网络编码与多路径传输在互联网视频直播中的应用研究

张锦辉[1] 邓 茜[2] 李振宇[2]

(华为技术有限公司南京研究所 南京 210012)[1] (中国科学院计算技术研究所 北京 100190)[2]

摘 要 互联网视频直播的普通用户可以实时上传视频,其他用户可以实时在线观看。这种模式使其无法根据流行度提前将视频推送到各CDN节点缓存,而且要求视频从产生到分发到用户观看的延迟尽量低。视频上传的性能对互联网视频直播观看体验的影响最大,上传流的停顿常常造成下载流中无数据可发,在停顿现象中的占比达87.6%。针对这一问题,文中提出了一种将网络编码与多路径传输结合后应用到互联网视频直播上传中的方法。首先,通过网络编码对实时产生的视频数据进行冗余编码,增强抗丢包性;其次,对上传终端到CDN的多条链路测速,利用效用函数评估每条链路的性能;最后,根据不同链路的效用值将编码后的数据分配到各链路上进行传输,到达接收端后还原。理论分析和实验表明,与目前普遍采用的TCP传输方式相比,在2%丢包率、50 ms时延的网络条件下,所提方法的传输速率是TCP的7.6倍。实验结果表明,将网络编码与多路径传输结合后应用到互联网视频直播上传中,可以显著提升上传速率,快速感知网络的变化情况,增强对多变的网络环境的适应性。

关键词 互联网,视频直播,流媒体,网络编码,多路径传输

中图法分类号 TP393.02 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.08.028

Study on Application of Network Coding and Multipath Transmission in Internet Live Video Broadcasting

ZHANG Jin-hui[1] DENG Qian[2] LI Zhen-yu[2]

(Nanjing Research Institute. Huawei Technologies Co., Ltd., Nanjing 210012, China)[1]

(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract Internet live video users can upload videos in real time, and other users can watch online in real time. This mode makes it impossible to push video to CDN node in advance based on popularity, and requires that the delay from video generates to viewers be as low as possible. The performance of video uploading has the greatest impact on the viewing experience of Internet live video. The pause of the upload stream often causes no data to be sent in the download stream, accounting for 87.6% of the video freeze. In response to the above questions, the paper proposed a method of combining network coding and multi-path transmission into Internet live video uploading. Firstly, the real-time generated video data are redundantly coded through network coding to enhance anti-dropping packets. Secondly, the speed of each link of the uploading terminal to the CDN is measured, and the performance of each link is evaluated by the utility function. Finally, the encoded data are allocated to the links for transmission according to the utility values of different links. Theoretical analysis and experiments show that the transmission rate is 7.6 times that of TCP under the network condition of 2% packet loss rate and 50ms delay compared with the TCP transmission method currently widely used. The results show that the combination of network coding and multi-path transmission into Internet live video uploading can significantly improve the upload rate, quickly sense the changes of the network, and enhance the adaptability to the changing network environment.

Keywords Internet, Live video, Streaming, Network coding, Multipath transmission

1 引言

随着具有视频上传功能的可携带智能设备的普及和网络带宽的增加,互联网视频直播应用呈井喷式发展,据中国互联网络信息中心统计[1],截至2017年12月,网络直播用户规模达到了4.22亿,较2016年增长22.6%,游戏直播、真人秀直播用户规模的增速更是分别达到了53.1%和51.9%。互联网视频直播已经成为网络社交、自媒体和广告营销的新方式。

互联网视频直播的普通用户可以实时上传视频,其他用户可以实时在线观看。这种模式使其无法像点播那样提前根据流行度将视频推送到各内容分发网络(Content Delivery Network,CDN)节点进行缓存[2-3],而且要求视频从产生到分

到稿日期:2018-10-07 返修日期:2019-03-11

张锦辉(1978—),男,硕士,工程师,主要研究方向为视频承载网络、IP制播网,E-mail:watw.zhang@live.hk(通信作者);邓 茜(1992—),女,硕士生,主要研究方向为传输协议;李振宇(1980—),男,博士,研究员,CCF会员,主要研究方向为计算机网络。

发到用户观看的延迟尽量低[4-6]，并且互联网视频直播不同于传统的广播电视采用无线广播或专网广播，也不同于交互式网络电视(Internet Protocol Television，IPTV)在封闭的 IP 网络中采用组播协议直播。互联网直播的运营者通过互联网分发直播视频，视频的实时上传和实时下载都必须使用单播传输，并且很可能需要穿越广域网，这使得互联网视频直播对网络传输提出了更高的要求。文献[7]中的研究发现：视频上传的性能对互联网视频直播观看体验的影响最大，上传流的停顿常会造成下载流中无数据可发，此原因造成的停顿占比达 87.6%。

当前互联网视频直播系统[8-11]通常先由普通用户与 CDN 服务器建立 TCP 连接，然后将视频上传到 CDN，视频请求者观看直播视频时也直接从 CDN 获取。这个过程中 CDN 会根据当前用户所在地及 CDN 的负载情况，返回一个最优 CDN 服务器 IP。从互联网视频直播系统端到端来看，当前互联网视频直播上传的瓶颈主要体现在：1)单一链路容易受瓶颈链路的带宽限制，当前带宽通常达不到直播传输的带宽要求；2)CDN 服务器端存在性能瓶颈[12]，其分布范围对传输性能和延迟的影响较大。

从当前技术来看，已有一些针对网络上传进行优化的方案，主要可以分为 3 类：

(1)针对单条路径 TCP 传输优化。如 Large Initial CWND[13]将初始窗口由 3MSS 增大到 10MSS；BBR[14]利用丢包判断拥塞状况，通过调整窗口控制发送速率。这些方法被应用在互联网视频直播时，并没有摆脱 TCP 本身在支持互联网视频直播传输时产生停顿的问题。

(2)MPTCP[15]，其借助源的多宿主特性，在源和目的节点之间建立多条 TCP 子连接，从而利用源和目的的多条路径传输数据。但是，该方案不能解决 CDN 服务器端的性能瓶颈，此外 TCP 固有的超时重传带来的传输停顿等问题依然存在。

(3)针对 UDP 的优化，如 QUIC[16]在 UDP 的基础上增加了基于 XOR 的包级别的前向纠错(Forward Error Correction，FEC)，同时增加了重传机制。但是，此方法仍然使用单条路径传输，并未利用多条路径来提升传输性能。

针对这些问题，本文提出了一种网络编码与多路径传输相结合的策略。首先，通过网络编码对实时产生的视频数据进行冗余编码，增强抗丢包性；然后，对上传终端到 CDN 的多条链路测速，利用效用函数评估每条链路的性能；最后，将编码后的数据根据不同链路的效用值分配到各链路上传输。多路径传输将提高总的有效带宽，而网络编码的加入消除了编码段内部数据的顺序相关性，从而实现了多条路径高效协同地传输数据。

2 基于多路径网络编码的互联网视频直播系统

2.1 总体思路

在互联网视频直播系统中由于存在多个 CDN 服务器，从源端到各个 CDN 服务器的路径一定不可能完全重合，会在网络某节点处转向不同的下一跳。因此，实际上存在多个 CDN 服务器和多条与之相连的路径。

本文利用当前存在多条路径的网络环境，使上传者利用多条路径向多个 CDN 服务器同时上传数据，利用多条链路的带宽有效地解决网络瓶颈。但是，与此同时，多条路径的传输策略又带来了很多新的挑战，主要包括以下几点：

(1)路径选择问题。当 CDN 服务器数量庞大时需要从中选取一定数量的最优路径。

(2)不同路径协同问题。由于不同路径的速率、时延等性能指标不同，因此需要一定的传输策略协调数据在多路径上传输，以避免单条路径传输数据量与速率不匹配，从而增大数据到达延迟。

(3)单路径网络性能变差。传输的过程中可能出现某条路径性能突然变差并导致丢包等问题，需要另外选择性能较优的路径进行丢包恢复等。

本文利用网络编码、单路径实时测速等技术实现多路径的协同传输，以提高网络带宽利用率。

2.2 背景：网络编码

为了解决多路径带来的难题，本文引入网络编码技术。网络编码的核心思想是在网络中的各个节点上对各条信道上收到的信息进行线性[17]或者非线性[18]的处理，然后将其转发给下游节点，中间节点扮演着编码器或信号处理器的角色。

网络线性编码的原理为(见图 1)：对 s 个数据块进行编码，给每个块乘上一个秩为 t 的系数矩阵得到编码后的数据，利用线性代数原理，只要接收到任意的 $(s-t)$ 块原始数据和 t 块编码后的数据，在两部分数据线性独立的情况下，就可以解码得到所有原始数据。本文取 t 为 s，将所有的数据转化为编码后的数据块进行转发，接收方只需要接收到 s 个线·性无关数据块就可以解码得到原始数据。

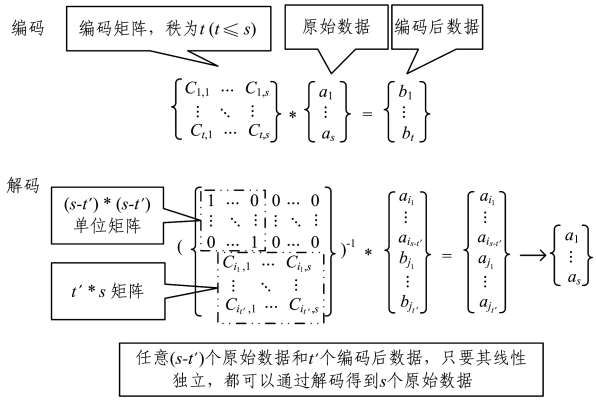

图 1 网络编码原理

Fig. 1 Principle of network coding

本文对网络中间节点将接收到的数据进行分段编码，将接收到的数据分割为大小相同的段(segment)，在 segment 的内部再继续将其分成 k 个块(block)，对这 k 个 block 进行编码之后再在多条路径上进行分发。

在同一个 segment 内部，各个编码块的顺序无关紧要，只需要将同一个 segment 的数据收集到一起，块数达到要求之后即可开始解码。因此，基于多路径传输时，网络编码的优势主要有以下几点：

(1)无需考虑多路径环境中发送和接收的顺序是否一致，

只需要将同一段的数据收集在一起，当数据块数量达到一定值时即可解码还原数据，简化了 MPTCP 等多路径传输中的同步问题。

(2)可以简化丢包恢复。由于其顺序无关性，如果发生了丢包，但已收到 s 个报文则不需要重传；若已收到的报文不足 s 个，则不需要重传某个特定的编码块，而只需要重传线性无关块即可进行解码。

2.3　系统总体结构

系统总体结构如图 2 所示。

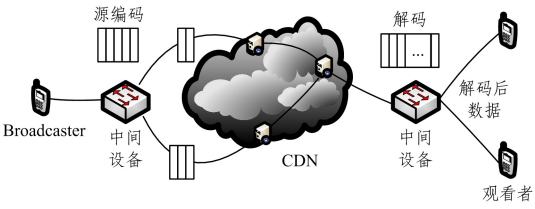

图 2　多路径传输

Fig. 2　Multipath transmission

多路径传输的流程如下：

(1)编码中间设备将视频数据进行网络编码；

(2)将编码后的数据块通过多条路径向多个 CDN 接收服务器传输；

(3)接收服务器把编码块推送给连接观看者的服务器，并继续向观看者转发；

(4)解码中间设备在收集数据块后对其解码并将其发送给观看者。

下文将介绍各个阶段的传输过程和其中的关键技术。

2.4　中间设备编码

上传端中间设备的主要工作是：将收到的原始视频数据缓存起来进行编码，然后在多条路径上进行分发。其处理流程如图 3 所示。

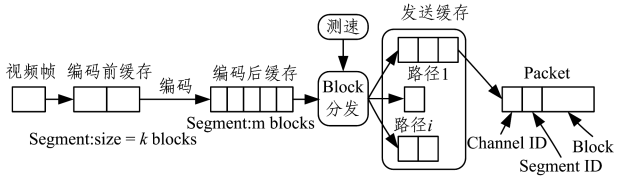

图 3　上传端中间设备的处理流程

Fig. 3　Processing flow of uploading intermediate device

(1)视频数据缓存和编码

中间设备监听并接收来自特定端口的数据包，并将收到的数据放到缓冲区，当收集到的数据达到 segment 大小时就进行编码。block 的大小需要根据视频的封装格式设定，传输完整的封装包。以 ts 封装格式为例，视频 ts 包的大小为 188 B，在以太网的 MTU1500 B 下，可以直接封装 7 个 ts 包，一共是 188 * 7＝1316 B，因此在 ts 视频封装环境下，本文将 block 大小设置为 1316(以下均采用 1316 作为块大小)，编码块会在原始数据的基础上加上一些编码信息，将 segment 设置为 block 大小的整数倍。

为了区别不同的 segment，本文在每个块的数据头部加上了“seg:”段号进行标识。由于不同的直播频道有不同的视频流，为了区分频道还可以在头部加上“channel ID”，用(channel ID，segment ID)二元组来标识某个频道的视频片断。

为了解决采用 UDP 协议传输可能产生的乱序问题，block 数据用 RTP 协议[19]封装后，再通过 UDP 协议传输。RTP 头中的 sequence number 标识发送者所发送的 RTP 报文的序列号，每发送一个报文序列号就加 1。接收端可以根据该 sequence number 将收到的报文排序还原。

(2)数据分发

由于与不同 CDN 服务器连接的链路状况不同，在传输时需要对不同链路进行测量和筛选，选择最优的几条路径进行传输，并根据链路速度差异将编码块分发到不同的链路上进行传输。上述流程可能受到网络状况变化的影响，为了适应网络变化，本文采用的关键技术主要包括：

1)防止编码等待时间过长。在收集编码数据时，可能遇到上传端数据迟迟不到达的情况。此时，已经收集到一部分数据，但是还不足以用于编码。为此，设置一个定时器，在等待时间超过阈值(T_1 ms)时还没有足够数据可用于编码的情况下，直接将需要的数据进行填充(0 填充)，并进行编码发送，以防止等待时间过长。

2)主动冗余。由于网络编码必须收集到 s(原始数据块数)个线性无关的数据块才能进行解码，编码块的丢失将导致整个数据段不能解码。针对这个问题，系统采用了主动冗余的方法，即发送 $m>k$ 个块(k 为一个 segment 内的块数)。m 的取值在每收到解码中间设备发送的反馈后更新一次。解码中间设备每成功解码或者放弃一个 segment，就向编码中间设备发送一次信息，以反映离成功解码所需要的块数差 d，如果成功，则 d 为 0。

$$m_{t+1}=\begin{cases}\min\{(1+\alpha)m_t,(m_{t+1})\}, & F<1\\ \max\{(m_t-1),k\}, & \text{连续 3 次 } F=1\end{cases} \tag{1}$$

其中，$F=1-d/k$；α 是 F 的滑动平均值，初始值为 0。

$$\alpha=(1-g)*\alpha+g*F$$

其中，参数 g 反映新数据的比重。当发生重传时，冗余比例的增长与 F 成正比关系，而当连续 3 次均成功($d=0$)时，则每次减小一个数据包的冗余量。初始 m 值为：

$$m_0=\min\{(k+1),(1+\rho)*k\}$$

其中，ρ 为网络丢包率经验值，目前的丢包率为 2%～3%，因此 ρ 可设为 0.03。m 的选择主要考虑减小丢包率带来的影响。比如在 k 值取 100 的情况下，m_0 取 103，理论上可以消除当前丢包率带来的影响。在丢包率较小的情况下，本文初始时至少发送一个冗余包，如果链路没有丢包，则 m 的值会自行往下调整。

3)路径选择和多条路径协同分配。对不同路径进行实时评分，为此定义路径 i 的效用函数为 $u(i)$。效用函数周期性(如间隔 1 min)更新。链路 i 的效用函数由其传输速率 thr_i 和丢包率 ρ_i 共同决定。

$$u_i=thr_i*sigmoid(\rho_i-\rho_0)$$

其中，$sigmoid(x)=1/(1+e^{\alpha*x})$。效用函数定义的出发点如下：1)平均速率反映了路径的可用带宽，丢包率反映的是路径拥塞程度，虽然二者有一定关联，但并不能用一个指标完全表

达另一个指标的信息。2)丢包率非常重要,因为丢包恢复时间往往较长,使用主动冗余时冗余比例也会很高,因此使用 sigmoid 函数来刻画其重要程度。具体来说,ρ_0 为平均丢包率,可以根据经验来初始化,比如取 $\rho_0=0.03$。使用 sigmoid 函数后,如果丢包率 $\rho_i>\rho_0$,则当 α 取较大值(如 $\alpha>200$)时,sigmoid 的取值会迅速下降,使得该路径的效用函数迅速下降。在 α 取不同值的情况下,当 $\rho_0=0.03$ 时,$sigmoid(\rho_i-\rho_0)$ 的取值如图 4 所示。

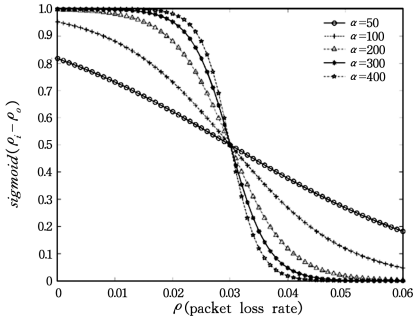

图 4 sigmoid 函数取值

Fig. 4 Value of sigmoid function

由图 4 可以看出,可以通过调节 α 的取值来调节 sigmoid 函数的取值,从而调整丢包率在效用函数中的重要程度。如果丢包率不可获得,则可以设置每条路径丢包率均为 0,此时相当于 sigmoid 函数并不起作用。

发送 segment 时,选取效用函数最高的若干条路径进行传输,并按效用函数大小来决定每条链路应该发送的块的个数。具体来说,一个 segment 包含 m 个块,第 i 条路径传输的块的个数(取整)为:

$$m*\left(\frac{u_i}{\sum_{i=1}^{h}u_i}\right)$$

其中,h 为从编码设备到 CDN 服务器群选定的路径总数。

4)单条路径网络状况突然恶化。由于编码块经由不同的路径进行传输,如果某一条路径突然发生了大量丢包,导致正在传输的编码块丢失,此时发送方重传任意与已传输编码块线性无关的编码块即可。在重传时,选择传输性能最优的路径进行重传。需要说明的是,由于已经有主动冗余,解决了非突发丢包带来的问题,因此只有网络状况突然恶化时,才需要重传编码块,这种情况发生的概率非常低。

CDN 服务器的内部传输与现有机制相同,本文不再赘述。

2.5 中间设备解码

观看端中间设备的处理流程主要包括:1)对编码数据按 segment ID 进行缓存;2)收集足够的 block 后对数据进行解码;3)将解码后的数据转发给观看者。

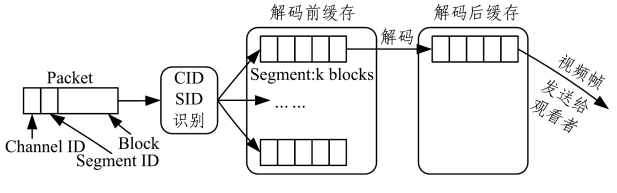

图 5 观看端中间设备的处理流程

Fig. 5 Processing flow of watch side middleware

由于多条链路延迟不同,因此发送到观看端中间设备的数据是乱序的。观看端中间设备可以根据 RTP 报文头中的 sequence number 依次将收到的报文排序,将其还原为原始顺序。本文在中间设备将 segment ID 解析出来后,将同一个 segment 的数据存放在一起。当收集的块数可以解码之后使用 Gauss-Jordan 消解方法解码获得原始 segment,并将其发送给观看者。为了解决由于 block 丢失或者延迟造成的收不齐 k 个线性无关 block 的问题,为每个 segment 设置一个定时器。若定时器失效,则放弃该 segment。

由于多条链路的延迟存在差异,解码设备可能需要同时存储多个未接收完全的 segment,其平均个数可以表示为:

$$Q=\frac{\Delta delay}{playtime\_segment}=\frac{\max\limits_{i=1,\cdots,h}(delay_i)-\min\limits_{i=1,\cdots,h}(delay_i)}{\frac{k*block\_size}{avgbitrate}}$$

其中,$\Delta delay$ 为延迟最大的路径和最小路径之间的延迟差值,这段时间到达的所有 *segment* 均需要同时缓存,分母为一个 *segment* 的播放时间,其用 *segment* 除以视频播放比特率得到。

因此,未解码数据的缓存总大小为:

$$C=Q*(k*block_{size})$$
$$=\left[\max_{i=1,\cdots,h}(delay_i)-\min_{i=1,\cdots,h}(delay_i)\right]*avgbigrate$$

同时,解码后的数据在转发之后并不立即释放。本文利用中间设备的缓存功能将部分数据缓存起来。将每个解码后的 segment 缓存 Ts(T 默认为 1),当有新的观看者时,可立即推送缓存的 segment,从而缩短首屏时延。

3 延迟分析

从上传端改变动作到观看者感受到这个变化中间存在一定的延迟,如图 6 所示。该延迟可以分为以下几段。

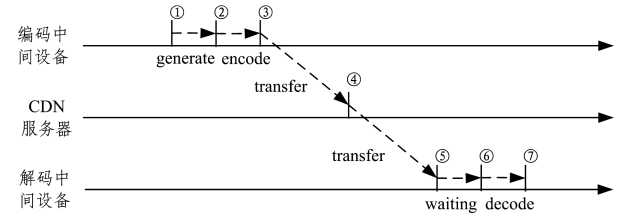

图 6 端到端延迟

Fig. 6 End-to-end delay

(1)数据产生延迟。摄像头捕获图像并产生视频数据所需要的时间(见图 6 中的① －②)表示为$\frac{S}{bit_rate}$。

(2)编码延迟。将每个 segment 进行编码所产生的延迟(见图 6 中的②－③)表示为 T_{coding}。

(3)数据传输延迟。从上传者发送到 CDN,并从 CDN 推送到边缘服务器的数据传输延迟(见图 6 中的③－⑤)表示为 $\frac{S}{\sum_{1}^{m}T_m}+\max_m(\frac{1}{2}*RTT*(1-loss\%)+RTO*loss\%)$。

(4)缓存等待延迟。收集齐同一个 segment 内足够的块数需要等待的时延(见图 6 中的⑤－⑥)表示为 Δ。

(5)解码延迟。将得到的编码块进行解码的延迟(见图 6

中的⑥－⑦）也表示为 T_{coding}，因为解码和编码所用的时间基本一致。

其中，s 为 segment 大小，bit_rate 为数据生成的比特率，k 为 segment 内的 block 数，T_m 表示 m 条链路的传输速率。

因此，从理论上来看，总的延迟为：

$$Delay=\frac{S}{bit\_rate}+T_{coding}+\frac{S}{\sum_{1}^{m}T_m}+\max_m\left(\frac{1}{2}*RTT*(1-loss\%)+RTO*loss\%\right)+T_{coding}+\Delta \quad (2)$$

式(2)的推导过程如下。

1）数据产生延迟 $\frac{S}{bit\_rate}$，其中 bit_rate 为视频播放的码率，s 为一个 segment 的大小。这个公式计算的是：对于一个 segment 大小的数据，将其转换为视频时能够进行播放的时长，也就是产生一个完整的 segment 的数据所需要的时间。

2）编解码延迟 T_{coding}，网络编码已有理论[20]证明，编解码时间与块数成平方增长，与段大小成正比增长，即 $T_{coding}=T_0*(\frac{k}{k_0})^2*\frac{B}{B_0}$，其中 T_0 为编解码由 k_0 个大小为 B_0 的块组成的 segment 所用的时间。实测发现，当 $k_0=10$，$B_0=1$ KB 时，$T_0=0.1$ ms。因此，$T_{coding}=0.1*(\frac{k}{10})^2*\frac{B}{1\text{KB}}$。

3）数据传输延迟：$\frac{S}{\sum_{1}^{m}T_m}+\max_m(\frac{1}{2}*RTT*(1-loss\%)+RTO*loss\%)$。该延迟主要包含两部分：①$\frac{S}{\sum_{1}^{m}T_m}$是数据在多条链路的传输时间，假设数据在每条链路上的发送大小已经分配完成，那么计算最大带宽链路的时间即为：$k*\frac{T_{\max}}{\sum_{1}^{m}T_m}*\frac{B_{size}}{T_{\max}}=\frac{S}{\sum_{1}^{m}T_m}$，$k$ 是一个 segment 内的块数，$\frac{T_{\max}}{\sum_{1}^{m}T_m}$是分配给最大带宽链路的块数，$B_{size}$是块大小，$T_{\max}$是最大带宽；②平均传播时间，每条链路的平均传播时间在没有丢包的情况下是 $\frac{1}{2}*RTT*(1-loss\%)$，假设每次丢包都会发生超时重传，则平均传播时间为 $(\frac{1}{2}*RTT*(1-loss\%)+RTO*loss\%)$，我们取所有链路中平均传播时间的最大值作为传播延迟。

4）缓存等待延迟：Δ。

根据当前网络中的一些经验值可以粗略估计出数据时延。

对式(2)中的相关变量做如下取值：

1）RTT：链路延迟较大时平均 $RTT=100$ ms；

2）RTO：最坏情况下，每次发生丢包都触发 RTO，根据 RFC 可知 RTO 约为 $4RTT$；

3）$loss$：广域网的丢包率在 2%左右。

在实际的直播系统中，很多视频编码器视频 ts 包的大小是 188 B，因此用 UDP 直接封装的报文大小是 $188*7=1316$ B，将 block 大小设置为 1316，segment 大小将随块数变化。利用上述时延公式，可得到 $Delay$ 和 k 的关系，如图 7 所示。

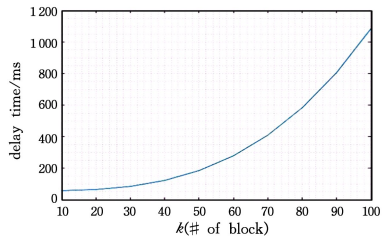

图 7　$Delay$ 与 k 的关系

Fig. 7　Relationship between $Delay$ and k

实际的参数值可以根据延迟目标选择，比如将延迟控制在 100 ms 以内时，编码块数需要足够进行分配，同时延迟不能太长。那么可以选择 $k=30$，此时可以由图 7 得到 segment_size 约为 40 KB。确定 k 的取值之后，实际发送的冗余块数根据式(1)即可获得。

4　实验结果

传输性能测试的环境如图 8 所示。通过 Linux 系统中的 Traffic Control 来控制带宽，在代码中控制段大小、块大小等参数。Traffic Control(下文简称 TC)工作于 Linux 内核，它通过建立处理数据包的队列，并定义队列中的数据包被发送的方式，来实现对流量的控制。下面举例给出对网络带宽、时延、丢包的控制命令。

(1)限制 eth0 网卡的出口带宽为 40 Mbps(报文在队列中最长等待 200 ms)：

tc qdisc add dev eth0 root handle 1：0 tbf rate 40 Mbit burst 40960 latency 200 ms

(2)在上述限速的网卡上配置 20 ms 时延：

tc qdisc add dev eth0 parent 1：1 handle 10：netem delay 20 ms

(3)在上述限速的网卡上配置随机丢弃 2%报文：

tc qdisc add dev eth0 parent 1：1 handle 20：netem loss 2%

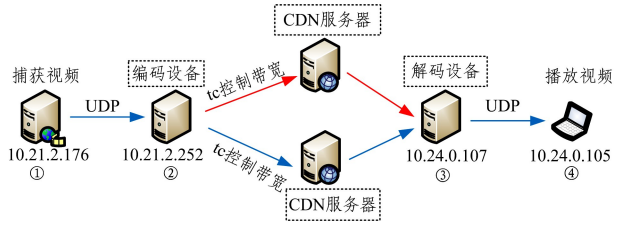

图 8　测试环境

Fig. 8　Test environment

在编码设备(见图 8 中的②)处，用 TC 设置两块网卡的出口带宽，例如将其都设置为 20Mbps 等；在网络编码模块中设置 segment_size 和 block_size 等属性。

在 CDN 服务器与解码设备(见图 8 中的③)之间的链路没有限制带宽，默认是千兆带宽，实验传输速率小于50 Mbps，远小于 1000 Mbps，可以认为在该链路上没有瓶颈。

(1)视频传输流程如下：

图 8 中，在①上捕获视频之后，通过 UDP 直接传输到②上；在②上基于开源网络编码包 Kodo[21]进行网络编码，并在

原有系统的基础上添加一个 UDP 接收程序，接收到 UDP 报文之后将其通过原有系统多路径发送出去，并在③处缓存解码；③基于 Kodo 进行网络解码，将解码后的数据通过 UDP 发送程序模拟 VLC（一款开源的跨平台多媒体播放器及框架）发送过程发送给④，并通过 VLC 进行播放。

（2）具体参数如下：本文选择 block_size 为 1316，因为视频的 ts 包固定为 188 B，在一个以太网 MTU 的 1500 B 下最多只能包含 7 个 ts 包，一共 1316 B。

本文选择 segment_size 为 10 块，因为 10 块已经可以有差别地在两条链路上分发。选 20 块或者其他值也是可行的，只要能在最后发送时分解成 1316 B 的 UDP 包即可。

（3）测试方法为：将编码设备②与 CDN 服务器间的带宽分别限制为 40 Mbps 和 20 Mbps，在各种网络条件下，分别采用 TCP 和本系统传输 100 MB 文件，进行 5 次传输测试后，取 5 次结果的平均值，然后进行比对。

4.1 丢包-传输速率的测试结果

对两条路径分别加上一定大小的丢包率，然后测试传输速率。实验结果如图 9 所示。

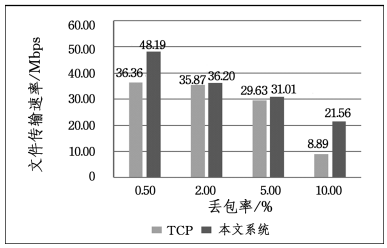

图 9 丢包率对传输速率的影响

Fig. 9 Packet loss rate effect on transmission rate

从图 9 可以看出，改变链路的丢包率时，TCP 的传输速率在丢包率较低时略小于多路径传输时间；而在丢包率较高时，本系统传输速率明显优于 TCP，其主要原因在于 TCP 的拥塞控制机制在丢包率较高时会严重降低窗口大小，从而使传输性能急剧下降。

4.2 时延-传输速率的测试结果

对两条路径分别加上一定大小的时延，然后测试传输速率。实验结果如图 10 所示。

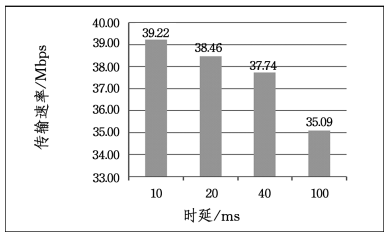

图 10 延迟对传输速率的影响

Fig. 10 Delay effect on transmission rate

从图 10 可以看出，时延大小对传输速率的影响不是很明显，虽然有一定的减小，但是总体来说减小程度不大，并且传输速率值较为稳定。

4.3 丢包＋时延-传输速率的测试结果

模拟广域网环境，丢包率采用 0.5% 和 2%，时延采用 20 ms 和 50 ms（由于丢包加上高时延，传输速率会急剧减小，因此时延主要采用 20 ms 和 50 ms 进行对比）。丢包加时延的测试结果如图 11 所示。

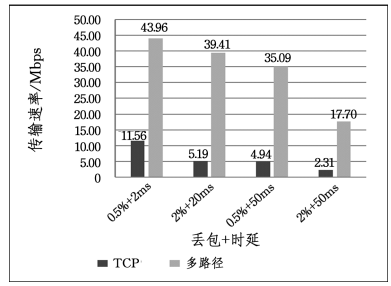

图 11 丢包＋延迟对传输速率的影响

Fig. 11 Packet loss & delay effect on transmission rate

从图 11 可以看出，丢包＋时延的组合对 TCP 的影响非常明显，TCP 的传输速率在丢包加高时延环境中会急剧下降，其主要原因是 TCP 的拥塞管理机制对丢包和时延敏感。在 2% 丢包率、50 ms 时延的网络条件下，TCP 的传输速率降低到 2.31 Mbps，而本系统的传输速率仍达到 17.70 Mbps，是 TCP 的 7.6 倍。

由以上测试结果可知，基于多路径网络编码的互联网视频直播系统在典型丢包和时延的网络环境下，对传输速率有明显的改进。

结束语 本文通过将网络编码与多路径传输相结合应用到互联网视频直播中，解决了互联网视频直播对用户体验影响最大的上传瓶颈和丢包问题。多路径传输策略有效地避免了单路径的瓶颈带宽，充分利用了多条路径的带宽，从而有效地提高了总吞吐率，也解决了网络编码导致冗余流量增加的问题。而网络编码消除了编码段内部数据的顺序相关性和丢包带来的影响，实现了多条路径高效协同传输数据，避免了丢包之后漫长的重传恢复时间，减小了丢包对直播体验带来的影响。实时测速并利用效用函数评估每条链路的性能，并根据不同链路的效用值分配各链路上传输的数据量，可以快速感知网络的变化情况，增强了互联网视频直播对于多变的网络环境的适应性。希望本文的研究能够对改进互联网视频直播在国内的研究与发展提供一些帮助。

参 考 文 献

[1] 中国互联网络信息中心. 第 41 次中国互联网络发展状况统计报告[OL]. http://www.cnnic.net.cn/hlwfzyj/hlwxzbg/hlwtjbg/201803/P020180305409870339136.pdf.

[2] CARBUNAR B, PEARCE M, VASUDEVAN V, et al. Predictive Caching for Video on Demand CDNs[C]// Global Telecommunications Conference. IEEE, 2011.

[3] ZHOU Y, CHEN L, YANG C, et al. Video Popularity Dynamics and Its Implication for Replication[J]. IEEE Transactions on Multimedia, 2015, 17(8): 1273-1285.

[4] HAIMSON O L, TANG J C. What Makes Live Events Engaging on Facebook Live, Periscope, and Snapchat[C]// CHI Conference on Human Factors in Computing Systems. ACM, 2017: 48-60.

[5] SIEKKINEN M, MASALA E. A First Look at Quality of Mo-

bile Live Streaming Experience: the Case of Periscope[C]// ACM on Internet Measurement Conference. ACM, 2016: 477-483.

[6] RODRIGUEZ-GIL L, ORDUÑA P, GARCÍA-ZUBIA J, et al. Interactive live-streaming technologies and approaches for web-based applications[J]. Multimedia Tools & Applications, 2017(6130): 1-32.

[7] ZHOU J, WU Q, LI Z, et al. A study on TCP performance of crowdsourced live streaming[J]. High Technology Letters, 2017, 23(1): 109-116.

[8] WANG B, ZHANG X, WANG G, et al. Anatomy of a Personalized Livestreaming System[C]// ACM on Internet Measurement Conference. ACM, 2016: 485-498.

[9] PANG H, ZHI W, CHEN Y, et al. Content Harvest Network: Optimizing First Mile for Crowdsourced Live Streaming[J]. IEEE Transactions on Circuits & Systems for Video Technology, PP(99): 1-1.

[10] DENG J, TYSON G, CUADRADO F, et al. Internet Scale User-Generated Live Video Streaming: The Twitch Case[C]// International Conference on Passive and Active Network Measurement. Springer, Cham, 2017.

[11] RODRÍGUEZ-GIL L, GARCÍA-ZUBIA J, ORDUÑA P, et al. An Open and Scalable Web-Based Interactive Live-Streaming architecture: The WILSP Platform[J]. IEEE Access, 2017, 5(99): 9842-9856.

[12] ZHOU J, WU Q, LI Z, et al. Demystifying and mitigating TCP stalls at the server side[C]// ACM Conference on Emerging NETWORKING Experiments and Technologies. ACM, 2015: 9.

[13] DUKKIPATI N, REFICE T, CHENG Y, et al. An argument for increasing TCP's initial congestion window[J]. Acm Sigcomm Computer Communication Review, 2010, 40(3): 26-33.

[14] CARDWELL N, CHENG Y C, STEPHEN C G, et al. BBR: congestion-based congestion control[J]. Communications of the Acm, 2017, 60(2): 58-66.

[15] FORD A, RAICIU C, HANDLEY M, et al. TCP Extensions for Multipath Operation with Multiple Addresses: draft-ietf-mptcp-multiaddressed-03[OL]. https://dial.uclouvain.be/pr/boreal/en/object/boreal%3A71362.

[16] ROSKIND J. QUIC-a multiplexed stream transport over UDP [OL]. https://www.chromium.org/quic.

[17] LI S Y R, YEUNG R W, CAI N. Linear network coding[J]. IEEE Transactions on Information Theory, 2003, 49(2): 371-381.

[18] KOETTER R, MEDARD M. An algebraic approach to network coding[J]. IEEE/ACM Transactions on Networking, 2003, 11(5): 782-795.

[19] SCHULZRINNE H, CASNER S, FREDERICK R, et al. RTP: A Transport Protocol for Real-Time Applications, RFC 3550 [S]. IETF, 2003, 7.

[20] WU Q, LI Z, TYSON G, et al. Privacy-Aware Multipath Video Caching for Content-Centric Networks[J]. IEEE Journal on Selected Areas in Communications, 2016, 34(8): 2219-2230.

[21] PEDERSEN M V, HEIDE J, FITZEK F H P. Kodo: An Open and Research Oriented Network Coding Library[C]// NETWORKING 2011 Workshops-International IFIP TC 6 Workshops. Springer-Verlag, 2011: 145-152.