Blackfin 平台上视频服务器①

元艳素[1,2]，孙建伟[2]，李 丹[3]，杨海波[2]，贾军营[1,2]

[1](中国科学院 研究生院，北京 100049)
[2](中国科学院 沈阳计算技术研究所，沈阳 110168)
[3](东北电网有限公司，沈阳 110006)

摘 要：随着网络通信技术和多媒体技术的迅猛发展，嵌入式的视频监控设备需求量也在增加。提出了一种嵌入式的视频服务器解决方案，方案中采用高压缩比、高鲁棒性的 H.264 视频编码算法；对开源代码 Live555 进行功能扩展并作为流媒体服务模块，负责处理客户端 RTSP 连接请求、音视频数据的 RTP 打包和发送等功能。最后，使用流媒体客户端 VLC 对视频服务器进行功能测试，可获得实时性较好、流畅度较高的图像和声音。

关键词：视频监控；H.264；Live555；流媒体

Video Server on Blackfin

YUAN Yan-Su[1,2], SUN Jian-Wei[2], LI Dan[3], YANG Hai-Bo[2], JIA Jun-Ying[1,2]

[1](Graduate University, Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology, Chinese Academy of Sciences, Shenyang 110171, China)
[3](Northest China Grid Company, Shenyang 110006, China)

Abstract: As the rapid development of network communication and multimedia technology, demand of embedded video surveillance devices is also increasing. This paper provides an embedded video server solution, which uses a high robustness and high compression algorithm-H.264, to encode video source and transplants the open-source Live555 as streaming media module which is responsible for receiving external RTSP connection request, packing sound and video data in a RTP way, as well as sending them. In the end, lots of tests have been done by using standard streaming media client VLC, which prove a timely, fluent and better picture/sound quality.

Key words: video surveillance; H.264; live555; streaming media

1 引言

视频监控是安全防范系统的重要组成部分，它以直观、准确、信息内容丰富和实时等特性而得到广泛的应用。随着嵌入式、多媒体和网络通讯技术的迅猛发展，安防监控等行业发展更加趋向于全面的数字化和网络化，利用嵌入式终端进行监控的需求量也将大大增加。嵌入式视频监控系统对于提高监控系统的数字化、网络化、减少成本等方面具有积极的意义。

本文结合流媒体和嵌入式技术，提出了一种嵌入式终端上的视频服务器解决方案。该方案基于 Blackfin 平台，采用 uClinux 操作系统，包含音视频数据的实时采集与编码、流媒体服务和媒体流发送等功能模块。

文中首先给出了视频服务器的硬件结构，接着详细介绍了软件部分的设计与实现，最后对视频服务器进行了功能测试。

2 系统硬件框架

视频服务器硬件部分主要包括核心处理器、音频采集设备、视频采集设备、SPORT 接口、UART 接口、PPI 接口和无线模块等，其硬件结构如图 1 所示。

核心处理器采用 Blackfin 双处理器三核心架构，主处理器（BF536）负责网络交互，协处理（BF561

① 收稿时间:2011-10-31;收到修改稿时间:2011-12-19

双核）负责视频编码；主/协处理器之间通过高速同步串口（SPORT 接口）通信。视频编码采用高压缩比、高鲁棒性的 H.264 编解码算法，由单独的处理器负责视频编码工作，满足了 H.264 算法对运算能力的要求，提高了编码效率。

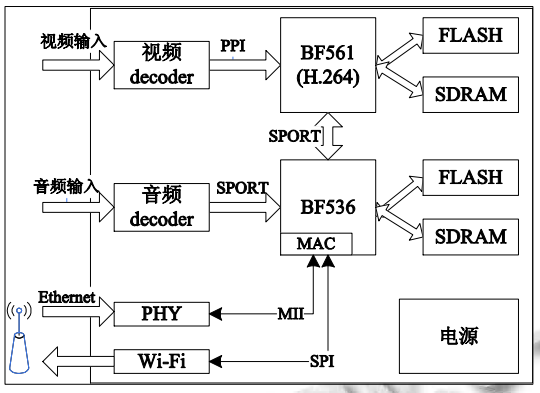

图 1　视频服务器-硬件结构

协处理器采用了对称双核的架构，集成了两个工作频率均高达 756MHz 的 Blackfin 处理器内核，这种特性保证了其强大的数字信号处理能力；音频采集使用 24 位立体声音频编解码芯片 wm8731，该芯片具备高性能、低功耗等优点，其拥有的多种功耗降低模式，为用户避免了电池功率消耗，通过 SPORT 高速串口与主处理器通信；视频解码器使用 ADV7180 芯片，通过 I2C 总线接口进行寄存器配置，并通过 PPI 并行外设接口与协处理器通信；Wi-Fi 模块采用的 MARVELL 芯片组，通过 SPI 接口实现与主处理器的通信。

3　系统软件设计

BF561 上运行一个优化的 H.264 编码算法库，实现对视频流的编码处理；BF536 上移植了 Uboot 和 uClinux 操作系统，实现对音视频数据的流化处理及其网络传输，软件整体结构如图 2 所示。

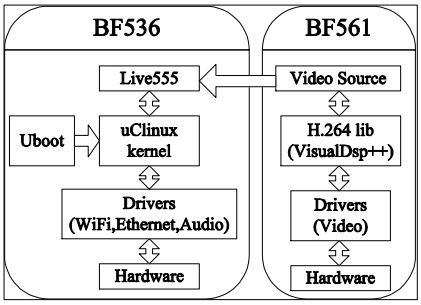

图 2　视频服务器-软件框架

在 uCLinux 系统上移植了开源代码 Live555 并对其进行了功能扩展，作为视频服务器的流媒体服务模块，实现对音视频流的实时读取、RTP 打包和 RTSP 会话管理等功能。

3.1 H.264 视频编码

3.1.1 H.264 算法

H.264[1]是一种高性能的视频编解码技术，它拥有有很高的数据压缩比率，同等图像质量的条件下，其压缩比是 MPEG-4 的 1.5~2 倍，在网络传输过程中所需要的带宽更少；它还拥有图像流畅度高，容错能力和网络适应性强等优点。

3.1.2 视频编码实现

文献[2]设计的 H.264 编码器没能达到实时性，它只把编码后的视频流保存成文件的形式，再通过流媒体播放器来播放文件。而本文创新点是（1）在 BF561 处理器中采用已优化的 H.264 编码算法[3]，（2）编码后的视频流通过高速同步串口（SPORT 接口）发送给 BF536，BF536 上流媒体服务模块可读取 SPORT 口获取 H.264 视频流，（3）流媒体服务模块采用标准的流媒体传输协议进行媒体流传输。

3.2 音视频流化处理

Live555[4] 项目包括四个基本库：UsageEnvironment、BasicUsageEnvironment、groupsock 和 liveMedia。UsageEnvironment 和 BasicUsage Environment 用于事件调度，包括对事件的异步读取、句柄设置和错误信息的输出；groupsock 用于数据包的接收和发送；liveMedia 包含了 RTSP Server 类和针对不同流媒体类型编码的类，如*MediaSubsession，*Source，*Sink 等。

Live555 源码支持传输 H.264 视频文件，但不支持音视频实时流的传输，可根据 DeviceSource 类模板来设计程序框架[5]。本文需同时获取音视频实时流，设计和实现的两个类分别为：H264VideoDeviceSource 和 PCMAudioDeviceSource，用来实时地读取音视频流。

3.2.1 H264VideoDeviceSource 类

由于 Live555 中读取视频实时流和音频实时流的处理逻辑相似，下面只是介绍 H264VideoDeviceSource 类的实现框架：

```
class H264VideoDeviceSource: public FramedSource {
……
```

```
//省略部分成员函数和成员变量
public:
   virtual timeval getPresentionTime();
   virtual void initSourceParam();//init device
private:
 virtual void doGetNextFrame();
public:
void CaptureVideoSample();
    bool ReadVideoSample(unsigned char* buf, unsigned int* pLen,struct timeval *pT);
 CRingBuffer m_RingBuffer;//用来维持同步机制
public:
 void deliverFrame();//从缓冲区中读取视频数据
 //开启线程处理从 SPORT 口读视频流到缓冲区
 void* ThreadProcLinux(void *params);
 unsigned int StartThread();//开启读线程
 void StopThread();
};
```

3.2.2 音视频实时流读取

流媒体模块实时地读取音视频流的方式如下：

① 读取音频数据

```
fd_adc = open("/dev/dsp", O_RDONLY, 0);
size = read(file_adc, pcm_buf, sampleSize);//读音频流线程调用此函数
```

② 读取编码后的视频数据

```
fd_img = open("/dev/adv7180", O_RDWR);
fd_enc = open("/dev/sport_io", O_RDWR);
ioctl(fd_enc, SPORT_IOC_SETADDR, tmp_buff);
size = read(fd_enc,tmp_buff,0);// 以阻塞方式从 SPORT 接口读取视频帧。
```

流媒体服务模块以阻塞方式从 SPORT 接口读视频帧，该接口无数据时，调用函数无法及时返回。当流媒体服务模块同时处理音视频数据流时，会造成视频卡死的现象。针对此问题，本文提出一种缓冲机制，缓冲机制的实现方式见 3.2.3 小节。

3.2.3 音视频流缓冲机制

本文设计加入了多线程的缓冲机制[6]，开启一个线程负责获取 H.264 视频数据流。该线程与 Live555 共用一个缓冲区，该线程负责把从 SPORT 接口读到的视频数据放到缓冲区，而 Live555 读取视频帧时，则从该缓冲区中取数据。（注：音频数据处理过程类似）

① CRingBuffer 类

CRingBuffer 类实现对 FreeBuffer 和 DataBuffer 两个队列的维护。队列元素的数据结构如下所示：

```
typedef struct {
    unsigned char* buf;//数据缓冲区
    unsigned int   capacity;//容量
    unsigned long  t;//时间
    unsigned int   len;//数据长度
} DeviceSample;
```

CRingBuffer 的实现主要包括 GetFree()，Pop()，Push()和 Reclaim()四个函数。其中 GetFree()/Reclaim()为 FreeBuffer 队列取出/回收一个元素，Pop()/Push()为 DataBuffer 队列取出/压入一个元素。

② 写缓冲区

视频流读线程对 CRingBuffer 缓冲区写视频数据的执行操作流程如图 4 所示。

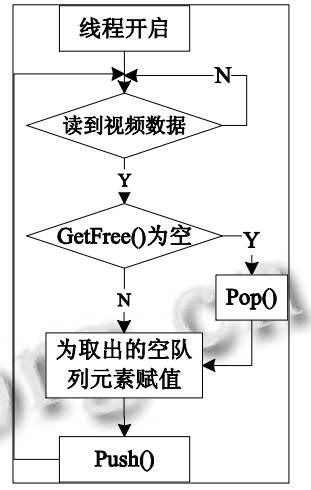

图 4 视频流读线程执行流程

当读到视频数据，而 FreeBuffer 为空时，则采取替换 DataBuffer 中最旧元素的机制。

③ 读缓冲区

Live555 从 CRingBuffer 缓冲区读视频帧的执行操作流程如图 5 所示。

BF536 采用阻塞方式从 SPORT 接口读取视频流，当外部请求播放音视频流时，这种阻塞方式会加大 Live555 处理延迟；而采取多线程的方式，Live555 取视频帧时直接从缓冲区取数据，当 DataBuffer 中无数据时，调用函数可以马上返回。

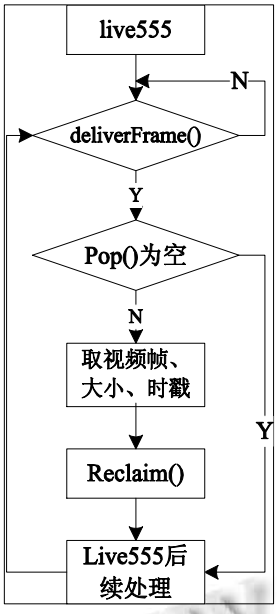

图 5　Live555 取视频帧执行流程

多线程音视频流缓冲机制的加入，解决了流媒体服务模块处理音视频流时，视频卡死的现象。

3.3 网络传输

3.3.1 传输协议

实时数据流传输要用到以下三个协议标准：RTSP、RTP/RTCP 和 SDP。其中，RTSP 是流媒体通信协议，主要用来控制具有实时特性的数据发送，它本身并不传输数据；RTP 是流媒体数据传输协议，为数据流提供时间信息和实现流同步，依靠 RTCP 提供可靠的传送机制和拥塞控制；SDP 会话描述协议，RTSP 服务器回应 DESCRIBE 命令需求的媒体初始化描述信息时用到。

Live555 的 liveMedia 库中包含了对上述协议的实现，本方案的流媒体服务模块对其进行了功能扩展，客户端进行 RTSP 请求时，同时建立音频流和视频流，会话建立的过程中 Client 端向 Server 端发送的命令如下所示。

```
OPTION
DESCRIBE
SETUP                //指定视频传输机制
SETUP                //指定音频传输机制
PLAY
……..//音视频流传输过程
TEARDOWN             //会话结束信令
```

3.3.2 发包策略

流媒体服务模块发送媒体流采用组播的方式，当至少有一个 RTSP 连接建立时才向组播地址发送媒体流，否则丢弃。

无客户端请求时，采取丢弃数据包的方式，减少了无连接请求时的网络带宽占用。

4　功能测试

4.1 参数设置

本方案支持的图像分辨率和码率的设置，如表 1 所示。

表 1　图像分辨率参数

分辨率	码率控制	码率	量化因子	高度	宽度
D1	可变	-	28	720	576
	固定	1500	-		
HD1	可变	-	30	720	288
	固定	768	-		
CIF	可变	-	30	352	288
	固定	384	-		

视频的帧率设置为 25 帧/秒，I 帧间隔为 25。

4.2 实验验证

4.2.1 功能测试

系统的软硬件环境都搭建完成后，用 VLC 客户端请求播放 URL 地址为 rtsp://192.168.1.78/Capture，可得到实时性较强、较为流畅的音视频质量（图像视觉延迟小于 1s），VLC 截图如图 6 所示。

图 6　VLC 播放示图

4.2.2 带宽测试

通过 wireshark 工具，抓包并分析在固定码率和可变率下的三种视频分辨率（D1、HD1、CIF）所占网络带宽的情况，分别如图 7 和图 8 所示。两图中各包含四条线，最上面一条线标记为 I 线（拥有三个波段），中间较平稳的一条线标记为 II 线，第三和第四条线（距离横坐标轴较近）标记为 III 和 IV 线。

I 线（9763 Sport）代表 Video RTP 带宽占用，III 线（9762 Sport）代表 Video RTCP 带宽占用，II 线（9761 Sport）代表 Audio RTP 带宽占用，IV 线（9760 Sport）代表 Audio RTCP 带宽占用。

图 7 固定码率-分辨率及其网络带宽占用

I 线三个波段分别表示视频为 D1、HD1 和 CIF 分辨率时所占用的带宽情况，其中 D1 分辨率占用的带宽最大，其次是 HD1，最小的为 CIF，如上图 7 所示。

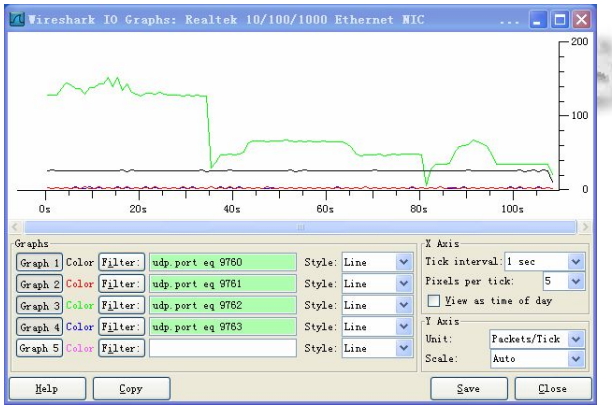

图 8 可变码率-分辨率及其网络带宽占用

可变码率下图像画面变化剧烈时，占用的网络带宽也会相应增加；反之，视频占用的网络带宽对比固定码率所占带宽要小。

5 结论

本文设计并实现了一种基于 Blackfin 平台的视频服务器。它采用了 Blackfin 双处理器三核心的架构，两个处理之间使用高速串口进行通信；用单独的处理器负责视频编码工作，提高了编码效率；视频采用已优化的 H.264 算法进行编码，可对视频进行参数设置，默认为 D1 分辨率；音频采用 Live555 库中的 G711 方式编码。对开源代码 Live555 进行了功能扩展，并移植到 uClinux 系统下作为流媒体服务模块，负责音视频流的分析、RTP 打包和发送等。流媒体服务模块在获取音视频媒体实时流时，采取了多线程的缓冲机制，解决了无视频数据时 Live555 调用函数无法及时返回的问题；采用组播方式发数据包，当至少有一个外部 RTSP 连接建立时才向组播地址发包。实际的功能测试中，用标准流媒体客户端软件 VLC 对播放地址的进行播放请求，能获得实时性较强（视觉延迟小于 1 秒）、较为流畅的视频和声音质量。

参考文献

1 毕厚杰.新一代视频压缩编码标准.北京:人民邮电出版社,2005.97–166.

2 王庆辉,高颖.基于 ADSP-BF561 的 H.264 编码器设计.电子技术应用,2009,292:156–158.

3 H. 264 BASELINE PROFILE ENCODER. http://www.analog.com/zh/processors-dsp/blackfin/BF_H264_BP_ENCODER/processors/product.html.

4 LIVE555 Streaming Media. http://www.Live555.com/ liveMedia/.

5 王思嘉,裴海龙,陈锲.基于无人机的 Linux 红外视频采集平台实现.微计算机信息,2010,19:71–73.

6 Vun N, Ansary M. Implementation of an Embedded H.264 Live Video Streaming System, IEEE ISCE 2010.