# 基于多路径QUIC的视频流传输优化

包嘉琪,胡　畅,赵国强,黄俊杰,崔傲叠

(武汉纺织大学 计算机与人工智能学院,湖北 武汉 430200)

**摘　要:** 视频流应用场景虽然广泛,但存在传输质量低、视频质量较差的问题。为此,利用软件定义网络(SDN)对网络的可编程特性结合快速UDP Internet连接(QUIC),设计了一种针对视频流的基于SDN的分段路由多路径QUIC传输框架(SDNMQS)。该框架可根据网络条件动态分配路由,实现高质量传输服务。在仿真平台的模拟试验结果表明,SDNMQS在各项性能测试上优于其他传统方法,能提升视频传输质量,可为优化视频传输提供参考与借鉴。

**关键词:** QUIC;软件定义网络;分段路由;多路径传输;视频流

**DOI:** 10. 11907/rjdk. 222216　　**开放科学(资源服务)标识码(OSID):**

**中图分类号:** TP18　　**文献标识码:** A　　**文章编号:** 1672-7800(2023)009-0138-09

# Multi-path QUIC-based Video Streaming Transmission Optimization

BAO Jiaqi, HU Chang, ZHAO Guoqiang, HUANG Junjie, CUI Aodie

(*School of Computer and Artificial Intelligence*, *Wuhan Textile University*, *Wuhan* 430200, *China*)

**Abstract:** Video streaming application scenarios are widespread and common, but there are problems with low transmission quality and poor video quality. To this end, a segmented routing multi-path QUIC transmission framework (SDNMQS) for video streams was designed using the programmable characteristics of software defined network (SDN) and fast UDP Internet connection (QUIC). This framework can dynamically allocate routes based on network conditions, achieving high-quality transmission services. The simulation test results on the simulation platform show that SDNMQS outperforms other traditional methods in various performance tests and can improve the quality of video transmission, providing reference for optimizing video transmission..

**Key Words:** QUIC; software defined network; segmented routing; multipath transmission; video streaming

## 0　引言

2020年Cisco的年度互联网报告显示,视频流是互联网流量的主要部分,预计占2022年全球视频总流量的82%[1]。5G的快速发展,为网络创造了高带宽、低时延的环境,但由于多种因素影响,如何保证良好的用户体验质量(Quality of Experience, QoE)仍是一项具有挑战性的任务,尤其是对传输QoE敏感的应用,例如音视频传输、在线游戏等。

MPTCP(Multipath TCP)作为TCP的扩展,被广泛运于不同领域,是目前主流的传输协议。该协议可聚合多条路径的带宽提升网络应用程序性能。在实际传输过程中,TCP为了保证数据的可靠性会按照顺序传输数据,当某一个数据包过大时,后面的数据需要等待该数据成功传输至应用层后才能组装成完整的数据,也可能由于某一个数据包因网络中断或传输过程中丢失,只能在重新传输后才能继续后续步骤,然而一旦发生上述问题将导致视频流传输发生明显卡顿,影响用户观看体验。因此,TCP不适用于能容忍少量丢包的流媒体服务。

此外,MPTCP的使用前提需要用户设备配备多宿主接口,系统内核需要更新以支持MPTCP,对于流媒体服务而言部署代价较大。为此,本文提出了一种新的传输协议快速UDP Internet连接(Quick UDP Internet Connection, QU-

**收稿日期:** 2022-10-17

**基金项目:** 包嘉琪(1996-),女,武汉纺织大学计算机与人工智能学院硕士研究生,研究方向为网络通讯;胡畅(1997-),男,武汉纺织大学计算机与人工智能学院硕士研究生,研究方向为机器学习;赵国强(2000-),男,武汉纺织大学计算机与人工智能学院硕士研究生,研究方向为软件开发;黄俊杰(1984-),男,博士,武汉纺织大学计算机与人工智能学院讲师、硕士生导师,研究方向为网络通讯;崔傲叠(1998-),男,武汉纺织大学计算机与人工智能学院硕士研究生,研究方向为计算机图形学。本文通讯作者:黄俊杰。

IC)，设计了一种针对视频流的基于SDN的分段路由多路径QUIC传输框架(SDNMQS)，以提高流媒体的传输性能。

## 1　相关研究

### 1.1　QUIC

QUIC相较于TCP进行了多项改进，例如基于UDP设计了多路复用的操作，不存在队头阻塞情况。如图1所示，在一条QUIC连接上可发送多个请求(Stream)，Stream间相互独立，Stream2丢失了一个Pakcet，不会影响Stream3、Stream4，如此可有效解决视频卡顿问题，提升音视频资源的访问效率[2]。

QUIC还有一个显著特征是0RTT连接，如图2所示。QUIC从请求连接到正式接发送HTTP数据一共消耗1个往返时延(Round-Trip Time，RTT)以获取Server Config，后续连接如果客户端缓存了Server Config就可直接发送HTTP数据，建立0RTT连接，对于视频传输而言首帧更快、延迟更小。此外，QUIC属于用户态协议，无需更新内核即可进行修改，进一步提升了QUIC的可扩展性，还可灵活调整可靠传输机制、拥塞控制算法等。

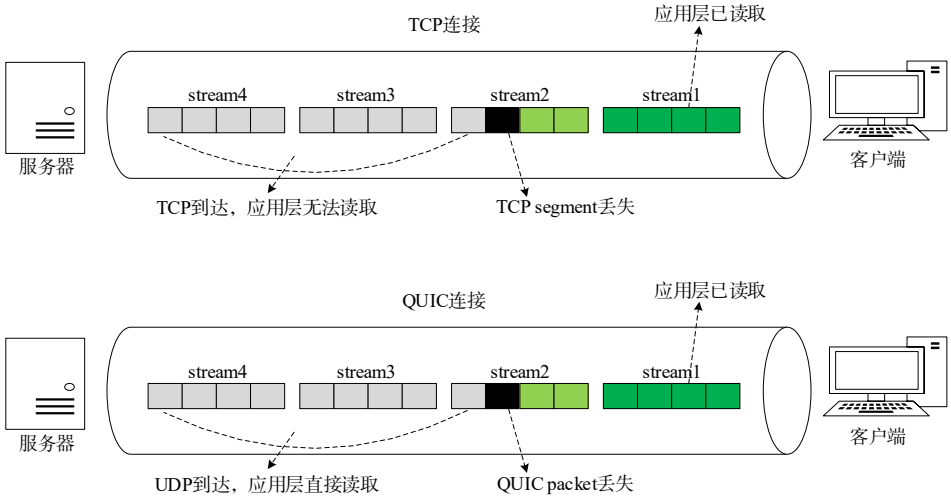



**Fig. 1　TCP head-of-line blocking and QUIC multiplexing**

**图1　TCP队头阻塞和QUIC多路复用**

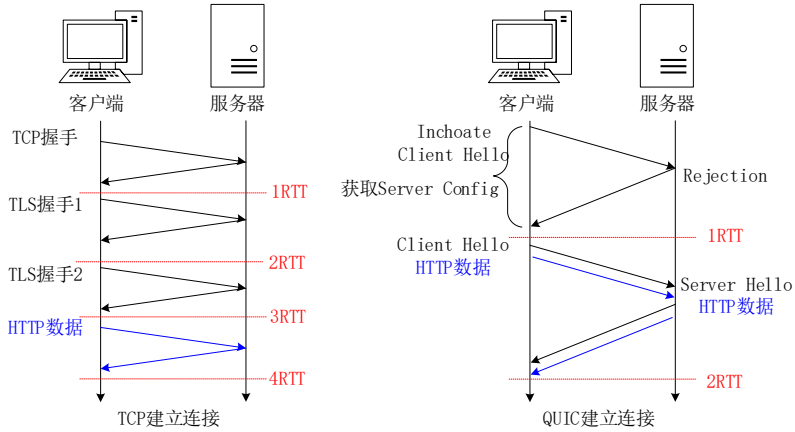



**Fig. 2　TCP connection establishment and QUIC connection establishment**

**图2　TCP建立连接和QUIC建立连接**

2022年，IETF QUIC和HTTP工作组成员Robin Mark在推特上正式发布基于QUIC协议的HTTP/3，标志着QUIC将有可能为互联网数据传输打开新局面。多路径QUIC(Multipath QUIC，MPQUIC)通过多个网络路径组成一个QUIC连接传输数据，但协议标准目前只规定了多路径实现的机制，而多路径调度需要额外单独实现。因此，如何合理调度多路径QUIC数据包，为用户提供一个高带宽、低时延的网络环境仍需时间进行解决。

由于新型网络架构软件定义网络(Software Defined Network，SDN)具有灵活性、可管理性及对流量需求快速变化的响应性[3]。因此，在多路径QUIC传输中根据网络条件，可通过SDN控制器合理分配每个MPQUIC子流，但在多路径场景中多媒体流被划分为多个子流，将每个子流视为一个独立数据流并在数据平面单独传递，将显著增加存储在SDN交换机转发条目的数量及SDN控制器和交换机的负载。为了解决该问题，Filsfils等[4]提出一种源路由

方法——分段路由(SR),将路由信息以有序的标签列表形式编码到数据包头部中,以此极大减少网络节点中的转发规则数量。

综上,针对用户在线观看视频的应用场景,本文设计了一种具有分段路由的软件定义网络多路径QUIC传输框架(SDNQMS),为视频流提供高效的路由分配,并提出多路径QUIC路由算法,通过分段路由合理分配路径以提升网络吞吐量和视频传输质量,在SDN网络中利用MPQUIC和分段路由为用户提供高质量的视频流传输。

从流量工程(Traffic Engineering, TE)角度而言,采用集中式SDN控制器可根据可用网络资源智能分配多条不相交的路径以平衡流量负载,提升网络资源利用率。从用户体验质量角度而言,根据实时网络条件动态控制子流、分配路径,能有效减少数据包混乱的情况,提升了视频传输的质量。

此外,为了优化视频内容传输链,提升终端用户体验质量(Quality of Experience, QoE),学术、工业界专家在SDN技术、多路径传输和QUIC在传输领域进行了许多研究,取得了一定的效果。

### 1.2 SDN技术与多路径传输

Sandri等[5]提出一种在支持OpenFlow网络中使用MPTCP的方案,以保证相同MPTCP连接的子流通过不相交的路径发送。Zannettou等[6]提出一个MPTCP感知的SDN控制器,使用数据包检测为路径提供确定性子流分配,并优化了MPTCP子流的替代路由机制。Nam等[7]使用SDN动态添加、删除MPTCP路径,以减少大量无序数据包,提升了自适应速率视频流的下载速度和用户体验质量。Duan等[8]提出一种响应式的MPTCP系统,采用集中式SDN控制器计算子流的转发路径,并在每台服务器上运行监视器来主动调整子流数量。

虽然上述方法均提升了网络吞吐量,但在SDN交换机上安装了更多的流规则,从而增加了SDN交换机的负载,并且根据网络变化情况缺乏有效的路由和自适应机制控制子流数量。实际上,基于SDN的分段路由可有效管理资源,并在网络中提供更好的流量工程(Traffic Engineering, TE)解决方案[9]。例如,李艺等[10]提出在软件定义网络中基于分段路由的多路径调度算法,有效提升了网络吞吐量,降低了传输时延,同时具有较低的流表项存储开销。

### 1.3 QUIC传输优化

国内外关于QUIC传输优化的研究较多,但大部分与数据包调度相关。Vu等[11]开发了MPQUIC流复用器与MPQUIC调度器,利用选择性多路径冗余控制实时视频流的尾部损失延迟。Mogensen等[12]添加了具有严格优先级的选择性数据复制来扩展MPQUIC传输协议,以支持使用无线5G多UE设备的关键应用,相较于单路径连接和最新MPQUIC协议传输延迟更低。Rabitsch等[13]认为MPQUIC与MPTCP的区别在于数据传输粒度不同,调度器设计应考虑流完成时间,受MPTCP ECF调度器[14]启发设计了流级调度器,减少了页面加载时间(Page Load Time, PLT)的流完成时间。Shi等[15]设计了一个考虑路径带宽的流级调度器,基于优先级机制同时传输多个流。Wang等[16]通过ACK快速响应来减少PLT,及时恢复丢失的数据。文献[14-16]虽然在网络应用中减少了PLT,但调度器不适合视频流场景。为此,李炫杉等[18]针对QUIC应用在实时通信场景下的优缺点改进QUIC协议,使QUIC更适应实时通信场景,并自适应优化CUBIC拥塞控制算法,对改进后的QUIC协议进行应用与验证。

近年来,国内大型互联网公司对QUIC进行了研究与应用,快手公司在2019年结合自身业务特点设计了KQUIC算法[19],针对短视频场景对QUIC进行一系列优化,但技术细节尚未公布。腾讯公司的腾讯云云计算负载平衡业务目前已支持使用QUIC协议[20],这也是国内首家支持QUIC协议的云厂商。

## 2 系统框架设计

### 2.1 系统总体框架

本文系统框架由管理平面与数据平面组成,基于SDN的多路径QUIC视频流传输框架(SDNMQS)如图3所示。该框架为视频流服务提供了有效的管理,并提升了用户观看视频的质量。框架并未将子流路径作为转发规则安装在SDN的交换机中,而使用分段路由方法将有序的分段列表配置到入口交换机的转发表,以根据不断变化的网络条件为SDN网络提供多路径保护和动态链路恢复机制,具体工作流程如图4所示。

### 2.2 控制平面

控制平面通过SDN的扩展Ryu控制器实现,所有MPQUIC数据包流量均由Ryu控制器控制。Ryu控制器由MPQUIC流量管理器、分段路由模块、网络资源管理、数据库模块、网络拓扑收集器和配置模块组成。

#### 2.2.1 网络拓扑收集模块

该模块主要监控网络状态,负责计算链路带宽和时延,特别当一个链路或节点发生故障时将这些网络信息报告给SDN控制器,以便立即采取操作(例如恢复故障链路)。

#### 2.2.2 MPQUIC流量管理器

该模块计算最短且连接数最小的路径,然后分配子流路径。为了最大限度减少链路拥堵对数据传输造成的质量影响,MPQUIC模块采用接入控制,只有所分配的子流速率之和不超过链路容量,MPQUIC子流才能在每个链路上被允许传输。与此同时,MPQUIC模块动态控制在入口源节点产生子流数量,并根据收集到的链路信息,将资源分配给子流路径,以满足用户观看体验。

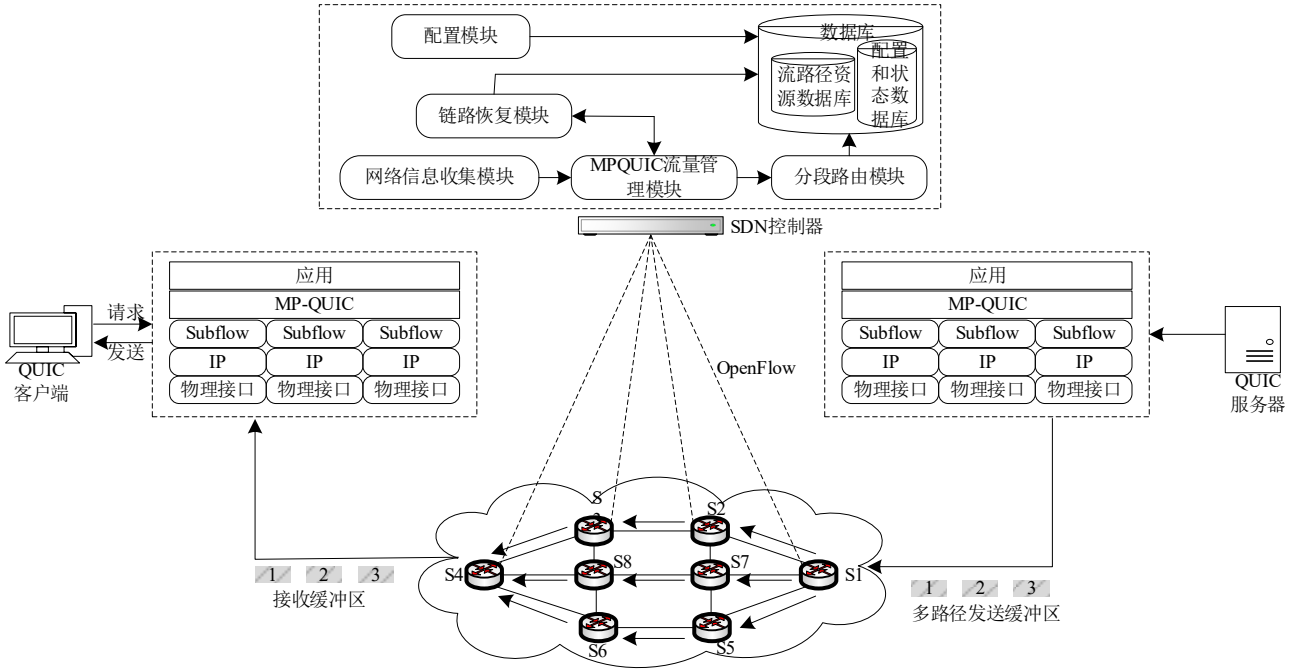



**Fig. 3　SDN-based multi-path QUIC video streaming transmission framework**

**图3　基于SDN的多路径QUIC视频流传输框架**

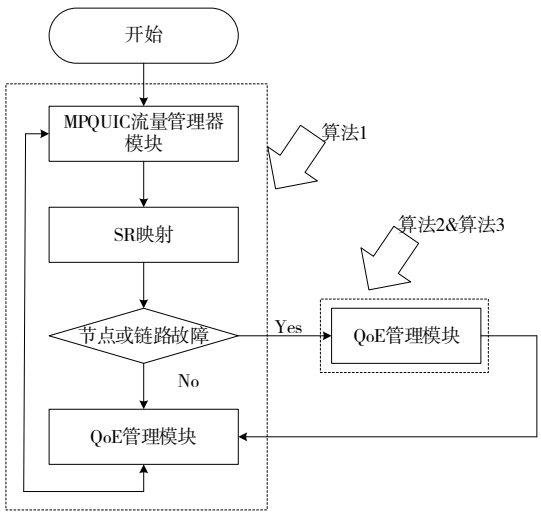



**Fig. 4　SDNMQS framework workflow**

**图4　SDNMQS框架工作流程**

2.2.3　分段路由模块

控制器将路径计算模块计算的子流路径映射到SR路径。本文使用文献[10]提出的算法,在等价最优路径集合中寻找一条SR段(SID)数目最少的路径,作为该网络流的传输路径。例如,给定一个MPQUIC客户端请求的视频流 $f$,入口节点为交换机 $S1$,出口节点为交换机 $S4$,那么子流 $sf1$ 与中间节点 $\{S2,...,SN-1\}$ 的完整路径为 $P_{sf1}=\{S1\rightarrow S2\rightarrow S3\rightarrow S4\}$,如图5所示。

该算法将路径与拓扑结构图作为输入,映射子流的具体路径,并将返回段序列作为指定SR路径的输出。具体为,考虑子流路径的入口节点 $S1$、出口节点 $S4$,如果只存在一条最优路径并且等于从 $S1$ 到 $S4$ 的子流路径,入口节点将作为从 $S1$ 到 $S4$ 的SID节点,这个子流路径到SR路径的映射结束(见图5);当发生映射的最优路径不等于子流路径或存在多条等价最优路径时,可考虑其他节点段。

2.2.4　配置模块

该模块为网络资源设置提供接口,还为终端用户的QoE配置提供接口,例如多媒体应用流规则、吞吐量、丢包。

2.2.5　数据库模块

该模块将网络配置参数与监测状态存储在数据库中,将条目设置在预定时间内过期,以便路径随流量分布和链路故障变化而刷新。当SDN控制器接收到一个新的MPQUIC连接子流时,该模块首先在数据库中查询与该MPQUIC连接相对应的现有路径,如果路径存在,该子流将被分配到同一MPQUIC连接先前分配的子流路径中的一个特定路径,否则将使用路径计算模块为该子流计算新路径。同时,新路径将存储在数据库中,便于后续被同一MPQUIC连接的子流使用,以进一步降低SDN控制器的CPU利用率。

2.2.6　网络资源管理模块

该模块实现了流媒体应用相关的资源分配。为了确保系统整体性能,该模块既测量延迟、抖动、网络吞吐量和丢包率等,还执行链路故障检测和恢复机制,以确保配置、恢复SDN网络中的任何故障点,从而不影响终端用户的观看质量。

**2.3　数据平面**

数据平面由支持SR技术的SDN交换机组成,使用有线连接或公共无线信道进行互联,交换机分为加载交换机

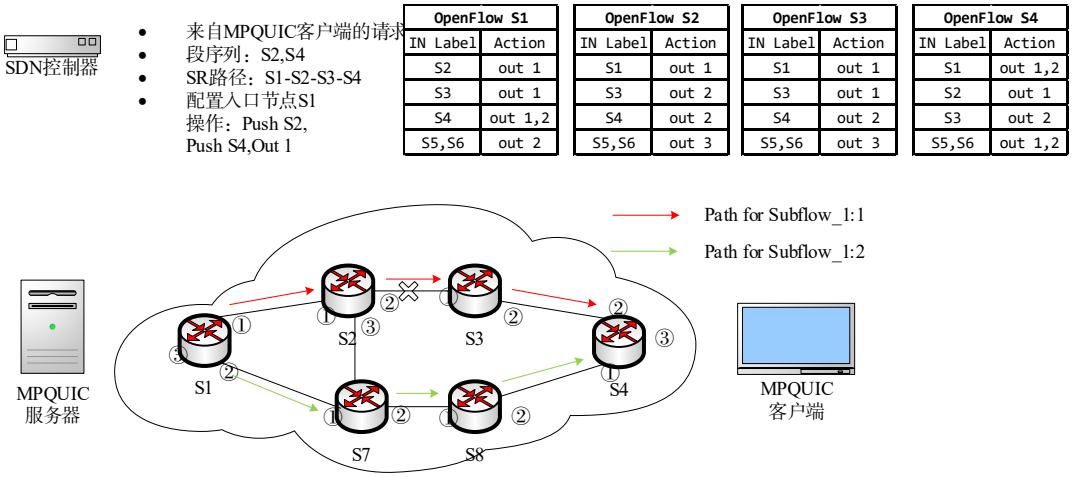



**Fig. 5 Example of SDN-based segmented routing operation**

**图5 基于SDN的分段路由操作示例**

和转发交换机。加载交换机即SR路径中的入口交换机，负责将段序列加载到数据包中；转发交换机只根据SR数据包中的头字段转发数据包。

图5为基于SDN的分段路由操作示例。当MPQUIC客户端向MPQUIC服务器发送请求时，SDN控制器基于网络状况（链路延迟和带宽）计算传输路径 $P_{sf1}=\{S1\rightarrow S2\rightarrow S3\rightarrow S4\}$，该路径被映射到入口交换机$S1$并存储在数据包报头中，段序列仅由标识SR域目标节点的标签组成，即$SL=\{S4\}$。然后，SDN控制器根据该段序列标签配置入口交换机$S1$的转发表，一个子流的数据包通过中间节点从$S1$转发至$S4$，当数据包到达中间节点时，顶部标签被弹出。接下来，使用代表下一个标签的段节点将数据包转发至目的点，当网络节点或链路发生故障时，SR可支持并提供基于SDN网络的动态流量恢复。

例如，当链路$S2\rightarrow S3$发生故障时，网络拓扑收集模块可通过备份路径$\{S1\rightarrow S2\rightarrow S7\rightarrow S8\rightarrow S4\}$重定向流量，在节点S2弹出顶部标签，并通过其相邻的SID 3将数据包传输至$S7$。在$S7$，数据包使用接口2的邻接SID沿着路径达到目的地。

### 2.4 SDN网络的多路径QUIC分配算法

在用户在线观看视频的应用场景下，利用SDN和多路径QUIC的优势设计算法1，该算法通过多条不相交路径来路由网络流量，提升网络资源利用率，保障用户观看视频的质量。

**算法1** 多路径QUIC路由算法

输入：网络拓扑$G=(V,E)$，视频流$f$。

输出：段序列$SL$。

1.根据式(1)和带宽计算链路权重；

2.找出网络中所有子流的最短路径$p\in P$；

3.**Foreach** $p\in Psrc\rightarrow dst$；

4.**if** $p.used+sf_k\cdot B_{sf}^k<b_{ij}$ **then return** $p$；

5.**else Go to step** 3；

6.执行子流路径$p_{sf}$到分段路由的映射；

7.将路径$p$及其相关的$SL$保存在$DB_p$中；

8.开始对MPQUIC子流进行传输；

9.当链路$(i,j)$发生故障时，转至算法2；

10.**if** $f_{new}$为一个新流 **then** 在$DB_p$查询$f_{new}$的子流路径；

11.**if** $pf_{new}$不在$DB_p$中，则转到步骤2并发出$f_{new}$的路径添加请求；

12.**end if**；

13.**end if**；

14.**end for**；

15.只要$B_{sf}^k<b_{ij}$，继续传输；

16.使用$l_c$和拥塞指数避免拥塞.

网络拓扑图$\boldsymbol{G}=(\boldsymbol{V},\boldsymbol{E})$，$\boldsymbol{V}$为网络中的节点集合(交换机)，$\boldsymbol{E}$为节点间边的集合(链路)。每个$e\in\boldsymbol{E}$均与一个非负的整数链路权重相关，表示为$W(e)$。在现实网络场景中，链路权重为一个非固定参数，受链路带宽、丢包、延迟和链路利用率影响，本文利用这些信息计算从源到目的地的最短路径最优集合，通过$b_w$表示一对节点的可用链路带宽，$B_{sf}^k$表示MPQUIC的一个子流$sf$连接到$k$所需的带宽，流量需求$f$从源$s\in\boldsymbol{V}$到目的地$t\in\boldsymbol{V}$的最短路径表示为$p(s,t)$。

本文将链路$(i,j)$权重$We_{ij}$定义为链路延迟$dl_{ij}$和丢包率$pl_{ij}$值的加权和，$\beta$为比例因子，当$\beta$较大时路由对丢包率更敏感，反之路线选择对延迟更敏感。

$$We_{ij}=(1-\beta)\mathrm{dl}_{ij}+\beta\mathrm{pl}_{ij}\quad 0\leqslant\beta\leqslant 1 \tag{1}$$

为了得到最小成本路由MPQUIC子流路径，本文定义了一个目标函数，如式(2)所示。传输连接$k$的MPQUIC子流选择最短路径中的每条链路$(i,j)$，通过最小成本优化视频质量。

$$O_f=\min\sum_{(i,j)\in E}\sum_{k\in K}\left(We_{ij}sf_{ij}^k\right) \tag{2}$$

通常情况下，一个视频流$f$由子流$\{sf_1,sf_2,\ldots,sf_N\}$组成，

为了路由MPQUIC连接的子流，本文首先根据链路权重和网络情况寻找网络中端点间的最短路径。在真实网络运行环境中，当一条链路无法提供所需资源时可能出现瓶颈问题。为此，本文引入链路临界值$l_c$参数解决该问题。假设$\boldsymbol{P}_{st}$为MPQUIC服务器到MPQUIC客户端前$h$条的最短路径集合，$P_{st}(c)$为链路$e \in \boldsymbol{E}$被包含在前$h$条最短路径中的次数，对于任何一对通信节点，链路$e$在前$h$条最短路径中的出现率可计算为$P_{st}(c)/h$，链路$e$的总预期负载可表示为SDN系统中连接源和目的地所有可能路径对该链路预期流量需求数之和。

$$l_c = \sum_{(s,t) \in TD} P_{st}(c)/h \tag{3}$$

式中：$TD$为每个时间$T$内记录并存储在数据库中的流量需求。

本文通过控制器更新每个时间$T$后的链路临界值，使网络负载均衡。首先，通过链路约束条件计算最短路径；然后，控制器将这些MPQUIC子流映射并分配到分段路由的路由表中；最后启动MPQUIC连接，在视频流传输期间报告QoE指标。

### 2.5　SDN网络中链路恢复算法

在传统网络中，当检测到链路故障时通常采用回退路由方法，即流量被退回到源节点，然后从源节点再次转发到目标节点。由于路径长度较长，故障检测窗口随之增加，回退路由增加了网络恢复时间。为避免该问题，本文使用了SR的路径保护和链路恢复机制新方法，如图6所示。

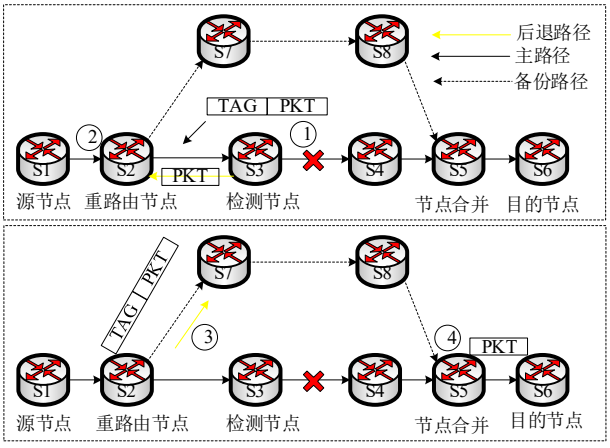



**Fig. 6　Link recovery example**

**图6　链路恢复示例**

当链路故障时，检测到链路故障的节点重新路由数据包（见图6步骤①），直至寻找到能将数据包转发至目的节点的新路由路径。该方法的创新之处在于：首先标记MPQUIC流的相同数据包（例如使用包含故障链路信息的SR标签）；然后通过主路径发回，在接收到标记的数据包后（见图6步骤②）重新路由节点（例如S2）；最后将标记的数据包转发到其目的节点（见图6步骤③）。

当重路由节点处理第一个标记的数据包时，在OpenFlow交换机中执行状态转换，创建故障转移表并安装备份操作，其中来自源节点的所有后续数据包在重新路由的节点上转发（见图6）。该方法的优点是减少了路径成本、网络恢复时间和备份路径长度。

为了保障在链路故障后快速恢复，必须使用重新计算的新路由路径绕过故障链路$l_{ij}$。

$$X_{s,t}^{p_i} = 0, \forall p \in \boldsymbol{N} \tag{4}$$

式中：$\boldsymbol{N}$为链路$(i,j)$故障后重新计算的最短路径集合；$P_i$为第$i$条重路由路径$(1 \leqslant i \leqslant n)$；$n$为链路$(i,j)$故障后重新计算的路由路径数量。

重新计算的路径还应存在一个循环避免约束，如式(5)所示。

$$\sum_{t \in V,(s,t) \in E} \left(X_{s,t}^{p_i}\right) \leqslant 1, \forall s \in \boldsymbol{V} \tag{5}$$

当SDN网络发生故障时，分配给重路由路径$P_i$的中断MPQUIC流的带宽不能超过路径$P_i$的可用容量。当链路$(i,j)$发生故障时，使用算法3修改、更新网络拓扑。算法2将更新后的网络拓扑作为输入，并根据公式(1)重新计算链路成本。SDN控制器计算MPQUIC流从$s$传输到$t$的最短路径，然后将MPQUIC子流路径$p_{sf}$映射到SR路径。当客户端和服务器建立MPQUIC连接时，根据用户服务需求启动子流传输，控制器监控系统事件和网络拓扑状态。具体流程如算法2所示。

**算法2**　故障链路恢复算法

输入：New_topology $G' = (V', E')$。

输出：段序列标签$SL'$。

1.使用算法3删除故障链路$(i,j)$

2.找到网络中所有子流最短路径$p' \in P'$

3.**foreach** $p' \in P'$ do

4.**if** $p.used + sf_k \cdot B_{sf}^k < b_{ij}$ **then return** $p'$

5.**else Go to** step 3

6.**end if**

7.将所有子流最短路径$p'_{sf}$映射到$SR'$路径

8.**Go to** step 6 in 算法1 **end for**

**算法3**　从网络拓扑中删除故障链路的脚本

```
1.def removeLink(self,node1,node2,port1=None,port2=None,
**opts)
2.if not opts and self.lopts:then
3.self.addPort(node1,node2,port1,port2)
4.key = tuple(self.sorted([node1,node2]))
5.self.link_info[key] = opts
6.self.g.remove_edge(*key)
7.return key
```

## 3.1　实验环境搭建与性能分析

### 3.1　实验环境

为了证明本文框架的可行性，使用3台Linux系统

(Ubuntu V18.04 LTS)虚拟机作为实验平台。其中一台安装了 Mininet2.3.0、Ryu4.1.8 控制器和 OpenSwitch2.4.0,用于模拟 SDN 环境,另外两台作为 MPQUIC 服务器连接到 Mininet 网络上,如图 7 所示。

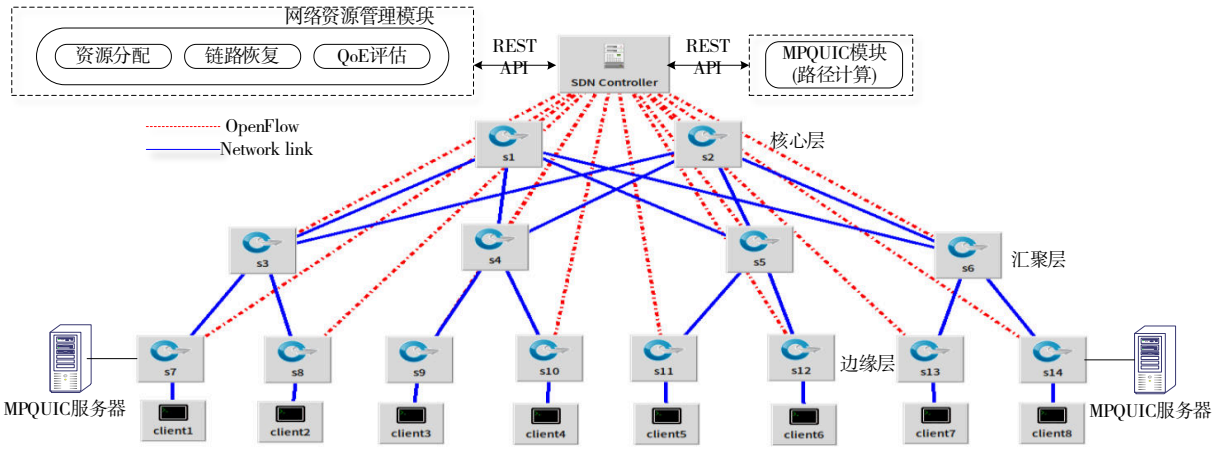



**Fig. 7 Experimental environment**

**图 7 实验环境**

由图 7 可见,Mininet 网络拓扑结构包括边缘层、聚合层和核心层 3 个层级,分别由 8、4、2 个 OpenFlow 交换机组成,并且边缘层中每个交换机均接入一个 MPQUIC 客户端,同时将 AStream 作为客户端的播放器。AStream 是通过 Python 编写的一个开源虚拟播放器,可模拟视频播放过程并收集 QoE 结果,例如播放比特率、回退和质量切换。

### 3.2 实验数据集

本文选用 Blender 基金会制作的著名开源动画电影 Big Buck Bunny 作为测试视频,通过多媒体视频处理工具 ffmpeg 5.0.1 的 libx265 以 1080 p、720 p、480 p 进行编码,视频编码率分别为 3.473 Mb/s、2.496 Mb/s 和 1.536 Mb/s,并且在路径管理器的配置上限制每个 MPQUIC 连接只能有 3 个 MPQUIC 子流,实验设置的链路参数如表 1 所示。

视频从 MPQUIC 服务器到 MPQUIC 客户端重复传输 50 次,将本文框架与 MPQUIC 在传统 SDN 下的传输方式(尽力而为的方式),在吞吐量、故障链路恢复时间和终端用户视频接收质量方面进行比较。

**Table 1 List of network link parameters**

**表 1 网络链路参数列表**

| 层级 | 链路带宽/Mb/s | 链路时延/ms | 丢包率/% |
|---|---|---|---|
| 边缘层 | 3 | 15 | 1 |
| 汇聚层 | 2 | 20 | 3 |
| 核心层 | 3 | 30 | 3 |

### 3.3 实验结果与分析

#### 3.3.1 系统吞吐量

50 次传输后所取得的平均吞吐量如图 8 所示,具体数据如表 2 所示。由此可见,本文框架吞吐量均高于 MPQUIC 在传统 SDN 下的传输方式,原因为在传输视频过程中 SDNMQS 通过多条不相交的最优路径进行传输,并在数据平面使用无需任何路径信令的分段路由(SR),因此可选择最有效的视频流转发路径。

#### 3.3.2 故障链路恢复时间

在比较故障链路恢复时间时,将本文提出的故障链路

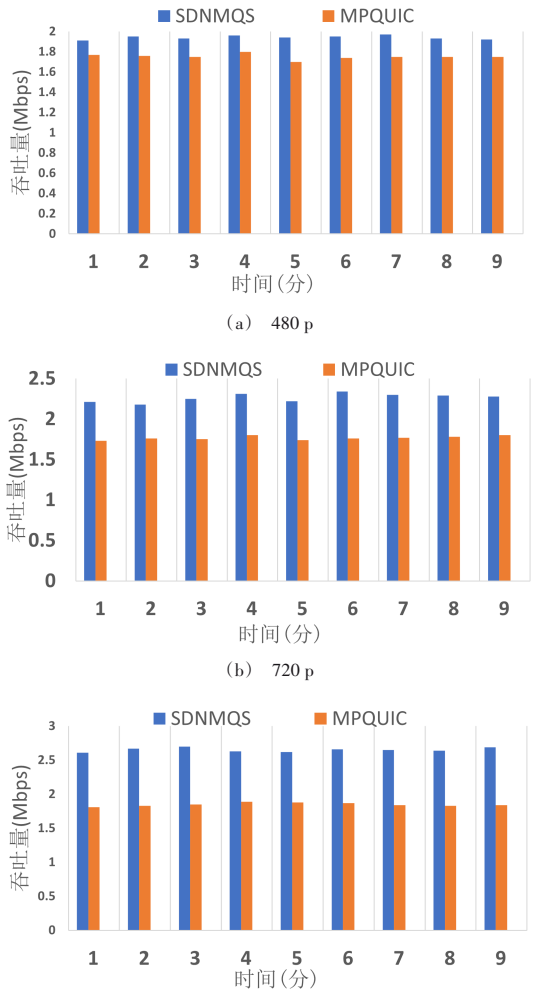



**Fig. 8 Throughput of different resolutions**

**图 8 不同分辨率的吞吐量**

**Table 2　Average throughput**

**表 2　平均吞吐量**　　**(Mb/s)**

| 视频分辨率/p | SDNMQS | MPQUIC |
|---|---|---|
| 480 | 1.94 | 1.75 |
| 720 | 2.22 | 1.76 |
| 1 080 | 2.65 | 1.87 |

算法与基于OpenFlow协议组表恢复方法进行比较,实验结果如图9所示。当链路{$S1 \rightarrow S4$}发生故障时,由于链路{$S1 \rightarrow S4$}属于核心层与汇聚层,交换机$S1$与SDN控制器的直接连接可极大缩短故障链路恢复时间。当链路{$S5 \rightarrow S11$}发生故障时,由于链路{$S5 \rightarrow S11$}属于汇聚层和边缘层,因此故障恢复时间最大。由于OpenFlow方法恢复故障只在本地有效,所有流量必须转发到故障点才能触发故障恢复组表功能,因此延迟时间过长。本文所提算法相较于基于OpenFlow协议组表恢复方法降低了流表数量,控制器及时选择最优路径进行转发流量,缩短了传输时间。

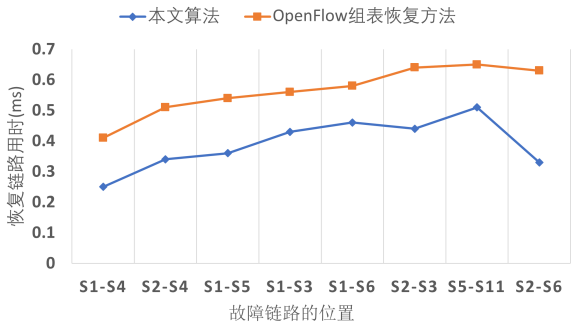



**Fig. 9　Fault link recovery time**

**图9　故障链路恢复时间**

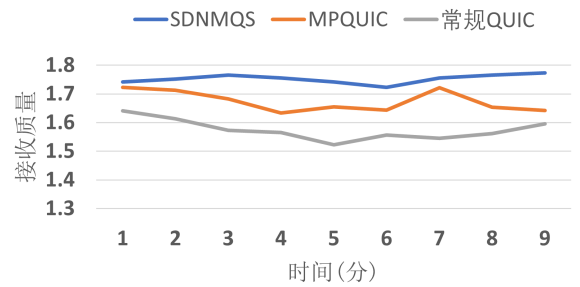
3.3.3　视频接收质量

本文针对视频质量,使用下载吞吐量和视频码率间的比率——接收质量($\rho$)为指标[17]。如果$\rho > 1$,证明该视频具有良好的接收质量,否则质量较差。图10为不同分辨率视频流接收的质量,由此可见本文所提传输框架相较于传统SDN传输方式,在3种视频分辨率下的接收质量最好。

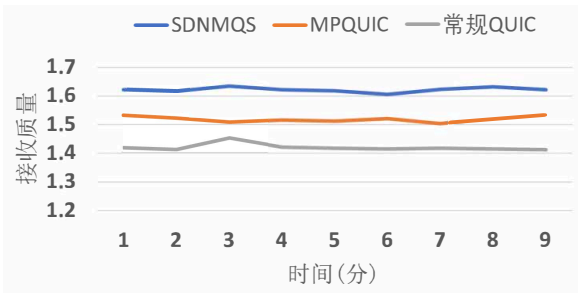
## 4　结语

本文针对视频流传输优化开展研究,提出了一种基于SDN的分段路由多路径QUIC传输框架,设计了SDN网络上的多路径QUIC路由算法,通过多条最短路径传输MPQUIC子流。

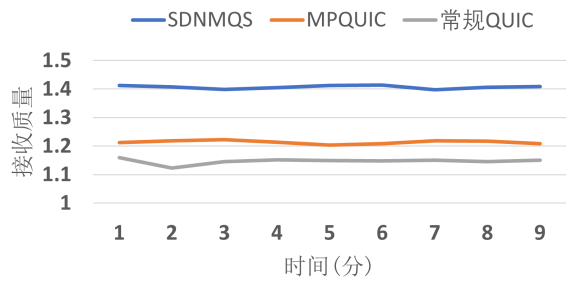
在仿真平台Mininet与传统方式的比较实验结果表明,在SDN控制器中设计的多个模块能提升QUIC的路由控制和资源分配能力,提升系统吞吐量,缩短故障链路恢复时间,使得高质量传输视频流得到保障。




(a)　480 p




(b)　720 p




(c)　1 080 p

**Fig. 10　Reception quality of different videos**

**图10　不同视频接收质量**

**参考文献:**

[1] CISCO. Cisco annual Internet report (2018-2023) white paper[EB/OL]. https://www. cisco. com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490. html.

[2] IYENGAR J, THOMSON M. QUIC: a UDP-based multiplexed and secure transport[EB/OL]. https://datatracker. ietf. org/doc/id/draft-ietf-quic-transport-29. html.

[3] Open Networking Foundation. Software-defined networking: the new norm for networks[EB/OL]. http://www. valleytalk. org/wp-content/uploads/2012/05/wp-sdn-newnorm. pdf.

[4] FILSFILS C, NAINAR N K, PIGNATARO C, et al. The segment routing architecture[C]// 2015 IEEE Global Communications Conference, 2015: 1-6.

[5] SANDRI M, SILVA A, ROCHA L A, et al. On the benefits of using multipath tcp and openflow in shared bottlenecks[C]// 2015 IEEE 29th International Conference on Advanced Information Networking and Applications, 2015: 9-16.

[6] ZANNETTOU S, SIRIVIANOS M, PAPADOPOULOS F. Exploiting path

diversity in datacenters using MPTCP-aware SDN[C]// 2016 IEEE Symposium on Computers and Communication, 2016: 539-546.

[7] NAM H, CALIN D, SCHULZRINNE H. Towards dynamic MPTCP path control using SDN[C]// 2016 IEEE NetSoft Conference and Workshops, 2016: 286-294.

[8] DUAN J, WANG Z, WU C. Responsive multipath TCP in SDN-based datacenters[C]// 2015 IEEE International Conference on Communications, 2015: 5296-5301.

[9] GIORGETTI A, SGAMBELLURI A, PAOLUCCI F, et al. Segment routing for effective recovery and multi-domain traffic engineering[J]. Journal of Optical Communications and Networking, 2017, 9(2): A223-A232.

[10] LI Y, TANG H, MA S Q. Multi-path scheduling algorithm based on segment routing in SDN[J]. Application Research of Computers, 2021, 38(5):1514-1519.
李艺,唐宏,马枢清. 软件定义网络中基于分段路由的多路径调度算法[J]. 计算机应用研究,2021,38(5):1514-1519.

[11] VU V A, WALKER B. On the latency of multipath-quic in real-time applications[C]// 2020 16th International Conference on Wireless and Mobile Computing, Networking and Communications, 2020: 1-7.

[12] MOGENSEN R S, MARKMOLLER C, MADSEN T K, et al. Selective redundant MP-QUIC for 5G mission critical wireless applications[C]// 2019 IEEE 89th Vehicular Technology Conference, 2019: 1-5.

[13] RABITSCH A, HURTIG P, BRUNSTROM A. A stream-aware multipath QUIC scheduler for heterogeneous paths[C]// Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC, 2018: 29-35.

[14] LIM Y, NAHUM E M, TOWSLEY D, et al. ECF: an MPTCP path scheduler to manage heterogeneous paths[C]// Proceedings of the 13th International Conference on Emerging Networking Experiments and Technologies, 2017: 147-159.

[15] SHI X, WANG L, ZHANG F, et al. PStream: priority-based stream scheduling for heterogeneous paths in multipath-QUIC[C]// 2020 29th International Conference on Computer Communications and Networks, 2020: 1-8.

[16] WANG J, GAO Y, XU C. A multipath QUIC scheduler for mobile HTTP/2[C]// Proceedings of the 3rd Asia-Pacific Workshop on Networking, 2019: 43-49.

[17] HOßFELD T, SCHATZ R, BIERSACK E, et al. Internet video delivery in YouTube: from traffic measurements to quality of experience[M]. Heidelberg : Springer , 2013.

[18] YU B, LI X S, WANG W, et al. Research and application of real-time communication optimization based on QUIC[J]. Journal of Chinese Computer Systems, 2021, 42(8):1753-1757.
于波,李炫杉,王卫,等. 基于QUIC的实时通信优化研究与应用[J]. 小型微型计算机系统,2021,42(8):1753-1757.

[19] KuaiShou. Technology developed kQUIC by itself: how to realize tens of millions of QPS clusters [EB/OL]. https://zhuanlan. zhihu. com/p/163550953.
快手. 快手自研kQUIC:千万级QPS集群是如何实现的?[EB/OL]. https://zhuanlan. zhihu. com/p/163550953.

[20] TencentCloud. CLB supports the QUIC protocol[EB/OL]. https://cloud. tencent. com/document/product/214/46008.
腾讯云. CLB支持QUIC协议[EB/OL]. https://cloud. tencent. com/document/product/214/46008.

(责任编辑:刘嘉文)