基于 RED 的差异型丢包队列管理算法

朱海婷 丁 伟
(东南大学计算机科学与工程学院 南京 211189)
(江苏省计算机网络技术重点实验室 南京 211189)

摘 要 网络流量中 UDP 成分的逐渐增加可能导致网络存在拥塞缓解失效的隐患. 通过引入 TCP 流量与非 TCP 流量的区分丢包互斥机制,使用 Lotka-Volterra 竞争模型证明在该机制下 TCP 与非 TCP 流量在网络中必然存在平衡点,作者提出基于 TCP 与非 TCP 差异型丢包的队列管理机制. 该机制依据 TCP 模型推导出的 TCP 协议流量的丢包概率,利用当前缓存队列中的 TCP 和非 TCP 数据包的状态,对不同的传输层协议产生差异型的动态丢包概率以确保 AQM 的稳定性和传输层协议间的公平性.

关键词 拥塞控制;主动队列管理;Lotka-Volterra 模型;区分丢包
中图法分类号 TP393 **DOI 号** 10.3724/SP.J.1016.2015.00567

Queue Management Algorithm Using Differential Dropping Based on RED

ZHU Hai-Ting DING Wei
(*School of Computer Science & Engineering, Southeast University, Nanjing* 211189)
(*Key Laboratory of Computer Network Technology in Jiangsu, Nanjing* 211189)

Abstract The gradually increasing UDP proportion in network traffic may lead to congestion mitigation failure. A queue management mechanism based on differential dropping with TCP and non-TCP traffic was proposed. A mutual exclusion differential dropping mechanism between TCP traffic and non-TCP traffic was introduced, and a balance point of the TCP and non-TCP traffic under the mechanism was proven to exist by using Lotka-Volterra competition model. Our algorithm also used TCP protocol's packet loss probability derived from TCP traffic model and adjusted packet loss probability according to the current queue cache status. Finally, dynamic packet loss probabilities in differences transport layer protocol to ensure the stability of AQM and fairness between the transport layer protocols.

Keywords congestion control; active queue management; Lotka-Volterra model; differential dropping

1 引 言

拥塞控制对互联网的稳定运行具有十分重要的意义,其中拥塞控制算法的研究也一直是互联网领域的热点问题. 一般而言,拥塞控制研究包括在网络边缘设备中使用的源算法和在网络中间节点上使用的链路算法两种. 源算法的原理是根据反馈信息调整源点端系统的发送速率;链路算法是在网络中间节点上检测拥塞,对拥塞进行反馈并进行相应的响应操作,以缓解拥塞.

拥塞控制算法需要针对其所在互联网本身的异构复杂性和拥塞控制分布性的特性,还需兼顾拥塞控制算法的性能要求,因此其算法设计有较高难度.

收稿日期:2013-07-11;最终修改稿收到日期:2014-11-28. 本课题得到国家"九七三"重点基础研究发展规划项目基金(2008BA37B04)和国家科技支撑计划(2009CB320505)资助.朱海婷,女,1983 年生,博士,讲师,主要研究方向为网络测量、网络安全、网络管理. E-mail: htzhu@njupt.edu.cn. 丁 伟,女,1962 年生,博士,教授,主要研究领域为网络测量、网络行为学.

尽管研究者在拥塞控制方向已开展了大量的研究工作,但是到目前为止拥塞控制问题还没有得到完美的解决.目前主流的链路拥塞控制算法是主动队列管理(Active Queue Management,AQM)[1-3],其中得到最广泛认可的随机早期检测算法 RED(Random Early Detection)[4].RED 是 IETF 推荐的基于中间节点的拥塞避免机制,研究表明 RED 比 DropTail 具有更好的性能,其主要优点是可以降低数据包丢包和降低数据包的经过延迟,同时还可以避免"死锁"情况的发生[4].虽然已有厂商设备能够支持,但因为 RED 有 2 个主要的问题,使其至今还没有在互联网中得到广泛的实际使用.其一是其性能对算法的参数设置十分敏感[5],其二是在公平性方面有所欠缺[6].

RED 会产生不公平现象的主要原因是其平等地看待每个报文.这对报文数量少的小流不公平,目前主要的与公平性相关研究工作均从这个角度展开[7-9],通过惩罚大流来降低小流报文被丢弃的概率.本文的研究也从公平性出发,但考虑的角度是传输层协议.

目前网络流量中 TCP 和 UDP 是使用率最高的 2 个传输层协议,根据 CAIDA 公布的数据①和我们在 CERNET 某省网边界的观测②表明二者合计可以占到总流量的 98%以上.很多研究直接或间接地表明 UDP 流量会影响 TCP 传输的通畅程度③,这种影响均是由拥塞导致的,如果在链路拥塞控制算法中平等地看待这两种协议,对 TCP 来说是不公平的,因为 TCP 使用端点拥塞控制算法,而 UDP 不使用.

关于 TCP 与 UDP 在实际流量中比例方面,文献[10]在对国内某运营商的实测数据进行分析的基础上认为从所有角度看 UDP 流量已经占用了全部网络资源的 50%左右.根据我们在本地被管网络上的观测,从 2005 年到起逐步从 5%增加至目前的 40%左右,而 CAIDA 的数据和文献[11]认为国外主干上 UDP 的比例在 5%~20%之间.文献[12]认为以 PPS 为代表的使用 UDP 的 P2P 视频软件是导致国内网络流量中 UDP 比例偏高的原因.

基于以上的分析,本文提出了一种基于 RED 的改进 AQM 算法 SF-RED,其核心思想是在拥塞发生时按不同的方案处理 TCP 和 UDP 报文,来达到总体更加公平的目的.算法在设计思路上借鉴了生物学中种群的竞争模型,通过将 TCP 与 UDP 作为争夺相同资源(带宽)的两种物种,对资源紧缺的情况下二者的竞争关系进行建模.在模型基础上,我们给出使用拥塞控制算法合理性的推理证明.与 RED 相比,SF-RED 算法还通过引入拥塞状态标识,对一次拥塞作为一个过程进行控制和管理,使用动态的参数设置丢包率来缓解与 RED 类似的参数敏感性问题.

2 相关工作

2.1 基于 RED 的 AQM 算法

RED[4]是最经典的 AQM 算法,也是本文算法工作的基础.基于 RED 进行改进的算法较多,具代表性的有 GRED(Gentle RED)④、SC-RED[13](Self-Configuring RED)、ARED(Adaptive RED)⑤等,他们都在 RED 的稳定性方面进行了改进.在公平性方面,较早对 RED 做出改进的有 FRED 算法[7](Flow RED),算法使用各个流在当前队列缓存中所占比例作为丢包判定指标,需要维护单流信息,因而不具备良好的可扩展性.后来这方面的研究逐渐转移为无需流信息维护的公平 AQM 算法,其中基于 RED 的算法是 RED-PD[14].RED-PD 算法在 RED 算法基础上加入了对当前丢包的记录,利用丢包记录来判断是否存在过度竞争或者存在非响应的流.它利用两个参数作为判断依据:(1)最近 N 次丢包记录;(2)某个流在这 N 次记录中出现次数.若该次数达到预先设定阈值,则判定为过度竞争流,并对其进行惩罚性丢包.但 RED-PD 算法的参数设置对算法效果影响较大,对拥塞响应时间也较长,而且针对短流丢包率较高[15].

2.2 其他改善公平性的 AQM 算法

除了在 RED 算法基础上进行公平性的修正外,还有其他一些在公平性上进行改进的拥塞控制算法.在不需单流信息的公平 AQM 算法中比较有影响力的包括 BLACK[9]、CHOKe[8]、SFB[16]、SAC[17]

① The Cooperative Association for Internet Data Analysis (CAIDA) [EB/OL]. http://www.caida.org/research/traffic-analysis/tcpudpratio/

② IPTAS[EB/OL]. http://iptas.edu.cn/src/system.php

③ La R J, Ranjan P, Abed E H. Nonlinear dynamics of mixed TCP and UDP traffic under RED. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.5899&rep=rep1&type=pdf

④ Floyd S. Recommendation on using the "gentle" variant of RED. http://www.aciri.org/floyd/red/gentle.html

⑤ loyd S, Gummadi R, Shenker S. Adaptive RED: An algorithm for increasing the robustness of RED's active queue management, 2001. http://www.icsi.berkeley.edu/pubs/networking/adaptivered01.pdf

和 DCN[18] 等.

BLACK[9] 算法通过对缓存队列进行数据包抽样，推断各个数据流在缓存中的占比，从而找出其中的竞争流并进行惩罚处理. CHOKe[8] 算法是一个不需要预先判定恶性竞争流，而通过概率方式能实现近似公平的 AQM 算法. CHOKe 算法采用 FIFO 队列，当需要丢包时则从当前缓存队列中随机抽取一个数据包，若该数据包与刚到达的数据包属于同一个流，则同时丢弃这两个数据包；否则，对取出数据包放回队列中，同时对刚到达的数据包进行丢弃判定. CHOKe 算法简单有效而且易于部署和实施，但当流数量增加时，其有算法效性也会下降. 在 CHOKe 的基础上，Jiang 等人[17] 提出了 SAC(Self Adjustable CHOKe)算法，其中增加了参数动态调整方案，使得 CHOKe 方法能够适应更多变的网络状况. Yang 等人[18] 提出了 AME-CHOKe，算法针对 CHOKe 中引入多次比较参数的动态设置方案. Eshete 等人[19] 提出 gCHOKe 算法，在 CHOKe 基础上通过增加对非响应流的惩罚粒度来保护适响应流[20]. WARD[21] 算法提出的一种“无状态”的近似公平 AQM 算法，它通过给队列不同的位置设置不同的权值，量化了各流的实际速率与理想(公平状态)速率的差异，借此判断是否需要丢包. SFB[16] 是 Stochastic Fair BLUE 的简称，其算法在 BLUE 算法基础上利用多级哈希散列的 BloomFilter，找出恶性竞争流. BLUE 用数据包的丢失率以及链路带宽利用率而不是队长来衡量链路的状况. SFB 算法对检测出的恶性竞争流仅仅设置一个速率上限. 但由于在实际环境中可以影响该阀值的因素太多，尚没有合理的计算方案，尤其在流数目增加时，SFB 算法的性能会下降[14]. DCN[18] (Differential Congestion Notification)算法，利用互联网中短流的数量占优，而长流的传输数据量占优的特点，算法仅对长流触发拥塞通知，保护短流. DCN 算法仅需要维护少量长流信息，算法实际是使用了较小的代价获得相对合理的网络公平性，同时确保较低的丢包率和较好的链路利用率. 但在实际网络中大量存在的扫描、DDoS 攻击等无功甚至有害流量均为短流，因此这个算法不适合在高扫描和 DDoS 攻击的环境下使用.

另外，借鉴其他思想的 AQM 算法也有推陈出新，如控制论思想的 PI[1] 算法、REM 算法[5]、引入虚拟队列概念 AVQ 算法[2]. 但它们在实用方面还存在问题，需要进一步的研究.

从另外一个视角看，网络中的带宽分配上异质流的公平性问题在上述算法中目前还没有针对性的解决方案. 在拥塞控制基础工作的机理模型上，对 TCP 协议的传输流量模型等，目前已经有较为深入的研究[22-24]，UDP 协议或其他非响应的协议控制和数据流虽然相对简单，但相对应的理论模型目前还有待完善，以 UDP 为代表的非响应流对 TCP 流的影响机理及其网络模型的研究工作仍还有待深入. 关于 TCP 和 UDP 协议流共存时，UDP 流对 TCP 流的影响的机理目前还没有相关理论上的深入研究[25]，仅 Song 等人[26] 提出了基于性能测量基础上的高速网络传输层拥塞控制方案.

3　SF-RED 队列管理算法

3.1　RED 队列管理

RED[4] 的基本思想是通过 Exponential Weighted Moving Average(EWMA)方法(具体表现为式(1))，其中 w_q 为常数，q 为瞬时队长)维护面向每个到达的数据包的平均队长测度 q_{avg}，并用该测度的值与固定阀值的关系确定丢包率(式(2))，具体丢包率与平均队列长度的关系如图 1.

$$q_{avg} \leftarrow (1-w_q)\times q_{avg} + w_q \times q \tag{1}$$

$$p=\begin{cases} 0, & 0 \le q_{avg} \le Q_{min} \\ \dfrac{q_{avg}-Q_{min}}{Q_{max}-Q_{min}} P_{max}, & Q_{min} < q_{avg} \le Q_{max} \\ 1, & q_{avg} > Q_{max} \end{cases} \tag{2}$$

RED 采用等概率随机分布丢包，可以缓解因丢包同步而随之产生的 TCP 流的同步现象. RED 算法可保持相对较低的平均队长，有利于吸收突发流. 但是 RED 算法仍然存在两大问题：(1) 参数的设置问题. 即无法找到在任何负载变化条件下都合适的参数；(2) 带宽的公平性问题. 即在不同的时延、拥塞窗口、数据包的大小、目标速度以及不同的协议情况下难以权衡.

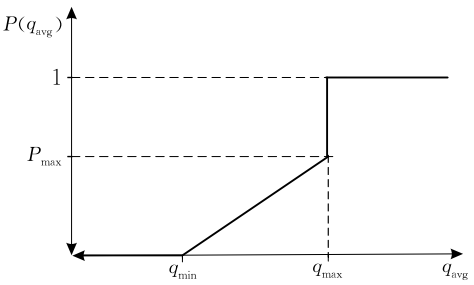


图 1　RED 丢包概率

3.2 SF-RED 算法思路

从第 2 节的讨论可以看出，现有的改善公平性的算法的主要思路是面向流的，其核心思路都是在拥塞发生时，通过惩罚具有某种特征的流来获得“公平”. 其中的部分算法需要维护所有的并发“流”信息，这对目前主干路由器来说还是非常困难的.

从报文协议的角度来看 RED 的公平性问题，由于 RED 平等地对待所有报文，当拥塞发生时，如果丢弃的是 TCP 报文，端系统能很快感知并自动放缓发送速度，使得拥塞能够很快缓解，但如 UDP 报文等非 TCP 报文不具备这样的能力. 当 UDP 比例过高时，TCP 让出的带宽还会被 UDP 占用. 本文基于这样的基本问题，提出了一个面向协议间公平性的 SF-RED 主动队列管理算法. 该算法工作机理基于 RED，但对 TCP 和非 TCP 流量采用区分丢包的控制方式. TCP 流量的丢包控制方式源于 TCP 模型推导的控制方程，而非 TCP 流量的丢包则通过与 TCP 之间的损失等价控制方式达到公平的平衡点.

3.3 区分丢包模型

3.3.1 TCP 流量的丢包计算

首先讨论本算法中针对 TCP 流量的丢包方式. RED 算法在 $Q_{\min}<q_{avg}\leq Q_{\max}$ 时丢包率取决于对应时刻计算的平均队列长度和 $P_{\max}$ 的值，丢包率的变化是线性的，这与 TCP 流量的行为模式有一定的差异. 本文中 TCP 流量的丢包算法是单独处理的，所以采用更符合 TCP 的流量行为的丢包模型. 具体而言，是选择了 TCP 协议中特有的参数“窗口”作为拥塞控制中计算 TCP 流量的丢包率的依据.

记由源端 S_i 到目的端 D_i 的 TCP 流窗口大小在 t 时刻为 $w_i(t)$，记路由器 R 上的在 t 时刻数据包的丢弃概率为 $p(t)$. 对窗口 $w_i(t)$ 中的每一个数据包，其被丢弃的概率都是 $p(t)$，当 $p(t)$ 很小时，对于窗口 $w_i(t)$ 来说，其中没有数据包需要重传的概率为 $\prod_{k=1}^{w_i(t)}(1-p(t))\approx 1-p(t)w_i(t)$. 因此，无数据包丢失的情况窗口变化速率为 $1/t_{\text{RTT}}$ 乘上丢包，即 $\frac{1-p(t)w_i(t)}{t_{\text{RTT}}}$. 窗口 $w_i(t)$ 中有数据包丢失的概率为 $1-\prod_{k=1}^{w_i(t)}(1-p(t))\approx p(t)w_i(t)$，此时的窗口减小一半，同样有数据包丢失情况下窗口的变化速率为 $-\frac{p(t)w_i^2(t)}{2t_{\text{RTT}}}$，所以总体的窗口变化速率表示为式(3).

$$\frac{\mathrm{d}w_i(t)}{\mathrm{d}t}=\frac{1-p(t)w_i(t)}{t_{\text{RTT}}}-\frac{p(t)w_i^2(t)}{2t_{\text{RTT}}} \tag{3}$$

当链路发生拥塞时，控制算法希望尽快阻止流量继续上升. 对于每个 TCP 流来说，则是需要保证窗口不会增大，使得单个 TCP 传输窗口不增加的条件为式(4).

$$\frac{\mathrm{d}w_i(t)}{\mathrm{d}t}\leq 0 \tag{4}$$

式(3)代入式(4)得到式(5).

$$\frac{1-p(t)w_i(t)}{t_{\text{RTT}}}-\frac{p(t)w_i^2(t)}{2t_{\text{RTT}}}\leq 0 \tag{5}$$

由于 TCP 中往返时延均视为正值，因此通过式(5)可以获得在窗口不增加情况下 $p(t)$ 与 $w_i(t)$ 间的关系，见式(6).

$$p(t)\geq\frac{2}{(w_i^2(t)+2w_i(t))} \tag{6}$$

从拥塞控制的角度看整体 TCP 流量，TCP 每个流可能具有不同的窗口大小，假设存在一个理想的窗口大小为 $w^*(t)$，使得 t 时间整体 TCP 流量在丢包率 $p^t(t)$ 下增长率为 0，它们之间关系应该满足式(7).

$$p^t(t)=\frac{2}{(w^{*2}(t)+2w^*(t))} \tag{7}$$

SF-RED 算法中使用该式作为 TCP 流量的丢包率计算模型. 该计算模型在丢包率较大时可能会出现较大的误差，因此后面在使用该计算模型时，窗口参数也是估算值，其大小浮动变化用于弥补可能存在的误差.

3.3.2 其他流量的丢包计算

如上所述，丢包对 TCP 流量的影响超过对以 UDP 为主的非 TCP 流量的影响，因此在设计非 TCP 流量的丢包模型时，将主要基于这个影响因素进行，即应尽量将 2 者的损失比例控制在同一个数量级上.

根据 AIMD 原则，在丢包率为 $p^t(t)$ 下，整体 TCP 窗口为 $w^*(t)$ 下，TCP 流量的丢包数期望值为 $drop^t(t)=p^t(t)\times N_{\text{tcp}}$，其中使用 N_{tcp} 表示 TCP 的整体流量，而这些被丢失的数据包会对以 $w^*(t)$ 为平衡窗口大小的整体的 TCP 流量产生预期的流量损失大小可以估计为 $loss^t=drop^t(t)w^*(t)/2$，因此 TCP 流量的损失比例为 $loss^t/N_{\text{tcp}}=p^t(t)w^*/2$.

若将同一时刻非 TCP 的损失比例控制在同一水平上，则有式(8)成立：

$$p^o(t)=cp^t(t)w^*(t)/2 \tag{8}$$

其中 c 为常量，若要使得非 TCP 流量因为其本身不

能对丢包进行反馈而增加其惩罚程度，即满足条件 $p^o(t)\geq p^t(t)$，则常量 c 需要满足 $c\geq 2/w^*(t)$. 因窗口的大小满足 $w^*(t)\geq 1$，因此推荐 c 的取值范围为 $[2,3]$.

将式(6)代入式(8)，因为窗口 w^* 为正，得到式(9).

$$p^o(t)=c\frac{w^*(t)}{w^{*2}(t)+2w^*(t)}=\frac{c}{w^*(t)+2} \quad (9)$$

SF-RED 算法中使用该式作为非 TCP 流量的丢包率计算模型.

图 2 给出式(9)中 c 取值为 1、2、3 时，2 种类型流量在不同窗口大小下的丢包率变化情况.

根据式(6)，当 $w^*>1$ 时 TCP 的丢包率是 w^* 的单调递减函数，当 $w^*=1$ 时，其最大值为 2/3. 非 TCP 的丢包率则随着窗口增加单调递减，当 $w^*=1$ 时，其最大值为 $c/3$.

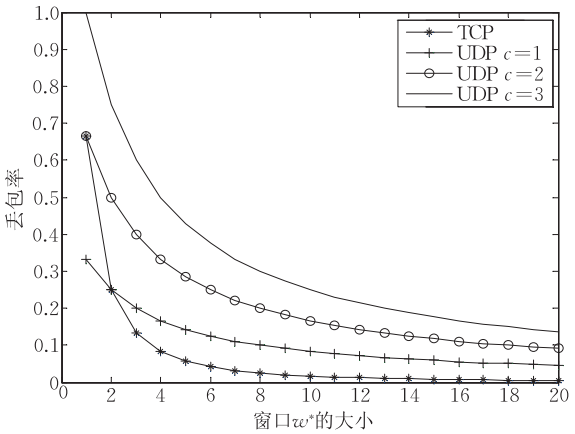


图 2　丢包率与窗口的关系

3.4　算法描述

SF-RED 算法使用 4 个含义与 RED 算法完全相同的参数，分别是队列长度最小门限 $Q_{\min}$、最大门限值 $Q_{\max}$、$P_{\max}$ 和 q_{avg}.

算法的核心思路是当 $q_{\text{avg}}>Q_{\min}$ 时，定义 SF-RED 算法定义进入拥塞状态，用布尔变量 $cong$ 进行标记. 在拥塞状态下，按 3.3 中的讨论，分 2 种方式对 TCP 流量与非 TCP 流量进行区分丢包策略以到达 2 者稳定共存的目的.

算法中使用的其他参数：

(1) 当前队列长度 q：在报文的到达和离开时维护；

(2) 平均 TCP 队列长度 q^t_{avg}：维护方法同 q_{avg} (EWMA 方法)；

(3) 当前 TCP 队列长度 q^t，队列中 TCP 的队列长度：维护方式与 q 相同；

(4) TCP 报文丢包数 $drop^t$ 和其他报文丢包数 $drop^o$：进入本次拥塞后 2 种类型报文的丢包记录；

(5) TCP 的丢包概率 p^t 和非 TCP 丢包概率 p^o. 根据窗口参数 $w^*(t)$ 和式(7)和式(9)实时计算出的丢包率；

(6) 窗口参数 $w^*(t)$，用固定增加值 a 和固定减少值 b 维护(a，b 初始化值均为 1)，用于计算 TCP 的报文丢包率 p^t 值.

SF-RED 算法的伪代码描述如下：

```
初始化算法参数 Qmin、Qmax、Pmax、a、b;
FOR 每一个到达报文 P{
更新队列状态变量 q,qavg,q^t,q^t_avg;
IF q>缓冲最大值,进入强制丢包流程;EXIT;
IF qavg<Qmin,cong=0 非拥塞状态,报文正常加入队列;EXIT;
IF cong=0 //进入拥塞状态
 初始化拥塞算法参数 w*,drop^t,drop^o;
 本次不丢包,cong=1;
 ELSE{ //随机抽取队列中的一个对比报文 P*
   IF P'fid=P*'fid
     丢弃 P 和 P*
   ELSE //与到达报文 P 进行区分类型丢包
   IF P'type=P*'type
   //type 仅分 2 类,TCP 和非 TCP
   根据类型用式(7)或式(9)随机丢弃 P;
   ELSE
   根据类型用式(7)或式(9)随机丢弃 P*;
 IF 发生丢包
  更新丢包参数 drop^t,drop^o;
}
 FOR 每离开一个报文{
 更新队列队长 q,q';}
```

算法中各参数的初值. 最小阈值 $Q_{\min}$、最大阈值 $Q_{\max}$ 需要根据实际情况自行设置，$P_{\max}$ 默认值采用同 RED 的默认设置为 0.1.

算法中参数 w^* 的初始化使用式(7)中丢包率与窗口的关系进行变换获得. 由于窗口大小为正值，所以 w^* 使用式(10)进行初始化. 其中的 p_0 与 RED 的初始丢包率计算方式相同，具体表达式为式(2). 参数 w^* 的维护参考当前队列中两种报文的比例与当前两种报文的丢包数比例. 设当前队列中的 TCP 报文比例为 r^t_{queue}，TCP 报文丢包比例为 r^t_{drop}，若 $r^t_{queue}>r^t_{drop}cw^*/2$ 成立，则增大 w^*；否则减小 w^*.

$$w^*=\sqrt{1+2/p_0}-1 \quad (10)$$

算法中使用了从队列中选择报文，在其与到达报文类型不一致时丢弃的方法. 这一基本思路源于种群竞争 Lotka-Volterra(L-V)模型，有关的分析将

在下一小节给出.

4 基于 Lotka-Volterra 模型的算法收敛性分析

本节使用 L-V 模型对 TCP 流量和非 TCP 流量的竞争关系进行刻画.带宽作为有限资源可以认为是传输层协议 TCP 与非 TCP 的共享资源.因此,可以将 TCP 与非 TCP 流量作为对同种资源进行竞争的两个种群来看待,符合面向特定资源的种群竞争 L-V 模型的条件.

4.1 Lotka-Volterra 模型

Lotka-Volterra 模型(Lotka-Volterra 种间竞争模型)是 20 世纪 40 年代,Lotka(1925)和 Volterra(1926)提出的种间竞争方程对现代生态学理论,它奠定了种间竞争关系的理论基础.Lotka-Volterra 模型是对 Logistic 模型的延伸[27].适用于描述在两种物种争夺有限的同一种食物来源和生活空间时而进行的生存竞争的情况.

模型中设定甲乙两个种群,当它们独自在一个自然环境中生存时,数量的演变遵从 logistic 规律.记 $x_1(t)$、$x_2(t)$分别代表两个种群的数量.r_1、r_2表示它们的固有增长率,N_1、N_2是它们的最大容量.于是对于种群甲有 $\dot{x}_1(t)=r_1x_1(t)\left(1-\frac{x_1(t)}{N_1}\right)$其中因子$\left(1-\frac{x_1(t)}{N_1}\right)$反映由于甲对有限资源的消耗导致的对其增长的阻滞作用,$\frac{x_1(t)}{N_1}$为相对于 N_1而言单位数量的甲消耗的食物量(设食物总量为 1).

当两个种群在同一自然环境中生存时,考察乙消耗同一种有限资源时对甲的增长产生的影响,可以合理地在因子$\left(1-\frac{x_1(t)}{N_1}\right)$中再减去一项,该项与种群乙的数量 x_2(相对于 N_2而言)成正比,得到种群甲的增长方程

$$\dot{x}_1(t)=r_1x_1(t)\left(1-\frac{x_1(t)}{N_1}-\delta_1\frac{x_2}{N_2}\right) \quad (11)$$

其中 δ_1的意义是,单位数量乙(相对于 N_2而言)消耗的食物量为单位数量甲(相对 N_1)消耗的食物量的 δ_1倍.

类似地,甲的存在也影响了乙的增长,种群乙的增长方程是

$$\dot{x}_2(t)=r_2x_2(t)\left(1-\delta_2\frac{x_1(t)}{N_1}-\frac{x_2(t)}{N_2}\right) \quad (12)$$

根据式(11)和(12),对于两个种群相互竞争的结局进行分析,通过计算可求出 4 个平衡点和在每个平衡点下的稳定性条件见表 1[27].

表 1 种群竞争模型平衡点及稳定性表说明

平衡点	稳定条件
$P_1(N_1,0)$	$\delta_1<1$, $\delta_2>1$
$P_2(0,N_2)$	$\delta_1>1$, $\delta_2<1$
$P_3\left(\frac{N_1(1-\delta_1)}{1-\delta_1\delta_2},\frac{N_2(1-\delta_2)}{1-\delta_1\delta_2}\right)$	$\delta_1<1$, $\delta_2<1$
$P_4(0,0)$	不稳定

从结果看,一共有 3 个稳定的平衡点.到达平衡点 P_1和 P_2时会导致其中一个的物种消亡,到达平衡点 P_3表示两物种保持二者在一定比例上共存.

4.2 SF-RED 算法的收敛性分析

本节借用 Lotka-Volterra 模型分析使用 SF-RED 算法条件下网络中 TCP 流量与非 TCP 流量的竞争关系.

将 TCP 流量看作物种甲,非 TCP 流量看物种乙.TCP 与非 TCP 流量在网络环境中对带宽的竞争相互影响,稳定共存的局面,满足 Lotka-Volterra 模型条件,可以用式(11)和(12)描述.其中 $x_1(t)$、$x_2(t)$分别代表 TCP 流量和非 TCP 流量.系数 r_1、r_2表示 TCP 和非 TCP 流量的固有增长率,N_1、N_2是网络中间节点上 TCP 和非 TCP 流量各自的最大容量.于是在网络节点上流量增加对于单纯的全 TCP 和非 TCP 流量都假设符合式(11)的形式,其中因子$\left(1-\frac{x_i(t)}{N_i}\right)(i=1,2)$反映由于可供网络资源有限导致的对流量增长的阻滞作用,$\frac{x_i(t)}{N_i}(i=1,2)$为相对于 N_i而言单位数量的某方消耗的带宽资源.δ_1表示非 TCP 对 TCP 的抑制作用,δ_2表示 TCP 对非 TCP 的抑制作用.

根据表 1,平衡点 P_1表示网络中 TCP 流量占据了所有带宽.平衡点 P_2表示非 TCP 占据了所有的带宽.平衡点 P_3表示 TCP 与非 TCP 二者共存,共同享有带宽,二者处于动态平衡状态.

如果平衡点 P_3存在,即使网络发生拥塞(即网络资源消耗到达极限),TCP 和非 TCP 流量可以在一定比例上稳定共存,所以可以认为这个平衡点是我们在网络资源紧缺时为保证传输协议间公平竞争需要实现的目标.

在网络中流量之间的竞争能力强弱体现在 Lotka-Volterra 模型中竞争能力系数 δ_1和 δ_2,其值

是与协议自身的算法相关的，网络中间节点对流量的控制无法改变其本质属性. SF-RED 通过借助基于类型的丢包方式来修正该模型下平衡点.

由 3.4 节的算法描述可知，SF-RED 在拥塞状态下的丢包策略是随机抽取当前队列中的一个数据包，如果类型与当前达到的数据包相同则对到达数据包进行丢弃判定；否则，丢弃抽取的数据包. 这样的丢包策略使得一个新到的 TCP 数据包的丢弃不仅取决于当前队列的长度也与当前队列中的非 TCP 数据包数量有关. 同样，非 TCP 数据包的丢弃也不仅取决于当前队列长度也与当前队列中的 TCP 数据包数量相关. 这个判定方式的原理是强行干预生物种群模型中 2 种物种的竞争模型，使得 2 种数据包之间的竞争关系式(12)发生了改变. 在 SF-RED 算法下，新到的 TCP 的丢包率会随着队列中非 TCP 包的数量的增加而减少，同样新到的非 TCP 的丢包率也会随着队列中 TCP 包的数量的增加而减少. 因此，SF-RED 下两种类型的数据包在争抢带宽上满足微分方程组(15).

$$\begin{cases}\dot{x}_1(t)=r_1x_1(t)\left(1-\dfrac{x_1(t)}{N_1}+\delta_1\dfrac{x_2(t)}{N_2}\right)\\ \dot{x}_2(t)=r_2x_2(t)\left(1+\delta_2\dfrac{x_1(t)}{N_1}-\dfrac{x_2(t)}{N_2}\right)\end{cases}\tag{15}$$

对上述方程组进行分析，发现无法求出 $x_1(t)$ 和 $x_2(t)$ 的解析表达式，但是可以根据方程组得出在 $t\to\infty$ 时的平衡点.

假设从初始条件出发，存在 $x_1=x_1^0$，$x_2=x_2^0$ 满足 $\lim\limits_{t\to\infty}x_1(t)=x_1^0$，$\lim\limits_{t\to\infty}x_2(t)=x_2^0$. 这样可以得到新的 4 个平衡点 $P_1^*(N_1,0)$，$P_2^*(0,N_2)$，$P_3^*\left(\dfrac{N_1(1+\delta_1)}{1-\delta_1\delta_2},\dfrac{N_2(1+\delta_2)}{1-\delta_1\delta_2}\right)$，$P_4^*(0,0)$.

因为仅当平衡点满足 $(x_1,x_2\geq0)$ 才有实际意义，微分方程组(15)令其导数等于 0，使用方程组(16)表示，利用将方程组(16)在平衡点进行 Taylor 展开，取一次项，得出近似线性方程后，系数矩阵为 $\boldsymbol{A}$，形式见式(17).

$$\begin{cases}f(x_1,x_2)\equiv r_1x_1\left(1-\dfrac{x_1}{N_1}+\delta_1\dfrac{x_2}{N_2}\right)=0\\ g(x_1,x_2)\equiv r_2x_2\left(1+\delta_2\dfrac{x_1}{N_1}-\dfrac{x_2}{N_2}\right)=0\end{cases}\tag{16}$$

$$\begin{aligned}\boldsymbol{A}&=\begin{bmatrix}f_{x_1}&f_{x_2}\\ g_{x_1}&g_{x_2}\end{bmatrix}\\&=\begin{bmatrix}r_1\left(1-\dfrac{2x_1}{N_1}+\dfrac{\delta_1x_2}{N_2}\right)&\dfrac{r_1\delta_1x_1}{N_2}\\ \dfrac{r_2\delta_2x_2}{N_1}&r_2\left(1+\dfrac{\delta_2x_1}{N_1}-\dfrac{2x_2}{N_2}\right)\end{bmatrix}\end{aligned}\tag{17}$$

再根据 $\det(\boldsymbol{A}-\lambda\boldsymbol{I})=0$ 求特征方程根，得到方程组(18)

$$\begin{cases}\lambda^2+p\lambda+q=0\\ p=-(f_{x_1}+g_{x_2})\big|_{P_i^*},\ i=1,2,3,4\\ q=\det\boldsymbol{A}\big|_{P_i^*},\ i=1,2,3,4\end{cases}\tag{18}$$

根据对方程组的特征根分析，求出的平衡点和稳定性条件见表 2(p、q 参照式(18)).

表 2　竞争模型平衡点

平衡点	p	q	稳定条件
$P_1^*(N_1,0)$	$r_1-r_2(1+\delta_2)$	$-r_1r_2(1+\delta_2)$	不稳定
$P_2^*(0,N_2)$	$-r_1(1+\delta_1)+r_2$	$-r_1r_2(1+\delta_1)$	不稳定
$P_3^*\left(\dfrac{N_1(1+\delta_1)}{1-\delta_1\delta_2},\dfrac{N_2(1+\delta_2)}{1-\delta_1\delta_2}\right)$	$\dfrac{r_1(1+\delta_1)+r_2(1+\delta_2)}{1-\delta_1\delta_2}$	$\dfrac{r_1r_2(1+\delta_1)(1+\delta_2)}{1-\delta_1\delta_2}$	$\delta_1>-1,\delta_2>-1$
$P_4^*(0,0)$	$-(r_1+r_2)$	r_1r_2	不稳定

对该方程组进行分析可知，在满足 $\delta_1>0,\delta_2>0$ 情况下，平衡点 P_3^* 的条件成立. 此时，另外 3 个之前的平衡点都为不稳定点.

上述分析表明 SF-RED 算法中的基于类型的丢包方式(15)，在 Lotka-Volterra 模型下可以保证 TCP 与非 TCP 在队列管理中能够按照其自身的属性达到一个二者共存的平衡点 P_3^*.

4.3　SF-RED 算法的复杂度分析

本节对照 RED 与 CHOKe 来分析该算法的复杂度. RED 的时间复杂度和空间复杂度均为 $O(1)$，这也是 RED 算法的优势. 尽管 CHOKe 需要随机选出一个候选报文进行对比，其时间复杂度和空间复杂度也同样是 $O(1)$.

上文提出的 SF-RED 算法，每到达一个报文进行的处理与 RED 和 CHOKe 基本相同，其区别为在确定丢包率的算法上存有一些不同的处理. 因此算法的时间复杂度仍旧是 $O(1)$. 空间复杂度上增加几个需要的变量，变量数目不随着规模的增加而增加，因此其复杂度仍旧为 $O(1)$. 整体上看，SF-RED 算法保持了 RED 与 CHOKe 算法的优点.

另外需要指出的是，在收敛性方面，RED 并不能阻止队列不断增长，如果用控制理论建模，它不能收敛到较短的目标队列长度. SF-RED 通过引入区分丢包的方式对不同的流量进行不同方式的处理. 其中 TCP 的丢包方式根据 TCP 的吞吐量模型进行推导，丢包率推导条件就是队列长度维持在增长率为 0 的界限（即队列长度不再增加，式(4)）. UDP 的丢包方式根据当前的算法中参数进行动态的计算（与 TCP 丢包率有关（式(8)）. SF-RED 的控制模型，结合了 TCP 的吞吐特点和 UDP 非响应流的特点，增加了 CHOKe 的惩罚机制，通过计算合理的丢包率可以很好的抑制队列长度的增长速度.

5 基于 RTT 的综合 TCP 传输性能测度

为了验证上述 SF-RED 算法的有效性，使用 NS2 仿真环境对不同的非 TCP 流量比例下的网络状况进行测试.

5.1 仿真环境

实验在 NS2 中使用如图 3 所示的拓扑结构. 仿真使用单瓶颈的哑铃拓扑. 有 N_1 个 TCP 链接和 N_2 个 UDP 链接与路由器分别相连接，与路由器之间的链路默认使用丢尾（drop-tail）方式，链路上限为 2 Mbps，延迟为 10 ms. 路由器 R1 和 R2 之间链路默认带宽限制 30 Mbps，延迟 20 ms，队长为 20 个数据包.

实验在较高的网络负载下进行，在不同的非 TCP 流量比重下测试每个算法的表现. R1 到 R2 的链路上使用 RED，CHOKe 和 SF-RED 算法.

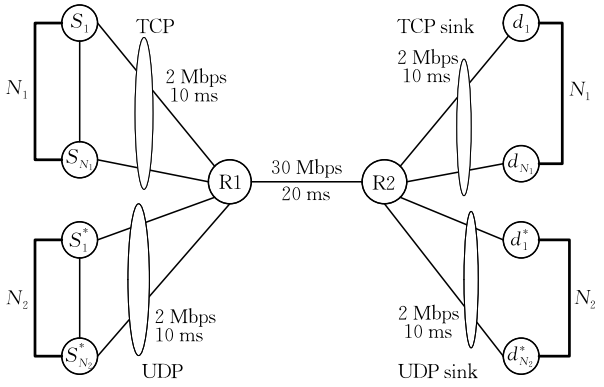
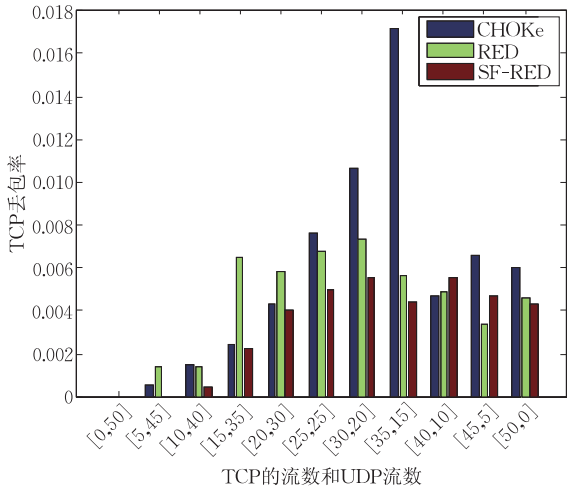


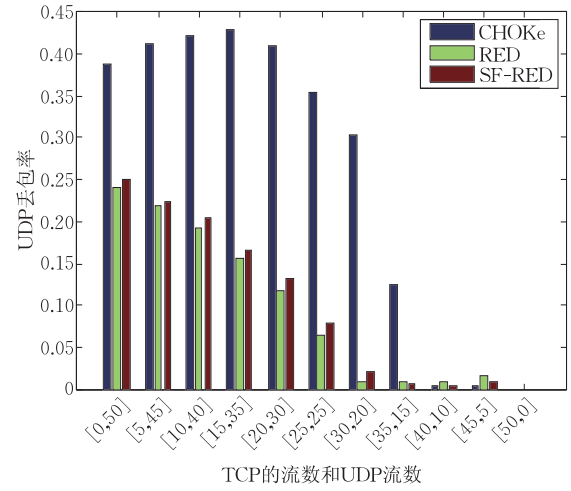
图 3 仿真拓扑图

5.2 TCP 流与 UDP 流共存的仿真结果

仿真通过改变 TCP 和 UDP 流的数量，即通过不同的 N_1 和 N_2 的设置，测试算法的有效性. SF-RED 中采用参数 a,b,c 都固定设置为 1. 图 4 和图 5 表示在不同的流数比 $[N_1,N_2]$ 组合情况下，3 种算法在相同的网络配置下，TCP 报文丢包率和 UDP 报文丢包率对比.



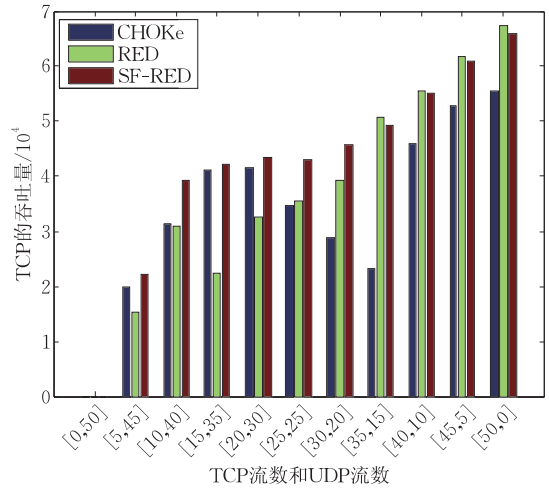
(a) TCP丢包率比较



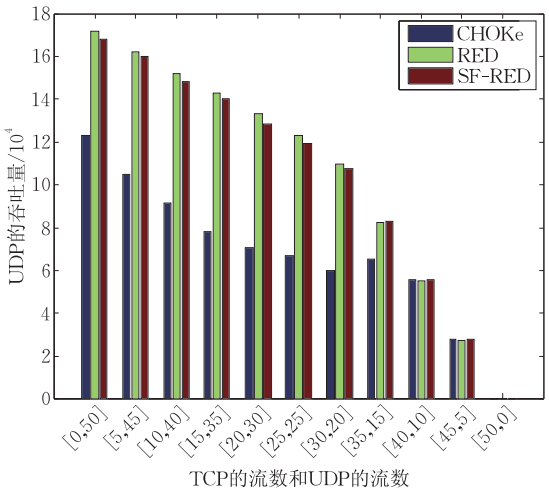
(b) UDP丢包率比较

图 4 丢包率对比

由于 CHOKe 对于恶意流的鉴别不区分流量种类，因此对于 TCP 和 UDP 流的丢包率在 3 种算法中都偏高（图 4），CHOKe 的这一特点在流较少的情况下确实能防止少数流量恶意流量独占大量带宽. 根据文献[28]，对 CHOKe 的建模分析表明，CHOKe 能保证最大的单流占用带宽不超过 26%. 然而，当流的数量增加到万级，CHOKe 的优势反而会限制 TCP 为代表的传输协议. 因为 TCP 数据窗口增大后，相同吞吐量的 TCP 与恒速的其他应用在 CHOKe 下会获得更大的丢包概率. 因此，CHOKe 并不适合在骨干路由器上的配置. 从图 5 可知，



(a) TCP吞吐量



(b) UDP吞吐量

图 5　吞吐量对比

CHOKe 的实际吞吐率是 3 种算法中最低的.

比较 SF-RED 与 RED 的丢包率，SF-RED 上 UDP 的丢包率略大于 RED，表明 SF-RED 算法对仿真中 UDP 的惩罚力度稍强于 RED. 从 TCP 的丢包率看，在高 UDP 的环境下(即 UDP 超过 50%)，SF-RED 对于 TCP 的丢包率明显小于 RED(见图 4(a)).

从吞吐量上看，SF-RED 与 RED 相比较，在混合流量下，SF-RED 的 TCP 吞吐率较优. UDP 的吞吐量与 RED 基本相等.

综上所述，实验中 SF-RED 在非 TCP 流量较高的情况下可以较好的保护 TCP 流量，并且通过窗口参数 w^* 的调节和利用 Lotka-Volterra 模型保证了混合流量的并存性和收敛性，在不同的情况下进行区别的丢包方式保护了弹性流的合理带宽，可用带宽的充分利用使得网络具有良好的吞吐率.

6　总　结

本文提出了一个基于 RED 的差异型丢包队列管理算法 SF-RED. 通过 TCP 的吞吐量模型引入一个窗口参数，对流量的拥塞控制进行动态调节；控制方式上采用了不同类型报文的差异性丢包方式. 从理论上证明了 SF-RED 算法控制方式存在异种流量共存的稳定平衡点. 算法的时间和空间复杂度较低. 仿真实验表明较 RED 和 CHOKe 相比，SF-RED 在高 UDP 流量下可以合理保护 TCP 流量，并对恶意的 UDP 流量产生抑制，并且同时能有效利用带宽，具有良好的吞吐率.

参　考　文　献

[1] Hollot C V, Misra V, Towsley D, Gong W. On designing improved controllers for AQM routers supporting TCP flows//Proceedings of the IEEE International Conference on Computer Communications. Anchorage, USA, 2001: 1726-1734

[2] Kunniyur S, Srikant R. Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management. ACM Computer Communication Review, 2001, 31(4): 123-134

[3] Low S H, Lapsley D E. Optimization flow control: Basic algorithm and convergence. IEEE/ACM Transactions on Networking, 1999, 7(6): 861-875

[4] Floyd S, Jacson V. Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on Networking, 1993, 1(4): 397-413

[5] Athuraliya S, Li V H, Low S H, Yin Q. REM: Active queue management. IEEE Network, 2001, 15(3): 48-53

[6] Zeng Zhen-Ping, Wang Bing-Wen. Survey on fairness research of Internet congestion control. Computer Science, 2008, 35(1): 19-23(in Chinese)
(曾振平，汪秉文. 因特网拥塞控制的公平性研究综述. 计算机科学，2008，35(1)：19-23)

[7] Lin Dong, Morris R. Dynamics of random early detection// Proceedings of the ACM Special Interest Group on Data Communication Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. Cannes, France, 1997: 127-137

[8] Pan R, Prabhakar B, Psounis K. CHOKe: A stateless active queue management scheme for approximating fair bandwidth allocation//Proceedings of the IEEE International Conference on Computer Communications. Tel Aviv, Israel, 2000, 2: 942-951

[9] Chatranon G, Labrador M A, Banerjee S. BLACK: Detection and preferential dropping of high bandwidth unresponsive flows//Proceedings of the International Conference on Communications. 2003, 1: 664-668

[10] Zhang Yi-Bin, Zhang Zhi-Bin, Zhao Yong, Guo Li. Comparative analysis on TCP and UDP network traffic. Application Research of Computers, 2010, 27(6): 2192-2197(in Chinese)
(张艺濒, 张志斌, 赵咏, 郭莉. TCP 与 UDP 网络流量对比分析研究. 计算机应用研究, 2010, 27(6): 2192-2197)

[11] Lee D, Carpenter B E, Brownlee N. Media streaming observations: Trends in UDP to TCP ratio. International Journal on Advances in Systems and Measurements, 2010, 3(3&4): 147-162

[12] Wang Gang, Ding Wei, Dong Shi. Classification of UDP traffic from P2P applications//Proceedings of the 17th IEEE International Conference on Network. Singapore, 2011: 252-257

[13] Feng Wu-Chang, Kandlur D D, Saha D, Shin K G. A self-configuring RED gateway//Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies. New York, USA, 1999, 3: 1320-1328

[14] Mahajan R, Floyd S, Wetherall D. Controlling high-bandwidth flows at the congested router. AT&T Center for Internet Research at ICSI (ACIRI): Technical Report TR-01-001, 2001

[15] Tang De-You, Luo Jia-Wei, Zhang Da-Fang, et al. Active queue management improving the stability and fairness. Journal of Computer Research and Development, 2005, 42(7): 1136-1142(in Chinese)
(汤德佑, 骆嘉伟, 张大方等. 一种提高稳定性和公平性的主动队列管理机制. 计算机研究与发展, 2005, 42(7): 1136-1142)

[16] Feng W, Kandlur D, Saha D, et al. Stochastic fair Blue: A queue management algorithm for enforcing fairness//Proceedings of the IEEE International Conference on Computer Communications. Anchorage, USA, 2001: 1520-1529

[17] Jiang Y, Hamdi M, Liu J. Self adjustable CHOKe: An active queue management algorithm for congestion control and fair bandwidth allocation//Proceedings of the IEEE Symposium on Computers and Communications. Kemer-Antalya, Turkey, 2003: 1018-1025

[18] Yang Ru, Zhai Jian-Hong. The AME-CHOKe protocol based on the estimate of the flow number. Journal of Heilongjiang Institute of Technology, 2011, 25(3): 1-5(in Chinese)
(杨茹, 翟健宏. 基于流数目估计的 AME-CHOKe 协议. 黑龙江工程学院学报(自然科学版), 2011, 25(3): 1-5)

[19] Eshete A, Jiang Yuming. Protection from unresponsive flows with geometric CHOKe//Proceedings of the 2013 IEEE Symposium on Computers and Communications. 2012: 339-344

[20] Le L, Aikat J, Jeffay K, et al. Differential congestion notification: Taming the elephants//Proceedings of the 12th IEEE International Conference on Network Protocols. Berlin, Germany, 2004: 118-128

[21] Ho C Y, Chan Y C, Chen Y C. A TCP-friendly stateless AQM scheme for fair bandwidth allocation//Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services. Washington, USA, 2005: 14-19

[22] Padhye J, Firoiu V, Towsley D, et al. Modeling TCP throughput: A simple model and its empirical validation. ACM SIGCOMM Computer Communication Review, 1998, 28(4): 303-314

[23] Misra V, Gong W B, Towsley D. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED//Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM) Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. Stockholm, Sweden, 2000: 151-160

[24] Low S H. A duality model of TCP and queue management algorithms. IEEE/ACM Transactions on Networking, 2003, 11(4): 525-536

[25] Zhu Hai-Ting, Ding Wei, Miao Li-Hua, Gong Jian. Effect of UDP traffic on TCP's round-trip delay. Journal on Communications, 2013, 34(1): 19-29(in Chinese)
(朱海婷, 丁伟, 缪丽华, 龚俭. UDP 流量对 TCP 往返延迟的影响. 通信学报, 2013, 34(1): 19-29)

[26] Song Li-Hua, Wang Hai-Tao, Cao Hai-Bing. Performance service based transport layer congestion control resolution for high-speed networks. Journal of PLA University of Science and Technology (Natural Science Edition), 2012, 13(3): 259-265(in Chinese)
(宋丽华, 王海涛, 曹海兵. 基于性能服务的高速网络运输层拥塞控制解决方案. 解放军理工大学学报(自然科学版), 2012, 13(3): 259-265)

[27] Jiang Qi-Yuan. Mathematical Model. 2nd Edition. Beijing: Higher Education Press, 1991(in Chinese)
(姜启源. 数学模型. 第 2 版. 北京: 高等教育出版社, 1991)

[28] Wang Jiantao, Tang Ao, Low Steven H. Maximum and asymptotic UDP throughput under CHOKe. ACM SIGMETRICS Performance Evaluation Review, 2003, 31(1): 82-90

ZHU Hai-Ting, born in 1983, Ph. D., lecturer. Her main research interests include network measurement, network security and network management.

DING Wei, born in 1962, Ph. D., professor. Her main research interests include network measurement and network behavior.

Background

Active queue management is essential to network facilities. Nowadays, UDP traffic has become a significant part in the network. UDP traffic could seriously affect the performance of TCP traffic when congestion happens. New AQM methods are needed to adapt the high UDP ratio traffic. However, almost all present AQM methods treat UDP and TCP traffic in the same way and it is hard to maintain fairness in this way. Our work made use the throughput model of TCP traffic to generate differential drop probabilities for TCP and UDP traffic. It is proved that our method could lead to a co-existing state for the mixed traffic. Simulation results showed this algorithm achieved good performance under high UDP ratio traffic.

This research is supported by the National Basic Research Program (973 Program) of China (2009CB320505); the National Key Technology Research and Develop Program of China (2008BAH37B04).