一种基于抖动约束的整形算法

郝俊瑞[1] 余少华[1,2]
(华中科技大学计算机科学与技术学院 武汉 430074)[1] (武汉邮电科学研究院 武汉 430074)[2]

摘 要 通过分析传统的通信量整形算法中由数据缓冲引入的延迟抖动问题,提出了一种基于抖动约束的通信量整形算法。在整形过程中,延迟是由数据缓冲的充满程度和令牌输出速率决定的。通过在线检测数据缓冲的充满程度和令牌的输出速率,计算出数据包的延迟和延迟抖动,然后根据延迟抖动约束动态调整整形器的参数,使数据分组在整形过程中经历的延迟抖动保持在约束范围之内。实验结果表明,该算法不仅可以平滑突发数据流,而且可以有效地降低由数据缓冲造成的延迟抖动。

关键词 延迟抖动,通信量整形,令牌桶

Jitter-constrained Shaping Algorithm

HAO Jun-rui[1] YU Shao-hua[1,2]
(College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)[1]
(Wuhan Research Institute of Posts & Telecommunications, Wuhan 430074, China)[2]

Abstract A jitter-constrained traffic shaping algorithm was proposed by analyzing the delay jitter problem of traditional traffic shaping algorithm. In traffic shaping, the delay was determined by length of data buffer and the input rate of the token. This algorithm estimates and computes the delay and the delay jitter of traffic on-line by monitoring the variety of data buffer's length and the rate of token. It updates the parameters of traffic shaper depending on the delay jitter of traffic so as to provide lower jitter stream. The experiments show that the algorithm can smooth the traffic and lower the delay jitter arisen from traffic shaper.

Keywords Delay jitter, Traffic shaping, Token bucket

1 引言

在目前的网络通信中,IP网络应用由单一数据业务扩展到包括话音、视频在内的实时通信业务。随着IPTV,视频监控等应用需求的不断增加,这种变化越来越明显。这些视频实时通信业务的增长对于网络设备也提出了更高的要求。在视频通信中,根据流量的变化,视频流量可以分为两种:一种是恒定码率(CBR)视频,一种是可变码率(VBR)视频[1]。它们是在编码时考虑不同的量化方法而得到的。CBR视频是基于固定输出码率而变化量化阶得到;而VBR视频是基于固定量化阶而变化输出码率得到。CBR视频和VBR视频各有其应用场合。使用CBR视频更有利于传输分析和控制,但CBR视频不能保持固定的视频质量,而且针对一定的视频质量由于内部有大量的填充字节而浪费了较多带宽。而VBR视频有利于保持一定的视频质量,易于对用户提供高质量的视频业务,但VBR在网络上传输时流量变化大、突发性强,如何为VBR分配带宽,保证传输服务质量是网络流量工程的重要课题。

CBR通信量的输出速率保持恒定不变,网络结点预留与通信量输出速率相同的带宽就可以保证传输服务的质量。VBR通信量的输出速率是一个随机的波动过程,网络结点通常要根据VBR通信量的峰值速率分配带宽,预留远大于实际传输所需要的带宽(有效带宽)来达到这些VBR通信量的QoS要求。对VBR通信量的这种带宽分配方式导致网络资源被极大浪费,而且VBR通信量仍然存在着很大的速率波动,这是造成网络拥塞的一个主要原因。因此,在进行网络接入时,通常都要对通信量进行整形,改变它的输出特性,降低峰值速率和减小输出速率的波动,提高网络资源的利用率。但是传统的通信量整形算法通常仅仅是降低了数据流的突发性,而没有考虑在整形过程中所引入的延迟抖动问题。对于像视频流这样对延迟抖动很敏感的实时应用来说,这种缺点是致命的。

本文提出的基于抖动约束的整形算法(Jitter-constrained Shaping,JCS)不仅能平滑VBR视频流的突发性,还可以在延迟抖动约束范围内自适应调整速率,降低输出流的延迟抖动,对于视频等实时应用的服务质量有较大提高。它通过计算缓冲区中数据包的延迟和延迟抖动来自适应地动态调整整形器的参数。在保证数据无丢失和抖动约束的前提下,使整形后的速率更接近源通信量的平均速率,同时降低峰值速率。相比传统的整形器,它能够有效降低传统整形器的延迟抖动,并

到稿日期:2008-03-03 本文获国家"863"项目MSR城域网实用化项目(项目编号:2005AA121411),新一代光纤通信技术和网络国家重点实验室的资助。

郝俊瑞(1975—),男,博士研究生,研究方向为城域以太网,E-mail:freemanhjr@yahoo.com.cn。

且可以在线估计源通信量的性质，产生更加平滑的输出数据流。

本文第 2 节介绍通信量整形的应用框架和一般的 MLB 算法，其中详细介绍应用最广的双漏桶 DLB(dual leaky buckets)整形算法，它也是本文算法的基础；第 3 节介绍本文提出的自适应速率整形算法的主要思想和设计过程，分析满足数据无丢失和最大延迟的条件约束，并且详细描述调整算法的工作过程；在第 4 节，通过实验比较传统 DLB 整形的输出流和经过自适应速率整形输出的数据流的延迟抖动，以及两种算法产生的输出流的平滑效果；最后是本文结论。

2 相关工作

通信量整形通常用于降低输出流的突发度，平滑输出速率。对组播数据流应用通信量整形可以减少网络节点为组播流预留的带宽，提高网络利用率，增强不同网络环境对组播数据的适应性，从而有可能为接收者提供更好的传输服务质量。通信量整形器通常应用在边缘网络的网络接入点，或者是源通信量产生后的下一结点。通信量整形器不仅可以对源通信量进行整形，还可以为接纳控制算法提供相应的通信量输出参数，例如平均带宽、最大突发度等。在文献[2]中描述了如何提供和利用通信量整形器的参数进行有效的接纳控制和通信量调度。

通信量整形器包括一个速率控制算法和一个数据缓冲。速率控制算法控制缓冲区的输出速率，缓冲区的输入由源通信量或者一个成型器(regulator)给出。文献[3-5]提出和讨论了用于通信量整形的几种速率控制算法，其中最通用的是多漏桶 MLB(multiple leaky buckets)整形算法，它既可以在网络结点也可以在端到端系统中实现。通信量整形算法通常分为两类：一类应用于数据无丢失和给定最大延迟前提下的确定性算法；另一类应用于对延迟敏感，但可以容忍少量数据丢失的基于统计的非确定性算法。传统的整形算法[6-10]根据预先设定的平均输入速率、峰值速率和最大突发度等参数设计整形器的数据缓冲的大小和令牌的输出速率。在实际应用中，源数据流的通信模型参数很难被精确测量，而且许多源数据流的平均输出速率在传输过程中是一个长期的波动过程。例如 VBR 视频流，输出的速率和原始视频的复杂程度有关，随着视频场景的变化，瞬时速率和平均速率都会产生变化。所以传统的整形算法可能导致数据包丢失，而且由于源通信量的突发性造成在整形过程中每个数据分组(或者每段分组)的延迟是不相等的。一般情况下，突发较大的部分在整形后的延迟也较大，而突发较小的部分则延迟会很小。在这种情况下，传统整形算法对于视频等这些对延迟抖动特别敏感的实时应用将是无能为力的。

本文中的分析对象是包含多媒体数据的通信量，主要分析 VBR 视频流。VBR 视频流是一个复杂的随机过程，本文对 VBR 视频流进行描述时采用流体模型。为了便于对流体模型的通信量进行分析，需要定义流的几个参数，假设基于流模型的通信量称为 F，$F(t)$表示时间段$[0,t]$中源输出的通信量，$s(t)$表示瞬时速率，λ 表示平均速率，P 表示峰值速率，B 表示突发度：

$$s(t)=\lim_{\Delta t\to 0}\frac{F(t+\Delta t)-F(t)}{\Delta t} \tag{1}$$

$$\lambda=\lim_{t\to\infty}\frac{F(t)}{t} \tag{2}$$

$$P=\max_{\forall t>0}\{s(t)\} \tag{3}$$

$$B=\max_{\forall b>a>0}\{C(b)-C(a)-\lambda(b-a)\} \tag{4}$$

通常 $r(t)$围绕平均速率 λ 进行上下浮动，最大值为 P，P 远大于 λ。网络节点在没有转发缓冲的情况下进行数据转发时，为了保证不出现数据丢失，必须为通信量分配至少相当于峰值速率的带宽。如果使用数据缓冲对具有突发特性的 VBR 视频流进行整形，并且允许引入传输延迟，那么网络节点分配的带宽可以降到小于峰值速率。采用数据缓冲的方法通过主动延迟发送，可以平滑输出速率；缺点是引入了额外延迟，而且需要缓冲资源。并且由于 VBR 流的突发性，使得视频流的不同分段的延迟不同，这就导致了延迟抖动的引入，而延迟抖动在视频应用中是一个对视频质量影响很大的关键指标。在本文提出的算法中，采用了数据缓冲方法来对视频流进行整形，同时解决数据缓冲带来的延迟抖动问题，提高了输出流的 QoS。

最常用的基于缓冲的确定性通信量控制算法是多漏桶算法(Multiple Leaky Bucket MLB)，MLB 中使用最多的是 DLB (Dual Leaky Buckets)模型[3-5]。采用 DLB 进行通信量整形的流程如图 1 所示。

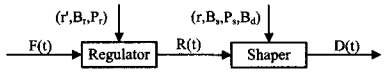

图 1 通信量整形结构图

图 1 中包含一个成型器(regulator)和一个整形器(shaper)，成型器和整形器实质上都是令牌桶整形器。在令牌桶整形器中，令牌以一定速率进入令牌桶，一个令牌表示允许发送一个分组，每发送一个分组，令牌桶中的令牌就减少 1。当令牌桶中的令牌个数为 0 时，停止发送分组。令牌桶的优点是：当主机空闲时，可积累令牌，积累的令牌数量由令牌桶容量来决定。在图 1 中，$F(t)$是源的输入速率，$R(t)$是成型器的输出速率，$D(t)$是速率整形器输出速率，r 是令牌的输入速率。B_r 和 B_s 分别是成型器和整形器的令牌(token)缓冲区，令牌缓冲区的充满程度可以用来反映输出通信量的突发度，P_r 和 P_s 分别是成型器和整形器设置的最大允许的输出速率，即峰值速率，DLB 的含义是指输出速率同时受令牌缓冲区和峰值速率的双重限制。B_d 是整形器用于存储转发时所用的数据缓冲，整形器的最大可允许时延 D_s 与 B_d 的大小有关。

成型器用于描述和限制输入流，只有符合成型器模型($r'(t)$，B_r，P_r)的流才会被允许通过，它主要用于通信量成型和网络的接纳控制。整形器用于对输入的流进行通信量控制，根据设定的服务参数控制输出流的速率，主要的目的是平滑输入流的速率，同时保证数据无丢失和最大引入延迟等限制条件。成型器和整形器都采用 DLB 模型，区别在于成型器中没有数据缓冲，因此成型器会造成输入数据丢失。而由于整形器引入了数据缓冲，因此在输出流中引入了延迟。在数据缓冲中的数据包的延迟是由数据缓冲的充满程度和令牌输出速率之间的关系决定的。通常的 DLB 模型并没有考虑数据缓冲的充满程度和令牌输出速率之间的关系，所以引入的延迟变化会比较大，最大延迟的情况为数据缓冲中充满了数据，

而令牌输出速率为 r 的情况；$D_s=B_d/r$；最小延迟的情况为数据缓冲中没有数据，而令牌输出速率为 P_s 最大的情况；这时延迟为 0；这种延迟变化就是延迟抖动。

3 基于抖动约束的整形算法(JCS 算法)

本文的主要研究对象是针对 VBR 流的，VBR 流具有明显的突发性，并且在长期的过程中，平均速率会产生波动，对于这样的源通信量无法使用单一的 DLB 模型进行控制，而且 DLB 模型的整形算法会引入较大的延迟抖动；由于视频流对于延迟抖动的敏感性，因此传统的 DLB 整形模型不能满足 VBR 视频流的应用情况。本文提出了一种基于抖动约束的整形算法(Jitter-constrained Shaping，简称 JCS 算法)，JCS 算法的目标是研究在有限资源下进行速率整形时应该如何设计速率调整策略才能满足延迟抖动的约束条件，同时又能达到平滑数据流的效果和数据无丢失等限定条件。

JCS 算法的主要思想是利用数据缓冲区的充满程度和令牌输出速率的关系，进而估计出当前缓冲区中数据的缓冲延迟，然后通过指数加权平均来得出分组在经过缓冲后的平均延迟，根据当前分组的延迟和平均延迟的差值计算出分组的延迟抖动值。然后根据数据缓冲长度的变化和抖动值的变化对令牌速率进行调整，使分组在数据缓冲中引入的延迟尽量平均，抖动值尽量小。令牌速率的调整方向一般适应输入速率的变化趋向，这样可以在平滑 VBR 流的情况下，降低分组的延迟抖动。调整后的令牌输入速率反映了该时段的平均速率，可以用来计算有效带宽。为了防止数据缓冲区溢出，造成数据丢失，还要注意控制数据缓冲的充满程度。在这一节中，首先给出 JCS 算法的理论基础，然后给出其具体实现。

对于一个令牌桶的整形器，假定 t 时刻的输入通信量的速率是 $F(t)$，输出通信量的速率是 $D(t)$，令牌缓冲区长度为 B_s，数据缓冲区长度为 B_d，令牌填充速度为 $r(t)$，最大输出速率为 $P_s(P_s>r(t))$。令牌桶整形器的工作方式是：初始时，令牌缓冲区为空，数据缓冲区为空。随后令牌以 $r(t)$ 速率进入令牌缓冲区，令牌缓冲区满时，后续的令牌丢失。通信量以 $R(t)$ 进入数据缓冲区，数据缓冲区和令牌缓冲区都是非空时进行分组输出，每输出一段分组，必须对等地从令牌缓冲区中移走具有相同比特的令牌，从缓冲区中移走数据的速率是 P_s。

令牌桶整形器的实际输出速率 $D(t)$ 被数据缓冲区和令牌缓冲区的充满程度以及移走数据的速率所共同决定。当数据缓冲和令牌缓冲都为空时，要求两个缓冲区都到达一个分组后，才能输出下一个分组，所以 $D(t)$ 由数据的输入速率和令牌输入速率的最小值决定，即 $D(t)=\min\{R(t),r(t)\}$；当数据缓冲和令牌缓冲都不为空时，分组输出不需要等待缓冲区的输入，$D(t)$ 由移走分组的最大速率决定，即 $D(t)=P_s$；数据缓冲不为空，而令牌缓冲为空时，分组需要等待令牌缓冲区中输入足够令牌才可以以速率 P_s 进行输出，根据 $P_s>r(t)$ 的约定，可知 $D(t)=r(t)$；令牌缓冲不为空，而数据缓冲为空时，只要输入一个分组，就可以按照 P_s 速率进行输出，则 $D(t)=\min\{R(t),P_s\}$。所以对于带有数据缓冲区的令牌桶模型，速率整形器的输出速率 $D(t)$ 有以下公式：

$$
\begin{aligned}
D(t) &= \min\{R(t),r(t)\}; && L_t=0, B_t=0;\\
&= \min\{R(t),P_s\}; && L_t=0, B_t>0;\\
&= r(t); && L_t>0, B_t=0;\\
&= P_s; && L_t>0, B_t>0; \qquad (5)
\end{aligned}
$$

这里 L_t 表示数据缓冲的长度，B_t 表示令牌缓冲的长度；由于引入了缓冲，经过令牌桶整形器输出的分组比未整形输出的分组增加了端到端的传输延迟，增加的延迟 $delay$ 由分组进入缓冲区时，缓冲区内数据的长度 L_t 和分组被输出之前缓冲区的输出速率 $D(t)$ 所决定。显然：

$$delay = L_t/D(t) \qquad (6)$$

在数据缓冲区不为空时，由式(5)知 $\min\{D(t)\}=r(t)$，由 B_d 是数据缓冲区长度知，最大延迟 $D_s=B_d/r(t)$。由于 VBR 流的突发特性，在引入延迟的同时也引入了延迟抖动。延迟抖动是通过当前的延迟和平均延迟的差值来计算的，在 JCS 算法中平均延迟采用指数加权平均的方法：

$$delay\_ave = delay\_ave*(1-W_d) + delay*W_d \qquad (7)$$

其中 W_d 为可调整的权值。则源通信量分组在数据缓冲整形过程中引入的抖动值通过下式可得：

$$jitter = delay - delay\_ave \qquad (8)$$

由式(5)可知，在正常工作状态下如果保持数据缓冲区大于 0 和令牌缓冲区充满程度等于 0 的状态，可以产生恒定速率为 $r(t)$ 的流，因此整形器的最佳工作状态应该是数据缓冲区非空和令牌缓冲区为空的情况。在这种情况下，如果输入流的速率大于令牌输出的速率，那么势必造成数据缓冲中数据的增加，而导致缓冲中的分组延迟加大；反之，如果输入流的速率小于令牌输出的速率，造成延迟减小。这种延迟的变化正是抖动产生的原因。而由于 VBR 流的突发性，这种变化是其固有的特性。为了控制分组的延迟抖动，就要控制缓冲区长度和令牌输出速率，使两者相匹配。

```
Initialization: delay=0; ave_delay=0;
Forever :
    if (L_{n-1} == L_n) then
        do not adjust parameters;
        delay = L_n/r(t);
        delay_ave = delay_ave*(1-W_d) + delay*W_d;
        break;
    else if (L_n > L_{n-1})
        check token bucket;
        if (B_n > 0)
            do not adjust parameters;
            delay = L_n/p_s;
            delay_ave = delay_ave*(1-W_d) + delay*W_d;
            break;
        else
            delay = L_n/r(t);
            delay_ave = delay_ave*(1-W_d) + delay*W_d;
            jitter = delay - delay_ave;
            if (| jitter | > threshold)
                adjust parameters;
                r_n = ( L_n/L_{n-1})*r_{n-1};
                break;
            else
                do not adjust parameters;
            endif
        endif
    else if (L_n < L_{n-1})
        check token bucket;
        if (B_n > 0)
            adjust parameters;
            delay = L_n/p_s;
            delay_ave = delay_ave*(1-W_d) + delay*W_d;
            r_n = r_{n-1} - ( 1 - L_n/L_{n-1})*p_s;
        else
            delay = L_n/r(t);
            delay_ave = delay_ave*(1-W_d) + delay*W_d;
            jitter = delay - delay_ave;
            if ( jitter < -threshold)
                adjust parameters;
                r_n = ( L_n/L_{n-1})*r_{n-1};
                break;
            else
                do not adjust parameters;
            endif
        endif
    endif
```

图 2 JCS 算法的具体实现

定义整形器的结构是 $(r(t),B_s,P_s,B_d)$，$r(t)$ 是令牌进入令牌缓冲的速率，B_s 是令牌缓冲的最大长度，规定了整形器输出的最大突发度，P_s 是缓冲区输出的最大允许速率，B_d 是

数据缓冲区的最大长度，我们使用(r_t, B_t, P_t, L_t)来表示整形器在 t 时刻的工作状态，B_t 表示令牌缓冲的充满程度，L_t 表示数据缓冲的充满程度。r_t 表示令牌的输入速率，P_t 表示输出流的峰值速率，一般大于最大输入速率。r_t 在工作过程中可以进行动态的调节。JCS 算法的具体实现如图 2 所示。

在算法中每隔一个时间片 Δt 进行一次计算。每次计算出的延迟通过指数加权平均来计算平均延迟，然后通过延迟和平均延迟的差值计算出数据分组的抖动。在算法实现过程中，采用 L_n 来表示第 n 次检查时刻数据缓冲 L_t 的状态；用 r_n 来表示第 n 次检查时刻令牌的输入速率 r_t；用 B_n 来表示第 n 次检查时刻令牌缓冲的状态 B_t。算法执行过程如下：

每次首先检查数据缓冲的充满程度 L_n 和令牌缓冲中的状态 B_n；如果数据缓冲的充满程度 $L_n = L_{n-1}$；表明数据缓冲的长度没有变，说明现在流的速率比较平滑，数据缓冲中数据分组的延迟相同，没有抖动产生，所以令牌的速率不需要调整；这时，如果 $B_n > 0$，则延迟通过 $delay = L_n/p_s$ 来计算，如果 $B_n = 0$；则延迟通过 $delay = L_n/r_n$ 来计算。

如果 $L_n > L_{n-1}$，表明数据缓冲的充满程度比上一个时间片增加。为了防止由于短期突发引起判断失误，这时再对令牌缓冲 B_n 检查，如果 B_n 不为 0，则表明在上一个时间段输入数据小于以 r_n 为平均输出速率的输出，因此数据缓冲区中的过多分组是由于短期突发引起，不应该进行调整；这时通过 $delay = L_n/p_s$ 计算延迟和平均延迟。如果 $B_n = 0$，说明是令牌输入速率过小，导致数据缓冲中的分组过多，延迟增大。这时计算延迟和抖动：$delay = L_n/r_n$；$jitter = delay - delay\_ave$；如果 $jitter$ 满足要求则暂不做调整；如果 $jitter$ 不满足要求，说明延迟超过门限值；要对速率进行调整，调整的准则是：使抖动尽量为 0，即延迟保持不变，$L_n/r_n - delay\_ave = 0$。为了能尽快地将流进行平滑，这里采取 L_{n-1}/r_{n-1} 来代替 $delay\_ave$ 即：

$$L_n/r_n - L_{n-1}/r_{n-1} = 0 \Rightarrow r_n = L_n/L_{n-1} \times r_{n-1} \quad (9)$$

所以应增加的速率 Δr 为：

$$\Delta r = (L_n/L_{n-1} - 1) \times r_{n-1} \quad (10)$$

如果 $L_n < L_{n-1}$，表明数据缓冲的充满程度比上一个时间片减小，检查令牌缓冲 B_n，如果 B_n 不为 0，则表明令牌输出速率过大，导致数据缓冲中的分组延迟减小；这时计算延迟 $delay = L_n/p_s$；$jitter = delay - delay\_ave$；这个时候肯定对令牌速率进行调整：调整的值为：$\Delta r = (1 - L_n/L_{n-1}) \times p_s$。

如果 $B_n = 0$，这时计算延迟和抖动：$delay = L_n/r_n$；$jitter = delay - delay\_ave$；如果 $jitter$ 满足要求则暂不做调整；如果 $jitter$ 不满足要求，说明延迟过小，超过门限值；要对速率进行调整，调整的值为：$\Delta r = (1 - L_n/L_{n-1}) \times r_n$。

现在来分析 JCS 算法在满足抖动约束的情况下，是否保持了数据的无丢失特性。当 $L_n > L_{n-1}$；数据缓冲的充满程度比上一个时间片增加，B_n 不为 0 时，根据前面的分析，在随后的一段时间里 L_n 不会增加，因而不会出现数据上溢。如果 B_n 为 0，则令牌输入速率调整幅度随着 L_n 的增大而增大，当 L_n 接近 B_L 时，r_n 逼近 p_s，因为数据输入的瞬时速率小于 p_s，所以缓冲区的有效数据长度不会再增长，缓冲区不会产生上溢，即满足数据无丢失的要求。

4 实验结果与分析

为了验证 JCS 算法对 VBR 流的平滑效果和经过整形后的抖动情况，在试验中采用 3 种过滤器对源数据流进行通信量整形，然后对比产生的通信速率图表和统计信息，比较最后的平滑效果。3 种过滤器分别是：第 1 种，不进行任何整形，缓冲区大小为 1 个包的最大长度，缓冲区内数据以最大速率 10Mbps 的速率输出；第 2 种，基于 DLB 模型的速率整形器，采用确定的峰值速率和令牌速率进行控制；第 3 种，采用 JCS 算法的整形器，采用在线估计的峰值速率和自适应的令牌速率对缓冲区的输出进行控制。

4.1 速率波动的比较

实验条件：源数据流是一段 MPEG2 的 VBR 视频流，平均速率约为 6.7 Mbps，连续突发数据可达 500kbps，对其进行 3 种整形器的过滤，各持续 10 分钟，每秒量度一次输出速率。初始时刻从接收到第 1 个数据包开始，初始的整形器参数定义来源于编码器的输出信息。图 3－图 5 是每种整形器在实验中持续 10 分钟的瞬时输出速率变化图，横坐标是时间，基本单位：秒，纵坐标是输出速率，基本单位：Mbps。

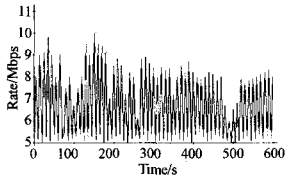

图 3 未整形的输出流的速率变化图

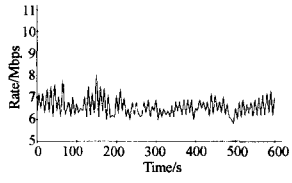

图 4 经过 DLB 整形算法整形的流

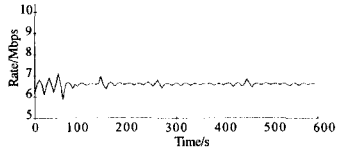

图 5 经过 JCS 整形算法的输出流

表 1 三种情况下输出流的比较

类型	平均速率/Mbps	最大速率/Mbps	最小速率/Mbps
未整形流	6.584	10.06	5.0
DLB 算法	6.81	7.8	5.85
JCS 算法	6.68	7.15	5.92

第 1 种情况是未整形的输出流，速率变化如图 3 所示；从图 3 可以看出，原始数据流的速率最大值可以达到 10M 左右，最小值在 5M 左右，而且速率变化较快，速率抖动明显；第 2 种情况是经过 DLB 整形后的输出流，整形器参数为(6.7 Mbps，500kB，10Mbps)，速率变化图如图 4 所示。从图中可以看出，速率最大值在 8M 左右，最小值接近 6M，相比于原始数据流，整形后的流平滑了很多。第 3 种情况是 JCS 算法的输出流，初始参数为 (6.7Mbps，500KB，10Mbps)，速率变化图如图 5 所示；从图 5 可知，最大速率为 7M 多，最小值为 6M 左右，而且速率变化多发生在整形刚开始的时候，在经过一段时间之后，输出流已经变得相当平滑。表 1 中记录了 3 种情况的下的平均速率、最大速率、最小速率的统计值。比较 3 种输出的通信量图和统计信息可知：3 个输出流的平均速率基本相同，相差不到 0.02%；速率波动可以采用速率标准差和最大、最小值量度，由表 1 可知，未加整形的原始流输出的速率波动最大，采用 DLB 整形后的流输出速率波动较小，而采用 JCS 算法的速率波动最小，而且经过 JCS 算法整形后的输

出速率最接近源数据流的平均速率。

4.2 延迟抖动比较

在实验中分别记录了 DLB 整形算法和 JCS 算法的平均延迟和最大延迟，最小延迟以及平均延迟；统计信息如表 2 所示。延迟方差反映了整个过程中延迟的变化波动情况，即反映了抖动的大小。从表中可以得知，DLB 算法和 JCS 算法之间的平均延迟虽然相差不是很大，DLB 整形算法为 4.21ms，而 JCS 算法为 3.18ms，但是它们之间的延迟方差相差却很大，DLB 算法的延迟方差达到了 271.4，而 JCS 算法的延迟方差只有 3.62，这充分说明了 JCS 算法的输出流的延迟比较恒定，也就是说抖动值较小。为了能直观地比较两种算法的延迟抖动，这里给出了试验过程中的抖动值变化，这里的抖动值是利用在算法过程中 jitter = delay－delay_ave 的公式计算的。从抖动变化曲线（如图 6 所示）可以看出，DLB 算法的抖动变化比较大，大约在－3ms 到 3ms 之间，而 JCS 算法的抖动值大约在－0.8 到 0.8 之间。

表 2 DLB 算法和 JCS 算法的延迟数据比较

类型	平均延迟/ms	最大延迟/ms	最小延迟/ms	延迟方差
DLB 算法	4.21	7.2	0.85	271.4
JCS 算法	3.18	3.6	2.72	3.62

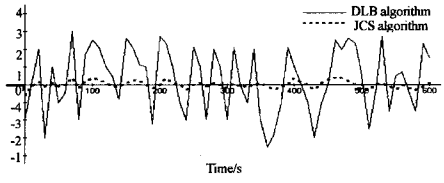

图 6 DLB 算法和 JCS 算法的抖动变化曲线图

从以上试验结果可以看出，JCS 算法不仅能起到平滑突发数据流的作用，而且可以有效降低整形过程中数据缓冲引入的延迟抖动，这样不仅降低了网络的资源利用率，而且对 VBR 流这样的视频应用的服务质量有很大的提高。对于视频流在 Internet 上提高端到端的 QoS 的研究有很大的帮助。

结束语 本文提出了一种基于抖动约束的整形算法，与传统的 DLB 算法不同，JCS 算法根据网络结点输入缓冲区充满程度的变化来动态调整令牌的输入速率，从而平滑输出流的速率和控制输出流的延迟抖动。经实验验证，JCS 算法可以有效地降低 VBR 视频流在整形过程中的延迟抖动，提高 VBR 视频流的服务质量，可以有效地提高网络节点对于 VBR 这样的视频流的支持。

参 考 文 献

[1] Eichler G, Riccito F. AQUILA Network Services [C]// Proceedings of COMCON8. Crete, GREECE, June 2001

[2] Fernando , Moreiral , Ruela J . A method for traffic scheduling based on token bucket QoS parameters[C]// The 3rd Conf. on Telecommunication. Porto, Portugal, 2001

[3] Elwalid A, Mitra D. Traffic shaping at a network node: theory, optimum design, admission control [C]// Proceedings of IEEE INFOCOM. 1997: 445-455

[4] Graf M. VBR video over ATM: Reducing network resource requirements through end system traffic shaping[C]// Proceedings of IEEE INFOCOM. 1997: 48-57

[5] Bechler M, Riter H, Schafer G. Shaping in end systems attached to QoS-supporting networks [C]// IEEE Symposium on Computers and Communications Proceedings. 2001: 296-301

[6] Verma D C, Zhang H. Delay Jitter Control for Real-Time Communication in a Packet Switching Network[C]// Proceedings of TriComm'91

[7] Alam M, Atiquzzaman M, Karim M. Shaping for MPEG video transmission over the next generation Internet [J]. Computer Communications, 2000, 23(14): 1336-1348

[8] Alam M, Atiquzzaman M, Karim M. Efficient MPEG video traffic shaping for the next generation Internet[C]// IEEE Global Telecommunications Conference v 1 (A) 1999. IEEE, Piscataway, NJ, USA, 1999: 364-368

[9] Ferrari D. Client Requirements for Real-Time Communication [J]. IEEE Communications Magazine, 1990, 28(11): 65-72

[10] Naser H, Leon-Gracia A. Performance evaluation of MPEG2 video using guaranteed service over IP-ATM networks[C]// Multimedia Computing and Systems Conference. 1998

（上接第 74 页）

[10] Wan H , Ishikawa N , Autonomous J H . Topology Optimization for Unstructured Peer-to-Peer Networks // 11th International Conference on Parallel and Distributed Systems (ICPADS'05). July 2005: 488-494

[11] Darlagiannis V, Mauthe A, Steinmetz R. Overlay Design Mechanisms for Heterogeneous, Large-Scale, Dynamic P2P Systems. Journal of Network and Systems Management, 2004, 12(3): 371-395

[12] Crespo A , Garcia - Molina H . Semantic Overlay Networks for P2P Systems. Technical Report. Stanford University, January 2003

[13] Goh K-I, Oh E, Kahng B, et al. Betweeness centrality correlation in social networks. Phys. Rev. E, 2003(67): 017101

[14] Saroiu S, Gummadi P, Gribble S. A Measurement Study of P2P File Sharing Systems. Technical Report UW-CSE-01-06-02. July 2001

[15] Chu J, Labonte K, Levine B N. Availability and Locality Measurements of Peer-to-Peer File Systems // ITCom: Scalability and Traffic Control in IP Networks II Conferences. July 2002

[16] Ripeanu M , Foster I . Mapping the Gnutella Network : Macroscopic Properties of Large-Scale Peer-to-Peer Systems. Lecture Notes in Computer Science, Springer-Verlag GmbH, 2002, 2429: 85-93

[17] Bustemante F E, Qiao Y. Friendships that Last: Peer Lifespan and its Role in P2P Protocols // Intl. Workshop on Web Caching and Distribution. September 2003

[18] Stutzbach D, Rejaie R, Sen S. Characterizing Unstructured Overlay Topologies in Modern P2P File-Sharing Systems // Proceedings of the ACM SIGCOMM Internet Measurement Conference. New Orleans, October 2005. Technical Report CIS-TR-05-01. University of Oregon, June 2005

[19] http://peersim. sourceforge. net/

[20] Newman M E J. A measure of betweenness centrality based on random walks. http://aps. arxiv. org/abs/cond-mat/0309045/