doi:10.3772/j.issn.1002-0470.2013.07.008

考虑 SVC 数据特性的 P2P 流媒体分片算法①

宋俊平②* 周 旭③* 张 棪* 唐 晖* 白 帆* 赵志峰**

(*中国科学院声学研究所高性能网络实验室 北京 100190)

(**华数传媒网络有限公司 杭州 310012)

摘 要 为解决 P2P 流媒体系统在异构环境下传输和共享视频数据的问题,对广泛应用于基于可扩展视频编码(SVC)的 P2P 流媒体系统中的等时长分片算法进行了研究,研究结果表明,该算法对 SVC 数据特性考虑不够充分,当网络丢包率高时可能导致视频质量严重下降。为了解决这一问题,提出了不等时长分片算法。该算法通过调整分片播放时长控制各层数据成功传输概率,优先保证重要性高的分层的正确传输。实验和仿真结果表明,与等时长分片算法相比,不等时长分片算法令分片传输时间更加稳定,且大大提高了重要性高的分层的成功传输概率,能够提升多种网络中客户端的视频质量,更加适合于应用在异构网络中。

关键词 分片算法, P2P 流媒体, 可扩展视频编码(SVC), 异构

0 引 言

随着移动互联网的发展和三网融合产业的兴起,终端之间的数据共享将更加频繁与普遍,当前互联网上广泛应用的 P2P 流媒体技术将获得更大的发展空间,但如何在异构的环境中传输和共享视频数据成为当前 P2P 流媒体系统所面临的主要问题。为了解决这一问题,多种自适应技术被提出,包括转码(transcoding)[1]、联播(simulcast)[2]、多描述编码(multiple description coding)[3]和可扩展视频编码(scalable video coding, SVC)[4]等。其中 SVC 由于其高效的编码效率、方便的自适应性和高共享度而受到工业界和学术界的广泛关注。SVC 可将视频编码为一个基础层(base layer, BL)和若干个增强层(enhancement layer, EL),基础层可独立解码并获得一个基本质量的视频,而增强层可以从空间、时间和质量三个维度对基础层视频质量进行扩展,解码的增强层越多获得的视频质量也就越高。应用 SVC 技术后,一方面 P2P 流媒体用户可以根据条件选择下载不同数目的增强层,易于进行质量自适应;另一方面用户间可以层为单位共享数据,提高数据共享效率。近年来基于 SVC 的 P2P 流媒体技术正在成为互联网视频领域的研究热点。

在 P2P 流媒体系统中数据共享的基本单元为分片(chunk)。在基于单层视频编码的 P2P 流媒体系统中,视频数据被分为等大小的分片进行分发[5]。当引入 SVC 后,为了提高数据分享效率并易于质量自适应,每个分层被作为一个相对独立的单元在网络中分发,因此分片的对象也是每个分层。然而由于层间数据具有相关性,因此客户端需要保证多层数据下载的同步,此时单纯的等大小分片方法不能满足这种层间同步性的需求。为了解决这一问题,有必要对现有的分片算法进行改进。目前为了满足层间下载同步性要求而提出的分片算法可以分为两类:第一类在等大小分片算法的基础上进行改进,分片大小仍保持相等[6,7];第二类为等时长分片算法,用此类算法,每个分片具有相等的播放时长,但分片大小不等[8-11]。其中第二类方法由于适用范围广且复杂度低而受到广泛应用。

等时长分片算法虽然解决了层间数据同步问题,但它对 SVC 数据特性的考虑并不充分,当网络丢包率高时可能导致视频质量严重下降。本文通过实验对 SVC 的数据特性进行了深入研究,结果表明 SVC 各层的数据量相差较大,且低层数据量常常大

① 国家科技重大专项项目(2010ZX03004-001, 2011ZX03005-004-02, 2011ZX03002-001-01)和国家自然科学基金(61102076)资助项目。

② 女,1985 年生,博士生;研究方向:P2P,流媒体技术等;E-mail: songjp@hpnl.ac.cn

③ 通讯作者,E-mail: zhoux@hpnl.ac.cn

(收稿日期:2012-07-19)

于高层数据量。由于等时长分片算法将各层分为相等数目的分片,因此低层分片的平均大小可能远大于高层分片。显然,在丢包率相等的情况下,分片越大成功传输的概率越小,因此低层分片的传输成功率通常低于高层。然而在 SVC 中,高层数据的解码依赖于低层,如果低层数据丢失,则高层数据即便正确传输也不能解码。由此可知,当网络丢包率高时,等时长分片算法可能导致视频质量的严重下降。为了提高重要性高的低层数据的传输成功率,本文改进了等时长分片算法,并提出了不等时长分片算法。该算法通过空间、时间和质量三个维度的层间数据量关系确定各层分片的播放时长。采用这一算法后层内分片播放时长相等,层间分片播放时长可成比例变化。总体来讲,各层分片的平均大小与其重要性成反比。我们用实验和仿真的方法对不等时长分片算法的性能进行了验证,结果表明本文提出的分片算法可以提高低层分片的传输成功率,尤其是在网络丢包率较高时,可大大提升用户体验。

1 SVC 数据特性分析

本节通过实验对 SVC 的数据特性进行研究。首先分别从空间、时间和质量三个维度观察分层数据量变化规律,并找出分层数据量整体变化趋势与三个维度上变化趋势的关系。然后观察分层内部视频数据的分布情况。最后分析 SVC 数据特性对分片算法的影响。

1.1 实验背景

SVC 可以从空间、时间和质量三个维度对视频质量进行扩展,因此它的每个分层都用一个三维的 (D, T, Q) 值标识,其中 D 为空间层标识,T 为时间层标识,Q 为质量层标识。使用 SVC 编码时,视频数据在传输或存储之前首先被映射或封装进网络提取层(network abstract layer,NAL)单元(NAL Unit,NALU)中,以便于在基于包的网络中进行传输。NALU 的头中包含 D、T、Q 的信息,(D, T, Q) 值相同的 NALU 组成同一分层。

本文选取多个视频片段进行实验,所有视频片段均从电影或电视节目中截取,播放时长均为 10min。用 JSVM Version 9.15[12] 对这些视频片段进行编码,编码参数也都相同。生成的 SVC 比特流中一共包含 18 个分层,其中 8 层的分辨率为四分之一通用中间格式(quarter common intermediate format,QCIF),帧率分别为1.5265、3.1250、6.25和12.5fps。另外 10 层的分辨率为 CIF,帧率分别为1.5265、3.1250、6.25、12.5和 25fps。此外,QCIF 和 CIF 均采用中等粒度质量分级(medium-grain scalability,MGS)进行编码,其中 QCIF 的量化步长分别为 34 和 28,CIF 的量化步长分别为 36 和 30。详细的比特流分层信息如图 1 所示。

Contained Layers:

Layer	Resolution	Framerate	DFQ
0	176x144	1.5625	<0,0,0>
1	176x144	3.1250	<0,1,0>
2	176x144	6.2500	<0,2,0>
3	176x144	12.5000	<0,3,0>
4	176x144	1.5625	<0,0,1>
5	176x144	3.1250	<0,1,1>
6	176x144	6.2500	<0,2,1>
7	176x144	12.5000	<0,3,1>
8	352x288	1.5625	<1,0,0>
9	352x288	3.1250	<1,1,0>
10	352x288	6.2500	<1,2,0>
11	352x288	12.5000	<1,3,0>
12	352x288	25.0000	<1,4,0>
13	352x288	1.5625	<1,0,1>
14	352x288	3.1250	<1,1,1>
15	352x288	6.2500	<1,2,1>
16	352x288	12.5000	<1,3,1>
17	352x288	25.0000	<1,4,1>

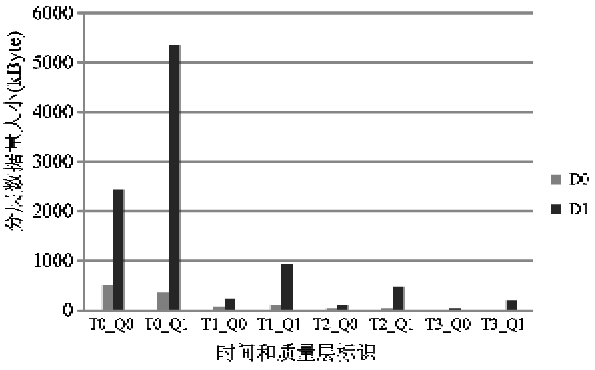
图 1 SVC 流中的分层信息

1.2 分层数据量变化规律

由于本实验在空间、时间和质量三个维度上都进行了分层,因此层的结构比较复杂。为了便于分析,当研究其中一个维度的变化时假设另外两个维度是不变的。例如,当研究质量分层时,层(1,2,1)为层(1,2,0)的增强层,而当研究空间分层时,层(1,2,1)则是层(0,2,1)的增强层。此外,本文用 T_i 表示时间标识为 i 的层的集合,同理,D_i 和 Q_i 分别表示空间和质量标识为 i 的层的集合。为了便于描述,将 T_0 称为时间基础层,同理,D_0 和 Q_0 分别为空间基础层和质量基础层。

空间维度

由于每个空间分层 i 都是空间层标识为 i 的多个层的集合,因此通过两空间层中相应层数据量的对比来观察分层数据量的变化规律。其中一个视频片段的对比结果如图2所示,从其他视频片段可以



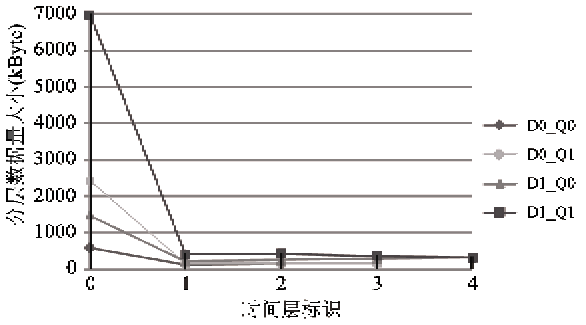
其中 Ti_Qj 表示该层的时间分层标识为 i,质量分层标识为 j

图 2 D_0 和 D_1 中相应层的数据量大小对比

得出类似结果。从图中可以看出,空间增强层要远大于空间基础层。造成这一结果的主要原因是 D_1 中像素的个数是 D_0 中的 4 倍,因此用于记录的信息量也会相应增大。

时间维度

与空间维度类似,通过对比各时间层中相应分层的数据量大小来分析时间维度数据量的变化趋势。从图 3 中可以看出:T_0 中的层远大于 T_1 中的层,而当时间层标识大于 1 时,随着时间标识的增大,层的大小稍有增大。需要指出的是,这一变化趋势是受编码方式、帧间间隔以及帧数的变化共同影响的。

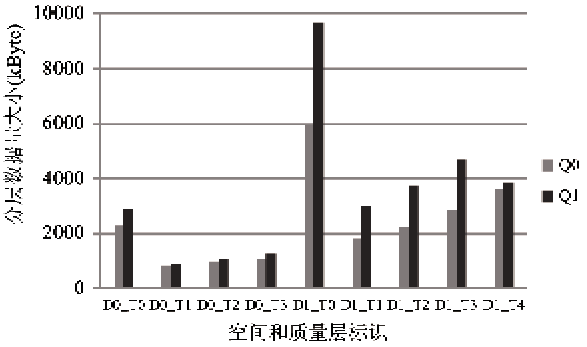


其中 Di_Qj 表示该层的空间分层标识为 i,质量分层标识为 j

图 3　各时间分层数据量大小对比

质量维度

两个质量层中相应层的对比结果如图 4 所示。可以看出,质量增强层一般大于质量基础层。一般来讲,质量分层的大小受编码参数中量化步长的影响,量化步长越小,层的数据量越大。



其中 Di_Tj 表示该层的空间分层标识为 i,时间分层标识为 j

图 4　各质量分层数据量大小对比

三个维度的综合

以上分别观察了空间、时间和质量三个维度下 SVC 的分层数据量变化趋势。当三个维度的分层编码综合使用时分层数据量的变化趋势也是以上三个维度的综合。即层 (i, j, k) 的大小 $S(i, j, k)$ 受 i、j 和 k 三个值的共同影响。以图 5 为例,通过对比 4 个视频片段中所有分层的数据量大小可以看出层(1,0,1)数据量最大。通过以上分析可知,这是由于(1,0,1)层具有最大的空间和质量层标识并且时间层标识为 0,受三个维度编码的共同影响,层(1,0,1)的数据量达到峰值。根据以上分析和讨论,$S(i, j, k)$ 可以近似表示为

$$S(i, j, k) = S(0,0,0) \times D(i) \times T(j) \times Q(k) \quad i, j, k \in N \quad (1)$$

其中 $S(0,0,0)$ 为基础层数据量,$D(i)$、$T(j)$ 和 $Q(k)$ 分别为空间、时间和质量维度的分层数据量变化函数。一般来讲,$D(i)$ 和 $Q(k)$ 分别为 i 和 k 的单调递增函数。$T(j)$ 为非单调函数,j 为 1 时 $T(j)$ 最小,j 增大或减小时 $T(j)$ 均增大,但 j 为 0 时 $T(j)$ 最大。

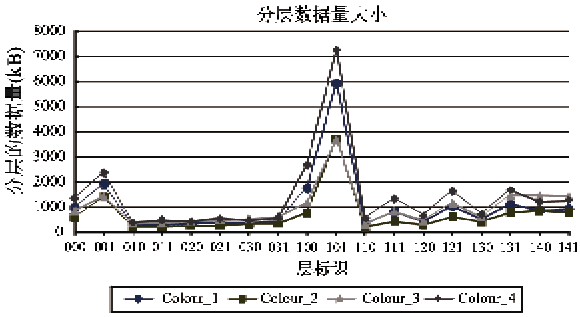


图 5　所有分层大小对比

1.3　层内视频数据分布

为了观察编码后视频数据在各层中的分布,从一个视频片段的 4 个分层中选择了 20 个连续的 NALU 进行对比。这 4 个层分别为(0,0,0)、(0,0,1)、(1,0,0)和(1,0,1)。由于这 4 个分层具有相同的时间标识"0",因此 4 个层中相同序号的 NALU 属于同一图像。图 6 为这些 NALU 的示意图,其中每个"块"表示一个 NALU,将各层中标号相同的块一起解码可以获得一副具有高分辨率和高清晰度的图像。从图 6 中可以看出:(1)同一图像在不同层中的数据量大小差距很大;(2)同一层内不同图像的数据量大小各不相同;(3)层(1,0,1)内图像的数据量明显大于其它分层。其中现象 1 和 2 主要是图像序列特征所致,例如图像序列本身的亮度、色彩、复杂度、运动程度等的不同。现象 3 的原因已经在 1.2 节中进行了说明。

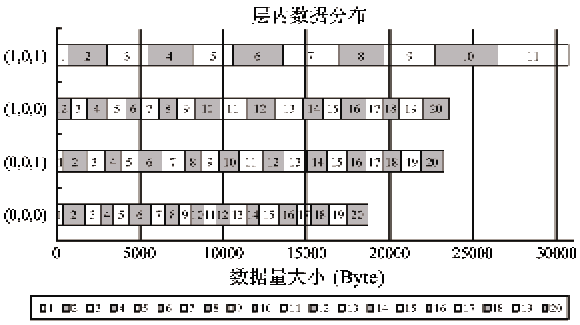


图 6 四个分层中 NALU 的分布情况

1.4 数据特性对分片算法的影响

从 1.3 节可知,不同图像的数据量大小各不相同,且同一图像在不同层中的数据量也相差很大。因此,同一图像的信息在不同层中的分布位置差别很大,此时如果采用等大小分片算法,则可能导致以下问题的出现:(1)同一图像信息分布在不同层中不同序号的分片中,客户端需要额外的索引信息才能对分片进行下载和解码调度;(2)属于同一图像的数据可能被封装到两个甚至多个分片中,进一步增加了数据同步的复杂度。为了解决以上问题,多数研究者在基于 SVC 的 P2P 流媒体系统中采用了等时长分片算法。该算法将各层中需要在同一时间间隔内解码的数据封装为一个个分片,客户端通过分片顺序号进行下载和解码即可满足层间同步性需求。

但是等时长分片算法对 SVC 的数据特性考虑不够全面。在 SVC 中,高层增强层的解码依赖于基础层和低层增强层,因此当低层分片丢失时视频质量下降非常明显。从 1.3 节可知,SVC 中各层数据量相差较大,且低层数据量常常大于高层。例如图 5中(0,0,0)和(0,0,1)层比接下来的(0,1,0)等层大很多。由于等时长分片算法将各层分为相等数目的分片,因此低层分片大小常常大于高层。如前所述,这会导致低层分片成功传输概率低于高层分片。当网络丢包率高时,这一现象尤为明显,会导致用户体验严重下降。

基于以上实验和分析,本文提出了一种不等时长分片算法,该算法在等时长分片算法的基础上进行改进,提高了低层数据的传输成功率,非常适合应用在异构环境中。

2 不等时长分片算法

本节详细介绍不等时长分片算法的流程。该算法通过调整各层分片的播放时长来调节各层分片大小,因此其重点在于如何确定各层分片的播放时长。由于 SVC 可以从三个维度进行扩展,因此层 (i, j, k) 中分片的播放时长 $L(i, j, k)$ 也由三个维度的时长参数 $d(i)$、$t(j)$ 和 $q(k)$ 共同决定,其中 $d(i)$ 为空间时长参数,$t(j)$ 为时间时长参数、$q(k)$ 为质量时长参数。这三个时长参数根据各维度分层数据量变化趋势进行设置,其大小决定了该维度上分片的相对时长。图 7 为不等时长分片算法的流程图,所需参数参照表 1。

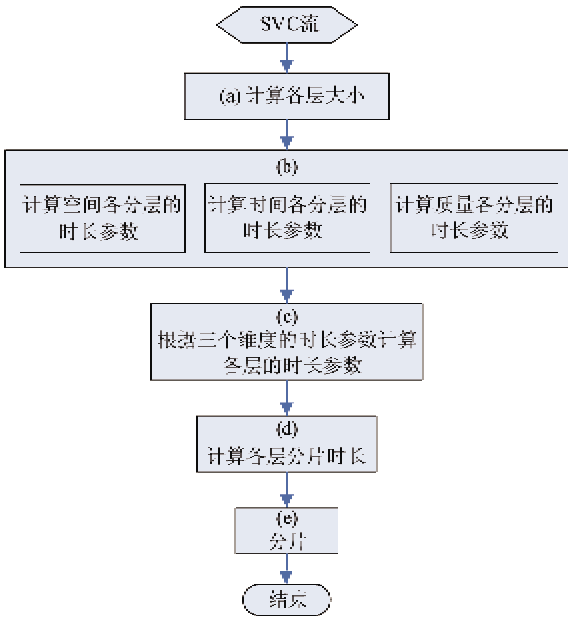


图 7 不等时长分片算法流程图

表 1 参数对照表

参数	描述
l	空间分层数目
m	时间分层数目
n	质量分层数目
$S(i, j, k)$	层 (i, j, k) 的数据量大小
$R(D_i:D_{i-1})$	空间 i 层和 $i-1$ 层数据量之比
$R(T_j:T_{j-1})$	时间 j 层和 $j-1$ 层数据量之比
$R(Q_k:Q_{k-1})$	质量 k 层和 $k-1$ 层数据量之比
$p(i, j, k)$	层 (i, j, k) 的综合时长参数

(a)计算各层大小

首先计算各层数据量大小。由于视频数据被封装在 NALU 中,且 NALU 的头中包含 (D, T, Q) 信息,因此层 (i, j, k) 的数据量可以通过计算 NALU 大小的和获得。通过改写 JSVM 的部分代码即可实现以上功能。

(b)设置空间/时间/质量各分层的时长参数

由于三个维度时长参数的计算过程类似,因此以空间维度为例进行介绍。如图 8 所示,首先将空间各层的时长参数均设为 1。然后从基础层开始逐层对比两个相邻分层的数据量大小。若高层数据量小于低层,则高层时长参数在低层时长参数的基础上整倍数增大,否则高层时长参数等于低层时长参数。以整倍数增大可以保证层间分片的同步性。另外需要指出的是:

(1) 由于每个空间分层 i 都是空间层标识为 i 的多个层的集合,因此两个空间层数据量的比值 $R(D_i:D_{i-1})$ 通过下式计算获得:

$$R(D_i:D_{i-1}) = \frac{1}{m \times n} \times \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} \frac{S(i,j,k)}{S(i-1,j,k)} \quad (0 < i \leqslant l-1) \quad (2)$$

(2) 时长参数变化幅度受 $R(D_i:D_{i-1})$ 大小的影响,两层相差越多,时长参数变化幅度大,如图 8 所示。

```
set d(i)=1, 0≤i≤l-1;
for i from 1 to l-1{
    if (R(D_i : D_(i-1)))<1){
        d(i)=d(i-1)×2^⌈-log2R(D_i:D_(i-1))⌉;
    }
    else{
        d(i)=d(i-1)
    }
    i=i+1;
}
```

图 8　空间时长参数计算过程

空间和质量维度的时长参数计算过程与空间维度类似,不同的是时间和质量层间数据量之比 $R(T_j:T_{j-1})$ 和 $R(Q_k:Q_{k-1})$ 分别以下两式获得:

$$R(T_j:T_{j-1}) = \frac{1}{l \times n} \times \sum_{i=0}^{l-1} \sum_{k=0}^{n-1} \frac{S(i,j,k)}{S(i,j-1,k)} \quad 0 < j \leqslant m-1 \quad (3)$$

$$R(Q_k:Q_{k-1}) = \frac{1}{l \times m} \times \sum_{i=0}^{l-1} \sum_{j=0}^{m-1} \frac{S(i,j,k)}{S(i,j,k-1)} \quad 0 < k \leqslant n-1 \quad (4)$$

(c)根据三个维度的时长参数计算各层综合时长参数

由于各层数据量大小由空间、时间和质量三个维度的变化趋势共同决定,参考式(1),层 (i,j,k) 的时长参数 $p(i,j,k)$ 可由下式计算获得:

$$p(i,j,k) = d(i) \times t(j) \times q(k) \quad (0 \leqslant i \leqslant l-1, 0 \leqslant j \leqslant m-1, 0 \leqslant k \leqslant n-1) \quad (5)$$

(d)计算各层分片时长

层 (i,j,k) 中分片的播放时长 $L(i,j,k)$ 为

$$L(i,j,k) = p(i,j,k) \times T \quad (0 \leqslant i \leqslant l-1, 0 \leqslant j \leqslant m-1, 0 \leqslant k \leqslant n-1) \quad (6)$$

其中 T 为一个常量,其值可以通过经验或实际需要进行设定。通过调整 T 的大小可以调整视频整体分片大小。在下一节的实验中,T 的值被设置为 GOP 播放时长的整倍数,目的是便于客户端解码调度。

(e)分片

最后根据 $L(i,j,k)$ 的值将 SVC 的各层分别分片。在层 (i,j,k) 中,每段 $L(i,j,k)$ 时长内的数据将会被封装为一个分片。

从以上算法可知,各层时长参数由空间、时间和质量三个维度的时长参数共同决定。如果高层数据量小于低层,则低层分片时长参数降低,相应的低层分片播放时长减小,则分片大小减小。通过这一调整改变层间分片大小关系,提高低层分片的成功传输率。

另外需要指出的是,由于不等时长分片算法只需提取 NALU 大小进行计算,与编解码过程无关,因此算法复杂度低。此外由于该算法只在视频被注入网络时执行一次,因此不会带来过多额外开销。客户端下载分片后只需将 NALU 重新组合后解码,其实现方式与等时长分片算法相同,实现复杂度低,不会影响视频的解码播放。

3　性能评估

本节分别通过实验和仿真的方式对不等时长分片算法的性能进行验证。首先分别用等时长和不等时长分片算法对同一视频片段进行分片,通过对比分片结果验证本文提出的算法的性能。然后在 OverSim[13] 仿真平台上搭建 P2P 流媒体仿真系统,并对比两种分片算法对客户端视频质量的影响。

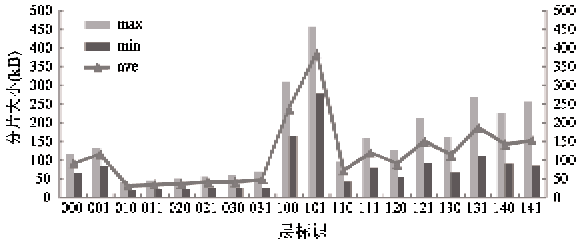
3.1　实验验证

分别采用等时长和不等时长分片算法对同一 SVC 视频片段进行分片,并统计各层分片的最大、最小和平均值。其中一个视频片段的分片结果如图 9 所示,从其它视频片段可以得到类似的结果。需要指出的是,为了保证对比的公平性,采用两种分片算法所得的分片数目近似相等。采用等时长分片算法的分片数目是 450 个,而采用不等时长分片算法的

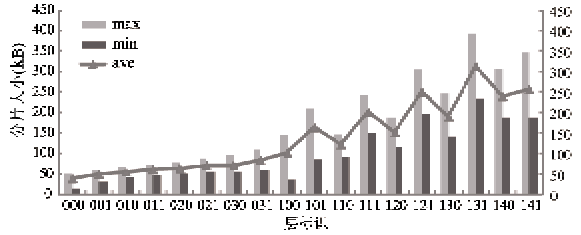
分片数目是 446 个。

从图 9 中可以看出：

(1)采用不等时长分片算法后，整个视频的分片大小差异变小。在图 9(a)中，层(0，1，0)的平均分片大小最小，为 33kB，而层(1，0，1)的平均分片大小最大，为 386kB，后者是前者的 11.7 倍。但是在图 9(b)中，最大和最小平均分片大小的比值为 8.0。当分片大小差距较大时，分片传输时间也相差较大，且不易预测，会加大分片下载调度的复杂度。所以不等时长分片算法可以增强数据传输的稳定性。



(a) 采用等时长分片算法时



(b) 采用不等时长分片算法时

图 9 各层分片的最大、最小和平均值

(2)采用不等时长分片算法后，各层的平均分片大小与该层重要性近似成反比。一般来讲，从层(0，0，0)到层(1，4，1)的重要性逐渐降低。从图 9(b)中可以看出，分层的重要性越高分片大小越小。例如重要性最高的基础层(0，0，0)，其平均分片大小是所有分层中最小的。由于分片越小其成功传输的概率越大，因此，采用不等时长分片算法后，重要性高的分层的传输成功率更高。

3.2 仿真验证

我们在 OverSim 仿真平台上搭建了一个 P2P 流媒体系统，并采用 SVC 格式的视频片段作为源文件。将两种分片算法结果应用在流媒体数据分发过程中，通过对比客户端视频质量验证算法性能。客户端视频质量通过统计分片下载情况获得。视频开始播放后每隔 200ms 统计分片下载情况，如果该时间点上的第 $i+1$ 层分片丢失，且前 i 层分片成功下载，则该时间点的视频质量记为 i。

仿真拓扑图如图 10 所示。网络中包含一个中心路由器和 10 个边缘路由器。tracker 服务器与中心路由器相连，而普通节点则与边缘路由器相连。仿真开始时创建 50 个普通节点，且每个节点的下载进度由一个随机函数决定。客户端接入网络后首先从 tracker 服务器获得邻居节点列表，然后与其中的部分邻居建立连接。客户端通过维护一个下载窗口对分片进行下载调度，如图 11 所示，窗口每隔一段时间滑动一次，只有窗口中的分片才可以被用户下载[14]。窗口中的分片从基础层开始被逐层下载，在某一分层内部，则从距播放点最近的分片逐个下载。需要指出的是，采用不等时长分片算法时，窗口的最小滑动单位为当前最大分片播放时长。

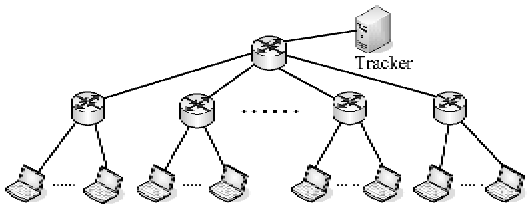
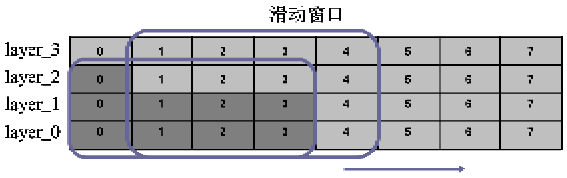
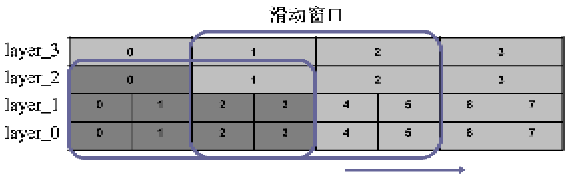


图 10 仿真拓扑图



(a) 采用等时长分片算法时



(b) 采用不等时长分片算法时

图 11 客户端滑动窗口

客户端的带宽被设置为可以下载所有请求的分层。每个分片的丢失率与链路丢包率和分片大小相关，当链路丢包率为 α 时，分片的丢失概率为

$$C_{\text{loss}} = 1 - (1-\alpha)^{\lceil \frac{S}{MTU} \rceil} \quad (7)$$

其中 S 为分片大小，MTU 为网络最大传输单元。仿真中 MTU 设为 1000 Byte，S 的大小根据实际分片结果进行设置。为了提高客户端视频质量，分片丢

失后可以进行重传。基础层的最大重传次数为 3 次,增强层的最大重传次数为 2 次。

首先仿真高速下行分组接入(high speed downlink packet access,HSDPA)网络。由于 HSDPA 网络的丢包率远大于无线局域网(WLAN)网络和固网[15],所以先后将 α 的值设为 1% 和 2%。此外,由于 HSDPA 网络的带宽有限,所以客户端最多请求 6 个分层。图 12 和图 13 是丢包率分别为 1% 和 2% 时客户端上的视频质量示意图。

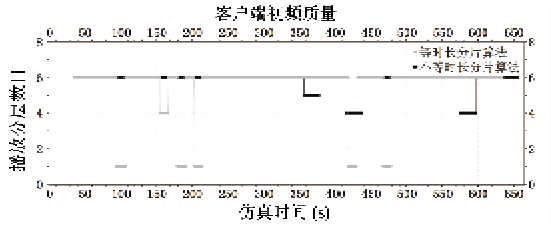


图 12　丢包率为 0.01 时 HSDPA 用户视频质量

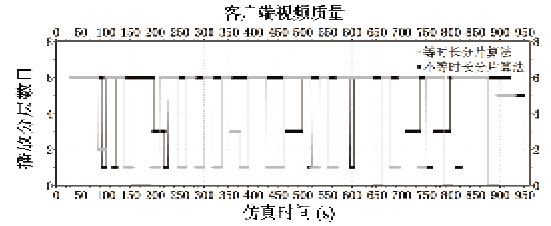


图 13　丢包率为 0.02 时 HSDPA 用户视频质量

从图 12 可以看出,采用不等时长分片算法时,视频质量下降的次数明显降低。这是由于采用不等时长分片算法后,分片大小差距降低,由于分片过大而导致的分片丢失次数降低。除此以外,由于低层分片的丢失概率降低,视频质量下降幅度也明显降低。综合来讲,采用不等时长分片算法后可大大提升客户端视频质量。

从图 13 中可以得出与图 12 类似的结果。除此以外还可以看出,当丢包率增大时,不等时长分片算法的性能更加明显。当丢包率为 1% 时,采用等时长和不等时长分片算法时客户端的平均播放层数分别为 5.5 层和 5.9 层。而丢包率为 2% 时,两种算法中客户端的平均播放层数分别为 4.0 和 5.4 层,质量提升更加明显。由此可知,不等时长分片算法更加适合于应用在丢包率高的移动网络中。

接下来对 WLAN 网络进行了仿真。由于 WLAN 网络的丢包率稍低且带宽较高,因此丢包率设为 1‰,且客户端请求层数设为 18 层。从图 14 中可以看出,采用不等时长分片算法时低层数据成功传输率依然有所提高。另外,由于采用本文所提算法时高层分片大小提高,所以其传输成功概率理应有所降低。但是由于高层视频数据只有在网络状况较好的情况下才会被请求,因此高层分片的传输情况所受影响并不明显,甚至可以忽略,如图 14 所示。

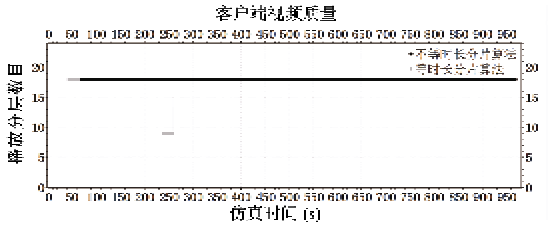


图 14　丢包率为 0.001 时 WLAN 用户的视频质量

综合以上实验和分析可知,不等时长分片算法在 HSDPA 和 WLAN 网络中都可以得到比等时长分片算法更好的性能。因此可以说,不等时长分片算法更加适合于应用在异构的网络中。

4　结　论

本文对基于 SVC 的 P2P 流媒体分片算法进行了研究。首先,对 SVC 的数据特性进行了深入的分析,结果表明 SVC 各层数据量相差很大,且重要性较高的低层常常具有更高的数据量。并且分析表明等时长分片算法没有考虑分片大小与分层重要性之间的关系,当丢包率高时可能导致视频质量严重下降。针对以上问题,本文提出了一种不等时长分片算法。该算法对不同层采用不同的分片时长进行分片。每个分层的播放时长通过该层的数据量大小和重要性综合计算获得。采用这一算法后,重要性越高的分层分片大小越小。实验和仿真结果表明不等时长分片算法可以提高低层数据的传输成功率。并且与等时长分片算法相比,不等时长分片算法更加适合于应用在异构环境下。

参考文献

[1] Vetro A, Christopoulos C, Sun H. Video transcoding architectures and techniques: an overview. *IEEE Signal Processing Magazine*, 2003, 20(2): 18-29

[2] Bouras C, Kioumourtzis G, Gkamas A. Simulcast transmission for video applications: performance evaluation with an integrated simulation environment. In: Proceedings of the 2009 International Symposium on Performance Evaluation of Computer and Telecommunication Systems,

Istanbul, Turkey, 2009. 339-346

[3] Wang Y, Reibman A, Lin S. Multiple Description Coding for video deliver. *Proceedings of IEEE*, 2005, 93(1): 57-70

[4] Schwarz H, Marpe D, Wiegand T. Overview of the Scalable Video Coding extension of the H.264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 2007, 17(9): 1103-1120

[5] Huang Y, Fu T, Chiu D, et al. Challenges, design and analysis of a large-scale P2P-VoD system. In: Proceeding of the ACM SIGCOMM 2008 conference on Data communication, NY, USA, 2008. 375-388

[6] Eberhard M, 21627 P2P-Next. http://www.p2p-next.org

[7] Sentinelli A, Anselmo T, Fragneto P, et al. Packetizing scalable streams in heterogeneous peer-to-peer networks. In: Proceedings of the 2011 IEEE International Conference on Multimedia and Expo, Barcelona, Spain, 2011. 1-6

[8] Sheu T, Wang Y. Dynamic layer adjustments for SVC segments in P2P streaming networks. In: Proceedings of the 2010 International Computer Symposium, Tainan, China, 2010. 793-798

[9] Lee T, Liu P, Shyu W, et al. Live video streaming using P2P and SVC. In: Proceedings of the 11th IFIP/IEEE international conference on Management of Multimedia and Mobile Networks and Services: Management of Converged Multimedia Networks and Services, Samos Island, Greece, 2008. 104-113

[10] Abboud O, Zinner T, Pussep K, et al. On the impact of quality adaptation in SVC-based P2P Video-on-Demand systems. In: Proceedings of the 2nd annual ACM conference on Multimedia systems, Santaclara, USA, 2011. 223-232

[11] Liu Z, Shen Y, Ross K, et al. LayerP2P: using layered video chunks in P2P live streaming. *IEEE Transactions on Multimedia*, 2009, 11(7): 1340-1352

[12] JSVM software manual, JSVM 9.15, September 24th 2008

[13] Baumgart I, Heep B, Krause S. OverSim: A flexible overlay network simulation framework. In: Proceedings of the 10th IEEE Global Internet Symposium in conjunction with IEEE INFOCOM 2007, AK, USA, 2007. 79-84

[14] Asioli S, Ramzan N, Izquierdo E. Efficient scalable video streaming over P2P network. In: Proceedings of the User Centric Media, Lecture Notes of the Institute for Computer Science, Social Information and Telecommunications Engineering, Venice, Italy, 2010. 153-160

[15] Sedoyeka E, Hunaiti Z, Almasri S, et al. Evaluation of HSDPA (3.5G) mobile link quality. In: Proceedings of the IEEE International symposium on Industrial Electronics, Cambridge, UK, 2008. 1446-1451

A P2P streaming segmentation algorithm considering data characteristics of SVC

Song Junping*, Zhou Xu*, Zhang Yan*, Tang Hui*, Bai Fan*, Zhao Zhifeng**

(*High Performance Network Laboratory, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190)

(**Huashu Media Network Co., Ltd, Hangzhou 310012)

Abstract

The equal playback length segmentation algorithm, widely used in P2P streaming systems based on scalable video coding (SVC), was studied to find an approach for P2P streaming systems to transmit and share video data in a helerogeneous environment. The results show that this algorithm does not consider the data characteristics of SVC adequately, and it may decrease video qualities on clients seriously when packet loss rate is high. To deal with this problem, the study proposed an unequal playback length segmentation algorithm. It adjusts the transmission success possibility of layers by changing the playback length of chunks, and guarantees the successful transmission of layers with high importance levels firstly. The proposed algorithm was evaluated by experiments and simulations and the results show that, compared with the equal playback length segmentation algorithm, the new proposed algorithm will smooth the transmission time of chunks, and increase the transmission success possibility of layers with high importance levels. Moreover, it performances better in different network environments and it is suitable to heterogeneous networks.

Key words: segmentation algorithm, P2P streaming, scalable video coding (SVC), heterogeneous