基于移动终端的视频监控系统的设计与实现

王涛

摘要：

近年来，随着互联网技术的发展以及带宽的提高，移动互联网的兴起，使得互联网的媒体载体发生了深刻的变革。之前受制于网络的带宽，网络媒体主要以文本图像为主，辅以少量的音频媒体；而现在，视频逐渐成为了互联网媒体载体的主流形式。相较于文本、静止的图像及音频媒体，视频媒体具有生动、信息量大等特点。因此，其在远程教育、远程医疗、视频会议、在线直播、视频电话和航空航天等领域得到了广泛的应。

为了进一步扩展视频流媒体的应用范围，本文提出了一种基于移动终端的视频监控系统。该系统采用客户端/服务器（Client/Server，C/S），由流媒体服务器和流媒体终端构成。在服务器的实现中，研究了嵌入式Linux开发环境、H.264视频编码压缩算法和RTP/RTCP等内容，从而提出了基于嵌入式Linux环境的流媒体服务器的方案。在客户端，通过安卓（Android）移动操作系统的提供的JNI（Java Native Interface，Java原生接口），实现对于实时流媒体视频的解码回放。

在论文的最后一章，通过实验说明，本文提出的视频监控系统创新地实现了在移动终端上的实时解码回放，为以后流媒体视频应用范围的进一步拓宽打下了基础。

关键词：流媒体视频、移动终端、Linux、H.264、Android

第一章、绪论

近年来，移动互联网的兴起以及互联网带宽的逐渐提高使得实时视频监控技术成为可能。传统的监控系统包括前端摄像机、传输线缆、视频监控平台。摄像机可分为网络数字摄像机和模拟摄像机，可作为前端视频图像信号的采集。视频监控以其直观、准确、及时和信息内容丰富而广泛应用于许多场合。

当下，在信息技术领域正在发生着前所未有的变革。云计算、大数据以及物联网是这场变革中的三驾马车。而物联网利用其自身的传感器网络所获取的数据更是云计算与大数据的基石。同时，鉴于视觉是人类获取信息最为直观和最主要的途径，因此，研究如何利用逐渐兴起的移动互联网,方便、高效、便捷地捕捉图像（视频）信息具有非常重要的实践意义。

1.1 课题研究背景及意义

网络视频监控系统是一种利用计算机技术将视频信息数字化来进行传输的监控系统，其具有远程控制、监控便捷和易于扩展的优点。网络视频监控系统的优点正好克服了传统的模拟闭路电视监控诸如传输距离近、系统扩展能力差的局限性。

首先，数字化视频可以在计算机网络（局域网或广域网）上传输图像数据，基本上不受距离限制，信号不易受干扰，可大幅度提高图像品质和稳定性；其次，数字视频可利用计算机联网，网络带宽可以复用，无须重复布线，还便于集中监视，集中控制。

发展到今天，视频监控系统的发展经历了三个阶段，并朝着第四代智能监控系统演进。

第一代：模拟视频监控系统

20世纪90年代以前，主要是以模拟设备为主的闭路电视监控系统。系统由摄像机、视频矩阵、磁带录像机和监视器等组成。

第二代：数字视频监控系统

20世纪90年代中期，随着计算机处理能力的提高和视频监控技术的发展，人们利用计算机的高速数据处理能力进行视频的采集和处理，利用显示器的高分辨率实现图像的多画面显示，从而大大提高图像质量。第二代数字监控以DVS、DVR为代表，其主要特点是图像经过编码压缩后再进行传送。

第三代：网络视频监控系统

进入21世纪，随着网络技术的发展，数字视频监控系统进一步发展成为具有网络功能视频监控系统。其集成了多媒体技术、数字图像处理技术以及远程网络传输等最新技术，可以实现图像传输、远程控制、现场信号采集等监控功能，提供高质量的监控图像和便捷的监控方式。

第四代：智能视频监控系统

智能视频监控系统通过在监控系统中增加智能视频分析模块,借助计算机强大的数据处理能力,对视频进行一系列分析，从视频中提取运动目标信息，发现感兴趣目标与事件，并根据预设模板或用户预设的规则，自动识别出感兴趣目标并得到感兴趣数据，将这些信息及时反馈给监控员。[1]

其中，第四代只能视频监控系统以网络为基础，以数字视频的压缩、传输、存储和播放为核心，以智能实用的图像处理和分析为特色，引发了视频监控的技术革命，受到了学术界、产业界和使用部门的高度重视。[2]

1.2 国内外研究现状

随着今年来技术的发展，视频监测正在从以前的模拟监测到现在的数字监测；从现场监测到远程监测；从有人值守监测到现在的无人值守监测，视频监测正朝着数字化、网络化、集成化的嵌入式监测方向蓬勃发展。

数字化是网络化的前提，网络化又是系统集成化的基础，所以，视频监测发展的最大两个特点是数字化和网络化。数字化是21世纪的特征，是以信息技术为核心的电子技术发展的必然。系统的网络化在某种程度上打破了布控区域和设备扩展的地域和数量界限。

目前国内还较多地使用本地模拟图像监控系统，主要是以模拟设备构成的闭路监控系统。系统通常由前端和监控中心构成。前端设备有摄像机、云台等；监控中心设备包括监视器、控制设备、模拟录像机等。虽然本地模拟图像监控系统能够保证采集得到的图像清晰、不失真，但是存在着传输距离小、系统可扩展性差、后期处理能力不强以及无法利用现有网络等缺点。

为了克服上述模拟视频监控的缺陷，出现了基于嵌入式技术的网络视频监控系统。

1.3 课题研究的主要内容

由于本文提出的视频监控系统所使用的为C/S架构，故而系统整体分为两个部分：流媒体服务器端和流媒体接收终端。在流媒体服务器端主要的研究内容包括：

通过摄像头采集视频数据；

基于H.264标准进行的视频数据的编码压缩；

基于RTP/RTCP协议的流媒体数据的网络传输；

而在流媒体客户端研究的主要内容包括：

利用安卓（Android）平台提供的JNI接口进行混合编程；

流媒体视频数据的接收与回放技术。

第二章、流媒体视频服务器的设计与实现

流媒体服务器在整体的视频监控系统中起到了最为主要的作用。其主要任务是通过外接的视频采集设备获取到原始的、未经处理的视频数据，然后在服务器内部完成视频的编码压缩，最终通过网络传输发送视频流媒体数据。鉴于流媒体服务器在视频监控系统中的重要性，其设计与实现就显得尤为，故而本文提出了如图一所示的流媒体服务器的软件架构。

图一

图中所示视频采集子系同利用V4L2(Video for Linux Two)接口通过外接的USB摄像头采集原始未经处理的视频数据，并送给视频编码压缩子系统；视频编码压缩子系统接收原始视频数据，并按照H.264标准进行视频数据的编码压缩；视频网络传输子系统接收以编码的视频数据并进行封装，之后向外发送形成视频流数据。

2.1 视频采集子系统

2.1.1 基于V4L2视频采集

Video4Linux（简称V4L）是由Linux操作系统提供的设备驱动框架以及应用编程接口的集合，其主要用于视频采集设备的驱动开发以及实时视频应用的开发。它为针对视频设备的应用程序编程提供一系列接口函数，这些视频设备包括现今市场上流行的TV卡、视频捕捉卡和USB摄像头等[3]。V4L2作为V4L接口的第二版修正了第一版当中的设计缺陷，并且提供了更为广泛的设备兼容性，支持更多的视频设备。

V4L2相比于V4L有更好的扩展性和灵活性，V4L2对V4L进行了彻底的改造，因而两者并不兼容。V4L2相较于V4L的最大优势是可以自定义设备驱动缓存数量，支持的硬件设备更多。V4L2最早出现在Linux 2.5 版本中，Linux 2.6发行版之后采用的接口大多是V4L2。

V4L2在视频采集中两个重要的系统调用时ioctl()和mmap()，ioctl()系统调用对设备的I/O通道进行控制，mmap()系统调用使得进程之间通过映射同一个普通文件来实现内存共享。普通文件被映射到进程地址空间后，进程可以像访问普通内存一样对文件进行访问，直接读写内存，不需要任何数据的拷贝，不必再调用read()、write()等操作，加速了I/O访问。采用共享内存的方式，大大提高了效率。

基于V4L2视频采集的软件流程图如图2所示。

图2

V4L2视频采集步骤主要分为如下过程：

（1）打开视频采集设备，获取该设备的文件描述符。视频采集可通过应用程序使用阻塞方式或非阻塞方式打开。如果使用非阻塞方式，即使数据尚未准备完成，驱动程序将把已经缓存的视频数据返回给应用程序，但是由于已缓存的视频数据可能并非所需的有效数据，故而可能造成数据错误。如果使用阻塞方式，只有在数据准备完成之后，驱动程序可以将已缓存的数据返回给应用程序。本文中所采用的方式为非阻塞的方式。

（2）查询设备兼容性，即查看设备所具有的功能，如是否支持流采集、所支持视频捕获的格式等。

（3）设置视频帧的参数，本文中将视频帧的宽设为640像素，高设为480像素，像素格式设置为YUYV，即YUV422的采样方式。

（4）准备视频采集。首先向驱动申请缓存区，并使用mmap()函数将申请到的缓存区映射到用户空间，这样不必复制即可直接操作采集到的视频帧。

（5）将申请到的缓存区放入视频采集队列，用来存放采集到的数据。

for(int i = 0; i < n\_buffers; ++i){

struct v4l2\_buffer buf;

CLEAR(buf);

buf.type = V4L2\_BUF\_TYPE\_VIDEO\_CAPTURE;

buf.memory = V4L2\_MEMORY\_MMAP;

buf.index = i;

if(-1 == xioctl(fd, VIDIOC\_QBUF, &buf))

errno\_exit("VIDIOC\_QBUF");

}

（6）开始采集视频。首先将数据写入申请的缓存区内，当一个缓存区的数据完成采集之后，驱动就会将其释放进输出队列，等待应用程序的处理。当所有的缓存区都进入输出队列后，驱动将停止采集，并等待缓存区重新放入采集队列。读取数据时，首先需要将一个缓存区出队列，驱动会从输出队列取出一个缓存区，并将其需要赋值给buf.index域，应用程序可以通过memcpy()函数将缓存区的数据复制到另一内存区域中。复制完成后，需要将缓存区重新放入采集队列队尾，如此即可实现循环采集。

（7）停止采集，首先使用VIDIOC\_STREAMOFF命令标识符停止视频采集，然后调用close()函数关闭视频采集设备。

2.1.2 YUV色彩空间

YUV色彩空间主要用于优化彩色视频信号的传输，使其向后兼容老式黑白电视。与RGB视频信号传输相比，它最大的优点在于使用的频宽极少，因为RGB视频信号要求三个独立的视频信号同时传输。其中“Y”表示明亮度（Luminance或者Luma），也就是灰阶值；而“U”和“V”表示的则是色度值（Chrominance或Chroma），作用是描述影像色彩及饱和度，用于指定像素的颜色。

YUV格式通常有两大类：打包（packaged）格式和平面（planar）格式。打包格式将YUV三个分量存放在同一个数组中，通常是几个相邻的像素组成一个宏像素；而平面格式使用三个数组分别存放Y、U、V三个分量，类似于一个三维平面。本文系统中使用的视频采集设备捕获到的视频数据为打包方式的YUV422。（详细说明YUV422的采样排列，存储排列，具体可查笔记本^\_^）

而在视频压缩中所使用编码库x264只接受平面格式的YUV420。（详细说明YUV420的采样排列，存储排列，具体可查笔记本^\_^）

因此，需要将打包格式的yuv422视频数据转换为平面格式的YUV420视频数据。

2.2 视频压缩子系统

通过视频采集设备捕获的原始数据含有大量的冗余信息，从而导致原始的视频数据量非常难以通过网络传输，即使可以传输也会耗费过多的网络带宽资源。因此需要特殊的视频编码压缩算法来对采集到的原始视频数据进行数据压缩，以实现视频数据的网络传输。

对于数据压缩技术而言，最基本的要求就是要尽量降低数字化后的数据量，同时要保持一定的信号质量。因此，所谓数据压缩，就是指在一定的数据存储空间要求下，将相对庞大的原始数据，重组为满足前述空间要求的数据集合，使得从该数据集合中恢复出来的额数据，能够与原始数据相一致，或者能够获得与数据一样的品质。[5]

目前主流的视频编码压缩标准主要有：MPEG-2，MPEG-4，H.264/AVC以及谷歌公司所提出的VP8标准。与其他压缩标准相比，在相同视频图像质量情况下，H.264能够节省50%左右的码率；同时，H.264/AVC标准在各种码率下能够提供稳定的视频质量，特别是在低码率条件下，也能够保持较高质量的视频图像。因此，H.264/AVC标准逐渐在实际应用中成为主流的编码压缩标准。

2.2.1 H.264/AVC标准的研究

H.264标准对网络传输具有非常好的支持功能。它引入了面向IP包的编码机制，有利于网络中的分组传输，支持网络中视频的流媒体功能。H.264具有较强的抗误码特性，可适应丢包率高、干扰严重的无线信道中的视频传输。H.264支持不同网络资源下的分级编码传输，从而获得平稳的图像质量。H.264能适应于不同网络中的视频传输，网络亲和性好。

该标准的压缩系统有视频编码层（Video Coding Layer，VCL）和网络抽象层（Network Abstraction Layer，NAL）两部分组成。VCL中包括VCL编码器与VCL解码器，主要功能是视频数据压缩编码和解码，它包括运动补偿、变换编码、熵编码等压缩单元。NAL则用于为VCL提供一个与网络无关的统一接口，它负责对视频数据进行封装打包后使其在网络中传送，它采用统一的数据格式，包括单个字节的包头信息、多个字节的视频数据与组帧、逻辑信道信令、定时信息、序列结束信号等。包头中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。类型标志用于指示图像数据的类型。

H.264标准可分为三挡：基本档次（其简单版本，应用面广）；主要档次（采用了多项提高图像质量和增加压缩比的技术措施，可用于SDTV、HDTV、和DVD等）；扩展档次（可用于各种网络的视频流传输）。H.264具有以下的技术特点。

1. 帧内预测编码

帧内编码用来缩减图像的空间冗余。为了提高H.264帧内编码的效率，在给定帧中充分利用相邻宏块的空间相关性，相邻的宏块通常含有相似的属性。因此，在对一给定宏块编码时，首先可以根据周围的宏块预测（典型的是根据左侧和上侧的宏块，因为这些宏块已经被编码处理），然后对预测值与实际值的差值进行编码，这样，相对于直接对该帧编码而言，可以大大减小码率。

2. 帧间预测编码

帧间预测编码利用连续帧中的时间冗余来进行运动估计和补偿。H.264的运动补偿支持以往的视频编码标准中的大部分关键特性，而且灵活地添加了更多的功能，除了支持P帧、B帧外，H.264还支持一种新的流间传送帧——SP帧，码流中包含SP帧后，能在有类似内容但有不同码率的码流之间快速切换，同时支持随机接入和快速回放模式。

3. 不同大小和形状的宏块分割

对每一个16x16像素宏块的运动补偿可以采用不同的大小和形状，H.264支持七种模式。小块模式的运动补偿为运动详细信息的处理提高了性能，减少了方块效应，提高了图像的质量。

4. 高精度的亚像素运动补偿

在H.263中采用的是半像素精度的运动估计，而在H.264中可以采用1/4像素或者1/8像素精度的运动估值。在要求相同精度的情况下，H.264使用1/4像素或者1/8像素精度的运动估计后的残差要比H.263采用半像素精度运动估计后的残差来得小。这样在相同精度下，H.264在帧间编码中所需的码率更小。

5. 多帧预测

H.264提供可选的多帧预测功能，在帧间编码时，可选五个不同的参考帧，提供了更好的纠错性能，这样更可以改善视频图像质量。这一特性主要应用于以下场合：周期性的运动、平移运动、在两个不同的场景之间来回变换摄像机的镜头。

6. 区块滤波器

H.264定义了自适应去除块效应的滤波器，这可以处理预测环路中的水平和垂直块边缘，大大减少了方块效应。

7. 整数变换

在变换方面，H.264使用了基于4x4像素块的类似于离散傅里叶变换（DCT）的变换，但使用的是以整数为基础的空间变换，不存在反变换因为取舍而存在误差的问题。与浮点运算相比，整数DCT变换会引起一些额外的误差，但因为DCT变换后的量化也存在量化误差，与之相比，整数DCT变换引起的量化误差影像并不大。此外，整数DCT变换还具有减少运算量和复杂度，有利于向定点DSP移植的优点。

8. 量化

H.264中可选32种不同的量化步长，这与H.263中有31个量化步长很相似，但是在H.264中，步长是以12.5%的复合率递进的，而不是一个固定常数。在H.264中，变换系数的读出方式也有两种：之字形（Zigzag）扫描和双扫描。大多数情况下使用简单的之字形扫描，双扫描仅用于较小量化级的块内，有助于提高编码效率。

9. 熵编码

视频编码处理的最后一歩是熵编码，在H.264中采用了两种不同的熵编码方法：通用可变长编码（UVLC）和基于文本的自适应二进制算术编码（CABAC）。

在H.263等标准中，根据要编码的数据类型如变换系数、运动矢量等，采用不同的可变长编码码表。H.264中的UVLC码表提供了一个简单的方法，不管符号表述什么类型的数据，都使用统一变字长编码表。其优点是简单；缺点是单一的码表是从概率统计分布模型得出的，没有考虑编码符号间的相关性，在中高码率时效果不是很好。

因此，H.264中还提供了可选的CABAC方法。算术编码使编码和解码两边都能使用所有句法元素（变换系数、运动矢量）的概率模型。为了提高算术编码的效率，通过内容建模的过程，使基本概率模型能适应随视频帧率而改变的统计特性。内容建模提供了编码符号的条件概率估计，利用合适的内容模型，存在于符号间的相关性可以通过选择目前要编码符号邻近的已编码符号的相应概率模型来去除，不同的句法元素通常保持不同的模型。

2.2.2 x264库介绍

（详细介绍x264库的使用，可参考leixiaohua的csdn博客内容）

2.3 视频传输子系统设计

传统互联网的应用主要是传输文本或简单图形，传输数据量较小，对数据传输的实时性、同步性要求不高。但是通过流媒体技术所传输的对象主要是以音频内容为主体的多媒体信息，传统的互联网传输协议无法保证这些数据的准确、迅速和实时传送，必须出现新型的网络传输协议以满足高带宽、低传输延迟、支持组播模式和通道同步等特点[6]，才能保证流媒体技术在网络上的广泛应用。当前，互联网工程任务组（IETF）已经设计出几种支持流媒体传输的协议，主要有实时传输协议RTP（Real-time Transport Protocol）、实时传输控制协议RTCP（Real-time Transport Control Protocol）、实时流协议RTSP（Real-time Streaming Protocol）等。

2.3.1 RTP实时传输协议的研究

实时传输协议RTP被定义为传输音频、视频、模拟数据等实时数据的传输协议。最初设计是为了数据传输的多播，但是其也可以用于单播。与传统的注重高可靠的数据传输的传输层协议相比，它更加侧重数据传输的实时性。必须注意的是，RTP没有提供仁和区确保按时传送数据的机制，也没有提供任何质量保证的机制，因而要实现服务质量（Quality of Service，QoS）必须由下层网络来提供保证。同样需要需要注意的是，RTP同样不保证数据包序号传送，即使在下层网络能保证可靠性传送的条件下，也不保证数据包按序号传送。包含在RTP中的序号可供接收方用于重构数据包序列，也可用于包的定位[7]。

RTP协议负责对流媒体数据进行封包并实现实时传输，每一个RTP数据分组都由头部和负载两部分组成，其中头部前12个字节的含义是固定的，而负载则可以是音频或者视频数据。每一个RTP包头数据格式如图三所示。

图三

V：版本号，2比特，标识RTP协议的版本号，要求通信双方的版本号相同，当前版本号为2；

P：填充标识，1比特，表明数据区后是否有填充域，若该位被置位，则报文中就携带一位或者多位的填充字节，这些填充字节在报文的尾部，不算做有效载荷；

X：扩展位，1比特，表明在固定头部后是否跟有扩展头，若该位被置位，则说明在固定报头后要跟且仅跟一个报文头扩展；

CC：4比特，表示CSRC标识符在固定头后的数量；

M：标志位，1比特，该位往往用来设置一帧视频图像中具有相同时间戳的一组数据包的最后一个。这样不必等到下个数据包的到来就能够判断一组数据包已经传输完毕；

Timestamp：时间戳，32位，表示RTP数据包中第一个字节的采样时间。发送端在数据包中插入的一个即时时间标记，随时间的推移而增加。当数据包到达接受端后，接收端根据时间戳重新建立原始音视频的时序，也可用于同步多个不同的数据流，帮助接收方确定数据到达时间的一致性；

Sequence number：顺序编号，16位，每发送一个RTP数据包，顺序号就加1。RTP协议通常使用UDP协议传输数据，但是UDP协议发送数据包时没有时间顺序，这就会造成数据包到达接收端时可能与发送端发出的数据包的顺序不同，为了对抵达的数据包进行重排就需要一种时间标记来进行标识。因此顺序编号主要是用来排序RTP分组，以消除重复分组，保持视频和音频流连续地播放；

Payload Type：负载类型，7位，标识RTP分组中有效负载的格式。RTP可支持128种不同的有效载荷类型，可以根据需要更多的负载类型。对于音频流，这个域用来表示声音使用的编码类型；对于视频流，有效载荷类型可以用来表示视频编码的类型。如果发送端在会话或者广播的中途决定改变编码方法，发送端可通过这个域来通知接收端。

RTP所能支持的声音有效载荷类型如表1所示

表1（查阅书本）

RTP所能支持的视频有效载荷类型如表2所示

表2（查阅书本）

SSRC：同步源标识，32位，表示RTP数据包流的来源，RTP会话的每个数据包流都有一个清楚的SSRC

CSRC：贡献源标识，32位，确认多个包中的同步源。该域最多可有15项，每项长度位32位，列举出产生该RTP数据包的所有源。

RTP协议的数据分组和控制分组使用相邻的不同端口，这样大大提高了协议的灵活性和处理的简便性。RTP协议本身包括两部分：RTP数据传输协议和RTCP传输控制协议。为了可靠、高效地传送实时数据，RTP和RTCP必须配合使用，通常RTCP包的数量占所有传输量的5%。RTCP协议作为RTP协议的一个重要的控制补充协议，以它的反馈机制实现对流媒体服务的QoS控制，配合传输层协议，保证了流媒体的实时性特征，满足了在IP网上对QoS的需求。

2.3.2 RTCP实时传输控制协议

RTCP协议是RTP协议的补充协议，是一个控制协议，其与RTP协议共同合作，为顺序传输数据包提供可靠的传送机制，并对网络流量和阻塞进行相关控制。在RTP回话期间，每个参与周期性地彼此发送RTCP控制包，包中封装了已发送的数据包的数量、丢失的数据包的数量、包抖动等发送端或接收端的统计信息。RTCP在流媒体传输中的主要作用为QoS管理与控制、媒体同步以及附加信息的传递。

RTCP数据包是控制包，由固定头和可变长结构元素组成，以一个32位边界结束。RTCP包可堆叠，不需要任何分隔符可将多个RTCP包连接起来形成一个RTCP组合包，以低层协议用单一包发送出去。由于需要低层协议提供整体长度决定组合包的结尾，在组合包中没有单个RTCP显式计数。RTCP报头如图四所示

图四

Version：识别RTCP协议版本号。RTP数据包中的该值与RTCP数据包中的一致。当前规定值为2；

P：间隙，填充位（Padding）；

RC：接收方报告计数。接受放报告块的编号包含在该数据包中。

Packet Type：包括常量200，识别一个RTCP SR数据包。

在RTCP通信控制中，RTCP协议的功能是通过不同的RTCP数据报来实现的，主要有以下几种类型：

1.SR：发送端报告，发送端是指发出RTP数据报的应用程序或者终端，发送端同时也可以是接收端。

2.RR：接收端报告，接收端是指仅接收但不发送RTP数据报的应用程序或者终端。

3.SDES：源描述，主要功能是作为会话成员有关信息的载体，如用户名、邮箱地址、电话号码等，此外还具有向会话成员传达会话控制信息的功能。

4.BYE：通知离开，主要功能是指示某一个或者几个源不再有效，即通知会话中的其他成员自己将退出会话。

5.APP：由应用程序自己定义，解决了RTCP的扩展性问题，并且为协议的实现者提供了很大的灵活性。

2.3.3 RTSP实时流协议

RTSP实时流协议，是一种应用层协议，用于实时地控制数据的传输。RTSP提供一个可扩展的架构来实现实时媒体资源的在线点播，如视频或音频内容。数据源可以是直播信号也可以是制作好的音视频文件。RTSP能够同时控制多个RTP的会话过程。

RTSP实时流媒体协议本身并不进行数据流的传输，而只是对数据流的传输进行控制，它包括了对RTSP连接进行建立以及类似于VCR（什么VCR）的会话控制功能，包括播放、暂停、快进、快退等。RTSP可以选择使用TCP或者UDP进行连接，因为RTSP只是发送控制信息，一般使用TCP的方式。RTSP与HTTP类似，但HTTP是一种无状态的协议，而RTSP是一种有状态的协议，客户端和服务器都可以互相发送RTSP请求。

RTSP请求报文的方法包括：OPTIONS、ANNOUNCE、DESCRIBE、SETUP、TEARDOWN、PLAY、RECORD、PAUSE、GET\_PARAMETER和SET\_PARAMETER等[8]。

2.3.4 网络传输系统的设计实现

（jrtplib库的代码内容及流程图）

第三章、流媒体视频客户端的设计与实现