计算机研究与发展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202330240
61(12): 3054−3068, 2024

实时多媒体传输延迟优化:架构、进展与展望

孟子立[1] 徐明伟[1,2]
[1](清华大学网络科学与网络空间研究院 北京 100084)
[2](清华大学计算机科学与技术系 北京 100084)
(zilim@ust.hk)

Latency Optimization in Real-Time Multimedia Transmission: Architecture, Progress and the Future

Meng Zili[1] and Xu Mingwei[1,2]
[1](*Institute for Network Sciences and Cyberspace*, *Tsinghua University*, *Beijing* 100084)
[2](*Department of Computer Science and Technology*, *Tsinghua University*, *Beijing* 100084)

Abstract Real-time multimedia transmission is one of the most important applications of the Internet, with the applications such as videoconferencing, cloud gaming, virtual reality and so on. In the same time, the real-time multimedia transmission systems therefore demand high requirements for the end-to-end transmission latency. Among them, latency fluctuation is the most challenging problem in latency optimization. However, traditional 'best-effort' transmission services in the Internet cannot meet the requirements of latency fluctuations for real-time multimedia transmission in many cases. We firstly elaborates the main challenges faced by real-time multimedia transmission. Secondly, we analyze the key issues that need to be addressed to optimize the latency of real-time multimedia transmission. Based on these issues, two key paths (control path and data path) and five core components in the architecture of real-time multimedia transmission system are summarized. Around the technologies involved in each component, representative research results are summarized and discussed, especially for the research efforts in the recent years. Based on the analysis above, the research branches for real-time multimedia transmission and low-latency applications are summarized, and the optimization algorithms and applications for each research branch are reviewed. A key finding in this paper is that the fluctuation of latency is the key metric that research needs to work on. Finally, we also discuss and propose possible future research directions for the readers.

Key words real-time multimedia transmission; latency fluctuation; real-time; network architecture; low-latency transport

摘 要 实时多媒体传输是互联网最重要的应用之一,其系统对于传输延迟提出了很高的需求.其中,延迟波动是延迟优化中最具有挑战性的问题.然而,传统的尽力而为的传输服务在很多情况下无法满足实时多媒体传输对延迟波动的要求.首先,阐述了实时多媒体传输面临的主要挑战.其次,分析了如果要优化实时多媒体传输的延迟亟需解决的关键问题.基于上述问题归纳了实时多媒体传输系统架构中的2个关键通路、5个核心组件.围绕各个组件涉及的技术,梳理了代表性研究成果.在此基础上,总结了面向实时多媒体传输及低延迟应用的研究分支,并对各研究分支优化算法与应用进行综述.通过分析发现,延迟的尾部波动是实时多媒体延迟优化应关注的主要目标.最后,提出了未来可能的研究方向.

收稿日期:2023-04-01;**修回日期**:2023-08-16
基金项目:国家自然科学基金项目(62221003)
This work was supported by the National Natural Science Foundation of China (62221003).
通信作者:徐明伟(xumw@tsinghua.edu.cn)

关键词 实时多媒体传输；延迟波动；实时性；网络体系结构；低延迟传输

中图法分类号 TP391

多媒体传输应用是互联网服务重要的组成部分. 多媒体传输应用包括音频、视频、图像、文本等多种多样的多媒体数据. 早在 2016 年时，多媒体流量便占了互联网总流量的一半. 据统计，2022 年，多媒体流量已经占互联网总流量的 82%[1]. 特别是自新冠疫情爆发以来，多媒体传输的实时性越来越受到关注. 国内的腾讯会议、国外的 Zoom 等软件广泛应用在教学、会议、远程办公等诸多场景. 此后，新兴的实时多媒体传输应用也吸引了广泛的关注. 实时多媒体传输已经扩展到了云游戏、虚拟现实、远程医疗等各方面，其面向的对象也从传统的人与人的通话扩展到了人机交互控制. 这其中，一些常见的场景包括全息视频会议、云游戏、虚拟现实、远程医疗、工业控制等方面.

延迟是实时多媒体传输最重要的指标，它直接与用户体验相关. 实时多媒体传输应用不仅需要低延迟，而且还要求延迟的稳定性. 例如，假设大多数时间，无线用户可以体验到满意的小于 100 ms 的往返时延. 但是如果网络往返时延的第 99 百分位数大于 400 ms，那么网络延迟将远远超过应用的延迟预算[2-3]. 在这种情况下，每 100 个数据包里面可能就有 1 个会出现高延迟，严重影响用户体验. 因此，降低尾延迟、稳定延迟波动对于实时多媒体传输应用来说至关重要.

实时多媒体传输的延迟优化有 2 个特点：

1）严格的截止时间要求. 由于交互式流媒体应用程序持续与人交互，因此控制端到端延迟对于获得较好的用户体验至关重要. 例如，视频会议希望端到端延迟小于 130 ms[4]，而云游戏则希望延迟小于 96 ms[5]. 实际上，服务器端和客户端处理通常需要的延迟约等于 30 ms[6-7]. 因此，网络的端到端往返时延不应超过 50~150 ms（视应用而定），这便是应用要求的截止时间[8].

2）高清高帧率的内容要求. 由于实时多媒体传输目前已经应用到了以云游戏、虚拟现实、远程手术为代表的需要高清晰度、高流畅度的应用场景，其便同样对网络有了很高的内容传输要求. 例如，远程手术一般会要求 2K 像素、4K 像素甚至更高的清晰度，才能将手术的每个细节都清晰地传输给远程的主刀医师；云游戏则需要 60 fps, 90 fps 甚至更高的帧率才能流畅地展现所有细节. 这样的综合结果如最近一些研究表明[9-10]，实时多媒体传输的比特率会达到数十甚至上百 Mbps.

综上所述，本文是对实时多媒体领域研究者提供的系统、架构与优化算法研究进展的概述与总结. 目前已经有了一些针对低延迟传输的研究成果，但它们对架构和算法关注较少. 例如，在 2000—2001 年期间有围绕实时视频及网络传输的整体架构性综述[11-12]，但其时效性较差. 近年来，有分别针对于纠错编码[13]、画面质量[14]以及实时多媒体传输与命名数据网络结合[15]等的综述论文，但缺乏系统性地综述分析现有延迟优化面临的主要问题的文章.

同时，随着实时多媒体传输的迅猛发展，本文也重点介绍了一些较新的研究工作，阐述本文提出的实时多媒体传输中对延迟分析应该采取的架构以及相关工作. 主要贡献包含 3 个方面：

1）分析了实时多媒体传输的延迟优化中可能会影响端到端延迟的组成部分，总结了其中主要组件与技术难题，并同时交代了实时多媒体传输架构的必要性；

2）归纳总结了现有的在数据通路和控制通路 2 个维度上对端到端系统的延迟优化工作，总结了尾延迟对实时多媒体的影响，并阐述了各类工作的性质及其应用；

3）基于现有研究进展及挑战，提出了实时多媒体传输优化未来可能的研究方向.

1 实时多媒体传输架构

当实时多媒体传输应用的优化目标转向尾延迟时，延迟的主要来源可能与原有架构中对中位数、平均延迟等分析的来源不再一致. 在本节中，我们着重分析，在现有的实时多媒体传输架构下，端到端延迟的波动到底源自哪里. 本节将首先在总体上对延迟波动产生的来源进行分析，接下来从控制通路和数据通路 2 个角度分析延迟波动产生的原因.

在传统的实时多媒体传输中，延迟的主要组成部分就是网络延迟. 当发送端和接收端的物理距离仍较远、拥塞控制等机制仍会产生很长的队列时，网络延迟便会占据大部分延迟. 然而，随着边缘节点的下沉部署、网络拥塞控制等机制的改进，网络延迟已经不再是延迟的主要组成部分. 当前的现实情况是，

在很多实时多媒体传输应用中，应用服务提供商会通过大量投资来换取延迟的降低. 因此，现在平均或者中位数甚至都可以低到 10~15 ms. 例如，像云游戏这一类应用，服务提供商会将服务器从原来可能全国的几个节点，下沉部署到每个省份、每个地区都有若干节点. 在这种情况下，针对绝大部分用户，在用户所在的城市就很有可能会有一个计算节点来为用户提供服务. 这样，数据包只需要在城域网之内传递，极大缩短了网络延迟. 类似地，当接入网技术不断从 4G 升级到 5G，WiFi 4 升级到 WiFi6 后，最后 1 跳接入网的无线链路传输的平均延迟也得到了很大程度的改善.

然而，当实时多媒体传输的关注点转向 0.1%，0.01% 的尾部延迟时，任何一个环节的微小波动均可能会导致端到端延迟在第 99.99 百分位上升. 这在网络技术升级换代的过程中也尤为明显. 例如，蜂窝网络从 4G 升级到 5G 以后，最后 1 跳接入网的无线链路尾部传输延迟有时候会变得极大，在网络状态切换过渡时，由于 5G 信号绕射能力差，其所提供的传输性能(延迟及带宽)可能还不及 4G 强信号提供的[16].

现有的实时多媒体传输架构在设计之时确实有考虑适应不同网络情况下的波动，但对网络从一个状态到另一个状态的过渡、收敛过程关注的还不够. 这也是自然的，因为当关注的延迟百分位在 50 分位乃至 90 分位时，其实并不需要怎么关心这些瞬态的收敛过程. 然而，当应用的目光放到这些很靠后的尾部延迟时，这些瞬态的收敛过程便十分关键. 因此，本节主要分析当实时多媒体传输关注这些 0.1% 的截止时间错失率、尾延迟之后延迟波动的可能来源.

本文工作发现的一个重要的延迟波动来源便是控制通路延迟的存在. 正如前所述，网络状态是时时刻刻都在波动的，因此端上的响应需要时刻根据网络状态来进行调整. 然而，由于控制回环的存在，端上的响应往往都是滞后的. 形式化地来说，在时刻t端上的响应动作$a(t)$并不能基于时刻t的网络状态$s(t)$，而是基于时刻$t-\tau_{\text{control}}$的网络状态$s(t-\tau_{\text{control}})$，其中$\tau_{\text{control}}$便是控制通路的延迟. 因此，当网络状态发生变化时，端上的响应动作$a(t)$往往会滞后于网络状态的变化，从而导致端到端延迟的波动. 这在当应用关注点从延迟转向尾延迟时，就变得非常重要了. 在此之前，当网络状态发生变化的时候，诚然，端上可能做出的响应稍晚了些. 但只要端上能够做出正确的反应，发送速率等参数便可以收敛到新的稳态值. 而这一瞬态的过程往往是短暂的，因此并不会对中位数、90 分位等延迟造成影响. 但网络波动确实是会偶尔发生的. 例如，Meng 等人[4] 的一个测量表明，在部分无线网络的真实数据下，网络带宽降低为原来 1/50 的概率可能高达 1%. 在这种情况下，控制通路的延迟便十分关键.

同时，数据通路中不同组成部分在端到端延迟中的角色也会有所变化. 随着边缘数据中心的部署、5G 和 WiFi 6 等新兴接入网技术的出现，网络端到端延迟在中位数一般情况下甚至可以做到 10~20 ms[10]. 在这种情况下，当我们观察到某一视频帧的延迟上升到数百毫秒时，造成其可能的原因便不再仅仅是因为双方的物理距离过远了. 应用层、传输层、网络层的延迟波动都有可能会导致瞬间的延迟上升.

本节将会对这 2 个方面进行分析. 图 1 展示了我们建模分析后的实时多媒体传输架构中可能对延迟抖动产生影响的一些组成部分. 一个视频帧首先在应用层会通过视频编码器以及应用层协议被编码成数个数据包(如空心方格所示). 传输层会进行一部分冗余编码，生成一些纠错数据包(如条纹方格所示)，然后网络层通过路由器将这些数据包发送到接收端. 接收端相反地首先进行丢包恢复，然后再由视频解码器进行解码，最终呈现给用户.

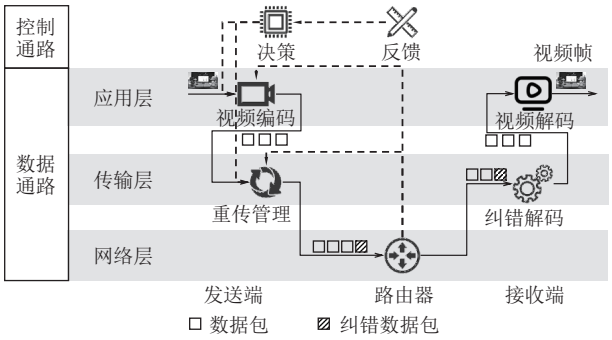

Fig. 1 Real-time multimedia transmission architecture

图 1 实时多媒体传输体系结构

在这其中，在数据通路上，应用层、传输层以及网络层的延迟也都会影响最终数据通路上的端到端数据延迟. 同时，在控制通路上，反馈延迟和决策延迟均会影响控制通路本身的延迟. 注意到，控制通路和数据通路是正交的，各个数据通路层均有控制延迟. 本文中涉及的多项工作也是按照控制通路和数据通路这 2 个方面分别进行展开，力求系统地解决实时多媒体传输中延迟波动的问题.

1.1 控制通路延迟

本节首先识别定位了控制通路对在尾部端到端

延迟出现波动的重要作用. 控制通路对实时多媒体传输的延迟影响是间接的, 况且仅仅在应用关注尾部延迟时才会有作用: 当网络状况发生了变化时, 如果多媒体传输的发送端调整自己的发送速率慢了些, 这便有可能导致延迟波动带来性能的下降. 这一点点响应的时间其实很关键: 如果每次网络状况变化时, 发送端都需要数百毫秒来响应, 那么这数百毫秒中多媒体传输的用户体验便是不好的. 然而, 互联网总是波动的. 如果每几分钟网络状态便有些许波动, 这就意味着用户会有千分之几的概率遇到延迟波动带来的性能下降. 这种情况下, 控制通路的延迟就变得非常重要了.

图 2 展示了一个例子来说明这个问题. 在互联网中, 一条流的可用带宽可能会因为无线信道的干扰、竞争流量模式的改变而时刻产生波动. 这时候, 实时多媒体传输的发送端的发送速率要随之改变, 以让自己的吞吐率实时适配当前的可用带宽. 不失一般性地, 当一个实时多媒体流在瓶颈路由器(图 2 中实线部分)的可用带宽突然下降为原来的 $1/k$ 时, 发送端的发送速率也需要尽快降低, 以适配新的可用带宽. 这一变化在近年来随着无线网络接入带宽的大幅度提升而变得更加明显[17-18].

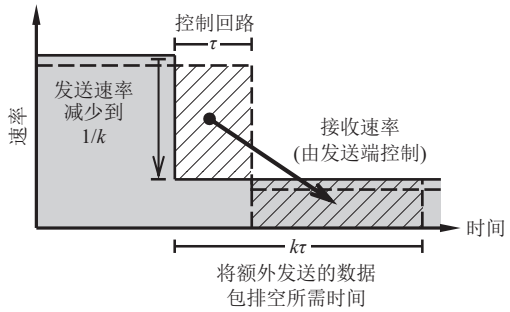

Fig. 2 An example of the control path delay when the available bandwidth drops

图 2 一个在可用带宽下降时控制通路延迟的例子

然而, 如前所述, 可用带宽的下降并非可以立刻被发送端知晓, 而是需要一个控制通路延迟 τ(即控制回路)才能最终反映发送速率下降. 在这种情况下, 发送端发送速率的减少便会由于可用带宽的减少向右偏移一些(图 2 中虚线部分). 图 2 中的实线是瓶颈路由器的可用带宽. 在这段时间内, 瓶颈队列仍然以原始发送速率接收数据包, 而其处理速率却因可用带宽的下降而大大减少. 因此, 这些过多的数据包会导致瓶颈队列的积压(图 2 中斜线阴影部分).

更糟糕的是, 当控制通路延迟为 τ 时, 用户体验到延迟恶化的时间却很可能远远不止 τ. 这正是本文的另外一个关于控制通路延迟很重要的观察: 当网络状况发生波动时, 由于事实上网络的可用带宽变差, 因此这些超出网络承载能力的数据包需要数倍于原来堆积起来的时间才能排空. 在这里也结合图 2 进一步分析. 具体来说, 到达瓶颈队列的数据包在控制回路 τ 期间需要 k 倍的时间($k\tau$)才能发送出去. 这是因为, 可用带宽下降前发送的数据率远大于下降后的新的可用带宽. 因此, 原来可能 1 个时间单位堆积起来的数据, 需要 k 个时间单位才能缓解掉. 这就像图 2 中所展示的那样, 事实上 2 块斜线阴影的区域面积是相等的. 在这段时间内, 所有发送出去的数据包都会经历增加的延迟, 从而降低用户的体验.

具体来说, 控制通路的延迟分为 2 部分, 即反馈延迟和决策延迟:

$$t_{\text{control}} = t_{\text{feedback}} + t_{\text{decision}},$$

其中反馈延迟 t_{feedback} 是指从发送端到接收端的反馈信号的传输时间, 决策延迟 t_{decision} 是指发送端根据反馈信号做出决策的时间. 而这 2 个环节的抖动均可能会导致最终的端到端延迟的波动. 本节首先介绍这 2 部分的功能, 其次讨论它们是如何影响最终的端到端延迟波动的.

1)反馈延迟抖动导致的端到端性能波动. 如何获取信息的反馈是几乎所有控制系统都会面临的重要问题. 从电路、信号到实时多媒体传输中的编码调整、容错调整、发送速率调整等都离不开反馈在决策中的重要作用. 这在网络领域诸多对现有工作的分析中也有所体现. 例如, QCN[19] 在分析稳定性时, 假设网络内交换机发送的 QCN 信号要经过时间 τ 后才能被发送端获取到. 因此, 这种反馈延迟事实上是普遍存在的.

端到端控制算法依赖于对网络状态的及时获取. 例如, TCP 拥塞控制算法会依据数据包的延迟、是否丢包以及一段时间内的速率变化来判断网络的拥塞程度. 当网络状态出现变化时, 这一变化会立刻反映到数据包的延迟、吞吐、丢包率等指标上. 然而, 我们发现, 在网络状态发生抖动时, 这些指标往往不能够立刻被发送端获知. 在这一瞬间过程中, 发送端发送速率与网络状态的不匹配, 便会导致端到端性能的波动.

2)决策不稳定导致速率调整滞后或出错. 获取网络状态信息后, 如何决策则是实时多媒体传输控制算法紧接着面临的另一个重要问题. 网络中的信

号往往是充满噪声与歧义的. 例如, 当发送端观测到一个丢包事件时, 这有可能代表发送端需要降低发送速率以缓解网络拥塞, 也可能仅仅代表无线链路的信道质量下降而事实上不需要发送端降低发送速率[20]. 因此, 决策算法一般倾向于只有当收集到足够多的信息才有足够的信心来做出决策, 这样决策才足够可靠.

近年来, 由于应用对性能的不断追求, 拥塞控制、视频码率调整等控制算法的决策逻辑已经越来越复杂. 从传统的寥寥数行的启发式方法开始, 研究者们逐渐转向使用神经网络[21-22]或整数规划[23]等方法来进行决策. 这些尝试是有益的: 一方面, 研究者们对网络链路的了解越来越深入, 他们能够有能力去针对性地对网络做出一些假设和建模; 另一方面, 受神经网络等新兴机器学习技术在计算机视觉、自然语言处理等领域的进展所鼓舞, 研究者们也倾向于相信这些神经网络在实时多媒体传输相关的领域内也大有可为.

1.2 数据通路延迟

本节首先定性地分析一下数据通路的端到端延迟的组成部分. 在非常理想的情况下, 数据通路的延迟会受到 3 个因素的影响:

$$t_{\text{data}} = t_{\text{app}} + (1 + RTX) \times t_{\text{RTT}}, \tag{1}$$

其中t_{app}, RTX, t_{RTT}分别是应用层的处理时间、重传次数和往返时延, 它们事实上对应着数据通路中应用层、传输层、网络层对端到端延迟的影响. 具体来说, 我们首先介绍这 3 个可能的设计缺陷会如何影响数据通路端到端延迟.

1) t_{app}是应用层的处理时间, 其主要和应用层的设计有关. 例如, 如果应用层需要对视频帧进行编码, 那么编码时间便会增加.

2) RTX是重传次数, 其主要和传输层的丢包恢复设计有关. 例如, 如果一个包丢了RTX次, 其便会在第$1+RTX$次传输时到达接收端.

3) t_{RTT}是往返时延, 其主要和网络层的队列管理设计有关. 例如, 网络内的队列越长, 往返时延便会越长.

例如, 假设某数据包在发送端的编码器首先处理了 5 ms(应用层). 然后, 该数据包在传输层开始准备进行传输. 当前的网络往返时延是 30 ms. 不幸的是, 这个数据包在网络中总是被丢弃, 直到第 4 次传输才成功到达接收端被确认. 假设这里在最理想的情况下, 发送端总能在 1 个往返时延内就判断到这一个数据包被丢弃了. 因此, 在传输层, 这个数据包总耗时为 30 ms×4=120 ms. 到了接收端之后, 数据包在应用层可能还会有一些附加的延迟. 例如, 视频的解码可能就会耗时 10 ms. 这样, 按照式(1)的估计, 总的端到端延迟大约是 15 ms+120 ms=135 ms.

这个模型当然存在一些近似之处. 例如, 事实上, 发送端不见得能在第一时间发现数据包被丢弃. TCP 在数据包第 1 次丢弃后, 需要等待后续 3 个数据包都成功到达并被确认才会触发快速恢复机制. 在数据包第 2 次丢弃后, 则要等到重传超时(RTO)后才会继续重传. 式(1)只是估计的一个理想情况, 我们总能通过改善协议设计来达到估算中的理想情况, 例如 WebRTC 框架中的 RTP/RTCP 的 NACK 设计. 但总之, 上述估计给出了一个关于数据通路延迟的主要组成部分的分析.

然而, 近年来, 在实时多媒体传输中, 这 3 部分的情形均各自发生了一些变化. 这 3 个环节的抖动也均会导致最终的端到端延迟的波动.

1)应用层延迟. 视频质量增加导致应用层编解码与网络协同过程中出现波动. 在应用层延迟中, 一个比较大的变化是在应用与协议栈接口处(如套接字缓冲区)中的等待延迟. 现有应用层的设计并没有考虑到应用出现瓶颈可能引入的延迟波动. 当前操作系统中套接字的缓冲区只会被动等待应用向其读取数据, 而不会进行主动队列管理. 当应用能够来得及处理当前网络发送过来的数据时, 应用会主动从缓冲区中读取数据. 当应用暂时来不及处理这些数据时, 这些数据便会堆积在缓冲区中等待. 当缓冲区逐渐减小时, 当前的 TCP 协议会相应修改流控窗口(advertised window), 来告知发送端减少发送的数据量. 当缓冲区被填满后, 接收端便不再接收网络中的新数据, 直到应用处理了一部分数据腾出空间为止.

然而, 这样的设计面临的一个直观的问题就是, 如果应用处理不及时, 队列会持续堆积直到堆满. 这就好比在网络中路由器上, 队列如果不主动管理就会堆满直至溢出一样, 在这种情况下, 就会产生很高的排队延迟. 这样的设计对包含实时多媒体传输在内的低延迟的应用是很不友好的.

这个队列的问题随着多媒体传输应用的发展而越来越严重. 随着新一代多媒体对画质的要求逐渐变高, 视频编解码的计算量也随之增加. 例如, 分辨率从过去的 240 像素发展到如今的 1 080 像素, 未来可能还会出现 4K 像素、8K 像素等更高分辨率的实时多媒体传输. 视频的帧率也从视频通话的约 24 fps

增长到 60 fps, 90 fps 甚至更高. 这都会让应用处理数据的负载越来越重, 进而导致端到端延迟的波动.

2)传输层延迟. 对延迟波动要求的提升使得现有传输层丢包恢复机制无法满足其条件. 在传输层延迟中, 本文注意到当对延迟的百分位的关注从 50 分位、90 分位分别提升到 99.9 分位、99.99 分位时, 现有的传输层丢包恢复机制已经无法满足这一要求. 当前传输层很多设计都没有考虑这些小概率尾部情况可能带来的时延问题. 一个典型的例子就是丢包恢复. 当数据包没有被丢失时, 在延迟上大家相安无事. 然而, 如果一个数据包不幸丢失了, 传输层现有的设计便在第 1 次触发快速恢复机制, 但第 2 次可能要等待 1 s 才能触发超时重传. 尽管有许多设计尝试加速这一过程(如 TLP[24]), 但重传通常仍然是不可避免的. 这样的一个问题是, 当真的比较极端的情况发生时, 在这些极端情况下的数据包的延迟可能会非常大. 例如, 如果 1 个数据包需要被传输 4 次才能到达接收端, 那么这个数据包的延迟会增加 4 倍. 而在丢包率为 10% 的时候, 1 个数据包被传输 4 次, 即连续被丢弃 3 次的概率也高达 0.01%. 这便会直接影响用户在尾部的体验.

需要注意的是, 尤其是针对实时多媒体传输的视频帧而言, 对于一般的解码器, 只有当 1 帧的所有数据包都到达接收端后, 这 1 帧才能被交付给应用进行解码、渲染. 也就是说假设 1 个视频帧可能有 50 个数据包, 那么哪怕有 1 个数据包出现了上述的情况, 那么这 1 个视频帧便会让用户的体验下降. 因此, 现有的丢包恢复的机制很难满足新一代多媒体传输对延迟抖动的极致要求.

这个尾部的问题也随着实时多媒体传输应用的发展而变得越来越严重. 游戏用户可能可以容忍 0.1% 的卡顿率, 但像是远程手术、远程辅助驾驶等应用其实会需要 0.01% 甚至更低的卡顿率. 想象一个复杂手术或者一段长途旅行可能持续数十个小时, 其中哪怕卡顿了几秒钟都可能是非常致命的. 这数十个小时中的几秒钟就大概是 0.01% 甚至更低的卡顿率要求. 同时, 随着视频传输码率的提高, 而网络中的最大传输单元(maximum transmission unit, MTU)并没有随之提高, 前述的问题中的 1 个视频帧包含多个数据包、被其中 1 个数据包拖慢的现象会更加严重. 因此, 对以丢包恢复为主的这些在设计时并未很好考虑尾部延迟波动的应用, 重新考虑便十分重要了.

3)网络层延迟. 拥塞控制算法的多样化使得网络层队列管理机制出现性能波动. 本文注意到, 近年来, 网络层的队列管理机制所面临的流量特征和应用需求也发生了一些变化. 如前所述, 网络层中延迟的主要来源就是排队. 而排队的造成一般即是由队列的到达速率与发送速率不匹配带来的. 在传统的网络层的队列管理中, 有 CoDel 等算法[25] 通过对队列的长度进行限制, 来避免队列过长导致的延迟波动, 并得到了广泛的部署. 然而, 这些算法在当今的网络传输的流量特征中面临 2 个问题:

1)队列管理算法的响应更多针对丢包而不是其他指标. 当前的队列管理算法, 在设计之初, 针对的是一些较为传统的 TCP 拥塞控制算法, 例如 Reno[26], CUBIC[27] 等. 这些算法最突出的特点, 就是它们高度依赖于丢包信号或者 ECN 信号来调整速率. 比方说, 当发送端没有观测到丢包时, 它们便会持续尝试去增加发送速率或拥塞窗口. 当发送端观测到了丢包时, 它便会减少发送速率. 然而, 以实时多媒体传输应用为代表的低延迟应用其实不再采用类似 CUBIC 这种丢包敏感的拥塞控制算法, 而是采用延迟敏感的拥塞控制算法以获取低延迟. 这就导致现有的队列管理算法可能不见得能有效控制延迟: 延迟敏感的拥塞控制算法不再对丢包信号响应. 这便让低延迟的保障变得不那么简单.

2)队列管理算法的设计更多关注于宏观表现而不关注微观表现. 当前队列管理算法在衡量性能时更多关注的是长时间尺度上的宏观表现. 例如, 在公平性上, 研究者们会去测量在比较长的时间尺度上的吞吐量公平性指标, 如 Jain's 公平性指数. 但对在瞬态下如何慢慢收敛达到这一公平性并没有太多关注. 在此, 当性能关注的重点转移到尾延迟时, 如前所述, 这一收敛过程的瞬态性能便同样很关键了. 尤其是当网络上的一些竞争流量, 如网页浏览也出现了一些新的变化时, 网络层的队列管理机制就难以保障多媒体传输的延迟波动在可接受的范围之内. 如果, 像现有方法一样, 并不关注在瞬态下性能的波动, 那么用户便会在尾部遭受性能损失与体验下降.

2 数据通路应用层相关工作

实时多媒体传输应用主要包含 3 个主要功能:

1)当视频源产生了画面之后, 编码器需要将画面编码为视频流. 例如, 视频会议的画面从用户端的相机中获取, 云游戏及虚拟现实等的画面由 GPU 渲染而成. 这都需要将原始视频进行编码才能够继续传输.

2)在视频编码的过程中,编码的参数(主要是码率)需要根据网络状态来进行实时调整.例如,当网络状态变好时,编码器可以适当提高编码码率,以为用户送达更加清晰的视频内容.而类似地,当网络状态变差时,编码器也会降低码率,以确保用户能够看到不卡顿的内容.

3)当内容准备好后,发送端还需要将这些内容通过应用特有的协议进行发送,以便应用更好地对内容进行管理.如本文引言所述,针对多媒体传输,在应用层及与应用联合设计方面,近年来的工作越来越多.本文将按照上面介绍的3个方面,如表1所示进行了整理,并在后面分别进行介绍.

Table 1 Related Work in Real-Time Multimedia Optimization from the Application Layer

表1 应用层的实时多媒体优化相关工作

学界/业界方案	解决途径	主要思路
Swift (NSDI'22)[28] CGEncoder (MMSys'20)[29] NAS (OSDI'18)[30] LiveNAS (SIGCOMM'20)[31] VP9 (Google'13)[32]	实时编解码器	通过更新编解码设计在弱网等情况下依然保证内容解码
BB (SIGCOMM'14)[33] Pensieve (SIGCOMM'17)[22] Puffer (NSDI'20)[34] AFR (NSDI'23)[10]	自适应码率算法	通过让码率适配带宽来降低卡顿
RTP/RTCP (RFC8888)[35] RTSP (RFC7826)[36] DTP (ICNP'21)[37]	多媒体传输协议	通过协议设计来传递应用需要的信息

2.1 编解码器优化

编解码器的发展历史很长,应用范围也很广.视频编码的主要原理是利用视频内容的时间相关性和空间相关性,大幅度通过存储差分值等方式压缩内容以节约带宽成本.目前比较广泛使用的编解码器以H.264为主.近年来,编解码器的优化工作主要集中在2个方面,一是优化编解码器的性能;二是针对应用的具体场景对编解码器进行针对性地设计和优化.在编解码器本身的性能优化上,近年来提出了新一代的H.265编解码机制,其能够在同样清晰度下节省大量的带宽成本.然而,由于这一新的机制涉及专利权等诸多问题,目前部署的情况远不如H.264理想.同时,最近还有研究者正在推动H.266编解码的标准化与示范性应用.而在实时多媒体传输领域,目前使用较为广泛的是由谷歌公司推动部署的VP9编解码器[32].其现在已经被包含在WebRTC实时音视频传输框架内,可以方便地被开发者使用.

此外,还有其他的研究工作,专注于优化其他应用指标,例如SSIM[38]或PSNR[39]针对如图像或视频质量;Alfalfa[40]针对实时多媒体传输的转码过程中大量用户的多线程并行进行了特殊优化;Salsify[41]则进一步让编码器注意到网络状态,并为网络预留了可以调整的空间;最近的工作Swift[28]更是利用神经网络等技术进一步优化编解码器的编码效率及处理延迟,让用户能够更加流畅地使用虚拟现实(virtual reality, VR)等技术;CGEncoder[29]则是结合了云游戏等游戏应用的特点,分析了游戏用户的具体需求,即游戏用户的用户体验可能不见得完全与PSNR或SSIM等清晰度的客观指标完全一致,并基于此进一步设计了编解码机制使得编解码器更适配于云游戏的场景.

同时,设计编解码器最大的问题是可部署性的问题.视频的解码又对实时性有着很强的需求:当1个视频帧播放完成时,下一帧应该已经完成解码准备播放才不至于使用户感受到卡顿.然而,编解码包含大量的数学运算,其在CPU内是极其消耗资源的.因此,现有解决方案中,一般在CPU或显卡内均有专门的编解码芯片来进行处理以加速解码过程.但是,这些硬件解码芯片不见得支持上述提到的新兴编解码机制.事实上,许多工作自身谈及的一大不足便是不能被现有硬件支持.因此,尽管H.264编码机制已经被提出了20余年,其仍是目前应用最为广泛的编解码机制.

除了编解码领域本身的优化工作,在网络领域也有一类工作尝试通过提升压缩效率来在传输同等质量的画面的情况下节约带宽.这类技术利用计算机视觉领域最新的"超分辨率恢复"技术来在发送端和接收端分别压缩和恢复视频流以节省带宽成本.例如,NAS[30]提出通过其设计的多层神经网络进行视频质量压缩,能够在同样网络带宽下有效提升传输质量.LiveNAS[31]则解决了NAS中所需要的针对每个模型分别训练的问题,针对直播等无法事前获得视频内容的场景进行了加强训练.Nemo[42]特别针对移动设备等资源受限的设备进行了计算代价的优化,使得这种技术也能应用在手机等资源较少的用户终端.

2.2 自适应码率优化

正如引言所述,自适应码率是现有多媒体传输机制中的重要组成部分之一.其主要功能是针对网络状况波动对编码器的编码码率进行修改,以确保编码器的码率不要超过网络的承载能力.在实时多媒体传输诞生的时候便有了相应的自适应码率算法.最近10余年来,比较典型的算法是由网飞公司提出的Buffer-Based算法[33].该算法创新性地提出通过估

计在点播多媒体视频中客户端缓冲区的情况，来调整网络中传输的多媒体码率. 当客户端缓冲区较低时，这意味着客户端卡顿的风险增高，此时就需要降低编码码率以使得新的内容尽快传送到客户端. 当客户端缓冲区较高时，这意味着编码器可以尝试探索更高的编码码率，以为用户提高更好的画质. 在这一方向上，还有类似 BOLA[43] 等算法，同样基于缓冲区的占用情况进行决策，但同时又能够基于李雅普诺夫稳定性进行理论分析. BOLA 算法目前是应用最广的点播流媒体框架 dash.js 的默认自适应码率算法. 在此之后，还有 BOLA-E[44] 等算法进行了改进与提升，进一步对基于缓冲区的方法进行优化.

除此之外，还有发送速率的估计算法，比较有代表性的是 PANDA[45] 与 SQUAD[46]. 其类似于拥塞控制，根据网络状态来估计出当前比较适合传输的视频码率. 并行地，还有许多混合基于缓冲区与基于网络状态的自适应速率控制算法. 例如，有学者提出采用整数规划对自适应码率问题进行系统性建模，并提出 RobustMPC 算法[23] 进行优化求解，得到最适合当前传输的码率决策. 近年来，更是出现了使用机器学习算法来优化自适应码率算法的热潮. Pensieve[22] 是采用深度神经网络来优化自适应码率选择的工作. 其采用深度强化学习，对自适应码率问题进行了建模，并设计了相应的状态空间、动作空间及奖励函数，采用一系列算法对自适应码率算法进行优化. 随后，还出现了 HotDASH[47] 等工作对神经网络的结构、优化方法等进一步优化以提升用户的体验. 这方面的工作也一直是学术界近年来的热点. 然而，目前的算法[22-23] 与一些分析给出的理论最优仍然存在较大的差距. 因此，本文认为还有很大的改进空间. 作为一个例子，学术界不断举办比赛寻求更好的 QoE 算法[48]. 总之，为了获得更好的性能，不断优化的方法已经并将继续被提出[48].

2.3 多媒体传输协议设计

近年来另一种重要的研究工作，是设计新的多媒体传输协议. 应用层协议在互联网体系结构中，也是不可或缺的. 在传输层协议之上，需要应用层也有相应的协议才能确保内容的正确传输. 最近应用最广的协议是 RTP/RTCP 协议[35]. RTP 和 RTCP 是一对基于 UDP 的协议. 其中，RTP 协议从服务器向客户端发送数据包以传输视频内容，所以其是数据通路的协议；RTCP 负责从客户端向服务器反馈网络状态、视频状态等状态信息，所以其是控制通路的协议. RTCP 协议会构造发送端报告（sender report, SR）、接收端报告（receiver report, RR）来报告发送端及接收端的信息. RTCP 协议还会构造 NACK（negative acknowledgement），TWCC（transport-wide congestion control）报文以向发送端报告丢包及延迟情况. 发送端可以选择性地决定是否要重传某些数据包. 在这种情况下，这样一个基于 UDP 的应用层协议其实也基本可以实现几乎可靠的传输. 无论是开源框架 WebRTC，还是工业界的解决方案如 Zoom 或 Google Meet，均采用了这种协议或其变种进行传输.

在历史上，还有 RTSP[36] 等协议针对多媒体传输应用进行传输. 这些协议基于 TCP 等可靠传输协议，以免去了在应用层确保传输可靠性的开销. 同时，最近还有一些学者注意到新兴应用中对于延迟的敏感需求，提出了诸如基于截止时间的传输协议（deadline-aware transport protocol, DTP）[37,49]，来在协议设计中搭载对应数据报文的截止时间信息. 在这种情况下，无论是网内设备还是接收端，均可以根据数据包的截止时间信息进行更合理的数据包调度，以使得最终交付的数据包能最大化地满足应用的需求.

3 数据通路传输层相关工作

传输层是互联网优化中，尤其是 SIGCOMM/NSDI 所代表的网络社区中，受关注非常多的一个组成部分. 传输层主要的功能就是确保应用层交付给它的数据能够及时可靠地到达接收端. 这在延迟出现变动、可用带宽波动，以及网络的丢包的时机发生变化下尤为难得. 因此，传输层的 2 个主要功能即是速率控制和可靠传输. 速率控制功能在实际互联网中以拥塞控制为主，而可靠传输则主要以丢包恢复为主. 速率控制更多地在宏观尺度上，尝试通过调整拥塞窗口等手段，避免互联网中出现长时间的排队并确保互联网的效率. 而可靠传输则更多地在微观尺度上，针对某一个或某几个丢失的数据包，将它们能够送达到接收端. 如表 2 所示，我们便将从 2 方面对现有工作中目标与实时多媒体传输接近的工作进行简要介绍.

3.1 拥塞控制

拥塞控制至今已有 40 余年的历史，在此不再赘述. 其中，低延迟拥塞控制受网络研究者关注已久. 早先的拥塞控制算法如 Reno[26]，CUBIC[27] 等方法是通过尽可能挤占队列的方式来提升网络资源利用率，但这会导致端到端延迟的上升. 近年来应用比较广泛的是谷歌公司 2016 年提出的 BBR[59] 算法. BBR 通

Table 2 Related Work in Real-Time Multimedia Optimization from the Transport Layer

表 2 传输层的实时多媒体优化相关工作

学界/业界方案	解决方案	主要思路
Sprout (NSDI'13)[50] GCC (MMSys'16)[51] NADA (RFC 8698)[52] SCReAM (RFC 8298)[53] Copa (NSDI'18)[54] Vivace (NSDI'18)[55]	拥塞控制	通过让发送速率适配带宽以减少延迟
WebRTC (ICIP'13)[56] AdaptFEC (MM'19)[57] StreamMelt (NSDI'23)[58] TLP (RFC 8985)[24]	丢包恢复	通过减少重传来减少端到端延迟

过测量链路的瓶颈带宽与往返时延来判断到底有多少数据报文应该在网络中,并以此速率为发送速率来发送数据包. 这样, BBR 不再需要挤占瓶颈队列,因此可以获得较低的延迟. 除此之外,还有 Sprout 算法[50]、Verus 算法[60]、Copa 算法[54] 等进一步利用延迟信息针对 TCP 协议进行更为精准的速率控制. 例如, Verus 是专门针对蜂窝网络的信道波动性而对延迟估计做出适应性调整的拥塞控制, Copa 是利用延迟波动的信号对端上的拥塞窗口进行调节,它们均能够较为有效地获得较低的延迟.

在实时音视频领域,上述算法[26-27,50,54,59-60] 也得到了一定的部署. 例如,脸书公司在其直播业务中采用 Copa 进行测试,并取得了较好的效果. 除此之外,也有许多专门针对实时音视频应用的拥塞控制算法被提出. 例如,谷歌公司提出的 GCC[51] 算法应用在 WebRTC 框架中. GCC 通过利用延迟梯度(delay gradient)的信息来控制发送速率:测量每个数据包的延迟通常是不准确的,因为区分排队延迟和传输延迟一直是端到端拥塞控制算法的难题. 因此, GCC 关注 2 个数据包的延迟的差,其称之为"延迟梯度"来进行速率控制:当数据包延迟在逐渐增加时,网内的瓶颈队列大概在堆积的过程中. 这时, GCC 便会降低其发送速率,反之亦然. 除此之外,思科公司和爱立信公司也分别提出 NADA 算法[52] 和 SCREAM 算法[53] 来专门针对实时音视频传输进行优化,并进一步利用了一些包含显式拥塞控制(explicit congestion notification, ECN)等信息来降低端到端传输的延迟.

然而,即使拥塞控制在管控延迟抖动上做出了许多努力,现有算法依然难以做到令实时多媒体传输应用达到满意的延迟. 用户依然在很多情况下经受着糟糕的网络体验. 这一方面当然是因为应用对延迟、卡顿的需求变得越来越高,另一方面也说明了单纯的端到端拥塞控制算法优化的局限性.

3.2 丢包恢复

丢包恢复是网络传输中的一个重要问题,它的目的是在网络出现丢包时通过恢复来保证传输的可靠性. TCP 协议区别于 UDP 协议的一个显著性特点便是其能够在内核中有效地对丢失的数据包进行恢复. 针对包括实时多媒体传输在内的绝大部分应用,当网络中出现了丢包时,主要的恢复方式是对这个数据包重新传输. 重传数据包也有许多设计需要进行考量,最主要的是如何判断一个数据包已经丢失,而不是乱序、延误等. TCP 初始的设计是采用 RTO 来进行判断:当等了一段时间(一般为 1 s 或 200 ms)依然没有收到原来数据包的确认包后,发送端会选择对这一数据包进行重新传输[61]. 随后,快速重传机制的提出让连续 3 个相同的确认包即可快速触发重传. 近来的工作则是提出了 TLP 尾丢弃探测[24] 等机制:在等待 3 个相同的确认包耗时较长的情况下,依然能够及时地将丢弃的数据包进行重传.

另外一条线上的研究便是通过引入冗余的方式来进行丢包恢复. 这种方式也较容易理解. 例如,当发送端打算发送 3 个数据包时,由于发送端担心数据包可能丢失,因此发送端可以采用前向纠错码(forward error correction, FEC)的方式编码出第 4 个数据包,并将这 4 个数据包一并发出. 此时,只要接收端接收到任意 3 个数据包,其均有能力将第 4 个数据包恢复出来. 在这一方向上,一种做法是采用现有的 FEC 技术,但根据当前网络状态动态调整其参数:当网络丢包率变高时,冗余包的比例就多些. 例如, Bolot 算法[62] 和 USF 算法[63] 分别根据历史丢包恢复的情况,视其恢复能力进行参数调整. 在这个方向上,后续还有 WebRTC 的 FEC 参数策略[56] 甚至近年来采用深度强化学习等机器学习工具进一步预测网络状态并优化冗余参数的算法[64]. 这些算法在不同的实验测试环境中均能取得较好的性能,有效地对数据包丢失的情况进行恢复.

除了对冗余的参数进行优化外,另一类做法便是直接对冗余编码的机制进行设计,这一般需要较强的群、环、域等数学相关的知识,其中的许多工作也发表在信息论相关的期刊上,例如 IEEE Transactions on Information Theory. 比较有代表性的是 AdaptFEC[57]、Chen 等人[64]、Fong 等人[65]、Krishan 等人[66] 的编码机制等. 然而,这些算法由于其复杂度实在太高,在实际部署中困难较多. 事实上,目前在实时多媒体传输中应用较为广泛的冗余编码机制是异或码,甚至一些稍复杂的如 RS 码应用都尚不成熟.

4 数据通路网络层相关工作

广域网中网络层的优化工作在近年来并不是很多. 这主要是因为网络层的主要组成部分是网络内部的路由器. 在数据中心等其他场景下, 路由器(或交换机)的更新换代较为频繁. 因此, 新的技术有机会能够得到较快的部署. 而在广域网下, 几乎不存在一个实体控制一条路径上所有设备的情况. 因此, 在下面讨论的工作中, 其可部署性是我们关注的很重要的一点.

在路由器上, 能够做的事情就是对流经路由器的数据包进行操作, 以隐式或显式地告知发送端当前的网络状态. 基于此, 隐式地告知发送端可以通过主动队列管理的技术来达到低延迟, 即当网络状态恶化时, 路由器可以选择性地丢弃一些包; 也可以通过直接调整队列大小来从物理上限制其延迟的最大值, 即如果缓冲区太小, 数据包不得不被丢弃, 这样尽管丢包率可能上升, 但延迟也能够有界, 而这对实时多媒体不见得是坏事. 还可以通过显式地构造新的网络层协议, 将网络状态捎带回发送端以达到信息传递的目的. 如表 3 所总结的那样, 本节将对这些工作分别进行综述.

Table 3 Related Work in Real-Time Multimedia Optimization from the Network Layer

表 3 网络层的实时多媒体优化相关工作

学界/业界方案	解决方案	主要思路
CoDel (CACM'12)[25] RED[67], BLUE[68], GREEN[69], Yellow[70]	主动队列管理	尽早丢包迫使 发送端降速以避免 超量发送数据
BDP/n (SIGMETRICS'21)[71] ABS (INFOCOM'22)[72]	队列大小优化	设置合适的队列 大小以降低延迟
XCP (SIGCOMM'02)[73] RCP (INFOCOM'08)[74] Kickass (ICNP'16)[75] ABC (NSDI'20)[76]	端网消息传递	携带更多维度的网内 状态以便决策

4.1 主动队列管理

在网络层, 主动队列管理(active queue management, AQM)是一种用来控制网络拥塞的常用方法. 路由器上有许多 AQM 算法. 较早的 AQM 算法是 RED[67], 其通过在早期以概率性随机丢弃的方式来告知当前网络恶化的情况. 目前在许多边缘路由器上部署的默认 AQM 算法是 2012 年提出的 CoDel[25], 其主要解决的问题是根据队列长度的估计难以适应不同带宽的路由器, 而相比之下采用在队列中的停留时间可以更精准地控制延迟目标.

除此之外, 还有更多的 AQM 算法被提出, 如 BLUE[68], GREEN[69], Yellow[70] 等. 最新的进展是 2023 年刚刚成为 IETF RFC 的 DualQ 算法[77], 其是 IETF 的 L4S 工作组的一部分, 通过将数据流区分成不同的类别然后分别进行主动队列管理. 除此之外, 在数据中心同样有大量的工作对数据中心交换机的队列进行管理. 包括 PIAS[78], pFabric[79], ABM[80] 等算法. 然而, 这些工作与广域网中主动队列管理最大的区别是它们可以假设端主机的配合: 一个企业的数据中心内, 该企业能够同时控制交换机与服务器. 这为流类型的区分、流量大小的估计等都提供了极大的便利. 然而, 在互联网中, 我们不能有这样的假设. 如果说某算法会对某一类流量进行优先调度, 那么所有互联网的用户都会把自己的流量伪装成这一类流量, 从而使这一机制失效. 事实上, 这也正是差异化服务[81]等机制在广域网下使用人数较少的原因之一.

相比之下, 主动队列管理机制还存在的一个共性问题是假设端上的拥塞控制算法对丢包或者 ECN 标记敏感. 然而, 随着 Copa, BBR 等新兴的基于速率的或者基于延迟的拥塞控制算法的提出, 它们不再对丢包或者 ECN 敏感. 因此, 如果期望依靠丢包使得端上的拥塞控制算法降低其发送速率, 需要丢包已经十分严重才行. 例如, BBR 对于丢包率在 20% 以下的丢包不会做出任何速率上的响应. 因此, 迫切地需要针对这种延迟敏感的拥塞控制算法设计优化新的主动队列管理机制.

4.2 队列大小优化

如何设置瓶颈队列大小一直是网络层管理的一个难题. 队列过小会导致网络当中应对突发流量时频繁出现丢包, 而队列过大则会导致在速率调整不及时的时候排队延迟过长. 因此, 设置合适的队列大小一直是网络管理员关心的一个问题, 也是降低端到端延迟的一个重要手段. 在这个方向上, 已有一系列经验性的工作探讨. 例如, 2019 年以斯坦福大学教授 Nick McKeown 为首的专家们专门组织了 Buffer Sizing Workshop 来对如何设置交换机队列大小进行讨论. 除此之外, 还有许多自适应队列大小的工作, 如 ABS[72] 等会根据网络流量的突发性来自适应地调整路由器中队列的大小.

另一类队列大小优化的主要工作就是理论上的分析. 最早, 学者提出, 瓶颈队列的大小应该不少于带宽延时积(bandwidth-delay product, *BDP*), 以使得当时的拥塞控制算法, 例如 AIMD 或者 Vegas 至少能够

在整个周期内都充分利用链路容量. 随后, 在 2004 年, 斯坦福大学的研究者们提出, 其实利用不同拥塞控制流的统计复用特性, 瓶颈队列的大小可以降低到 $BDP/\sqrt{N}$, 其中 N 为在该交换机上的流的数目[82]. 近年来, 又有学者指出, 随着 BBR 等新型拥塞控制算法的提出, 瓶颈队列的大小可以进一步降低到 BDP/N[71]. 这样, 交换机上可能的延迟的最大值也会因此不断降低.

然而, 这一设置一般只适用于核心骨干网交换机, 因为这些交换机通常有上百万条流. 在边缘路由器(例如家里的无线路由器)中, 可能绝大部分情况下只有数十乃至上百条流. 在这种情况下, 由于 N 很小, 这一结果其实是平凡的. 更严重的问题是, 在无线网络中, 如前所述, 由于其带宽波动很可能较大, 所以队列不得不设置得很长. 这其实也直接导致了很多最后 1 跳路由器的队列缓冲区都很“深”的情况. 在这种情况下, 高延迟的发生便是难以避免的.

4.3 端网消息传递

最后一类工作便是在网络层设计新的协议来在端主机和网络设备之间进行更好的消息传递. 良好的消息传递也可以方便地控制延迟, 因为在理想情况下, 如果端主机能够完美复刻网络可用带宽的变化, 便不会有因为自身调整失当带来的拥塞. 这方面典型的工作便是 2002 年的 XCP[73], 以及后续的 RCP[74]. 它们通过设计新的协议, 在协议包头字段包含瓶颈带宽的速率, 以精准调控发送速率. 这同样也包含一些可能没有设计新协议, 但同样在现有协议内捎带了网络状态信息的工作, 例如 Kickass[75] 和 ABC[76]. Kickass 将当前路由器上的某一条流的可用带宽这一信息, 通过 IP 分片的大小传递回发送端. ABC 则是利用了差异化服务遗留的 2 个在互联网上未广泛使用的比特(TOS 比特)来标记当前路由器认为这条流需要升速还是降速.

然而, 这些工作最大的问题依然是缺乏可部署性的问题. 正如在本节开头所述, 网络层的创新层出不穷, 但在互联网中能得到真正部署的却是凤毛麟角. 其主要原因正是修改网内设备难上加难. 上述方案[73-76] 都需要同时修改网内设备以及端主机设备. 这在实际场景中便十分困难, 因为端主机设备通常是内容提供商来维护, 如百度、阿里等公司; 而网内设备则是设备制造商来维护, 如华为、华三等公司. 同时协调两方进行修改以实现性能收益, 这在互联网发展的历史长河中早已被证明十分困难.

5 控制通路相关工作

如 1.1 节所述, 控制通路上主要包含反馈和决策 2 个处理模块. 其中, 反馈指的是从网络发生波动到端上感知到这一变化的过程, 决策指的是从端上最早可能感知到这一网络变化到最终发送速率等发送参数适配到新的链路状况这一过程. 近年来, 也有一系列工作分别试图优化反馈时间或缩短算法的决策时间. 如表 4 所总结的, 本节将分别就这 2 部分内容进行简要介绍.

Table 4 Related Work in Low-Latency Optimization from the Control Path

表 4 控制通路低延迟优化相关工作

学界/业界方案	解决方案	主要思路
DCQCN (SIGCOMM'15)[83] HPCC (SIGCOMM'19)[84] MACAdapt (MMSys'15)[85] Zhuge (SIGCOMM'22)[4]	反馈时间	尽早将网络信息的变化告知发送端
PiTree (MM'19)[86-87] Metis (SIGCOMM'20)[88] R-MPC (SIGCOMM'15)[23]	决策时间	尽早在网络信息发生变化后做出决策

5.1 缩短反馈时间

近年来, 在缩短反馈时间方面, 比较具有代表性的工作应该是 Zhuge[4]. 该工作针对无线网络最后 1 跳出现队列膨胀的现象, 通过将反馈信息编码到 TCP 确认包间隔等发送端可以不修改算法便能够直接读取信息, 除此之外, 也有一些工作类似地提出了在链路层等及时地将网络状态反馈给发送端来优化性能. 例如, MACAdapt 提出在点对点传输的音视频会议等场所第 1 跳出现瓶颈的时候, 尽早让发送端感知到网络状况发生了变化[85].

尽管针对实时多媒体传输的控制通路延迟优化的工作不多, 但在数据中心里, 倒是有一些工作试图缩减控制回路, 并取得了一定的影响力. 例如, DCQCN[83] 通过单独发送 QCN 帧, 可以将反馈回路缩减到小于 1 个往返时延. 类似地, 还有 HPCC[84] 等工作进一步对网络信息进行了精准反馈. 它们能够被部署并取得较大影响力的主要原因之一, 是在数据中心内, 网络管理员很容易对交换机和端主机一同修改.

5.2 缩短决策时间

近年来, 针对实时多媒体传输, 以及广义上的低延迟传输的决策算法过于复杂这一问题, 也有许多文章在这方面进行了深入研究.

一类工作尝试将已经提出的实时多媒体决策算法进行简化来缩短决策时间. 近年来, 随着基于整数规划、神经网络等的复杂算法在多媒体传输中的应用, 这些算法在决策时所需要的时间也越来越长. 例如, PiTree[86] 提出将基于整数规划、神经网络等复杂的多媒体传输中的码率调整算法转化为决策树, 以降低运行时开销. Metis[88] 则进一步在决策不稳定导致的延迟变长的问题上进行了分析, 并提供了解决方案. 同一团队的后续工作[87] 还进一步给出了理论分析, 提出了能够在性能不怎么损失的情况下简化决策算法、缩短决策时间的原因. Yin 等人[23] 也提出通过离线优化存储结果数据表、在线进行查表的方式来加速决策过程.

另一类工作则尝试通过实时决策来缩短决策的判断时间. 例如, 当带宽发生变化的时候, 传统算法如 BBR 算法[59] 需要在 6~8 个往返时延的反复测量、有足够信心后才会最终调整速率. 相比之下, 近年来的算法如 GCC[51] 可能在 1, 2 个数据帧的粒度上观测到变化便会对速率进行调整.

6 未来工作

实时多媒体是网络系统诞生数十年来经久不衰的研究课题, 但其面向的应用场景越来越复杂. 从网络电话到视频会议, 再到云游戏、远程手术, 最后到虚拟现实、增强现实等, 应用对网络的延迟需求越来越高, 场景也越来越多元化. 实时多媒体应用涉及一系列系统性的研究, 有些甚至都不仅仅是网络领域的科研问题, 未来还有 2 方面可以进一步探索.

1) 与操作系统进行联合优化. 随着边缘节点部署的逐渐推广以及新一代无线接入网技术, 如 WiFi6 及 5G 的规模化部署, 网络的净传播延迟已经越来越低. 这时, 在端侧的延迟瓶颈便逐渐凸显. 在操作系统等端领域当然有许多优秀的研究者也在试图降低这一延迟. 然而, 有时候, 在单一领域内数十年的优化会逐渐碰到该领域优化的瓶颈.

但需要注意的是, 网络延迟事实上是最有弹性的延迟组成部分: 网络总可以牺牲一部分吞吐来换取低延迟. 例如, 管理员可以通过调整冗余纠错编码参数的方式来控制因为丢包重传带来的尾部延迟变化. 因此, 如果能预感到端侧操作系统等出现延迟瓶颈时提前对全链路的延迟预算进行规划, 可以进一步降低延迟.

2) 与用户体验进行联合优化. 网络层目前的指标更多与网络服务质量相关. 即使卡顿率等指标事实上也是在应用层视频帧粒度进行统计的, 这也绝非是用户的真实体验, 而只是用户体验的估计. 更进一步地, 不同用户由于其生理、心理状态不同以及应用的不同, 对同一延迟、同一画质可能都会有不同的体验. 例如, 有的人视觉比较敏感, 对画面一点点的变化都可以细微地洞察出来. 有的人则不太敏感, 可能对 100 ms 以上的交互延迟都没办法感知.

如何了解到用户的真实体验, 并与此结合进行优化, 尤其是在这些新兴的应用场景逐渐进入人们视线之际, 同样是值得进一步深入研究的方向. 例如, 针对一部分用户, 由于其对延迟更加敏感, 则可以通过牺牲一部分画质清晰度来换取延迟的优化; 另一部分用户对画质更关注, 则反之牺牲延迟来换取画质提升, 达到整体上不同用户的体验均有提升的效果.

7 结 论

本文在充分调研、深入分析的基础上, 试图对近年来在实时多媒体传输的延迟优化上的研究进展进行综述. 首先, 分析了传统网络传输应用在实时多媒体应用中面临的挑战. 为解决上述挑战, 实时多媒体传输相应的协议栈应运而生. 其次, 总结了实时多媒体传输延迟优化的架构, 包括 2 个通路: 数据通路和控制通路. 1) 数据通路. 包含应用层、传输层和网络层带来的延迟抖动. 应用层重点围绕编解码、套接字缓冲区等部分的延迟进行了分析, 阐述了关键技术难题, 并梳理了进展. 传输层重点围绕拥塞控制和丢包恢复 2 个部分可能带来的延迟进行研究, 同样介绍了近期在这 2 方面上的研究进展. 网络层围绕主动队列管理、队列大小优化、端网消息传递等方面进行了综述. 2) 控制通路. 包含反馈和决策过程带来的延迟抖动. 本文围绕数据通路和控制通路这 2 方面分别介绍了现有的研究工作. 最后, 本文结合实时多媒体传输的研究现状, 提出了未来可能的研究方向, 希望能够对实时多媒体传输领域的研究人员带来一些启发.

作者贡献声明: 孟子立提出了框架并撰写论文; 徐明伟提出指导意见并修改论文.

参 考 文 献

[1] Cisco. VNI complete forecast highlights[EB/OL]. 2018[2023-08-

10].https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_Device_Growth_Traffic_Profiles.pdf

[2] Livingood J. Working latency — The next QoE frontier[EB/OL]. 2021[2023-08-10].https://blog.apnic.net/2021/12/02/working-latency-the-next-qoe-frontier/

[3] Mohan N, Corneo L, Zavodovski A, et al. Pruning edge research with latency shears[C]//Proc of the 19th ACM Workshop on Hot Topics in Networks. New York: ACM, 2020: 182−189

[4] Meng Zili, Guo Yaning, Sun Chen, et al. Achieving consistent low latency for wireless real-time communications with the shortest control loop[C]//Proc of the 36th ACM SIGCOMM Conf. New York: ACM, 2022: 193−206

[5] Kämäräinen T, Siekkinen M, Ylä-Jääski A, et al. A measurement study on achieving imperceptible latency in mobile cloud gaming[C]//Proc of the 8th ACM Multimedia Systems Conf. New York: ACM, 2017: 88−99

[6] Shi Shu, Cheng-Hsin H, Nahrstedt K, et al. Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming[C]//Proc of the 19th ACM Int Conf on Multimedia. New York: ACM, 2011: 103−112

[7] Wimmer R, Schmid A, Bockes F. On the latency of USB-connected input devices[C]//Proc of the 37th ACM CHI Conf. New York: ACM, 2019: 420−431

[8] Slivar I, Skorin-Kapov L, Suznjevic M. Cloud gaming QoE models for deriving video encoding adaptation strategies[C]//Proc of the 7th ACM Multimedia Systems Conf. New York: ACM, 2016: 185−196

[9] Li Tong, Zheng Kai, Xu Ke, et al. Tack: Improving wireless transport performance by taming acknowledgments[C]//Proc of the 34th ACM SIGCOMM 2020 Conf. New York: ACM, 2020: 15−30

[10] Meng Zili, Wang Tingfeng, Shen Yixin, et al. Enabling high quality real-time communications with adaptive frame-rate[C]//Proc of the 20th USENIX NSDI 2023 Conf. Berkeley, CA: USENIX Association, 2023: 1429−1450

[11] Wu Dapeng, Hou Yiwei Thomas, Zhang Yaqin. Transporting real-time video over the Internet: Challenges and approaches[J]. Proceedings of the IEEE, 2000, 88(12): 1855−1877

[12] Wu Dapeng, Hou Yiwei Thomas, Zhu Wenwu, et al. Streaming video over the Internet: Approaches and directions[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2001, 11(3): 282−300

[13] Huo Yongkai, Hellge C, Wiegand T, et al. A tutorial and review on inter-layer FEC coded layered video streaming[J]. IEEE Communications Surveys & Tutorials, 2015, 17(2): 1166−1207

[14] Zhang Jieliang, Jiang Weifang, Wang Xin, et al. Survey of video conference image quality assessment[C]//Proc of the 6th Chinese Guidance and Control Conf. Beijing: Publishing House of Electronics Industry, 2018: 207−211(in Chinese)
(张杰良, 蒋未芳, 王欣, 等. 视频会议画面质量评价方法综述[C]//第六届中国指挥控制大会论文集. 北京: 电子工业出版社, 2018: 207−211)

[15] Hu Xiaoyan, Tong Zhongqi, Xu Ke, et al. Video delivery over named data networking: A survey[J]. Journal of Computer Research and Development, 2021, 58(1): 116−136(in Chinese)
(胡晓艳, 童钟奇, 徐恪, 等. 命名数据网络中的视频传输研究综述[J]. 计算机研究与发展, 2021, 58(1): 116−136)

[16] Li Yang, Lin Hao, Li Zhenhua, et al. A nationwide study on cellular reliability: Measurement, analysis, and enhancements[C]//Proc of the 35th ACM SIGCOMM Conf. New York: ACM, 2021: 597−609

[17] Yang Xinlei, Lin Hao, Li Zhenhua, et al. Mobile access bandwidth in practice: Measurement, analysis, and implications[C]//Proc of the 36th ACM SIGCOMM Conf. New York: ACM, 2022: 114−128

[18] Yang Xinlei, Wang Xiaolong, Li Zhenhua, et al. Fast and light bandwidth testing for internet users[C]//Proc of the 18th USENIX NSDI Conf. Berkeley, CA: USENIX Association, 2021: 1011−1026

[19] Alizadeh M, Kabbani A, Atikoglu B, et al. Stability analysis of QCN: The averaging principle[C]//Proc of the 39th ACM SIGMETRICS Conf. New York: ACM, 2011: 49−60

[20] Dong Mo, Li Qingxi, Zarchy D, et al. PCC: Re-architecting congestion control for consistent high performance[C]//Proc of the 12th USENIX NSDI Conf. Berkeley, CA: USENIX Association, 2015: 395−408

[21] Abbasloo S, Yen C Y, Chao J. Classic meets modern: A pragmatic learning-based congestion control for the Internet[C]//Proc of the 34th ACM SIGCOMM Conf. New York: ACM, 2020: 632−647

[22] Mao Hongzi, Netravali R, Alizadeh M. Neural adaptive video streaming with pensieve[C]//Proc of the 31st ACM SIGCOMM Conf. New York: ACM, 2017: 197−210

[23] Yin Xiaoqi, Jindal A, Sekar V, et al. A control-theoretic approach for dynamic adaptive video streaming over HTTP[C]//Proc of the 29th ACM SIGCOMM Conf. New York: ACM, 2015: 325−338

[24] Cheng Yuchung, Cardwell N, Dukkipati N, et al. RFC8985: The RACK-TLP loss detection algorithm for TCP[EB/OL]. 2021[2023-08-10].https://datatracker.ietf.org/doc/html/rfc8985

[25] Nichols K, Jacobson V. Controlling queue delay[J]. Communications of the ACM, 2012, 55(7): 42−50

[26] Padhye J, Firoiu V, Towsley D F, et al. Modeling TCP Reno performance: A simple model and its empirical validation[J]. IEEE/ACM Transactions on Networking, 2000, 8(2): 133−145

[27] Ha S, Rhee I, Xu L. CUBIC: A new TCP-friendly high-speed TCP variant[J]. ACM SIGOPS Operating Systems Review, 2008, 42(5): 64−74

[28] Dasari M, Kahatapitiya K, Das S R, et al. Swift: Adaptive video streaming with layered neural codecs[C]//Proc of the 19th USENIX NSDI Conf. Berkeley, CA: USENIX Association, 2022: 103−118

[29] Zadtootaghaj S, Schmidt S, Sabet S S, et al. Quality estimation models for gaming video streaming services using perceptual video quality dimensions[C]//Proc of the 11th ACM Multimedia Systems Conf. New York: ACM, 2020: 213−224

[30] Yeo H, Jung Y, Kim J, et al. Neural adaptive content-aware internet video delivery[C]//Proc of the 15th USENIX NSDI Conf. Berkeley, CA: USENIX Association, 2018: 645−661

[31] Kim J, Jung Y, Yeo H, et al. Neural-enhanced live streaming: Improving live video ingest via online learning[C]//Proc of the 34th ACM SIGCOMM Conf. New York: ACM, 2020: 107−125

[32] Mukherjee D, Bankoski J, Grange A, et al. The latest open-source video codec VP9—An overview and preliminary results[C]//Proc of the 30th Picture Coding Symp (PCS). Piscataway, NJ: IEEE, 2013:

390−393

[33] Huang Teyuan, Johari R, McKeown N, et al. A Buffer-based approach to rate adaptation: Evidence from a large video streaming service[C]// Proc of the 28th ACM SIGCOMM Conf. New York: ACM, 2014: 187−198

[34] Yan F, Ayers H, Zhu Chenzhi, et al. Learning in situ: A randomized experiment in video streaming[C]//Proc of the 17th USENIX NSDI Conf. Berkeley, CA: USENIX Association, 2020: 495−511

[35] Sarker Z, Perkins C, Singh V, et al. RFC8888: RTP control protocol (RTCP) feedback for congestion control[EB/OL]. 2021[2023-08-10].https://datatracker.ietf.org/doc/html/rfc8888

[36] Schulzrinne H, Rao A, Lanphier R, et al. RFC7826: Real-time streaming protocol version 2.0[EB/OL]. 2016[2023-08-10].https://datatracker.ietf.org/doc/html/rfc7826

[37] Zhang Lei, Cui Yong, Pan Junchen, et al. Deadline-aware transmission control for real-time video streaming[C/OL]//Proc of the 29th IEEE ICNP Conf. Piscataway, NJ: IEEE, 2021[2023-08-10].https://ieeexplore.ieee.org/document/9651971

[38] Wang Zhou, Bovik A C, Sheikh H R, et al. Image quality assessment: From error visibility to structural similarity[J]. IEEE Transactions on Image Processing, 2004, 13(4): 600−612

[39] Hore A, Ziou D. Image quality metrics: PSNR vs SSIM[C]//Proc of the 20th Int Conf on Pattern Recognition. Piscataway, NJ: IEEE, 2010: 2366−2369

[40] Fouladi S, Wahby R S, Shacklett B, et al. Encoding, fast and slow: Low-latency video processing using thousands of tiny threads[C]// Proc of the 14th USENIX NSDI Conf. Berkeley, CA: USENIX Association, 2017: 363−376

[41] Fouladi S, Emmons J, Orbay E, et al. Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol[C]//Proc of the 15th USENIX NSDI Conf. Berkeley, CA: USENIX Association, 2018: 267−282

[42] Yeo H, Chong C J, Jung Y, et al. Nemo: Enabling neural-enhanced video streaming on commodity mobile devices[C]//Proc of the 26th ACM MobiCom Conf. New York: ACM, 2020: 363−376

[43] Spiteri K, Urgaonkar R, Sitaraman R K. BOLA: Near-optimal bitrate adaptation for online videos[C/OL]//Proc of the 35th IEEE INFOCOM 2016 Conf. Piscataway, NJ: IEEE, 2016[2023-08-10].https://ieeexplore.ieee.org/document/7524428

[44] Spiteri K, Sitaraman R, Sparacio D. From theory to practice: Improving bitrate adaptation in the DASH reference player[J]. ACM Transactions on Multimedia Computing, Communications, and Applications, 2019, 15(2): 1−29

[45] Li Zhi, Zhu Xiaoqing, Gahm J, et al. Probe and adapt: Rate adaptation for HTTP video streaming at scale[J]. IEEE Journal on Selected Areas in Communications, 2014, 32(4): 719−733

[46] Wang Cong, Rizk A, Zink M. SQUAD: A spectrum-based quality adaptation for dynamic adaptive streaming over HTTP[C]//Proc of the 7th ACM Multimedia Systems. New York: ACM, 2016: 1−12

[47] Sengupta S, Ganguly N, Chakraborty S, et al. HotDASH: Hotspot aware adaptive video streaming using deep reinforcement learning[C]//Proc of the 26th IEEE ICNP Conf. Piscataway, NJ: IEEE, 2018: 165−175

[48] Yi Gang, Yang Dan, Bentaleb A, et al. The ACM multimedia 2019 live video streaming grand challenge[C]//Proc of the 27th ACM Int Conf on Multimedia. New York: ACM, 2019: 2622−2626

[49] Shi Hang, Cui Yong, Qian Feng, et al. Dtp: Deadline-aware transport protocol[C]//Proc of the 3rd Asia-Pacific Workshop on Networking. New York: ACM, 2019: 1−7

[50] Winstein K, Sivaraman A, Balakrishnan H. Stochastic forecasts achieve high throughput and low delay over cellular networks[C]// Proc of the 10th USENIX NSDI Conf. Berkeley, CA: USENIX Association, 2013: 459−471

[51] Carlucci G, De Cicco L, Holmer S, et al. Analysis and design of the Google congestion control for web real-time communication (WebRTC)[C]//Proc of the 7th ACM Multimedia Systems Conf. New York: ACM, 2016: 133−144

[52] Zhu Xiaoqing, Pan Rong, Ramalho M, et al. RFC8698: Network-assisted dynamic adaptation (NADA): A unified congestion control scheme for real-time media[EB/OL]. 2020[2023-08-10].https://datatracker.ietf.org/doc/html/rfc8698

[53] Johansson I, Sarker Z. RFC8298: Self-clocked rate adaptation for multimedia[EB/OL]. 2017[2023-08-10].https://datatracker.ietf.org/doc/html/rfc8298

[54] Arun V, Balakrishnan H. Copa: Practical delay-based congestion control for the Internet[C]//Proc of the 15th USENIX NSDI Conf. Berkeley, CA: USENIX Association, 2018: 329−342

[55] Dong Mo, Meng Tong, Zarchy D, et al. PCC Vivace: Online-learning congestion control[C]//Proc of the 15th USENIX NSDI Conf. Berkeley, CA: USENIX Association, 2018: 343−356

[56] Holmer S, Shemer M, Paniconi M. Handling packet loss in WebRTC[C]//Proc of the 20th IEEE Int Conf on Image Processing. Piscataway, NJ: IEEE, 2013: 1860−1864

[57] Fong S L, Emara S, Li B, et al. Low-latency network-adaptive error control for interactive streaming[C]//Proc of the 27th ACM Int Conf on Multimedia. New York: ACM, 2019: 438−446

[58] Rudow M, Yan F, Kumar A, et al. StreamMelt: Efficient loss recovery for videoconferencing via streaming codes[C/OL]//Proc of the 20th USENIX NSDI Conf. Berkeley, CA: USENIX Association, 2023: 953−972

[59] Cardwell N, Cheng Yuchung, Gunn C S, et al. BBR: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time[J]. Queue, 2016, 14(5): 20−53

[60] Zaki Y, Pötsch T, Chen J, et al. Adaptive congestion control for unpredictable cellular networks[C]//Proc of the 29th ACM SIGCOMM Conf. New York: ACM, 2015: 509−522

[61] Sarolahti P, Kojo M, Raatikainen K. F-RTO: An enhanced recovery algorithm for TCP retransmission timeouts[J]. ACM SIGCOMM Computer Communication Review, 2003, 33(2): 51−63

[62] Bolot J C, Fosse-Parisis S, Towsley D. Adaptive FEC-based error control for Internet telephony[C]//Proc of the 39th IEEE INFOCOM Conf. Piscataway, NJ: IEEE, 1999: 1453−1460

[63] Padhye C, Christensen K J, Moreno W. A new adaptive FEC loss control algorithm for voice over IP applications[C]//Proc of the 19th IEEE IPCCC Conf. Piscataway, NJ: IEEE, 2000: 307−313

[64] Chen Ke, Wang Han, Fang Shuwen, et al. RL-AFEC: Adaptive

forward error correction for real-time video communication based on reinforcement learning[C]//Proc of the 13th ACM Multimedia Systems Conf. New York: ACM, 2022: 96−108

[65] Fong S L, Khisti A, Li B, et al. Optimal streaming codes for channels with burst and arbitrary erasures[J]. IEEE Transactions on Information Theory, 2019, 65(7): 4274−4292

[66] Krishnan M N, Shukla D, Kumar P V. Rate-optimal streaming codes for channels with burst and random erasures[J]. IEEE Transactions on Information Theory, 2020, 66(8): 4869−4891

[67] Floyd S, Jacobson V. Random early detection gateways for congestion avoidance[J]. IEEE/ACM Transactions on Networking, 1993, 1(4): 397−413

[68] Feng Wuchang, Kandlur D, Saha D, et al. BLUE: A new class of active queue management algorithms[EB/OL]. 1999[2023-08-10].https://www.cse.umich.edu/techreports/cse/99/CSE-TR-387-99.pdf

[69] Feng Wuchang, Kapadia A, Thulasidasan S. GREEN: Proactive queue management over a best-effort network[C]//Proc of the 21st Global Telecommunications Conf. Piscataway, NJ: IEEE, 2002: 1774−1778

[70] Long Chengnian, Zhao Bin, Guan Xinping, et al. The Yellow active queue management algorithm[J]. Computer Networks, 2005, 47(4): 525−550

[71] Spang B, Arslan S, McKeown N. Updating the theory of buffer sizing[J]. ACM SIGMETRICS Performance Evaluation Review, 2022, 49(3): 55−56

[72] Tang Jiaxin, Liu Sen, Xu Yang, et al. ABS: Adaptive buffer sizing via augmented programmability with machine learning[C]//Proc of the 41st IEEE INFOCOM Conf. Piscataway, NJ: IEEE, 2022: 2038−2047

[73] Katabi D, Handley M, Rohrs C. Congestion control for high bandwidth-delay product networks[C]//Proc of the 16th ACM SIGCOMM Conf. New York: ACM, 2002: 89−102

[74] Chia-Hui T, Zhu Jiang, Dukkipati N. Making large scale deployment of RCP practical for real networks[C]//Proc of the 27th IEEE INFOCOM Conf. Piscataway, NJ: IEEE, 2008: 2180−2188

[75] Flores M, Wenzel A, Kuzmanovic A. Enabling router-assisted congestion control on the Internet[C/OL]//Proc of the 24th Int Conf on Network Protocols (ICNP). Piscataway, NJ: IEEE, 2016[2023-08-10].https://ieeexplore.ieee.org/document/7784442

[76] Goyal P, Agarwal A, Netravali R, et al. ABC: A simple explicit congestion controller for wireless networks[C]//Proc of the 17th USENIX NSDI Conf. Berkeley, CA: USENIX Association, 2020: 353−372

[77] De Schepper K, Briscoe B, White G. RFC 9332: Dual-queue coupled active queue management (AQM) for low latency, low loss, and scalable throughput (L4S)[EB/OL]. 2023[2023-08-10].https://datatracker.ietf.org/doc/html/rfc9332

[78] Bai Wei, Chen Li, Chen Kai, et al. Information-agnostic flow scheduling for commodity data centers[C]//Proc of the 12th USENIX NSDI Conf. Berkeley, CA: USENIX Association, 2015: 455−468

[79] Alizadeh M, Yang Shuang, Sharif M, et al. pFabric: Minimal near-optimal datacenter transport[C]//Proc of the 27th ACM SIGCOMM Conf. New York: ACM, 2013: 435−446

[80] Addanki V, Apostolaki M, Ghobadi M, et al. ABM: Active buffer management in datacenters[C]//Proc of the 36th ACM SIGCOMM Conf. New York: ACM, 2022: 36−52

[81] Kwok-Ho C, Babiarz J, Baker F. RFC 4594: Configuration guidelines for diffserv service classes[EB/OL]. 2006[2023-08-10].https://datatracker.ietf.org/doc/html/rfc4594

[82] Appenzeller G, Keslassy I, McKeown N. Sizing router buffers[J]. ACM SIGCOMM Computer Communication Review, 2004, 34(4): 281−292

[83] Zhu Yibo, Eran H, Firestone D, et al. Congestion control for large-scale RDMA deployments[C]//Proc of the 29th ACM SIGCOMM Conf. New York: ACM, 2015: 523−536

[84] Li Yuliang, Miao Rui, Liu Hongqiang, et al. HPCC: High precision congestion control[C]//Proc of the 33rd ACM SIGCOMM Conf. New York: ACM, 2019: 44−58

[85] Chen Wei, Ma Liangping, Chien-Chung S. Congestion-aware MAC layer adaptation to improve video teleconferencing over wi-fi[C]//Proc of the 6th ACM Multimedia Systems Conf. New York: ACM, 2015: 130−141

[86] Meng Zili, Chen Jing, Guo Yaning, et al. PiTree: Practical implementation of abr algorithms using decision trees[C]//Proc of the 27th ACM Int Conf on Multimedia. New York: ACM, 2019: 2431−2439

[87] Meng Zili, Guo Yaning, Shen Yixin, et al. Practically deploying heavyweight adaptive bitrate algorithms with teacher-student learning[J]. IEEE/ACM Transactions on Networking, 2021, 29(2): 723−736

[88] Meng Zili, Wang Minhu, Bai Jiasong, et al. Interpreting deep learning-based networking systems[C]//Proc of the 34th ACM SIGCOMM Conf. New York: ACM, 2020: 154−171

Meng Zili, born in 1999. PhD candidate. His main research interest includes real-time video transmission.

孟子立, 1999 年生. 博士研究生. 主要研究方向为实时多媒体传输.

Xu Mingwei, born in 1971. PhD, professor. Member of CCF. His main research interest includes Internet architecture.

徐明伟, 1971 年生. 博士, 教授. CCF 会员. 主要研究方向为互联网体系结构