基于多协议融合的实时移动视频监控系统①

叶 锋[1,2], 傅燕云[1], 陈林芳[1], 钟晓君[1], 邓衍晨[1]

[1](福建师范大学 数学与计算机科学学院, 福州 350007)
[2](北京邮电大学 多媒体技术中心, 北京 100876)

摘 要: 提出一种支持多协议融合的实时视频监控系统的设计与实现方案. 系统可通过 Hi3520 嵌入式处理器平台和移动端摄像头进行视频采集, 采用 H.264/AVC 压缩编码和参数优化, 兼容 RTP/RTSP 和 RTMP 多种通信协议进行媒体流传输; 设计的 Android 的移动客户端可以实现视频数据的实时解析、网络串流等. 实验测试表明: PC 和采集端显示的画面仅有 1~1.5 秒延迟, 系统可以灵活实现不同应用场景下的高清、实时监控.

关键词: 视频监控; RTP/RTSP; RTMP; Android; H.264/AVC

Real-Time Mobile Video Surveillance System Based on Multi-Protocol Communications

YE Feng[1,2], FU Yan-Yun[1], CHEN Lin-Fang[1], ZHONG Xiao-Jun[1], DENG Yan-Chen[1]

[1](School of Mathematics and Computer Science, Fujian Normal University, Fuzhou 350007, China)
[2](Multimedia Telecommunication Centre, Beijing University of Posts & Telecoms, Beijing 100876, China)

Abstract: A real-time mobile video surveillance system based on multi-protocol communications is proposed. The video acquisition module is established with Hi3520 board and mobile camera, while parameter-optimized H.264/AVC is used for video compression. And streaming media protocols compatible with RTP/RTSP and RTMP are used for video transmission; Mobile client based on Android platform can realize real-time analysis for network video streaming data etc. Experimental results show that there is only 1~1.5 second delay for the PC Client and the Video Capture terminal display. Thus the proposed system is a practical solution for the real-time monitoring of multiple application scenarios. The system designed here can obtain continuous scenes and realize HD, real time video monitoring remotely.

Key words: video surveillance; RTP/RTSP; RTMP; Android; H.264/AVC

随着现代移动网络的飞速发展, 视频监控系统正朝着数字化、网络化、功能综合化的方向不断发展. 然而, 市场上广泛使用的第二代(基于“PC+多媒体卡”数字视频监控系统, DVR)及部分第三代视频监控系统(基于 IP 网络视频监控系, IPVS)由于其推广成本及灵活性等方面的缺陷已经越来越不能满足大众用户的需求. 因此, 构建可灵活配置的移动视频监控是未来监控系统发展的必然趋势. 同时, 伴随着 Android 智能终端的大面积普及应用, 开发面向 Android 平台的移动视频监控系统具有广阔的应用前景.

移动视频通信首先要解决大数据压缩和流媒体传输的问题. H.264/AVC[1]是由 ITU-T 和 ISO 联合制定的低比特率视频编码标准, 它具有压缩比高、图像质量好、网络适应性强等特点, 更能满足移动视频环境低功耗、网络状态不稳定的应用需求. 压缩后的数据需要通过流媒体协议来实时传输. RTP(Real-time Transport Protocol, 实时传输协议)是用于 Internet 上针对多媒体数据流的一种传输协议[2], 其目的是提供时间信息和实现数据流同步. RTP 本身只保证实时数据的传输, 它依靠 RTCP (Realtime Transport Control Protocol, 实时传输控制协议)实时反馈和调整提供可靠传输服务. RTP 数据包依 RTSP(Real-time Streaming

① 基金项目:国家自然科学基金(61271190);福建省自然科学基金(2012J01251);福建省教育厅项目(JB12033);2014 年国家级大学生创新创业训练计划
收稿时间:2014-07-25;收到修改稿时间:2014-09-16

Protocol, 实时流协议)完成数据包的发送. RTSP 是由 Real Network 和 Netscape 共同提出的应用层协议[3]. 该协议用来解决如何有效地通过 IP 网络传输流媒体数据, 它对流媒体提供了诸如暂停, 快进等控制, 而本身并不传输数据. RTMP (Routing Table Maintenance Protocol, 实时消息传送协议)是由 Adobe 公司提供的专门用来高速传送音视频和数据信息传输层协议[4]. 它能够提供永久的 Socket 链接, 支持声音、影像和脚本数据从服务器到客户端的双向多条线路的动态传输, 并使用 Flash Player 作为客户端.

基于上述流媒体协议和压缩标准, 本文提出了一种基于 H.264/AVC 视频压缩标准且兼容多种传输协议的实时移动视频监控系统. 重点研究网络传输环境下的流媒体视频质量参数优化、解决媒体流通过 RTP/RTSP 和 RTMP 协议的实时转换和实时传输问题. 以下从视频采集、流媒体服务器和移动视频监控终端三个部分阐述本项目的开发过程及相关软件设计思想.

1 系统概述

提出的移动视频监控系统采用 C/S 架构方式(如图 1 所示), 由视频采集压缩模块、流媒体服务器、Android 客户端三部分组成. 为了适用于不同的应用场景, 视频采集模块分成了基于 HI3520 的高性能视频采集模块和基于 libstrenming 的移动端视频采集模块. 摄像头采集的模拟视频信号经过 A\D 芯片转换成数字视频信号, 数字视频信号经过嵌入式处理器进行预处理, 并采用 H.264/AVC 压缩编码. 编码后的视频流通过网络传输到流媒体服务器, 由流媒体服务器负责对视频流进行网络传输优化与协议转换和分发, 以此保证视频传输的实时性. Android 客户端则通过 RTSP/RTMP 协议访问接收到的待解码视频流并对解码码流进行后处理.

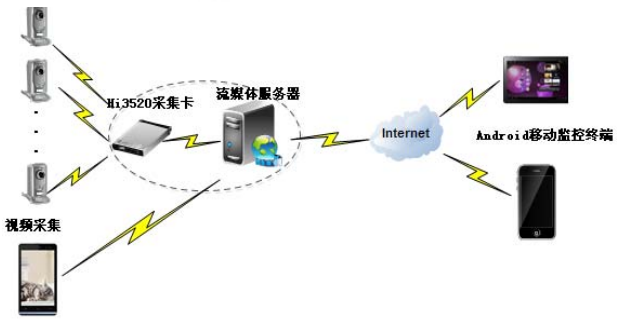

图 1 提出的移动视频监控平台架构图

2 视频采集压缩模块

2.1 基于 HI3520 的高性能视频采集模块

本模块主要用于视频分辨率和实时性要求较高的环境. Hi3520 是海思半导体推出的一款基于 ARM11 处理器内核以及视频硬件加速引擎的高性能通信媒体处理器[5]. 它采用双 DDR 架构, 处理频率达 600MHz; 并提供 H.264/AVC 和 MJPEG 多协议编解码和丰富的视频输入输出接口(CVBS、高清 VGA、BT1120), 能够带来更加清晰的画质和视频体验. 本系统采用 HI3520 Demo 单板作为采集端: 通过摄像头采集视频信号, 通过 CVBS 信号输入 TW2865(四通道视频解码芯片); 4 片 TW2865 采集到 16 路视频信号, 再把它们通过 BT.656 接口输入到 Hi3520 芯片; 芯片将收到的视频数据进行图像处理后进行非标准 H.264 编码, 音频信号经过芯片编码后通过 PCI-SATA 电路储存到硬盘上.

2.2 基于 libstreaming 的移动端视频采集模块

移动视频采集作为未来视频采集的一种重要方式拥有广大的应用前景. 为此, 提出的移动端视频采集模块主要将智能手机作为一个流媒体采集器, 将摄像头取景实时串流到无线局域网中, 输出的码流既可以通过流媒体服务器对其进行存贮和转发, 也可以通过网络中的其他手机或者智能终端上实时查看手机取景. 本功能模块主要基于著名开源项目 Spydroid 推荐的 SDK: libstreaming[6], 在 Android 环境下进行开发具体, 实现过程见 4.3 节.

2.3 视频编码参数优化

为了降低网络带宽开销, 需采用较高压缩率的 H.264/AVC 参数配置编码采集下的视频流. 这也同时降低了码流对丢包或误码问题的鲁棒性. 因此, 需要根据不同的网络带宽、丢包、时延等网络环境下视频的主客观质量, 选择最优的视频编码参数. 本文采用 WANem 网络模拟器, 分别测试了 370、500、680、900 Kbit/s 4 种网络带宽(Specify BW)下, 0%、2%和 6%丢包(Loss)3 种分组丢包率的客户端视频主客观质量. 可以发现丢包率越高, 视频主客观质量越差, 而带宽、码率越高, 视频重建质量越好. 当带宽为 500Kbit/s 时, D1 分辨率的视频流, GOP 在 25-30 左右视频主客观质量最好. 当 GOP 设置大于 40, 视频主观质量发生明显下降. 因为 GOP 间隔设置过大, 容易产生误码扩散现象, GOP 设置过短, 则会导致码率上升而效果一般. 相同 GOP 条件下, QP 参数在 15~23 之间, 可以获得较为

满意的主观评价.

3　基于RTSP/RTMP的流媒体服务器设计

流媒体服务器主要负责媒体流的存取调度, 管理和传输, 以流式协议(RTP/RTSP、RTMP 等)将视频文件传输到多个客户端, 供用户在线观看. 流媒体通过流(Streaming)的形式在网络中传输多媒体文件并进行数据传播, 客户端将已存储在缓冲区的媒体数据进行播放. 这样, 服务器端的媒体流持续不断地传送到客户端, 以实现“边载边播”的功能. 通常, 基于 HTTP 的渐进下载的流媒体播放只能支持点播, 实时性较差, 也不支持快进、快退等 VCR 操作. 因此, 本模块首先采用 RTSP/RTP 协议栈以保证媒体传输的实时性和可控性. 流媒体服务器从视频采集模块接收实时视频流, 进行 RTSP 会话交互后, 再通过 RTP 协议封装发送给客户端. 另一方面, 考虑到 RTMP 协议提供的流媒体无需安装客户端程序, 用户可以十分方便地通过浏览器收看流媒体. 本模块同时考虑了基于 RTMP 协议的直播和点播服务.

3.1 基于 RTSP/RTP 协议栈的实时通信服务

基于 RTSP/RTP 协议栈, 本文采用 C/S 模式设计的服务器. RTSP/RTP 协议栈如图 2 所示: RTSP 在体系结构上位于 RTP 和 RTCP 之上, 它使用 RTP 或 TCP 完成视频数据传输, 使用 RTCP 协议为应用层提供视频质量控制. RTSP 控制 RTP 数据包的发送, 使用 RTSP 时, 客户机和服务器双向都可以发出请求.

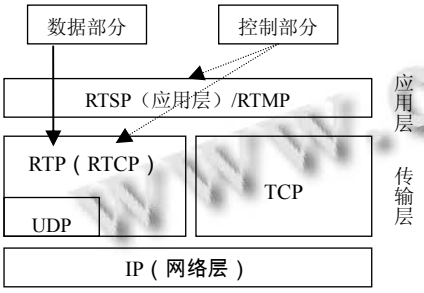

图2　RTSP/RTP 流媒体协议栈

服务器与客户端协议交互流程如图 3 所示. 服务器在默认的端口(端口号: 554)进行网络监听, 根据 RTSP 规定的格式对收到的消息(DESCRIBE)数据包进行解析. 服务器为每个客户端产生单独会话并根据 rtspURL 找到相应流媒体, 然后对应生成 RTSP 格式的应答消息数据包, 响应客户端的 SETUP 请求. 接收到流媒体数据或文件后, RTSP 服务器在本地申请和建立新的 RTP 端口和 RTCP 端口. 当处理播放请求时, 先对 rtspURL 等数据记性处理, 根据 Header 的数据对流进行处理: 首先设置 RTP 包头, 再填充 264 编码帧数据, 最后就发送 RTP 数据包和 RTCP(SR)包. 服务器通过任务调度器进行延时控制, 重新组包然后发送新的数据包, 这样客户端可以源源不断地收到服务器传来的 RTP 和 RTSP 包.

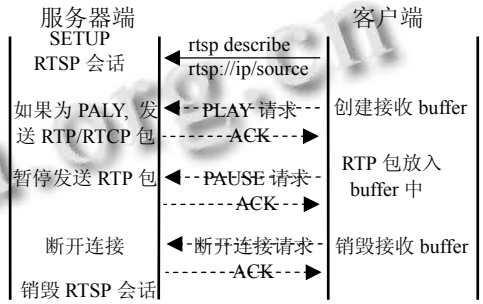

图3　服务器与客户端协议交互流程图

3.2 基于 RTMP 实时流媒体直播

本系统同时支持使用 RTMP 协议在流媒体服务器与客户端之间交换信令和媒体数据. 实际设计采用 Adobe 公司的 File Media Server(FMS)提供流媒体服务. 基于 RTMP 协议的流媒体服务如图 4 所示. 采集后视频分为两种情况进行 RTMP 直播: 一种是普通摄像头采集的视频经过 H.264/AVC 压缩编码后, 先通过 FFmpeg 转码成符合 flv 格式的视频流, 然后由 FMS 发布. 另一种情况, 为同时支持 RTSP 和 RTMP 协议, 协议转换模块将 HI3520 采集输出的 RTSP 视频流通过 Live555 进行码流协议分析后, 按 RTMP 流媒体格式封装成 RTMP 直播流, 提供给 FMS 直播. FMS 服务器通过编写 Action script 脚本实现视频流的管理控制.

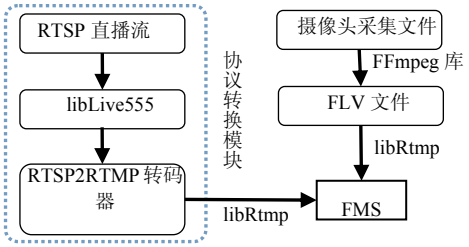

图4　RTMP 直播流程图

实现 RTMP 步骤主要分两步走: 第 1 步, 建立一个网络连接(NetConnection) ; 第 2 步, 基于该网络连

接建立一个网络流(NetStream).

4 基于ANDROID的移动客户端设计

Android 移动视频监控客户端针对接收到的流媒体数据进行解码并将监控画面呈现给用户. 此外, 为了给用户提供更加灵活的移动安防体验, 客户端同时实现了将摄像头取景实时串流到其他终端的功能. 根据上述分析, Android 客户端的设计主要包含以下 3 个模块: 流媒体设备地址管理模块、流媒体视频播放模块和摄像头取景串流模块.

4.1 流媒体设备地址管理模块

本模块主要实现对流媒体设备地址的管理, 包括增、删、改、查等数据操作, 此外, 当设备较多时, 可利用 XML 文件进行批量导入设备地址. 本模块主要涉及到的技术包括 SQLite 数据库与 XML 文件解析. 播放视频时, 用户通过 UI 将流媒体设备的地址存入 SQLite 数据库中. 当用户通过点击选择相应的设备时, 程序就会将选中设备的地址传入相应的方法, 最终调用 FFMPEG 视频解码库进行播放. 模块主要设计方法描述如下:

(1)三层架构

三层架构作为企业级开发的通用技术, 广泛地应用在数据库访问的领域中. 其具有三个层次, 即: 数据访问层(Data Access Layer, DAL, 负责底层的数据库操作)、业务逻辑层(Business Logic Layer, BLL, 负责数据有效性验证等业务逻辑)和表现层(UI, 负责与用户的直接交互). 这三层分别负责不同的功能, 将各层之间的耦合程度降到最低. 三层之间传递的数据模型是对流媒体设备地址的抽象, 其模型定义和 DAL 层对外主要接口定义如表 1.

表 1 模型定义和 DAL 层对外主要接口定义

Device			
-DeName ; -DeAddress			
+GetName(): String; +GetAddress(): String +SetName(): String; + SetAddress(): String			
方法	功能	参数描述	返回值
public void Add(Device de)	添加设备地址到数据库中	de: 要增加到数据库的数据模型	无
public bool Delete(Device de)	从数据库中删除指定的设备	de: 要从数据库删除的数据模型	如果删除成功, 返回 true, 否则返回 false
public bool Update(Device de)	修改指定的设备信息	de: 要修改的数据模型	如果修改成功, 返回 true, 否则返回 false
public Device GetDevice(Stringname)123	根据名字查找设备	name: 要查找的设备名	找到的设备数据模型

(2)Sqlite 数据库的使用

① 定义 SQLiteOpenHelper 的一个子类, 完成数据库的创建、升级等逻辑(SQLiteOpenHelper 是包装了数据库的创建、打开和更新的抽象类).

② 编写辅助类, 通过标准 SQL 语句完成流媒体设备增删改查功能, 实现上述 DAL 层的各种接口.

4.2 流媒体视频播放模块

流媒体播放是本系统核心的部分. 系统采用 Vitamio 框架(http://www.vitimao.org)作为接收和解码的 SDK. 它使用 FFmpeg 作为解码器, 同时开发了针对不同移动平台的硬解码方案, 能够支持 H.264/AVC、H.263、MPEG4 等常见的视频编码和 RTSP、RTMP、MMS、HTTP 等多种协议. 本模块设计主要方法如下:

(1)MediaPlayer 类的调用

Vitamio 框架的核心是 MediaPlayer 类, 在它的生命周期中, 有 Idel、Initialized、Prepared、Started、Stopped、Pauseed 等状态. 创建 MediaPlayer 对象后, 设置数据源并准备(调用其 prepare() 方法或者 prepareAsync()方法), 即可使之处于 Started 状态. 在此状态下, 可完成对视频的播放.

(2)视频播放操作代码

采用 Vitamio 框架中 MediaPlayer 类方法的视频播放操作代码如下:

```
//创建 MediaPlayer 对象
 mMediaPlayer = new MediaPlayer(this);
//设置数据源 mMediaPlayer.setDataSource("rtsp://ip:端口");
//设置播放区域
mMediaPlayer.setDisplay(holder);
//异步准备数据
mMediaPlayer.prepareAsync();
```

当 mMediaPlayer 数据准备完毕后, 调用其 start 方法开始播放.

4.3 摄像头取景实时串流模块

实时串流模块主要将手机作为一个流媒体服务器, 将摄像头取景实时串流到无线局域网中, 使得用户可以方便地在同一局域网的其他手机或者其他智能终端上实时查看手机的取景. 用户通过配置 RTSP 服务协议的各项参数, 并将其存入本模块的 Shared Preference, 再根据这些参数调用 RTSP 核心库, 启动

流媒体服务. 本模块基于开源项目 Spydroid 推荐的 SDK: libstreaming 实现的. 该 SDK 可定制性较强, 可方便的设置使用的流媒体协议、分辨率、码率和帧率, 有助于提高的软件开发的效率和软件本身的稳定性. 本功能所采用的传输协议是 RTP/RTSP, 需采用 libstreaming 的 RtspServer 服务类, 取景实时串流的主要过程实现如下(如图 5):

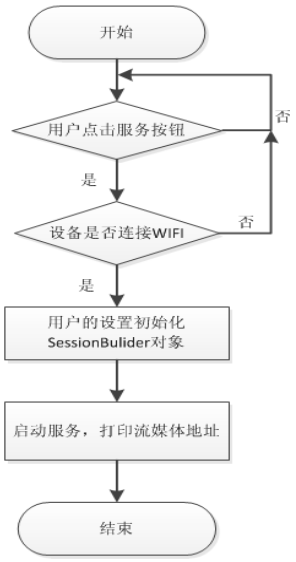

图 5　取景实时串流模块设计

(1)在 Android 程序的 Manifest 文件中对 RtspServer 服务进行注册.

<service android:name="net.majorkernelpanic.streaming.rtsp.RtspServer"/>

(2)采用 SessionBuilder 构建 session, session 是服务器与客户端间通信的载体. 此部分主要设置 sessionBuilder 的一些选项, 如指定音视频编码器等.

```
SessionBuilder.getInstance()
.setSurfaceHolder(mSurfaceView.getHolder())
.setContext(getApplicationContext())
.setAudioEncoder(SessionBuilder.AUDIO_AAC)
.setVideoEncoder(SessionBuilder.VIDEO_H264);
```

(3)启动或停止 Service

context.startService(new Intent(this, RtspServer.class))或 context.stopService();

5　实验结果及实时性分析

本节首先对流媒体客户端和服务器的负载性能进行测试. 流媒体服务器配置为: 操作系统 WINDOWS 7, CPU 酷睿 i7 2.4G, 内存 4G. 如第 2 节分析, 将视频编码 H.264/AVC 的 GOP 设为 26, 量化参数设为 21. 客户端网络环境包括 PC 网络线路直连、Wifi 网络和 3G 无线网络. 测试结果如表 2 所示, 可以发现在 Wifi 局域网的条件下, 移动设备还是客户端登陆均可获得不错的图像质量.

表 2　客户端性能测试

客户端配置	分辨率	视频路数	测试效果
WIFI			
PC:windows 7 酷睿 2.8GHz 2G 内存 256 显存	D1	16	视频质量高、无马赛克、无卡顿 流畅播放(15 帧/s)
	720P	4	视频质量高、无马赛克、无卡顿 流畅播放(9 帧/s)
小米 2s	D1	4	视频质量高, 无马赛克、延时小于 1.5 秒
	720P	1	视频质量高, 偶尔有马赛克、延时小于 3 秒
3G			
小米 2s	D1	4	视频质量较高, 偶尔有马赛克、无卡顿、延时小于 2 秒(15 帧/s)
	720P	1	视频质量高, 偶尔有马赛克、偶尔有卡顿、延时小于 3 秒(9 帧/s)

在小米 2s 上的测试结果为: PC 终端和采集端显示的画面仅有 1 ~1.5 秒延迟. 当一台手机作为采集端, 并进行流媒体实时串流(不经过服务器转发), 另一终端显示画面仅有 2~3 秒延迟. 由图 6 可以发现: 移动终端在电信 3G 信号覆盖下, 视频图像流畅清晰. 当设置帧率为 9 帧/秒, 3G 网络环境下, 可以实现 720p 分辨率的移动实时监控, 显示终端与采集端时延可以控制在 3 秒以内. 提出的实时移动视频监控系统实际监控画面如图 6 所示.

(a) 基于 RTP/RTSP 的手机端监控画面

(b) PC 端基于 RTMP 的监控画面

图 6 系统流媒体监控画面

图 6(a)为基于 RTP/RTSP 的手机移动视频监控画面和(b)为 PC 端基于 RTMP 流媒体协议的直播画面. 上述测试过程中, 流媒体服务器、客户端软件均表现稳定, 表明该系统可以满足实际应用需求.

6 结语

基于 Android 开发平台、H.264/AVC 视频压缩技术, 本文提出了一种兼容多种流媒体协议的实时移动视频监控的设计方案. 系统采用 RTP/RTSP 和 RTMP 等流媒体协议, 具有实时、高效、便携等特点, 可实际应用与当前视频监控领域. 系统在下一步工作中将采用 HEVC 进行视频编解码, 进一步改善码流、编码质量、延时和算法复杂度之间的关系, 以达到最优化设计.

参考文献

1 ITU-T. Recommendation H.264 Advanced Video Coding for Generic Audiovisual Services. Rec. H.264 and ISO/IEC 14496-10 Version 11, 2009.

2 张前进.基于 RTP 的 H.264 实时传输系统的设计与实现.企业开发技术,2011,30(23):1–2.

3 Schulzrinneh, RL. RFC232, Real time streaming protocol. 1998: 23–24.

4 Adobe Systems Incorporated. RTMP Specification. http://www.adobe.com/content/dam/Adobe/en/devnet/rtmp/pdf/rtmpspecification1.0.pdf. 2009.

5 深圳市海思半导体有限公司. Hi3520?Demo 单板用户指南. 2009:35–36.

6 Libstreaming.https://github.com/fyhertz/libstreaming. 2014.