文章编号: 1001-0920(2006)12-1392-05

一种新的网络拥塞控制算法：API-V

陆锦军[1,2], 王执铨[1]
(1. 南京理工大学 自动化学院, 南京 210094; 2. 南通职业大学 现代教育技术中心, 江苏 南通 226007)

摘 要: 针对PI控制器响应速度的不足, 提出一种自适应网络动态变化的主动队列管理算法: API-V 控制器. 在PI控制器的基础上, 根据瞬时队列长度增加速度控制, 根据实时测量链路的数据包丢失率获得当前的负载信息, 动态调整PI算法中的有关参数. 理论分析和仿真结果表明, 相对于PI控制器及其改进算法, API-V 控制器具有更快的响应速度、收敛速度和更小的队列抖动, 并且提高了缓冲区的利用率.

关键词: 主动队列管理; API控制器; API-V 控制器
中图分类号: TP393 **文献标识码**: A

New Network Congestion Control Scheme: API-V

LU Jin-jun[1,2], *WANG Zhi-quan*[1]
(1. Department of Automation, Nanjing University of Science and Technology, Nanjing 210094, China; 2. Center of Education and Technology, Nantong Vocational College, Nantong 226007, China. Correspondent: LU Jin-jun, E-mail: ljj@mail.ntvc.edu.cn)

Abstract: An adaptive proportional integral-velocity (API-V) controller for new active queue management scheme is proposed to improve the slow response of PI control, which is well suited to dynamic network environments. A velocity controller is added based on PI controller when the length of queue is larger. Load information is obtained by present measured dropping probability. The relevant parameters in PI scheme are adjusted dynamically. Analysis and simulation results show that API-V controller not only has faster convergence speed, quicker response and smaller queue oscillation than PI controller and improved PI controller but also increases the buffer utilization.

Key words: AQM; Controller of self-adaptation proportional integral; Controller of self-adaptation proportional integral-velocity

1 引 言

随着网络规模的扩大, 网上业务量不断增长, 网络拥塞现象越来越严重. 于是, 应用控制理论分析TCP/AQM 模型并设计相应的主动队列管理(AQM)算法, 成为当前TCP/IP 网络控制领域的研究热点之一.

Floyd 提出的RED 算法和PI控制器是两种著名的主动队列管理算法. RED 算法是根据平均队列长度计算报文丢弃概率, 理论分析和实验研究表明, RED 及其改进算法的性能与参数配置密切相关, 不正确的参数配置将导致很大的队列抖动[1]. 与其他主动队列管理算法相比, 自适应RED 算法对Web 的性能改进最小, 并且缺乏严谨的科学论证. REM[2] 算法是基于优化的方法, 更多地关注系统的稳态特性, 而不是队列的瞬态性能. Hollot 基于所建控制模型的线性化[3], 提出了PI算法. 理论分析和实验结果表明, PI算法比RED 算法具有更小的队列抖动, 从而在保证高带宽利用率的前提下, 为端用户提供更小的延时抖动. 但在PI算法中, 参数是固定设置的, 因此该算法在较小的目标队列长度下收敛速度

收稿日期: 2005-11-09; **修回日期**: 2006-02-15.
基金项目: 国家自然科学基金项目(60374066); 江苏省自然科学基金项目(BK2004132); 江苏省高校自然科学计划项目(04KJD120151).
作者简介: 陆锦军(1964—), 男, 江苏南通人, 副教授, 从事网络系统、智能控制的研究; 王执铨(1939—), 男, 武汉人, 教授, 博士生导师, 从事网络系统、鲁棒控制等研究.

较慢. 针对PI算法的缺点, 先后出现了一些改进算法[4~8], 但效果都不够理想.

本文提出一种新的自适应主动队列管理算法: API-V 控制器. 当瞬时队列长度小于启动门限值H时, 实现API算法, 即根据实时测量链路的数据包丢失率, 动态调整PI算法的配置参数, 保证算法有更快的收敛速度和更小的队列抖动; 当IP流量突发, 瞬时队列长度大于启动门限值H时, 启动速度控制项, 实现API-V算法, 保证对网络状态变化的快速响应, 并且提高了缓冲区的利用率. 理论分析和仿真结果表明, API-V算法在动态网络环境下, 性能优于PI算法以及文献[4~6]中的改进算法.

2　TCP/AQM (PI) 控制理论模型

Hollot 等提出的TCP/AQM (PI) 系统, 其开环传递函数为

$$G(s)=\frac{1+\tau s}{Ts}\cdot\frac{K_m e^{-Rs}}{(T_1 s+1)(T_2 s+1)}. \tag{1}$$

其中

$$K_m=\frac{(RC)^3}{4N^2},\ T_1=\frac{R^2C}{2N},\ T_2=R;$$

τ和T为待确定的参数, q_0为目标队列长度, p为数据包丢失率, N为系统的负载, R为连接的RTT.

当系统稳定时, $p(k)$在k时刻的数据包丢失率为

$$p(k)=a[q(k)-q_0]-b[q(k-1)-q_0]+p(k-1),$$

$$a=\frac{\tau}{T}+\frac{1}{Tf},\ b=\frac{\tau}{T}.$$

其中: $q(k)$是在k时刻的瞬时队列长度, f是PI算法采样的频率. 参数a和b是固定不变的, 不能随网络动态变化而灵活设置, 因而导致PI算法收敛速度较慢. 为此, 本文提出了自适应网络动态变化且响应速度和收敛速度更快的新算法: API-V 控制器.

3　API-V 控制器

3.1　API-V 控制器模型

API-V 控制器不仅在PI的基础上实现了API算法, 而且在API控制器的基础上增加了归一化的速度控制项v/C. 其计算分组标记/丢弃概率的差分方程为

$$p(k)=\begin{cases}p(k-1)+a[q(k)-q_0]-b[q(k-1)-q_0]+\lambda[v(k)/C], & q>H,\\ p(k-1)+a[q(k)-q_0]-b[q(k-1)-q_0], & q\le H.\end{cases} \tag{2}$$

其中: λ为常数, H为速度控制启动门限, $v(k)$为第k个抽样时间的瞬时输入速度, C为链路带宽, $p(k)$为第k个抽样时间的丢弃概率, $q(k)$为第k个抽样时间的瞬时队长. 常数a和b与PI控制中不同, 不是定值, 而是随着网络动态变化而调整.

当瞬时队列长度大于启动门限值H时, 在API的基础上启动速度控制项, 实现API-V 控制; 反之, 保持API控制器, 即根据实时测量链路中的数据丢失率, 获得当前负载信息, 动态设置PI控制器的相关参数. 对于当前节点的负载变化, 引入速度控制, 使得算法对网络环境变化的响应速度更快, 减小了由于计算队列长度引起的时延对响应速度的影响. 并且只有当瞬时队列长度超出期望队列长度一定的范围时, 才启动速度控制, 这样既能进一步提高响应速度, 又能保留API控制器在稳定系统响应方面的优点, 也可消除不同链路带宽对丢失概率值的影响. 根据仿真经验, $a\le\lambda\le 10a$, $2q_0\ge H\ge 1.5q_0$.

3.2　API-V 算法本质及其分析

当$q>H$时, 启动速度控制, 设队列长度为H, $p(0)=0$, 则

$$p(k)=k[(a-b)(H-q)+\lambda v/C]. \tag{3}$$

令API-V 控制器的响应时间为T_r, 队长由H降到q_0所需的丢弃概率为p, 则有

$$T_r=\frac{(p-0)T}{p(k)-p(k-1)}=\frac{pT}{(H-q_0)(a-b)+\lambda v/C}. \tag{4}$$

由式(4) 可以看出, 由于在API-V 中引入了速度控制, 相对于PI控制器, API-V 控制器响应时间变小. 同时也可看出, 如果要求到达相同的响应速度, API-V 只需很小的缓冲区间即可提高缓冲区的利用率. 因此当网络环境变化时, API-V 能使队列长度以最短的时间稳定到q_0附近, 从而保证时延、链路利用率等QoS 的要求.

瞬时队列长度和瞬时输入速度的关系为

$$q'(t)=v(t)-C, \tag{5}$$

其差分方程为

$$\frac{q(k)-q(k-1)}{T}=v(k)-C. \tag{6}$$

式(6) 代入式(2), 可得

$$p(k)=p(k-1)+a[q(k)-q_0]-b[q(k-1)-q_0]+\frac{\lambda}{C}\left[\frac{q(k)-q(k-1)}{T}+C\right] \tag{7}$$

在s域中表示为

$$C(s)=K_p+K_i/s+K_\alpha s,$$

即API-V 控制器为一个参数可调的PID 控制器.

TCP/AQM 的线性控制系统如图1 所示. 其中$p(s)e^{-Rs}$为需要控制的TCP/AQM 系统, 其模型为

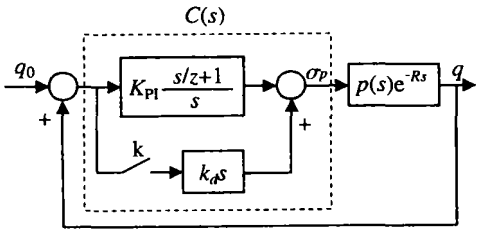

图 1 API-V-TCP/AQM 线性控制系统

$$\frac{(RC)^3}{4N^2}e^{-Rs}/\left(\frac{R^2C}{2N}s+1\right)(Rs+1).$$

其中: R 为平均往返时间, σ_p 为当前丢失概率与工作点丢失概率 p_0 的差值. 控制器根据当前队长和期望队长的差值计算出当前丢失概率. 由于源端 TCP 流对分组丢弃作出响应, 控制分组发送速度进而影响当前队长.

4 API控制器

4.1 根据数据包丢失率推算链路负载信息的理论依据

根据文献[9]有关 TCP 吞吐量的计算表达式

$$T=\frac{MSS}{R\sqrt{\frac{2p}{3}}+\tau_{RTO}\left(\frac{3}{2}\sqrt{\frac{3p}{2}}\right)p(1+32p^2)}, \quad (8)$$

其中: R 和 p 的意义同前, MSS 是报文大小, τ_{RTO} 是连接超时值, 一般取 $\tau_{RTO}=5R$. 则式(8)变为

$$T=\frac{MSS}{R\left[\sqrt{\frac{2p}{3}}+\left(\frac{15}{2}\sqrt{\frac{3p}{2}}\right)p(1+32p^2)\right]}. \quad (9)$$

令

$$h(p)=\left[\sqrt{\frac{2p}{3}}+\left(\frac{15}{2}\sqrt{\frac{3p}{2}}\right)p(1+32p^2)\right]/MSS. \quad (10)$$

若 TCP 链路变为 C, 连接的数据为 N, 每条连接的 PTT 为 R, 则式(9)变为

$$T=C/N=\frac{MSS}{R\left[\sqrt{\frac{2p}{3}}+\left(\frac{15}{2}\sqrt{\frac{3p}{2}}\right)p(1+32p^2)\right]}=1/h(p)R. \quad (11)$$

令

$$\alpha=\left[\sqrt{\frac{2p}{3}}+\left(\frac{15}{2}\sqrt{\frac{3p}{2}}\right)p(1+32p^2)\right]\frac{C}{MSS}=h(p)C, \quad (12)$$

则 $N=\alpha R$. 定义

$$\Omega_p=\{\langle N,R\rangle \mid N-h(p)CR=0\},$$

若链路丢失率 p 已知, 则可推测负载 N 和 R.

4.2 API算法参数调整的理论依据

定理 1[10] 对于 N 和 R 变化的系统, 当数据丢失率为 p 时, 取

$$\omega_g=\min\left\{\frac{\beta}{\sqrt{T_1T_2}}\,\middle|\,(N,R)\in\Omega_p\right\},$$

$$\beta\le\frac{\pi/4}{\sqrt{2h(p)}},\ \tau\ge\frac{1}{\omega_g},$$

$$T\ge\max\left\{\frac{k_m\tau}{\sqrt{(1-T_1T_2\omega_g^2)^2+(T_1+T_2)^2\omega_g^2}}\right\},$$

则 $\forall\langle N,R\rangle\in\Omega_p$, 系统稳定.

证明略.

定理 2 对于 N 和 R 变化的系统, 当报文丢失率为 p, 最大 RTT 为 R_{max} 时, 取

$$\omega_g=\frac{\pi/4}{R_{max}},\ \tau\ge\frac{1}{\omega_g},$$

$$T\ge\max\left\{\frac{k_m\tau}{\sqrt{(1-T_1T_2\omega_g^2)^2+(T_1+T_2\omega_g^2)^2}}\right\},$$

则 $\forall\langle N,R\rangle\in\Omega_p$, 系统稳定.

证明略.

说明 1 API算法比PI算法具有更大的开环带宽, 开环带宽越大, 系统的收敛速度越快. 当 $N<N_{min}$ 时, 由PI算法给出的稳定条件可能使系统不稳定, 但按定理进行参数调整, 仍可使系统处于稳定状态. 当丢失率为 p 时, API算法给出的稳定条件比PI算法给出的稳定条件更加严格.

4.3 关于丢失率 p 的改进计算

文献[11]中数据包的标注概率计算为瞬时队列长度的线性函数, 文献[7]论述了几何随机数计算 p 方法的不足之处. 本文采用均匀分布的随机数法.

每个包的标注概率均为 p, 每两个标注包之间的间隔次数 X 为所到达的包的数目. 为使 X 为{1, 2, …, $1/p$}中的均匀分布随机数(为简单起见, 假设 $1/p$ 为整数). 对每个达到的数据包的标注概率取 $p/(1-\text{count}\times p)$, 其中 count 是自上一个标注的包以来所达到的未标注包的数目. 实际上就是取 X 为{count, …, $1/p$}中均匀分布的随机数(count 可为 1, 2, …, $1/p$). 在这种情况下, 有

$$\text{prob}[x=n]=\frac{p}{1-(n-1)p}\prod_{i=0}^{n-2}\left(1-\frac{p}{1-ip}\right)=\frac{p}{1-(n-1)p}(1-p)\times\frac{1-2p}{1-p}\frac{1-3p}{1-2p}\cdots\frac{1-(n-1)p}{1-(n-2)p}=p,$$

$$1\le n\le 1/p;$$

以及

$$\text{Prob}[X=n]=0,\ n>1/p,$$

$$E[X] = \sum_{k=1}^{1/p} kp(X = k) =$$
$$p + 2p + \cdots + 1/p =$$
$$p(1 + 2 + \cdots + 1/p) =$$
$$1/(2p) + 1/2$$

该方法比其他方法的均值小很多, 因为正常情况下, 丢失的概率是值较小的小数. 可以看出, 这种方法两个标注包之间的间隔包的数目相对比较规范, 分布比较均匀.

4.4 API算法设计

根据前述理论依据和定理设计API算法, 即根据丢失率p推算参数α和b数值. 此处仅给出主要程序段, 所用变量均需先定义.

```
const float C = C1
const float pi = 3.14
const float R1 = 常量
int i = 0
  do {
  i++
  p = i/100
  if (p == 0) p = 0.001
Alpha= (sqrt(2*p/3+ (15/2)*sqrt(3*
    p/2)*p*(1+ 32*p*p))*
    C/MSS
  R = R1  //令R = R1
  N = Alpha*R1  //则N = Alpha R1
  T1 = (R1*R1*C)/(2*N)
  T2 = R1
KM = ((R1*C)*(R1*C)*(R*C))/
  (4*N*N)
  ωj = (Pi/4)/R  //R 相当于Rmax
  t = 1/ωj
T = KM/sqrt((1- T1*T2*ωj*ωj)(1-
  T1*T2*ωj*ωj)+ (T1+ T2)*
  (T1+ T2)*ωj*ωj)
  //据t和T 求α和b值
  a = t/T + 1/(T*f)  //f 为常量
  b = t/T}
while(i< 40)
```

5　仿真及其分析

下面利用NS2进行仿真, 以验证API-V 控制器的性能, 同时与PI控制器进行比较. 其模拟拓扑如图2所示.

在图2中, 1～n为发送节点, C_1～C_n为接收节点, A和B为路由器, $i \to A$和$B \to C_i$的传输延迟为[10 ms, 50 ms]之间平均分布的随机延迟, $A \to B$的链路构成瓶颈. A上分别运行API-V 算法和PI算法, 目标队列长度设置为100个数据; TCP使用Reno, 数据包大小为1 000字节; CBR报文大小为500字节, 报文发送间隔为0.5 s; HTTP短连接的大小为10 K字节, PI算法和API-V 算法的参数设置如表1所示.

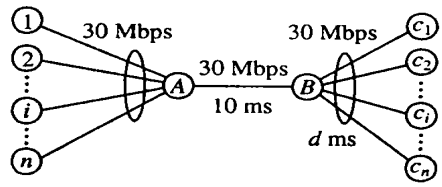

图2　模拟拓扑结构

表1　对比算法参数设置

算法	A	B	T/s	λ	H
PI	1.812×10^{-5}	1.826×10^{-5}	1/180		
API-V	1.812×10^{-5}	1.826×10^{-5}	1/180	10×10^{-5}	150

实验1比较了API-V 算法和PI算法的响应速度. 在0 s～10 s之间启动200条FTP连接, 然后在50 s～60 s之间启动200条FTP连接, 总的持续时间为100 s. 瞬时队列长度随时间变化的过程如图3所示. 从图3可以看出, API-V 算法不仅具有较快的收敛速度, 而且具有比PI算法更小的队列抖动.

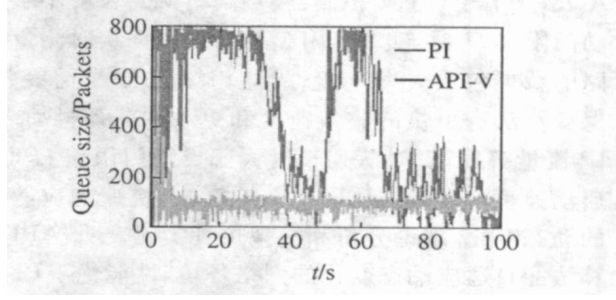

图3　实验1模拟结果

实验2验证了API-V 算法在重负载情况下的性能. 在0 s～10 s之间启动500条TCP连接, 模拟结果如图4所示. 从图4可以看出, 在重负载情况下, PI算法的收敛速度很慢, API-V 算法虽然经过了从API算法到API-V 算法的转变, 但响应曲线比较平滑, 其算法响应时间近为10 s. 由此可见, API-V 算法的性能优于PI算法.

实验3测试了API-V 算法在混合连接类型情况下的性能. 在0 s～10 s之间启动150条FTP连接, 在150 s时停止这些连接; 在20 s～30 s之间启动100条CBR连接, 在120 s时停止这些连接; 在40 s～50 s之间启动200条HTTP连接, 在60 s～70 s之间再启动70条FTP连接, 这些连接在140 s时停止; 在80 s～90 s之间启动100条CBR连接, 在160

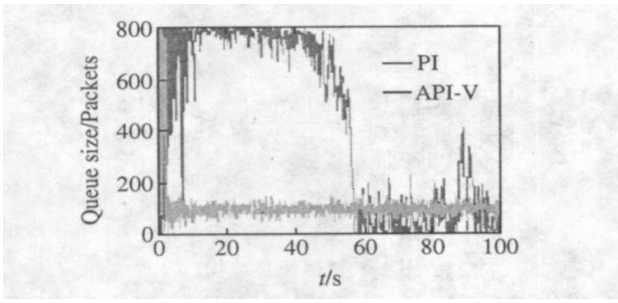

图 4 实验 2 模拟结果

s 时停止这些连接. 模拟结果如图 5 所示, 从实验 3 可以看出, 在有HTTP 连接和CBR 连接存在的情况下, API-V 算法仍然具有较快的收敛速度和较小的队列长度抖动, 性能优于 PI 算法.

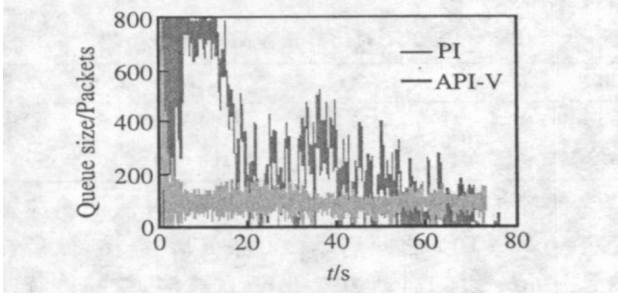

图 5 实验 3 模拟结果

6 结 论

本文针对PI算法的缺点, 提出一种新的自适应 AQM 算法: API-V 控制器. 当瞬时队列长度小于启动门限值H 时, 实现API算法, 即根据实时测量链路的数据包丢失率, 动态调整PI算法的配置参数, 保证算法有更快的收敛速度和更小的队列抖动. 当 IP 流量突发, 瞬时队列长度大于启动门限值H 时, 启动速度控制项, 实现API-V 算法, 快速响应网络的动态变化. 理论分析和模拟仿真实验表明, API-V 算法能自适应动态变化的网络环境, 性能优于PI算法及其改进算法.

参考文献(References)

[1] Hollot C V, Misra V, Towsley D, et al. A Control Theoretic Analysis of RED [A]. *Proc of the IEEE Infocom* [C]. Anchorage: IEEE Communications Society, 2001: 1510-1519.

[2] Athuraliya S, Low S, Li V H, et al. REM: Active Queue Management[J]. *IEEE Network*, 2001, 15(3): 48-53.

[3] Misra V, Gong W B, Towsley D. Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED[A]. *Proc of the ACM Sigcomm* 2000[C]. Stockholm, 2000: 151-160.

[4] Chen Q, Yang O W. A ST-PI-PP Controller for AQM Router[J]. *IEEE Int Conf on Communications*[C]. Atlanta, 2004, 4: 20-24.

[5] Wang C G, Bo L, Kazem S. API: Adaptive Proportional-integral Algorithm for Active Queue Management under Dynamic Environments[A]. *Proc IEEE HPSR* 2004[C]. Hong Kong, 2004: 51-55.

[6] Chang X L, Muppala J K, Yu J T. A Robust Nonlinear PI Controller for Improving AQM Performance[A]. *IEEE Int Conf on Communications*[C]. Atlanda, 2004, 4: 2272-2276.

[7] Cai X L, Wang X F, Wang Z Q, et al. Analysis and Improvement of PI Control for Active Queue Management[J]. *J of Nanjing University of Science and Technology*, 2005, 29(3): 368-371.

[8] Zheng B, Lin C, Li Y. A Queue Management Algorithm Fit for Network Processors[J]. *J of Computer Research and Development*, 2005, 42(10): 1698-1705.

[9] Padhye J, Firoiu V, Towsley D, et al. Modeling TCP Throughput: A Simple Model and Its Empirical Validation[A]. *Proc of The ACM Sigcomm* [C]. Vancouver: ACM Press, 1998: 303-314.

[10] Zhang M J, Zhu P D, Su J S. *API: An Adaptive PI Active Queue Management Algorithm* [R]. Taipei: National University of Technology, 2003.

[11] Hollot C V, Misra V, Towsley D, et al. Analysis and Design of Controllers for AQM Routers Supporting TCP Flows[J]. *IEEE Trans on Automatic Control*, 2002, 47(6): 945-959.