中图法分类号: TP301.6 **文献标志码**: A **文章编号**: 1006-8961(2011)04-0510-06
论文索引信息:左冬红. 面向 TCP 的流媒体传输编码码率自适应算法[J]. 中国图象图形学报,2011,16(4):510-515

面向 TCP 的流媒体传输编码码率自适应算法

左冬红
(华中科技大学电信系, 武汉 430074)

摘 要: 由于网络的时变性和异构性,以及在拥塞情况下的高丢包率,利用 TCP 传输流媒体数据是 Internet 流媒体分发系统提高流媒体分发质量的首选方案。由于 TCP 具有超时或错误重传机制,在网络拥塞情况下,难以保证高码率流媒体数据传输的实时性,因此提出一种面向 TCP 流媒体传输的编码码率自适应算法(TCP_RA)。该算法根据流媒体发送应用层缓冲区读写指针差值调整流媒体发送端的编码码率适应网络带宽的变化。仿真实验对比分析了该算法与基于 UDP 之上 TFRC 协议的流媒体传输码率自适应算法在流媒体传输质量上的差别。结果表明,该算法在网络环境较差的情况下有效地提高了流媒体传输质量。并且该算法容易实现,值得推广。
关键词: 流媒体传输; 自适应码率; TCP 协议

An algorithm for encoding rate adaptive streaming media transmitting based on TCP

Zuo Donghong
(*Department of Electronic and Information Engineering, Huazhong University of Science and Technology, Wuhan* 430074 *China*)

Abstract: Because of the network's time-varying and heterogeneity, and also the high probability of packet loss in congested network, transmitting media streaming over internet by TCP is a preferred scheme, which can improve the media streaming QoS. But due to the TCP retransmission for error or time out packets, in the congested network, it is difficult to provide real time transmission for high bit rate media streaming. This paper proposes an encoding rate adaptive media streaming transmitting algorithm (TCP_RA) based on TCP. It adjusts the media streaming encoding rate according to the difference between reading and writing points in the application layer buffer, in order to suit the network bandwidth' variety. It compares the difference for rate adaptive media streaming QoS improvement between TFRC and TCP_RA by simulation in NS2, simulation results show that TCP_RA can effectively improve the media streaming QoS in bad network environment. The algorithm is easy to implement, so it's worth of popularizing.
Keywords: streaming media transmission; rate adaptive; TCP protocol

0 引 言

流媒体是指采用流式传输的多媒体格式。其视频影像采用边下载边播放的形式,流媒体技术在互联网上有着广泛的应用,包括视频点播、视频直播、视频会议、远程教育和数字图书馆等。在现有的时变网络以及异构网络情况下,带宽波动、延迟抖动和丢包现象使得流媒体传输服务质量无法得到保证。当传输速率高于网络带宽时,会发生网络拥塞造成突发的丢包和延时过大,但如果媒体流传输速率低于网络可用带宽又无法有效地利用网络资源。因

收稿日期:2009-09-11;**修回日期**:2009-11-10
基金项目:国家自然科学基金项目(60773193); 湖北省武汉市青年科技晨光计划项目(200850731351)。
第一作者简介:左冬红(1975—),女,讲师。2007 年华中科技大学获通信与信息系统博士,主要从事流媒体分发以及无线网络研究。E-mail:sixizuo@mail.hust.edu.cn。

此,必须研究针对网络状况的流媒体传输自适应码率技术。

目前大部分流媒体分发系统采用 UDP 协议进行流媒体传输,但是由于 UDP 没有拥塞控制机制,不具有 TCP 传输友好性,而且也不保证流媒体数据的可靠传输,往往在网络不稳定情况下,进一步恶化网络状况,无法保证流媒体传输质量。因此,业界研究了很多针对流媒体传输而设计的改进方案,如 TFRC[1],DCCP[2] 以及 DSS(Darwin stream server)中使用的 Reliable UDP[3](可靠用户数据包协议)能够较好地适应网络带宽的变化。这些方案共同的缺点是需要在网络环境本已拥塞的情况下增加额外的协议流量,而且在网络拥塞时并不保证流媒体数据传输的可靠性,算法实现比较复杂。

由于 TCP 协议具有拥塞控制机制,而且能够保证数据的可靠传输,因此在网络环境比较差的情况下利用 TCP 传输流媒体数据是一种比较好的选择。根据 TCP 协议特点,提出一种面向 TCP 的流媒体传输自适应码率算法。该算法根据流媒体发送应用层缓冲区读写指针差值调整流媒体发送端的编码码率适应网络带宽的变化,只需要在发送端进行码率控制,无须增加网络额外流量;能够很好地适应网络带宽的变化,实现流媒体数据的可靠传输,并且算法实现简单。

1　基于 TCP 的流媒体分发系统架构

基于 TCP 的流媒体分发系统,由于 TCP 传输具有错误重传功能,因此要求在流媒体分发系统的发送和接收端都建立缓冲区。发送缓冲区用于保存由于网络拥塞没有及时发送的数据,而接收缓冲区用于缓存部分流媒体数据以防止网络状况波动带来的播放抖动,保持流媒体数据的播放顺畅。其基本架构如图 1 所示。

接收缓冲区大,能够较好地消除网络波动带来的播放抖动,但是将导致播放启动时延较大;接收缓冲区小,播放启动时延较小,但是在网络环境较差的情况下,将导致播放抖动较大。因此必须根据一定的应用要求,设置有效的接收缓冲区大小。

发送缓冲区通常设计成环形缓冲区,用来缓存网络拥塞没能及时发出的流媒体数据,缓冲区越大,能够缓存的数据越多。当流媒体数据采集码率大于网络传送速度时,没有能够及时发出的数据都将缓

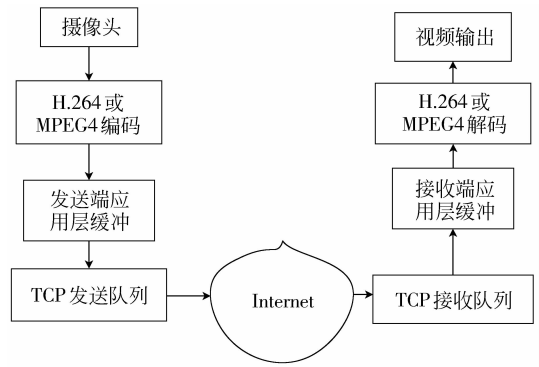

图 1　基于 TCP 的流媒体分发系统架构

Fig. 1　Streaming media delivery system based on TCP

存在此缓冲区里。如果缓冲区大小一定,网络速度一直比较慢,缓冲的流媒体数据将越来越多,最终导致回环,即新采集的数据将替换掉前面采集的但还没有发出的流媒体数据,这就导致流媒体数据在应用层丢包。这样接收端由于接收到的数据不连贯,导致接收到的部分数据无法解码,从而使得流媒体播放质量下降。

因此,在基于 TCP 的流媒体分发系统中,即使 TCP 保证了流媒体传输的可靠性,同样需要采取有效的措施防止应用层数据丢包。防止应用层发送缓冲区由于回环导致丢包的有效措施就是根据网络状况自动调整流媒体采集码率。只有当流媒体采集码率低于或等于网络端到端传送码率时,应用层才不会发生丢包的问题。因此,如何及时了解网络端到端传送码率,这就是需要解决的问题。通常的做法是在端到端之间发送一些探测包[4-5],通过计算探测包的传输时延和丢包率来估计网络带宽,这种做法是在本来已经拥塞的网络上再增加了一些额外的探测数据流量。另外一些做法是通过修改 TCP 协议栈[6],在 TCP 协议栈里,利用有效传输数据组成一些探测包,同样通过计算探测包的传输时延和丢包率来估计网络带宽,这些算法实现比较复杂,而且因为需要增加一级缓冲,这样将增加拥塞时段内包的传输时延。这些策略在实时流媒体分发系统中都不可取。

如前文所述,由于 TCP 的传输特点,当网络传输码率低于流媒体编码码率时,应用层发送数据缓冲区将发生累积现象。因此,本文将直接利用发送缓冲区缓存数据的累积程度来估测网络带宽的变化,从而进行流媒体编码码率自适应调整。

2 码率自适应调整策略

流媒体发送数据缓冲区结构如图 2 所示,数据输入为 H.264 或 MPEG4 编码器,数据写入位置用写指针 w 表示。此缓冲区数据输出由 TCP 发送端读取,该数据读出位置用读指针 r 表示。缓冲区每个区间的大小为一固定值 M_b,可根据 MTU 值进行调整,通常取 1 024,以便 TCP 一次发送一个完整的数据包。对缓冲区数据读取操作的单位为一个单位缓冲区间,即写指针根据编码器每单位时间产生数据的数量可能填满一到多个缓冲区间;而读指针一次将一个单位缓冲区间作为一个数据包发送到 TCP 发送队列,如果写指针没有写满一个单位缓冲区间,则读指针将等待写指针写满后才进行读取。

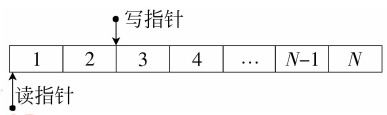

图 2 应用层缓冲区结构

Fig. 2 Application buffer structure

编码器编码码率为 R_W,最高编码码率为 $R_{H\text{-}thresh}$,最低编码码率为 $R_{L\text{-}thresh}$。TCP 端到端的发送码率为 R_T。写指针的移动速度受编码码率影响,而读指针的移动速度受 TCP 发送码率影响。码率自适应调整将根据当前的 R_T 来调整 R_W,使得 $R_W \leqslant R_T$,且尽可能最大化 R_W,从而有效利用当前网络带宽,减少应用层流媒体数据的丢包,提高流媒体传输质量。

下面首先定义自适应码率调整的几种状态。

稳定状态:读写指针之间位置差值保持为 0,若编码码率没有达到上限,则一旦提高编码码率将使得写指针移动到读指针前面或者编码码率已经达到上限。也就是说如果 $R_W < R_{H\text{-}thresh}$,则 $R_W = R_T < R_{H\text{-}thresh}$;否则 $R_W = R_{H\text{-}thresh} < R_T$。

编码码率降低调整状态:读写指针之间的位置差值 $(w-r)>0$,且不断发生改变,此时 $R_W > R_T$。

编码码率升高调整状态:读写指针之间的位置差值为 0,增大编码码率读写指针差值仍然保持为 0。此时 $R_W < R_T$,且 $R_W < R_{H\text{-}thresh}$。

以上状态并不是一直维持不变,而是相互转换的。各状态之间的转换关系如图 3 所示。

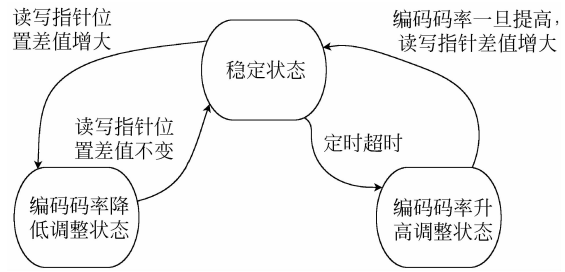

图 3 状态转换图

Fig. 3 State flowchart

下面将根据以上两种调整状态来分别讨论码率自适应调整策略。

1)流媒体编码码率大于网络传送码率,此时进入编码码率降低调整状态。

假定流媒体开始传输时读写指针在缓冲区的位置差值为 $d_0>0$,一段时间 t 之后读写指针的位置差值可以采用下式表示。

$$w - r = (R_W - R_T) \cdot t/M_b + d_0 \tag{1}$$

如果 $R_W = R_T$,则读写指针的位置差维持在 d_0 不变。由式(1)可知,读写指针单位时间内位置差值增长速度反映了流媒体编码码率与发送码率之间的差值。若 d_i 表示第 i 个单位时间内读写指针差值的增长量,则可由下式表示

$$d_i = w - r - \sum_{n=0}^{i-1} d_n \tag{2}$$

另外

$$d_i = (R_{W_i} - R_{T_i}) \cdot t/M_b \tag{3}$$

由式(2)(3)可知,网络发送码率 R_{T_i} 可由式(4)表示

$$R_{T_i} = R_{W_i} - \left(w - r - \sum_{n=0}^{i-1} d_n\right) \cdot M_b/t \tag{4}$$

如果 $R_{L\text{-}thresh} \leqslant R_{T_i} \leqslant R_{H\text{-}thresh}$,则在第 i 个单位时间后,流媒体的编码码率应该调整为 R_{T_i}。因此,在第 i 个单位时间,流媒体编码码率的调整量降低为 $R_d = \left(w - r - \sum_{n=0}^{i-1} d_n\right) \cdot M_b/t$。如果 R_{T_i} 不在此范围,则需要区分为两种不同的情况,如果 $R_{T_i} > R_{H\text{-}thresh}$,说明网络带宽足够传输高品质的流媒体数据,则直接设定流媒体编码码率为其上限值;如果 $R_{L\text{-}thresh} > R_{T_i}$,说明网络带宽不够传输最低品质的流媒体数据,则设定流媒体编码码率为其下限值,并且进一步应用主动丢包的策略丢弃一些非关键帧,建议仅传输可独立解码的 I 帧数据。

2)流媒体编码码率小于网络传送码率且没有

达到编码码率上限,此时进入编码码率升高调整状态。

当网络速度大于或等于流媒体编码码率时,由于读指针不可能移动到写指针前面,此时读写指针差值 d 为 0。由于无法根据读写指针的差值变化速度来反映流媒体编码码率与网络发送码率之间的差值,因此采用试探性的方法来提高流媒体编码码率,即每次提高一固定值的编码码率。如果提高编码码率之后读写指针差继续维持为 0,则可以进一步提高编码码率,如此循环;否则退回到前一次设定的流媒体编码码率,进入稳定状态。

由编码码率升高调整状态进入稳定状态的条件为:提高编码码率之后,读写指针的差值 $d>0$,但恢复编码码率为前一次设置的编码码率时读写指针差值为 0。而由稳定状态进入调整状态的条件有两类:(1)若维持了稳定状态一段时间,则在此时间之后即进入编码码率升高的调整状态;(2)在稳定状态下,出现了读写指针差值 $d \gg 0$ 的情况,进入编码码率降低的调整状态。

3　码率自适应算法实现

码率自适应算法中 3 种状态的流程图分别如图 4—6 所示。

在算法的具体实现过程中还需要注意以下事项:

1) TCP 数据传输有两种调用方式,一种是非阻塞式,一种是阻塞式。非阻塞式是当应用层向 TCP 发送队列写入数据时,发送调用将直接返回,并告诉

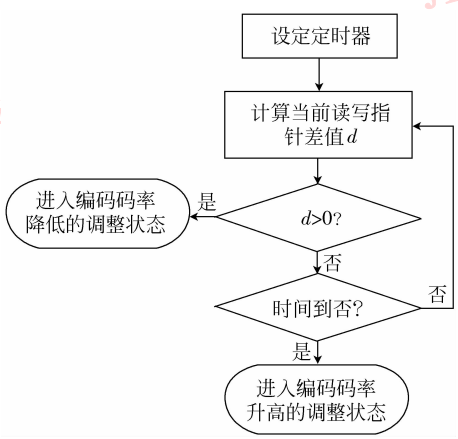

图 4　稳定状态流程图

Fig. 4　Steady state flowchart

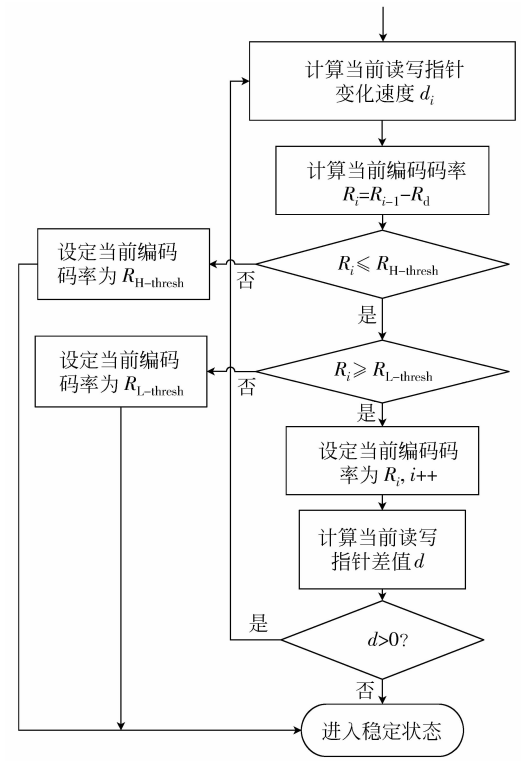

图 5　编码码率降低调整状态流程图

Fig. 5　Encoding rate decreasing state flowchart

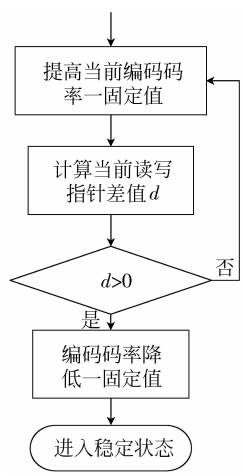

图 6　编码码率升高调整状态流程图

Fig. 6　Encoding rate increasing state flowchart

应用层成功发送了多少数据。不能保证一次发送一个完整的数据包,也就是说应用层不能每次完整地发送一个单位缓冲区间,为保证数据的顺序发送,应用程序需要进行复杂的处理。而阻塞式则是当应用层向 TCP 发送队列写入数据时,发送调用将等待成功地将数据完全写入 TCP 发送队列后才会返回,否则一直阻塞,这样数据没有发送成功,读指针的位置

就不会发生变化,易于实现。因此本文建议采用阻塞式发送调用。

2) TCP 传输具有慢启动和拥塞退避算法,当网络出现拥塞时,TCP 传输速度将会有一个较小的波动,因此在实现自适应码率调整时,必须忽略该波动带来的影响。具体实现方法为,当非常短的时间内出现了读写指针差值较小的变化时,无须进行码率调整。

3) 由于需要定时测量读写指针差值的变化,因此在应用程序中在以上 3 种状态下都必须启动一个定时器,定时计算读写指针差值的变化。

4 实验结果及分析

由于 TFRC 是目前流媒体应用系统中比较常用的一种能够较好实现自适应码率调整的基于 UDP 传输的算法,因此,本文将针对 TFRC 算法与本算法 TCP_RA 在流媒体分发系统中自适应码率调整性能方面进行实验对比分析。

本实验采用 NS2 仿真工具进行实验,流媒体传输和质量评估仿真架构采用文献[7]所描述的架构,该架构由台湾金门技术学院柯志亨移植到 NS2 仿真环境下[8],并且由挪威科技大学 Arne Lie 等人[9]进一步修改支持自适应码率算法。因此本文基于此仿真环境,增加基于 TCP 的码率自适应仿真算法。仿真后期数据的处理完全借助于 Arne Lie 提供的自适应算法仿真后期处理工具[10]。

仿真场景设置如下:在整个仿真环境中只有一条主干路径,主干路径由路由器 1 和路由器 2 连接构成,路由器之间的链路带宽为 16 Mbit/s,时延为 10 ms。在每个路由器上分别接入流媒体服务源节点和目的节点,接入链路带宽为 10 Mbit/s,接入时延为 5 ms。视频源采用 foreman_cif. yuv,此视频源包含 300 帧 CIF 格式的 YUV 视频数据。本文采用 ffmpeg 开源工具[11]将此视频源编码成 MPEG4 媒体流。自适应码率调整通过设置不同的量化参数从而调整码率,量化参数的取值范围为 2—31,仿真时间为 40 s。仿真实验主要分析了在分别采用 TFRC 算法和 TCP_RA 算法的情况下 40 个媒体流竞争主干链路,主干链路的丢包率以及利用率;在同样的网络拥塞、接收端缓冲区大小不同情况下某个媒体流的分发质量,以及接收端缓冲区固定为 250 ms、不同网络拥塞情况下某个媒体流的分发质量。这里流媒体的分发质量主要指在特定接收缓冲区大小限制条件下接收到的有效视频帧数以及这些有效视频帧的平均峰值信噪比。

表 1 列出了 TFRC 和 TCP_RA 算法在只有 40 个媒体流竞争主干网带宽资源情况下主干路径丢包率以及利用率等仿真结果。由于 TCP_RA 算法基于 TCP 传输,因此主干链路丢包率和链路有效利用率都比 TFRC 表现好。

表 1 主干路径仿真结果

Tab. 1 Simulation result on backbone

算法	丢弃字节/KB	发送字节/MB	丢包率/%	利用率/%
TFRC	650	74	8.6	93
TCP_RA	504	76	6.5	95

表 2 列出了两种算法在接收端缓冲区大小不同,40 个媒体流竞争主干路径带宽的情况下,某个媒体流接收端能够正确接收到的流媒体视频帧数以及平均峰值信噪比。从表 2 可以看出,当接收缓冲区较大(数秒以上)时,TFRC 算法接收端接收到的有效视频帧较多,但视频帧平均峰值信噪比较低;而当接收缓冲区较小时,TCP_RA 算法接收端接收到的有效视频帧较多,且平均峰值信噪比也较高。这是由于接收缓冲区越大,通过网络无序到达的流媒体数据包可以进行有效的排序,而不至于因为超过播放有效期限被丢弃,因此接收缓冲区越大,接收到的有效流媒体数据越多。由于 TCP 传输具有错误重传机制,因此当发送的数据包出现错误时,会重传发生错误的数据包,从而使得整体发送的不同的数据包的数量比 UDP 少,这也就造成了接收到的不同数据包比 UDP 少。但是如果接收缓冲区不够大,即使 UDP 接收到的不同数据包比 TCP 多,但是很多先发后到的数据包都将因超时被丢弃,从而造成当接收缓冲区变小时,TFRC 算法接收到的有效帧数急剧降低。而 TCP 传输能够基本保证网络数据的有序到达,因此当接收缓冲区变小时,TCP_RA 算法比 TFRC 算法接收到的有效视频帧要多。再加上 TFRC 采用 UDP 进行传送,对于错误的数据包不会进行重传,因此数据包发生比特错误的概率比 TCP_RA 大,这样就导致整体接收到的视频帧的峰值信噪比比 TCP_RA 低。而接收端缓冲区的大小,反映了流媒体播放启动时延的大小。由此可见,TCP_RA 算法能更好地适应于播放启动时延要求较少的流媒体分发场景。

表 2　不同接收端缓冲区大小仿真结果

Tab. 2　Simulation results based on different buffer size of receiving node

算法	缓冲区大小	接收到的帧数	平均峰值信噪比
TFRC	∞	295	28.02
	5 s	295	28.02
	500 ms	111	35.64
	250 ms	106	35.52
	100 ms	85	23.14
TCP_RA	∞	233	30.16
	5 s	161	36.67
	500 ms	142	36.92
	250 ms	134	36.79
	100 ms	117	24.56

表 3 列出了接收端缓冲区大小固定为 250 ms 时两种算法在不同数量的媒体流同时竞争主干路径网络带宽资源时,某个媒体流接收端能够正确接收到的流媒体视频帧数以及平均峰值信噪比。由表 3 可知,同时竞争的媒体流越多,基于 TCP_RA 算法的媒体流接收端越能接收到更多的有效视频帧,且平均峰值信噪比较高,因此,TCP_RA 算法更适应于具有较高的网络拥塞丢包率,且播放时延要求较少的流媒体分发应用场景。

表 3　不同数量的媒体流仿真结果

Tab. 3　Simulation results based on different numbers of media streaming

算法	媒体流数目	接收到的帧数	平均峰值信噪比
TFRC	10	299	36.82
	20	168	35.70
	40	106	35.52
	80	171	27.65
TCP_RA	10	270	37.17
	20	168	35.94
	40	134	36.79
	80	181	28.62

5　结　论

由以上对比仿真实验可知,在网络环境较差的情况下,基于 TCP 的流媒体自适应传输算法能够更好地适应具有较高丢包率的网络环境,而且算法实现简单,不需要添加复杂的应用层协议,也不需增加额外的网络反馈流量。因此,该算法可以推广到具有高丢包率的无线网络应用环境。

参考文献(References)

[1] Handley M, Floyd S, Padhye J, et al. The Internet Engineering Task Force. RFC3448 TCP Friendly Rate Control (TFRC): Protocol specification [S]. New York: IETF, 2003.

[2] Lai Yuancheng. DCCP congestion control with virtual recovery to achieve TCP-fairness [J]. IEEE Communications Letters, 2008, 12(1):50-52.

[3] Bova T, Krivoruchka T. RELIABLE UDP PROTOCOL [EB/OL]. (1999-02-01) [2010-12-14]. http://tools.ietf.org/html/draft-ietf-sigtran-reliable-udp-00.

[4] Jain M, Dovrolis C. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput [C]//Proceedings of the ACM SIGCOMM 2002 Conference-Applications, Technologies, Architectures, and Protocols for Computer Communications. Pittsburgh, PA, United States: ACM, 2002, 32(4):295-308.

[5] Hu N, Steenkiste P. Evaluation and characterization of available bandwidth probing techniques [J]. IEEE Journal on Selected Areas in Communications, 2003,21:879-894.

[6] Cao M L T, Hasegawa G, Murata M. Available bandwidth measurement via TCP connection [C]//John V, David H. Proceedings of IFIP/IEEE MMNS. Berlin, German: Springer, 2004:38-44.

[7] Klaue1 J, Rathke1 B, Wolisz A. EvalVid: A Framework for Video Transmission and Quality Evaluation [M]. Berlin, German: Springer, 2003:255-272.

[8] Ke C H, Shieh C K, Hwang W S, et al. An evaluation framework for more realistic simulations of MPEG video transmission [J]. Journal of Information Science and Engineering, 2008, 24:425-440.

[9] Lie Arne, Klaue Jirka. Evalvid-RA: Trace driven simulation of rate adaptive MPEG-4 VBR video [J]. Multimedia Systems, 2008, 14(1):33-50.

[10] Lie Arne. Evalvid-RA[CP/OL]. (2009-7-30) [2009-8-15]. http://www.item.ntnu.no/~arnelie/Evalvid-RA.htm.

[11] Michael Niedermayer. Ffmpeg[CP/OL]. (2010-10-18) [2010-12-14]. http://www.ffmpeg.org/index.html.