

# 基于 H. 264 的网络流媒体播放控件的设计与实现

吴 杰, 吴 宁, 周 阳, 郭荣辉

(南京航空航天大学 电子信息工程学院, 江苏 南京 210016)

**[摘要]** 网络流媒体播放控件能很大程度地减轻程序开发人员的负担, 编程人员只需简单地设置控件的属性就可轻松完成视频的实时播放。基于 x264 解码代码, 在 Visual Studio 2008 平台下, 考虑实际网络状况以及代码复杂性问题, 利用双缓冲循环队列、管道通信等技术, 构建了一个完整的 H. 264 流媒体播放路径, 整合成一个参数可配置、功能丰富的流媒体播放控件。该控件结构清晰, 配置灵活, 功能丰富, 使用方便。

**[关键词]** 控件, 流媒体, H. 264, x264

**[中图分类号]** TP316.5 **[文献标志码]** A **[文章编号]** 1672-4292(2011)04-0053-04

## Design and Realization of a Streaming Media Playback Component Based on H. 264

Wu Jie, Wu Ning, Zhou Yang, Guo Ronghui

(College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

**Abstract:** A network streaming media playback component could reduce the burden of programmers a lot. Programmers could get the real time playback just through some simple settings. In this paper, to reduce the network jitter and code complexity, a complete playback path is built on the basis of x264 for the H. 264 streaming media by using double buffering circular queue and pipeline, and then integrated into a streaming media playback component in Visual Studio 2008 platform. The component's structure is clear and configuration options are flexible. It is simple to handle.

**Key words:** component, streaming media, H. 264, x264

随着流媒体播放、视频监控等应用程序的开发越来越普遍, 网络流媒体播放控件的设计与实现就显得日趋重要。控件是用户可与之交互以输入或操作数据的对象。使用控件可以把可重复使用的用户接口和行为集成到一个功能完备的包中, 以便在同一应用程序或多个应用程序中重复使用。H. 264 视频编码标准在编码质量和压缩比上比原有的视频编码标准都有了明显的提高。在相同的视觉感知质量上, 编码效率比 H. 263、MPEG-2 和 MPEG-4 提高了 50% 左右。H. 264 不仅具有优异的压缩性能, 且具有良好的网络亲和性<sup>[1]</sup>。因此, H. 264 被普遍认为是具有影响力的流媒体视频压缩标准。

在网络上传输音/视频等多媒体信息目前主要有下载和流式传输两种方案。下载的主要缺点是必须等全部内容传输完毕, 然后才能在本地的机器打开。而采用流式传输方案, 声音、影像或动画等数据由音/视频服务器向用户计算机连续、实时地传送, 用户不必等到整个文件全部下载完毕, 只需经过几秒或十几秒的启动延时即可进行观看。文献[2]用 DirectShow 技术实现了流媒体播放控件, 该方法 filter 之间接口复杂, 且解码部分通常使用的是二进制程序, 不能对解码的细节进行配置。

本文将控件<sup>[3]</sup>和 H. 264 两种技术相结合, 基于 x264 解码器源码, 设计并实现了网络流媒体播放控件, 使得流媒体播放、视频监控等应用程序的开发迅速, 配置多样化, 应用方便。

## 1 系统框图

多媒体流式播放包含 3 个方面的技术: 服务器与客户端的通信技术, 包括多媒体数据的传输、控制命

收稿日期: 2011-08-20.

通讯联系人: 吴 宁 教授, 博士生导师, 研究方向: 数字系统理论与技术、系统的自动测量、控制与故障检测等. E-mail: wunee@nuaa.edu.cn

令等;客户端对接收到的多媒体流实时解码技术;多媒体流实时播放技术.网络通信使用 Windows Socket 技术,多媒体流的解码使用现有的 x264 解码源码,多媒体流的实时播放过程需要进行颜色空间的变换.

流媒体播放控件的工作流程如图 1 所示.网络通信模块通过双缓冲队列给 x264 实时解码模块传递 H.264 编码的视频数据流,经解码模块处理后输出的 YUV 颜色数据通过命名管道传递给 YUV 播放模块.YUV 播放模块将输入的 YUV 数据转换为 RGB 格式的数据,并刷新屏幕.

流媒体播放控件的软件结构如图 2 所示.主线程是 YUV 播放模块,用 C-sharp 语言实现;子线程是通信和解码模块,用 C 语言实现,做成动态链接库.

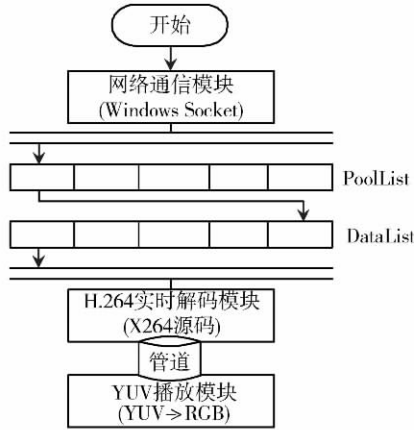


图 1 流媒体播放控件工作流程  
Fig.1 System workflow

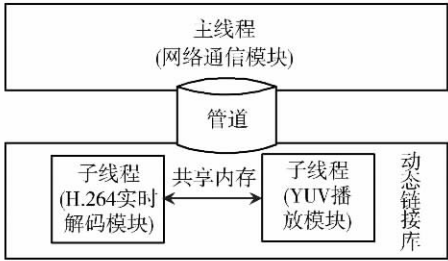


图 2 流媒体播放控件软件结构图  
Fig.2 Software structure

2 网络通信模块

Winsock 控件可以使用两种协议: TCP 协议和 UDP 协议,分别是面向连接的服务和无连接的服务.应用面向连接的服务时,客户端和服务端在进行数据发送前,彼此向对方发送控制分组,使得客户端和服务端都做好分组交换准备.该准备是松散的,面向连接服务与很多其他的服务捆绑在一起,包括可靠的数据传输、流量控制和拥塞控制等,依赖连接以正确的顺序无错地传递所有数据,并使用确认和重传机制实现可靠性.而无连接服务则没有握手过程,当一方想发送数据时就直接发送,没有流量控制和拥塞控制,这样数据传输得更快,非常适合因特网电话、视频会议等应用领域.该控件默认采用 UDP 协议,UDP 端口可配置,并支持配置成采用 TCP 协议.

网络通信模块的主要功能是接收服务器发来的流媒体数据,放入双缓冲循环队列中,以实现数据的接收和向下一级传递数据.采用双缓冲循环队列的理由是:

- (1) 通信模块必须动态地接收数据,并持续地把新数据提供给解码模块,双缓冲循环队列既充分利用内存空间,又能为解码模块提供稳定的数据源.
- (2) 缓冲队列可以稳定码率,有效减小网络延时、阻塞和抖动的影响.

通信模块的工作流程为:建立两个队列,一个是空闲的缓冲队列 PoolList,用以接收存放数据;另一个是尚未处理的数据缓冲队列 DataList,等待解码模块读取.当通信模块使用 Socket 接收到数据后,从 PoolList 队列的头上取出一个缓冲块,存放数据,然后将这个缓冲块加入到 DataList 的尾部,等待解码模块的读取.而在解码模块要求读取时,解码模块会从 DataList 的头上取出一个缓冲块,读取数据,再将读完的缓冲块加到 PoolList 的尾部,等待再一次的数据接收.

通信模块中的主要数据结构如下.

```
typedef struct cqueue {
    sem_t sem_free;
    sem_t sem_valid;
    struct block* front;
    struct block* rear;
    pthread_mutex_t mutex;
    int size;
} cqueue_t;

typedef struct block {
    int size;
    int used;
    struct block* next;
} block_t;

BYTE* buffer;
```

### 3 x264 实时解码模块

H.264 是目前最新的视频编码标准,性能优良,但复杂度高,难以实时应用。x264 是基于 H.264 的一种简化实现,是一种兼容 H.264 标准码流的编码器。与 H.264 参考软件 JM 系列相比, JM 系列与标准更接近,利于学习,其特点是注重实用;而 x264 做了很多的优化,在不明显降低编码性能的前提下,努力降低编码的计算复杂度,摒弃了 H.264 中一些对编码性能贡献微小但计算复杂度极高的新特性,如多参考帧等,在很大程度上提高了实用性。

由于 x264 的开发者至今并没有给出相应的解码器, JM 系列的解码器虽然可以对 x264 编码的图像进行解码,但速度很慢,不能满足实时解码的要求。文献[4-5]在软件平台上设计并实现的用于实时解码的 x264 解码器的解码速度可以达到 JM96 解码器的 6~10 倍,可以很好地满足 x264 的实时解码需求。本文采用 ffmpeg 工程组成员 hust\_xcl 从 ffmpeg 中提取出的 x264 解码器源码。

#### 3.1 H.264 编解码器的功能结构

H.264 标准定义了编解码器的整体结构<sup>[6-7]</sup>,主要分为视频编码层(VCL)和网络提取层(NAL),如图3所示。

自 H.261 以来的所有 ITU-T 和 ISO/IEC JTC1 视频标准, VCL 的设计都遵循基于块的混合视频编码方法,每个编码图像都表现为色度和亮度采样的宏块单元。

NAL 定义了数据封装的格式和统一的网络接口。数据承载在网络提取层单元(NALU)中,有利于数据经打包后在网络中传输。对于面向比特流和面向数据包的传输, NALU 采用统一的数据格式,每个 NALU 包含单个字节的包头信息和多个字节的数据。包头信息包含存储标志和类型标志,存储标志用于指示当前数据不属于被参考的帧,从而便于服务器根据网络的拥塞情况进行丢弃;类型标志用于指示图像数据的类型。

#### 3.2 x264 解码器

由编码器的 NAL 输出一个压缩后的 H.264 压缩比特流,这个压缩比特流送入解码器进行解码工作。和所有的 H.264 解码器一样, x264 沿用了最基本的解码框图,如图4所示。

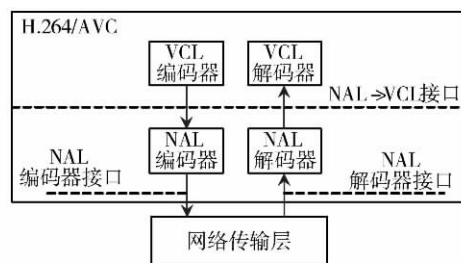


图3 H.264 编解码器结构  
Fig.3 Structure of H.264/AVC video codec

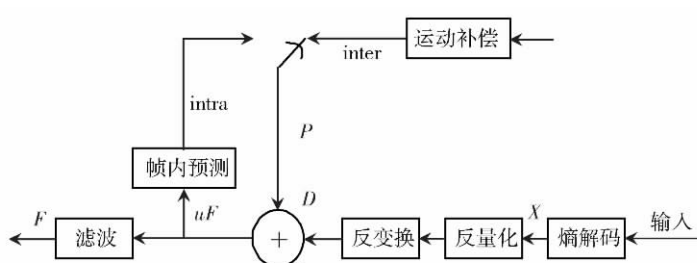


图4 基本解码框图  
Fig.4 Structure of x264 decoder

解码器从 NAL 中接收压缩的比特流,经过对码流进行熵解码获得一系列量化系数  $X$ ; 这些系数经过反量化和反变换得到残差数据  $D$ ; 解码器使用从码流中解码得到的头信息创建一个预测块  $PRED$ ,  $PRED$  与残差数据  $D$  求和得到图像块数据  $uF$ ; 每个  $uF$  通过去方块滤波得到重建图像的解码块  $F$ 。

### 4 YUV 播放模块

H.264 码流解码后,得到大量 YUV 格式的数据,这些数据需要传送到播放模块进行播放。线程间的大数据量传输可以使用共享内存、命名管道两种方式。YUV 播放模块使用 C-sharp 语言实现,解码模块使用 C 语言实现。由于 C-sharp 和 C 之间同步信息的处理机制不同,若使用共享内存的方式,则同步信息的处理会非常复杂。因此本文采用命名管道的方式进行 C-sharp 和 C 之间的数据传输。

YUV 格式的数据通过命名管道技术传输到播放模块。计算机采用 RGB 格式显示彩色图像,需要进行颜色空间的转换。

x264 解码器解码得到的码流为 YUV420 格式,又被称为 YV12 格式。以  $640 \times 480$  大小的一帧图像为

例,每个像素点采样一个  $Y$  值;两个横向与纵向共 4 个相邻像素点采样一次  $U$ 、 $V$  值。所以  $Y$  数组的大小为  $640 \times 480$ ,  $U$ 、 $V$  数组的大小相同,为  $640 \times 480/4$ 。码流的存储格式顺序是  $Y$ 、 $V$  和  $U$ 。

采用 csc.c lib 库函数 `void YV12_to_RGB24(unsigned char * src0, unsigned char * src1, unsigned char * src2, unsigned char * dst_ori, int width, int height)` 将 YV12 转换到 RGB24 转换的公式为:

$$R = Y + (1.4075 * (V - 128)) ,$$

$$G = Y - (0.3455 * (U - 128) - (0.7169 * (V - 128))) ,$$

$$B = Y + (1.7790 * (U - 128)) .$$

转换后的数据直接用来刷新屏幕。

## 5 流媒体播放控件的应用

将本文设计的网络流媒体播放控件应用于无线多媒体传感器网络的视频终端,结果如图 5 所示。

多个多媒体节点采集的视频数据,通过 802.11b 无线多媒体传感器网络,由不同的路径传输到汇聚节点,汇聚节点再把数据传输到需要相应视频信息的视频终端。视频终端可以点播最多 4 个多媒体节点实时回显的视频信息,图中标示着有两个控件在进行流媒体的实时播放。两路视频的播放速度都是 25 帧/s,播放过程清晰流畅。控件左下角显示当前时间,右上角显示多媒体节点的经纬度信息。



图 5 流媒体播放控件的使用

Fig.5 Application of streaming media components

## 6 结语

本文在 x264 解码代码的基础上,构建了一个完整的 H.264 流媒体播放路径,整合成一个参数可配置、功能丰富的流媒体播放控件。为提高该控件的可配置性,增强控件的性能,进一步的工作是对 x264 的解码细节进行研究,提取出关键的参数,整理成可配置的选项,以适应各种不同的应用场合的需求。

## [参考文献](References)

- [1] Joint Video Team( JVT) of ITU-T VCEG and ISO/IEC MPEG. Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification[S]. ITU-T/ISO/IEC 14496-10 AVC. 2005.
- [2] Zhang Fan, Li Bo. DirectShow based internet video on demand system[C]// International Conference on Microwave and Millimeter Wave Technology Proceedings 2008. Nanjing, 2008: 2 077-2 080.
- [3] 彭锋, 林和志, 黄联芬. 基于 Directshow 的 H.264 网络视频监控客户端实现[J]. 现代电子技术, 2011, 34(8): 118-120.  
Pen Feng, Lin Hezhi, Huang Lianfen. Implementation of directshow-based H.264 network video monitoring at client[J]. Modern Electronic Technique, 2011, 34(8): 118-120. (in Chinese)
- [4] 韩峥, 夏志进, 唐昆, 等. x264 解码器的设计与实现[J]. 微计算机信息, 2007, 23(18): 85-86.  
Han Zheng, Xia Zhijin, Tang Kun, et al. The design and realization of x264 decoder[J]. Microcomputer Information, 2007, 23(18): 85-86. (in Chinese)
- [5] 王立青, 李瑞祥, 王延政. 基于 x264 和流媒体的嵌入式视频监控系统[J]. 计算机安全, 2010(7): 13-15.  
Wang Liqing, Li Ruixiang, Wang Yanzheng. A video surveillance system based on X264 and streaming media[J]. Computer Security, 2010(7): 13-15. (in Chinese)
- [6] Wiegand T, Sullivan G J, Bjntegaard G, et al. Overview of the H.264/AVC Video Coding Standard[J]. IEEE Trans Circuits Syst Video Technol, 2003, 13(5): 60-76.
- [7] Deng Hongtao, Zhu Xu, Chen Zheng. An efficient implementation for H.264 decoder[C]// Proceedings-2010 3rd IEEE International Conference on Computer Science and Information Technology. Chengdu, 2010: 41-44.

[责任编辑: 严海琳]