are dropped, queuing delay can not accurately reflect network congestion. In this paper, a congestion control model is proposed based on queuing delay and packet loss rate, in which dual-mode control is used. When the buffer is large enough, queuing delay is used as a congestion measure, the model achieves stable dynamic properties and weighted proportional fairness among heterogeneous flows. When the buffer is not large enough and packet loss is inevitable, packet loss rate is used as a congestion measure, and the model maintains almost the same performance as when there is no packet loss. The model is stable as it switches between the two modes and the transition is smooth.

Key words: Network congestion control; Congestion measure; Queuing delay; Packet loss probability

1 引言

拥塞控制是保证网络稳定运行的重要手段，其基本方法是发送方根据从网络获得的拥塞反馈信息调整发送速率。网络链路可为发送方提供显式或隐式的拥塞指示，TCP 及其各类改进算法通常不需要链路提供显式的拥塞指示。如果路由器不主动提供显式的拥塞指示，发送源只能利用丢包事件和往返时延/排队时延作为反馈信息。拥塞控制算法按照采用的拥塞指示信息可分为基于丢包和基于时延的拥塞控制。基于丢包的拥塞控制算法除当前的 TCP (Reno)外，还有 HS-TCP[1]，E-TCP[2] ，CUBIC[3] 等，而基于时延的拥塞控制算法主要有 FAST TCP[4]，TCP Vegas[5]，BIC-TCP[6]，文献[7]等。

作为拥塞反馈，与丢包率相比，排队时延本身就是多比特信息，通常更易测量，因此基于时延的模型通常能比基于丢包信息的模型获得更加平滑的吞吐量[4]。另外，基于排队时延的拥塞控制算法可避免或尽量减少网络中丢包事件的发生，起到一定的拥塞避免的作用。在基于时延的拥塞控制算法中，近年来提出的FAST TCP 充分发挥了利用排队时延作为反馈信息的上述优点，在大带宽-时延积的网络中可获得稳定的流动态性，对不同 RTT(Round-Trip Time)的异构流具有加权成比例公平性。由于 FAST TCP 的优良特性，近年来出现了大量相关研究[8−11]。FAST TCP 以排队时延作为拥塞量度，控制发送速率使各流在瓶颈链路缓存中的分组维持在

2009-09-15 收到，2010-01-25 改回
国家部委基金资助课题
通信作者：谢钧 xiejun73@263.net

一个适当的数目，从而各异构流公平共享瓶颈带宽。但 FAST TCP 存在一个难以克服的问题，即当路径中的路由器缓存不够大时，共享瓶颈链路的流在到达平衡点前会遭遇丢包。此时 FAST 仍然要采用类似 AIMD 的算法，从而失去稳定的流动态性和成比例公平性[4]。在高速网络中，由于技术或成本的原因，路由器很可能只有小的缓存[12−14]。另外当与其它基于丢包的拥塞控制算法(如：Reno)共存时，不可避免会因为这些流挤占带宽而导致丢包。因此，对丢包事件的特殊处理使得FAST TCP在理论上很完美但在实际有背景流的条件下却不能获得理想的性能。文献[11]对此有所改进，但以损失一定的公平性和反应速度为代价。

研究表明，时延和丢包之间的相关性很弱，很难利用时延准确预测丢包[15]。即仅利用排队时延不能完全避免丢包，一旦丢包，排队时延就不再增长，这时利用排队时延作为拥塞度量已不能有效控制发送速率。因此本文提出了一种基于丢包率和排队时延的拥塞控制模型，采用双模控制方法，在瓶颈路由器上有足够缓存时，尽可能发挥用排队时延作为拥塞度量的优点，尽量避免丢包，使各流获得稳定的动态性和成比例公平性。而当瓶颈路由器上没有足够缓存时，丢包率较大时，模型以丢包率为拥塞度量，使各流仍能获得与不丢包情况下相近的流特性。该模型在这两种模式的切换中仍能保持稳定性，实现平滑过渡。

2　基于丢包率和排队时延的拥塞控制算法

2.1 模型的设计思路

作为拥塞度量，虽然排队时延比丢包率有更多的优势，但仅利用排队时延并不能完全避免丢包。在网络不发生丢包或者丢包较少时，选择排队时延作为拥塞度量能获得稳定的流动态性，而当丢包较大时，选择排队时延已不能正确反应网络拥塞情况，必须对丢包事件做出反应。因此，需要在这两种工作模式间寻找一种平滑过渡的方式，两种情况下算法在平衡点、稳定性和公平性上要尽可能相近，同时在切换过程中仍能保持流的稳定性。

无论是基于丢包率还是基于排队时延，都不乏高效稳定的成熟模型，可选择合适的模型使其在新系统的两种模式下分别发挥作用。但模型的选择不仅要考虑各自的性能，更重要的是两种模型结合的可行性。为保证新模型不同工作模式下的平衡点具有统一的公平性，两种模型的平衡点最好具有相同的公平性，例如同为“max-min”公平或成比例公平。此外，系统还要满足以下条件：假设模型 A 的平衡点是 $[x_1,x_2,...,x_N]$，N 是流个数，模型 B 的平衡点是 $[y_1,y_2,...,y_N]$，为保证模型的稳定性，应限制 $x_i/y_i=k,i=1,2,...,N$，k 是与 i 无关的常数。

综合以上考虑，本文选择基于时延的 FAST TCP 模型和修改了的基于丢包率的 Kelly 模型，两个模型都具有高效稳定的特点，并且平衡点处吞吐量都符合或近似符合成比例公平。本文模型在文中称为 KFAST。

FAST TCP 假定链路缓存有能力容纳所有流的冗余分组，在平衡时每个流在链路缓存保持一定数目的冗余分组。其窗口 w 调整的递推公式为

$$w_i(t+1)=\gamma\left\{\frac{d_i w_i(t)}{d_i+q_i(t)}+\alpha_i\right\}+(1-\gamma)w_i(t),\quad i=1,2,\cdots,N \tag{1}$$

其中加权系数 $\gamma\in(0,1]$ 用来调整窗口变化的平滑程度，α_i 用来控制冗余分组的参数，N 是流个数，$q_i(t)$ 是流 i 在第 t 个调整时刻测量的排队时延，d_i 是流 i 的传播时延(往返)，$d_i+q_i(t)$ 为流 i 的往返时延。

FAST TCP 平衡点处发送速率具有 $x_i^*=\alpha_i/q_i^*$ 的形式，即平衡时刻流 i 在路径缓存中共维护 α_i 个冗余分组，其吞吐量符合 α_i 加权的比例公平。文献[16]证明了该模型在忽略反馈时延，单瓶颈链路条件下的平衡点具有全局渐近稳定性。

Kelly 模型[17]是一种经典的基于速率等式的拥塞控制模型，其速率调整的递推公式为

$$x_i(t+1)=x_i(t)+\theta_i(\eta_i-p_i(t)x_i(t)) \tag{2}$$

$p_i(t)$ 是网络给出的反馈，θ_i,η_i 分别是调整速率和公平性参数。Kelly 模型本身没有规定反馈的具体形式，可以是丢包信息，也可以是排队时延。选择不同的反馈，可以得到具有不同平衡点和公平性的模型。

在单瓶颈链路条件下，Kelly 模型在平衡点处吞吐量符合 η_i 加权的成比例公平，多瓶颈链路情形下，只要流获得的网络反馈等于流途经所有链路上的反馈之和，平衡点处吞吐量也符合 η_i 加权的比例公平。如果流获得网络反馈等于途经所有链路上的最大反馈，则平衡点处可获得符合 max-min 公平的吞吐量，如 MKC[18]。以丢包率为反馈，单瓶颈链路，各流具有相同反馈时延条件下 Kelly 模型的全局渐近稳定已由文献[18]证明。

本文要设计的是基于窗口等式的源算法，为此对 Kelly 模型做相应修改，并在此后描述本文模型时用 t 表示第 t 个时段，而非时刻。修改的 Kelly 模型中窗口调整的递推公式为

$$w_i(t+1)=(1-p_i(t))w_i(t)+\beta_i(d_i+q_i(t)) \tag{3}$$

$p_i(t)$ 和 $q_i(t)$ 分别是流 i 在时段 t 结束时刻测得的丢包率和排队时延，d_i 是流 i 的传播时延(往返)，参数 β_i 用于控制平衡时各流的丢包率。与 Kelly 模型不同，这里的 $p_i(t)$ 是源测得的丢包率，而不是网络提供的显式拥塞指示。当丢包率较小时，各流丢包率可以近似等于流途经的各链路丢包率之和。因此在丢包率较小时，多链路情形下平衡点处各流的吞吐量近似符合 β_i 加权的比例公平。平衡点处的发送速率都具有 $x_i^*=\beta_i/p_i^*$ 的形式，除了拥塞度量不同外，这一点与 FAST TCP 具有相同的形式。

由上可见，FAST TCP 与修改后的 Kelly 模型在平衡点处发送速率具有相似的形式，吞吐量也近似符合相同的公平性，这些都为它们的有效结合提供了保证。

除了选择合适的模型外，还要确定两种工作模式的切换方式。为保证平衡点的全局渐近稳定，在每种参数和网络条件下都只能有唯一平衡点，同时保证在有限时间之内，系统将不再需要在两种模式之间切换，并可持续地工作于一种模式之下。实际上，这种两种工作模式下的控制问题属于双模控制问题，其稳定性研究颇具难度。本文选择了一种实现上较为简单的切换模式，即优选两个模型更小的改变量作为新模型的改变量。实验表明，这种切换方式保证了系统的稳定性。在理论上，本文给出了简化情形下平衡点的唯一性和稳定性分析。

2.2 模型的数学描述

不考虑反馈时延的情形下，流个数记作为 N，流 i 在时段 t 内的窗口大小记作为 $w_i(t)$，则本文模型为

$$\left.\begin{aligned}
&w_i(t+1)=\min\{w_i^f(t+1),w_i^k(t+1)\},\\
&\quad i=1,2,\cdots,N;\ t=1,2,\cdots\\
&w_i^f(t+1)\triangleq\frac{d_i w_i(t)}{d_i+q_i(t)}+\alpha_i\\
&w_i^k(t+1)\triangleq(1-p_i(t))w_i(t)+\beta_i(d_i+q_i(t))\\
&\beta_i/\alpha_i=\beta_j/\alpha_j,\ i,j=1,2,\cdots,N,i\neq j
\end{aligned}\right\}\quad(4)$$

d_i 是流 i 的传播时延(往返)，$p_i(t)$ 和 $q_i(t)$ 分别是流 i 在时段 t 结束时刻测得的丢包率和排队时延，每个流都是一个 RTT 调整一次。由于各流的 RTT 不同，因此它们的调整周期并不相同。

对于单瓶颈链路情形，各流共享同一个链路，假定链路在选择数据包进行丢弃以及排队服务策略时对各个流是公平的，那么各流获得的丢包率和排队时延都相等。则式(4)中的 $p_i(t)$ 和 $q_i(t)$ 可分别记作为 $p(t)$ 和 $q(t)$。进一步地，如果单瓶颈链路情形下各流具有相同的传播时延 d，则 $p(t)$ 和 $q(t)$ 可通过下式计算。

如果 $\sum_i w_i(t)>Cd+L$

$$p(t)=\frac{\sum_i w_i(t)-Cd-L}{\sum_i w_i(t)},\ l(t)=L,\ q(t)=\frac{L}{C}\quad(5)$$

如果 $\sum_i w_i(t)\le Cd+L$

$$\begin{cases}
p(t)=0\\
l(t)=\begin{cases}\sum_i w_i(t)-Cd, & \sum_i w_i(t)-Cd>0\\ 0, & \text{其他}\end{cases}\\
q(t)=\dfrac{l(t)}{C}
\end{cases}\quad(6)$$

其中 L 是链路的缓存大小，$l(t)$ 是时段 t 结束时刻的队列长度，C 是链路带宽。

对式(5)，式(6)做简单的解释，流在时段 t 的开始时刻根据式(4)调整窗口大小，把窗口里的分组发完之后开始等待，一直到收到第 1 个分组的 ACK 以后进入时段 $t+1$。时段 t 发出的第 1 个分组的排队时间是由时段 $t-1$ 结束时刻链路缓存中的队列长度 $l(t-1)$ 决定的，那么第 1 个分组的往返时延就是 $d+q(t-1)$，也就是说时段 t 持续的时间长度是 $d+q(t-1)$。在此期间内，流向链路共发送了 $\sum_i w_i(t)$ 个分组，链路转发了 $C(d+q(t-1))$ 个分组，链路的缓存可以存放 $L-l(t-1)$ 个分组，链路共有能力处理 $C(d+q(t-1))+L-l(t-1)=Cd+L$ 个分组。如果流发出的分组个数超过了链路能处理的分组个数，在时段 t 的结束时刻，缓存被充满，并且发生丢包，即得到式(5)。

在不发生丢包的情形下，时段 t 期间，链路本来存放了 $l(t-1)$ 个包，流向链路共发了 $\sum_i w_i(t)$ 个包，链路转发了 $C(d+q(t-1))$ 个包，所以时段 t 的结束时刻的队列长度 $l(t)$ 应该等于 $l(t-1)+\sum_i w_i(t)-C(d+q(t-1))=\sum_i w_i(t)-Cd$，即得到式(6)。

实验结果表明式(4)定义的系统平衡点具有全局渐近稳定性，然而理论分析只能给出简化情形下的一些结论。

定理 1 设单瓶颈链路带宽为 C 缓存为 L，各流具有相同的传播时延 d。忽略反馈时延，并假设各流享有相同的排队时延和丢包率，则

(1)当 $L\ge\sum_i\alpha_i$ 时，系统的平衡点是

$$w_i^*=\frac{\alpha_i}{q^*}(d+q^*),\ i=1,2,\cdots N,\ p^*=0,\ q^*=\sum_i\alpha_i/C\quad(7)$$

(2)当 $L<\sum_i\alpha_i$ 时

(a)如果 $\left(\sum_i \alpha_i - L\right) \Big/ \sum_i \alpha_i \le \sum_i \beta_i \Big/ \left(C + \sum_i \beta_i\right)$，系统的平衡点是

$$\left.\begin{aligned} w_i^+ &= \frac{\alpha_i}{q^+}(d+q^+), \quad i=1,2,\cdots N \\ p^+ &= \frac{\sum_i \alpha_i - L}{\sum_i \alpha_i} \\ q^+ &= L/C \end{aligned}\right\} \tag{8}$$

(b)如果 $\left(\sum_i \alpha_i - L\right) \Big/ \sum_i \alpha_i > \sum_i \beta_i \Big/ \left(C + \sum_i \beta_i\right)$，系统的平衡点是

$$\left.\begin{aligned} w_i^- &= \frac{\beta_i}{p^-}(d+q^-), \quad i=1,2,\cdots N \\ p^- &= \frac{\sum_i \beta_i}{C + \sum_i \beta_i} \\ q^- &= L/C \end{aligned}\right\} \tag{9}$$

由于在 FAST TCP 模型达到平衡时，流 i 在链路缓存中保留 α_i 个冗余分组[4]。当 $L \ge \sum_i \alpha_i$ 时，链路缓存有能力容纳所有流的冗余分组，流可以达到式(7)定义的平衡点。如果 $L < \sum_i \alpha_i$，那么由式(7)定义的 w^*, p^*, q^* 将不再是平衡点，丢包不可避免，那么 FAST TCP 和 Kelly 模型中哪一个平衡点的丢包率更小，该平衡点就会成为系统的平衡点。

关于平衡点的全局渐近稳定性，本文给出了单瓶颈链路单流条件下的结论，如定理 2 所述。

定理 2　设单瓶颈链路带宽为 C，缓存大小为 L，只有一个流，则

(1)当 $L \ge \alpha$，系统的平衡点

$$w^* = Cd + \alpha, \quad p^* = 0, \quad q^* = \alpha / C \tag{10}$$

是全局渐近稳定的。

(2)当 $L < \alpha$，

(a)当 $(\alpha - L)/\alpha \le \beta/(C+\beta)$，系统的平衡点

$$w^+ = \frac{\alpha}{L}Cd + \alpha, \quad p^+ = \frac{\alpha - L}{\alpha}, \quad q^+ = L/C \tag{11}$$

是全局渐近稳定的。

(b)当 $(\alpha - L)/\alpha > \beta/(C+\beta)$，系统的平衡点

$$w^- = (C+\beta)(d + L/C), \quad p^- = \frac{\beta}{C+\beta}, \quad q^- = L/C \tag{12}$$

是全局渐近稳定的。

限于篇幅，定理 1 和定理 2 的完整证明在此略去。

3　仿真实验

3.1 NS-2 仿真

本文用 NS-2 对 KFAST 进行了仿真，并对 KFAST 的有效性和性能进行了验证。

在仿真实现中，用最小 RTT 来估计传播时延，与文献[4]的方法相同。算法的速率控制基本上是基于窗口的。但考虑到突发分组会使大 RTT 流比小 RTT 流在链路上遭遇更大的排队时延，导致不公平性，因此在实现中窗口内分组不是连续发送的，而是根据 RTT/W 计算窗口内分组发送的间隔，定时发送窗口内的分组。为避免不同速率源在瓶颈链路获得不同的丢包率，而导致不公平性，与文献[2]类似对实际设置的发送间隔时间进行了随机扰动(以 RTT/W 为均值的负指数分布)。

仿真实验的网络拓扑如图 1 所示。分组的大小 1000 byte，链路缓存管理采用弃尾算法。

3.2 单瓶颈链路

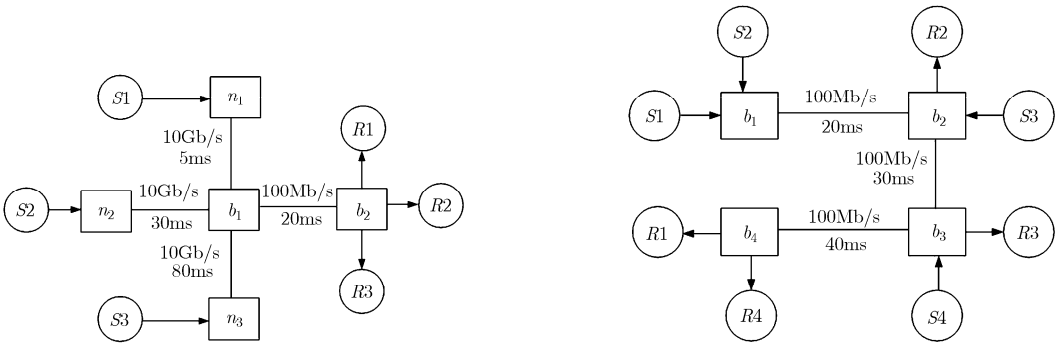
实验1　网络拓扑如图1(a)所示。$\alpha_i = 400$ 分组，$\beta_i = 40$ 分组/s，$i = 1,2,3$，瓶颈链路缓存 $L = 1000$ 分组，带宽 $C = 100$ Mb/s $= 12500$ 分组/s。流1从开始时刻一直持续到600 ms，流2在100 ms进入，500 ms离开，流3在200 ms进入，400 ms离开。在200 ms之前，链路缓存($L = 1000$)大于各流冗余分组个数之和($\alpha_1 + \alpha_2 = 800$)，因此链路没有丢包。在流3进入以后，各流冗余分组个数之和变成了 $\alpha_1 + \alpha_2 + \alpha_3 = 1200$，大于链路缓存，因此出现丢包。一旦流3离开，链路又恢复到无丢包的状态。 各流吞吐量以及丢包率随时间的变化如图2所示。其中图2(b)还显示了从300 ms到400 ms(系统处于平衡时)的累积丢包率。可以看出，算法在无丢包和有丢包情况下都能获得稳定、公平的流特性。当流比较少，链路缓存足够大时，算法在尽可能不丢包的前提下使各流获得尽可能大的带宽，并公平共享瓶颈带宽(与各流的RTT无关)。而当流比较多，链路缓存不能满足要求时，算法在维持一定丢包率的前提下，使各流获得与无丢包情况下相近的性能。

3.3 多瓶颈链路

实验 2　如图 1(b)所示，$\alpha_i = 200$ 分组，$\beta_i = 40$ 分组/s，$i = 1,2,3,4$，3 段链路 b_1b_2, b_2b_3, b_3b_4 均有缓存 $L = 500$ 分组，带宽 $C = 100$ Mb/s $= 12500$ 分组/s，在 3 段链路处分别有 $\alpha_1 + \alpha_2 = 400 < L, \alpha_1 + \alpha_3 = 400 < L, \alpha_1 + \alpha_4 = 400 < L$，因此每条链路上都没有丢包。

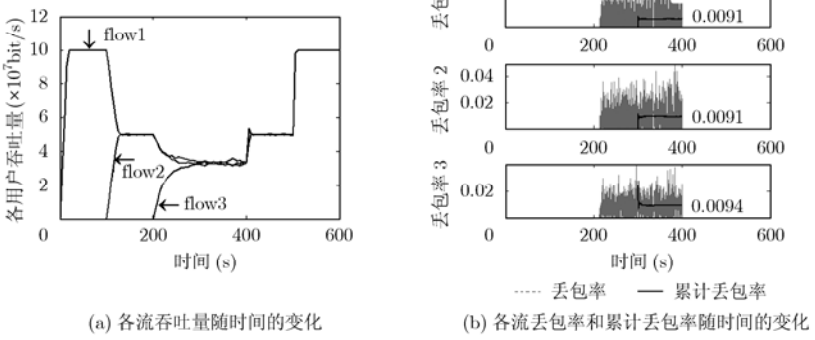
各流的吞吐量变化如图 3 所示(各链路丢包率均为 0)。可见当链路缓存足够大时，模型与 FAST TCP 的行为相似，获得稳定的流特性和成比例公平性。

实验 3　如图 1(b)所示，$\alpha_i = 60$ 分组，$\beta_i = 40$ 分组/s，$i = 1,2,3,4$，3 段链路 b_1b_2, b_2b_3, b_3b_4 均有缓存 $L = 60$ 分组，带宽 $C = 100$ Mb/s $= 12500$ 分组/s。对于流 1，假设它在每一段链路上的冗余分组都近



(a) 单瓶颈网络拓扑结构　　(b) 多瓶颈网络拓扑结构

图 1 实验中的网络拓扑



(a) 各流吞吐量随时间的变化　　(b) 各流丢包率和累计丢包率随时间的变化

图 2 单瓶颈链路实验 1 的结果

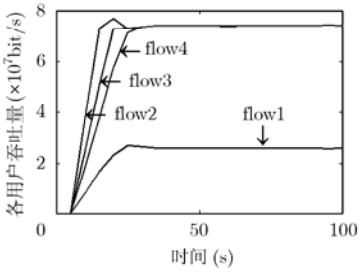


图 3 实验 2 的结果

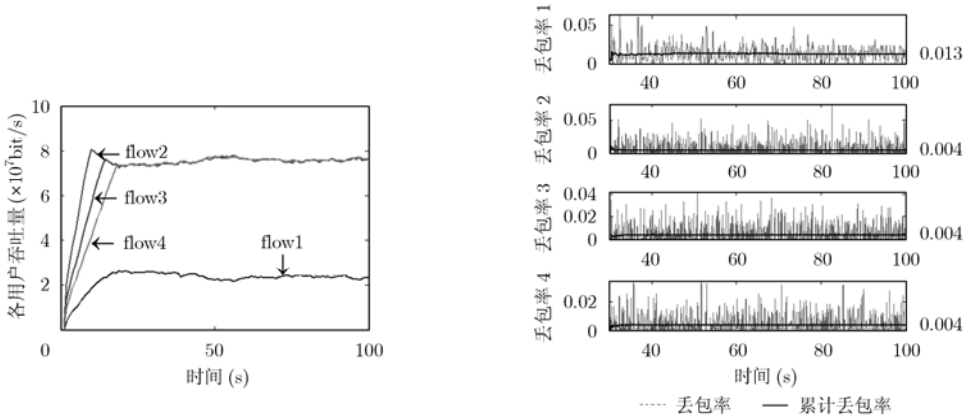
似相等。那么，在 3 段链路处缓存($L=60$)都小于冗余分组之和($60+60/3=80$)，系统达到平衡时，3 段链路上都会有丢包，每个流都会稳定在有丢包的平衡点，考虑到在这种参数条件下，修改的 Kelly 模型平衡时的丢包率远小于FAST TCP有丢包条件下的平衡时的丢包率，因此 4 个流都应该稳定在修改的 Kelly 模型的平衡点。各流的吞吐量、丢包率及累计丢包率随时间的变化如图 3 所示。

实验 4　如图 1(b)所示，$\alpha_i=40$ 分组，$\beta_i=40$ 分组/s，$i=1,2,3,4$，3 段链路 b_1b_2, b_2b_3, b_3b_4 带宽均为 $C=100$ Mb/s $=12500$ 分组/s，b_1b_2, b_3b_4 缓存为 $L_1=L_3=200$ 分组，b_2b_3 缓存为 $L_2=40$ 分组。显然，在 3 段链路处分别有 $\alpha_1+\alpha_2=80<L_1$，$\alpha_1+\alpha_3=80>L_2$，$\alpha_1+\alpha_4=80<L_3$，在系统达到平衡时，第 2 段链路 b_2b_3 会有丢包，则途经该链路的流 1 和流 3 会丢包。流 2 和流 4 没有丢包，应该稳定在 FAST TCP 的平衡点。流 3 丢包，且由于 $(\alpha_1+\alpha_3-L_2)/(\alpha_1+\alpha_3)>(\beta_1+\beta_3)/(C+\beta_1+\beta_3)$，由定理 1 流 3 应该稳定在修改的 Kelly 模型的平衡点。流 1 分别经历了有丢包和无丢包的链路，因此可能稳定在有丢包的平衡点，也可能稳定在无丢包的平衡点，根据调整规则，它会选择稳定在数值较小的平衡点。考虑到 $\alpha_1=\alpha_3$，如果流 1 稳定在有丢包率的平衡点，那么因为流 1 和流 3 丢包率相同，因此其平衡点也应该相同，那么流 1 在链路 b_2b_3 应该分得近似一半带宽的吞吐量，因为 3 段链路带宽相等，流 1 在其他两段链路上也会分得接近一半带宽，这显然将产生矛盾，因此流 1 只能稳定在 FAST 的平衡点。由于它经历 3 段链路，它的时延至少是流 2 的 2 倍，所以它能分得的带宽不会超过链路带宽的 1/3。各流的吞吐量、丢包率及累计丢包率随时间的变化如图 5 所示。

在实验 2 至实验 4 中，虽然参数设置不同，导致丢包情况各不相同，但 3 个实验中的吞吐量曲线比较接近。由此可见，在多瓶颈链路条件下，无论是无丢包、有丢包，还是部分流有丢包的情况，算法都能获得相近的公平性和稳定的流特性。

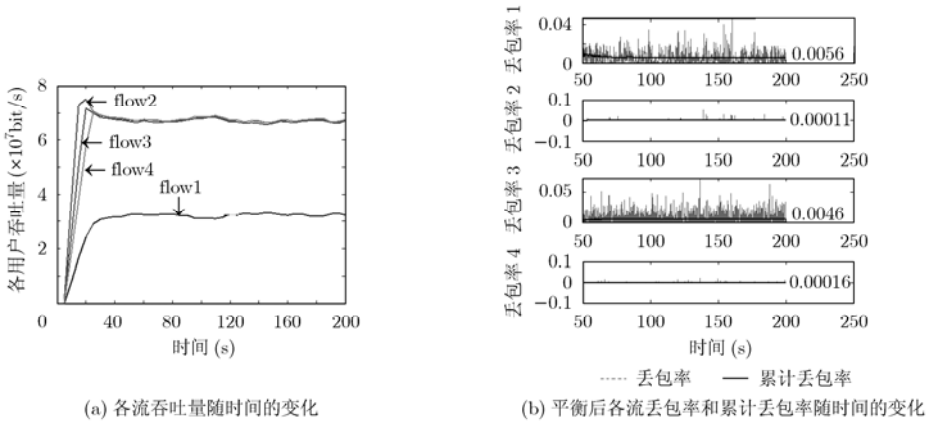
4 结束语

本文提出的基于丢包率和排队时延的 TCP 拥塞控制模型采用双模控制的方法，在瓶颈路由器有足够缓存时充分发挥排队时延作为拥塞度量的优点，尽量避免丢包，使各流获得稳定的动态性和成比例公平性。而当瓶颈路由器没有足够缓存，丢包率较大时，模型以丢包率为拥塞度量，使各流仍能



(a) 各流吞吐量随时间的变化　　(b) 平衡后各流丢包率和累计丢包率随时间的变化

图 4 实验 3 的结果



(a) 各流吞吐量随时间的变化　　(b) 平衡后各流丢包率和累计丢包率随时间的变化

图 5 实验 4 的结果

获得与不丢包情况下相近的流特性。该模型在两种情况的切换中保持稳定的流特性，实现平滑过渡。文献[19]提出了一种组合排队时延和显式拥塞标记作为拥塞度量的 FAST TCP 改进算法，但该算法需要路由器提供显式的拥塞指示，而本文算法不需要路由器的显式支持。

参 考 文 献

[1] Floyd S. High Speed TCP for large congestion windows. Internet draft draft-floyd-tcp-highspeed-02.txt, work in progress, http://www.icir.org/floyd/hstcp.html, 2003, February.

[2] Gu Y, Towsley D, and Hollot C V, *et al.*. Congestion control for small buffer high speed networks. Proceedings of IEEE INFOCOM 2007, New York, May 2007: 1037–1045.

[3] Ha S, Rhee I, and Xu L. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review*, 2008, 42(5): 64–74.

[4] Wei D X, Jin C, and Low S H, *et al.*. FAST TCP: Motivation, architecture, algorithms, performance. *IEEE/ACM Transactions on Networking*, 2006, 14(6): 1246–1259.

[5] Mo J and Walrand J. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 2000, 8(5): 556–567.

[6] Xu L S, Harfoush K, and Rhee I. Binary increase congestion control for FAST long distance networks. Proceedings of IEEE INFOCOM 2004, Hong Kong, March 2004: 796–805.

[7] Chen M Y, Zhang J S, and Murthi M N, *et al.*. Delay-based TCP congestion avoidance: A network calculus interpretation and performance improvements. *Computer Networks*, 2009, 53(9): 1319–1340.

[8] Belhaj S and Tagina M. VFAST TCP: An improvement of FAST TCP. Proceedings of the Tenth International Conference on Computer Modeling and Simulation, IEEE Computer Society, Los Alamitos, CA, USA, 2008: 88–93.

[9] Zhang H, Peng L, and Fan B, *et al.*. Stability of FAST TCP in single-link multi-source network. Proceedings of the 2009 WRI World Congress on Computer Science and Information Engineering, IEEE Computer Society, Los Angeles, California, USA, 2009: 369–373.

[10] Zhou J, Zhao F, and Luo Z. Parameter tuning of FAST TCP

based on Lyapunov function. Proceedings of the 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, IEEE Computer Society, Wuhan, China, 2008: 738–742.

[11] Yuan C, Tan L S, and Lachlan L H, *et al.*. A generalized FAST TCP scheme. *Computer Communications*, 2008, 31(14): 3242–3249.

[12] Appenzeller G, Keslassy I, and McKeown N. Sizing router buffers. Proceedings of ACM SIGCOMM 2004, Portland, Aug. 2004: 281–292.

[13] Goringsky S, Kantawala A, and Turner J S. Link buffer sizing: A new look at the old problem. IEEE Symposium on Computers and Communications (ISCC 2005), Murcia, Cartagena, Spain, June 2005: 507–514.

[14] Enachescu M, Ganjali Y, and Goel A, *et al.*. Part III: routers with very small buffers. *ACM SIGCOMM Computer Communication Review*, 2005, 35(3): 83–59.

[15] Martin J, Nilsson A, and Rhee I. Delay-based congestion avoidance for TCP. *IEEE/ACM Transactions on Networking*, 2003, 11(3): 356–369.

[16] Wang J, Wei D X, and Low S H. Modeling and stability of FAST tcp. Proceedings of IEEE INFOCOM 2005, Miami, FL, March 2005: 938–948.

[17] Kelly F P, Maulloo A K, and Tan D K H. Rate control for communication networks: Shadow prices, proportional fairness and stability. *Journal of Operations Research Society*, 1998, 49(3): 237–252.

[18] Zhang Y, Kang S, and Loguinov D. Delay-independent stability and performance of distributed congestion control. *IEEE/ACM Transactions on Networking*, 2007, 15(5): 838–851.

[19] Chen M Y, Fan X Z, and Murthi M N, *et al.*. Normalized queueing delay: congestion control jointly utilizing delay and marking. *IEEE/ACM Transactions on Networking*, 2009, 17(2): 618–631.

谢 钧： 男，1973 年生，博士，副教授，研究方向包括计算机网络、网络管理等.

俞 璐： 女，1973 年生，博士，讲师，研究方向包括计算机网络、智能信息处理等.

金风林： 男，1972 年生，博士生，讲师，研究方向包括计算机网络、网络管理等.