arXiv:2303.04626v1 [cs.NI] 8 Mar 2023

基于K最短路径优化的多MEC协作VR视频传输与缓存*

Jingwen Xia、Luyao Chen、Yong Tang(✉)、Ting Yang和Wenyong Wang

电子科技大学，中国成都 worldgulit@uestc.edu.cn

摘要。 在近期的网络架构中，为降低VR视频的传输时延，引入了多MEC协作缓存机制，通过利用MEC服务器的计算能力和缓存能力来优化传输过程。然而，许多利用MEC服务器计算能力的方案忽略了编解码过程所消耗的额外算力，因此无法实际应用。此外，最小缓存单元通常是整个VR视频，导致缓存效率低下。

为应对这些挑战，我们基于当前流行的网络架构，将VR视频拆分为瓦片文件进行缓存，并提供可靠的传输机制和有效的缓存策略。由于不同瓦片文件的数量 N 过于庞大，现有的协作缓存算法无法处理如此大规模的输入数据。我们进一步分析了该问题，并提出了一种优化的K最短路径（OKSP）算法，其时间复杂度上界为 $O((K\cdot M+N)\cdot M\cdot \log\ N)$），适用于边数受限的最短路径问题，其中 K是协作域中全部 M 个MEC服务器能够缓存的瓦片总数。我们证明，OKSP算法在任何情况下都能计算出平均时延最低的缓存方案，这意味着所给出的解是精确解。仿真结果表明，OKSP算法在求解大规模数据方面速度卓越，并且在实验中始终优于其他缓存算法。

关键词： 360° 虚拟现实视频 · 虚拟现实 · 移动边缘计算 · K-最短路径。

1 引言

虚拟现实（VR）正成为5G无线网络的一项杀手级应用。用户可以通过VR眼镜等移动设备观看VR视频，获得沉浸式体验。本文所述的VR视频是指水平 360°× 垂直 180°的全景视频。

*四川省科学技术厅项目（编号：2021YFHO171）资助。

VR视频的传输问题已受到学术界和工业界的广泛关注 [13]。在确保良好观看体验的前提下，VR视频所需的比特率远高于传统视频。如果用户从数据中心获取所需的所有VR视频，5G网络回传链路的带宽消耗将大幅增加。此外，沉浸式VR体验要求网络提供超低传输时延（低于20毫秒），以避免用户感到不适 [29]。然而，在传统网络架构下，核心网络和互联网中的流量会动态变化，导致网络负载波动，进而使观看VR视频的性能出现难以预测的下降。

5G网络技术中提出的移动边缘计算（MEC）可以大幅缓解上述问题 [15]。在传统网络架构中引入MEC服务器，并将其部署在用户接入的基站附近。MEC服务器缓存当前热门视频，从而显著降低移动VR设备的请求时延。然而，与数据中心相比，单个MEC服务器的资源非常有限 [23]。为了充分利用有限的存储空间以满足更多用户请求，与地理位置相近的多个MEC服务器进行协作并共享其存储空间已成为必然趋势。

MEC服务器的计算能力也可用于VR视频传输。在进一步分析之前，我们需要了解VR视频从制作到渲染的过程 [20], ，其中包括：（1）拍摄与拼接：通过多摄像机阵列采集视频，然后将采集到的视频帧投影到球面空间中，再进行拼接以获得全景视频；（2）映射与编码：将全景视频图像映射并投影到二维平面（例如使用等距柱状投影技术）以获得二维平面图像，然后使用通用视频编码器对视频进行编码；（3）传输：通过网络将编码后的视频传输至VR设备；（4）渲染：VR设备对VR视频进行解码，将二维视频帧投影到球面空间中，然后在屏幕上显示。

许多论文 [4,13,18,29] 研究了将（4）中的渲染任务卸载到 MEC服务器，然后将渲染后的视频传输到 VR 设备，以降低总体时延。然而，这种优化存在许多问题。

步骤 (4) 中涉及的操作很简单，只需对二维视频帧中每个点的坐标应用一系列三角函数即可。对于单个像素点的操作，可以认为其时间复杂度为常数 [22], ，因此渲染过程不会耗费太多时间。将渲染任务卸载到MEC服务器后，VR视频需要先解码才能进行渲染，这一解码操作会额外耗费时间。由于MEC服务器可能配备了专用于处理视频的GPU，因此可以缩短渲染时间。然而，渲染后，VR视频的大小等于原始未压缩视频的大小，这取决于压缩比（采用H.265编码时压缩比最高可达1000:1 [28]）。若不进行压缩，传输比特率如此之高的视频并不现实。但如果MEC服务器对其进行压缩

渲染后的视频，额外编解码所耗费的时间甚至会超过渲染节省的时间，更不用说图像质量损失了。总的来说，利用MEC服务器的计算能力处理渲染任务，并不能改善用户观看VR视频的体验，反而会使问题变得越来越复杂。

但是，MEC服务器的计算能力可用于执行计算密集型任务，例如预测用户头部运动，从而减少数据传输量。当用户观看 360° VR视频时，视场（FoV）通常仅占全景视场（$90°\times\ 90°/\ 360\ °\times\ 180°$ [12]）的约12.5%。换言之，如果传输整个VR视频，那么观看过程中大部分画面区域对用户来说都是不可见的。为减少传输 360° 环绕图像所导致的大带宽消耗和高时延问题，大多数现有传输方案[10,17,21] 只传输用户视场内图像的高比特率版本。具体而言，在VR视频制作过程中（2），经过等距圆柱投影的全景视频会在空间上被等分为$x\times y$ 可独立编码和解码、且时长与原视频相同的完整时长瓦片视频文件 [4], 。由于用户观看视频时视线会发生转移，完整时长瓦片视频的请求率会随时间波动，并且在某些时段可能较低 [19]。如果将完整时长瓦片视频文件作为最小缓存单元，MEC服务器的缓存效率就会较低，因此可以按时间将其切分为大小相同的瓦片视频，称为瓦片文件或瓦片。

如果只有在 VR 设备检测到需要渲染特定瓦片文件时才发出请求，就会导致可容忍时延非常短（例如不到 20 毫秒），用户体验也会很差。如果缓存了整帧，拆分操作就毫无用处。因此，有必要提前预测并缓存用户可能观看的瓦片文件，这需要预测用户在未来一段时间内的头部运动轨迹。相关研究 [27]表明，近年来已有多种头部运动预测模型被提出，用于高准确率地预测短期头部轨迹变化。文献 [5],提出的基于瓦片的预测模型，在预测周期设为 1 秒时，预测准确率可达 84.22%。

尽管现有预测模型的准确率很高，但仍会出现这样的情况：VR设备准备渲染时，所需的视频帧尚未提前请求。在这种情况下，如果请求缺失视频帧所属的整个瓦片文件，就很容易超时，导致用户感觉到本地屏幕卡顿。为解决这一问题，服务器可以提取当前缺失的瓦片帧进行传输，而不是传输整个瓦片文件。然而，只传输缺失帧也会带来问题。因为主流视频编码 [6] 包含 3 种帧：I帧（帧内编码）、P帧（预测编码）和 B帧（双向预测编码）。I帧包含完整的图像数据。P帧或 B帧包含前后 I帧或 P帧之间的变化信息（这些信息用于生成最终图像）。因此，如果缺失帧是 P帧或 B帧，就无法在 VR设备上直接解码。结果，

一种可行的服务器端补救方法是重建缺失帧，将其压缩为 I帧，然后响应 VR设备的请求。

总之，可以将头部运动预测以及在预测失败时采取的补救措施卸载到MEC服务器，使VR设备能够专注于视频解码和显示。基于上述分析的优化要点，我们旨在构建一种高效的移动VR视频传输框架，并采用多MEC协同缓存机制。我们的贡献总结如下：

- 我们描述了一种多MEC协作网络架构，并提供了可靠的传输机制。- 针对大规模瓦片缓存问题，我们分析了协作架构下瓦片的缓存收益，并将瓦片缓存问题等价地转化为k最短路径问题。- 鉴于原始k最短路径算法[30] 在本文应用场景中的时间复杂度较高，我们提出了一种优化的k最短路径（OKSP）算法。OKSP算法的时间复杂度上界为 $O((K \cdot M + N) \cdot M \cdot logN)$，其中 K是协作域中所有 M 个MEC服务器可缓存的瓦片总数， N是不同瓦片文件的数量。我们还证明，OKSP算法的解是精确解。

本文其余部分安排如下。第2节回顾相关工作。第3节介绍系统架构和传输机制。第4节分析缓存利润，并展示OKSP算法的具体优化细节。第5节讨论仿真结果。第6节总结全文。

2 相关工作

近年来，一些论文利用VR视频的特点设计传输网络架构。由于用户在任意时刻只能观看全景视频的一部分，传输用户当前视场可显著降低带宽消耗。因此，[7] 提出了一种基于瓦片的方案来传输VR视频。 [5] 进一步基于真实数据训练，推导用户的头部运动轨迹模型，从而提升用户体验质量。 [5,7] 重点研究用户观看VR视频时头部运动的预测，描述了预测模型的细节并给出了实验结果。[14,24]采用线性回归方法进行轨迹跟踪。 [23]的作者提出了一种根据用户头部运动和网络条件确定瓦片分辨率的方法。 [14]提出了一种 360° 视频流传输方案，通过预测用户的视口并预取其即将观看的视频片段，最大限度地提升用户体验。

一些研究采用不同的比特率传输方式处理视场内外区域，以减少数据传输量。Ghosh等人 [9] 假设

在获取不可见部分的最低分辨率版本并利用带宽预测确定可见部分的分辨率之前，已准确获知视场。有人提出一种启发式方法 [26] ，通过为当前视场内外的不同区域分配不同权重来分配视频分辨率。 [25,34] 计算当前视场的受欢迎程度，以预测未来一段时间内视场的受欢迎程度，其中 [25] 提出了一种基于视点分配比特率的新算法。

此外，一些论文借助MEC技术实现VR视频传输。[4,29] 提出了一种基于MEC的网络架构，用于交付移动VR视频。在 [29], 中，提出了一种基于视场的缓存架构，以节省带宽消耗并满足VR视频的低时延要求。 [4] 提出了一种基于MEC的架构，用于 360° VR视频交付，并积极利用用户视场的可预测性来缓存VR视频。然而，这两项研究都只涉及单个MEC服务器，未考虑多MEC协作的潜力。

Hou等人 [11], 、Ndikumana等人 [23] 和Wang等人 [32] 注意到了多MEC协作的优势。Wang等人 [32] 提出了一种基于区域的协作内容缓存与交付方案，并设计了一种启发式协作内容缓存策略。Hou等人 [11] 还提出了一种基于多MEC协作的服务架构，以及一种适用于该服务架构、基于迁移学习的协作缓存策略。Ndikumana等人 [23] 提出了一种基于多MEC协作架构、采用计算、缓存、通信与控制联合优化的缓存策略。然而，这些研究都集中于通用视频内容，未利用VR视频的特性。

利用多MEC协作传输，一些研究发挥了VR的特性。Ge等人 [8] 提出了一种多路径协作路由方案，以促进边缘数据中心、基站与用户之间360度MVRV的传输。研究者提出了一种移动边缘计算系统 [33] ，利用移动VR设备的计算和缓存资源，减少通信资源消耗。为实现高质量VR流媒体，Chakareski等人[3] 整合了可扩展分层360视频切片、视口自适应速率分配、最优资源分配和边缘缓存。

3 系统架构

3.1 网络架构

基于多MEC协作的网络架构如图 1 所示。该网络架构由三部分组成：位于数据中心的远程内容服务器（例如云服务器）、5G核心网和一个MEC协作域。我们假设云服务器拥有海量存储空间，用于存储所有编码切片文件。由于基站的覆盖区域相对较小（$\leq$ 500m [31]），因此通常会在居住区或商业区部署多个基站，这些基站在地理位置上彼此接近。为了共享计算和缓存资源，可以将这些基站附近的MEC服务器划分为一个MEC协作域。

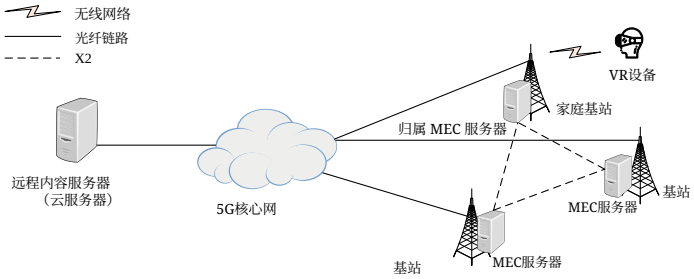

图 1。 网络架构图。

假设一个协作域中有 M 台MEC服务器，记为 {MEC服务器1，...，MEC服务器 M}。每台MEC服务器都维护一张资源访问表，记录其缓存的瓦片文件。与VR设备直接连接的基站称为归属基站，部署在归属基站附近的MEC服务器称为归属MEC服务器。

3.2 传输过程

如前文所述，VR设备有必要预取未来一段时间内将使用的瓦片。近期提出了各种头部运动预测模型，可高精度预测短期轨迹变化 [27]。因此，在用户的观看过程中，必须持续执行预测任务。预测工作需要一定的计算量，将其卸载到归属 MEC 服务器可以减轻 VR设备处理器的负载。当 VR设备专注于解码和显示 VR视频时，归属 MEC 服务器负责持续预测用户的短期头部运动轨迹，并根据预测结果传输可能涉及的瓦片文件。

然而，由于预测偏差，VR设备在渲染 VR视频时可能会发现未提前请求相应的瓦片文件。在这种情况下，VR设备必须立即请求当前缺失的瓦片视频帧。每当 VR设备刷新屏幕时，都可以将用户当前的观看信息发送到归属 MEC 服务器，以便及早发现预测遗漏并采取补救措施。

在本文中，VR设备和归属 MEC 服务器各自执行以下传输过程。

对于 VR 设备。用户开始观看视频时，VR设备会与其归属 MEC 服务器建立长连接。每隔一个屏幕渲染间隔 Δt_{fps} time，系统都会检测当前播放状态。如果视频播放结束，交互过程随即结束，VR设备也会断开与其归属 MEC 服务器的连接。否则，执行屏幕渲染操作；同时，将用户当前视场和视频播放时间 t_v 发送至其归属 MEC 服务器。

对于 MEC 服务器。归属 MEC 服务器会记录其服务区域内用户的缓存状态。当 MEC 服务器收到来自 VR设备的信息包时，可以根据该信息包中的视场和 t_v 判断用户是否缺少渲染帧。若缺少，则先执行补救操作；否则，直接执行预测操作。

补救操作。 如引言中所述，由于此时请求的容忍时延（20ms）相当严格，归属 MEC 服务器会返回 VR设备缺失的帧，而不是瓦片文件。根据资源访问表，服务器会查找此渲染过程中生成的所有缺失视频帧，并将它们发送回 VR设备，然后执行预测操作。

预测操作。 归属 MEC 服务器将用户的当前视场添加到用户轨迹中，并运行预测模型，预测用户下一时段的轨迹（$t,\ t+\Delta t]$ 以及该轨迹涉及的瓦片文件。之后，服务器会查找设备尚未缓存的所有瓦片文件，并将它们发送回 VR设备。由于一个瓦片文件包含多帧，其中一些帧可能早于 t_v，这意味着它们永远不会被观看，因此除非必要，我们只发送晚于 t_v的帧。

这些例外是由视频压缩方法造成的。P帧和B帧无法在VR设备上直接解码，因此我们需要向前查找到最近的I帧位置，并从该位置开始发送。考虑到单个瓦片文件的大小较小， t_v 之后的所有帧都会一并发送，即使它们将来可能不会被渲染。

4 缓存策略

为了设计理想的缓存算法，我们将分析每条可能请求路径所涉及的时延，建立时延减少量的收益模型，最后提出一种用于求解带有最短路径边数约束的 k 条边不相交最短路径问题的算法。该算法最终将给出利润最高的缓存方案。

4.1 时延分析

计算缓存方案时，假设整个MEC协作域中有 N 个不同的瓦片被请求，这些瓦片表示为{瓦片$_1$,...、瓦片$_N$}。VR设备对瓦片$_n$ 发出的请求$_r$ 的数据规模为 sz_r，用户与MEC服务器之间的带宽为 Bu_r ，MEC服务器之间的带宽为B_m ，MEC服务器与云服务器之间的带宽为 B_{dc} 。MEC服务器上的请求数据包和响应数据包在处理和发送之前都需要排队。我们假设协作域中的所有MEC服务器在运行时负载适中，因此排队时延不会太长。 T_{q_1} 用于表示请求到达MEC服务器的接收缓冲区后等待处理的最长时间， T_{q_2} 用于表示

响应数据发送前的最长排队时延。所有请求均经过以下一种或两种通信过程。

VR设备 ↔ 归属 MEC 服务器。VR设备与其归属 MEC 服务器之间只有一跳。由于无线电波以光速传播，且两者之间的距离通常不到 1 公里，因此传播时延可以忽略不计。此情况下的时延为

$$ch(sz_r, Bu_r) = \frac{sz_r}{Bu_r} + T_{q1} + T_{q2}. \tag{1}$$

归属 MEC 服务器 ↔ 其他 MEC 服务器。协作域中的基站通过 X2 接口通信[23], ，底层物理链路为光纤，是一条单跳有线链路。MEC服务器之间的距离不超过 10 公里，因此传播时延也可以忽略不计。协作域中的 MEC服务器具有相似的网络配置，因此任意两台服务器之间的通信带宽相同。此情况下的时延为

$$cmm(sz_r) = \frac{sz_r}{B_m} + T_{q1} + T_{q2}. \tag{2}$$

任意 MEC 服务器 ↔ 数据中心。由于归属 MEC 服务器到远程数据中心之间的网络条件较为复杂，包括多跳链路以及线路不稳定导致的网络波动，因此无法准确计算时延。为了确定数据包通过这些链路所需的时间，我们采用通信时延 $T_{m\leftrightarrow dc}$，其会在一定范围内波动。此情况下的时延为

$$cc(sz_r) = \frac{sz_r}{B_{dc}} + 2 \cdot T_{m\leftrightarrow dc}. \tag{3}$$

如上所示，该方程组可用于估算预取请求和修复请求的时延。然而，修复请求还会产生处理延迟，因为需要从缺失瓦片文件中提取压缩所需的帧。由于最近的 I 帧信息是按需获取的，因此只能在存有缺失瓦片文件的移动边缘计算服务器上执行修复。处理延迟与缓存策略无关。因此，我们不考虑通过缓存策略降低处理延迟。

随后，我们讨论每个瓦片文件的缓存分布如何影响请求时延。设移动边缘计算服务器 m 上关于瓦片$_n$ 的请求为 $R_{m,\ n}$。我们发现，只要协作域中存在瓦片$_n$ 的任意副本，就会产生全局收益

$$v_n^{glo} = \sum_{m=1}^{M} \sum_{r \in R_{n,m}} cc(sz_r) - cmm(sz_r), \tag{4}$$

因为一旦协作域内存在副本，所有针对瓦片$_n$ 的请求路径都会改变，请求端点也会从云服务器转移到协作域内的MEC服务器。这部分全局利润只能

累计一次。此外，特殊MEC服务器 m 上的瓦片n 缓存会产生本地利润

$$v_{n,m}^{loc} = \sum_{r \in R_{n,m}} cmm(sz_r), \tag{5}$$

由于移动边缘计算服务器 m 服务区域内的所有瓦片n 请求都由其直接处理，因此消除了MEC服务器间通信。

4.2 抽象方法

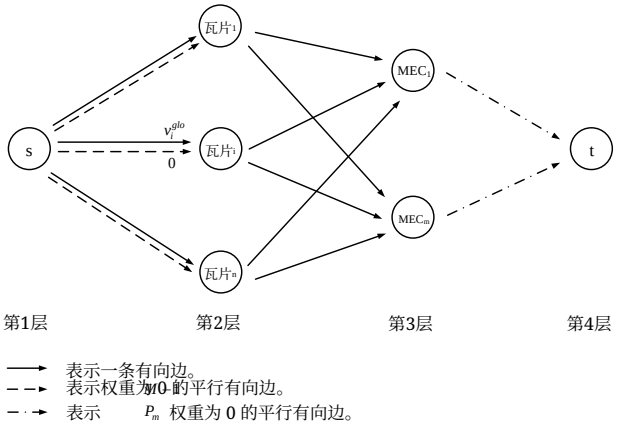

图 2. 将缓存问题重构为k最短路径问题的示意图。

将所有原始VR视频拆分为大小相同的瓦片文件后，令 S 表示一个瓦片文件的大小， $P_m = {}^{C_m}_{S}$表示存储在移动边缘计算服务器 m 中的瓦片文件最大数量，其中 C_m 为该移动边缘计算服务器m的存储容量。在我们的具体情形中，缓存分发问题可重构为图结构 $G = (V, E)$，如图 2 所示，该图具有一个源节点和一个汇节点，是一个有向无环图（DAG）。

为方便起见，深度为 d 的节点称为第d 层节点。在DAG G中，只有四层。第1层是源节点 s，第2层是对应于每个瓦片文件的瓦片节点，第3层是对应于协作域内各移动边缘计算服务器的MEC节点，第4层是汇节点 t。

如果从节点 i 到节点 j只有一条有向边，则用 $e_{i,j}^t$表示；如果有多条，则用 $e_{i,j}^t$ 表示这些 t^{th} 平行有向边。边权重表示为 $c(e_{i,j}^t)$。 G 的详细信息将在下文介绍。

$s \rightarrow$ 瓦片节点。 在协作域中，每个瓦片文件最多存储 M个副本。因此，从节点 s 到瓦片节点 i有 M条平行有向边 $\{e_{s,i}^1, ..., e_{s,i}^M\}$。由于全局收益只能累积一次，$c(e_{s,\ i}^1) = v_i^{glo}$，并且 $c(e_{s,\ i}^2)$ ..... $c(e_{s\ ,\ i}^M)$ 应为 0。

瓦片节点 $\rightarrow$ MEC节点。对于任意第3层 MEC节点 j，其可选存储对象包括所有瓦片文件，因此对于任意第2层瓦片节点 i，都存在一条有向边 $e_{i,\ j}$，且 $c(e_{i,\ j}) = v_i^{loc}{}_{,\ j}$。

MEC 节点 $\rightarrow$ t。为限制 MEC 的存储容量，对于任意第 3 层 MEC 节点 j，存在 M条从 j到 t、权重为 0 的平行有向边 $\{e_{j,t}^1, ..., e_{j,t}^M\}$。

源节点 s 与汇节点 t 之间的任意路径 G 都代表原问题中的一次缓存操作。选择一条简单有向路径 $< s,\ i,\ j,\ t >$等价于将瓦片文件 i 存储在 MEC服务器 j上。对所有边的权重取反后，原问题被重新表述为一个k 条最短边不相交路径 问题，该问题在网络优化文献中已得到广泛研究 [1]。

令 $K = \sum_{m=1}^{M} P_m$ 表示协作域可存储的瓦片文件最大数量。我们的目标变为寻找从源节点 s 到汇节点 t的 K 条路径 $\{p_1, ..., p_K\}$，使这些路径的总成本最小；该问题可以使用 k 最短路径（KSP）算法求解 [30]。

算法1给出了原始KSP算法的伪代码概要，KSP算法的简要流程可描述为: (1) 构建上文所述的图G ； (2) 计算从 s 到 t的最短路径 p ； (3) 在 G中，将 p 中的所有边反向，并将其权重取反；(4) 重复步骤(2)(3) $k-1$ 次。

在第 l^{th} 次迭代中，将通过最短路径算法计算一条增广路径 ap_l ； 令 $AP_l = \{ap_1, ..., ap_l\}$ 表示截至第 l次迭代所计算出的所有 l 增广路径的集合，而 G 的负边将构成最短路径集合$P_l = \{p_1, ..., p_l\}$。特别地，在每次迭代之后，图 G中都不存在负环，这一点已在 [30]中得到证明。

4.3 模型分析

表 1. 符号定义

符号	描述
N	不同瓦片文件的数量
M	协作域中的MEC服务器数量
P_m	MEC服务器 m 最多可存储 P_m 个瓦片文件
K	协作域最多可存储 K 个瓦片文件
$e_{i,j}^t$	从节点 i 到节点 j的第 t^{th} 条平行有向边
$c\ (e_{i,\ j}^t)$	边的权重 $e_i^t{}_{j}$

算法1的最坏情况复杂度为 $O(K \cdot M \cdot N)$，这意味着原始KSP算法无法处理 N 变大的情况。下面，我们展示如何通过一些优化显著降低算法的时间复杂度。我们发现，第3层节点的数量明显

少于第2层节点的数量（$M \ll N$），因此我们可以从这一点入手优化KSP算法。最终，我们提出了优化的k最短路径（OSKP）算法。

定理1。 在 G中，存在一个最短路径集 $AP_K^* = \{ap_1, ..., ap_K\}$；对于任意路径 $ap \in AP_K^*$， ap 中的每个节点都只会出现一次，但第2层中的瓦片节点除外。

证明。 由于 G 在每次迭代中均不存在负环，且移除非负环不会增加路径长度，因此我们可以运行KSP算法得到集合 AP_K，然后反复移除不满足要求的路径中的环，直到 AP_K 中的所有路径都满足要求。

定理2。 在增广寻路过程中，如果路径的端点位于第3层节点 j，那么经由第2层节点 i 传输到另一个第3层节点 k 的条件是：移动边缘计算服务器 j 存储了瓦片$_i$，而移动边缘计算服务器 k 没有存储该瓦片。

我们将定理2中的过程称为一次传输，该过程会产生v_i^{loc}传输损失$_{,\ j} - v_{i\ k,}^{loc}$，并将 i 称为中转节点。

定理3. 对于任意 $ap \in AP_K^*$， ap 的边数不超过 $2 \cdot M + 1$。

证明。 路径 ap 经过前两条边后将到达一个第3层节点（节点 s 不会重复出现），随后会在第3层节点之间传输，或到达汇节点 t。由于 ap 中的每个MEC节点都不会重复出现，因此初始时已有一个第3层节点被访问；每次传输后，已访问的第3层节点数量增加1。共有 M 个第3层节点时，传输 次数不超过$M-1$，且每次传输 后边数增加2。因此， ap中的边数不超过 $2 \cdot M + 1$（传输 不会增加深度，而 ap 需要3条边才能到达汇节点 t）。

基于上述定理，我们提出了OKSP算法。该算法等价于KSP算法，但在我们的模型中具有较低的时间复杂度。由于本文场景中的瓦片数量 N 远大于MEC服务器数量 M，以下分析假设 $N > M$。

优化1。 使用有序数组维护从源节点 s 到每个第3层节点的最短距离。

假设在增广寻路过程中，路径的端点为 s，需要向前经过两条边（$s \to i \to j$）才能到达 MEC节点 j。共有 $2 \cdot N$ 种可能的路径，如果检查每个瓦片节点，时间复杂度为$O(N)$。然而，最终路径会选择利润最高的两条边（边权重之和最小）。

前两条边选择哪条并不重要。路径的端点始终会变为节点 j，节点 j 会成为已访问的 MEC 节点，而边集$\{e^k_{from,to} | from = s\ or\ to = j\}$ 将不再可选。在这种情况下，最好选择能够产生最大利润的两条边。

首先，我们将从节点 s 到每个移动边缘计算节点j 的所有 $2 \cdot N$ 条可能路径加入数组 $PathArray_j$ ，并按利润降序排序。在路径搜索过程中，每当发生从节点 s 到移动边缘计算节点 j的传输时，我们只需从 $PathArray_j$中选取利润最高的路径，并在每次迭代时动态移除不可行解。详情请参阅算法2中的 $Initialize\text{-}Possible\text{-}Paths$ 和$Maintain\text{-}Path\text{-}Array$ 。

优化2。 使用优先队列维护第3层节点之间的最短距离。

假设在增广路径搜索过程中，路径的端点是移动边缘计算节点 j，并准备传输到移动边缘计算节点 k。可能有多个中转节点可供选择，如果需要检查所有瓦片节点，时间复杂度将为 $O(N)$。但可以证明，路径最终会选择传输损失最小的中转节点。

因为无论选择哪个中转节点，路径的端点始终会变为移动边缘计算节点 k，移动边缘计算节点 k 会成为已访问的移动边缘计算节点，而边集$\{e^k_{from,to}|from = j \ or \ to = k\}$ 将不再可选。在这种情况下，最好选择会导致传输损失最小的中转节点。

优先队列 $LossQueue_{j,\ k}$ 可用于动态维护从移动边缘计算服务器 j 传输到移动边缘计算服务器 k的所有可行中转节点。详情请参见算法2中的$Maintain\text{-}Loss\text{-}Queues$ 。

经过上述优化后，第2层的瓦片节点被隐藏，原始图 G 可转换为图3中节点数为 $M + 2$ 的新图 G' 。

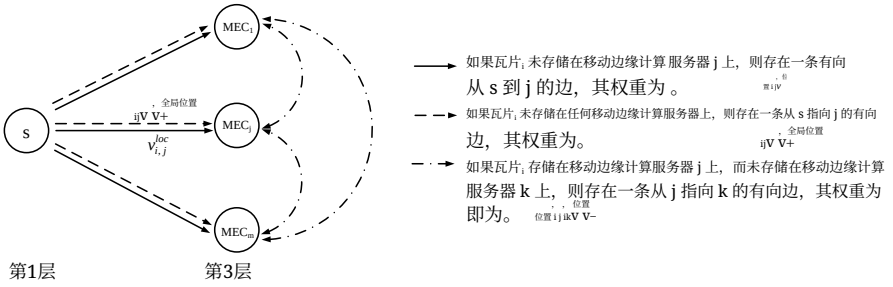

图3。 优化 k最短路径问题的示意图。请注意，该图展示了协作域中瓦片$_i$ 利润的可能情况。

定理4。 *OKSP* 算法计算出的 k 条最短路径的总长度等于 *KSP* 算法计算出的总长度。

证明。 假设在第 l^{th} 次迭代中通过 G' 获得的增广路径 ap_l 为（$s \rightarrow$ 移动边缘计算$_1 \rightarrow ... \rightarrow$ 移动边缘计算$_j \rightarrow ...t$）。根据优化1和2，我们知道

$s \rightarrow$ 移动边缘计算$_1$ 将贪心地选择利润最大的路径，而移动边缘计算$_j \rightarrow$ 移动边缘计算$_{j+1}$ 将选择传输损失最小的中转节点；算法在维护 $PathArray$ 和 $LossQueue$ 时涵盖了所有可用选项，并且所有最短路径的长度都等于所有增广路径的长度。因此，定理4成立。

下面分析时间复杂度。OKSP算法的主要时间开销如下。（1）计算增广路径。共进行K次迭代，每次都使用Dijkstra算法。时间复杂度为 $O(K \cdot M^2)$。（2）维护MEC路径数组。路径数组需要对每个MEC节点的所有可能缓存选择的利润进行排序，时间复杂度为 $O(M \cdot N \cdot \log\ N)$。（3）维护MEC损失队列。由于每次迭代得到的增广路径的边数不会超过 $O(M)$，因此添加操作不会超过 $O(K \cdot M^2)$次。优先队列添加操作的时间复杂度为 $O(\log\ N)$；移除操作的次数不会超过添加操作的次数，因此可以忽略。

因此，总时间复杂度为 $O((K \cdot M+N) \cdot M \cdot \log\ N)$。对于随机输入，每次迭代的增广路径边数远小于 M，运行时间将比预期更快。

在MEC协作域中，协作系统会定期运行OKSP算法，选择该协作域中负载最低的MEC服务器来计算缓存方案，并将其分发给其他MEC服务器。其他MEC服务器一收到缓存方案，就会请求它们尚未拥有且需要缓存的瓦片文件。

5 实验

为了评估OKSP算法的低复杂度及其对大规模输入数据的适用性，我们进行了数值仿真实验。此外，通过与其他协作缓存算法进行比较，我们证明了在所有情况下，OKSP算法都能计算出平均请求时延最低的缓存方案。

对于参数，我们根据基本物理情况并参考类似研究来设定其取值。我们假设协作域中的所有MEC服务器具有相同的存储空间。VR设备的带宽为 [50,100] Mbps，MEC服务器的带宽为500 Mbps，云服务器的带宽为1 Gbps。

瓦片文件大小为 10MB。假设修复请求请求的内容占瓦片大小的 10%$\sim$ 20%，预取请求请求的内容占瓦片大小的 50% $\sim$ 100%。与大多数研究一样 [2,32]，我们假设瓦片的流行度服从 Zipf分布，且 Zipf分布的形状参数 α 为 1.5。通常，形状参数 α越大，热门内容越集中。假设通信过程中的排队时延 T_{q1} 为 1ms， T_{q2} 为 2ms。VR设备与云服务器之间链路上的分组传输时延 $T_{m \leftrightarrow dc}$ 为 [50,100]ms。

在协作域中，缓存命中产生的请求时延低于云端响应。请求时延的降低称为请求时延优化。同一瓦片的所有请求时延优化之和等于该瓦片的缓存收益。通过降低请求时延，我们旨在改善用户体验。因此，我们采用协作域中的平均请求时延（ARL）优化作为评估算法的指标。我们将 OKSP算法与以下缓存算法进行比较。

- 分布式：协作域中的每台MEC服务器都缓存完全不同的瓦片，以使协作域中的瓦片数量尽可能多。分布式算法的时间复杂度为 $O(M \cdot N \cdot logN)$。 - $Self\text{-}Top$：协作域中的每台MEC服务器都缓存其服务区域内最热门的瓦片。Self-Top算法的时间复杂度为 $O(M \cdot N \cdot logN)$。 - $MixCo$：在MixCo中，[32],每台MEC服务器的存储空间被划分为子区域专用存储空间和区域共享存储空间。在MixCo的实现中，为每台MEC服务器分配专用存储空间后，会执行一个时间复杂度为 $O(M \cdot N)$的算法，以计算当前协作域存储状态下的平均请求时延。MixCo算法的最坏时间复杂度为 $O(L \cdot M^2 \cdot N)$。

在上述时间复杂度表示中， L 表示单台MEC服务器缓存的最大瓦片数量， M 表示MEC服务器的数量， N 表示不同瓦片的总数。

5.1 运行时间

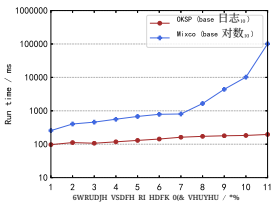

图4. 不同存储空间下OKSP和Mixco算法的运行时间比较

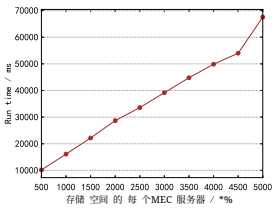

图5. 不同存储空间下OKSP算法的运行时间

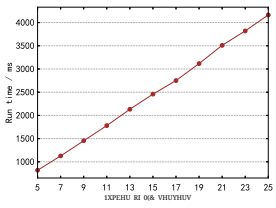

图6. 不同MEC服务器数量下OKSP算法的运行时间

本节将评估 OKSP算法的运行速度。从时间复杂度表示来看，Self-Top和分布式算法的速度都快于OKSP算法。因此，我们不将它们纳入考量。我们对OKSP算法和MixCo算法进行了速度对比实验。

本实验将瓦片数量和MEC服务器数量分别固定为10,000和10，并调整MEC服务器的存储空间。结果如图4所示。随着存储空间增加，OKSP算法的运行时间趋于线性增长，而Mixco算法的运行时间则趋于指数增长。由于MEC服务器可用的存储空间增多，MixCo会生成多个专用空间划分迭代，其时间复杂度为$O(M \cdot N)$，从而影响算法性能。因此，在实际的大规模数据环境中应用MixCo算法并不现实。

为了探究OKSP算法的速度上限，我们进一步增大数据规模。我们将瓦片数量和MEC服务器数量分别固定为500,000和10，并增加每台MEC服务器的存储空间。图5展示了结果。随着存储空间增加，OKSP的速度呈线性增长，这与根据时间复杂度直观推断出的存储空间对其速度的影响一致。

此外，我们继续实验MEC服务器数量变化对OKSP速度的影响，结果如图6所示。本实验固定瓦片数量为200,000，并将每台MEC服务器的存储空间固定为2TB。由于MEC服务器数量的数量级较小，结果表明其对OKSP速度的影响呈线性。总体而言，OKSP算法适用于大规模数据问题。

5.2 性能比较

在接下来的实验中，我们专门研究移动边缘计算存储空间、瓦片数量以及Zipf分布的形状参数 α 对上述四种缓存算法性能的影响。MixCo算法不适用于大规模数据，因此我们先用规模较小的数据测试这四种算法的性能，再移除MixCo算法，用大规模数据测试其余三种算法。以下结果为三次仿真运行的平均值。

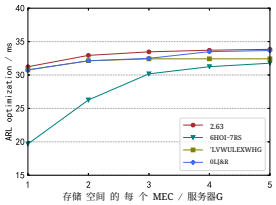

图7. 不同存储空间下的性能比较

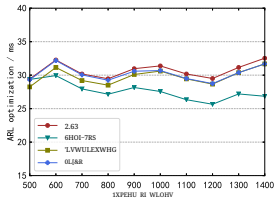

图8. 不同瓦片数量下的性能比较

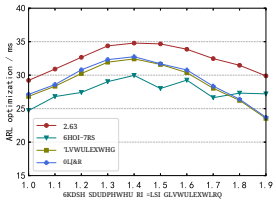

图9. 不同形状参数 α下的性能比较

首先，我们比较移动边缘计算服务器存储空间变化时的ARL优化。我们将移动边缘计算服务器的数量和瓦片数量分别固定为5

和1500。我们将移动边缘计算服务器的存储空间从1 GB增加到5 GB。图7展示了结果。随着存储空间增加，所有缓存算法的ARL优化均有所提升。这是因为增加存储空间后，MEC协作域中可以缓存更多瓦片，从而避免更多请求被发送到云端。

图8展示了不同瓦片数量下的ARL优化。该实验假设MEC服务器存储空间为2 GB，MEC服务器数量为5。由于热门瓦片分布较为分散，且各MEC服务器上的热门瓦片不同，Self-Top算法表现不佳。

图9展示了形状参数 α 对ARL优化的影响。我们将MEC服务器数量固定为5，MEC服务器存储空间固定为2 GB，瓦片数量固定为1000。在图9中可以观察到，随着 α 增大，四种算法的ARL优化在初期均有所提升。然而，随着 α 进一步增大，四种算法的ARL优化会下降。这是因为随着 α 增大，热门瓦片变得更加集中。也就是说，少数瓦片占据了大部分请求流量。因此，对这些瓦片进行协作域缓存能够带来可观的利润。随着我们继续增大α，尽管总体时延节省增加，但每个瓦片的请求数量也会增加，最终导致ARL优化下降。

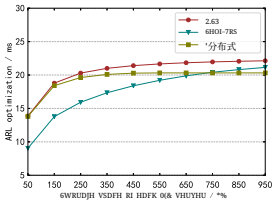

图10. 不同 MEC服务器存储空间下 OKSP、Self-Top 和分布式算法的性能比较（各 MEC服务器上的热门瓦片相似）

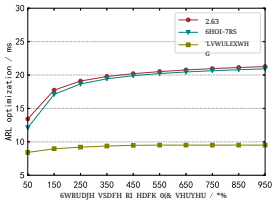

图11. 不同 MEC服务器存储空间下 OKSP、Self-Top 和分布式算法的性能比较（各 MEC服务器上的热门瓦片不同）

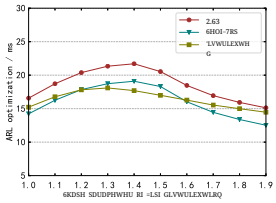

图12. 不同形状参数 α下 OKSP、Self-Top 和分布式算法的性能比较

最后，我们在不使用 MixCo 的情况下，使用更大规模的数据测试其余三种算法的 ARL优化。图 10 和图 11 展示了 MEC服务器存储空间变化时的 ARL优化。假设有 10 台 MEC服务器和 500,000 个瓦片。在图 10 中，每台 MEC服务器的流行度相近；而在图 11 中，每台 MEC服务器的流行度完全随机。在不同的流行度分布场景下，Self-Top 和分布式算法的表现差异很大。相比之下，流行度分布不会限制 OKSP算法的性能，无论热门瓦片是分散还是集中，它都能保持稳定表现。

图 12 展示了 α 变化时的 ARL优化。我们假设 MEC服务器数量为 10，MEC服务器存储空间为 500 GB，瓦片数量为 500,000。图 12 的结果呈现出与小规模数据实验（图9）相似的现象。总体而言，OKSP算法的表现始终优于其他算法。

6 结论

本文在当前流行的多MEC协作网络架构下，提供了一种可靠的传输机制和有效的缓存策略。我们利用MEC服务器的计算与缓存能力，提升用户的观看体验质量。在计算方面，我们提出由MEC服务器负责预测，并在预测失败时进行补救，而VR设备仅专注于解码和播放VR视频。在缓存方面，我们将VR视频拆分为瓦片文件，从而提高缓存命中率。

为解决数量级巨大的瓦片缓存问题，我们首先分析瓦片的缓存收益，然后运用图论将该问题表述为k最短路径问题。随后，我们提出OKSP算法，以快速解决重新表述后的问题。该算法的时间复杂度上限为 $O((K \cdot M + N) \cdot M \cdot logN)$，适用于边数受限的最短路径问题。最后，我们通过运行时间和性能比较对所提出的OKSP算法进行了评估。结果表明，在我们的应用场景中，OKSP算法能够高速处理大规模数据，并且在任何情况下，其平均时延都低于其他缓存方案。

作为模型构建过程的一部分，我们分析了协作域负载对平均排队时延的影响。然而，实际情况更为复杂。在本研究计划的下一步中，我们将引入一个合理的模型来描述移动边缘计算服务器上的负载，并对本文中的模型进行改进。

A 附录 - KSP算法

在本附录中，算法1给出了原始k最短路径算法的伪代码概要[30], 算法2给出了优化的K最短路径算法的伪代码。在算法2中，经过多次迭代后，图中会出现负权重的边。我们采用Johnson算法 [16]中的技巧对所有边重新赋权，使其权重均为正，然后在每次迭代中应用Dijkstra算法计算最短路径树。感兴趣的读者可参阅 [30]了解更多详情。

算法1: 原始 K 最短路径算法

函数 *Reverse*(G, p)
    /* 反转 p 中所有边的方向,
    并将它们的权重取反。 */
    对于 $e^t_{i,\ j} \in p$ **执行**
        添加 $e^t_{j,i}$ in G
        $c(e^t_{j,\ i}) = -c(e^t_{i,\ j})$ 删除 $e^t_{i\ \ j}$ 来自 G
    返回 G
函数 *Original-K-Shortest-Paths*
    构建第 4.2 节所述的初始图 G
    对 $l \leftarrow 1$ *to* K **执行**
        $AugmentingPath_l \leftarrow$ *Shortest-Path-Algorithm*$(G,\ s,\ t)$
        如果 $AugmentingPath_l$ *not exists* **，则**
            中断
        $G \leftarrow$ 反转$(G,\ AugmentingPath_l)$
    $P \leftarrow$ 根据负边在 G 中查找最短路径
    返回 P

算法2： 优化K最短路径第1部分

```
函数 Initialize-Possible-Paths
    对于 j ← 1 to M 执行
        对于 i ← 1 to N 执行
            /* 瓦片_i 存储在MEC服务器 j中的两种利润。
               */
            将 (v_i^{loc},j + v_i^{glo}, i) 添加到 PathArray_j
            将 (v_i^{loc},j, i) 添加到 PathArray_j
        按 PathArray_j 降序排序
函数 Maintain-Path-Array
    对 j ← 1 to M 执行
        /* 从 PathArray 中移除不可行路径。                    */
        当 PathArray_j is not empty 时执行
            (profit, i) ← 中的最大利润元素
            如果 tile_i is stored in the collaboration domain，则
                /* 检查瓦片_i 的全局收益是否已经
                   累积。                                    */
                如果 profit > v_i^{loc}, j 或 tile_i is stored in MEC server j，则
                    从 PathArray_j 中移除（profit, i）
                    继续
            中断
函数 Setup-Graph(G')
    /* 由于最短路径算法，需要对
       所有边                                               */
    初始化：G' 中的所有节点彼此不可达对于 j ← 1 to M 执行
        如果 PathArray_j is not empty，则
            (profit, i) ← PathArray_j 中利润最大的元素
            c(e_{s, j}) ← −profit
        如果 MEC server j has storage space left，则
            c(e_{j, t}) ← 0
        对于 k ← 1 to k 和 k ≠ j 执行
            如果 LossQueue_{j, k} is not empty 则
                (loss, i) ← 中的最小损失元素 LossQueue_j, k
                c(e_{j, k}) ← −loss
```

函数 *Maintain-Loss-Queues(spath, CacheTable)*
    /* $spath$ 的边表示为 $(j \to k, i)$，表示
       MEC节点 j 通过瓦片$_i$传输至节点 k ；特别是，
       路径 $(s \to sj, si)$ 的第一条边表示该移动边缘计算
       服务器 sj 新存储了瓦片 si，该瓦片未被传输
       来自另一台移动边缘计算；最后一条边（$tj \to t$， $null$）不包含
       所传输的瓦片；其余边均视为
       transferred edges。 */
    标记移动边缘计算服务器 sj 正在 $CacheTable$缓存瓦片si
    协作域 += 1中缓存的瓦片$_{si}$ 副本数量
    移动边缘计算服务器 tj -= 1的剩余存储空间
    /* 添加新的转移节点。 */
    遍历 $(remove \to add, i) \in transferred\ edges$，**执行**
        标记 MEC服务器 add 正在缓存瓦片$_i$，位于 $CacheTable$
        标记 MEC服务器 $remove$ 不再缓存瓦片$_i$，位于 $CacheTable$
        遍历 $j \leftarrow 1\ to\ M$，**执行**
            如果 $j \neq add$ 且 $MEC\ j\ does\ not\ cache\ tile_i$，**则**
                添加（$v^{loc}_{i,add} - v^{loc}_{i,}$， j）至 $LossQueue_{add,\ j}$
            如果 $j \neq remove$ 且 $MEC\ j\ is\ caching\ tile_i$，**则**
                添加（$v^{loc}_{i\ ,\ j} - v^{loc}_{i\ ,\ remove}$， j）至 $LossQueue_{j,\ remove}$
    /* 移除不可行传输节点。 */
    对于 $j \leftarrow 1\ to\ M$，**执行**
        对于 $k \leftarrow 1\ to\ k$ 和 $k \neq j$，**执行**
            当$LossQueue_{j,\ k}\ is\ not\ empty$ **时，执行**
                （$loss$， i）←中 $LossQueue_j$， k的最小损失元素
                如果 $MEC\ server\ j\ does\ not\ cache\ tile_i$ 或 $MEC\ server\ k\ is\ caching\ tile_i$ **那么**
                    从 $LossQueue_j$， k中移除（$loss$， i）
                    继续
                中断

函数 *Optimized-K-Shortest-Paths*
    缓存表 ← 用于记录缓存状态的哈希表
    构建 G' 包含 M 个移动边缘计算节点、 s 个节点和 t 个节点
    Initialize-Possible-Paths()
    构建图(G')
    $p_1 \leftarrow Flody\text{-}Algorithm(G',\ s,\ t)$
    重新加权所有边(最短路径树节点 s) **对于** $l \leftarrow 1\ to\ K$ **执行**
        构建图(G')
        $p_l \leftarrow Dijkstra\text{-}Algorithm(G',\ s,\ t)$
        如果 $p_l\ not\ exists$ **则**
            中断
        重新加权所有边（最短路径树节点 s）
        维护损失队列（p_l，缓存表）
    返回 缓存表

参考文献

1. Berclaz, J.、Fleuret, F.、Turetken, E.、Fua, P.：使用K最短路径优化的多目标跟踪。IEEE模式分析与机器智能汇刊 **33**(9)，1806–1819（2011） 2. Borst, S.、Gupta, V.、Walid, A.：内容分发网络的分布式缓存算法。载于：2010年IEEE INFOCOM会议论文集。第1–9页（2010），iSSN：0743-166X 3. Chakareski, J.：VR/AR沉浸式通信：缓存、边缘计算与传输权衡。载于：虚拟现实与增强现实网络研讨会论文集。第36–41页。VR/AR Network '17，美国计算机协会，美国纽约州纽约市（2017） 4. Cheng, Q.、Shan, H.、Zhuang, W.、Yu, L.、Zhang, Z.、Quek, T.Q.：基于移动边缘计算和主动缓存的 360° 移动VR视频流传输的设计与分析。IEEE多媒体汇刊 **24**，1529–1544（2021） 5. Fan, C.L.、Lee, J.、Lo, W.C.、Huang, C.Y.、Chen, K.T.、Hsu, C.H.：头戴式虚拟现实中 360° 视频流传输的注视点预测。载于：第27届网络与操作系统对数字音频和视频支持研讨会论文集。第67–72页。NOSSDAV'17，美国计算机协会，美国纽约州纽约市（2017） 6. Filippov, A.、Norkin, A.、Alvarez, J.R.：视频编解码器要求与评估方法。征求意见稿RFC 8761，互联网工程任务组（2020）。https://doi.org/10.17487/RFC8761，https://datatracker.ietf. org/doc/rfc8761 7. Gaddam, V.R.、Riegler, M.、Eg, R.、Griwodz, C.、Halvorsen, P.：交互式全景视频中的分块：方法与评估。IEEE多媒体汇刊**18**(9)，1819–1831（2016），会议名称：IEEE多媒体汇刊 8. Ge, X.、Pan, L.、Li, Q.、Mao, G.、Tu, S.：用于增强现实与虚拟现实传输的多路径协作通信网络。IEEE多媒体汇刊 **19**(10)，2345–2358（2017），会议名称：IEEE多媒体汇刊 9. Ghosh, A.、Aggarwal, V.、Qian, F.：基于瓦片的360度视频流传输码率自适应算法。arXiv预印本arXiv:1704.08215（2017） 10. Guntur, R.、Ooi, W.T.：无线局域网中感兴趣区域视频流传输的瓦片分配。载于：第22届网络与操作系统对数字音频和视频支持国际研讨会论文集。第59–64页（2012） 11. Hou, T.、Feng, G.、Qin, S.、Jiang, W.：利用迁移学习实现移动边缘计算的主动内容缓存。国际通信系统期刊**31**(11)，e3706（2018） 12. Hou, X.、Dey, S.、Zhang, J.、Budagavi, M.：用于实现移动360度与VR体验的预测式自适应流传输。IEEE多媒体汇刊**23**，716–731（2021），会议名称：IEEE多媒体汇刊 13. Hsu, C.H.：面向移动边缘计算辅助的视场感知和体验质量驱动的虚拟现实自适应360度视频流传输。载于：2020年第16届移动性、感知与网络国际会议（MSN）。第291–298页。IEEE（2020） 14. Hu, H.、Xu, Z.、Zhang, X.、Guo, Z.：针对随机头部运动的最优视口自适应360度视频流传输。载于：ICC 2019-2019 IEEE国际通信会议（ICC）。第1–6页。IEEE（2019） 15. Jedari, B.、Premsankar, G.、Illahi, G.、Di Francesco, M.、Mehrabi, A.、Yl¨a-J¨a¨aski, A.：无线边缘的视频缓存、分析与交付：综述及未来方向。IEEE通信调查与教程**23**(1)，431–471（2020）

16. Johnson, D.B.：稀疏网络中的最短路径高效算法。美国计算机协会会刊（JACM）**24**(1)，1–13（1977） 17. Kimata, H.、Ochi, D.、Kameda, A.、Noto, H.、Fukazawa, K.、Kojima, A.：移动多设备交互式全景视频分发系统。载于：第1届IEEE全球消费电子会议2012。第574–578页。IEEE（2012） 18. Liu, Y.、Liu, J.、Argyriou, A.、Wang, L.、Xu, Z.：多小区移动边缘计算网络中的渲染感知型VR视频缓存。IEEE车辆技术汇刊**70**(3)，2728–2742（2021） 19. Mahzari, A.、Taghavi Nasrabadi, A.、Samiei, A.、Prakash, R.：面向自适应 $360°$ 视频流传输的视场感知型边缘缓存。载于：第26届ACM国际多媒体会议论文集。第173–181页。MM '18，美国计算机协会，美国纽约州纽约市（2018） 20. Mangiante, S.、Klas, G.、Navon, A.、GuanHua, Z.、Ran, J.、Silva, M.D.：VR已进入边缘：如何在移动网络中交付360度视频。载于：虚拟现实与增强现实网络研讨会论文集。第30–35页（2017） 21. Mavlankar, A.、Girod, B.：支持交互式平移/倾斜/缩放的视频流传输。载于：高质量视觉体验，第431–455页。施普林格（2010） 22. Moon, P.、Spencer, D.：球坐标（r， θ、 ψ）。载于：《场论手册：包括坐标系、微分方程及其解》，第24–27页。施普林格出版社（1988） 23. Ndikumana, A.、Tran, N.H.、Ho, T.M.、Han, Z.、Saad, W.、Niyato, D.、Hong, C.S.：大数据多接入边缘计算中的联合通信、计算、缓存与控制。IEEE移动计算汇刊**19**(6)，1359–1374（2019） 24. Nguyen, D.V.、Tran, H.T.、Pham, A.T.、Thang, T.C.：一种用于视口自适应360度视频流传输的最优基于瓦片的方法。IEEE新兴与精选电路及系统期刊**9**(1)，29–42（2019） 25. Ozcinar, C.、Cabrera, J.、Smolic, A.：利用视觉注意力驱动的动态分块实现VR全向视频流传输。载于：2018年IEEE视觉通信与图像处理会议（VCIP）。第1–4页。IEEE（2018） 26. Ozcinar, C.、De Abreu, A.、Smolic, A.：利用瓦片实现面向虚拟现实的视口感知型自适应360度视频流传输。载于：2017年IEEE国际图像处理会议（ICIP）。第2174–2178页。IEEE（2017） 27. Rond´on, M.F.R.、Sassatelli, L.、Aparicio-Pardo, R.、Precioso, F.：视频中头部运动预测方法的统一评估框架。载于：第11届ACM多媒体系统会议论文集。第279–284页。MMSys '20，美国计算机协会，美国纽约州纽约市（2020） 28. Sullivan, G.J.、Ohm.J.R.：第13届JCT-VC会议报告（2013），http: //phenix.it-sudparis.eu/jct/doc

– – – =7746

29

end user/current document.php?id 。Sun, Y.、Chen, Z.、Tao, M.、Liu, H.：移动虚拟现实的通信、缓存与计算：建模与权衡。IEEE通信汇刊**67**(11)，7573–7586（2019） 30. Suurballe, J.W.：网络中的不相交路径。Networks**4**(2)，125–145（1974） 31. Verizon：5G的覆盖范围有多远？（2020），https://www.verizon.com/about/news/ how-far-does-5g-reach 32. Wang, N.、Shen, G.、Bose, S.K.、Shao, W.：采用移动边缘计算的无线接入网基于区域的协作内容缓存与分发。IEEE Access**7**，4031–4044（2019），会议名称：IEEE Access 33. Yang, X.、Chen, Z.、Li, K.、Sun, Y.、Liu, N.、Xie, W.、Zhao, Y.：面向无线虚拟现实的通信约束移动边缘计算系统：调度与权衡。IEEE Access**6**，16665–16677（2018），会议名称：IEEE Access

34. Zhang, Y., Jiang, X., Wang, Y., Lei, K.：通过命名数据网络缓存和交付VR视频。载于：IEEE INFOCOM 2018-IEEE计算机通信会议研讨会（INFOCOM WKSHPS）。第280–285页。IEEE（2018）