

# 无线视频传输中 MPEG-4 自适应分包算法

许昂兢, 罗志祥, 曹明翠

(华中科技大学光电子科学与工程学院, 武汉 430074)

**摘 要:** MPEG-4 视频流在无线信道中传输时, 视频数据必须符合一定的结构, 以便当信息丢失和比特错误时, 能够进行差错控制。将视频净载荷按一定结构进行适当分组处理称为打包或者分包。该文对分包的理论模型进行了分析, 提出了一种兼顾封装效率和图像质量的自适应分包算法, 并利用实时传输协议在 GPRS 无线信道中进行传输, 取得了理想的视觉效果。

**关键词:** MPEG-4; 分包; 有效误码率; 实时传输协议

## Adaptive Packetization of MPEG-4 Visual Stream in Wireless Transmission

XU Guo-jing, LUO Zhi-xiang, CAO Ming-cui

(School of Optoelectronic Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074)

**【Abstract】** When the MPEG-4 visual stream is transferred in wireless channel, the visual data must be in accordance with certain structure for better error control when the bit error happens. The grouping of the payload of the visual data according to certain structure is called as packetization. The model of the packetization is analyzed and an algorithm of adaptive packetization based on both encapsulation efficiency and video quality is proposed. Experiments using real-time transport protocol over GPRS (general packet radio services) network are used to validate the analysis of the most efficient scheme.

**【Key words】** MPEG-4; packetization; effective error rate; real-time transport protocol

视频流的传输在流媒体传输中具有比较核心的地位, 同时也非常难以保证其在高误码率的无线信道中传输的质量。为了对比比特错误进行控制和恢复, 国内外在早期对这方面做了大量研究<sup>[1]</sup>。MPEG-4 视频编解码标准确定后, 这些研究结果也被沿用了 MPEG-4 的标准中<sup>[1]</sup>。

但是 MPEG-4 在设计思路与 MPEG-2 和 H.263 具有一定的区别, 一个很大的特点就是 MPEG-4 在易发生差错的环境下(如: 无线信道)依靠同步字、数据分割和逆向可变长编码等工具使其自己具有一定的鲁棒性。另外, 使用 TCP 协议进行流媒体传输破坏了流媒体应用的实时性, 所以通常采用 UDP 协议对 MPEG-4 视频数据进行传送。由于 UDP 协议本身只是对数据包的不可靠投递, 无法对乱序、丢包和差错进行处理, 必须采用实时传输协议(RTP)对 MPEG-4 数据进行封装<sup>[2]</sup>, 并用实时传输控制协议(RTCP)来提供可靠的控制。本文遵照文献[2]中定义的 MPEG-4 封装基本准则, 对 MPEG-4 数据的分包长度在一定误码率下对整个数据包差错情况的影响进行了分析, 并且提出一种综合考虑了封装利用率和图像质量的自适应分包方法, 并使用这种方法在 GPRS 无线信道中进行了实现。

### 1 MPEG-4 的 RTP 封装技术

实现 MPEG-4 视频在网络中的高质量传输的关键之一是如何将 MPEG-4 的视频数据封装成 RTP 包。RFC3016<sup>[2]</sup>中定义的封装形式是现在应用最普遍的一种封装形式, 视频封装方法中把视频对象平面(video object plane, VOP)作为封装的基本单元, 将视频直接分布并映射到 RTP 包中。该协议对如何在 RTP 协议中对 MPEG-4 进行划分和封装进行了一些约定和限制。

另外 RFC3016 还给出了几种封装的示例, 图 1 给出了其中最常用的两种, 其他的示例可以在 RFC3016 中查询。



图 1 RTP 组包的 MPEG-4 视频码流封装

在图 1(a)中, 一个视频包被打包到一个 RTP 包中。当网络中包丢失率很高时推荐采用该方法。甚至当包含有 VOP 头的 RTP 包被丢弃时, 其它 RTP 包还可通过使用视频包头中的 HEC 信息进行解码。

图 1(b)为多个视频包打在一个 RTP 包中的情况。在底层网络速率很低时这种组包方式可高效地节约 RTP/IP 开销。不过, 由于一个 RTP 包的丢失会导致将多个视频包同时丢失, 这种方法会降低丢包恢复率。

### 2 改进传输效率的工具

传输错误, 诸如比特位错误或包丢失, 可能会导致视频解码器与待解码的 VLC 不再同步。这样会导致解码器对错误发生后的一些或所有信息不能正确地解码。下面介绍的几种工具能够改进码流的传输性能, 当网络错误发生的概率较高时是非常有效的。

**作者简介:** 许昂兢(1983 - ), 男, 硕士研究生, 主研方向: 网络视频通信, 嵌入式系统; 罗志祥, 副教授; 曹明翠, 教授、博士生导师  
**收稿日期:** 2006-11-14 **E-mail:** normankingxu@smail.hust.edu.cn

## 2.1 视频包

一般把一个 VOP 分成多个视频包,它由一个再同步标记(Sync)、一个头(Header)和一系列以光栅顺序扫描编码的宏块(Macroblock data)等部分组成。再同步标记后面跟着下一个宏块的索引(Macroblock number),这使得解码器能够正确定位视频包的第一个宏块。接着是量化参数(Quant scale)和头部扩展码(HEC)标记。如果 HEC 为 1,其后是当前 VOP 头的拷贝(VOP header),这增加了需要传输的信息量,但使得解码器在第一个 VOP 头有错的情况下还能恢复 VOP 头。所以,视频包可以有效地防止错误的传播和扩散。

## 2.2 数据分割

数据分割<sup>[1]</sup>使得编码器具备在视频包中重新组织编码数据而减少传输错误影响的能力。数据分割将一个 MPEG-4 视频包分成两部分:第一部分是再同步码、视频包头信息和复制 VOP 的头信息、轮廓和运动矢量(如果是 INTRA 帧就是 DC 数据)信息,这一部分对差错的敏感度比较高;第二部分是纹理数据(AC TCOEFF),这个部分对差错的敏感度很低。它们之间用一个再同步码来标示分开。如果第一部分发生误码,那么整个视频包都要被丢弃。当差错不敏感的第二部分纹理数据发生一些误码时,视频解码器可存储视频分组中无差错的运动和轮廓数据,使用差错屏蔽技术进行解码,只会有很小的视觉失真。

## 2.3 基于循环冗余码(CRC)的差错保护

无线信道中误码率很高而且速率很低,如何在保证冗余载荷最小的情况下来进行差错恢复是笔者关心的问题。前向纠错(forward error correction, FEC)技术可以减少差错对解码视频质量的影响。在对图 1(b)中的视频包进行前向纠错之后,当视频包中某个比特发生比特错误时,可以尽量不影响到视频包的其他部分。速率兼容的紧致码(rate-compatible punctured codes, RCPC)和循环冗余码(cyclic redundancy check, CRC)是两种常用的前向纠错的方法。RCPC 的纠错能力很强,但是由于反馈信息和对损伤信号的重传可能会引入过多的延迟,对实时应用不是很适合,而且差错校验码的插入使得比特流不兼容标准视频编解码器;使用 CRC 校验码的差错控制很好地解决了比特流与标准视频解码器不兼容的问题,它在视频包的第二部分数据后面插入两个 16bit 的 CRC 定长码字作为用户数据,这样无论有没有错误发生,用户解码的时候都可以跳过这两个 CRC 校验码进行解码<sup>[3]</sup>。两个 CRC 校验码的插入如图 2 所示。

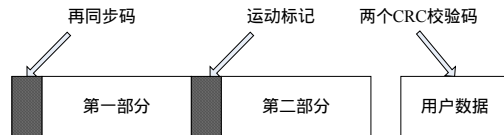


图 2 将 16 比特校验码插入 MPEG-4 视频包中的示意图

## 3 RTP 分包与误码分析

文献[4]提出了两种分包机制,分别是图 1 中的两种封装机制结合视频包、数据分割和插入 CRC 校验码的方法得到的。对图 1 中提出的两种分包算法进行有效误码率的分析发现,第一种分包方式效率明显低于第二种机制<sup>[4]</sup>。本文只列出第二种分包机制的有效误码率分析过程。

### 3.1 视频包有效误码率分析基本思想

首先,必须对解码器在解码遇到误码时所采用的丢弃规则进行说明:当 IP/UDP/RTP 包头产生错误时,整个 RTP 包

都被丢弃;如果第一部分或者第一部分的 CRC 校验码发生错误也会导致整个视频包的丢弃;如果第二部分发生错误,则从错误比特开始,视频包余下的部分被丢弃;如果第二部分的 CRC 校验码出现错误则丢弃整个第二部分。

为了更加清楚地解释有效误码率的基本分析方法,本文把封装模型进行简化,假设视频包只由同步字和载荷组成,载荷长度为  $L$ ,而且该视频包第一个错误发生在载荷的  $m+1$  个比特处。

在解码器接收到这个视频包后,检测出第  $m+1$  个比特有错误后,从这个比特开始到下一个同步字之间的数据全部丢弃。如果单个比特的信道误码率为  $e_{ch}$ ,误码对整个视频包造成的有效误码率为  $e_{eff}$ ,可以得到:

$$e_{eff} = \frac{e_{ch}}{L} \sum_{m=1}^{L-1} (1-e_{ch})^m (L-m) \quad (1)$$

### 3.2 分包机制的有效误码率分析

当一个视频包进行数据分割和 CRC 冗余校验之后,其有效误码率就不能再简单地采用式(1)进行计算。

在进行误码率分析之前,必须对一些变量进行定义。 $L$  为整个视频包的长度; $X$  为视频包第一部分的长度; $Y$  为视频包第二部分的长度,如果假设  $\alpha$  为第一部分占整个视频包长度的百分比,那么可以得到:

$$\alpha = \frac{X}{X+Y} = \frac{X}{L} \quad (2)$$

根据 3.1 节中讨论的丢弃规则和有效误码率的计算方法,可以得到第一种分包机制各个部分的误码情况对整个 RTP 包的有效误码率的影响。

由式(2)可以得到  $X = \alpha L$  和  $Y = (1-\alpha)L$ ,另外,假设  $a = 1 - e_{ch}$ 。

第二种分包机制中含有多个视频包,但是视频包的大小并不是完全一样。由于每次都必须把一个完整的 MB 包含到一个视频包中,一般视频包中尽可能多装入 MB,当发现下一个视频包可能会导致视频包超过预设值时,就再开始一个新的视频包。由于 MB 的大小一般比较小,因此使视频包的预设值近似于视频包的长度,因而可以得到如果一个 RTP 包中有  $k$  个视频包,那么前  $k-1$  个视频包的长度都是一样的,第  $k$  个视频包为所有最后剩下的兆字节数,其大小肯定不大于其他视频包的大小。由此可以得到前  $k-1$  个视频包的长度为

$$L_1 = L_2 = \dots = L_{k-1} = L_p \quad (3)$$

而第  $k$  个视频包的长度为

$$L_k = L_v - 336 - (L_p + 16) \left\lfloor \frac{L_v - 320}{L_p + 16} \right\rfloor \quad (4)$$

其中,  $L_v$  是整个视频对象面(visual object plane, VOP)和协议比特的总长度。所以可以得到协议信息头部分的有效误码率为

$$e_{eff1} = \sum_{n=0}^{319} (1-e_{ch})^n e_{ch} = 1 - (1-e_{ch})^{320} \quad (5)$$

接下来计算单个视频包中各部分的有效误码率。假设  $e_{eff2}$  为视频包中第一部分数据的有效误码率;  $e_{eff3}$  为视频包中第二部分数据的有效误码率;  $e_{eff4}$  为视频包中第一部分数据 CRC 校验码的有效误码率;  $e_{eff5}$  为视频包中第二部分数据 CRC 校验码的有效误码率。  $e_{eff2} \sim e_{eff5}$  的值如下所示:

$$e_{eff2} = \frac{L}{L_v} (1-e_{ch})^{320} (1 - (1-e_{ch})^X) \quad (6)$$

$$e_{eff3} = \frac{a^{320+X}}{e_{ch} L_V} (a^{Y+1} + e_{ch} (Y+1) - 1) \quad (7)$$

$$e_{eff4} = \frac{L}{L_V} (1 - e_{ch})^{L+320} (1 - (1 - e_{ch})^8) \quad (8)$$

$$e_{eff5} = \frac{Y+8}{L_V} (1 - e_{ch})^{L+328} (1 - (1 - e_{ch})^8) \quad (9)$$

这样可以得到在第二种分包机制中，单个视频包的有效误码率是一个以视频包长度  $L$  为变量的表达式：

$$e_{peff}(L) = e_{eff2}(L) + e_{eff3}(L) + e_{eff4}(L) + e_{eff5}(L) \quad (10)$$

所以对于整个 RTP 包，其有效误码率可以计算得到

$$e_{eff} = e_{eff1} + \left[ \frac{L_V - 320}{L + 16} \right] e_{peff}(L) + e_{peff}(L_k) \quad (11)$$

无线信道的 MTU 一般为 576B，所以本文采用的  $L_V$  值为  $576 \times 8 = 4608$ 。假设  $\alpha = 0.3$ ，可以得到 3 种不同信道误码率下，视频包长度对有效误码率的影响，如图 3 所示。

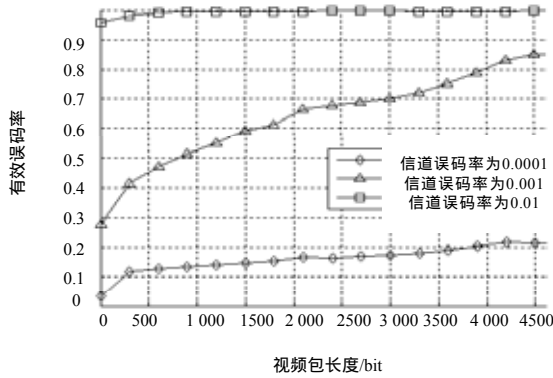


图 3  $\alpha = 0.3$  时在不同视频包长度下的有效误码率曲线

如图 3，误码率为 0.001 时，在视频包长度大于 1000bit 之后，整个 IP 包的有效误码率已经大于 50%，超过了能容忍的限度。实际的 GPRS 的 CS-1 信道比特差错率 (BER) 为  $10^{-4}$ ，有效误码率在可以容忍的范围内，所以在后面的计算中， $e_{ch}$  的值都假设为  $10^{-4}$ 。

#### 4 视频包的最佳长度

对于视频包的长度的选择，研究人员曾做过大量的研究工作<sup>[4]</sup>。Worrall 等人提出的根据第一部分与整个视频包的比值  $\alpha$  来判断视频包最佳长度的办法，是建立在对一些标准测试序列进行测试的基础上，得出的经验性的结论如下：

$$L = \begin{cases} 1200 - 2000\alpha & 0.1 < \alpha < 0.4 \\ 400 & \alpha > 0.4 \\ 1000 & \alpha < 0.1 \end{cases}$$

这个模型的缺点是没有太多的理论依据，而且没有考虑到带宽的利用率。

##### 4.1 视频封装的利用率

本文提出了一个基于单位带宽内的有效利用率最大原则的理论模型，来计算视频包的最佳长度，使得带宽利用率和视频质量均得到提高。模型的意义在于，如何调整视频包的长度使得单位带宽内发送最多的正确数据。为了在无线信道有限的带宽内使得接收到的数据错误率最小，考虑到信道误码和封装效率在这里引入一个有效封装利用率的公式：

$$U = \frac{L_{payload}}{L_{whole}} (1 - e_{eff}) \quad (12)$$

结合式(12)和式(11)中的结果  $e_{eff}$ ，计算得出，在第二种封装机制下，封装利用率为

$$U_{whole} = \frac{L_V - 320 - \left( \frac{L_V - 320}{L + 16} + 1 \right) \times 50}{L_V} (1 - e_{eff}) \quad (13)$$

#### 4.2 结合利用率和误码率的综合评估

只考虑封装利用率是不够的，因为用户不仅需要单位时间内有尽可能多的正确数据传送到用户端，而且也希望接收到每一帧图像质量都能够在容忍的范围之内。所以，在实际应用中，在考虑封装利用率的同时必须考虑到有效误码率本身的值，也就是说，信道带宽的利用率和最后的图像质量都是很重要的。本文提出一个综合的方案，使用式(14)来对封装效率和图像质量进行整体的评估。

$$S = U_{whole} \times \left( \frac{1}{e_{eff}} \right)^\beta \quad (14)$$

式(14)中的  $\beta$  为质量因子； $S$  是综合封装利用率和图像质量的结果。 $\beta$  的取值决定了最后的结果对于图像质量的偏重。假设封装利用率和图像质量同样重要， $\beta$  取 1，则可以得到：

$$S = U_{whole} \times \frac{1}{e_{eff}} \quad (15)$$

使用 Matlab 工具对式(15)从  $\alpha = 0.1$  到  $\alpha = 1$  时进行峰值搜索，得到不同  $\alpha$  值下使得  $S$  值最大的最优视频包长度曲线以及二次拟合和三次拟合曲线，如图 4 所示。

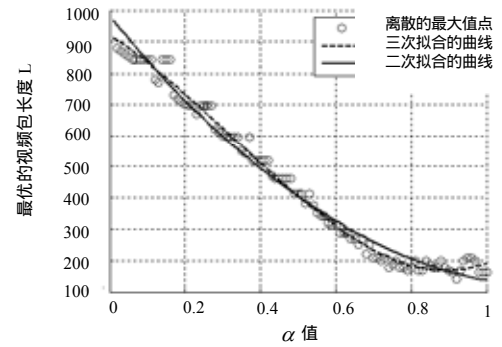


图 4 不同  $\alpha$  值下的最优视频包长度曲线

得到三次拟合曲线系数为 [1165.7 1145.6 -747.4 919.2]，二次拟合曲线系数为 [620.4 -1464.4 981.0]。从图 4 中可以看到，三次拟合的曲线基本与原始数据保持一致，在后面的实验中，采用的就是三次拟合的方法。

#### 5 实验结果

实验使用 S3C2440 ARM 嵌入式开发板作为编码发送端，连接 GPRS 发送模块，使用 PC 机作为接收端，测试数据为  $176 \times 144$  的 Suzie 标准测试序列。采用文献[4]中的方案与本文方案进行对比，得到最后接收到的图像序列的 PSNR 值曲线如图 5 所示。

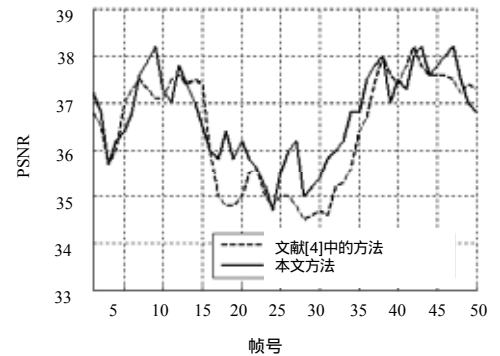


图 5 文献[4]中方法与本文方法的比较(Suzie 序列)

可以发现，Suzie 在第 20 帧附近会有一个较大的头部晃 (下转第 275 页)