

doi:10.3969/j.issn.1007-7162.2013.04.015

H. 264 视频码流自适应传输的研究与实现

刘国英, 章云, 陈泓屹

(广东工业大学 自动化学院, 广东 广州 510006)

摘要: 针对 IP 网络带宽的有限、时变和“尽力而为”的特性, 为了保障视频传输的服务质量(QoS), 提出了一种新的 GAIMD_Buffer 自适应传输方案. 该方案在 GAIMD 算法的基础上增加了服务端和客户端的 H. 264 视频流缓冲控制策略. 当网络处于稳定或拥塞状态时, 若下一个 RTT 时刻缓冲区占有量的估计值位于设定的阈值内, 则按照 GAIMD 算法来调整发送速率, 否则通过调整视频的编码码率来保证缓冲区不会发生上溢或下溢. 实验表明该方案在嵌入式系统中能够更有效地平滑速率抖动, 减少数据丢失, 降低播放延时, 进一步提高视频播放质量.

关键词: H. 264 视频; 服务质量(QoS); 缓冲控制; 自适应

中图分类号: TN919.82

文献标志码: A

文章编号: 1007-7162(2013)04-0083-05

Research on and Implementation of Adaptive Transmission of H. 264 Video Stream

Liu Guo-ying, Zhang Yun, Chen Hong-qi

(School of Automation, Guangdong University of Technology, Guangzhou 510006, China)

Abstract: Taking into account that the network bandwidth has limitation, instability and "making the best" service, it proposes a novel adaptive transmission scheme named GAIMD_Buffer to ensure QoS of video transmission. Based on GAIMD algorithm, the scheme introduces a buffer control strategy of H. 264 video streaming for both servers and clients. When the network is in a stable or congested state, and the estimated value of the buffer occupancy is within the given threshold in the next RTT as well, the speed of transmission is adjusted by GAIMD algorithm; otherwise, the stability of the buffer can only be ensured by adjusting the video code rate, which ensures the buffer will not overflow or underflow. Experimental results demonstrate that the proposed scheme works well in the embedded system, which shows more effective smooth jitter, less packet loss and play latency, and improves video quality accordingly.

Key words: H. 264 video; quality of service(QoS); buffer-driven; adaptive

随着互联网和视频压缩技术的高速发展, 视频传输成为当前网络应用的一个热点. 视频传输需要服务质量(QoS)如丢包、抖动、延时和带宽的保证. 然而互联网本身不提供服务质量保证, 例如网络拥塞会造成数据丢失和过度的延时, 带宽的有限和时变等特征会影响到接收端视频的播放效果, 因此需要一种自适应传输策略来实现拥塞控制.

目前基于终端的自适应传输策略主要是基于速率的拥塞控制, 例如常用的 AIMD 算法和 TFRC 算

法. AIMD 算法根据线性增加乘性减小的简单方法来调整发送速率, 但容易造成发送速率的波动. 在文献[1]中提出的改进 AIMD 算法使用了丢包率和延迟抖动作为网络状况的评估参数, 通过延迟抖动预测出网络拥塞状况的变化趋势, 在一定程度上平滑了发送速率的波动. TFRC 算法根据 TCP 吞吐量公式来调整发送速率, 保证了实时流相对于 TCP 流的友好性, 但缺乏对传输延迟和快速响应方面的控制. 在文献[2]中提出的 TFRC-CJUTD 算法使用了丢包

收稿日期: 2012-12-15

基金项目: 中国教育部博士基金项目(2012442013000)

作者简介: 刘国英(1987-), 男, 硕士研究生, 主要研究方向为音视频编解码与传输技术.

率和传输延迟抖动作为网络状况的评估参数,降低了实时流单向传输的延时抖动. 在文献[3]中提出的 STFRC 算法通过对往返时间 RTT、重传超时值 RTO 和丢包率 p 使用加权计算,获得了平滑和快速响应之间的自适应调整.

上述自适应传输策略都没有考虑终端缓冲区对视频传输服务质量(QoS)的影响,这点在存储资源有限的嵌入式平台上显得尤为重要. 例如服务器端缓冲区可用来保存由于网络拥塞未能及时发送的数据,防止数据的丢失. 而客户端缓冲区可用来缓存接收到的数据,防止由于网络抖动和延时造成视频播放的不连续. 在文献[4]中提出的 TCP-RA 算法直接使用发送缓冲区中缓存数据的累积程度来估计网络带宽的变化,并对视频编码码率进行自适应调整,该算法避免了网络反馈流量占用额外的网络带宽,但由于缺乏对网络状况的直接检测使得调整效果欠佳. 在文献[5]中提出的基于服务器端和客户端联合控制的多模式自适应缓冲方案,该方案通过在服务器端和客户端采用多级辅助缓冲机制来降低丢包率和误码率,但多级缓冲模式增加了视频的播放延时. 在此基础上,本文结合 H. 264 视频码流的可伸缩编码特征和缓冲技术提出了一种新的 GAIMD_Buffer 自适应传输方案.

1 理论基础及相关工作

1.1 端到端视频传输模型

由于 H. 264 视频流编码码率与视频流发送速率之间并不存在直接的联系,所以在服务器端的视频流在发送到网络上之前必须先缓存到发送缓冲区中. 另外在客户端接收到的视频流必须先缓存到接收缓冲区中,在缓冲区中经过处理后再交给解码器解码. 图 1 显示了端到端视频流传输模型.

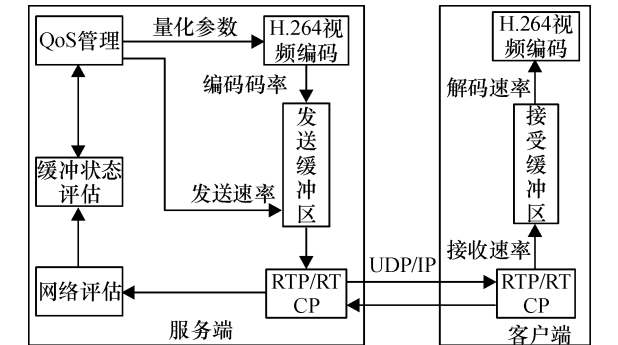


图 1 端到端视频流传输模型^[6]

Fig.1 End-to-end transmission model of video stream^[6]

1.2 缓冲描述

缓冲技术常用于平衡模块的写入和读出快慢不均匀的情况,避免由于读出速度过慢使得写入等待并导致响应速度下降,同样可以避免由于写入速度快慢不均匀导致读出忙闲不一. 缓冲区一般是一个先进先出(FIFO)队列^[7],写入线程负责将数据插入队尾,读出线程负责将数据从队首弹出,同时读写线程之间必须有同步处理机制. 本文使用了一种环形队列缓冲区来平滑网络抖动,且环形队列可以避免动态的内存分配和释放,固定的内存空间可以反复使用,如图 2 所示.

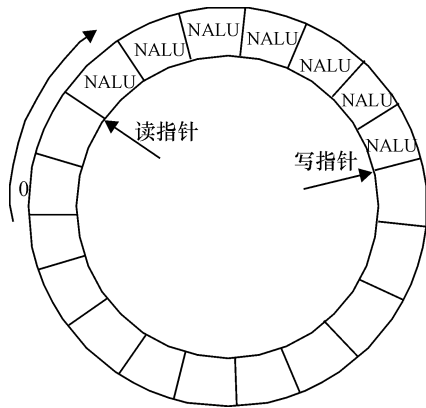


图 2 环形队列缓冲

Fig.2 Circular queue buffer

原始视频数据经过 H. 264 编码器压缩封包后生成 NALU 序列,然后填入到发送缓冲^[8]. $R_e(t)$ 表示 t 时刻 NALU 序列填入缓冲区的速度, $R_e(t)$ 实际上等于视频的当前编码码率. $R_s(t)$ 表示 t 时刻 NALU 序列流出缓冲区的速度, $R_s(t)$ 实际上等于当前网络的发送速率. $A_s(t)$ 表示 t 时刻发送缓冲区的累积填充大小. 三者之间满足式(1)^[9]:

$$A_s(t) = \int_0^t R_s(t) dt - \int_0^t R_e(t) dt. \tag{1}$$

在理想状态下, $t = T$ 时,其中 T 表示结束时间, $A_s(t) = 0$,即 $\int_0^T R_e(t) dt = \int_0^T R_s(t) dt$. 但是由上述分析可知, $R_e(t)$ 取决于当前的编码码率, $R_s(t)$ 取决于当前网络状况,且两者之间并没有直接的联系. 所以如果不在发送端增加控制,当出现 $\int_0^T R_e(t) dt > \int_0^T R_s(t) dt$ 情况时会导致数据丢失. 当出现 $\int_0^T R_e(t) dt < \int_0^T R_s(t) dt$ 情况时会降低网络带宽利用率. 同理,在接收端也需满足 $\int_0^T R_r(t) dt = \int_0^T R_d(t) dt$,其中 $R_r(t)$

为接收速率, $R_d(t)$ 为解码速率.

1.3 GAIMD 算法

为了保证 TCP 友好性, 目前关于拥塞控制的研究大多数是基于 TCP 协议的拥塞控制机制: 在发送端使用一个窗口来控制发送速率, 如果在 RTT 时间内没有丢包发生, 则窗口大小增加 1 个数据包, 否则窗口大小降低到原来的一半. 但该机制会导致发送速率的较大振荡, 因此 Yang 等^[10] 提出了 GAIMD 算法, 该算法使用 α, β 分别表示窗口大小的增加因子和减小系数, 且 $\alpha > 0, 0 < \beta < 1$, GAIMD 算法如下所示:

if (在 $[t + \text{RTT}]$ 内没有丢包)

$\omega(t + \text{RTT}) = \omega(t) + \alpha$;

else

$\omega(t + \text{RTT}) = \beta\omega(t)$;

其中 $\omega(t)$ 表示 t 时刻的窗口大小. Yang 等^[10] 同时推出了 GAIMD 流的吞吐量公式为

$$T_{\alpha, \beta}(p, \text{RTT}, T_0, b) = \frac{1}{\text{RTT} \sqrt{\frac{2b(1-\beta)}{\alpha(1+\beta)} p} + T_0 \min(1, 3\sqrt{\frac{(1-\beta^2)b}{2\alpha} p}) p(1+32p^2)}, \quad (2)$$

其中 RTT 表示往返时间, b 表示两个应答包之间发送的数据包数量, p 表示丢包率, T_0 表示重传超时时间. 如果 α 和 β 满足式 (3)^[10], 则 GAIMD 流满足 TCP 友好性.

$$\alpha = \frac{4(1-\beta^2)}{3}. \quad (3)$$

Yang 等^[11] 指出了 β 越接近 1, 发送速率调整越平滑, 但较大的 β 值可能会降低发送端对网络状况变化的响应速度, 最后推出了最佳值 $(\alpha, \beta) = (0.3125, 0.875)$. 在此基础上, Huang 等^[12] 提出的 ADAIMD 算法能根据网络状况及时调整 (α, β) 值来解决平滑性和响应速率之间的矛盾. 上述算法都是在基于缓冲区足够理想的前提下, 但在资源受限的嵌入式平台上实现视频传输就必须考虑到缓冲区大小对视频传输 QoS 的影响, 所以本文提出了一种新的 GAIMD_Buffer 自适应传输方案.

2 GAIMD_Buffer 方案描述

2.1 缓冲状态评估

$t+1$ 时刻的发送缓冲区占有量 $O(t+1)$ 可以通过 t 时刻的占有量 $O(t)$ 来估计, 如式 (4) 所示^[6]:

$$O(t+1) = O(t) + (R_s(t) - R_e(t)) \times \text{RTT}, \quad (4)$$

其中, $R_e(t)$ 表示 t 时刻的编码码率, $R_s(t)$ 表示 t 时

刻的发送速率. 如果不考虑网络延时的影响, t 时刻接收端的接收速率 $R_e(t)$ 等于发送端的发送速率 $R_s(t)$, 解码速率 $R_d(t)$ 等于发送端的编码速率 $R_e(t)$, 那么在 $t+1$ 时刻的接收缓冲区占有量也可使用式 (4) 来表示. 本文使用式 (4) 可以同时是对发送缓冲区和接收缓冲区的状态进行估计, 降低了实现难度.

2.2 GAIMD_Buffer 自适应传输算法

GAIMD_Buffer 算法把网络状况分为快速启动、稳定调整和拥塞控制 3 个阶段. 启动阶段设置 $(\alpha, \beta) = (1, 0.5)$ 来达到快速启动的目的, 且当第一次检测到拥塞标志 f_{con} 置 1 后进入到稳定调整阶段或拥塞控制阶段. 当网络处于稳定或拥塞状态时, 首先判断下个 RTT 时刻缓冲区占有量的预估值 occ, 若 occ 处于阈值范围内, 则按 GAIMD 方法来调整发送窗口大小, 若 occ 下溢则增大编码码率, 若 occ 上溢则减小编码码率. 具体实现如下:

常量:

$(\alpha, \beta) = (0.3125, 0.875)$: GAIMD 系数;

C_{TOT} : 缓冲区总大小;

$C_{\text{MAX}} = 0.9 \times C_{\text{TOT}}$: 缓冲区上限;

$C_{\text{MIN}} = 0.1 \times C_{\text{TOT}}$: 缓冲区下限;

变量:

ω : 窗口大小;

RTT: 往返时间估计值;

occ: 下一个 RTT 时刻缓冲区占有量估计值;

f_{con} : 拥塞标志;

f_{str} : 启动阶段标志;

初始化:

$\omega = 1$; RTT = 1; $f_{\text{str}} = 1$;

while (等待接收 RR 包) {

设置 f_{con} 变量; 更新 RTT、occ 变量;

快速启动阶段:

if ($f_{\text{con}} = 0$ && $f_{\text{str}} = 1$) {

$\omega = \omega + 1$; continue; }

if ($f_{\text{con}} = 1$ && $f_{\text{str}} = 1$) {

$\omega = \omega/2$;

$f_{\text{str}} = 0$; // 进入下一个阶段

continue; }

稳定调整阶段:

if ($f_{\text{con}} = 0$) {

if (occ < C_{MIN}) 增大编码码率;

elseif (occ > C_{MAX}) 减小编码码率;

else $\omega = \omega + \alpha$; }

拥塞控制阶段:

```
else {  
    if(occ < CMIN) 增大编码码率;  
    elseif(occ > CMAX) 减小编码码率;  
    else ω = ω × β;  
} 退出循环;
```

2.3 相关参数的计算

上述算法的实现需要以下3个参数的计算.

2.3.1 往返时间(RTT)计算

本文使用了RTP/RTCP协议作为视频传输应用层协议,发送端通过RTCP包的时间戳可计算出RTT的采样值,如式(5)所示^[3]:

RTT_{sam} = T_{rec} - T_{sen} - T_{del}, (5)

其中RTT_{sam}表示RTT的采样值,T_{rec}表示发送端当前接收到RR包的时间,T_{sen}表示发送端上次发送SR包的时间;T_{del}表示上次发送SR包后的延迟.为了平滑RTT,本文使用了RTT的加权计算,如式(6)所示^[3,13]:

RTT(k) = ∂ × RTT(k - 1) + (1 - ∂) × RTT_{sam}(k), (6)

其中∂=0.9;RTT(k - 1)表示上次RTT值,RTT(k)和RTT_{sam}(k)分别表示当前RTT值和采样值.

2.3.2 拥塞标志(f_{con})的设置

RFC3550定义了一个字节长的fraction_{lost}表示丢包率.另外又定义了4个字节长的jitter表示延时抖动,为了避免把网络的暂变行为当成拥塞状况,本文把丢包率和延时抖动作为是否发生网络拥塞的判断条件,算法如下:

```
if( fractionlost > 0&& jitter > J)  
    fcon = 1;  
else  
    fcon = 0.
```

其中f_{con}表示拥塞标志,J表示一常量.

2.3.3 缓冲区占有量(occ)的估计

根据式(4)可知,t + RTT时刻缓冲区占有量occ的估计值可以通过当前t时刻缓冲区占有量计算得到,如式(7)所示

occ(t + RTT) = occ(t) + (ω / RTT - R_c(t)) × RTT =
occ(t) + ω - R_c(t) × RTT. (7)

3 仿真与结论

在NS2仿真软件里,本文使用GAIMD_Buffer流和TFRC流做了发送速率平滑性的对比分析,使用

GAIMD_Buffer流和GAIMD流做了接收端视频信噪比(PSNR)的对比分析.仿真拓扑如图3所示.

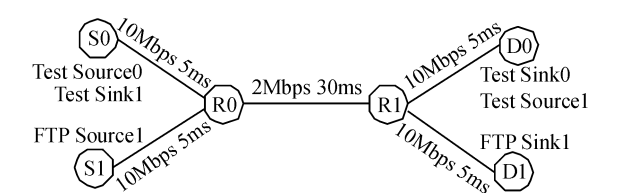


图3 NS2仿真拓扑图
Fig.3 NS2 simulation network topology

图3中R0和R1两个路由器之间的链路带宽为2Mb/s,延时为30ms,队列调度采用RED机制.节点S0至节点D0作为TFRC流、GAIMD流或GAIMD_Buffer流的传输链路.节点S1至节点D1作为TCP流的传输链路,所有的发送端和接收端与路由器之间均以10Mb/s带宽,5ms延时的链路相连接,仿真时间为40s^[14-15].仿真结果如图4和图5所示.

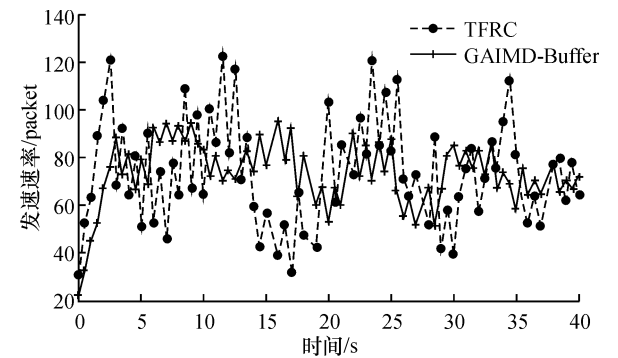


图4 TFRC流和GAIMD_Buffer流速率平滑性对比图
Fig.4 Contrast diagram of rate smoothness for TFRC stream and GAIMD_Buffer stream

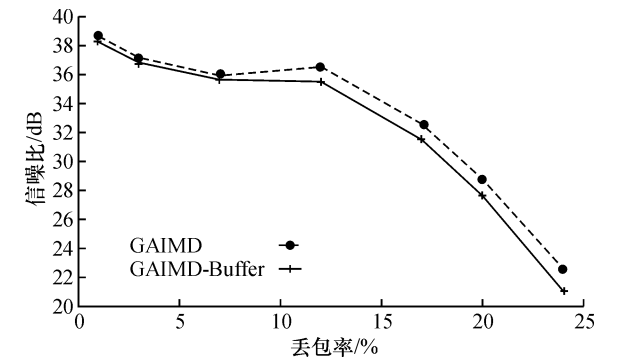


图5 GAIMD流和GAIMD_Buffer流接收端信噪比对比图
Fig.5 Contrast diagram of PSNR for GAIMD stream and GAIMD_Buffer stream in receiving end

图4表明相对TFRC流,GAIMD_Buffer流发送速率的调节更加平滑,图5表明在网络丢包较高时,相对GAIMD流,GAIMD_Buffer流在较高的网络丢

包率时具有更高的峰值信噪比,图像失真更小。

本文提出的 GAIMD_Buffer 视频码流自适应传输方案基于 GAIMD 算法,当网络处于启动阶段时,GAIMD_Buffer 流使用 $(\alpha, \beta) = (1, 0.5)$ 来快速调整发送速率;当网络处于稳定或拥塞阶段时,GAIMD_Buffer 流使用 $(\alpha, \beta) = (0.3125, 0.875)$ 来降低发送速率的波动,实验证明 GAIMD_Buffer 流发送速率的调节更加平滑。同时该方案考虑了终端缓冲区对视频服务质量(QoS)的影响,根据对下一个 RTT 时刻缓冲区占有量的估计来动态调整 H. 264 视频流的编码码率,从而降低数据包的丢失,实验证明 GAIMD_Buffer 流在拥堵的网络环境下具有更高的峰值信噪比,在客户端的视频播放效果更佳。所以 GAIMD_Buffer 视频码流自适应传输方案可以应用于大多数实时流的传输应用中。

参考文献:

- [1] 冷波. 流媒体自适应传输的研究与实现[D]. 武汉:武汉理工大学,2007,41-46.
- [2] 甘泉,薛质. 控制端到端传输延迟抖动的改进 TFRC 算法[J]. 计算机工程,2008,34(10):1-5.
Gan Q, Xue Z. Improved TFRC algorithm aiming at controlling jitters of end-to-end transport delay[J]. Computer Engineering, 2008, 34(10): 1-5.
- [3] Young G K, Jong W K. TCP-friendly internet video with smooth and fast rate adaptation and network-aware error control[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2004, 14(2): 1-13.
- [4] 左冬红. 面向 TCP 的流媒体传输编码码率自适应算法[J]. 中国图像图形学报,2011,16(4):1-7.
Zuo D H. An algorithm for encoding rate adaptive streaming media transmitting based on TCP[J]. Journal of Image and Graphics, 2011, 16(4): 1-7.
- [5] 欧巧凤,邓谦. 无线流媒体系统的多模式自适应缓冲设计[J]. 南昌航空大学学报,2010,24(3):1-6.

- Ou Q F, Deng Q. Design of multi-mode adaptive buffer scheme for wireless stream media system[J]. Journal of NanChang HangKong University, 2010, 24(3): 1-6.
- [6] Zhu H Y, Wang S Y, He K Y. Research on adaptive transmission of H. 264 video stream and QoS guarantee based on SIP[R]. 2012 Fourth International Conference on Computational and Information Sciences, 2012: 41-44.
- [7] Zou Y, Zhao H, Zhao M, et al. Design and application of buffer simulation of realtime streaming media[J]. IEEE, 2009: 97-101.
- [8] 王春. 流媒体传输自适应前向纠错方案设计与实现[D]. 武汉:华中科技大学,2009,4-6.
- [9] 杨格兰,曹理宇,张建明. 基于 MPEG-4 的流媒体服务器端缓冲区管理[J]. 计算机工程,2007,33(5):208-210.
Yang G L, Cao L Y, Zhang J M. Management of server buffer for streaming media system in MPEG-4[J]. Computer Engineering, 2007, 33(5): 208-210.
- [10] Yang Y R, Lam S S. General AIMD congestion control[R]. Technical Report TR-2000-09, 2000: 187-198.
- [11] Yang Y R, Lam S S, Kim M S. Transient behaviors of TCP-friendly congestion control protocols[R]. Technical Report TR-2000-14, 2000: 1716-1725.
- [12] Huang Y H, Xu G W, Huang L J. Adaptive AIMD rate control for smooth multimedia streaming[J]. IEEE, 2003: 171-177.
- [13] 梁少波,卢正鼎. 嵌入式系统中流媒体服务质量研究[J]. 计算机仿真,2007,24(12):1-4.
Liang S B, Lu Z D. Research on stream media Qos of embedded system[J]. Computer Simulation, 2007, 24(12): 1-4.
- [14] Ye D J, Wu Q F, Zhang Z. Receiver-buffer-driven approach to adaptive Internet video streaming[J]. Electronics Letters, 2002, 38(22): 1405-1406.
- [15] 罗治国,王行刚. 一种 MPEG-4 FGS 流媒体自适应传输系统[J]. 计算机工程,2005,31(14):1-3.
Luo Z G, Wang X G. Adaptive transmission system for MPEG-4 FGS streaming media[J]. Computer Engineering, 2005, 31(14): 1-3.