文章编号:1671-5896(2010)01-0077-07

基于自适应参数设置的 AIMD 算法

张丽娟[a,b],杨晓萍[a,b],陈 虹[a,b],张振宇[a,b]
(吉林大学 a. 汽车动态模拟国家重点实验室; b. 通信工程学院,长春 130025)

摘要: 为提高 TCP (Transfer Control Protocol) 流的平均发送数率,减少发送数率的波动,提高网络性能,提出了一种基于自适应参数设置的 AIMD (Additive Increase Multiplicative Decrease) 算法 (A-AIMD 算法)。仿真结果表明,在网络处于稳态运行时,使用 A-AIMD 算法能提高 TCP 流的平均发送速率,减小发送速率的波动性;在网络中有可利用的链路资源时,使用 A-AIMD 算法能快速地对可用资源进行最大占用,同时与 Reno 算法保持一定的 TCP 友好性。

关键词: 通信技术; TCP 拥塞控制; 加增乘减; TCP 友好
中图分类号: TP393 **文献标识码:** A

Enhanced AIMD Mechanism Based on Adaptive Parameter Settings

ZHANG Li-juan[a,b], YANG Xiao-ping[a,b], CHEN Hong[a,b], ZHANG Zhen-yu[a,b]
(a. State Key Laboratory of Automobile Dynamic Simulation;
b. College of Communication Engineering, Jilin University, Changchun 130025, China)

Abstract: To improve the number of TCP (Transfer Control Protocol) flows, the average sending rate, to reduce the volatility of the rate of sending a few to improve network performance, adaptive parameter settings is proposed based on AIMD (Additive Increase Multiplicative Decrease) algorithm (A-AIMD algorithm). Simulation results show that the network is in steady-state operation, the use of A-AIMD algorithm can improve the TCP stream, the average sending rate, reducing the sending rate volatility. The network has the link available resources the use of A-AIMD algorithm can quickly occupied most of the resources available at the same time with the Reno algorithm to maintain a certain degree of TCP-friendly.

Key words: communication technology; transfer control protocol (TCP) congestion control; additive increase multiplicative decrease (AIMD); TCP-friendly

引 言

进入 20 世纪 90 年代以来,Internet 成为一个重要的和无处不在的基础设施,以网络协议 IP (Internet Protocol) 为基础的 Internet 呈爆炸式增长,使 Internet 的流量急剧增加,由此引发的网络拥塞已经成为制约网络发展和应用的关键问题[1]。网络拥塞会引起数据传输延迟和延迟抖动等服务质量 QoS (Quality of Service) 性能指标下降,并严重影响带宽、缓存和吞吐量等网络资源的利用率。因此,有效地解决拥塞问题对于提高网络性能具有重要意义。如何更好地预防和控制网络拥塞一直是近年来国内外网络研究领域的热点问题[2]。

收稿日期:2009-10-10
基金项目:国家杰出青年科学基金资助项目(60725311)
作者简介:张丽娟(1986—),女,河南西平人,吉林大学硕士研究生,主要从事网络拥塞控制研究,(Tel)86-13104308799 (E-mail) zhangjuan200320@163.com; 杨晓萍(1963—),女,黑龙江鹤岗人,吉林大学副教授,博士,主要从事网络拥塞控制研究,(Tel)86-13604441452 (E-mail) yxp@jlu.edu.cn;陈虹(1963—),女,浙江桐乡人,吉林大学教授,博士生导师,主要从事网络优化控制和预测控制研究,(Tel)86-431-85094831 (E-mail) chenh@jlu.edu.cn

1988年Jacobson[5]指出了传输控制协议TCP（Transfer Control Protocol）在网络拥塞控制方面的不足，提出了“慢启动”、“拥塞避免”算法。1990年出现的TCP Reno版本增加了“快速重传”、“快速恢复”算法。TCP的拥塞控制就是通过这4个核心算法控制一些重要参数而实现的[6]。TCP拥塞控制机制在Internet中发挥了行之有效的作用，但随着人们对网络QoS要求的不断提高，现有的TCP拥塞控制机制已经无法满足人们的需要。笔者提出了一种自适应参数设置的和式增加积式减少AIMD（Additive Increase Multiplicative Decrease）算法（A-AIMD算法）。仿真结果证明了该算法对于提高网络性能的有效性。

1 AIMD算法

Jacobson提出的TCP拥塞控制算法使用AIMD的窗口调节机制。算法用数学公式表示为

$$\text{I: } W_{t+R} \leftarrow W_t + \alpha,\ \alpha > 0; \qquad \text{D: } W_{t+R} \leftarrow \beta \times W_t,\ 0 < \beta < 1 \tag{1}$$

其中I表示在一个回路响应时间RTT（Round-Trip Time）内接收到ACK（ACKnowledge Character）确认包而引起的窗口（速率）增加算法，W_t是t时刻窗口的大小，R代表回路响应时间RTT，α是加性增加因子；D表示遇到拥塞后窗口（速率）减小算法，β是乘性减少因子。Chiu等[5]研究了AIMD算法的稳定性和公平性，指出了α，β应满足的条件，如式(1)所示。在传统TCP中$\alpha=1$，$\beta=0.5$。传统AIMD算法能较快速地使用网络中的有用资源，但当网络拥塞程度加大时，又能较为急剧地降低其数据发送速率，减轻网络拥塞，致使其数据发送速率波动较大、平均发送速率不是很高、对互联网的某些应用支持的不好，这已经成为了其主要缺陷。

针对该缺陷，许多研究人员进行了相关的改进研究。如Robert等[5]提出了一种通信网络中自适应的拥塞控制协议，把Frobenius-Perron理论和网络状态相结合控制网络拥塞；Alex等[7]提出一种自适应AIMD拥塞控制机制，根据接收到的反馈调整参数，但它并不具有TCP友好性；Yang等[8]提出的GAIMD机制具有较高的TCP友好性；Hayder等[9]提出的一种新AIMD拥塞控制算法兼具效率性和公平性，但推导模型较理想化；徐昌彪等[10]提出基于双重AIMD的TCP拥塞控制，该算法中已经含有了自适应设置AIMD参数的思想，它根据发送速率把网络分为两种不同的拥塞状态，分别使用不同参数的AIMD算法等。但在具体实现时，存在约束条件不易判决、各个网络状态分界不明显、并没有实现参数完全的自适应设置等问题，从而导致拥塞控制效果不理想。研究基于自适应参数AIMD算法的TCP拥塞控制算法是解决当前网络拥塞问题、提高TCP源端性能的有效途径之一。

2 A-AIMD算法

AIMD算法中参数α、β的不同会影响TCP连接的发送速率[11]。当α较大时，拥塞窗口增加很快，使网络可利用带宽迅速地被占用；当α较小时，拥塞窗口增加较慢，发送速率增加也较平缓；当β较大时，遇到网络拥塞，发送窗口减小较平缓；当β较小时，遇到网络拥塞，发送窗口减小较剧烈，速率抖动较大。根据以上分析可知：如果TCP拥塞控制机制使用不同参数值的AIMD算法，网络数据传输时将会得到不同的拥塞控制效果。

A-AIMD算法的设计目标就是通过网络参数将网络传输情况分为不同的拥塞状态，并根据不同拥塞状态动态地选用不同参数值的AIMD算法对网络数据传输进行拥塞控制，自适应地调节TCP发送速率，降低网络稳定运行时发送速率的抖动性，提高TCP流的平均发送速率，同时提高对网络中可用资源占用的响应速度，使网络运行在较理想的状态。

该算法是针对AIMD算法的不足，基于文献[12]推导的具有TCP友好性[13]时AIMD算法参数的设置关系而提出的。首先根据网络中数据包的回路往返时间（RTT）变化情况，把数据传输时网络拥塞状态分为3种：起始状态、响应状态和平稳状态。A-AIMD算法是在不同拥塞状态使用不同参数值的AIMD算法。网络中的TCP连接建立后，对发送端发送的每M轮拥塞窗口（Cwnd）内数据包的RTT统计平均值$R_{\text{A-RTT}_{\text{M-ave}}}$进行取样，根据相邻取样值的偏差（$\Delta R_{\text{A-RTT}_{\text{M}}}$）大小以及这时平均回路往返时间$A_{\text{ave-RTT}}$的大小判定当时所处的网络状态。使用对应网络状态下的AIMD参数设置值对数据传输进行拥塞

控制。

轮是指发送端从当前窗口开始发送第一个数据包到接收到最后一个数据包确认的整个过程。通常选取发送端按照当前窗口大小发送第一个数据包作为一轮的开始，收到该窗口内所有数据包的确认包表示本轮结束和下一轮开始。$R_{\text{A-RTT}}$（轮内平均回路往返时间）指在一个轮次内所发送的数据包被成功接收到时，每个数据包的回路往返时间的平均值；$R_{\text{A-RTT}_{\text{M-ave}}}$是指连续 M 轮内一个数据包的平均回路往返时间；$\Delta R_{\text{A-RTT}_{\text{M}}}$是指连续两个 M 轮内平均回路往返时间的偏差；$A_{\text{ave-RTT}}$是指已经确认的所有数据包的平均回路往返时间。

3 种状态的设计详细介绍如下。

1）起始状态。A-AIMD 算法的参数值设置为 $\alpha=1$，$\beta=0.5$，其对应 AIMD 算法关系

$$\begin{cases}\text{I}: W_{t+R}\leftarrow W_t+1\\ \text{D}: W_{t+R}\leftarrow 0.5W_t\end{cases}\tag{2}$$

进入该状态的触发条件是：TCP 连接建立、复位时，以及进入响应状态和平稳状态的条件不满足时，起始状态使用的 AIMD 算法参数值和通用的 Reno 算法中参数值完全一样，以保证使用 A-AIMD 算法的 TCP 流在连接建立后数据传输的起始阶段和现有 TCP 流一致，且该数据流传输时具有友好的切入特性，不过份侵占可用的网络带宽。

2）响应状态。A-AIMD 算法的参数值设置为 $\alpha=1.62$，$\beta=0.3$，对应 AIMD 算法关系

$$\begin{cases}\text{I}: W_{t+R}\leftarrow W_t+1.62\\ \text{D}: W_{t+R}\leftarrow 0.3W_t\end{cases}\tag{3}$$

由于 TCP 协议使用的是“累积确认”，即每收到一个 ACK 包，确认 $b(b\geqslant 1)$ 个数据包，所以实际设计时如果当前拥塞窗口为 W_t，选取

$$W_{t+b\times R}=W_t+1.62b\tag{4}$$

其中 $W_{t+b\times R}$为 t 时刻后经过 b 个往返时间时的窗口大小。对 $W_{t+b\times R}$进行取整运算，将其值作为 b 个 RTT 后的拥塞窗口值。同理，当窗口减小时也要对调整后的拥塞窗口值做取整运算。

进入响应状态的触发条件是：$R_{\text{A-RTT}_{\text{M-ave}}}\leqslant\Theta$ 且 $\Delta R_{\text{A-RTT}_{\text{M}}}<\omega$（$\Theta$，$\omega$ 均为常数）。通过触发条件的约束可判定出此时网络可用带宽剩余很大，可使用参数值较激进的 AIMD 算法对可用带宽进行占用，发送窗口迅速加大，传输速率达到了快速响应的目的。为了跟随链路资源，在最大限度地占有可利用链路资源的同时与资源增减保持一致性，这时遇到网络数据丢包时拥塞窗口的减小程度也较为激进，数据传输的抖动较大。

3）平稳状态。平稳状态时 A-AIMD 算法的参数值 $\alpha=0.33$，$\beta=0.8$，其对应的 AIMD 算法关系式

$$\begin{cases}\text{I}: W_{t+R}\leftarrow W_t+0.33\\ \text{D}: W_{t+R}\leftarrow 0.8W_t\end{cases}\tag{5}$$

实际算法设计时选取

$$W_{t+b\times R}=W_t+0.33b\tag{6}$$

对 $W_{t+b\times R}$进行取整运算，将其值作为 b 个 RTT 后的拥塞窗口值。同理，当窗口减小时也要对调整后的拥塞窗口值做取整运算。

进入平稳状态的触发条件是：$R_{\text{A-RTT}_{\text{M-ave}}}>\xi$ 且 $R_{\text{A-RTT}_{\text{M-ave}}}<\eta$（$\xi$，$\eta$ 为常数且两者的大小都接近于 $A_{\text{ave-RTT}}$值）。通过触发条件的约束，可以判定出此时网络中数据包的 RTT 较大，但小于平均回路往返时间 $A_{\text{ave-RTT}}$，且波动较小，这时网络带宽已经充分利用。所以对其使用参数值较为保守的 AIMD 算法，可保证其数据发送时拥塞窗口平缓增加、缓慢减小，实现数据传输时发送速率稳定在较高的阶段，从而提高了 TCP 平均发送速率。$R_{\text{A-RTT}_{\text{M-ave}}}$值愈接近 $A_{\text{ave-RTT}}$值，数据平均发送速率将趋于稳定，波动性较小，与 Reno 算法具有较好的 TCP 友好性。

以上是对 A-AIMD 算法在 3 种网络状态下运行参数的设计，触发条件是 Θ，ω，ξ，η 的选取是基于多次仿真试验取得的最优经验值，且在实际 TCP 运行中，由状态切换而改变 AIMD 算法参数值的行为会

对状态切换前后网络的 RTT 带来影响，进而使触发条件的判决产生偏差。为了消除这种影响，算法设计时设置了一个状态固有滞留时间，一个新的状态必需经过一个状态固有滞留时间后才能再次切换工作状态。设置状态固有滞留时间为 L 轮数据包的发收时间，设定当切入到新状态时，要保持该状态下的 AIMD 算法参数值至少发送 L（L 为常数）轮数据包后才能进行状态转移前的条件判决。

在 A-AIMD 算法中，TCP 根据状态切换条件跳转到相应的网络状态下工作，其 TCP 拥塞控制机制除 AIMD 算法参数值不同外，其余的具体操作和 TCP Reno 中相同。A-AIMD 核心算法的设计流程图如图 1 所示。

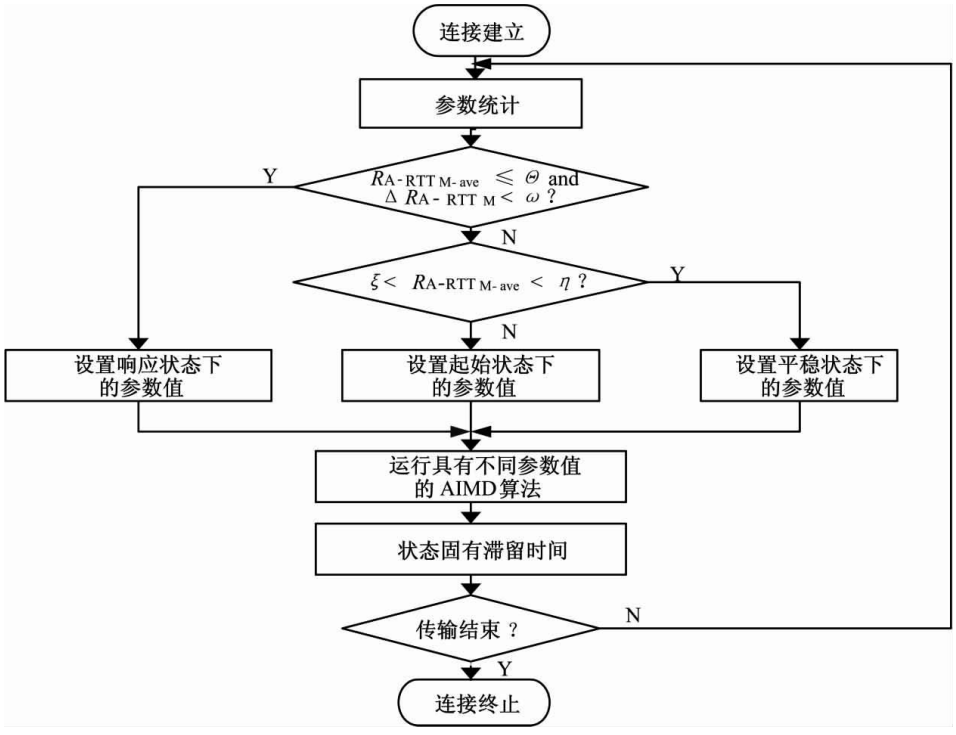

图 1 A-AIMD 核心算法的设计流程图

Fig. 1 Flow chart of the core A-AIMD algorithm

3 算法仿真

为验证所设计 A-AIMD 算法的有效性，笔者设计了以下仿真实验并对 A-AIMD 算法和 Reno 算法的 TCP 发送速率进行对比。使用 NS2 作为仿真工具[4]，经典的哑铃型拓扑结构，拓扑如图 2 所示。其中 A，B 为发送端，C，D 为中间路由器，E，F 为接收端。A，B 发送的都是 FTP 单播数据流，A 发送 A-

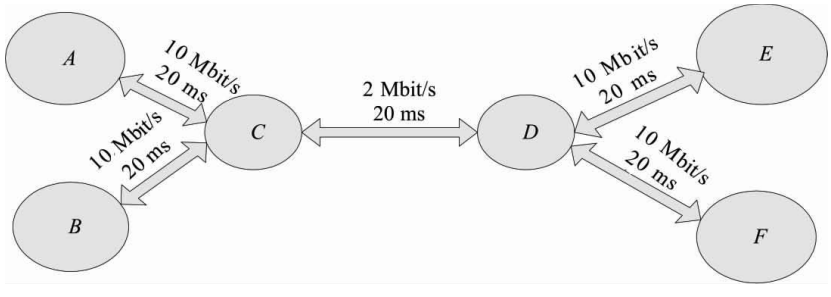

图 2 网络拓扑结构

Fig. 2 Network topology

AIMD 算法的数据流到 E，B 发送 TCP Reno 数据流到 F，两组单播数据流使用相同中间节点上的瓶颈链路。为了验证所设计算法在不同队列管理算法下的性能情况，分别在路由器 C 使用 DropTail 和 RED

(Random Early Detection) 队列管理算法[5]。仿真时设置发送包的大小为 1 000 Byte，仿真时间为 20 s，发送速率由拥塞窗口 (cwnd) 大小计算得到。RED 算法使用默认参数设置。两种算法下均设置：Buffersize = 30 包。由于 A-AIMD 算法要对比发送窗口相邻两个 M 轮数据包的平均回路往返时间偏差 ($\Delta R_{\text{A-RTT}_M}$)，所以要求状态固有滞留时间 $L \geqslant 2M$，这里仿真时取参数值：$M=2$，$L=5$，$\Theta = 0.7\text{A}_{\text{ave-RTT}}$，$\omega = 0.2\ A_{\text{ave-RTT}}$，$\xi = \Theta$，$\eta = 1.1\ A_{\text{ave-RTT}}$。

图 3 分别是使用 DropTail 和 RED 队列管理算法时 TCP 流工作在 A-AIMD 和 Reno 算法发送速率的仿真对比图。由图 3 可以看出，A-AIMD 算法与 Reno 算法相比，TCP 流的平均发送速率得到了提高。在数据传输刚刚开始阶段两者发送速率相差不大，随后两者的发送速率差距有所加大。当网络平稳运行时 ($t > 8$ s)，A-AIMD 算法的瞬时发送速率波动较小，发送速率值变化较平缓。

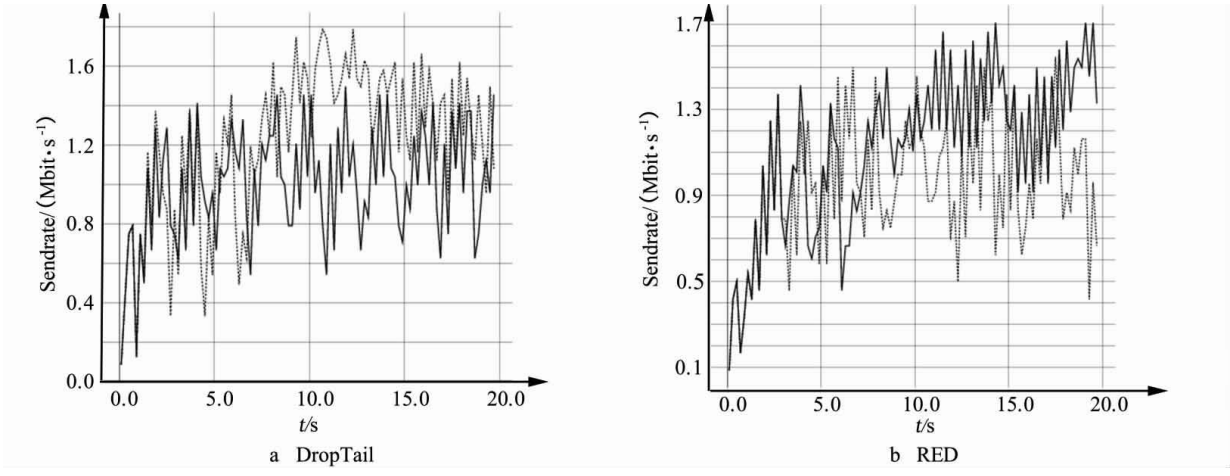

图 3　A-AIMD 和 TCP Reno 算法的发送速率

Fig. 3 Comparison of sending rates between A-AIMD and TCP Reno

使 A 端发送 Reno 流，其他网络环境设置不变。将其与 A 端发送 A-AIMD 流时记录的仿真数据进行比较，统计得到二者后分别使用 DropTail 和 RED 队列管理算法仿真时的平均发送速率并汇总于表 1，其中 $\Delta S = S_i - S_0$ ($i = 1, 2$)，S_1 为 A 端发送 A-AIMD 流时的平均发送速率，S_2 为 A 端发送 Reno 流时的平均发送速率，$S_0 = 1$ Mbit/s，即为满足绝对公平性时每端的发送速率。

表 1　A-AIMD 算法和 Reno 算法的 TCP 平均发送速率

Tab. 1 Average sending rates of A-AIMD and TCP Reno flows

算法	队列管理算法	平均发送速率/ (Mbit·s^{-1})	$\lvert \Delta S/S_0 \rvert$ /%
A-AIMD	DropTail	1.163 0	16.30
	RED	1.127 2	12.72
Reno	DropTail	1.034 7	3.47
	RED	0.974 4	2.56

由表 1 可以看出，在设定相同的网络环境下，队列管理无论使用 DropTail 还是 RED，使用 A-AIMD 算法的 TCP 流平均发送速率都略高于 TCP Reno 算法的平均发送速率，但 A-AIMD 算法仍具有一定的友好性。说明 A-AIMD 流比 TCP Reno 流竞争带宽的能力略强，但仍然不失为友好流。

为检验 A-AIMD 算法对网络中的可用带宽资源的响应特性，在图 2 仿真场景中增加了两个端点，其拓扑结构如图 4 所示。

图 4 中 G，H 为新增加的链路端点，G 使用 TCP NewReno 算法发送 TCP 流到 H。仿真时发送端 G 的发送时间为 10 s，发送端 A，B 的发送时间为 20 s，各个端点都从 $t = 0$ 时开始发送数据。

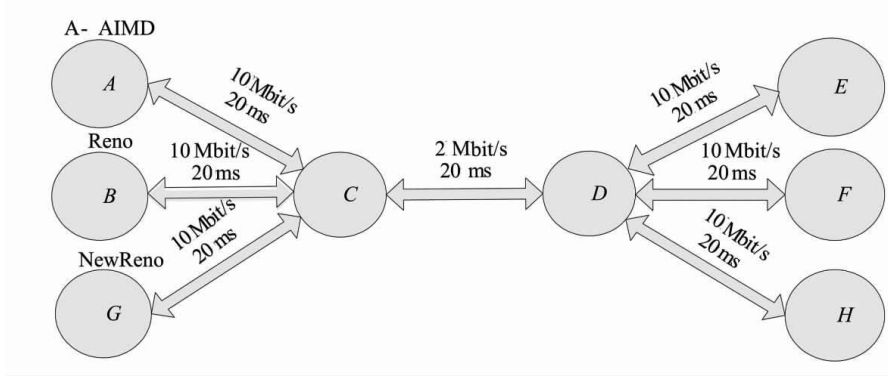

图4 多源端时仿真的拓扑结构图

Fig. 4 Network topology for many sources

图5给出了多源端时3种算法的发送速率。由图5可看出，在前10 s内，3种算法的TCP发送速率相差不大，网络中共存着3种算法的TCP数据流。在第10 s时，NewReno算法的数据流停止发送，这时瓶颈链路中出现了原先NewReno数据流大小的可用带宽资源。由两个仿真图5a和图5b可以看出，在第10 s以后A-AIMD算法的TCP流首先达到对可用带宽的最大占用，发送速率略高于Reno算法下的发送速率，同时，此刻又未过分侵占Reno流的已有带宽，表现出与Reno算法具有一定的TCP友好性。

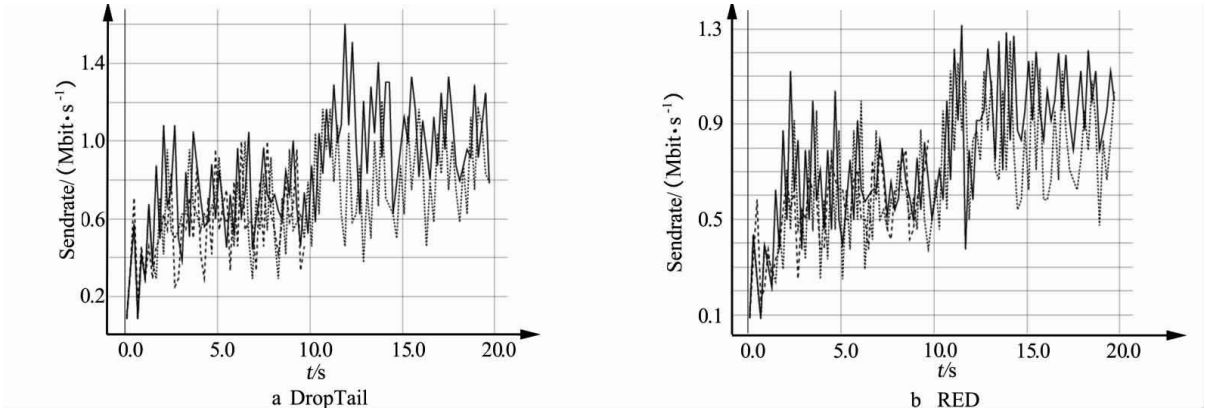

图5 多源端时3种算法的发送速率

Fig. 5 Comparison of sending rates among A-AIMD, Reno and Newreno

使A端发送Reno流，其他网络环境设置不变。将其与A端发送A-AIMD流时记录的仿真数据做比较，统计得到二者分别使用DropTail和RED队列管理算法时，在仿真过程中第10 s到20 s瓶颈链路的平均链路利用率如见表2所示。由表2可知，在设定相同的网络环境下，队列管理无论使用DropTail还是RED，调用A-AIMD算法时的平均链路利用率都要高于调用TCP Reno算法。

仿真对比表明，与TCP Reno算法相比，在起始状态A-AIMD算法下发送速率的行为特征和Reno下的行为特征是一致的，保证该数据流传输时具有友好的切入特性，不过份侵占可用的网络带宽；在响应状态A-AIMD算法的发送速率首先达到最大值，响应速度较快；在平稳状态时A-AIMD算法的TCP发送速率变化较平稳，从而提高了资源利用率。同时A-AIMD仍具有一定的TCP友好性。

表2 多源端时A-AIMD和Reno算法的平均链路利用率

Tab. 2 Link utilization of A-AIMD and TCP Reno flows

算法	队列管理算法	平均链路利用率/%
A-AIMD	DropTail	92.806 0
	RED	88.658 5
Reno	DropTail	79.464 5
	RED	74.048 0

4 结　语

笔者提出了一种基于自适应设置参数的 AIMD 算法。该算法根据网络中回路往返时间 RTT 的变化情况，将网络运行情况分为 3 种不同的网络拥塞状态。不同网络拥塞状态下使用不同参数值的 AIMD 算法，能动态地调整数据传输时所使用的拥塞控制机制，进而使网络数据传输运行在较理想的状态。最后对所设计的 A-AIMD 算法进行实现，并结合不同的队列管理算法与 TCP Reno 进行了对比仿真。仿真结果表明，当网络处于稳态运行时，使用 A-AIMD 算法能提高 TCP 流的平均发送速率、减小发送速率的波动性；当网络中有可利用的链路资源时，使用 A-AIMD 算法能快速地对可用资源进行最大占用，且与 Reno 算法下的数据流保持一定的 TCP 友好性。

笔者提出的算法在提高现有 TCP 拥塞控制算法性能的同时，算法复杂度变化不大，系统额外开销小，有利于实现和应用，且与 TCP 一起使用时，可以实现网络资源的公平竞争，并提高网络资源的利用率。算法在实际环境下还需要结合实际网络环境对网络状态细化和参数设置作进一步的探讨。

参考文献:

[1] FLOYD S, FALL K. Promoting the Use of End-to-End Congestion Control in the Internet [J]. IEEE/ACM Transactions on Networking, 1999, 7 (4): 458-472.

[2] LIU Gang, ZHANG De-yun, LIU Jing, et al. Improving Throughput for TCP Vegas [J]. The Journal of China Universities of Posts and Telecommunications, 2004, 11 (2): 60-65.

[3] JACOBSON V. Congestion Avoidance and Control [J]. ACM Computer Communication Review, 1988, 18 (4): 314-329.

[4] KEVIN FALL, FLOYD S. Simulation-Based Comparisons of Tahoe, Reno, and SACK TCP [J]. ACM Computer Communication Review, 1996, 26 (3): 5-21.

[5] CHIU D M, JAIN R. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks [J]. Computer Networks and ISDN Systems, 1989, 17 (1): 1-14.

[6] ROBERT SHORTEN, DOUGLAS J LEITH, PETER WELLSTEAD. An Adaptive AIMD Congestion Control Protocol for Communication Networks [C] //Networking2004. Berlin: Springer, 2004: 699-711.

[7] ALEX KESSELMAN, YISHAY MANSOUR. Adaptive AIMD Congestion Control [J]. Algorithmica (New York), 2005, 43 (1/2): 97-111.

[8] YANG RICHARD YANG, SIMON S LAM. General AIMD Congestion Control [C] //International Conference on Network Protocols. Washington, DC, USA: IEEE Computer Society, 2000: 187-198.

[9] AYDER NATIQ JASEM, ZURIATI AHMAD ZUKARNAIN, MOHAMED OTHMAN, et al. The TCP-Based New AIMD Congestion Control Algorithm [J]. International Journal of Computer Science and Network Security, 2008, 8 (10): 331-338.

[10] XU Chang-biao, LONG Ke-ping, YANG Shi-zhong. Improving Network Performance by Ameliorating TCP Congestion Control Mechanism [J]. The Journal of China Universities of Posts and Telecommunications, 2002, 9 (1): 1-6.

[11] 杨晓萍，史帅，陈虹. 一种改进的 TCP 拥塞控制算法 [J]. 吉林大学学报：工学版，2006，36 (3)：433-437.
YANG Xiao-ping, SHI Shuai, CHEN Hong. Enhanced Congestion Control Algorithm of TCP [J]. Journal of Jilin University: Engineering and Technology Edition, 2006, 36 (3): 433-437.

[12] 杨晓萍. TCP 拥塞控制和区分服务队列管理研究 [D]. 长春：吉林大学，2007.
YANG Xiao-ping. Research on TCP Congestion Control and Queue Management in DiffServ Network [D]. Chuangchun: Jilin University, 2007.

[13] WIDMER J, DENDA R, MAUVE M. A Survey on TCP-Friendly Congestion Control [J]. IEEE Network, 2001, 15 (3): 28-37.

[14] UCN/LBL/VINT. Network Simulator-NS2 [EB/OL]. [2003-09]. http://www.mash.cs.berkeley.edu/ns.

[15] 杨晓萍，陈虹，翟双. 基于路由器的 RED 和 Droptail 算法比较 [J]. 吉林大学学报：信息科学版，2005，23 (1)：69-74.
YANG Xiao-ping, CHEN Hong, ZHAI Shuang. Comparison of RED and Droptail Algorithms Based on Router [J]. Journal of Jilin University: Information Science Edition, 2005, 23 (1): 69-74.

(责任编辑：刘东亮)