嵌入式无线局域网中H.264视频传输的QoS研究

李文新 李宇光 胡延苏 慕德俊
(西北工业大学自动化学院 西安 710072)

摘 要 就H.264在无线环境中的传输问题展开研究,针对嵌入式特点对现有的视频传输策略进行了改进,在服务器端引入基于反馈和缓冲驱动的发送端动态传输算法,同时在客户端采用了零缓冲+慢启动的动态实时播放算法。这样可以将网络、缓冲和实时性等加以综合考虑,既保证了视频传输的实时性又提供了一定的网络QoS。且经过实验验证,这两种方法都取得了满意的效果。

关键词 嵌入式无线局域网,H.264,网络QoS,动态传输,动态实时播放

中图法分类号 TP393 **文献标识码** A

QoS Research of H.264 Video Transmission in Embedded Wireless LAN

LI Wen-xin LI Yu-guang HU Yan-su MU De-jun
(College of Automation, Northwest Polytechnical University, Xi'an 710072, China)

Abstract H.264 transmission issues in the Wireless LAN were studied and the existing video transmission strategy was improved according to the characteristic of embedded systems. First, a dynamic transmission algorithm based on feedback and buffer-driven was proposed on the server, and then a zero-buffer and slow-start algorithm for dynamic real-time playing was applied on the client. The improved strategy which takes the network, buffer and real-time into account not only ensures the real-time video transmission, but also provides the network QoS. And the experiments show that the two algorithms achieve satisfactory results.

Keywords Embedded wireless LAN, H.264, Internet QoS, Dynamic transmission, Dynamic real-time playing

H.264是两大国际组织ITU(International Telecommunication Union)和ISO/IEC(International Electrotechnical Commission)联合研究并制定的一个面向未来IP和无线环境下的视频压缩标准。H.264采用了一些切实有效的技术方法,具有前所未有的高压缩效率,在相同的图像质量下所需的码流量更低。现有的研究表明,H.264所需码率约为MPEG-2的36%、H.263的51%、MPEG-4的61%,并且随着今后实现优化性工作做得更好,其压缩性能方面的优势将更为突出。

H.264的编码结构在算法概念上分为两层:视频编码层VCL负责高效的数字视频数据压缩,网络提取层NAL负责以网络所要求的恰当的方式对数据进行打包和传送。VCL数据流在传输或存储之前要先映射到NAL单元,NAL接收从VCL传过来的视频压缩码流,并将这些数据打包成NALU(NAL Unit)数据单元,从而使视频码流能够适合在各种基于包交换的网络上进行传输[1]。H.264标准不涉及NALU在具体网络上传输和协议封装细节,只是定义了一种接口,其目的主要是为了把视频压缩和传输在概念上分离,使标准的结构更清晰灵活。

由于TCP协议丢包重传的特性增加了网络的抖动和失真,不符合流式传输对时延的严格要求,应用中一般采用UDP协议来传输H.264视频流。但UDP是面向无连接的传输协议,并不保证可靠传输,因此需要与更高层协议RTP/RTCP结合使用。考虑到嵌入式系统的特点,本文结合控制理论,主要针对发送速率控制和动态回放控制,在兼顾到服务器端和客户端因素的基础上给出了一种自适应流媒体传输机制。

1 发送方自适应拥塞控制

RTP/RTCP在设计之初就被定义为一种反馈信息协议,反馈包中包含的丢包率、延迟和抖动信息等给服务器端提供了一种判断网络状态的依据。服务器端数据采集速率一般是恒定的,而网络状况是时变的,客户端主观视频质量的提升很大程度上得益于服务器对数据发送速率的调控。目前,在实时应用中大多采用基于速率的拥塞控制算法对视频流进行控制[2]:如模仿TCP协议的AIMD方法对实时视频流的速率进行控制[3];D. Sisalem提出DAA(Direct Adjustment Algorithm)算法;实现TCP友好的传输速率增减方法的LDA(Loss-delay Adaptation)算法以及增强LDA算法LDA+[4];根据网络的丢包数和RTT两个状态参数进行类似TCP的速率动态调整的TLFC(TCP-like Control Algorithm)算法等。就现在的研究成果来看,基于公式的拥塞控制方法将会在今后的多媒体实时传输中扮演一个重要的角色。其中有代表性

到稿日期:2010-06-30 返修日期:2011-01-20 本文受国家自然科学基金(60803158)资助。

李文新(1966-),男,硕士生导师,主要研究方向为软件研制和系统重构;**李宇光**(1988-),男,学士,主要研究方向为计算机网络;**胡延苏**(1985-),女,博士生,主要研究方向为计算机网络、网络QoS控制,E-mail:huyansu@gmail.com(通信作者);**慕德俊**(1963-),博士生导师,主要研究方向为计算机网络、信息安全。

的基于公式的拥塞控制方法是 TFRC(TCP, Friendly Rate Control)[5]。但是这些算法一般基于客户机/服务器平台,对于处理能力较弱的嵌入式系统,这些算法无疑复杂度太高。因此,该系统主要采用 UDP 协议进行视频传输,TCP 流量很小,故设计了一种简单的自适应拥塞控制传输方案。

1.1 发送端缓存监控

图 1 所示为系统的结构框图,IP 网络是"尽力而为"服务网络,为了改善 QoS,视频在发送前必须缓存,在理想情况下,发送速率和采集速率平衡,即缓存的帧个数 F 维持在一个恒定值 $Frame\_thred$。但是由于 Linux 的多任务特性,平衡会受到其它任务的影响而被打破。此时如果还按照原来的速率发送会导致 F 不断增加,随即延迟会不断累积,这对实时传输来说是不允许的。所以,必须根据 F 的大小实时调整发送速率。本文给出的传输机制基于控制理论中负反馈模型,提出并实现了一种缓冲区驱动+网络状态监控的自适应拥塞控制传输方案:在网络状态好的情况下,使缓存的队列长度与发送速率成正比,保证缓存区的数据占有量在一个固定的水平上;而在网络状态不好时,可根据 RTCP 的反馈信息判断当前网络的恶化程度,调整其发送速率,防止网络继续恶化。

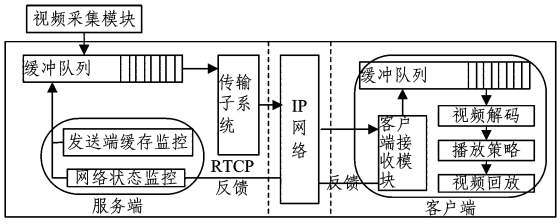

图 1 基于反馈的自适应视频传输框图

具体来说,当 $F>Frame\_thred$ 时,发送速率线性增大,但是不能超过 Max_*rate*;当 $F<Frame\_thred$ 时,发送速率线性减小,但不能低于 Min_*rate*。其中 $Frame\_thred$ 为缓冲阈值,Max_*rate* 为最大发送速率,Min_*rate* 为最小发送速率。最大最小发送速率的选取不能给网络和系统造成负担,同时利用人眼的特性在正常的速率基础上调整,如 25 f/s 的采集设备的发送速率可以选择最低为 20 f/s,最高为 30 f/s。

1.2 网络状态监控

欲在调控算法中加入网络状态影响,首要条件是正确判断网络状态。当网络负载较重而发生拥塞时,往返时间(Round Trip Time,简称 RTT)和丢包率会增大。因此,网络中可用带宽的估计可由端系统通过往返时间或丢包率来间接得到。文献[6]中证明,基于 RTT 反馈拥塞算法的链路吞吐率低于使用基于丢包率的链路。因此,利用丢包率作为判断网络是否拥塞的标准,同时以 RTT 作为辅助。

本文通过给丢包率设置拥塞上限和下限阈值,把网络状态分成 3 个层次:上限 $High\_lost\_thred$ 和下限 Low_lost_thred。对于 H.264 来说,当丢包率小于 5%时会得到很好的效果;当丢包率达 10%,使用 FMO 时视频失真低到需要训练有素的眼睛才能识别。所以上限选取 10%,丢包率下限 $Low\_lost\_threshold$ 选取为 5%。客户端会每隔 5 秒发送一个 RTCP 反馈包,其中包含间隔内的丢包率、平均网络传输延迟等信息。发送端则根据网络丢包率 $Lost$ 对网络状况作估计:当 $Lost<Low\_lost\_threshold$ 时,网络空闲;当 $Lost>Hight\_lost\_threshold$ 时,网络拥塞;介于两者之间时,网络处于满载状态。RTCP 包中的延迟和抖动信息也反映了网络的状况,在 RTCP 的 RR 报告中延迟需要通过 NTP 计算得来,抖动 Jitter 则可以直接读取。

1.3 发送速率自适应

经过网络负载判断后的速率调整算法:当网络空闲时,按照 1.1 节方法调整发送速率;当网络满载时,根据抖动状态加减性调整发送速率;当网络拥塞时,发送速率乘性减小。但最小都不低于 Min_*rate*,最大不超过 Max_*rate*。

具体算法伪代码如下:

if($Lost<=Low\_lost\_thred$)

$$\begin{cases}\text{if}(F<Frame\_thred)\\ \quad R=\{\text{MAX}(r+R_{last},\text{Min\_}rate)\}\\ \text{if}(F>Frame\_thred)\\ \quad R=\{\text{MIN}(r+R_{last},\text{Max\_}rate)\}\\ \text{else}\\ \quad R=R_{normal}\end{cases}$$

if($Low\_lost\_thred<Lost<High\_lost\_thred$)

$$\begin{cases}\text{if}(Jitter>0)\\ \quad R=\{\text{MAX}((C_{last}-\theta)\times R_{last},\text{Min\_}rate)\}\\ \text{if}(Jitter<0)\\ \quad R=\{\text{MIN}((C_{last}+\theta)\times R_{last},\text{Max\_}rate)\}\\ \text{else}\\ \quad R=R_{last}\end{cases}$$

if($Lost>=High\_lost\_thred$)

$$R=\{\text{MAX}(k_{last}^2\times R_{last},\text{Min\_}rate)\}$$

式中,r 为速率步进值;R_{last} 为速率调整前的速率;R_{normal} 为正常速率;k_{last},C_{last} 为调整系数;θ 为系数调整步进值。

2 接收方动态播放算法

客户终端为了增强可能由于网络拥塞而导致回放时发生延迟、抖动或者乱帧的抵抗能力,使得播放画面流畅,通常会在接收端设置一定大小的缓冲区来消除丢包、延迟抖动和乱序。在缓冲区长度一定时,如何根据缓冲区动态监测确定回放速率,是提高播放效果和低延迟的重要因素。

由于网络延迟抖动以及突发数据包的不可预计,还有系统多线程直接的影响等都可能导致接收终端缓存的溢出(即饱和)或枯竭(即饿死),效果上表现为播放中的停顿和跳跃。因此不能等到缓冲区真正清空或塞满时才进行调整,必须提前预警。这就必须实时检测缓冲区,根据缓冲区的占用情况动态调整播放速率。一般的流媒体动态播放算法为:设当前队列中缓冲帧的个数为 K,回放下限为 L,上限为 H,缓冲区的容量为 N,且 $N=L+H$;最小播放速率为 $V_{\min}$,最大播放速率为 $V_{\max}$,标准播放速率为 V_{stan};回放过程中,回放线程实时地监测缓冲队列的长度,根据门限值 L,H 确定绘屏的时间间隔。若 $L\leqslant K\leqslant H$,则采用标准播放速率;若 $K<L$,则播放速率为 $\max\{V_{\max},\max(K,L)*V_{\text{stan}}/L\}$;若 $K>H$,则播放速率为 $\min\{V_{\max},\min(K,L)*V_{\text{stan}}/H\}$。

但上述算法存在两个问题:第一,缓冲队列中 K 达到 L 时才开始播放,即策略要求先进行缓冲再回放,这样会产生播放延迟;第二,由于队列中的缓冲帧个数到达 H 前都是使用正常的回放速率,通常,接收速率和回放速率是相同的,那么在极端情况下,缓冲队列中帧的个数 $K=H$ 将一直维持下去直到打破平衡。此时至少延迟了 H/F 秒,F 为帧率。在这种情况下实时性能很差。

为了解决这个问题,不能简单照搬照抄上面的播放策略,

要结合具体情况加以改进，提高回放实时性。其改进的实时动态播放策略模型如图 2 所示。针对问题一，在播放启动阶段采用了零缓冲＋慢启动的策略[7]，即客户端的回放线程启动后，一旦检测到队列中有可以播放的视频帧便以最低播放速率播放，同时检测队列中帧的个数是否增大(防止特殊情况下得不到缓冲)，如果增大则每播放一帧速率慢慢加大，最终达到正常播放速率。这样可使用户第一时间获得画面信息，提高主观画面的实时性；同时缓冲队列中的帧个数会慢慢地达到 L，仍然可以获得相同缓冲的效果。针对问题二，播放策略中引入网络状态参数，如果网络状况比较好，且 $K>L$，则可以适当加快播放速率，使 K 尽快回落到 L。

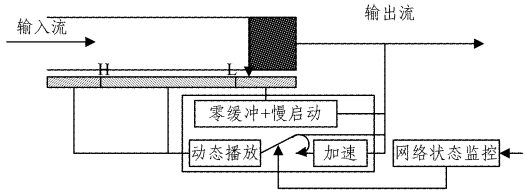

图 2　改进后的实时动态播放策略模型

3　实验验证

实验采用两台真实终端(见图 3)，A 为服务端，B 为客户端，RTP 库每 1s 发送一个反馈数据包。反馈包中的丢包率、延迟、抖动等都是在规定的范围内随机产生，用来模拟网络的时变。实验持续 800s。

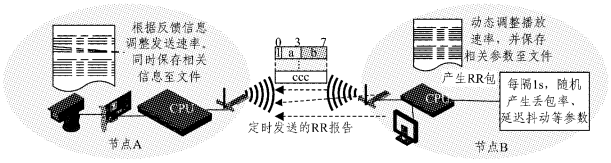

图 3　实验模型

服务器端主要验证：1)网络空闲状态下，即丢包率小于 10%时，发送速率的调整状态，观察此时是否是缓存监控；2)网络满载状态下，即丢包率在 10%～20%之间，发送速率的调整是否以延迟抖动信息为依据；3)在网络拥塞状态下，即丢包率大于 20%，发送速率是否会迅速下降并达到最低发送速率。

客户端主要模拟验证：1)播放的实时性；2)能否根据网络状况调整播放速率，即具备间接感知网络的能力。

服务器端实验验证结果如图 4(a)，(b)所示。以横轴时间作为统一参考点。当网络空闲时，即丢包率在 0～5%之间变化时(图 4(b))，对应图 4(a)中可以看到发送速率始终跟随着缓冲区中缓冲帧个数变化而变化，而没有受到丢包率和延迟抖动的影响，且保证了缓冲队列中视频帧的个数在一个给定的水平上，验证了缓存监控；当网络满载时即丢包率在 5%～10%之间变化时，影响发送速率的主要因素是延迟抖动，由于延迟抖动是随机的，此时可以看到速率和缓冲帧个数不再有很明显的规律；当网络拥塞丢包率在 10%之上变化时，可以看到发送速率迅速降低至最小发送速率，而且缓冲区中帧的个数也迅速增大至一个固定值，这个固定值就是缓冲队列的长度。当网络状态慢慢恢复时，发送速率也随之变化，缓冲区中的视频帧逐渐减少，最后回到空闲状态保持的水平，可见第 1 节描述的发送速率控制算法在图 4(a)，(b)中得到了很好的体现。

接收端采用的零缓冲＋慢启动的动态实时播放算法如图 4(c)所示。可以很明显地看到，速率曲线在开始阶段有一个缓坡，并从零时刻开始直到队列达到缓冲阈值(图中为 25 帧)为止，这就是算法中描述的零缓冲＋慢启动阶段。这个阶段提高了视频播放的主观实时性。当缓冲队列中达到缓冲阈值 $H=L$ 时，调控策略立即调整为缓冲驱动的动态播放策略，播放速度会随着播放队列中视频帧的个数不断的调整。图中两次大的波动对应着发送端发送速率的两次大的调整，可以看出在发送策略图变换剧烈的时刻在接收端也有所体现，并能够进行相应的调整。这也是播放算法间接地感知网络状况的一种体现。播放队列中待播放的视频帧个数绝大部分时间都在一个固定值处震荡。这正是动态播放策略中为了提高实时性而使 $H=L$ 时的体现，这样简单改变就达到了很好的实时效果。

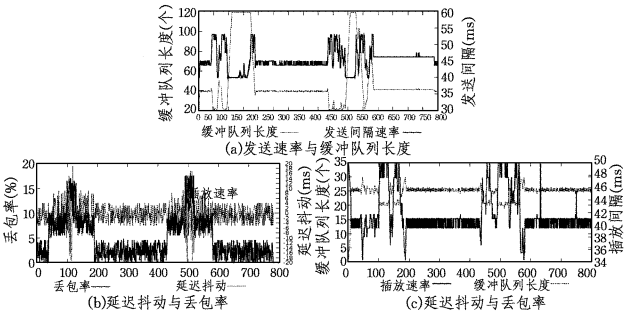

图 4　实验结果图

结束语　本文结合适于无线传输的 H.264 视频压缩标准，对视频通信的关键技术进行了研究，并针对嵌入式系统的特点在数据缓存、服务策略、实时传输与播放和流量控制等方面做了一系列改进，实现了高效的缓存池设计、基于反馈和缓冲驱动的动态算法以及零缓冲＋慢启动的动态播放算法，改善了网络传输的 QoS，并通过实验验证了该方案下视频数据可在嵌入式无线环境中高速高质传输。

参 考 文 献

[1] Stockhammer T, Hannuksela M, Wenger S. H.26L/JVT Coding Network Abstraction Layer and IP-based Transport[C]//IEEE ICIP. Rochester, NewYork, 2002, 2: 485-488

[2] Rejaje R, Handley M, Estrin D, et al. Layered Quality Adaptation for Internet Video Streaming[J]. IEEE Journal of Selected Areas in Communications, 2000, 18(20): 2530-2543

[3] Bansal D, Balakrishnan H. TCP-Friendly Congestion Control for Real-Time Streaming Applications [R]. MIT-LCS-TR-806. MIT, 2000

[4] Sisalem D, Wolisz A. LDA+TCP-Friendly Adaptation: A Measurement and Comparison Study [A]//Proc. Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video(NOSSDAV) [C]. 2000

[5] Floyd S, Handley M, Padhye J, et al. TCP Friendly Rate Control (TFRC): Protocol Specification[S]. Network Working Group RFC 3448, 2003-01

[6] 高旭，沈苏彬，顾冠群. 网络多媒体实时传输协议浅析[J]. 计算机应用研究，2000，9：6-8

[7] 余藜藜，邵晟，朱谦，等. 基于 H.264 编解码的无线视频可靠传输[J]. 复旦学报：自然科学版，2007(1)：108-109