密级：　　保密期限：

北京郵電大學

硕士学位论文

题目：纠删码和可靠 UDP 相结合的无线视传输技术研究与实现

学　　号：2012110649

姓　　名：徐盈盈

专　　业：计算机科学与技术

导　　师：马华东

学　　院：计算机学院

2015 年 1 月 12 日

独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：_____ 日期：_____

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

保密论文注释：本学位论文属于保密在__年解密后适用本授权书。非保密论文注释：本学位论文不属于保密范围，适用本授权书。

本人签名：_____ 日期：_____

导师签名：_____ 日期：_____

纠删码和可靠 UDP 相结合的无线视频传输技术研究与实现

摘　要

通过无线网络进行音视频等多媒体数据的传输是目前的一个研究热点。然而，无线信道本身存在误码率高、延迟大、传输速率低等缺陷，因此如何提高传输可靠性一直是无线视频传输研究的重点和难点。纠删码是近年来被应用到视频传输领域的一种容错技术，具有不存在数据重传时延的优点，然而其纠错能力有限且受信道影响大；另一方面，可靠 UDP 协议在传统 UDP 协议的基础上添加了传输可靠性，其信道适应性强但数据重传时延大。因此，将纠删码和可靠 UDP 协议结合起来进行研究可以实现二者的优势互补。

首先，我们利用 UEP（Unequal Error Protection）Raptor 作为纠删码，它具有标准 Raptor 码的无码率限制特性、线性的译码复杂度以及良好的数据容错性等优势，并在此基础上保持 GOP（Group Of Pictures）总的冗余度不变，同时增大其中关键帧的冗余度，从而对视频数据提供了不等差错保护。其次，我们从连接机制、确认机制和重传机制等方面实现了应用程序级的可靠 UDP 协议，并将其应用于无线视频传输过程。最后，设计视频发送端根据接收端的反馈信息进行关键帧的数据重传以及发送速率和 Raptor 编码强度等参数调整，以此将 UEP Raptor 和可靠 UDP 协议结合起来，共同保证无线视频传输的可靠性。

本文搭建了无线视频传输系统，对所提出的方法进行了测试和分析，表明 UEP Raptor 和可靠 UDP 协议相结合的传输技术能够有效地改善无线网络中视频的实时传输质量。

关键词：纠删码　UEP Raptor　可靠 UDP　视频传输　无线网络

RESEARCH AND IMPLEMENTATION OF COMBINATION OF ERASURE CODE AND RELIABLE UDP FOR VIDEO TRANSMISSION IN WIRELESS NETWORKS

ABSTRACT

Nowadays, the transmission of audio, video and other multimedia data over wireless networks has become a hot topic. However, due to various defects of wireless channels, including high bit error rate, large transmission latency and limited actual transmission rate, it is a key and difficult problem to improve the video transmission reliability in wireless networks. Recently, as a kind of fault-tolerant technique, erasure codes are used for video transmission in wireless networks, which do not have retransmission latency, but have limited error-correction abilities and are very sensitive to wireless channels. On the other hand, reliable UDP is based on traditional UDP, which can guarantee transmission reliability and has strong channel adaptability but large retransmission latency. Therefore, the combination of erasure codes and reliable UDP will achieve their complementary advantages.

First, we use UEP Raptor as the erasure code, which has advantages of standard Raptor code, including unlimited data rate, linear complexity and good error tolerance. Besides, by keeping the GOP (Group Of Pictures) redundancy invariant but increase the redundancy of the key frames in the GOP, UEP Raptor provides video transmission with unequal error protection. Secondly, we realize reliable UDP in the respects of connection mechanism, acknowledgement mechanism and retransmission mechanism as an application layer protocol, and use it for video transmission in wireless networks. Finally, we design the video sender to

retransmit the lost data packets of key frames in the GOP, and adjust the data transmission rate and redundancy of Raptor code according to the feedback information of the video receiver. In this way, UEP Raptor and reliable UDP are combined to guarantee the reliable video transmission in wireless networks.

We build a wireless video transmission system to test and analyze our proposed method. The results show that the combination of UEP Raptor and reliable UDP can effectively improve the real-time quality of video transmission over wireless networks.

KEY WORDS: erasure code, UEP Raptor, reliable UDP, video transmission, wireless networks

目录

第一章　绪论

1.1　课题的研究背景

自 1997 年出现的第一个无线局域网标准 IEEE802.11（Institute of Electrical and Electronics Engineers802.11）至今，无线网络的峰值传输速率经历了 2Mbps 到 3.6Gbps 的历史变迁[1-2]。传输速率的提升，加速了视频、音频等多媒体数据通过无线网络进行远距离传输的步伐。近年来 3G、4G 和卫星等无线技术以更加便捷灵活的网络接入方式促进了无线网络和骨干网的融合，提升了大众的视频通信体验，进一步推动了无线视频通信的发展。

无线视频的发展历史可以追溯到 20 世纪 70 年代利用红外线传输技术的无线家庭影院[3]，此后随着 1999 年后出现的一系列 WiFi 标准，无线家庭影院的通信距离和传输速率都得到了很大提升。自 2008 年以来 3G、4G 逐步完成了从国际标准到市场化技术的转变，以此为契机无线视频通话逐渐得到普及[4]。2012 年华为推出了能够与 IBMSametime、微软 Lync、以及 Skype 网络电话等平台互联互通的视频通信产品，从而实现了方便的无线视频移动办公[5]。近年来无线视频监控系统发展迅速，现已深入交通监控、工业监控、安全监控和家庭监控等众多领域[6]。综上可知，无线视频通信在给人们带来更为直观的视觉体验的同时，加速了人们生活的数字化和智能化进程。并且随着更新更强的无线传输技术和移动通信标准的到来，无线视频传输无疑将拥有越来越大的发展空间和越来越广的应用前景。

尽管无线视频传输发展迅速，无线信道本身依然存在着抗干扰能力差、实际可用带宽有限、数据传输延迟大等固有缺陷，同时视频数据具有量大、对时延和传输错误敏感等特性，如何提高数据的可靠性一直是无线视频传输研究的重点和难点。针对这一问题，人们提出了一些研究方案，基于数据容错的信道编码和基于数据重传的可靠 UDP 是其中两种较新研究成果。然而，信道编码的纠错性能有限并且受无线信道影响大，数据重传的时延较大无法满足视频传输的实时性要求。现阶段无线视频传输效果仍然有待改善和提高。

1.2　课题的研究意义和目标

随着无线网络的发展，家庭影院、视频会议、视频电话等多种无线视频应用层出不穷。此外，3G 技术的诞生促使视频监控迎来了发展的新高峰。现阶段视频监已控融合互联网、移动通信和安全防范于一体，并以其独有的高移动性越来越多地作用于工厂、交通、家庭等安全防护方面以及水质监测、古文化遗产监测

等资源保护方面[6]。由此可见，无线视频传输技术的研究具有重要的现实意义。

尽管无线网络的数据峰值传输速率得到了很大提升，由于网络设备昂贵、峰值速率对网络环境要求过分理想等原因，实际的无线网络数据传输速率仍然无法满足实时流畅的视频传输需求。此外，无线信道还存在着噪声干扰、带宽有限和延迟明显等固有缺陷。加之视频数据本身数据量大、对实时性要求较高以及差错数据对画面显现影响大等，如何结合无线信道特点保证视频数据的实时可靠传输是无线视频传输亟待解决的问题之一。因此，无线视频传输技术的研究具有重要的理论意义。

本课题将对无线视频传输的特性和现阶段主要使用的传输技术进行研究和分析，选择使用纠删码抵制无线网络中的删除错误[7]，同时利用可靠 UDP[16]降低网络丢包引起的传输质量下降，最终将二者相结合得到一种对无线网络带宽要求低且实时性好、可靠性高的无线视频传输方案。

1.3 课题的研究现状

纠删码是常用的信道编码之一，主要用来纠正位置可知的传输错误，其中喷泉码打破了传统纠删码的固定码率限制，从而适用于任何网络情形[7]。Raptor 码是喷泉码的一种具体实现，它具有线性的编译码复杂度和良好的容错性，现已逐渐应用到视频传输领域[9-12]。

2009 年 David Goz ález 等人提出将 Raptor 作为一种应用层的前向纠错（FEC—Forward Error Correction）技术应用到移动电视广播方面，为视频流服务提供一种针对突发错误的保护措施[8]。2011 年 Christos Bouras 将 Raptor 应用于 LTE（Long Term Evolution）移动通信系统，以提高视频等多媒体流服务的健壮性[9]。2013 年 Shiuan-Tung Chen 等人将应用层 Raptor 码应用于 WiFi 广播信道，从而恢复 WiFi 广播过程中包括视频等多媒体数据在内的丢失数据[10]。考虑到经压缩后的视频数据按照重要性程度可分为关键数据和非关键数据这一客观事实，为了更加充分地利用网络带宽并提高接收端的整体视频质量，人们提出了具有不等差错保护特性的 Raptor 码，即 UEP Raptor。刘国等人于 2013 年提出了基于 URP-Raptor 的视频图像压缩传输，并指出该方法适用于不同的信道环境[11]。Yeqing Wu 等人于 2014 年将 UEP Raptor 和 RCPC（Rate Compatible Punctured Convolutional）一起运用到无线视频传输上，其中 UEP Raptor 运行在应用层帮助提供差错保护，RCPC 运行在物理层对数据分组提供优先级控制[12]。不难看出，UEP Raptor 的研究正在逐步向无线视频传输领域发展。

UDP 协议是无线视频主要使用的传输协议，但它无法保证数据传输的可靠性[13-15]，于是人们提出了可靠 UDP 协议[16]。可靠 UDP 协议使用 UDP 协议进行数据传输，但在应用层通过虚拟连接、数据包确认、数据包自动请求重传以及流

量管理等机制提高了数据传输的可靠性。自 1999 年可靠 UDP 草案[16]提出以来，可靠 UDP 逐渐发展并应用到视频图像传输方面。Bova T 等人于 2009 年将可靠 UDP 用于具有高移动性的机载网络并通过无线信道进行视频通信[17]。2013 年美国三星研究院和加利福尼亚大学计算机科学与工程系联合推出了具有流量控制的可靠 UDP，使应用层和传输层相结合共同改善无线网络中的视频传输质量[18]。此外，国内也出现了一些基于可靠 UDP 的无线视频传输技术的研究，并取得了一定的研究成果[19-21]。

在无线网络中进行视频实时传输，若仅使用纠删码，虽然数据不存在重传时延但纠错能力有限而且容错性能受信道影响大，传输效果不稳定。若只依赖可靠 UDP 协议，则信道适应性强但数据重传时延大，无法很好地满足视频传输的实时性要求。由此可见，将纠删码和可靠 UDP 结合起来可以实现它们的优势互补，共同促进无线视频的传输效果。尽管近年来国际上陆续出现了一些将前向纠错编码和重传机制相结合的无线视频传输方法[22-24]，但将 UEP RaptorRaptor 码和可靠 UDP 相结合的传输技术至今鲜少有人涉及。因此，本文将对 UEP Raptor 码和可靠 UDP 相结合的无线视频传输技术进行研究和实现。

1.4　论文主要内容及结构

本文的内容主要包括以下几个方面：无线视频传输系统的设计、纠删码的研究、相关传输协议的研究、纠删码和可靠 UDP 相结合的研究、实验设计以及实验结果的对比和分析。其中，无线视频传输系统为本文研究的传输技术提供了具体的运行场景，主要从发送端和接收端两个方面进行设计和实现。Raptor 码作为本文的重点研究对象之一，本质上属于纠删码。而纠删码是一个相对广泛的概念，它包含 RS 码、LDPC 码以及喷泉码等多种常见的编码体系，每种编码体系下又含有一些具体的算法实现。因此，首先从编、解码原理、算法复杂度和适用场景等方面对各种编码算法进行调研，然后将各种编码算法进行比较从而说明选择 Raptor 码作为本课题研究对象的原因，最后对本课题中使用的 UEP Raptor 编码算法进行详细的设计和实现。数据重传是本文研究课题的另一个重要理论依据，所以在介绍相关传输协议（指控制传输过程的协议，并非指传输层的协议）之前首先简述自动请求重传的基本原理和分类。相应地，相关传输协议介绍部分选取具有数据重传特性的 TCP 协议和可靠 UDP 协议进行研究，并从协议工作原理、协议本身的优缺点以及协议的适用场景三个方面对二者进行简要说明和比较。针对本课题中使用的可靠 UDP，给出详细的功能设计和技术实现。最后，搭建实验环境、设计实验方法，记录实验结果，并对结果进行统计和分析，得出最终的研究结论。

本文的章节结构如下：

第一章绪论。这部分首先介绍了课题的研究背景，然后从无线视频的应用场景、发展前景以及现阶段仍然存在的问题等方面突出课题的研究意义，并据此提出课题的研究目标。最后，简要介绍了现阶段国内外无线视频传输技术的研究现状。

第二章无线视频传输技术概述。查阅与无线视频传输技术相关的资料，主要从编码和数据重传两个方向上对其进行分类和总结，在各个分类下又结合具体的工作原理和适用场景进行分析和比较。

第三章 UEP Raptor 和可靠 UDP 相结合的设计。首先给出了本文用来测试的无线视频传输系统的设计，旨在交代 UEP Raptor 和可靠 UDP 相结合的无线视频传输技术具体的使用场景。其次，从功能设计层面详细介绍了了 UEP Raptor 和可靠 UDP 两个部分。针对 UEP Raptor，本文主要介绍了编码算法、解码算法、不等差错保护算法、帧划分和重组算法。对于可靠 UDP，则重点描述了包结构、数据队列、确认机制、重传机制和局部可靠性等。最后，给出 UEP Raptor 和可靠 UDP 具体的结合方法。

第四章 UEP Raptor 和可靠 UDP 相结合的实现。这一章与第三章相呼应，针对第三章中重点介绍的算法，从实现层面给出详细的数据结构和功能模块。

第五章系统测试与结果分析。搭建实验环境，设计测试用例，并使用此前介绍的无线视频传输系统对 UEP Raptor 和可靠 UDP 相结合的传输方法进行系统地测试。记录实验结果，并给出结果对比和分析。最后，得出实验结论。

第六章总结与展望。总结课题完整的研究过程，分析课题研究结果的缺点和不足，并据此提出下一步优化的方向。

第二章　无线视频传输相关技术介绍

本文调研了最近十余年无线视频传输技术方面的研究成果，发现常用的无线视频传输技术大致可以分为基于编码的传输技术和基于重传的传输技术两大类，以下分别对其进行总结和比较。

2.1　基于编码的传输技术

通过摄像头等视频采集设备获取的原始视频数据量高达上万字节，将如此大量的视频原始数据直接通过网络进行传输显然是不现实的。此外，无线信道带宽有限、易受噪声干扰、容易产生丢包和延迟等固有缺陷，对实时视频在无线网络中的传输提出了新的挑战。针对以上问题，人们提出了编码理论。在数据传输领域，编码又可分为信源编码和信道编码[25]。

2.1.1 信源编码

多数现实世界的数据相互之间都存在着某种关系，因而不可避免地存在着信息冗余。信源编码[5]就是为减少信源输出数据中的冗余而提出的。

就视频数据而言，每一幅图像都由很多个像素点组成，各个像素点之间由于其空间位置而存在一定的相关性，即空间相关性。对这种相关性差异进行编码，通过已知一些像素点与位置差异的叠加就能够计算得到其他像素点。并且这种差异本身的数据量要比像素点小得多，所以大大减小了传输数据的长度，有效节约传了输带宽。此外，一个场景的多个图像之间往往也有很大的相似性，即空间相关性。同样地，使用后面图像和前面图像的差异代替后面图像本身的传输，也能降低信源输出符号的数量。利用视频数据的空间相关性和时间相关性进行压缩编码，是信源编码的重要理论依据。

具体到视频传输方面，H.264 是目前使用最普遍的信源编码，以下给出简要介绍。

H.264[26]是 2003 年由 ISO（International Organization for Standardization）和 ITU（International Telecommunication Union）共同发布的新一代视频压缩标准，主要定义视频编码规范。与此前的标准相比，H.264 的压缩效率高、带宽占用少、容错能力强且网络接口友好，已经应用于几乎所有的视频服务领域。

帧和宏块是 H.264 定义的两种数据结构。H.264 包含了帧内预测编码和帧间预测编码，分别用来消除图像的空间相关性和时间相关性。前者编码一帧内像素的预测值和实际值的差值，后者则编码同一宏块在不同帧间的位置差异。帧间预测又分为单向预测和双向预测。一副图像中多数是内容变化缓慢的平坦区域，即图像中低频区占大部分，高频区占小部分。通过一些很小的相关性变换系数就可

以实现图像从空间域到频域的转变。对这些变换系数进行的编码就是变换编码，离散余弦变换是常使用的变换编码。最后，利用图像的统计特性可以进行码率压缩，即熵编码，哈夫曼编码和算术编码是典型的熵编码算法。以上就是 H.264 主要的编码算法。

H.264 编码后得到 I 帧、P 帧和 B 帧三种帧类型。I 帧本身即是一副完整的图像，属于内部编码帧，可以独立解码。P 帧是前向预测帧，其解码需要参考前向的 I 帧。B 帧是双向参考帧，即解码过程需要前面和后面的 I 帧、P 帧甚至有可能是 B 帧的参与。

最后，H.264 相比于之前的严压缩标准还增加了许多新特点：信号化帧内预测模式、树状结构的运动补偿技术、加权预测方法、4×4 整数离散余弦变换技术、基于上下文的自适应二进制算术编码、优化了的熵编码、基于 Lagrangian 优化算法的控制模型等。在错误恢复方面，H.264 标准在继承前面标准中优秀错误恢复工具的同时，又进行了多种改进和创新。

2.1.2 信道编码

信道编码通过在原始数据中添加冗余信息来克服信道中的噪声干扰和数据丢失。具体即在信息码元中人为地加入一些监督码元，在接收端利用这些监督码元与信息码元之间的关系发现和纠正差错，从而提高信息码元传输的可靠性。

前向纠错码（FEC）是常用的信道编码，而用来纠正位置可知的错误的纠错码又称为纠删码[27]，纠删码具有良好的纠错功能。纠删码的原理可简述为：k 个源数据包经过发送方编码得到 n（$n>k$）个编码包，然后将这 n 个编码包进行网络传输，接收方只需收到一定数量的（如 m 个，$m>=k$）编码包就能够恢复出 k 个源数据包。令$x = (x_0, x_1, \cdots, x_{k-1})$表示 k 个源数据包，$y = (y_0, y_1, \cdots, y_{n-1})$表示 n 个编码包，则一个（k,n）线性纠删码还可以表示为 $y=xG$，其中 G 称为生成矩阵。

目前比较常用的纠删码有 RS（Reed-solomon codes）、低密度奇偶检验码 LDPC（Low Density Parity Check Code）和数字喷泉码等[27]，以下分别进行简要介绍。

（1） RS 码

在（k,n）纠删码的传输过程中，假设接收方收到 m（$m>=k$）个数据包，若使用其中任意 k 个数据包都能够重构出 k 个源数据包，那么就称这种纠删码为极大最小距离可分码，即 MDS（Maximum Distance Separable）码。

RS 码是一类纠错能力很强的 MDS 码，它既可以纠正突发错误又可以纠正随机错误。根据其生成矩阵的不同，RS 纠删码可以分为范德蒙码和柯西码。式 2-1 是范德蒙矩阵 G，满足条件——任意 k 列组成的子方阵G'都是非奇异矩阵。

$$G=\begin{bmatrix} g_0^1 & g_1^1 & g_2^1 & \cdots & g_{n-1}^1 \\ g_0^2 & g_1^2 & g_2^2 & \cdots & g_{n-1}^2 \\ g_0^3 & g_1^3 & g_2^3 & \cdots & g_{n-1}^3 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ g_0^k & g_1^k & g_2^k & \cdots & g_{n-1}^k \end{bmatrix} \tag{2-1}$$

使用范德蒙矩阵[8]生成的 RS 码是非系统码（编码后得到的前 k 个数据包是 k 个源数据包的码成为系统码），不具有系统码的优势。相比之下，使用柯西矩阵得到可以直接构造出系统码，而不必经过系统化操作。

$$G=\begin{bmatrix} \frac{1}{X_1+Y_1} & \frac{1}{X_1+Y_2} & \frac{1}{X_1+Y_3} & \cdots & \frac{1}{X_1+Y_n} \\ \frac{1}{X_2+Y_1} & \frac{1}{X_2+Y_2} & \frac{1}{X_2+Y_3} & \cdots & \frac{1}{X_2+Y_n} \\ \frac{1}{X_3+Y_1} & \frac{1}{X_3+Y_2} & \frac{1}{X_3+Y_3} & \cdots & \frac{1}{X_3+Y_n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \frac{1}{X_k+Y_1} & \frac{1}{X_k+Y_2} & \frac{1}{X_k+Y_3} & \cdots & \frac{1}{X_k+Y_n} \end{bmatrix} \tag{2-2}$$

式 2-2 是柯西矩阵[8]G，其中$X=\{X_1,X_2,\cdots,X_k\}$和$Y=\{Y_1,Y_2,\cdots,Y_n\}$为同一个有限域的两个元素子集，满足式条件：1）X 中任意元素和 Y 中的任意元素相加不为 0；2）X 中任意两个元素的值互不相等、Y 中任意两个元素的值互不相等并且 X 中任意元素和 Y 中任意元素互不相等。

范德蒙码和柯西码的编码时间复杂度均为$O(n^2)$，但前者的译码复杂度比后者高。因此，柯西码的总体复杂度低于范德蒙码。

（2） 低密度奇偶校验码

RS 码包含大量的矩阵运算，时间复杂度和空间复杂度都比较高，实现难度大。为了解决这个问题，人们提出了基于扩展图的低密度奇偶校验码（LDPC 码）。LDPC 码利用生成矩阵完成源数据到编码数据（信息码+校验码）的转变，对于该生成矩阵 G 存在一个与之对偶的奇偶校验矩阵 H，它标志着编码的校验特征。正是奇偶校验矩阵 H 的稀疏特性保证了 LDPC 码[8]的低密度特征。

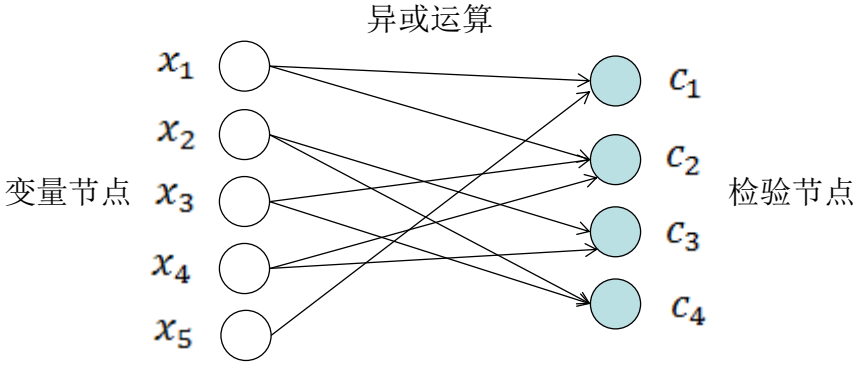


图 2-1　LDPC 码的二分图

图 2-1 简要展示了 LDPC 码的编码二分图，其中x_1-x_5成为变量节点，表示源数据包；c_1-c_4是校验节点，表示冗余包。源数据包按照一定的度分布算法得到

冗余包。发送端将源数据包和冗余包一齐发送出去，接收端则按照算法 2-1 进行译码。

算法 2-1

1. 找到所有度为 1 的检验节点，视为已译码的变量节点；
2. 将已恢复变量节点和与其存在连接关系的校验节点作异或运算，结果覆盖该检验节点；
3. 删除 2 中使用的连接，并将该校验节点的入度值减去 1；
4. 如果所有变量节点均已译码得到，则译码成功，退出；
   若仍存在未译码的变量节点，继续查找入度为 1 的校验节点；若存在这样的校验节点，则对应的变量节点视为已译码；反之，认为译码失败，退出；
5. 跳至步骤 2。

Tornado 码是一种广泛用于处理删除错误的 LDPC 码。度分布是 Tornado 码编码中非常重要的部分，下面对度分布进行简单介绍。

设 E 为 Tornado 码二分图的总边数，k 为变量节点数，β 为冗余度，λ_i和ρ_i分别表示度为 i 的变量节点和校验节点的与对应二分图总关联数的比值，可以推导出 Tornado 的度分布：

$$\lambda_i = \frac{1}{(\sum_{i=1}^{D} 1/i)(i-1)}, (i = 2, \cdots, D+1) \tag{2-3}$$

$$\rho_i = \frac{e^{-\alpha}\alpha^{i-1}}{(i-1)!}, (\mathrm{i} \geq 1) \tag{2-4}$$

限制条件如下：

$$\frac{\alpha e^{\alpha}}{e^{\alpha}-1} = \alpha_r \tag{2-5}$$

由于编码和译码过程都只用到简单的异或运算，Tornado 码的编码和译码的时间复杂度都接近$\mathrm{O}(n ln\, 1/\varepsilon)$，处理速度明显快于 RS 码。然而，与 RS 码一样，Tornado 码有固定的码率，因此在许多无码率的应用中并不适用。

（3） 喷泉码

数字喷泉码[27-28]是一类没有码率限制的纠删码。发送端使用 k 个源符号通过异或运算源源不断地生成编码符号，接收端只要从这些编码符号中收到任意 m 个（m 略大于 k），就能够以很高的概率恢复出全部的源符号。

（3.1）LT 码

LT 码[29]是最早实现的一种喷泉码。以下简要介绍其编码和译码过程。

图 2-2 为 LT 码的编码二分图。其中，左侧表示源数据包，右侧表示根据某种度分布得到的编码包，连线表示异或运算。

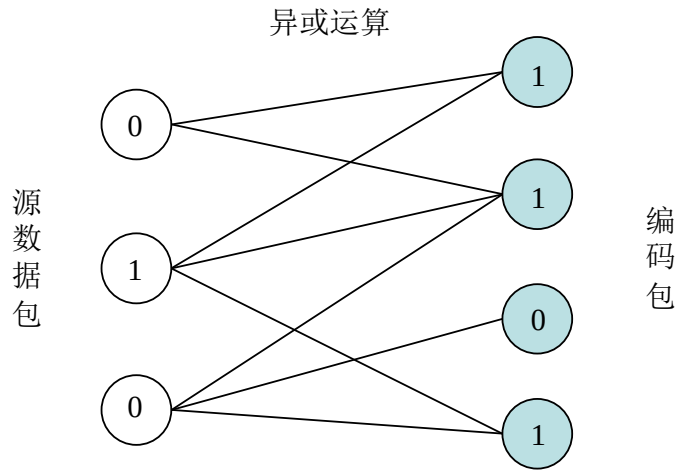


图 2-2　LT 编码的二分图

假设源视频数据按照 T 单元长度等分得到 k 个源数据包（长度不能整除时在数据末尾补零），使用度分布$\Omega = (\Omega_1, \Omega_2, \Omega_3, \cdots, \Omega_k)$表示随机选择度 d（d 个源数据包参与编码）的概率为Ω_d。LT 码的编码过程见算法 2-2。

算法 2-2

1. 根据度分布等概率地随机选取一个度值 d；
2. 等概率地随机选择 d 个互不相同的源数据包；
3. 将 2 中选中的源数据包进行异或运算得到一个编码包；
4. 跳转至步骤 1，直到产生 n 个编码包后退出。

与编码过程相对应地，算法 2-3 简要描述了 LT 码的解码过程。

算法 2-3

1. 查看是否存在度为 1 的校验节点；若有，转步骤 2；反之，转步骤 3；
2. 选取一个度为 1 的校验节点，根据其关联关系找到对应的变量节点，释放该校验节点、删除该关联关系，并标记该变量节点已译码，转步骤 4；
3. 查看是否仍存在未译码的变量节点；若存在，则译码失败；反之，译码成功；退出；
4. 依次将该变量节点与其剩余的各关联关系对应的校验节点做异或运算并删除关联关系，转步骤 1。

度分布直接决定了 LT 码的运算复杂度和译码成功率，因而对 LT 编码算法至关重要。Luby 于 2002 年提出了理想孤立子度分布（ISD）模型。根据 ISD 的度分布公式可以证明，对于 k 个源数据包的情况，每个校验节点的度的平均值只有 $O\ln(k)$。相应地，每次进行算法 2-3 中的步骤 4 时，每个校验节点被释放（异或运算后该校验节点的度为 1）的概率也只有 $1/k$。这说明使用理想孤立子分布

得到的校验节点的度平均值低，但是译码成功率也低。

为了解决这一问题，Luby 等人之后又提出了健壮孤立子度分布（RSD）模型。RSD 通过精心设计增大了算法 2-3 步骤 4 中剩余关联关系的数目，从而增大了各校验节点的译码成功率。

然而，即使使用健壮孤立子度分布，LT 码在实际操作过程中仍然存在译码失败的问题。在 LT 编码过程中，如果某个源数据包未参与任何一个编码包的编码过程，并且该数据包在传输过程中丢失，那么在这种情况下接收端必然译码失败。这就要求编码包的度维持在一个较大的值上，然而这将引起另外的问题——高译码复杂度和小的可译码结合。此外，译码时间非线性，度分布常数难以准确把握。

（3.2）Raptor 码

Raptor 码[29-30]的出现正是为了克服上述 LT 码的缺点，它在 LT 码的基础上改进得到，由预编码和 LT 编码双重编码组成。k 个源数据包首先经过预编码得到 L 个中间数据包，这些中间数据包再经过一次 LT 编码，得到最终的 Raptor 编码数据包。预编码一般使用 LDPC 码，它在具有纠删特性的同时降低了 LT 码对原数据包覆盖率的要求，在保证整体译码成功率的前提下降低了整体的译码复杂度。此外，Raptor 码中的 LT 编码无法单独完成译码过程，所以这种 LT 码又被称为弱 LT 码。图 2-3 简要展示了 Raptor 码的编码过程。

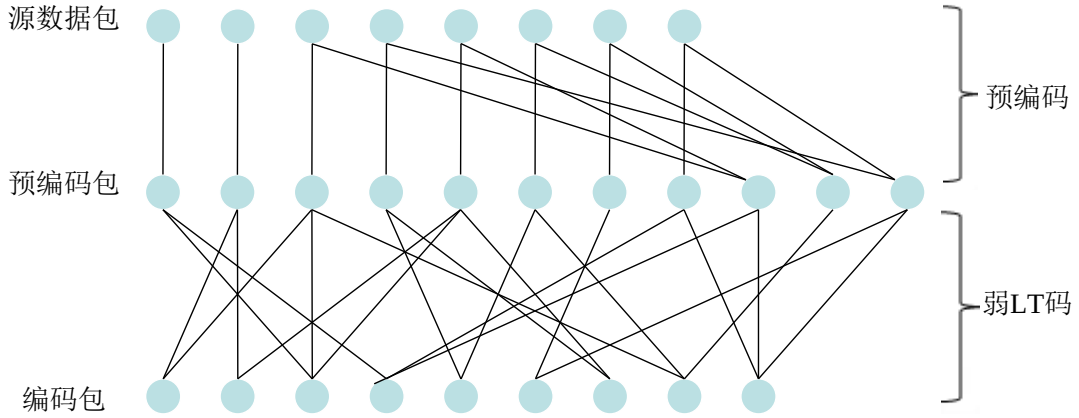


图 2-3 Raptor 的编码过程

当采用健壮的孤立子分布时，LT 码的译码时间复杂度为 $O(kln(k))$，可以看出是非线性的。相比之下，Raptor 的译码时间复杂度为 $O(k(ln(D)+3))$，其中 D 是一个正整数。显然，Raptor 码具有线性的译码时间复杂度。

不同于一般数据，压缩编码后的视频数据有 I 帧、P 帧和 B 帧三种类型，由它们的数据内容和彼此间的参照关系可知其重要性依次降低。考虑到视频数据量大且对传输实时性敏感等特性，在码率一定时，对视频数据按照重要性不同提供不相等的保护策略，有助于改善接收端呈现的画面质量，这就是不等差错保护

（UEP）思想的主要理论依据。

（3.3）UEP Raptor 码

自从 Raptor 码作为一种新的喷泉码被提出以来，人们在 Raptor 码的不等差错保护方面进行了很多研究。Ahmad 等人提出了一个 UEP 方法，通过简单的副本修改传统喷泉码的度分布算法，但是他们没有深入探讨该方法在视频传输领域的适用性[31]。Vukobratovic 等人通过增大喷泉码的窗口为可扩展视频多播提出了一个 UEP 方法，此方法在喷泉码中添加了一个窗口选择的过程[32]。Cataldi 等人通过具有滑动窗口机制的 Raptor 码将 UEP 用于可扩展视频广播，该方法在 Raptor 编码时将源数据块长度限制在一个滑动窗口中[33]。Hellge 等人通过在 Raptor 编码过程中整合一个层感知 FEC 方法实现了 UEP，他们的目标是设计出一个跨越所依赖视频层的 Raptor 算法[34]。

上述方案均通过修改 Raptor 码本身结构实现 UEP 机制，这不利于应用层的数据保护。同时与标准的 Raptor 码相比，这种方法存在着引入额外错误的可能性。Luo Zhengyi 等人提出了一种基于 GOP 层的 UEP Raptor 算法[35]，该方法将一个 GOP 组中所有分片按照失真影响（丢包失真影响和量化失真影响）从大到小的顺序进行排序，再将得到的视频分组序列划分为三个重要性等级，然后把所有处于最低等级的分组全部使用第一重要等级的修复分组进行替换，最后设计一个参数优化算法，求出每个重要等级的分组数目的一个最优的规划使该 GOP 组的整体失真影响均方差降到最低。使用这种方法进行失真分析时需要进行大量的组合运算，参数优化算法本身的时间复杂度接近$O(n^3)$，其中 n 为全部分组数目。在码长较长时，这种 UEP Raptor 算法的运算时延增长迅速，最终导致该算法在系统资源有限的嵌入式系统中无法发挥良好的编码性能。

2.2　基于重传的传输技术

数据重传是保证数据传输可靠性的另一种设计思路，也是本文研究的传输技术的主要理论依据。因此，在介绍基于重传的传输技术之前，首先简要说明自动重传请求 ARQ（Automatic Repeat-reQuest ）的相关知识。

ARQ 帮助传输层进行数据纠错，它通过错误检测、正面数据确认、负面数据确认和超时重传等机制力图在不可靠的网络上提供可靠的数据传输服务。

传统的 ARQ 分成为停等式 ARQ 和连续 ARQ 两种，连续 ARQ 又进一步细分为回退 N 式 ARQ 和选择重传 ARQ。其中，停等式 ARQ 协议[17]是 ARQ 最早的版本。停等式 ARQ 的实现机制非常简单，它规定接收方必须对发送方的每一个数据分组进行确认，因此可靠性强但传输效率也低。为了提高网络传输效率，发送方完全可以在等待确认分组的时间间隔内不间断地发送数据分组，即使用连续 ARQ 协议。可以使用公式证明，在无数据错误发生的情况下连续 ARQ 协议

的信道利用率接近 1，从而极大地提高了网络传输效率。

除了传统的 ARQ，还有混合 ARQ[22-24]。混合 ARQ 将前向纠错编码和 ARQ 相结合，在增大信道适应性的同时减小了传输时延，从而提高了数据传输的可靠性。现有的混合 ARQ 大多用在无线移动通信领域，而且 FEC 部分多采用 CRC（Cyclic Redundancy Check）、卷积码以及 LDPC 等比较传统的纠错算法[17-19]。尽管使用 Raptor 码的混合 ARQ 尚未出现，但这种将 FEC 和 ARQ 相结合的设计思想为本文提供了理论指导和启发。

2.2.1 TCP 协议

TCP（Transmission Control Protocol）协议[37]是一种面向连接的的协议，应用于 IOS 七层网络模型的传输层，它提供基于数据流的可靠数据服务。TCP 要求通信双方通过“三次握手”建立数据连接，通信结束后再经过“四次挥手”拆除连接。这种通信连接限制 TCP 只能用于端到端的数据通信。TCP 包含分组序列号和确认机制，保证了数据的按序提交。TCP 使用重传机制，保证了数据分组的传输可靠性。TCP 包头中含有数据分片和定界的标志位，用于数据块的分片和字节流的定界。TCP 使用滑动窗口和拥塞控制算法，允许发送方调整发送速率。所有上述特点保证了使用 TCP 协议进行数据传输的可靠性。

然而，在视频传输方面，TCP 协议多用在专网或者存在带宽预留的网络中。实际上，TCP 协议并不适用于无线实时视频传输。首先，TCP 使用稍待确认和超时重传机制以保证每个数据分组可达，容易造成视频数据的传输延时和画面抖动。其次，TCP 系统开销大导致传输效率低。使用 TCP 传输视频数据时，由于分组大致使 TCP 自动进行数据分片。分片数目越多，花费在确认包、数据分片、数据重组等方面的系统开销就越大，相应地，带宽利用率也就越低。最后，TCP 采用的拥塞控制机制容易导致视频数据分组在信道空闲时迅速占用信道带宽，而在发生网络拥塞时又会因急剧减小的发送窗口致使数据分组迅速填满发送缓冲队列。

2.2.2 可靠 UDP 协议

可靠 UDP 协议[16-20]使用 UDP 传输协议进行数据通信，并在应用程序层面添加了轻量级的面向连接、数据确认、数据重传以及拥塞控制和流量控制等机制，从而为数据在网络中的传输提供了一定的可靠性保障。

经 H.264 编码后的视频数据分为关键数据与非关键数据。其中，关键数据包括关键帧(I 帧)和 NALU 头信息，非关键数据包括 P 帧和 B 帧等。人为设置 H.264 关键数据与非关键数据的丢失，然后对相应的视频质量损失进行详细的量化对比。实验表明关键数据的丢失对视频回放质量的影响比非关键数据大得多。显然，可靠 UDP 协议对待传输的数据分组并不加以区分。因此，在丢包发生时，没有对

关键帧提供特别的保护，不能很好的保证视频传输质量。

2.3　本章小结

本章从编码和数据重传两个方向上对近年来比较常用的无线视频传输技术进行了总结。其中，编码又可划分为信源编码和信道编码。前者主要通过消除数据间的时间相关性、空间相关性和统计相关性来进行数据压缩，后者主要通过添加一定的冗余信息对传输过程中的数据差错起到一定的抵制作用。针对两种编码各自在视频传输领域的经典算法给出了简要介绍。第二小节首先简要介绍了AQR 重传机制的基本原理和主要分类，并对 TCP 协议和可靠 UDP 进行了单独的描述和比较。

第三章　UEP Raptor 和可靠 UDP 相结合的设计

任何算法的设计都必须基于一个具体的应用场景，故本章将首先给出文中使用的无线视频传输系统的整体设计架构，综合视频发送端和视频接收端两大部分进行主要功能模块的划分，并给出各功能模块的设计方法。其中，UEP Raptor 和可靠 UDP 相结合的无线视频传输方法贯穿发送端和接收端的整个通信过程，也是本章介绍的重点。这部分内容将从 UEP Raptor、可靠 UDP 以及二者具体的结合方法三个方面进行详细设计和说明。

3.1　无线视频传输系统的设计

本文设计的无线视频传输系统主要包含视频发送端和视频接收端两大部分，二者通过无线信道进行数据通信。其中，视频发送端运行在三星的 S3C6410 集成开发板上，该开发板通过 OV3640 数字摄像头进行视频采集[40]。发送端将采集到的数据进行压缩编码和可靠性处理后，通过无线网络发送到视频接收端。视频接收端运行在 PC 机上，它对收到的视频数据进行解码等操作后，将得到二进制格式的视频源数据显示在屏幕上。

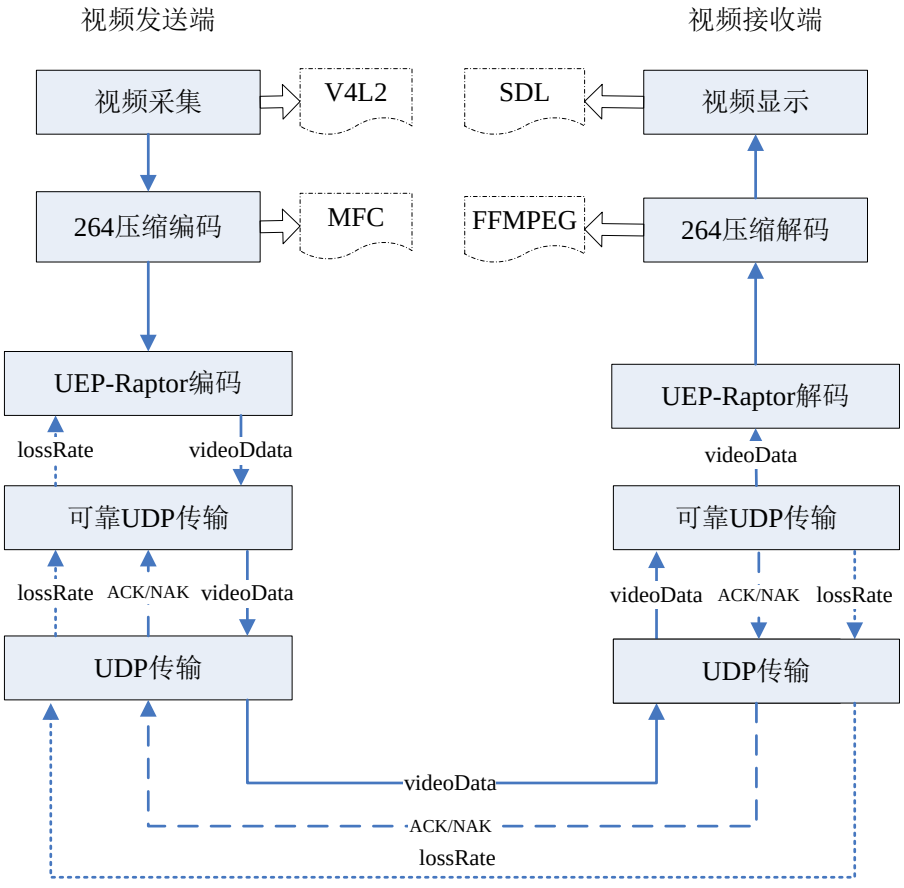


图 3-1　无线视频发送端和接收端数据流图

图3-1展示了无线视频传输系统中视频发送端和视频接收端主要的数据流图。从图中可以看出，视频发送端和视频接收端对视频数据的主要处理流程、各个阶段中使用的主要方法以及视频接收端对视频发送端主要的数据调控方法。以下综合考虑视频发送端和接收端，分别从视频采集、264编解码、视频显示以及UEP Raptor与可靠UDP相结合的传输方法三个部分进行介绍，其中传输方法又从UEP Raptor、可靠UDP和它们之间具体的结合方法三个方面进行详细描述，是本章的重点研究内容。

3.2　视频采集的设计

视频采集节点工作在S3C6410集成开发板上，即在嵌入式linux操作系统下通过摄像头完成视频数据的采集。应用程序使用V4L2（Video For Linux Two）实现原始视频数据的获取过程[40]。V4L2是一套功能强大的由内核提供的访问音、视频驱动的接口。具体工作流程详见图3-2。此外，由于视频采集过程直接调用V4L2的接口函数而且整个数据处理流程比较固定，在此不做细述。

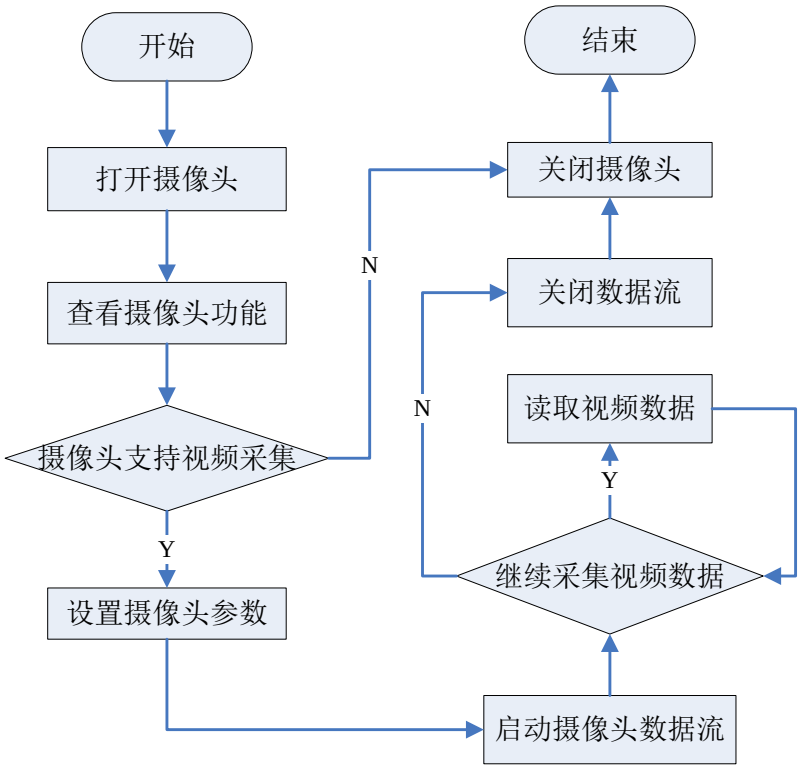


图3-2 视频采集流程图

3.3　264编解码的设计

（3.1）264编码

S3C6410集成开发板使用ARM11内核，ARM微处理器具有支持多种编码方式的硬件电路MFC（Multi Format Codec）[40]。由于MFC使用硬编码方式，因此编码过程并不占用CPU，且运算速度快，能够满足视频数据的实时性要求。

MFC 有一套成熟的接口可供应用程序调用，文中采用 H.264 压缩标准。基本的编码过程和使用的主要接口函数如图 3-3 所示。

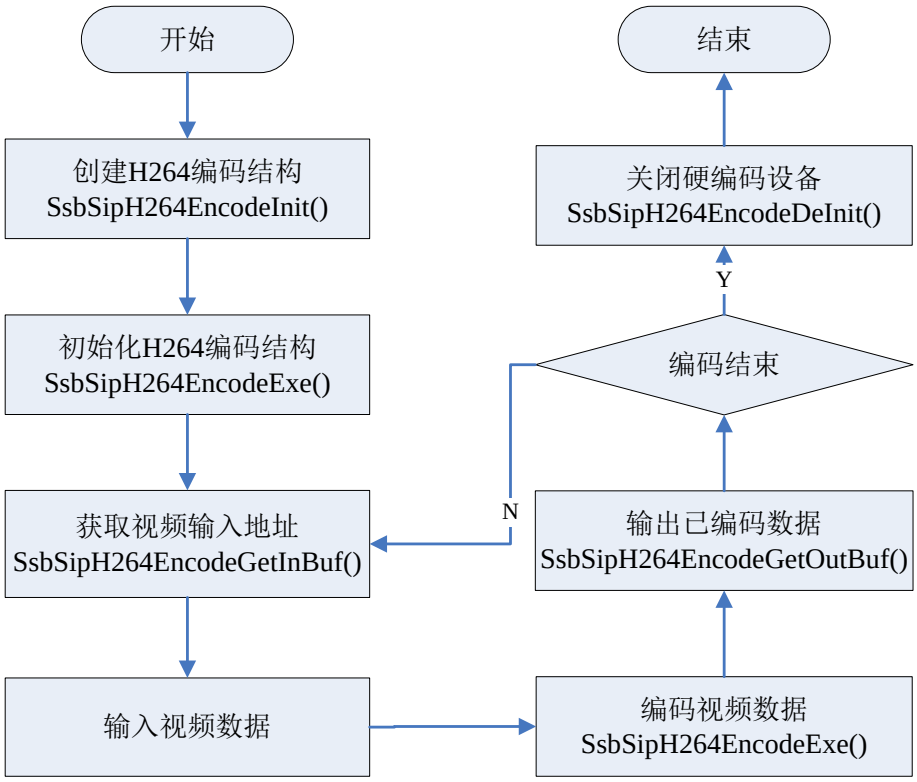


图 3-3　H.264 编码工作流程

最后，为了实现视频发送端数据采集和数据发送的并行性，将数据采集和数据发送分别抽象为视频采集线程和视频发送线程。采用“生产者——消费者”模型，一组对应的视频采集线程和视频发送线程相互之间通过数据缓冲区产生关联，并使用互斥信号量 mutex 进行同步。视频采集线程每次访问数据缓冲区之前，必须保证有空闲位置并获得互斥信号量 mutex。同样地，视频发送线程每次从数据缓冲区取数据之前，必须确保缓冲区非空且获得互斥信号量 mutex。如此，严格地实现采集和发送两个过程的同步。

（3.2）264 解码

264 解码用到 ffmpeg[38]框架下的 libavformat 和 libavcodec 两个库文件，前者用来解析视频格式，后者完成视频解码。

解码过程主要经过以下步骤：1.初始化 ffmpeg 库，注册所有音视频格式；2.根据已知的视频编码格式（图像宽、图像高、解码器类型、错误隐藏个数、调试类型等参数）找到对应的解码器；3.并打开解码器；4.设置一个的指针变量指向解码后的帧数据；5.读取一帧待解码数据；6.使用解码器进行解码；7.转换解码后数据为 RGB 格式；8.调用 SDL 库函数将 RGB 格式的视频数据在屏幕上进行画面展示。

图 3-4 简要展示了一帧视频数据的解码过程和使用到的主要库函数。

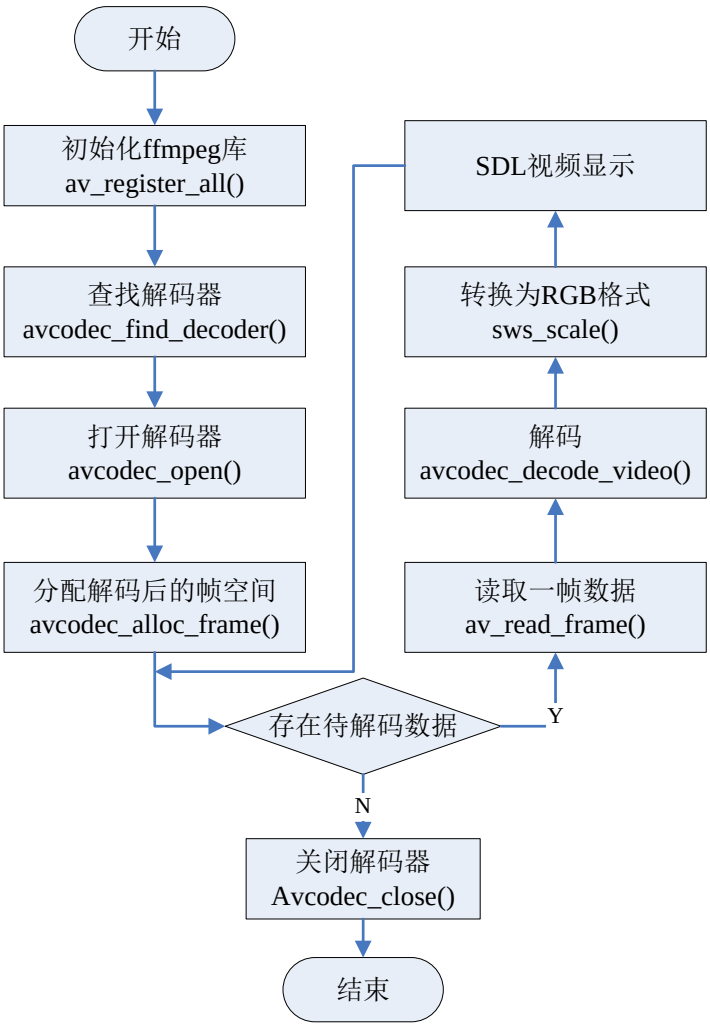


图 3-4 264 解码工作流程

3.4　视频显示的设计

使用 SDL(Simple DirecMedia Layer)[39] 库进行视频显示。具体操作流程如下：1.SDL_Init()完成视频库的初始化；2.SDL_SetVideoMode()创建一个播放窗口并设置其宽、高、单位像素位长和屏幕颜色；3.SDL_CreateYUVOverlay()创建一个 YUV 覆盖；4.SDL_LockYUVOverlay()给 YUV 覆盖加锁；5.img_convert()转变格式为 PIX_FMT_YUV420P；6.SDL_DisplayYUVOverlay()解锁 YUV 覆盖；7.SDL_PollEvent()捕捉并处理 UI 事件；8.SDL_DisplayYUVOverlay()像素投射，即画面显示。

3.5　传输方法的设计

通过比较常见的纠删码可知，相比于 RS 码、Tornado 码，Raptor 码具有无码率限制和编解码复杂度低的优势，相比于 LT 码，Raptor 码利用预编码技术实现了无损误码性能条件下的线性度分布复杂度，从而提高了译码效率。按照 3GPP（3rd Generation Partnership Project）标准[29]得到的 Raptor 码是系统码，即编码

后得到的前 k 个符号就是源符号本身。当接收端收到全部的源数据分片之后即可开始解码并通知发送端立即停止冗余分片的发送，以提高网络的带宽利用率和改善接收端视频显示画面的实时性。通过上述比较，本文选择纠删码中的 Raptor 码进行研究。此外，传统的 Raptor 码对所有数据进行相等差错保护。考虑到视频数据进行 264 压缩编码后得到重要性不等的 I 帧、P 帧和 B 帧这一客观事实，认为完全可以在 GOP 这一层上对重要性高的数据提供更多的保护，同时调整全部数据分片的长度以保证整体的冗余度控制在一个合理的范围内。

由于本文提出的 UEP Raptor 算法基于标准的系统 Raptor 码，旨在实现 GOP 层的不等差错保护特性，并不涉及 Raptor 码本身的度分布管理，因此使用的编/解码算法与系统 Raptor 完全相同。

3.5.1 UEP Raptor 的设计

（1）Raptor 编码的设计

本文按照 3GPP 制定的系统 Raptor 规范[29]进行具体的 Raptor 编、解码过程实现。Raptor 码的编码过程由预编码和 LT 编码两部分构成，具体即输入的源符号经过预编码生成包含了冗余信息的中间符号，这些中间符号再经过一次 LT 编码产生最终的编码符号。

预编码用到 LDPC 码、Half 码以及 LT 码三种编码方案，同时与一个中间辅助矩阵密切相关。假设每个输入的源符号长度为 T 字节，一共有$(C'[0],\cdots,C'[K-1])$等 K 个源符号，经过预编码生成的 L 个中间符号用$(C[0],\cdots,C[L-1])$表示，其中$L=K+S+H$，S 和 H 分别表示 LDPC 编码符号个数和 Half 编码符号个数，二者均可由 K 计算得到。经过精巧的设计，源符号$(C'[0],\cdots,C'[K-1])$和中间符号$(C[0],\cdots,C[L-1])$通过满足如下关系：

$$\begin{bmatrix} \begin{Bmatrix}0\\ \vdots \\ 0\end{Bmatrix} S \\ \begin{Bmatrix}0\\ \vdots \\ 0\end{Bmatrix} H \\ \begin{Bmatrix}C'[0]\\ \vdots \\ C'[K-1]\end{Bmatrix} K \end{bmatrix} = \begin{bmatrix} (\boldsymbol{G}_{LDPC})_{S\times K} & \boldsymbol{I}_{S\times S} & \boldsymbol{0}_{S\times H} \\ (\boldsymbol{H}_{Half})_{H\times(S+K)} & & \boldsymbol{I}_{H\times H} \\ & (\boldsymbol{G}_{LT})_{K\times L} & \end{bmatrix} \cdot \begin{bmatrix} C[0] \\ \vdots \\ C[L-1] \end{bmatrix} \tag{3-1}$$

其中，$I_{S\times S}$ 和 $I_{H\times H}$ 为单位矩阵，$0_{S\times H}$ 是全零矩阵，$(G_{LDPC})_{S\times K}$ 矩阵和矩阵 $(H_{Half})_{H\times(S+K)}$ 分别表示经过 LDPC 编码和 Half 编码后得到的相应矩阵，$(G_{LT})_{K\times L}$ 矩阵表示经过 LT 编码后得到的矩阵。G_{LDPC}、H_{Half} 和 G_{LT} 三个矩阵最初都被初始化为全 0 矩阵，因此由相应编码产生矩阵的过程实际上相当于给矩阵元素置 1 的过程，各个编码过程在文献[29]中都有详细的公式化定义，在此不作细述。

需要注意的是，LT 矩阵$\mathrm{G_{LT}}$的产生过程用到生成三元组的 Triple 函数，三元组决定了冗余符号由哪些源符号经过异或操作得到，该函数的实现方法见算法 3-1。

$$\mathrm{A} = (53591 + \mathrm{J[K]} * 997)\%\mathrm{Q};$$
$$\mathrm{B} = \big(10267 * (\mathrm{J[K]} + 1)\big)\%\mathrm{Q};$$
$$\mathrm{Y} = (\mathrm{B} + \mathrm{X} * \mathrm{A})\%\mathrm{Q};$$
$$\mathrm{v} = \mathrm{Rand}(\mathrm{Y}, 0, 2^{20});$$
$$\mathrm{d} = \mathrm{Deg}(\mathrm{v});$$
$$\mathrm{a} = 1 + \mathrm{Rand}(\mathrm{Y}, 1, \mathrm{LL} - 1);$$
$$\mathrm{b} = \mathrm{Rand}(\mathrm{Y}, 2, \mathrm{LL});$$

算法 3-1[29]

其中，Q =65521，表示不超过2^{16}的最小素数；X 为正在编码的源符号的序列号；J[K]数组各元素的值经过精心设计和严密计算，主要用来保证辅助编码矩阵$A_{L\times L}=\begin{bmatrix}(\boldsymbol{G}_{LDPC})_{S\times K} & \boldsymbol{I}_{S\times S} & \boldsymbol{0}_{S\times H}\\ (\boldsymbol{H}_{Half})_{H\times(S+K)} & & I_{H\times H}\\ & (\boldsymbol{G}_{LT})_{K\times L} & \end{bmatrix}$满秩并可逆；LL 是不大于 L 的素数；Rand 函数是一个伪随机数发生器，具体表示为

$$\mathrm{Rand[X}, i, \mathrm{m}] = (\mathrm{V0}[(X + i)\%256] \text{ xor } V_1)[(\mathrm{floor(X}/256) + i)\%256]\%\mathrm{m} \quad (3\text{-}2)$$

此外，*Deg*(*v*)是度生成函数，表 3-1 显示了其函数分布。

表 3-1　度生成函数分布[29]

Index　j	f[j]	d[j]
1	10 241	1
2	491 582	2
3	712 794	3
4	831 695	4
5	948 446	10
6	1 032 189	11
7	1 048 576	40

由式（3-1）可知，将含有输入源符号的矩阵和辅助编码矩阵$A_{L\times L}$的逆矩阵——对$A_{L\times L}$矩阵进行高斯消元法得到——相乘即可得到中间符号。然后，将中间符号再做一次如算法 3-3 所示的 LT 编码，就能得到最终的编码符号。

（2）Raptor 解码的设计

Raptor 的解码过程与编码过程相似，同样需要构建中间辅助矩阵并求其逆矩阵，然后计算出中间符号，最后对中间符号进行 LT 编码，从而恢复出全部的源符号。

假设$(E[x_0],\cdots,E[x_{N-1}])$表示接收端收到的$N(N>K)$个编码符号，仍然使用

$(C[0],\cdots,C[L-1])$表示 L 个中间符号，则二者之间满足如下关系：

$$\begin{bmatrix} \begin{Bmatrix}0\\ \vdots \\ 0\end{Bmatrix} S \\ \begin{Bmatrix}0\\ \vdots \\ 0\end{Bmatrix} H \\ \begin{Bmatrix}E[x_0]\\ \vdots \\ E[x_{N-1}]\end{Bmatrix} N \end{bmatrix} = \begin{bmatrix} (\boldsymbol{G}_{LDPC})_{S\times K} & \boldsymbol{I}_{S\times S} & \boldsymbol{0}_{S\times H} \\ (\boldsymbol{H}_{Half})_{H\times(S+K)} & & \boldsymbol{I}_{H\times H} \\ & (\boldsymbol{G}_{LT})_{N\times L} & \end{bmatrix} \cdot \begin{bmatrix} C[0] \\ \vdots \\ C[L-1] \end{bmatrix} \tag{3-3}$$

其中各个矩阵的计算方法以及相关参数的意义和计算方法与本小节（1）中所讲的完全相同，不再重复。

通过上述讨论可知，3GPP 中系统 Raptor 码的编、解码算法主要的运算量体现在矩阵变换和异或运算上，故整体的运算复杂度不高，即使在嵌入式环境下也同样适用。

（3）不等差错保护算法设计

受文献[11][35]的启发，本文提出一种基于 GOP 层的简单 UEP Raptor 算法。本算法无需进行复杂的算术运算，适用于嵌入式开发环境。同时，在不改变标准 Raptor 算法本身的基础上，为视频数据提供应用层级别的不等差错保护。

将一个 GOP 中的全部数据分组作为一个处理单元，同时将其中的每个数据分组视为一个 Raptor 码的元符号。从 2.1.1 可以看出，I 帧由于被 P 帧参考编码而比 P 帧更重要。采用相等差错保护（EEP）时，I 帧和 P 帧使用相同的冗余度（(*n*-*k*)/*k*）。现在设计一个不等差错保护方案：假设使用 H.264 编码产生格式为“IPPPP…”的视频帧序列（B 帧编码复杂，在资源有限的嵌入式环境下不适用）。首先，采集并缓存一个完整的 GOP，且帧序列总长度为 *M*，则该 GOP 包含一个 I 帧和 *M*-1 个 P 帧。然后，将每帧切分为长度为 *T* 的分片（长度不够时在末尾补 0），令k_i表示第 *i* 帧的分片数，*K* 表示分片总数，则有$\sum_{i=1}^{M} k_i = K$（$1 \le i \le M$）。假设R_i表示第 *i* 帧的编码比率，n_i为第 *i* 帧编码后的分片数，即有$R_i = k_i/n_i$。随着R_i的不同，编码之后重要性不同的帧被赋予不同比例的修复分组。对 I 帧采用较低编码比率，以增大其修复分组数，提高传输可靠性。对 P 帧采用较高的编码比率，以减小其修复分组数，提高传输效率。为了使总的编码比率R_{tot}保持不变，则必须满足以下条件：

$$\sum_i^M k_i / (\sum_i^M k_i/R_i) = R_{tot} \tag{3-4}$$

降低编码比率能够加大信息保护力度，但同时也降低了传输效率。由于 Raptor 码本身的特点，当总的编码比率R_{tot}降低到一定值时，总的传输可靠性就会趋于稳定，即不再随着冗余的增加而提高。相反地，由于信道码率的关系，系统的整体性能甚至可能随着冗余的增加而下降。因此，算法的最后一步就是设法

找到可靠性达到稳定时的总编码比率R_{tot}以及在该R_{tot}下平均画面质量达到最佳时关键帧和非关键帧各自的编码比率。在指定的传输环境下，这些参数可以通过经验值方法得到。

（4）帧划分和重组算法设计

Raptor 码属于块编码的范畴，3GPP 标准中明确要求输入的源符号数 K 需要满足 4≤ K ≤8192。理论上 K 值不小于 4 即可，然而考虑到 Raptor 的编码原理——根据输入的源符号进行有限随机编码，可知源符号数越大参与随机编码的输入数据越多，编码过程越充分，即冗余信息覆盖的源符号越容易均匀分布，从而整体的容错性能越好。由公式 3-5 可知，在视频数据总长度 F 一定时，符号长度 T 越小，划分得到的符号数目 K 越大。

$$F = K \cdot T \tag{3-5}$$

然而，T 的值也并非越小越好。T 值越小，相应地 K 值越大，与 K 成正相关的 S 和 H 的值也越大，进而辅助编码矩阵$A_{L\times L}$的行列数 L=S+H+K 越大，即参与矩阵求逆和异或运算的数据量随之增加。实验证明当 L>5000 时，Raptor 码的编码速度明显下降，这在一定程度上降低了 Raptor 码的编码性能。此外，T 值较小时，若将单个编码符号直接交给传输层送往接收端，考虑到数据链路层的最大传输单元（MTU）一般为 1500 字节，显然过小的数据不能充分利用网络带宽从而造成带宽浪费。实践证明数据部分的长度接近 1024 字节时，带宽利用率达到最佳。

综上所述，可以考虑将一帧的视频数据块切分为长度较小的源符号，进行 Raptor 编码，将得到的编码符号进行重组，使整体长度最接近 1kB。此外，由于产生的编码符号具有相同的帧编号、符号大小 T，且各编码符号的帧内序列号连续，以及一帧切分得到的源符号数目 K 固定，因此包含所有这些信息的帧头只需向接收端传送一个即可。故而，设计如表 3-2 所示的数据结构保存最终发往视频接收端的数据部分。

表 3-2 分组重组后的数据结构

帧头	符号数 N	符号 1	符号 2	…	符号 N

由于符号划分和重组的过程均不涉及任何复杂数学运算，只包含简单的字符串拆分和拼接工作，因此划分和重组过程基本不占用运算时延，但能够有效提高编码充分性和网络带宽利用率。

3.5.2　可靠 UDP 的设计

视频数据量大、对实时性要求高的特性，决定了无法在带宽有限的无线网络中使用具有高可靠性的 TCP 协议进行视频传输。而传统的 UDP 协议无需对传输数据进行确认，也因此不必对丢失数据进行重传。UDP 的这种特性一方面保证

了视频数据的连续发送解决了时延问题，另一方面无法避免随之而来的负效应——视频数据传输不可靠。

综合视频数据和无线信道两部分的特点，考虑参照 TCP 协议对 UDP 协议做一些轻量级的改进，以期在保持数据传输实时性的同时在一定程度上提高数据传输的可靠性。

（1）基本协议原理

可靠 UDP 协议[22]工作在传输层之上，是一个应用程序级协议，其具体协议层次如图 3-5 所示。可靠 UDP 是建立在 UDP 协议的基础上，并参照 TCP 协议添加了连接管理、数据确认和重传等轻量级保证数据传输可靠性机制的协议。此外，考虑到经过信源编码后得到的视频数据本身可分为关键数据（I 帧）和等关键数据（P 帧、B 帧），在无线网络因阻塞而出现带宽非常有限的情况时，仅仅对关键数据进行重传，从而在尽量减少网络传输压力的同时改善接收端的画面呈现效果。

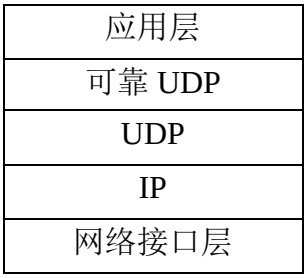


图 3-5　可靠 UDP 协议层次结构[22-23]

UDP 协议本身是无连接协议。用来传输视频数据尤其关键数据时，若出现物理连接失效等状况时，由于通信双方没有连接管理，发送发无法在第一时间获悉此消息，因而会继续以原速率发送数据。最终导致发送方缓冲队列大量占用，同时，接收端出现大量连续数据丢失，从而给通信双方带来额外传输负担。

因此，在参考 TCP 协议的基础上，可靠 UDP 提供面向连接的服务。在发送首个视频数据包之前，首先通过简单的“3 次握手”建立虚拟数据连接，此后通信双方可连续发送视频数据和连接保活确认等通信控制信息。当发送发数据发送完毕，或接收方没有数据要传送时，均可提出单方面的连接关闭请求，等到对方也发起连接关闭请求，本次连接彻底关闭，即可进行发送/接收线程推出、缓冲区等系统资源释放以及计数器清零等收尾工作。

图 3-6、图 3-7 分别展示了具体连接建立和释放过程。其中，SYN 表示连接建立请求标志、FIN 表示连接拆除请求标志、Seq 表示相应的包序列号、ACK 和 SYN 标志或 FIN 标志同时使用，分别用来回应对端的连接请求消息和回应对端的连接拆除消息。

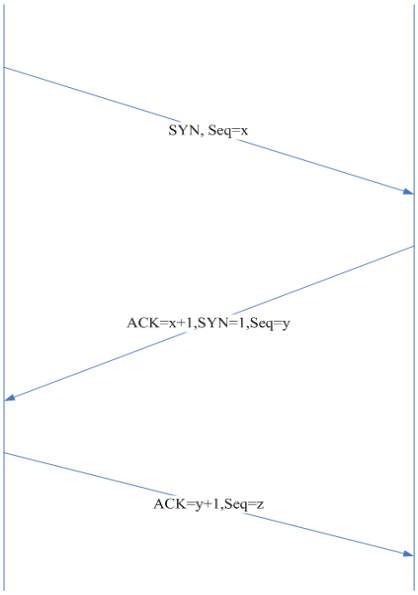


图 3-6　连接建立过程

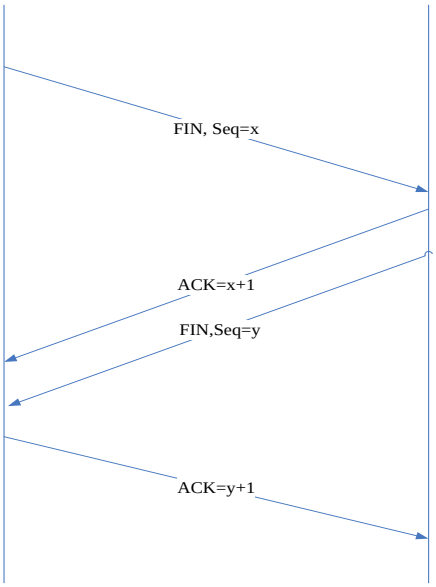


图 3-7　连接拆除过

（2）包结构设计

可靠 UDP 包含两种包结构，即数据包和控制包。顾名思义，数据包用来传输视频数据，控制包则用来传输通信控制信息。具体一个可靠 UDP 包是哪种类型，由包头的第一个数据位判定，0 代表数据包，1 代表控制包。

表 3-3　局部可靠 UDP 数据包结构

0　1　2　3　4　　31

0	序列号	
时间戳		
socket ID		
数据类型	数据长度	数据
数据		

表 3-3 详细描述了数据包结构。以下就各字段的含义和作用予以简要说明：

0：包类型标识位，0 代表数据包。

序列号：31 位长的无符号整数，唯一标识一个可靠 UDP 包，每个新数据包的序列号较前一个数据包加 1。不仅用来进行数据包确认和重传，而且帮助接收端进行一段时间间隔内的丢包数统计。

时间戳：32 位的无符号整数，表示包中数据的第一个字节的获得时间（系统时钟表示），用来帮助实现数据收发双方的同步。

socket ID：客户端的同一个通信端口允许建立多个 socket 链接，每个链接被分配一个唯一的 socket ID。各链接将自身的 socket ID 添加到数据包包头中，以便服务器区分当前数据包来自客户端的哪个 socket 链接。相应地，客户端也通过

收到的数据包中的socket ID判断当前数据包是服务器对哪个socket链接的响应。

数据类型：视频数据类型的标志位，0表示I帧，即关键数据，1表示P帧或B帧，即非关键数据。用来控制关键数据的重传，从而实现协议本身的局部可靠性。

数据长度：16位无符号整数，表示视频数据总长度，帮助进行数据定界。

数据：无符号字符型，表示经264压缩编码和UEP Raptor编码之后的视频数据。

表 3-4 控制包结构

0	1	4	16	31
1	类型	类型扩展	序列号	
时间戳				
socket ID				
控制信息				

表3-4详细描述了控制包结构，其中各字段的含义和作用如下：

1：包类型标识位，1代表控制包。

类型：3位二进制数，代表不同类型的控制包，与下述“控制信息”字段一起实现各种控制包的控制作用。

类型扩展：保留位，3位“类型”字段至多表示$2^3 = 8$种控制包，“类型扩展”字段可供用户自由地定义更多控制包类型。

序列号：16位无符号整数，唯一标识一个控制包。

时间戳：含义和作用同上。

socket ID：含义和作用同上。

控制信息：保存详细的控制信息，与“类型”字段相结合共同表示通信控制消息，详细说明见表3-5。

表 3-5 控制包类型

类型	控制信息	说明
000	1. 协议版本号 2. 初始数据包序列号 3. 包尺寸最大值	连接建立请求（SYN）
001	1. 收到的连续数据包的最大序列号 2. RTT（Round Trip Time）	接收确认（ACK）
010	1. 丢失包序列号的数组	丢包确认（NAK）
011	1. 协议版本号 2. 数据包序列号	连接拆除请求（FIN）
100		心跳保活
101	1. 平均丢包率	网络状况回馈
110		保留
111		保留

（3）数据队列设计

视频数据通过无线网络传输，不可避免地存在数据延迟和数据丢失等现象，继而造成包丢失、包失序等问题。如果接收端每次收到数据立即传送给解码层进行解码，解码层将通过逻辑判断得出帧数据不完整的结论因而丢弃当前保存的视频数据，滞后到达的视频数据也被判定为过时帧而丢弃。这种失序现象将持续进行，并最终造成接收端连续解码失败，从而严重影响画面呈现效果。因此，接收端需要设置一缓冲队列，用来暂存一帧的数据分组，即将各数据分组按照其序号存放在缓冲区的相应位置，从而保证了数据分组的有序性。最后，当接收到一帧的完整数据时，将整个缓冲区中数据交由解码器进行解码。

此外，为了实现关键视频数据传送的可靠性，数据确认和重传是必不可少的协议机制。如此，发送端发送出去的数据不能立即丢弃，而是需要保存在一发送队列中。同时，将其数据包序列号保存在一个待确认包序列号链表内，并通过某个简单的散列函数将数据包序列号和对应的视频数据在发送队列中的位置产生映射关系。只有当收到接收端对这批视频数据的接收确认时，才将对应的视频数据从发送队列中移除，并将待确认包序列号链表中的对应项清除。反之，若收到来自接收端的 NAK 消息，则发送端需要立即根据 NAK 控制包中的丢失包序列号数组通过散列函数找到对应数据在发送缓冲队列中的位置，并重新发送到接收端。此时，发送缓冲队列和待确认包序列号均不改变。直到重传次数达到设定的发送数上限，则认为该数据包已经失效，从而清理发送缓冲队列和待确认包序列号链表。

同理，对接收端也需要设置一接收缓冲队列，并维护一个丢失包序列号链表。接收缓冲队列暂存已接收的视频分组，即将视频分组按照其序号插入队列的相应位置，从而保证了保证了帧内数据分组的有序性。直到一帧数据接收完整，或接收的数据分组足以保证正确解码，即将接收缓冲队列内数据提交到上层进行解码显示，同时清空接收缓冲队列。反之，若丢失包序列号链表非空，且接收缓冲队列中数据分组数不足以解码，则当设定的 NAK 计时器超时时，接收端将丢失包序列号链表转换成整数数组作为 NAK 控制包的控制信息发送给数据发送端，请求重传。

（4）确认机制设计

在面向连接的传输协议中，数据确认是数据包是否成功到达对端进而是否需要重传的判断依据，因而是提高数据可靠传输可靠性的必要步骤。同时，确认机制决定了确认包的回送条件，具体涉及到确认包的发送频率以及数据发送端需要等待的时延等，所以又与整个网络的传输效率密切相关。

传统的面向连接传输协议大多采用基于时间的确认机制。基本实现原理是：接收端从接收到第一个数据包开始，启动一个定时器开始计时，之后持续接收发送端发送的数据包并放入接收缓存队列。直到该定时器超时，接收端发送一个

ACK（无数据包丢失的情况）或者 NAK（检测到数据包丢失）给发送端，通知其继续发送后续的数据包或者对丢失的数据包进行重传。接收端则再次设置该定时器，重新开始计时，开始下一个确认的循环过程。这种确认机制的设计和实现都相对简单，而且通常情况下运行良好。然而，在网络带宽充足的情况下，如果发送端以一个较高的发送速率向接收端传递数据，即发送端维护的待确认数据包队列以一个较快的速度增长。此时有可能存在这样的情形——接收端的定时器还没有超时，但发送端由于待确认数据报缓存队列已满而不得不停止数据发送。在接收端定时器超时之前的时间间隔内，发送端一直处于盲等状态，网络带宽得不到充分利用。

针对上述问题，很容易想到一种补充的确认机制，即基于数据报数目的确认机制。基本实现原理是：接收端从受到第一个数据包开始，启动一个定时器开始计时，同时使用一个计数器从零开始统计当前收到的数据包数目。当网络带宽充足且发送端发送速率较高时，即使接收端的计时器未超时，计时器已经达到了设定的阈值。此时，接收端立即向发送端发送一个确认包，对发送端的待确认数据包缓冲队列中数据进行确认。然后，将计数器清零并重置计时器，开启下一确认流程。这种情况下，发送方可持续发送数据而无需盲目等待，从而使信道带宽的利用率得到提高。反之，若当前网络带宽有限，计时器起主要确认作用，同样保证了发送端重传的正常进行。接收端同样需要在发送确认之后清零计数器并重置计时器。

（5）重传机制设计

数据包丢失的现象在无线网络通信中非常普遍，因而想要实现数据传输的可靠性，发送端必须重新发送未被收端成功接收的数据包，即进行数据重传操作。

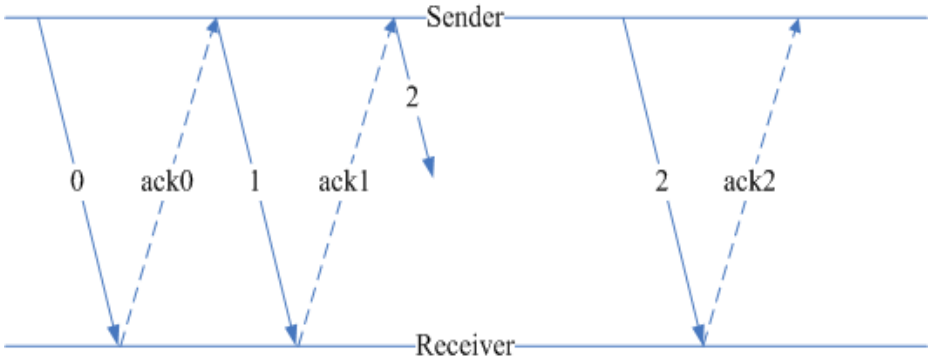


图 3-8 停等式数据重传机制

图 3-8 描述了采用停等式确认方式的数据重传机制。从中可以看出，发送端每向接收端发送一个数据包 *I*，就立即停止数据发送过程，直等到接收到来自对端的对数据包 *I* 的确认消息，才继续发送下一个数据包。其间一旦某个数据包（如图中数据包 2）等待确认的计时器超时，立即重新发送该数据包（数据包 2），然

后进入等待确认状态。不难看出，这种对数据包逐一确认的方式在提供高可靠性的同时严重增加了数据传输时延和确认包等系统开销，显然不适合视频数据的实时传输。

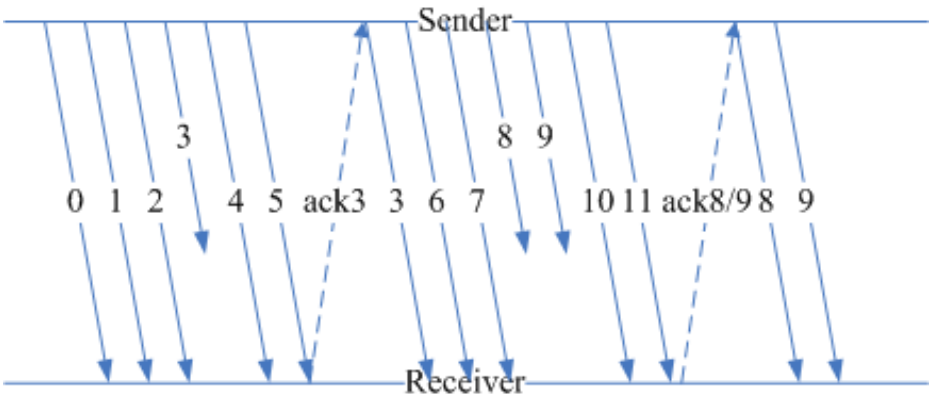


图 3-9 批量确认协议中的重传机制

图 3-9 描述了批量确认机制的基本原理。发送端连续发送一定数量（图中为 5 个）的数据包后停止，进入等待确认状态。同时，接收端收到一定数量（图中为 5 个）的数据包时，才向发送端发出一个确认消息，其中包含了最后一个接收到的数据包序列号的下一个序列号。发送端收到该确认消息，即判定上一批数据全部发送成功，并立即开始下一批数据的传输。反之，若存在数据包丢失，发送端将首先重传丢失数据，直等到本次重传成功或超出最大重传次数时，才继续发送后面的数据包。和逐一确认机制相比，这种确认机制显著减少了网络中确认包等控制信息的开销，节省了一定的网络带宽占用。同时，大大降低了发送端的传输时延，在一定程度上改善了接收端视频显示的实时效果。然而，发送端仍然存在等待确认包的时间间隔，并且出现数据丢失时，重传过程会导致之后数据的延迟传输。这在网络带宽充裕时，仍然在一定程度上限制了带宽利用率和传输效率，因而存在一定的改进空间。

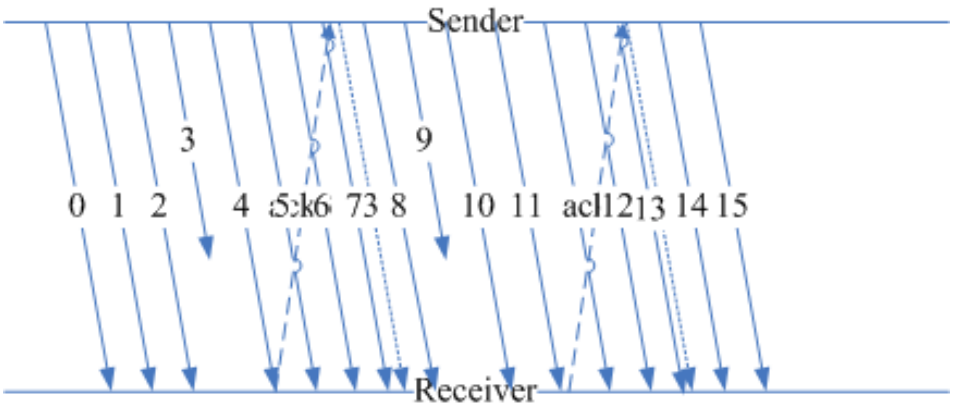


图 3-10　可靠 UDP 中数据重传机制

通过上述讨论，在可靠 UDP 中引入补发机制[20]，即发送端的数据发送线

程和数据重传线程并行执行，两个过程分别由发送指针和重传指针控制进行。这样，当检测到网络质量良好时（丢包率小，带宽充足），发送端的数据发送线程不停地向接收端传输数据包，即发送指针沿着发送缓冲队列不断前进。同时，当收到接收端的 NAK 消息时，重传线程根据 NAK 中包含的丢失数据包序列号使用重传指针进行数据重传。如此，两个指针同时对发送缓冲队列操作，极大地提高了数据传送效率。具体过程如图 3-10 所示。

（6）局部可靠性设计

无线网络实际可用带宽往往非常有限，由于信道阻塞引起的突发性连续丢包现象屡见不鲜。如果对所有丢失的数据包都重新传输，那么整个重传过程中大量的控制包无疑加重了网络带宽的负担，在极端情况下甚至可能引发连接崩溃。此外，与有线网络相比，无线网络容易受到频率相近的各种电磁信号干扰，导致传输数据出现差错和失真，这将进一步引起发送端数据重传。最后，视频数据本身对传输时延比较敏感，频繁的重传操作必然带来额外的传输延迟，最终影响到接收端的实时视频质量。因此，在无线网络中力图对所有丢失的视频数据都重新传输以求提高数据传送可靠性是不切实际的。

经过压缩编码后得到的视频数据有 I 帧、P 帧和 B 帧三种类型。其中，I 帧本身即是一副完整的图像，其解码过程无需参考任何其他图像数据。P 帧是差别帧，其内保存的数据是当前图像相对于前一完整图像的差别。因此，对 P 帧进行解码时必须和前一个 I 帧进行叠加才能得到最终画面。B 帧也是差别帧，与 P 帧不同的是，B 帧保存的是当前画面和前面一个或多个画面以及后面一个或多个画面的差别。所以，B 帧的解码过程离不开前后所有参考帧的参与，压缩率最高，解码过程也最为复杂。

通过上述对各种帧的数据内容和基本解码原理的讨论可以看出，与 P 帧和 B 帧相比，I 帧最为重要。在进行实时视频传输时，I 帧解析成功足以在接收端呈现出一副完整的视频画面。而 P 帧、B 帧的少量丢失会引起画面的轻微花屏现象，但并不影响实时视频的整体呈现效果。

因此，在无线网络的视频传输过程中，无需对所有丢失数据进行重传，只有当接收端检测到对视频呈现效果起主要作用的 I 帧发生数据丢失，并且无法正确解码时，才向发送端进行 NAK 反馈，要求其对关键帧的源符号进行重传。这意味着，发送端的待确认包序列号链表只需保存待确认的关键帧的源符号数据，同时接收端的丢失数据包序列号链表中也只保存丢失的关键帧的源符号数据。这样设计体现了可靠 UDP 协议对关键数据的传输可靠性，即局部可靠性。这种特性的好处在于，尽量减少网络带宽压力，保持 UDP 数据传输实时性优势，并适当改善接收端整体画面显示质量。

3.5.2　UEP Raptor 和可靠 UDP 结合方法的设计

UEP Raptor 编码和可靠 UDP 之间具体的结合方式可以从数据交互和参数调控两个层面进行总结。

（1）数据交互

数据交互层面是比较基础的结合方式，并且在视频发送端和视频接收端都是必不可少的协作过程。在视频发送端，UEP Raptor 编码和可靠 UDP 协议通过发送缓存队列相互作用：经过 UEP Raptor 编码后的数据符号首先被送入发送缓存队列，由可靠 UDP 协议对发送缓存队列直接操作，即何时发送数据符号、根据 ACK 包对发送缓存队列中的数据分组进行确认并移出发送缓存队列、根据 NAK 包中的数据分组序列号重新发送相应数据分组以及根据发送次数判断某个数据分组是否达到重传次数上限而可以视为过期并丢弃等等。同样地，在接收端，Raptor 解码和可靠 UDP 协议通过接收缓冲区进行交互：可靠 UDP 协议根据接收缓存队列中的数据包序列号判断是否有数据包丢失，并将丢失的数据包序列号存入一个链表，同时由 Raptor 解码程序判断当前数据包能否正确解码，当计时器超时或者收包计数器到达阈值时，综合链表和能否进行 Raptor 解码的判定结果，给发送端回送一个带有详细消息的确认包 ACK 或 NAK。

（2）参数调控

参数调控包含视频发送端根据视频接收端反馈对数据发送速率进行调整和视频发送端根据视频接收端反馈对 UEP Raptor 编码强度进行调整两个具体的过程。

视频发送端根据视频接收端反馈对数据发送速率进行调整，具体即视频接收端将统计得到的丢包率定期地发送给视频发送端，视频发送端跟据当前丢包率和近期丢包率变化情况判断是否有必要进行数据发送速率的调整，若需要则按照某一调整策略调节当前数据发送速率。具体调整策略参考 TCP 协议的拥塞管理办法。

视频发送端根据视频接收端反馈对 UEP Raptor 编码强度进行调整，具体指视频接收端按时统计丢包率并回馈给视频发送端，视频发送端根据当前丢包率和近期丢包率的变化情况判断是否需要进行 UEP Raptor 编码强度的调整，若需要则按照某一策略调节当前数据发送速率。具体调整思想即调整 GOP 总的冗余度以及该冗余度下 I 帧和 P 帧各自的冗余度，以期整体传输质量达到最佳。

最后，需要说明的是，理论上还可以重新设置视频采集速率（帧率）。然而，视频驱动会根据预先设定的摄像头分辨率自动调整帧率以达到最佳的视频采集性能。因此，这部分主要指丢包率对视频发送端发送速率和 FEC 编码强度的调控。

3.6 本章小结

本章重点介绍 UEP Raptor 和可靠 UDP 相结合的算法设计，考虑在陈述算法设计之前需要给出具体的应用场景，因此简要说明了文中使用的无线传输系统的整体设计架构。针对该架构综合视频发送端和视频接收端两部分进行主要功能模块的划分，并给出各个功能模块的设计方法。

UEP Raptor 和可靠 UDP 相结合的无线视频传输方法贯穿无线视频传输系统全部的通信过程，是本章的重点内容，3.5 节给出了详细的设计过程。

第四章　UEP Raptor 和可靠 UDP 相结合的实现

第三章从算法设计层面以本文使用的无线视频传输系统为环境背景给出了各个功能模块的设计思路，本章与第三章相呼应，将从数据结构、功能函数、处理流程等细节方面给出主要功能模块的详细实现。然而，考虑到视频采集、264 编解码以及视频显示等过程有一些既定的实现方法并且相对比较简单，本章集中讲述 UEP Raptor 和可靠 UDP 相结合的传输方法的实现。此外，UEP Raptor 和可靠 UDP 本身就是两个独立的过程，且各自内部又包含许多较为细节的功能划分。为了更详细地给出它们各自的实现过程并突出二者的具体结合方法，本章首先分别给出 UEP Raptor 和可靠 UDP 各自的实现方法，然后通过具体的结合方法将它们连接起来，使之成为一个系统的传输方法。

4.1　UEP Raptor 算法的实现

本文提出的 UEP Raptor 算法使用标准系统 Raptor 码[29]的度分布，因此编解码部分均按照 3GPP 标准制定的规范进行过程实现。

4.1.1 编码算法的实现

首先，介绍编码算法中用到的主要数据结构。根据第三章所讲的算法思想可知，中间辅助矩阵决定了源符号到中间符号的转换过程，即预编码。因此，需要有专门的数据结构保存中间辅助矩阵极其逆矩阵。使用如下数据结构表示一个矩阵：

```
sturct MyMatrix{
        unsigned int row;
        unsigned int colum;
        unsigned char ** rowpoint;
};
```

其中，row 和 colum 分别记录矩阵的行数和列数，由于不同矩阵的行列数不同，因此使用一指向无符号字符的二维指针 rowpoint 动态申请矩阵空间。

算法使用一系列的三元组表征修复符号和源符号的异或运算关系，变量 a 和 b 参与产生随机数，变量 d 则表示度分布，三者共同对 LT 编码起作用。设计如下数据结构表示一个三元组：

```
struct Triple{
        unsigned int a;
        unsigned int b;
        unsigned char d;
};
```

针对每个代表源符号数目的 *K* 值，都有与之对应的 LDPC 编码符号数 *S* 和 Half 编码符号数 *H*。鉴于 *S*、*H* 与 *K* 有固定的函数关系，可预先计算出 *K* 在[4，5000]范围内的一系列 *S* 和 *H*，并使用 *K* 值作为下标使之与 *K* 产生简单的映射关系。这在一定程度上节省了编码的运算量和运算时间。此外，由于编码过程用到中间辅助矩阵及其逆矩阵，因此将二者也一并存入表征 Raptor 编码的数据结构中：

```
struct RaptorPara{
        unsigned int K;
        unsigned int S;
        unsigned int H;
        Triple * trip;
        MyMatrix * A;
        MyMatrix * A_1;
};
```

至此主要数据结构已介绍完毕，以下简单说明编码使用到的主要功能函数。其中，矩阵函数专门用来进行矩阵空间申请、释放以及异或、求逆等基本的矩阵运算。编码函数部分则主要包括中间辅助矩阵的各部分元素置 1 算法和核心 Raptor 编码算法。具体如表 4-1 所示。

表 4-1 Raptor 编码主要函数列表

类别	函数名	函数功能
矩阵函数	matrixInit	矩阵初始化
	matrixInverse	矩阵求逆
	matrixXor	矩阵行间求异或
	matrixClear	矩阵空间释放
编码函数	raptorInit	Raptor 参数初始化
	LDPCCode	求辅助矩阵中的 LDPC 矩阵
	HalfCode	求辅助矩阵中的 Half 矩阵
	LTCode	LT 编码
	raptorInterCode	求中间符号
	raptorEncode	Raptor 编码
	raptorClear	Raptor 参数释放

图 4-1 给出了编码算法的基本执行流程。对于总长度为 F 的视频数据，按照某一固定长度 T 切分成 K 个源符号，作为参与 Raptor 编码的基本操作单位。首先需要进行一些初始化工作，如由 K 计算对应的 S、H 和 L，申请存放辅助矩阵和辅助矩阵逆矩阵的空间并初始化为 0 以及申请并初始化编码运算过程中的三元组。然后，进入编码过程。这一阶段又依次需要初始化保存中间符号的数组，初始化保存最终输出的编码符号的数组，通过生成 LDPC 矩阵、Half 矩阵、LT 矩阵得到完整的辅助矩阵，求解辅助矩阵的逆矩阵，根据辅助矩阵逆矩阵为 1

的元素位置将包含源符号的输入矩阵的相应行和列作异或运算得到中间符号，最后对中间符号进行 LT 编码得到包含源符号在内的全部编码符号。此时的编码符号就是包含了冗余信息的最终输出符号，直接将其插入发送缓存队列。在正式结束 Raptor 编码过程之前，还需进行一些释放之前所申请内存空间以及置空相应指针等清理操作。

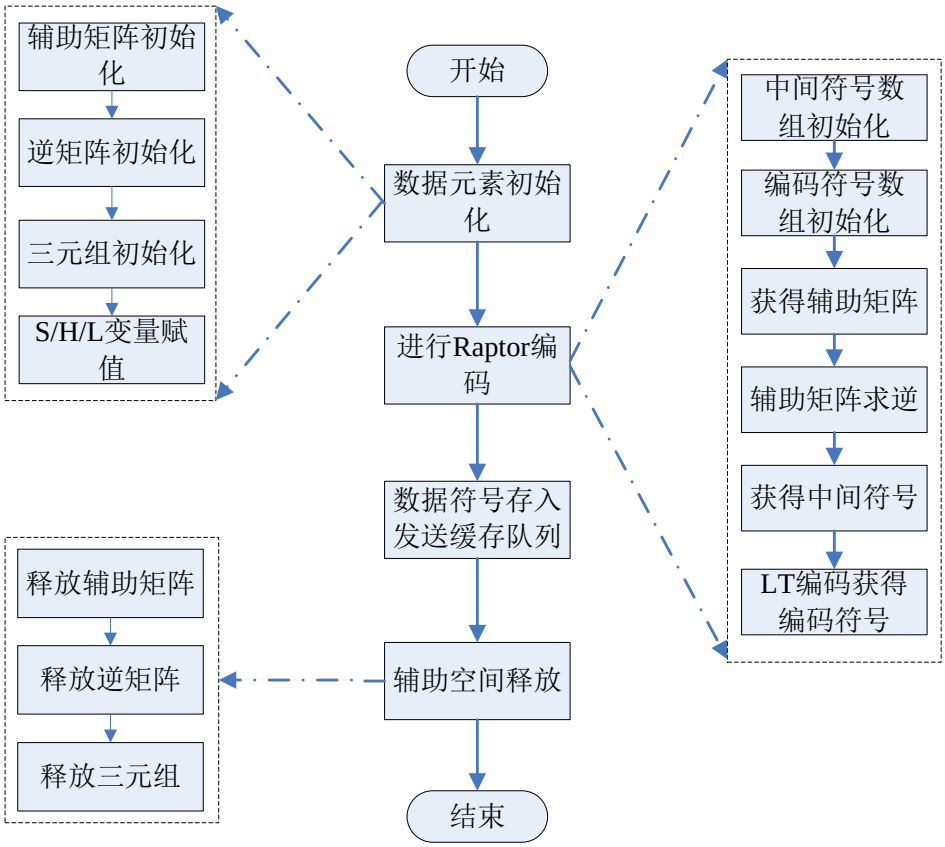


图 4-1　Raptor 编码流程

此外需要特别指出的是，实践证明，当发送端视频数据采集速率很高时，频繁的动态内存申请和释放操作会产生大量的内存碎片，系统对内存碎片的管理会消耗一定的时间。随着时间的累积，这种时延会越发明显，并对视频显示效果产生影响。因此，可以在 Raptor 编码之前一次性地申请足够空间保存中间辅助矩阵，然后根据实际的 K 值计算相应的 S 和 H，并对矩阵进行有限行列的异或等操作，不再每次都动态申请和释放中间辅助矩阵。最后，在视频采集线程结束之前释放中间辅助矩阵空间，防止内存泄漏。

4.1.2 解码算法的实现

Raptor 解码过程基本可以看作编码过程的一个逆顺执行，即首先根据 K 值计算出 S、H 和 L，一次生成 LDPC 矩阵、Half 矩阵和 LT 矩阵得到中间辅助矩阵，对中间辅助矩阵求逆，使用逆矩阵左乘含有 N 个编码符号（$N \geq K$）和 $S+H$

个空符号的输入矩阵，得到中间符号数组。对中间符号数组再进行一次 LT 编码，得到的输出符号即视频数据源符号。与解码过程不同的是，由于算法要求接收到的编码符号数 N 必须不小于源符号数 K，并且接收到的编码符号的帧内序列号不一定连续。因此，在构建中间辅助矩阵时 LT 编码的循环过程中，控制生成三元组的第二个参数不再是简单的循环计数变量，而是该编码符号的帧内序列号。同时，中间辅助矩阵的维度不再是$L \times L$，而变为$M \times L$（其中$L = K + S + H, M = N + S + H$）。

就算法使用的数据结构而言，保存矩阵的数据结构 MyMatrix 和保存三元组的数据结构 Triple 均与编码算法相同。不过，Raptor 参数 RaptorPara 多了一个已接收的编码符号数 N 和保存各个编码符号的帧内序列号数组：

```
struct RaptorPara{
        unsingned int K;
        unsigned int S;
        unsigned int H;
        unsigned int N;
        MyMatrix * A;
        MyMatrix * A_1;
        unsigned int * list;
};
```

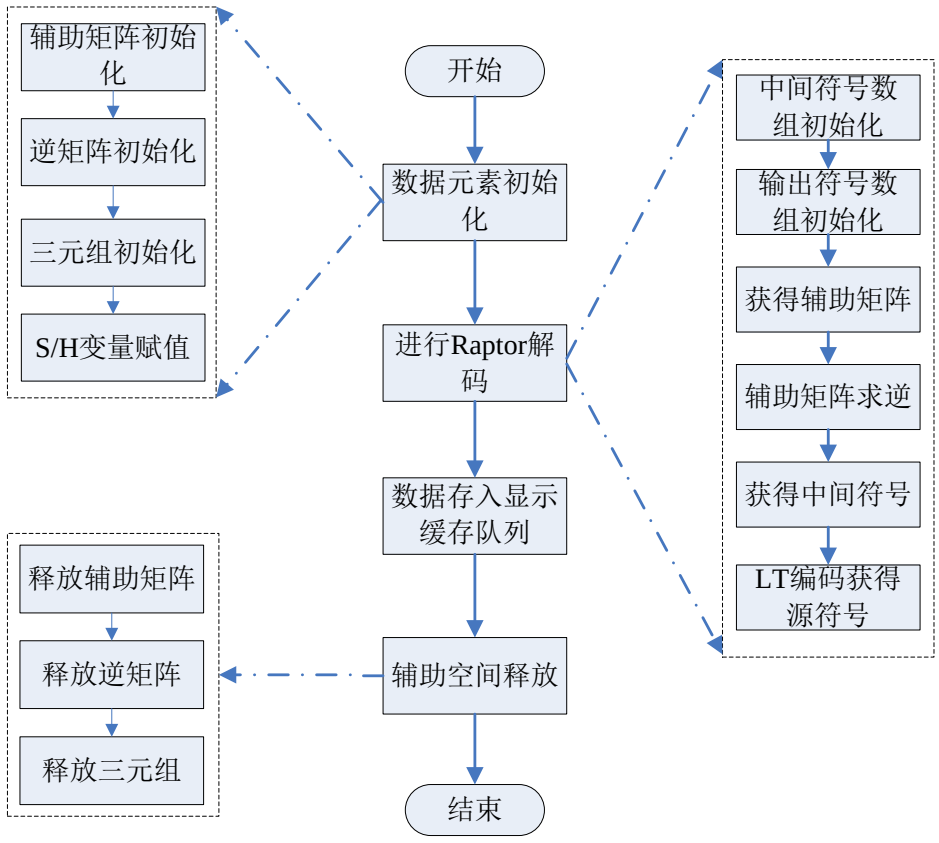


图 4-2　Raptor 解码流程

就算法包含的功能函数而言，除了使用解码函数 RaptorDecode 代替了原本的编码函数 RaptorEncode 之外，其他均与编码算法包含的功能函数完全相同，在此不再重复列出。图 4-2 给出了解码算法的主要工作流程。

4.1.3 不等差错保护算法的实现

通过修改 Raptor 码本身结构实现 UEP 机制的方案，一方面不利于应用层的数据保护，另一方面存在着引进额外错误的可能性。本文提出的基于 GOP 层的 UEP Raptor 算法，在不改变标准 Raptor 算法本身的基础上，为视频数据提供应用层级别的不等差错保护，力求将视频数据整体的失真影响均方差降到最低，最大限度地保证接收端的视频画面质量。

使用 H.264 编码产生格式为“IPPPP…”的视频帧序列，首先采用基于经验值的方法确定总的编码比率R_{tot}。参考文献[11]和[39]令在信道丢包率分别为 0.1、0.15、0.2、0.25、0.3 时，保持其它条件相同，在[0，0.5]的范围按照步长 0.1 内调整R_{tot}，使用长度为 500 帧、分辨率为 480×272 的视频各自测试 50 次。采用峰值信噪比（PSNR）作为效果评价指标，计算各种各种情况下视频每帧的平均 PSNR，并统计平均 PSNR 最高时的R_{tot}。

表 4-2 显示了不同丢包率下R_{tot}的实验结果，可以看出信道质量越好（丢包率越小）时，Raptor 编码达到稳定时所需的冗余度越小，传输效率也越高。

表 4-2　不同丢包率下的总编码比率

丢包率	0.1	0.15	0.2	0.25	0.3
R_{tot}	0.92	0.89	0.82	0.71	0.57

仍然采用基于经验值的方法确定各种情况下关键帧和非关键帧的编码比率比值。令在信道丢包率分别为 0.1、0.15、0.2、0.25、0.3 时，保持其它情况相同，在满足式 3-4 的条件下，使用长度为 500 帧、分辨率为 480×272 的视频各自测试 50 次。采用峰值信噪比（PSNR）作为效果评价指标，计算各种各种情况下视频每帧的平均 PSNR，并统计平均 PSNR 最高时的 I 帧和 P 帧各自的编码比率。表 4-3 给出了不同丢包率下的实验结果，其中R_I和R_P分别表示 I 帧和 P 帧的编码比率。可以看出，随着丢包率的增加，在满足总的编码比率条件时，关键帧的冗余度总是大于非关键帧，这也证明了对关键帧的提供更多的差错保护能够取得更好的视频画面质量。

表 4-3　不同丢包率下 I 帧和 P 帧的最佳编码比

丢包率	0.1	0.15	0.2	0.25	0.3
R_I	0.91	0.88	0.81	0.70	0.66
R_P	1.00	0.97	0.91	0.85	0.74

4.1.4 帧划分和重组算法的实现

所谓帧划分就是将长度 F 很长的视频数据进行细粒度的切分，得到长度为 T

的源符号的过程。划分算法非常简单不作细述。

经过 Raptor 编码后得到的长度为 *T* 的编码符号首先被放入发送缓存队列，发送线程从该队列中取出编码符号。保留第一个编码符号的首部，之后的编码符号都只提取出数据部分，并依次加入发送缓冲数组 outputBuf，同时修改该数组中包含的编码符号计数变量 sliceNum。直到遇到新一帧的编码符号，或者当前 outputBuf 的长度接近 1k，将当前输出缓存数组中的数据交由可靠 UDP 进行发送，然后开始新的 outputBuf 重组。图 4-3 详细展示了重组算法的工作流程。

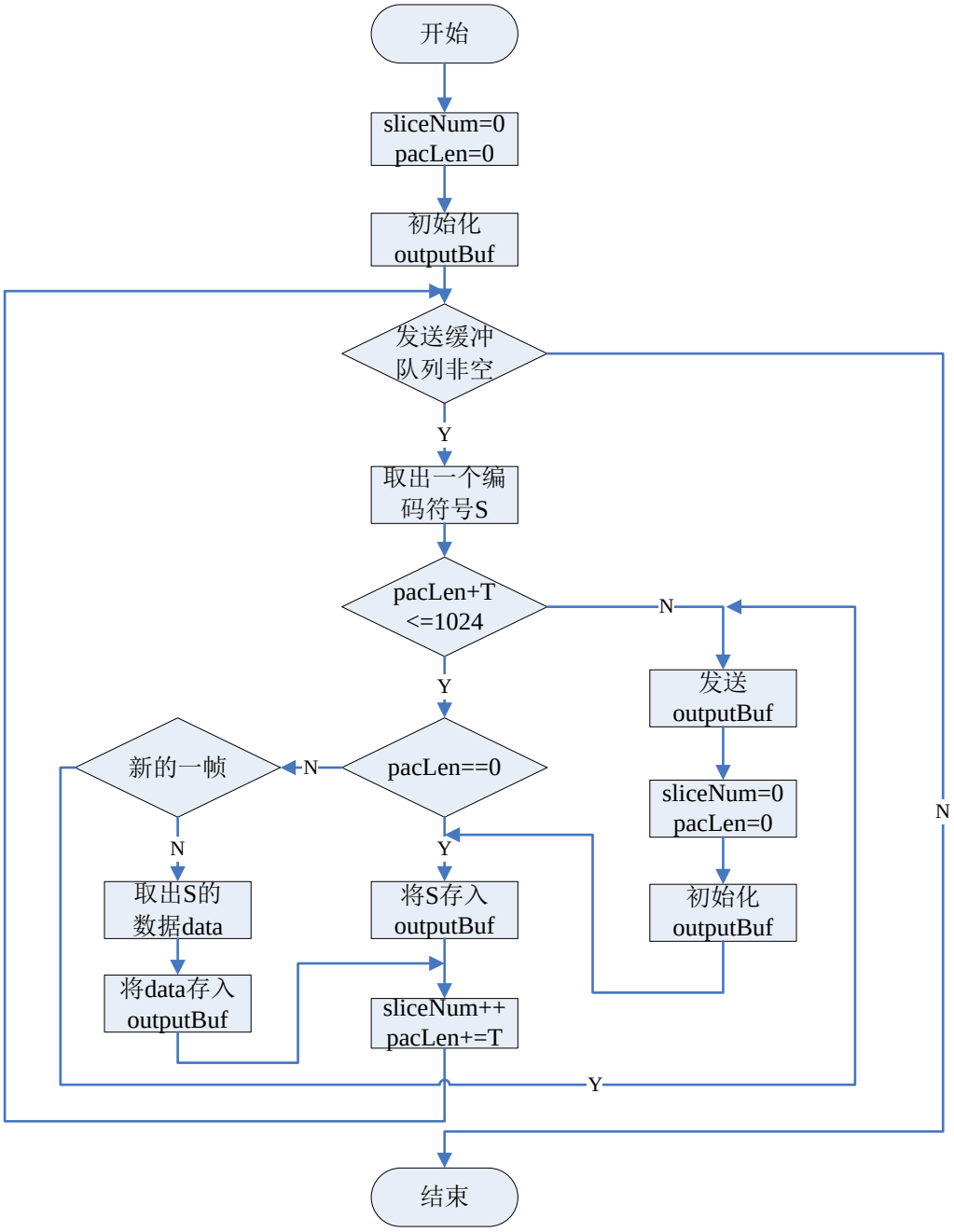


图 4-3　数据重组流程

4.2　可靠 UDP 算法的实现

4.2.1 包结构的实现

由 3.2.2 小节可以看出，控制包和数据包的包头均不超过 16 字节，即包头大小不大于 128 比特。此外，两种包的不同之处在于，数据包的包头之后连接的是实际的待发送数据，而控制包包头后紧跟着详细的控制信息。尽管待发送数据和控制信息的长度存在差异，但是均可以用一个指向字符数组的指针来表示。所以，可以将控制包和数据包的结构定义成相同的数据结构，至于两种包结构内部字段的具体填充和解析，二者则各自具有相应的功能函数。一个可靠 UDP 包的结构定义如下：

```
struct MyPacket{
        unsigned int mHead[4];
        int len;
        char * dataMsg;
};
```

其中，数据包和控制包各自的含有的字段变量定义如表 4-3 所示：

表 4-3　可靠 UDP 包内字段

包结构类型	字段变量名	字段变量说明
数据包	unsigned int mSeqNo	数据包序列号
	unsingned int mTimestamp	时间戳
	bool mIsIFrame	是否为关键帧数据标志
	int mDataLen	数据段长度
	char* mData	数据段
控制包	unsigned int mMsgNo	控制包序列号
	unsigned int mTimestamp	时间戳
	unsigned int mConType	控制包的类型
	char*mMsg	详细的控制信息
	unsigned int mVersion	可靠 UDP 协议版本
	unsigned int mRTT	RTT 值
	unsigned int mReqNo	ACK 确认的数据包序列号
	double mPktLossRate	接收端的统计丢包率

根据上述包结构的定义可知，包内含有的各字段需要解析得到。表 4-4 给出了数据包和控制包内各字段的解析函数。

需要特别说明的是，将保存包（数据包/控制包）的变量的作用域声明为全局，因此上述包内字段解析函数没有将其作为输入变量。此外，由于这部分的解析大多通过与一十六进制数进行按位与/或操作实现，相对比较简单，此处不再详细描述各个函数的具体实现。

4.2.2 数据队列的实现

由于同一个端口可以维持多个不同的 socket 链接，这些需要通过套接字描述符加以区别。各个 socket 链接独立进行一系列功能相同的函数，各自使用并维护

一组相同的计时器，并出发或处理相同的计数器超时事件。因此，针对每个 socket 链接创建一个单独的管理线程，专门处理该链接的各种通信过程。具体到队列管理方面即：在发送端，根据每一个来源 socket ID 创建一个处理线程，该线程具有唯一的 socket 套接字描述符 ID、暂存待确认数据包的发送缓存队列 sendQueue、与发送缓存队列相应的一个标记所有待确认数据包简要信息的链表 sendList 以及控制对接收端发送的控制包进行确认的定时器 ACKTimer、连接管理时间间隔的定时器 linkTimer。在接收端，根据每一个目的 socket ID 创建一个处理线程。与发送端的处理线程相似，接收端的任何一个处理线程也具有如下变量——目的 socket 套接字描述符 ID、接收缓存队列 recvQueue、标记未接收到数据包序列号的重传链表 reqList，以及控制确认时间间隔的定时器 ACKTimer、控制数据重传的定时器 NAKTimer、连接管理时间间隔的定时器 LinkTimer、控制向数据发送端传送丢包率的计时器 pacLosTimer。

表 4-4 包解析函数

包类型	函数原型	函数功能说明
数据包	unsigned int mGetSeqNo() const	获取数据包序列号
	unsingned int mGetTimestamp() const	获取时间戳
	unsigned int mGetSockID() const	获取目的套接字 ID
	bool mGetIsIFrame() const	判断是否为关键帧数据
	int mGetDataLen() const	获取数据段长度
	int mSetDataLen(cons int& len)	设置数据段长度
	char* mGetData() const	获取数据段
控制包	unsigned int mGetMsgNo() const	获取控制包序列号
	unsigned int mGetTimestamp() const	获取时间戳
	unsigned int mGetSockID() const	获取目的套接字 ID
	unsigned int mGetConType() const	获取控制包的类型
	char* mGetMsg() const	获取详细的控制信息
	unsigned int mGetVersion() const	获取版本
	unsigned int mGetRTT() const	获取 RTT 值
	unsigned int mGetReqNo() const	获取 ACK 数据包序列号
	double mGetPacLoss() const	获取丢包率

在数据发送端，视频采集线程需要将经过 264 压缩编码和 FEC 信道编码等处理的视频数据插入发送缓存队列，而视频发送线程则需要不断地从发送缓存队列中读取视频数据发送到接收端，并根据接收端回送的 ACK 控制包将经过确认的数据从发送缓存队列中删除。由此可见，视频采集线程和视频发送线程都需要对同一个发送缓存队列进行操作，因此数据插入和数据删除操作需要进行同步。同步过程通过信号量实现，使用操作系统中标准的“生产者——消费者”模型。

同样地，在数据接收端，接收线程将接收到的数据包插入接收缓存队列，

Raptor 解析线程读取接收缓存队列中的数据进行 Raptor 解码，并交由显示线程进行画面呈现。因此，对于接收缓存队列同样存在入队和出队的同步控制，并且使用和数据发送端相似的原理实现。

表 4-6 总结了队列管理方面主要的功能函数。由于数据发送端和数据接收端的队列管理函数除了函数内部的部分变量数据结构不同之外，基本功能相同，故在此不再进行详细划分。此外，缓存队列的管理与保存相应记录的链表密切相关（发送缓存队列与待确认链表、接收缓存队列和重传请求链表），所以将对链表操作的主要功能函数在此一并进行展示。

表 4-5　线程内主要变量定义

数据端	变量定义
发送端	unsigned int srcSockID
	unsigned char**sendQueue

	struct listNode{
	unsigned int seqNo;
	unsigned int timestamp;
	int sentTimes;
	int dataLen;
	};
	listNode* sendList;

	long long ACKTimer
	long long linkTimer
接收端	unsigned int destSockID
	unsigned char**recvQueue

	struct listNode{
	unsigned int seqNo;
	int reqTimes;
	int locaInQue;
	};
	listNode* reqList;

	long long ACKTimer
	long long NAKTimer
	long long linkTimer
	long long pacLosTimer

考虑到频繁的队列（链表）单元申请和释放不利于程序整体的运行性能，根据启发式实验验证为每个 socket 链接设置一个大小为 20 的数组模拟环形缓存队列。相应地，链表也使用尺寸为 20 的相应 listNode 型的数组表示。发送缓存队列的初始化主要将入队指针 nextPut、发送指针 nextSend 和补发指针 reSend 置 0，将全部队列元素清空。待确认链表的初始化主要将全部链表元素清空，并且把插入指针 nextPush 置 0。同样地，接收缓存队列的初始化主要将入队指针 nextPut、出队指针 nextPop 置 0，将全部队列元素清空。重传链表的初始化与待确认链表

完全相同。

表 4-6 队列管理相关函数列表

函数原型	输入	输出	功能
int InitQueue(void)	无	函数执行状态— 0：成功 -1：失败	进行队列初始化
int InsertQueue(unsigned char* data,int dataLen)	视频数据 数据长度	函数执行状态— 0：成功 -1：失败	视频数据（待发送/已接收）插入（发送/接收）缓存队列
int PopQueue(unsigned int seqNo)	数据包序列号	函数执行状态— 0：成功 -1：失败	将一个数据包从缓存队列中删除
int InsertList(struct listNode& node)	保存数据包包头信息的数据结构	函数执行状态— 0：成功 -1：失败	数据包包头信息插入（待确认/重传）链表
int PopList(int loca)	某数据包序列号在链表中的位置	函数执行状态— 0：成功 -1：失败	将序列号为 seqNo 的数据包记录从（待确认/重传）链表中删除
int InitList(void)	无	函数执行状态— 0：成功 -1：失败	（待确认/重传）链表初始化
int SearchList(unsigned int seqNo)	待查找数据报序列号	x（$0 \le x < listSize$）：数据包在缓存队列中的下标 -1：不存在	查找具有某一序列号的数据包在缓存队列中的位置

表 4-7 入队操作伪代码

```
int InsertQueue(unsigned char* data，int dataLen)
{
    if(队列未满)
    {
        if(P(queMutex))
        {
            将 data 插入 nextPut 指向的位置；
            nextPut = (nextPut + 1) % QueSize;
            V(queMutex);
            return 0;
        }
        else return -1;
    }
    else return -1;
}
```

表 4-8　出队操作伪代码

```
int PopQueue(unsigned int seqNo)
{
    int loca = SearchList(seqNo);
    if(loca >= 0 && loca < QueSize)
    {
        if(P(queMutex))
        {
            将下标为 loca 的队列单元清空；
            V(queMutex);
            重新调整队列，使数据连续存放；
            更新 nextPut 和 nextSend；
            调用 PopList(loca)将相应链表中信息删除；
            根据队列重新调整相应链表；
            return 0;
        }
        else return -1;
    }
    else return -1;
}
```

需要特别指出的，无论在发送端还是在接收端，要发送/接收的数据部分总是放在插入指针 nextPush 指向的位置，但是要求将对应的包头信息按照升序顺序插入待确认/重传链表中。因此，函数 SearchList 在队列中进行具有某一序列号的数据包在缓存队列中的位置时，可使用常见的二分查找法，在此不作详细交代。链表的插入/删除操作和队列的相应操作过程基本相似，故只是用表 4-7 和表 4-8 分别给了出缓存队列的插入和删除操作的伪代码。

4.2.3 确认机制的实现

接收端在收到来自发送端的数据包后，需要定期地给发送端返回一个包含了最大接收数据包序列号的确认包，以便发送端将经确认的数据从发送缓存队列中移除，为后续数据包的发送提供发送窗口。或者，接收端在 NAK 定时器超时时，发现用来保存未成功接收的数据包序列号的重传链表不为空，主动向发送端发送一个包含了重传链表的确认包，要求重新传送丢失数据。此外，通信的一方收到来自对端的链接建立/拆除的握手包，需要回一个包含了相同序列号的握手包，以便虚拟通信连接的建立/拆除。以上这些过程都是确认机制的使用场景。

从表 4-6 中可以看出，发送端用到的定时器有 ACKTimer 和 linkTimer，分别表示 ACK 定时器和连接维护定时器。接收端相比于发送端还有另外两个定时器 NAKTimer 和 pacLosTimer，分别表示 NAK 定时器和丢包率定时器。为提高定时器的精确度，使用 64 位的系统时钟表示时间间隔。任一定时器超时都会引发相

应超时事件，然后定时器复位。

接收端对发送端数据包的确认采用定时器和收包数目相结合的双重确认机制。具体即：接收端对每一个 socket 链接使用一个确认定时器 ACKTimer 和一个名为 packCount 的整数同时管理确认包 ACK 的发送。其中，ACKTimer 初始设置为根据 RTT 估算出的一个值 ACKInter（默认 0.01s），packCount 初始赋值为 0。程序循环地查询 ACKTimer 是否超时或者 packCount 是否超过设定的阈值 packMax（默认为接收队列长度的 2/3），立即创建一个 ACK 确认包并发送给发送端并重置 ACKTimer 和 packCount。这个过程使用表 4-9 所示的算法表示如下：

表 4-9 发送 ACK 确认

```
if(ACKTimer == 0 || packCount == packMax)
{
    创建一个 ACK 确认包；
    发送该 ACK 确认包到对端；
    ACKTimer = ACKInter； packCount = 0；
}
```

发送端在收到接收端发来的 ACK 确认包后，需要对发送缓存队列、待确认链表和定时器 ACKTimer、定时器 linkTimer 进行相应处理，详细过程见表 4-10。

表 4-10 处理 ACK 确认包

```
void onACK(MyPacket ackPkt)
{
    解析确认包 ackPkt；得到最大数据包序列号 ackSeqNo、控制包序列号 msgSeqNo 和 RTT 值 rtt；
    将发送缓存队列中序列号<=ackSeqNo 的数据全部移除；
    将待确认链表中序列号<=ackSeqNo 的记录全部移除；
    更新本地保存的最大确认序列号为 ackSeqNo；更新本地 RTT 为 rtt；
    复位 linkTimer；
}
```

此外，将接收端的 NAK 定时器 NAKTimer 超时、发送端接收到 NAK 包，以及收发两端处理连接定时器 linkTimer 超时等情形一并归为确认处理范畴。其中，连接定时器超时时发送一个保活的心跳包到对端并复位 linkTimer 即可。NAK 确认包的发送过程和处理过程分别见表 4-11 和表 4-12。

表 4-11 发送 NAK 确认

```
void NAKTimeout(const listNode& reqList)
{
    查找 reqList 中所有前一次反馈时间<=(2k*RTT)(1 ≤ k ≤ 3)的包序列号，并将其请求重传次数加 reqTimers 加 1；
    将上一步步中得到的包序列号组成数组，作为消息部分构建一个 NAK 包；
    发送 NAK 确认包到数据发送端；复位定时器 NAKTimer；
}
```

表 4-12　处理 NAK 确认包

```
void onNAK(MyPacket nakPkt)
{
    解析确认包 nakPkt，得到发送失败的数据包序列号数组 lossArry；
    for(int i = 0; i < lossArrySize; i++)
    {
        查找 sendList 中序列号为 lossArry[i]的包在发送缓存队列中的位置 loca；
        将发送缓存队列中 loca 位置的元素重新发送接收端，并将其链表记录
        中的发送次数 sendTimes 加 1；
        if(sendTimers >= 4)
        {
            该数据包的发送次数以达到上限，视为无效包；
            将该数据包从发送缓存队列和待确认链表中移除；
        }
    }
    复位定时器 linkTimer；
}
```

4.2.4 重传机制的实现

重传机制主要指发送端接收到 NAK 确认包后将指定的未被成功接收的数据包重新传送到接收端的过程。为了实现重传过程和原发送过程的并发进行，需要使用一个重传指针 reSend 指向发送缓冲队列中的丢失数据包，同时另起一个重传线程使用与原发送线程相同的 socket 描述符完成的重传过程。原本指向发送缓存队列中下一个待发送数据包的指针 nextSent 始终控制原数据包的发送过程，并不因丢失数据包的重发而停止。

需要注意的是，在发送端和接收端的缓存队列可动态调节，且前者大于后者的情况下，重传又可细分为补发和重发两种方式。补发指发送端重复发送序列号在一个范围内的连续数据包，这段数据之后到达接收端的数据被插入接收缓存队列。重发指发送端将发送缓存队列中序列号从某一数值开始的数据包全部重新发送，此时接收端将这段数据之后到达的数据包全部丢弃，并且要求发送端暂停原数据发送线程。原因在于，接收端的接收缓存队列长度有限。若丢失的数据包过多，那么即使将后续的数据包插入接收缓存队列，也只会导致缓存队列空间不足而影响重发过程，同时给发送端赞成额外的带宽浪费和队列维护开销。

4.2.5 局部可靠性的实现

所谓局部可靠性指仅对 I 帧的数据包提供可靠性保护。

在发送端，将视频数据打包成数据包之前，首先判断是否为 I 帧数据。如果是 I 帧数据，首先将数据部分插入发送缓存队列，然后构建包头、提取 listNode 结构的相关信息插入到待确认链表，最后组成完整的数据包并送往接收端。

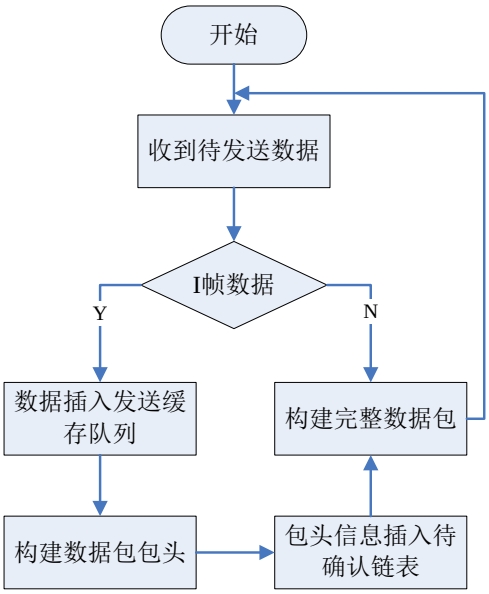


图 4-4　发送端局部可靠性

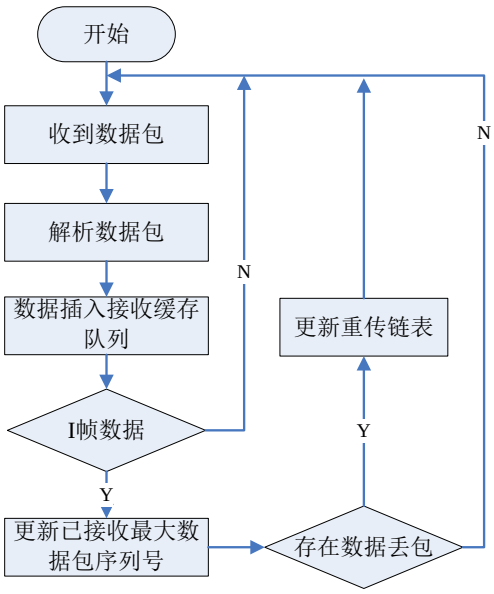


图 4-5　接收端局部可靠性

接收端在收到一个数据包后，首先进行包解析，获得其 mIsIFrame 字段。根据 mIsIFrame 字段当前数据判断是否为 I 帧数据，即关键数据。如果是 I 帧数据，则将数据部分加入接收缓冲队列，更新已接收的最大数据包序列号。与上一已接收最大数据包序列号比较，若存在丢包情况，则将中间丢失的数据包加入到 reqList 链表中。反之，当前数据非 I 帧数据，直接提取数据部分插入接收缓存队列。

4.3　UEP Raptor 和可靠 UDP 结合方法的实现

由本文的 3.5.2 小节可知，UEP Raptor 和局部可靠 UDP 的结合方式主要表现在数据交互和参数调控两个层面。其中，数据交互层面主要指发送端根据接收端反馈的 ACK 和 NAK 处理发送缓存队列和待确认链表等操作，这部分又和 4.2.3 的确认机制紧密相关，具体过程见表 4-10“处理 ACK 确认包”和表 4-12“处理 NAK 确认包”。以下重点介绍 UEP Raptor 和局部可靠 UDP 在参数调控层面上的结合方法。

UEP Raptor 和局部可靠 UDP 在参数调控层面的结合方式具体体现在两个方面：1）发送端根据接收端反馈的丢包率动态调整数据发送速率；2）发送端根据接收端反馈的丢包率动态调整 UEP Raptor 的编码强度，具体即通过调整 GOP 总的编码比R_{tot}以及 I 帧和 P 帧的编码比（参见 4.1.3）来调整 GOP 中 I 帧和 P 帧各自的冗余度。这两个方面都要求接收端周期性地向发送端反馈统计丢包率，因此接收端对丢包率的统计是 UEP Raptor 和局部可靠 UDP 相结合的技术基础。

4.3.1 接收端统计丢包率

接收端的统计丢包率 mPktLossRate 具体包含两种丢包率：从数据链接建立

至今接收端的全局丢包率 totalLossRate 和从上次统计时刻到当前时刻为止的一段时间内的阶段丢包率 interLossRate。式 4-1、4-2、4-3 分别给出了三种丢包率的计算方法：

$$totalLossRate = \frac{totalRecvPkts}{totalSendPkts} \times 100\% \quad (4\text{-}1)$$

$$interLossRate = \frac{periodRecvPkts}{maxRecvSeqNo - lastACKSeqNo} \quad (4\text{-}2)$$

$$mPktLoss = max(totalLossRate, interLossRate) \quad (4\text{-}3)$$

上述公式中使用到的变量表 4-13 所示。

表 4-13　丢包率相关变量列表

统计方法	变量定义	变量说明
全局丢包率	long long totalSendPkts	发送端发送的总包数
	long long totalRecvPkts	接收端接收的总包数
阶段丢包率	int lastACKSeqNo	上次统计确认的最大包
	int maxRecvSeqNo	至今接收的最大包
	int periodRecvPkts	至今接收的总包数

需要注意的是，接收端在统计阶段丢包率 interLossRate 时只考虑数据包的丢包情况，既没有将控制包数计算在内，也没有统计每个数据包的传送次数。这种统计结果尽管相对比较粗燥，但仍然在一定程度上反应了信道的丢包情况，对发送端而言同样具有评估价值。

4.3.2 发送端调整数据发送速率

发送端根据丢包率对数据发送速率的调整原则如下：若丢包率连续两次超过 20%，则认为网络出现拥塞，发送速率减半。之后每遇到丢包率超过 20%的情形，就将发送速率继续减半。直到丢包率连续两次小于 20%，就将发送速率增加 10%。如果后续的丢包率始终不超过 20%，则每次将发送速率增加 10%，直到达到上次发生网络拥塞时（数据发送速率减半时）的发送速率为止。

这部分主要用到两个功能函数——adjustOrNot 和 sendRateAdjust。其中，adjustOrNot 用来判断当前是否需要进行数据发送速率的调整，sendRateAdjust 则根据 adjustOrNot 的判定结果执行具体的发送速率调整过程。

表 4-14 和表 4-15 分别简要描述了调整判断函数 adjustOrNot 和发送速率调整函数 sendRateAdjust。在函数 adjustOrNot 中，参数 moreThan20 表示丢包率连续超过 20%的次数、lessThan20 表示丢包率连续低于 20%的次数、pktLossRate 和 PreSendRate 分别表示当前收到的丢包率和上次发生网络阻塞时的数据发送速率；在函数 sendRateAdjust 中，参数 judgement 表示调整判断函数 adjustOrNot 的判定结果、sendRate 表示当前数据发送速率、PreSendRate 表示上次发生网络阻塞时的数据发送速率。

本文设计的这种发送速率调整策略以 TCP 协议的网络拥塞算法作为参考，

主要目的是力图将丢包率维持在 20%以内，这一数值的设置与 FEC 信道编码的码率有关，且参考文献[11]和文献[39]。此外，通过实验证明该方法具有一定的可行性。

表 4-14 调整条件判断

```
int adjustOrNot( int *moreThan20,   int *lessThan20,
                 double pktLossRate,   double *PreSendRate)
{
    if(pktLossRate 大于 20%){
          *moreThan20 加 1；*lessThan20 清零；
        if((*moreThan20 >= 2){
            修改*PrePktLossRate 为当前 pktLossRate；
            return 1；//需要调整
        }
    }else if(pktLossRate 小于 20%){
        *lessThan20 加 1；*moreThan20 清零；
        if(*lessThan20 >= 2){
            if(*sendRate+10% > *PreSendRate)
                return 2；//需要调整
            else
                return 3；//需要调整
        }
    }
    return 0；//保持不变
}
```

表 4-15 发送速率调整

```
void sendRateAdjust (int judgement, double *sendRate, double *PreSendRate)
{
    if(judgement == 1)
        *sendRate 减半；
    else if(judgement == 2)
        修改*sendRate =为*PreSendRate；
    else if(judgement == 3)
        *sendRate 增加 10%；
}
```

4.3.3 发送端调整 UEP Raptor 编码强度

无线网络中的丢包率在很大程度上反映了网络的实际可用带宽以及数据传输能力。采用 UEP Raptor 编码对源视频数据进行容错保护的同时，冗余数据不可避免地增大了无线信道的带宽代价。因此，发送端根据接收端反馈的丢包率动态调整源数据的 FEC 编码强度，能够从源头上控制流入无线网络的数据数量。在发送端将对数据发送速率的调整和对源数据编码强度的调整相结合，有利于更加稳健地改善接收端视频质量。

发送端对 UEP Raptor 编码强度的调整主要包含调整条件判断和调整策略实施两个部分。前者主要指如何根据丢包率判断是否进行编码强度的调整，后者则指如何根据丢包率合理调整 FEC 编码强度，以下对这两个方面分别进行说明。

发送端对 UEP Raptor 编码强度的调整和对数据发送速率的调整同步进行。因此，调整条件的判断与 4.3.2 中发送速率的调整条件基本相同，此处不作赘述。

发送端调整 UEP Raptor 编码强度的主要策略是：根据文中 4.1.3 的表 4-2“不同丢包率下的总编码比率”和表 4-3“不同丢包率下 I 帧和 P 帧的最佳编码比”的统计结果，依据当前的丢包率合理调整 I 帧和 P 帧的编码比，以促使整体传输效果达到最佳。表 4-16 展示了发送端对编码强度进行调整的函数 redunAdjust。其中，judgement 表示编码强度调整与否的判定结果，*redunOfI 和*redunOfP 分别表示 I 帧和 P 帧的冗余度，pktLossRate 表示当前丢包率。

表 4-16　编码强度调整

```
void redunAdjust(int judgement, double *redunOfI,
                 double *redunOfP, double pktLossRate)
{
    if(!judgement){
        划分 pktLossRate 的值区间；
        根据表 4-3 中的丢包率区间计算 I 帧的冗余度*redunOfI；
        根据表 4-3 中的丢包率区间计算 P 帧的冗余度*redunOfP；
    }
}
```

4.4　本章小结

本章从数据结构、变量定义、数学公式、函数过程以及流程图等层面给出了 UEP Raptor 和可靠 UDP 相结合的具体实现，实际论证了算法的可行性。

第五章　测试与结果分析

本章将首先介绍测试环境和使用到的设备，然后指出具体的测试方法并总结测试过程中遇到的问题，最后给出视频发送端和视频接收端的运行结果。其中，视频发送端的运行结果必须能够简单地指出 UEP Raptor 的编码执行过程以及发生丢包时关键数据的重传过程。视频接收端的运行结果则必须能够直观地看出视频的画面质量。最后，将对同一段视频样本进行多次重复测试，以峰值信噪比（PSNR）作为评估标准，以得到较为精确的画面质量评测结果。

5.1　测试环境和设备

测试使用到的设备有 S3C6410 集成开发板、OV3640 数字摄像头、AWM2464 电脑线、E318233 串口线、PC 机、交换机和网线等。其中，视频发送端运行在 S3C6410 开发板上的嵌入式开发环境下，视频接收端运行在 PC 机上的 QT 集成开发环境下。AWM2464 电脑线两端分别连接 S3C6410 开发板和 E318233 串口线，而 E318233 串口线的另一端连接 PC 机，以便 PC 机使用 Putty 连接软件通过 USB 转换成的串口登陆 S3C6410 开发板并控制视频发送端程序的执行。

无线网络固有的缺陷最终体现在数据包丢失和数据包出错两个方面，而出错的数据包经过数据链路层的错误检测和网络层的数据校验两层过滤后基本无法到达应用层。因此，测试过程主要通过丢包率反应当前网络状况。考虑到测试要求网络的丢包率等条件完全相同，而实际的无线网络中这些参数的值并不恒定，容易导致测试结果缺乏可比性。所以测试过程的使用的网络模型是：S3C6410 开发板作为视频采集节点使用网线连接到交换机上，PC 机作为服务器（视频接收端）使用网线连接到同一个交换机上，配置二者 IP 地址为同一网段以便进行相互通信。最后，也是最为关键的部分，在服务器上运行的视频接收端程序中使用 Gilbert 模型[42]模拟网络随机丢包，通过相关参数的设置可以方便地调整网络的丢包率和数据连续丢失的长度。考虑到有线信道的稳定性和数据安全性，只要模型中设置的相关参数不变就能够保证两次测试的无线网络信道环境相同，即确保了测试结果的科学性。

5.2　测试方法和问题

采用重复和对比的测试方法进行测试。具体即：设置图像分辨率为 480×272、码率为 30kbps、帧率为 30、GOP 为 5、QP 为 28，其他变量采用默认值，保持上述所有变量的值不变，参考文献[11]和[35]依次设置接收端的丢包率为 0.1、0.15、0.2、0.25、0.3，使用保存在发送端的一段长为 500 帧的视频序列各测试 50 次，

每次都将接收到的视频（解码后得到的 YUV 格式的视频，并填充丢失数据保证帧对齐）保存 PC 机上。使用 VQMT（Video Quality Measurement Tool）工具计算保存在 PC 上的各视频帧序列的 PSNR，并统计每种情况下所有视频帧的平均值作为最终测试结果。

测试过程中发现了两个主要的问题，现在总结如下：

（1）数据分组失序导致接收端发送错误的重传请求。数据传输过程中有时会发生数据分组无序到达的情况，当传输关键帧的数据分组时，帧内序列号排在后面的源数据分组比排在前面的分组提前到达，此时接收端就会错误地判定前面的源数据分组已丢失并将其帧内序列号加入重传请求链表，当现有的分组无法解码成功且 NAK 定时器超时时，就会错误的请求后续收到的源数据分组。具体解决方法是：每接收到一个源数据分组，如果发现该源数据分组的帧内序列号与上一接收数据分组的序列号不连续，就查找重传请求链表，并从中删掉该数据分组的序列号，以防错误的重传请求。

（2）频繁的丢失分组重传请求。当设置的丢包率时，会发生连续的关键帧源数据分组请求的情况。假设帧 M 因存在源数据丢失而导致无法正确解码，因而其源数据分组序列号被加入重传请求链表，并被请求重传。帧 M+1 也出现这种情况，同样地引发遍历重传请求链表进行重传请求的操作。此时帧 M 的源数据可能已经在重传过程中，于是这种情况下对帧 M 的源数据分组的重传请求就是重复且无效的，甚至影响网络的传输效率。具体的解决方法是：为连续帧的丢失源数据分组设置一个独立的定时器，只有当定时器超时时，才引起对该帧的重传请求。

5.3　运行结果与分析

5.3.1 发送端运行结果

图 5-1、图 5-2 和图 5-3 均为运行在 S3C6410 集成开发板上的发送端窗口截图，综合反映了发送端的运行效果。

```
COP_1
YUV_Size = 195840
Frame_I: F = 15987 lostRate = 0.200000

YUV_Size = 195840
Frame_P: F = 9792 lostRate = 0.200000

YUV_Size = 195840
Frame_P: F = 9792 lostRate = 0.200000

YUV_Size = 195840
Frame_P: F = 9792 lostRate = 0.200000

YUV_Size = 195840
Frame_P: F = 9792 lostRate = 0.200000
```

图 5-1　发送端结果截图 1

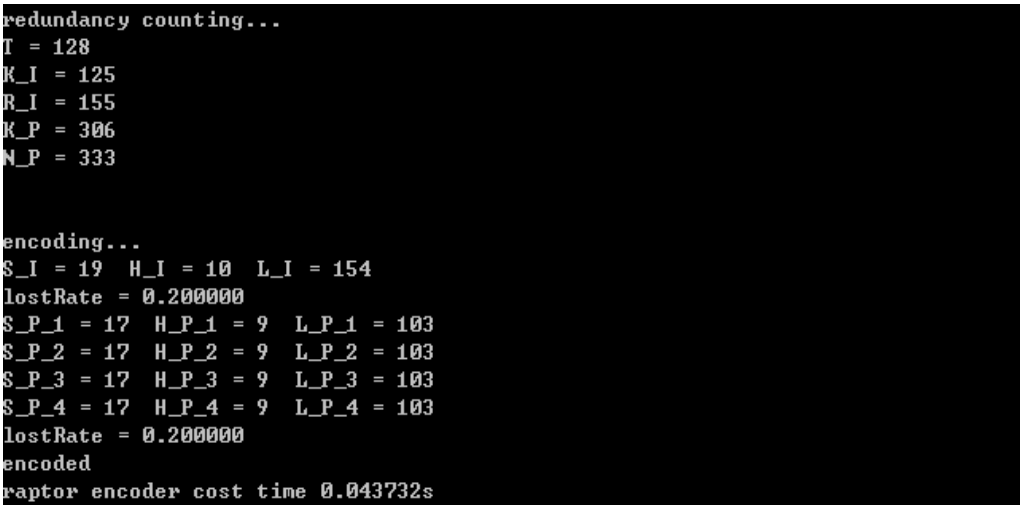


图 5-2 发送端结果截图 2

从图 5-1 中可以看到，每张 480*272 的 YUV420 格式的图片大小 195840 字节，这个数值可以通过公式 480×272×3/2 计算得到。设置参数 GOP=5、QP=28，在丢包率 lostRate 为 20%时，经过 H.264 压缩编码后得到序列 IPPPP，其中 I 帧大小为 15987 字节，P 帧大小为 9792 字节。按照 UEP Raptor 在 GOP 层上进行 I 帧和 P 帧 2 个重要等级的 FEC 编码。当分片大小 T 为 128 字节时，计算得到一个 GOP 的源数据分片数 15987÷128+9792÷128×4=431。为根据 4.1.3 中的统计结果，设置 I 帧和 P 帧的编码比率分别为 0.81 和 0.92，可知每个 I 帧一共 155 个分片，每个 P 帧一共 84 个分片，其中每帧图像独立编码和解码。最后给出了每个 GOP 编码使用的时间约为 44ms，在可容忍的延迟范围内。图 5-2 可以简单地跟踪查看 UEP 的参数优化过程以及标准 Raptor 的编码过程。

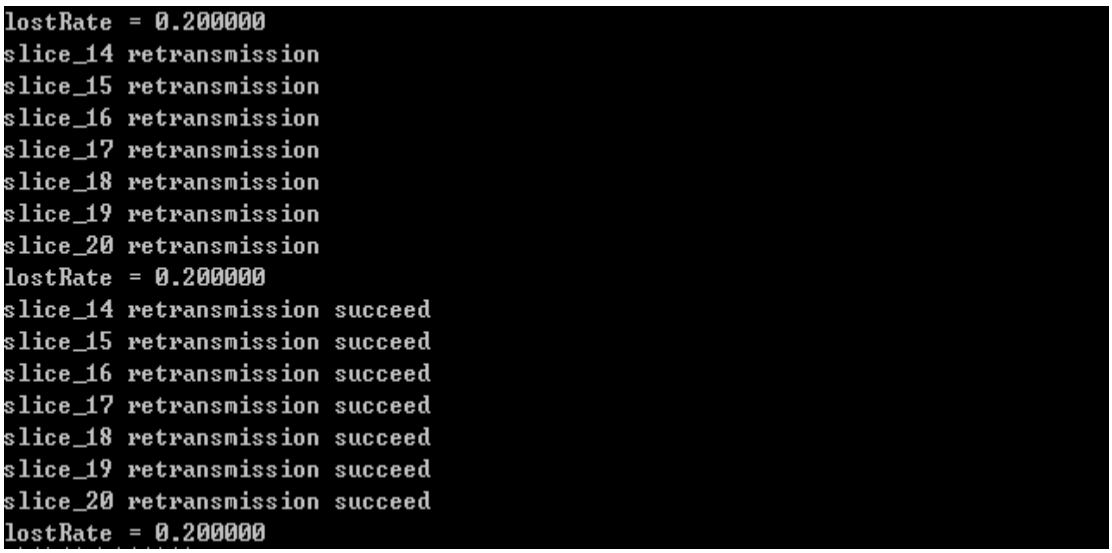


图 5-3 发送端结果截图 3

从图 5-3 中可以看出以下几点：

（1）在丢包率为 0.200000 时，帧 24 到帧 20 出现连续数据丢失，接收端接

收到 NAK 并开始数据重传。

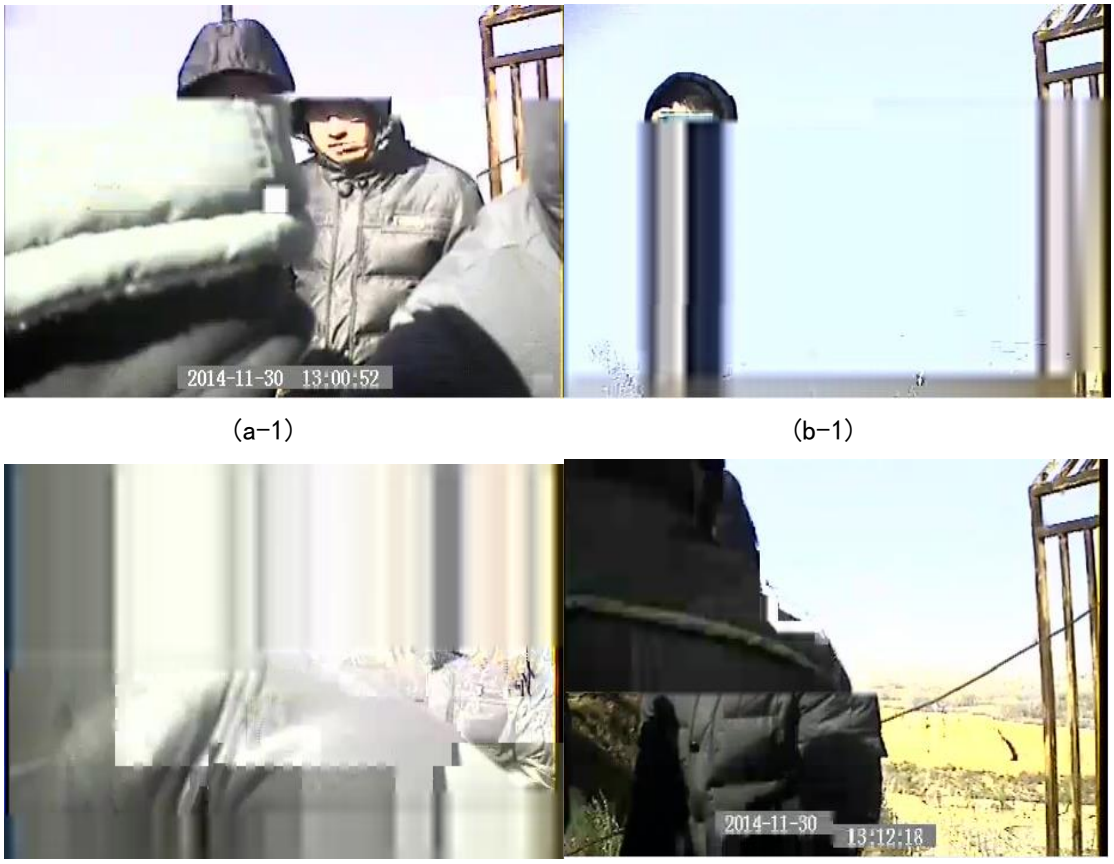
（2）一个 GOP 共有分组 155+84×4=491 个，其中 I 帧分组 155 个。帧划分和重组过程均在一个独立帧内进行，而编号在 14 到 20 之间的分组正好落[0,154]之间，说明接收端请求数据重传的是 I 帧（第一重要等级）的视频数据，体现了可靠 UDP 协议的局部特性。

（3）设置每个分组长度为 128 字节，加上视频分组首部 32 字节，以及局部可靠 UDP 数据包头部的额外长度，可知每个 UDP 数据包的中至多含有 7 个视频分组。因此，在一个 UDP 数据报中的 7 个分组到达接收端或同时丢失，这也可由图 5-3 中反映出来。

5.3.2 接收端运行结果

视频质量的评定可以使用主观质量评定和客观质量评定两种方法，前者从观察者的视觉感知出发，后者则从有效视频信号和噪声信号的比值方面给出客观评测标准。本小节主要针对发送端的视频质量给出主观质量评价。

使用 5.2 中给出的测试方法进行测试，发现在不同的网络丢包率下，在不使用任何可靠传输技术、使用标准 Raptor 进行 FEC 编码技术以及使用 UEP Raptor 和局部可靠 UDP 相结合的传输技术三种情况下的结果对比基本相同，以下对丢包率为 20%的测试结果进行讨论。



（a-1）　（b-1）

（c-1）　（d-1）

图 5-4　无保护结果

图5-4为丢包率为20%时室外实地测试所录视频中的任意四帧在不使用任何保护措施下，接收端的一组画面显示结果。其中，P帧保存的是当前画面和前一画面的差异部分，即连续画面的变化，因此P帧的丢失只引起画面的小部分马赛克现象。而I帧保存的是整幅图片的完整信息，其数据丢失将引起成片的画面模糊。

(a-2)　　(b-2)

(c-2)　　(d-2)

图 5-5　Raptor 编码结果

图5-5记录了上述四帧图片在使用标准Raptor编码时接收端的显示效果。由于标准 Raptor 编码对各个视频帧添加了一定的冗余分组，因而可以在一定程度上弥补由于数据丢失引起的删除错误。从图中我们也可以看出这种情况下，能够得到相比于完全无保护的情形更为清晰的画面。然而标准 Raptor 码本身的纠错能力有限，无法很好地弥补丢失的源数据分组，因此画面容易出现较明显的模糊。

图5-6为上述四帧在采用UEP Raptor和可靠UDP相结合的传输技术时接收端呈现出的画面效果。UEP Raptor采用与标准Raptor相同的GOP总的冗余率，但是增大了其中关键帧（I帧）的冗余率，即在整体GOP冗余率保持不变的情况下，对I帧提供更多保护。同时，接收端通过可靠UDP协议产生的确认消息促使发送端重新传输丢失的关键帧数据分组。二者相结合，能够在很大程度上提高接收端的画面显示效果。可以看出，图5-6相比于图5-5具有更好的清晰度。

(a-3)　　(b-3)

(c-3)　　(d-3)

图 5-6　UEP Raptor 和可靠 UDP 相结合结果

(a-4)　　(b-4)

(c-4)　　(d-4)

图 5-7　无失真结果

图 5-7 为上述四帧在没有数据丢失时，视频接收端显示出的无失真效果。

从上面四组对比图片中可以看出，在设定的丢包率下，相比于无保护和标准 Raptor 编码，UEP Raptor 和可靠 UDP 相结合的传输技术具有更好的整体画面显示效果。

此外，之所以在各种情况下上述 4 帧的图像质量改进效果相差较大，是因为测试过程通过 gilbert 仿真设置丢包率和连续丢包个数来尽可能真实地模拟网络丢包情况。这就导致视频各帧的丢包情况比较随机，可能某些帧丢失的多数是 I 帧的源数据包，某些帧则丢失多数的是 I 帧的冗余包，而另一些帧丢失的是 P 的源数据包但其 I 参考帧解析完全，最后一种可能是丢失的是 P 帧的源数据包同时其 I 参考帧解析不完全。第一种情况会导致数据重传，但是由于丢失的源数据包多，因而即使使用本文设计的传输技术，在接收端也无法解析得到清晰的图像，例如(c-1)、(c-2)和(c-3)。第二种情况下需要重传的源数据包较少，相对更容易重传成功，从而在接收端显示出清晰的画面，例如(a-1)、(a-2)和(a-3)以及(d-1)、(d-2)和(d-3)。第三种情况即使丢失的 P 帧不会被重传，但因为是在其前一 I 帧的基础上解析得到的，而该 I 帧的解析效果良好，从而促使当前 P 帧得以良好显示，例如(b-1)、(b-2)和(b-3)。最后一种情况文中没有给出对比系列图，但也易于理解，故不再赘述。

5.3.3 结果对比与分析

本小节将从数值结果统计角度给出客观质量评价。

根据 5.2 的对比测试方法，使用工具 MTU VQMT 计算每一种情况下的平均峰值信噪比（PSNR）。对结果进行统计，图 5-8 是对应的 PSNR 曲线对比结果。

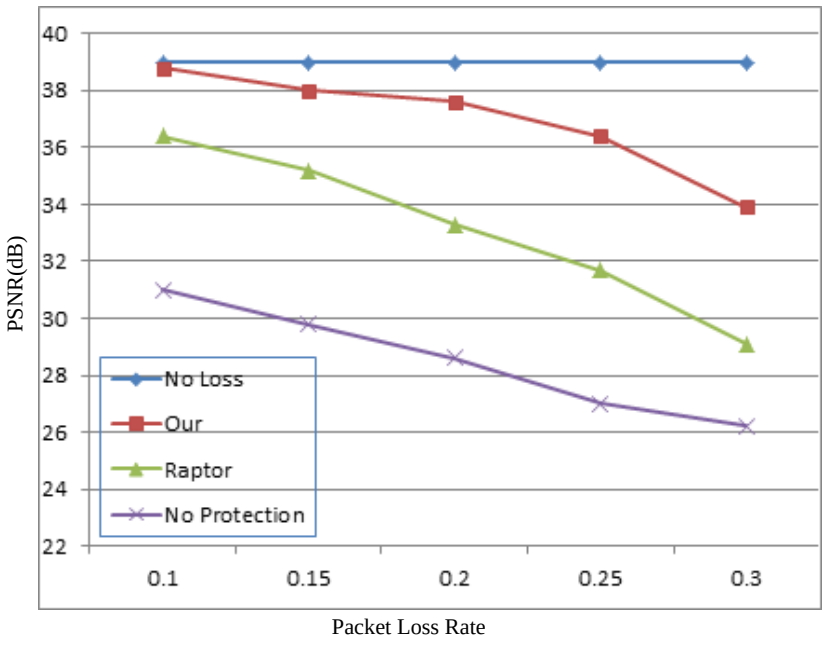


图 5-8 PSNR 结果对比图

从图 5-8 中明显可以看出，相比于无数据包丢失的情况，其它三种情况下，接收端的画面质量均随着丢包率的增大而降低。但是，在不采取任何保护措施时，接收端因为丢失的数据包而出现解码失败，因此呈现出的画面质量最差。使用标准 Raptor 码时，由于发送端对源视频数据添加的冗余信息对接收端丢失的数据包起到了一定的补偿作用，因此在相同的丢包率下相比无任何保护的情况具有较高的画面质量。然而，标准 Raptor 码对所有视频源数据进行相同力度的编码，即 I 帧和 P 帧的冗余度相同。考虑到所有数据包的丢失概率相同，但 I 帧由于丢包而产生的解码失败将导致之后所有与其有参考关系的 P 帧的解码失败，因此降低了整个 GOP 序列的画面质量，进而降低了整个测试视频序列的平均画面质量。相比之下，UEP-Rapor 在保证整个 GOP 的编码比不变的情况下（整个 GOP 的冗余分组数目保持不变），加大了对解码起关键作用的 I 帧的编码强度。在所有数据包的丢失概率相同的情况下，I 帧的冗余分组多因而容错性更强。此外，可靠 UDP 协议在接收端 I 帧解码失败时引发数据重传，这也在一定程度上提高了 I 帧的译码成功率。再者，P 帧本身保存的是当前画面相对上一画面的变化，因而即使译码失败也只是引起画面得局部模糊，不会导致整个画面质量的明显下降，进而 GOP 序列的整体画面质量不会受到严重影响。因此，使用 UEP Raptor 和可靠 UDP 相结合的传输方法相比于使用标准的 Raptor 码具有更好的平均画面质量。

从图 5-8 中同样可以看出，在丢包率不超过 30%时，相对于无保护状态和标准 Raptor 码，UEP Raptor 和可靠 UDP 相结合的传输技术具有更好的图像效果。

5.4 本章小结

本章第一小节主要介绍了测试环境和使用的设备，交代了具体的实验场景。第二小节给出了详细的测试方法以及测试中遇到的两个主要问题和相应的解决方法。最后，从单一视频帧的画面显示效果和一段视频的统计 PSNR 值两个方面给出了测试结果，并进行了对比和分析。测试结果表明，在无线网络中进行视频的实时传输，同比条件下使用纠删码和可靠 UDP 相结合的传输技术相比于不加任何可靠传输技术和仅使用标准系统 Rapor 码具有更好的画面显示效果。

第六章　总结与展望

6.1　工作总结

本课题从需求分析出发，明确了研究目的，通过查找相关资料对无线视频传输关键技术进行了学习和整理，并进一步明确了研究方向。设计课题的算法，并且实现了各个功能模块。搭建无线视频传输系统，设计测试用例并反复进行测试，最终对测试结果整理和分析，得出结论。

本文首先简要介绍了课题的研究背景、国内外的研究现状、研究意义和目标，其次从编码和传输协议两个方面介绍了无线视频传输技术，并据此提出本文的研究内容，即纠删码和可靠 UDP 协议相结合的无线视频传输技术。纠删码能够在一定程度上抵制数据丢失，但也存在着冗余信息消耗带宽的缺点，因此如何在控制 FEC 码率的情况下，改善接收端画面质量是下一步的研究方向。有针对的数据重传能够通过有限的带宽消耗显著地改善丢包引起的视频质量下降。然而，面向连接的 TCP 协议并不适合在无线网络中进行实时视频传输。因此，UDP 协议就成了视频传输的必然选择。但是 UDP 无法保证数据传输的可靠性，于是引入了可靠 UDP 协议。同样地，考虑到无线网络可用带宽有限，以及视频数据对传输时延的敏感性，在控制发送端 FEC 数据冗余的同时，需要合理控制接收端的重传请求，并最大限度地保证实时视频的整体画面效果。因此在可靠 UDP 的基础上添加了局部可靠特性，即只对丢失的关键数据进行重传。最后，将纠删码和可靠 UDP 协议相结合，从信道编码和传输协议两个方面为无线视频的传输可靠性提供保障。相应地，本文分别从算法设计和实现两方面对纠删码和可靠 UDP 协议相结合的传输方法给出了详尽的描述。最后，在不同的网络丢包率下，对纠删码和可靠 UDP 协议相结合的传输方法进行了反复测试，并与无保护状态和使用标准 Raptor 码的情况进行了比较，给出了最终的测试结果。

测试结果表明，本文设计的传输算法在丢包率不超过 30%的无线网络环境下能够有效改善数据丢失引起的图像失真，显著提高接收端的视频呈现效果。

6.2　展望及改进建议

本课题研究的纠删码和可靠 UDP 相结合的方法为视频在无线网络中的传输提供了可靠性保障，基本实现了研究目标。然而，针对算法本身在无线环境中的应用，仍然存在以下问题有待改进：

（1）UEP Raptor 通过经验值方法得到的编码比等参数具有一定的局限性。由于视频发送端运行在嵌入式系统上，考虑到系统资源对视频采集和处理速度的

限制，本文使用分辨率为 480×272 的视频图像序列进行实验。然而，分辨率不同的图像样本在相同的丢包率等条件下的传输效果可能存在差异。下一步可以采用诸如 512×512、1024×1024 等多种分辨率的视频样本进行实验，以得到更具普遍性的结果。

（2）可靠 UDP 缺乏拥塞控制。考虑到无线网络的带宽有限，本文设计的可靠 UDP 算法尽量限制发送到网络中的控制包数量，因此放弃了拥塞控制算法，仅保留了核心的数据确认和重传技术。

（3）纠删码和可靠 UDP 结合的方式有限。本文设计的纠删码和可靠 UDP 的结合方法只包含接收端丢包率对发送端发送速率和编码强度的简单调控，没有通过详尽的公式推导得出考虑了 H.264 编码参数和 Raptor 编码参数之后确切的函数关系。因此，下一步可以从这部分出发，研究丢包率对发送速率和编码强度更精确的调控方法。

最后，希望从上述问题出发对本算法进行进一步改进和完善，使之在无线视频传输应用中具有更普遍的适用性和更良好的性能表现。

参考文献

[1] 维基百科．IEEE802.11[EB/OL].http://zh.wikipedia.org/zh/IEEE_802.11, 2015-01-13.

[2] 齐江．无线局域网发展概述[J]．中国数据通信, 2002, 4(7): 6-10.

[3] Pengrui Duan, Liang Liu, and Zhao Zhang, A Cross Layer Video Transmission Scheme Combining Geographic Routing and Short-Length Luby Transform Codes[J], International Journal of Distributed Sensor Networks(accepted).

[4] 维基百科. 通用移动通讯系统[DB/OL].http://zh.wikipedia.org/wiki/通用移动通讯系统, 2013-12-12.

[5] 马华东，陶丹．多媒体传感器网络及其研究进展[J]. 软件学报，2006, 09:2013-2028.

[6] 栾学宣. 无线网络传输在视频监控中的应用[J].中国新通信，2013, 15(9): 57-57.

[7] 慕建君，焦晓鹏，曹训志. 数字喷泉码及其应用的研究进展与展望[J]．电子学报, 2009, 37(7): 1571-1577.

[8] Gomez-Barquero D, Gozálvez D, Cardona N. Application Layer FEC for Mobile TV Delivery in IP Datacast Over DVB-H Systems[J]. IEEE Transactions on Broadcasting, 2009，55(2): 396-406.

[9] Bouras C, Kanakis N, Kokkinos V, et al. AL-FEC for streaming services over LTE systems[A]. //Proc of the 14th International Symposium on WPMC[C], Brest: IEEE, 2011: 1-5.

[10] Chen S T, Chiao H T, Chang S Y, et al. An HD streaming system for WiFi multicast channels based on application-layer FEC[A]. //Proc of the 17th International Symposium on ISCE[C]，Hsinchu: IEEE, 2013: 85-86.

[11] 刘国，于文慧，吴家骥，等. 基于系统 Raptor 码不等差错保护的图像压缩传输[J]．电子与信息学报, 2013, 35(11): 2554-2559.

[12] Wu Y, Kumar S, Hu F, et al. Cross-layer forward error correction scheme using raptor and RCPC codes for prioritized video transmission over wireless channels[J]. IEEE transactions on Circuits and Systems for Video Technology, 2014, 24(6): 1047-1060.

[13] Zheng H, Boyce J. An improved UDP protocol for video transmission over internet-to-wireless networks[J]. IEEE Transactions on Multimedia, 2001, 3(3): 356-365.

[14] Sun Q, Li H. Research and application of a UDP-based reliable data transfer

protocol in wireless data transmission[A]. //Proc of the International Conference on CSSS[C], Nanjing: IEEE, 2011: 1514-1516.
[15] Chang S Y, Chiao H T, Yeh X Y, et al. UDP-based file delivery mechanism for video streaming to high-speed trains[A]. //Proc of the 24th International Symposium on PIMRC[C], London: IEEE, 2013: 3568-3572.
[16] Bova T, Krivoruchka T. Reliable UDP Protocol[S], IETF Internet-Draft, 25 February 1999.
[17] Le T, Kuthethoor G, Hansupichon C, et al. Reliable user datagram protocol for airborne network[A]. //Proc of the Military Communications Conference on MILCOM[C], Boston: IEEE, 2009: 1-6.
[18] Atya A O F, Kuang J. RUFC: A flexible framework for reliable UDP with flow control[A]. //Proc of the 8th International Conference for ICITST[C], London: IEEE, 2013: 276-281.
[19] 张维勇, 钱军, 王建新. 基于 UDP 协议的视频图像可靠传输的研究和实现[J]. 合肥工业大学学报: 自然科学版, 2008, 31(5): 698-700.
[20] 靳海力, 李俊. 具有补发机制的增强型可靠 UDP 的实现[J]. 小型微型计算机系统, 2010, 31(5): 904-907.
[21] 董帅甫. 基于局部可靠 UDP 的无线视频传输系统[D]. 辽宁: 大连理工大学, 2013: 27-49.
[22] Seferoglu H, Altunbasak Y, Gurbuz O, et al. Rate distortion optimized joint ARQ-FEC scheme for real-time wireless multimedia[A]. //Proc of the International Conference on ICC[C], London: IEEE, 2005: 1190-1194.
[23] Moid A, Fapojuwo A. Heuristics for jointly optimizing FEC and ARQ for video streaming over IEEE802. 11 WLAN[A]. //Proc of the International Conference on WCNC[C], Las Vegas: IEEE, 2008: 2141-2146.
[24] Zhang J, Liang W, Wu J, et al. A Novel Retransmission Scheme for Video Services in Hybrid Wireline/Wireless Networks[A]. //Proc of the 71st VTC[C], Taipei: IEEE, 2010: 1-5.
[25] 宋永献. 无线传感器网络数据可靠传输关键技术研究[D]. 江苏: 江苏大学, 2014: 30-56.
[26] 毕厚杰. 新一代视频压缩编码标准——H.264/AVC[M]. 北京: 人民邮电出版社, 2004: 97-133.
[27] 郭春梅, 毕学尧. 纠删码的分析与研究[J]. 信息安全与技术, 2010, 1(9): 38-42.

[28] 张晓晨，苑林，李晓光.数字喷泉码的研究[J]. 科技传播, 2011, 1(3): 217-218.
[29] 石东新，杨占昕，张铨. 3GPP MBMS 中 Raptor 编解码研究[J]. 数据采集与处理, 2010, 25(S): 121-124.
[30] Bouras C, Kanakis N, Kokkinos V, et al. Evaluating RaptorQ FEC over 3gpp multicast services[A]. //Proc of the International Conference on IWCMC[C], Limassol: IEEE, 2012: 257-262.
[31] Ahmad S, Hamzaoui R, Al-Akaidi M M. Unequal error protection using fountain codes with applications to video communication[J]. IEEE Transactions on Multimedia, 2011, 13(1): 92-101.
[32] Vukobratovic D, Stankovic V, Sejdinovic D, et al. Scalable video multicast using expanding window fountain codes[J]. IEEE Transactions on Multimedia, 2009, 11(6): 1094-1104.
[33] Cataldi P, Grangetto M, Tillo T, et al. Sliding-window raptor codes for efficient scalable wireless video broadcasting with unequal loss protection[J]. IEEE Transactions on Image Processing, 2010, 19(6): 1491-1503.
[34] C Hellge, D Gomez-Barquero, T Schierl, and T Wiegand. Layeraware forward error correction for mobile broadcast of layered media[J]. IEEE Transactions on Multimedia, 2011, 13(3): 551–562.
[35] Luo Z, Song L, Zheng S, et al. Raptor Codes Based Unequal Protection for Compressed Video According to Packet Priority[J]. 2014.
[37] 谢希仁. 计算机网络第五版[M]. 北京: 电子工业出版社, 2006: 136-148.
[38] FFmpeg. ffmpeg [EB/OL].http://ffmpeg.org/，2013-10-28.
[39] Vjacheslav Trushkin. SDL[EB/OL].http://forums.libsdl.org/，2013-01-17.
[40] 杨志伟. 嵌入式实时视频传输系统实现技术研究[D]. 陕西: 西安电子科技大学, 2005: 23-51.
[41] Blanchette J, Summerfield M. C++ GUI programming with Qt 4[M]. New York: Prentice Hall Professional, 2006: 213-359.
[42] 兰帆，张尧弼. 基于 Gilbert 模型的网络丢包仿真[J]. 计算机工程, 2004, 30(S): 200-203.

致谢

转眼之间，硕士研究生学习生涯已经进入尾声。在这两年的时间里，认识了挺多人，学了挺多知识，也做了挺多思考。回首往昔，有过焦虑、有过忙碌、有过失落、有过遗憾，但同时也有充实、有满足、有快乐、有希望、有期待，甚至更多更多。所有这些构成了我全部的研究生生活。现在算来，欢乐多过伤感、付出总有收获，真的觉得自己获得了一种人生记忆中的美好和难忘。在本文也将画上最终的句号之时，我要对学习、工作和生活中帮助过我的人致以真诚的谢意。

首先要感谢我的导师，马华东老师。在课题完成的过程中，马老师给予了我严格的督促和悉心的指导。他渊博的专业学识、严谨的治学态度和认真地教导风格让我受益匪浅，更终生难忘。

其次，我想感谢段鹏瑞老师和刘亮老师。段老师对待学生温和可亲，对待科研严肃认真。工作中经常与我们积极地讨论，给我们以启发式的建议和指导。生活上经常关心我们的身体健康和心情舒畅。刘老师博学多才、匠心独运，总能在我们的学习和研究遇到瓶颈时提供创新性的指导。高标准要求工作，积极态度对待生活，是个难得的好老师。

最后，我要感谢实验室小组的成员。与他们的相处和合作非常愉快，总能让我在轻松愉悦的氛围中发现工作和生活的乐趣。大家一起布环境、做实验、调程序的时光我永远都不会忘记。

作者攻读学位期间发表的学术论文目录

[1] 徐盈盈，段鹏瑞.水质监测系统中数据采集模块的设计与实现.中国科技论文在线.2014 年 12 月.论文编号 201412-507.