2023 IEEE Wireless Communications and Networking Conference (WCNC) | 978-1-6654-9122-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/WCNC55385.2023.10118639

Graph based Joint Computing and Communication Scheduling for Virtual Reality Applications

Fei Liu, Hongyan Li, Peng Wang, Keyi Shi, Yun Hu

State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, 710071, China
Email: feiliu3@stu.xidian.edu.cn, hyli@xidian.edu.cn, pengwangclz@163.com, kyshi10091@163.com, huyun@xidian.edu.cn

***Abstract*—Virtual Reality (VR) applications delivered over wireless networks have attracted interest from academia and industry. The delay of VR applications is mainly composed of computing delay and communication delay. Although cloud computing centers have adequate computing power, accessing them requires long communication delay. Mobile edge computing (MEC), which offloads the computing power from the cloud computing center to the edge, is regarded as a feasible way to alleviate communication delay. However, due to the differences in the capability and location of MEC nodes, the selection of MEC nodes will affect both the computing delay and communication delay. In this paper, we focus on the joint representation of computing and communication resources and the selection of the optimal MEC node. First, we adopt graph-based joint computing and communication resources (GCC) model for VR applications routing and formulate the VR routing problem as an ILP problem. Then we design a Computing Nodes Expanded (CNE) algorithm, which allows us to use the Dijkstra algorithm to quickly obtain the optimal computing node and the path of shortest total delay. Finally, we run numerical experiments to evaluate the performance of the proposal algorithm. Simulation shows that the CNE algorithm can reduce the total delay by 42.9% and increase the delay satisfaction ratio by 23.3% compared to other benchmark algorithms.**

***Index Terms*—Virtual Reality, joint computing and communication, routing, shortest delay.**

I. Introduction

Virtual Reality (VR) is gaining increasing attention in both academia and industry due to its ability to provide a fully immersive experience. Any VR application involves real-time connection and flow of mass information when it comes to online network applications, which brings new challenges to the computing and network conditions. Besides, it is also extremely time-sensitive, typically less than 20ms for motion-to-photon latency (MTP), otherwise it should cause dizziness.

However, achieving VR applications routing requires ultra-high transmission rate and computing power [1]. One basic approach is to put the heavy computing requests in a large cloud computing center and then send the results back to users [2], which may encounter a long communication delay. With the advent of mobile edge computing (MEC), the communication delay is improved by offloading computing power from computing centers to the edge. However, due to the differences in the capability and location of MEC nodes, the selection of MEC nodes will affect both the computing delay and communication delay. Therefore, it is important to find a feasible approach to select an optimal MEC node to obtain the extremely low total delay.

This work is supported by Natural Science Foundation of China (61931017).

Some researchers had focused on the optimization of communication delay by introducting MEC. Specifically, the authors in [1] illustrated the potential gain obtained from utilizing the offloading computing resources. The authors in [3] developed a communication-constrained MEC framework to reduce communication resource consumption via exploiting the computation and caching resources at mobile devices. The authors in [4] analyzed a typical VR framework and explored how much gain computing and cache offloading could bring to communication under both latency and local energy consumption constraints. But both [3] and [4] considered that all the MEC nodes had the same computing performance. The authors in [5] noticed the difference of MEC servers available computing power and designed an utility function for satisfaction of the client nodes. But the function could not directly reflect the delay and hardly optimize the delay from fine granularity. The authors in [6] adopted an improved Dijkstra algorithm to calculated the k paths with the shortest communication delay in the first stage and found the node in these shortest path which had the minimum computing delay in the second stage. However, the shortest path of communication delay does not mean that the total delay is also the smallest, because the node with the minimum computing delay may not be on the path.

There are also many papers that jointly consider the communication and computing resources in the MEC scenario. The authors in [7] developed a communications-constrained MEC framework to reduce communication-resource consumption by fully exploiting the computation and caching resources at the mobile VR device. The authors in [8] proposed an enhanced STAG to jointly represent the storage, communication and computing resources for satellite networks. But [8] was still a two-stage method, which found the transmission path before the calculation. The authors in [9]–[11] adopted mathematical strategies to solve multi-dimensional resource optimization problems, which has high solving complexity.

In this paper, we mainly focus on the graph-based joint communication and computing resources model characteriza-

Authorized licensed use limited to: Beijing Jiaotong University. Downloaded on October 13,2023 at 09:38:04 UTC from IEEE Xplore. Restrictions apply.

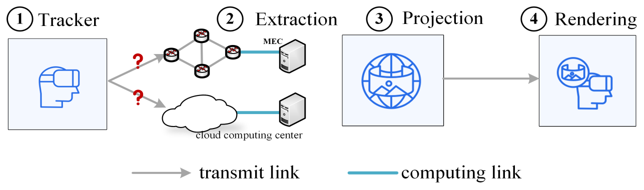

Fig. 1: A simplified process of VR service.

tion and selection of the optimal computing node to achieve the shortest total delay for VR services under assuming that the computing nodes have different computing capabilities. To illustrate the problem, we first analyze a typical VR service process [4], as shown in Fig. 1: 1) Tracker, which tracks the user's behavior and sends information such as the relevant location to the computing center; 2) Extraction, which extracts the 2D video from a spherical video to obtain 2D field of view (FOV) according to the tracking information; 3) Projection, which maps the 2D FOV to two slightly different photos to obtain the 3D FOV; 4) Rendering, which renders the 3D FOV on the users' device. Typically, spherical videos have been computed offline at the computing centers called stitching. The computing centers can be large cloud computing data centers or MEC servers. Besides, extraction and projection can also be performed at the same computing centers as stitching. Therefore, we mainly explore the routing problem from extraction to rendering of VR applications in this paper. The primary contributions of this work can be summarized as follows:

- First, we analyze each step of the VR process in detail, and map the VR process into a routing problem.
- We then present a graph-based joint computing and communication (GCC) model to denote the network of above routing problem. We also define a VR application model and analyze the gain of joint computing and communication routing with an example.
- Next,we formulated the VR routing problem in GCC as an Integer Linear Programming (ILP). Our objective of the problem is to find an end-to-end shortest delay path, which includes the communication delay and computing delay.
- To address this problem, we propose a Computing Nodes Expanded (CNE) algorithm. The algorithm can jointly consider computing and network resources to find the path with the shortest total delay for VR applications. The complexity of the algorithm is just depends on the number of nodes in the network, especially the computing nodes.
- Finally, we run numerical experiments to evaluate the performance of the proposal algorithm. Simulation shows that the CNE algorithm can reduce the total delay by 42.9% and increase the delay satisfaction ratio by 23.3% compared to other benchmark algorithms.

The rest of this paper is organized as follows. In Sec II, we present the GCC model and VR application model and analyze the gain of joint computing and communication routing with an example. In Sec III, we formulated the VR routing problem in GCC as an Integer Linear Programming (ILP). In Sec IV, we propose our CNE algorithm and describes the simulation results in Sec V. Finally, we get the conclusion of this paper in Sec VI.

II. SYSTEM MODEL

In this section, we present a graph-based joint computing and communication (GCC) model, which tightly couples computing and network resources. We then define a VR application model and the routing scenario of VR applications in GCC is described in detail.

A. GCC model

As a first killer application for 5G wireless network, VR consumes huge amount of data while pursuing extremely low latency less than 20ms. Traditional methods still focus on the two-stage method to find the optimal path and computing node that meet the delay requirements, which neglects the coupling relationship between computing and communication. To better find the shortest total delay, we propose a graph-based joint computing and communication model, which can be characterized as $GCC = \{(V, E, B_{uv}, C_v)|u, v \in V, uv \in E\}$ where

- V is the set of nodes,
- E is the set of links,
- B_{uv} is the available bandwidth of link uv,
- C_v is the available computing power of node v.

A simple example of GCC is shown in Fig. 2. The source node $\mathcal{S}$ can be the end users. The intermediate nodes can be considered as computing center with the basic forwarding function and computing capabilities. Thus, the number above these nodes indicates the available computing power. The unit of computing power is Giga Operations per second (GOPS). The values on the links denotes the available bandwidth and the unit of bandwidth is Gbps. The destination node $\mathcal{D}$ can be a mobile VR device.

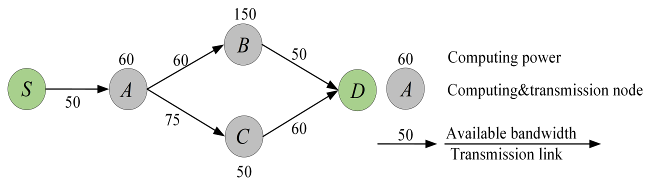

Fig. 2: A simple example of GCC.

B. VR application model

The heavy workload of data and ultra-low delay are the important features of VR application. So in this paper, we mainly consider the data and delay requirements of VR, which can be characterized as $R = (f, D, b, c)$ where

- f is the flow size of VR application R,
- D is the delay demand of VR application R,
- b is the bandwidth demand of VR application R,
- c is the computing power demand of VR application R.

Authorized licensed use limited to: Beijing Jiaotong University. Downloaded on October 13,2023 at 09:38:04 UTC from IEEE Xplore. Restrictions apply.

C. Routing scenario of VR in GCC

Although we are also exploring the end-to-end shortest delay path, it quite different from the traditional shortest delay path after the introduction of computing delay. First, the end-to-end shortest delay path may not be the shortest communication delay path. It is difficult to find the optimal solution if the transmission path and calculation node are selected by separate methods. As shown in Fig. 3(a), assuming the flow size of R is 300M. if we use Dijkstra algorithm for the shortest communication delay, the $S-A-C-D$ path will be easily arranged. However, whether the computation is performed on node A or node C, the total delay will eventually exceed 20ms as shown in Fig. 3(b). In fact, the optimal path is $S-A-B-D$ with a total delay of only 19ms. Second, there may be several feasible paths and nodes. Choosing different computing nodes to perform computing tasks will greatly affect the transmission delay and computing delay. Therefore, it is also important to choose the right computing node.

III. Routing Problem Formulation

In this section, we formulate the VR routing problem in GCC as an Integer Linear Programming (ILP). Some related constraints have been divided into communication, computing and delay constraints.

A. Communication constraints

Before presenting the constraints, we introduce a communication binary variable to better characterize the usage of links. Denote with $x_{uv}^i \in \{0,1\}$, where $x_{uv}^i = 1$ means that the the VR application R_i will be transmitted by link uv, and $x_{uv}^i = 0$ otherwise.

1) Flow conservation constraint: For the intermediate nodes, flows into node v equal to the flows out of node v. Thus, we have the flow conservation constraint as shown in (1).

$$\sum_{u\in V} x_{uv}^i f_i = \sum_{u\in V} x_{vu}^i f_i \tag{1}$$

especially, for the source node $\mathcal{S}$ and destination node $\mathcal{D}$ we have

$$\sum_{\mathcal{S}\in V} x_{\mathcal{S}v}^i f_i = \sum_{\mathcal{D}\in V} x_{v\mathcal{D}}^i f_i. \tag{2}$$

That is, the flows of arranged VR applications out of the source node $\mathcal{S}$ are equal to the flows into the destination node $\mathcal{D}$.

2) communication capacity constraint: If we arrange a VR application R_i to a physical link uv, the following capacity constraint must be satisfied.

$$\sum_{i=1}^{n} x_{uv}^i b_i \le B_{uv}, \tag{3}$$

where n is the number of VR applications.

3) Communication variable constraint: In the VR routing scenario, we consider that the flows of VR applications can not be spilt up, which is formulated as (4).

$$\sum_{v\in V} x_{uv}^i \le 1 \tag{4}$$

If all the communication variables equal to 0, it means no feasible path can support to transmit the R_i. Conversely, if there is a feasible path, the following constraint must be satisfied.

$$\sum_{uv\in E} x_{uv}^i > 0 \tag{5}$$

B. Computing constraints

Similar to the communication variable, we also define a computing binary variable, which is denoted with $y_v^i \in \{0,1\}$, where $y_v^i = 1$ means that the VR application R_i will be computed by node v, and $y_v^i = 0$ otherwise.

1) Computing capacity constraint: Since the computing capacity of each MEC server is limited, the total computing workload should not exceed the available computing power of the server, which is formulated as

$$\sum_{i=1}^{n} y_v^i c_i \le C_v. \tag{6}$$

2) Computing variable constraints: In this paper, we don't consider the distributed computing. An application must have one and only one computing node to serve its computing demand. Otherwise, we will consider that there is no viable computing nodes for R_i. Thus, we have

$$\sum_{v\in V} y_v^i \le 1. \tag{7}$$

C. Delay constraints

According to the flow size of R_i and available bandwidth of link uv, we can get the communication delay, which is formulated as the (8).

$$d_{uv}^i = \frac{x_{uv}^i f_i}{B_{uv}} \tag{8}$$

Similarly, according to the flow size of VR application and computing power of nodes, we can get the computing delay, which is formulated as

$$d_v^i = \frac{y_v^i f_i}{C_v}. \tag{9}$$

VR applications are dele-sensitive. Therefore, the MTP latency must be controlled below a certain threshold, which is recommended by the IETF standard to be no more than 20ms. In this paper, the dalay demand D_i of VR application R_i will continue to be optimized below this value. The delay constraint can be characterised as

$$\sum_{uv\in E} d_{uv}^i + \sum_{v\in V} d_v^i \le D_i. \tag{10}$$

Authorized licensed use limited to: Beijing Jiaotong University. Downloaded on October 13,2023 at 09:38:04 UTC from IEEE Xplore. Restrictions apply.

D. *Objective*

We focus on finding an end-to-end shortest delay path, which includes the communication delay and computing delay. Therefore, our optimal objective is minimum the total delay for R_i which is denoted as η_i.

$$\begin{aligned} &\min \eta_i = \sum_{uv \in E} d_{uv}^i + \sum_{v \in V} d_v^i \\ &s.t. \ (1) - (10). \end{aligned} \tag{11}$$

Notice that we just need to make sure the values of the binary variables x_{uv}^i and y_v^i and all the constraints are linear. The routing problem is obviously an Integer Linear Programming (ILP) problem.

IV. ALGORITHM

The standard algorithm will take a lof of time to obtain the optimal solution for an ILP. In this section, we propose a Computing Nodes Expanded (CNE) algorithm. We first expand the GCC network according to the computing nodes and then design an improved Dijkstra algorithm to obtain the end-to-end shortest delay path.

For a set of VR applications $R = \{R_1, R_2, \cdots, R_n\}$, we first define the priority for applications before performing them. The priority can be obtained by (12).

$$P(i) = \alpha \frac{f_i}{\max\limits_{j \in [1,n]} \{f_j\}} + \beta \frac{D_i}{\max\limits_{j \in [1,n]} \{D_j\}}, \tag{12}$$

where α, β are weighting factor to adjust the importance of delay and data requirements. The smaller the $P(i)$, the higher the priority. Thus, R_i with smaller data and the higher time sensitive have the higher priority.

Then, for the current VR R_i, according to the flow size f_i and the available bandwidth B_{uv} and computing power C_v, we can obtain the communication delay of each link and the computing delay of each node as shown in Fig. 3(b). In this way, we transform both heterogeneous resources into a delay property.

Next, we extend a virtual node from each computing node respectively such as A', B', C' in Fig. 3(c). A virtual link called computing link will connect the two correlative nodes such as AA', BB', CC' in Fig. 3(c). The value on the computing link is the above computing delay of the node. For example, if R is computed on the node A, the computing delay is 5ms as shown in Fig. 3(b). Therefore, the value of computing link AA' is updated to 5ms. To ensure that an application is only computed by one computing node, we keep the subtree after the original node still behind the extended virtual node. Meanwhile, one extended virtual node corresponds to one virtual destination node whose subscript is the original node such as D_A, D_B, D_C in Fig. 3(c). All virtual destination nodes are connected to the original destination node. The transmission delay between the virtual destination nodes and the original destination node is considered to be 0.

After we expand the GCC, we can easily get the delay matrix (DM) of expanded GCC. We then use the Dijkstra algorithm to find the optimal end-to-end shortest delay path. Some capacity constraints and delay constraints are required to be satisfied during the routing process. Finally, the optimal path will be used to deploy R. The available bandwidth of links on the path and available computing power of the selected computing node will be updated. The pseudocode of the CNE algorithm is shown in Algorithm 1.

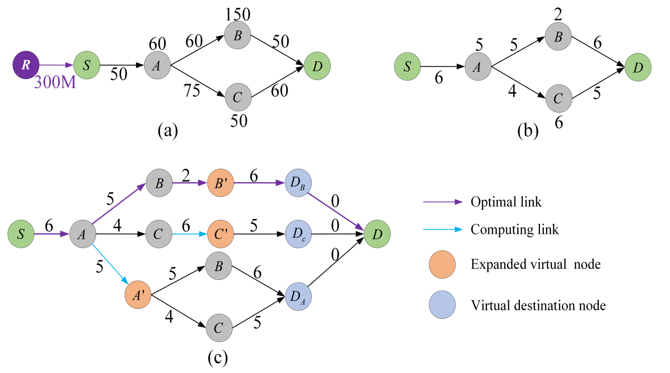

Fig. 3: An example of the CNE algorithm.

The complexity of the algorithm mainly comes from the process of finding the end-to-end shortest delay path and we use the Dijkstra algorithm for the process. As we all know the complexity of Dijkstra algorithm is $O(n^2)$, where n is the number of nodes. After expanding the GCC, the number of nodes in the network will increase dramatically. So in the worst case, there are m nodes in the network, n of which are computing nodes. Each computing node replicates the nodes in the network once during expansion. The number of nodes in the expanded GCC will increase to nm. The complexity of the CNE algorithm is $O(n^2m^2)$.

V. SIMULATION

In this section, numerical results are presented to demonstrate the performance of the proposed model and routing problem.

A. *Experimental setup*

Similar to [5], we verify the CNE algorithm in several common-used network datasets from Survivable Network Design Library [12], finally three realistic topologies are used to evaluate performance comparison in this section. The parameters of each topology is summarized in TABLE I. In experiments, the computing nodes are randomly selected in the topologies, accounting for 40% of the total nodes in this paper. In particular, we believe when computing nodes serve the VR applications, they still have the basic forwarding function.

Specifically, Abilene is created by the Internet2 community and connects regional network aggregation points to provide advanced network capabilities to over 230 Internet2 university, corporate, and affiliate member institutions in the US. It consists of 12 nodes and 30 links, where 5 nodes are selected as the computing nodes. Nobel-EU and Nobel-Germany is

Authorized licensed use limited to: Beijing Jiaotong University. Downloaded on October 13,2023 at 09:38:04 UTC from IEEE Xplore. Restrictions apply.

algorithm 1 CNE Algorithm

```
Input: $GCC = \{(V, E, B_{uv}, C_v)|u, v \in V, uv \in E\}$, $R = \{R_1, R_2, \cdots R_n\}$.
Output: End-to-End shortest delay path and delay value.
1: Init: $x_{uv}^i = 0$, $y_v^i = 0$.
2: Sort $R_i$ according to (12).
3: for $i = 1$ to $n$ do
4:     for $uv \in E$ do
5:         Calculating communication delay according to (8).
6:         if $\sum_{i=1}^{n} x_{uv}^i b_i \geq B_{uv}$ then
7:             $d_{uv}^i = \infty$.
8:         end if
9:     end for
10:    for $v \in V$ do
11:        Calculating computing delay according to (9).
12:        if $\sum_{i=1}^{n} y_v^i c_i \geq C_v$ then
13:            $d_v^i = \infty$.
14:        end if
15:    end for
16:    Expand the GCC according to computing nodes.
17:    Get the DM of the expanded GCC.
18:    Seek the shortest path $SP_i$ in DM by Dijkstra algorithm and the shortest delay $\eta_i$;
19:    if $\eta_i > D_i$ then
20:        return Reject the $R_i$.
21:    else
22:        $x_{uv}^i = 1, uv \in E_{SP_i}$.
23:        $y_v^i = 1$, $v$ is the selected computing node.
24:        return $SP_i$, $\eta_i$.
25:    end if
26: end for
```

originally defined in the COST 266 European project. The former consists of 28 nodes and 82 links, where 11 nodes are selected as the computing nodes. The latter consists of 17 nodes and 52 links, where 7 nodes are selected as the computing nodes.

TABLE I: Parameters of each topology

Topology Name	Total nodes	Total links	Computing nodes
Abilene	12	30	5
Nobel-EU	28	82	11
Nobel-Germany	17	52	7

In this work, we use MATLAB for algorithm performance analysis. The specific parameters are shown in TABLE II. Specifically, the available computing compower of cloud computing data center is set in $[1000, 4000]$GOPS. Since VR applications are delay-sensitive services, the weight factor β of delay requirements is set to 0.6, which is larger than α during the priority ranking. To better capture the CNE algorithm advantages, we define the VR applications delay satisfaction ratio (DSR) as shown in (13), which is the ratio of VR applications Q whose end-to-end delay does not exceed the delay demand D to all requested VR applications n.

$$DSR = \frac{Q}{n} \tag{13}$$

TABLE II: Simulation Parameters

Parameter	Value
C_{p^h,q^h}	$[10, 40]G$
f	$[100, 300]$Mb
D	20ms
b	$[5, 10]$Gbps
c	$[5, 10]$GOPS
n	$[5, 50]$
α, β	$0.4, 0.6$

B. Experimental results

In order to better illustrate the advantages of our algorithm, we compare the proposed algorithm with the following three algorithms:

- Two-stage algorithm [6], which adopted an improved Dijkstra algorithm to calculated the k paths with the shortest transmission delay in the first stage and selected the node in these shortest path which had the minimum computing delay.
- SameMEC algorithm [3], [4], which also considered the shortest transmission delay path while ignoring the difference among the computing nodes.
- Basic algorithm [2], which roughly handled all computational requirements in the cloud computing data center and then sent the results to the users.

Fig. 4 illustrates the average end-to-end delay by using different algorithm in three different network. From the Fig. 4, we can see that compared with other algorithms, the average end-to-end delay of our algorithm is the lowest, because our algorithm can always find a path with the minimum sum of computing delay and transmission delay. The two-stage algorithm is sometimes close to the performance of our algorithm as shown in Fig. 4 Nobel-EU, and sometimes close to the performance of the sameMEC algorithm. That's because if the computing node with the shortest computing delay is just on the shortest communication delay path, the two-stage algorithm is consistent with our algorithm. If the node is not on the shortest communication delay path, the computation request may be served by other computing nodes. After averaging multiple experiments, it is close to the result of the sameMEC algorithm, because in the sameMEC algorithm, the computing power of computing nodes is all set to the average value of the compute nodes in the CNE algorithm. The sameMEC algorithm rarely selects an optimal computing node, so its result is higher than our algorithm. The basic algorithm always achieves the minimum computing delay, but a long path still leads to a large total delay, which is difficult to meet

Authorized licensed use limited to: Beijing Jiaotong University. Downloaded on October 13,2023 at 09:38:04 UTC from IEEE Xplore. Restrictions apply.

the delay requirements of applications. Specifically, compared with the two-stage algorithm, the sameMEC algorithm and the basic algorithm, the CNE algorithm can reduce the total delay by 14.4%, 23.1% and 42.9% respectively.

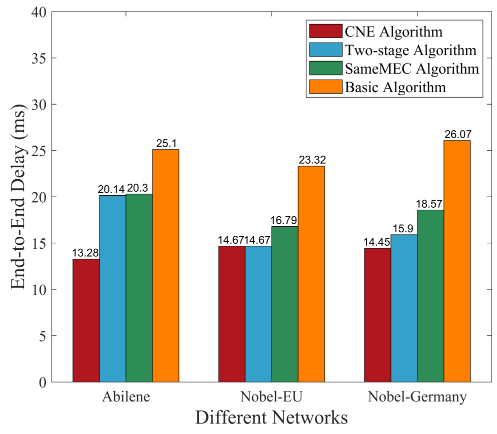

Fig. 4: End-to-end delay by different algorithm in three different networks.

As the number of applications increasing, the DSR shows a downward trend, but occasionally shows an upward trend in some cases, which is shown in Fig. 5. This is because as the number of applications increases, the bandwidth and computing power consumption increases, and the available bandwidth and computing power resources decrease which leads to the longer execution time of subsequent applications. However, due to the different application data size, bandwidth demands, computing power demands and other requirements, in some cases, the number of applications whose delay is satisfied will increase, so there will be a partial increase. Specifically, compared with the two-stage algorithm, the sameMEC algorithm and the basic algorithm, the CNE algorithm can increase the DSR by 13.3%, 17.1% and 23.3% respectively.

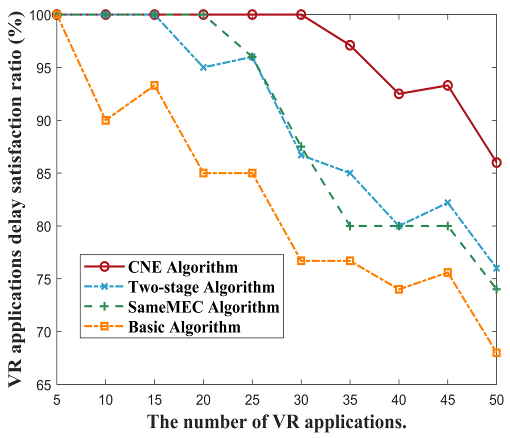

Fig. 5: VR applications delay satisfaction ratio.

VI. CONCLUSIONS

In this work, we have studied the routing problem form extraction to rendering of VR applications. We analyze a typical VR service process and present a GCC model. Then we formulate the VR routing problem in GCC into an ILP problem. Some related constraints which have been divided into communication, computing and delay constraints are also described in detail. To solve the ILP problem, we design an CNE algorithm, which allows us to always find the path with the minimum sum of computing delay and communication delay. We demonstrate with numerical experiments that the proposed algorithm can not only reduce the total delay but also improve the application delay satisfaction. By introducing time slots, the conflict between network and computing can be avoided and the deterministic delay can be realized, which will be further studied.

REFERENCES

[1] Ejder Bastug, Mehdi Bennis, Muriel Medard, and Merouane Debbah. Toward interconnected virtual reality: Opportunities, challenges, and enablers. *IEEE Communications Magazine*, 55(6):110–117, 2017.

[2] M. Chen, W. Saad, C. Yin, and Merouane Debbah. Data correlation-aware resource management in wireless virtual reality (vr): An echo state transfer learning approach. *Post-Print*, pages 1–1, 2019.

[3] X. Yang, Z. Chen, K. Li, Y. Sun, and H. Zheng. Optimal task scheduling in communication-constrained mobile edge computing systems for wireless virtual reality. In *2017 23rd Asia-Pacific Conference on Communications (APCC)*, 2017.

[4] Yaping Sun, Zhiyong Chen, Meixia Tao, and Hui Liu. Communication, computing and caching for mobile vr delivery: Modeling and trade-off. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6, 2018.

[5] Xueying Han, Yuhan Zhao, Ke Yu, Xiaohong Huang, Kun Xie, and Hua Wei. Utility-optimized resource allocation in computing-aware networks. In *2021 13th International Conference on Communication Software and Networks (ICCSN)*, pages 199–205, 2021.

[6] X. Liu, Y. Wang, and Y. Liu. QFR: A qoe-driven fine-grained routing scheme for virtual reality video streaming over sdn. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020.

[7] Xiao Yang, Zhiyong Chen, Kuikui Li, Yaping Sun, Ning Liu, Weiliang Xie, and Yong Zhao. Communication-constrained mobile edge computing systems for wireless virtual reality: Scheduling and tradeoff. *IEEE Access*, 6:16665–16677, 2018.

[8] Peng, Wang, Hongyan, Li, and Binbin Chen. FL-Task-aware Routing and Resource Reservation over Satellite Networks. *IEEE Global Communications Conference (GLOBECOM)*, 2022.

[9] Lei Liu, Ming Zhao, Miao Yu, Mian Ahmad Jan, Dapeng Lan, and Amirhosein Taherkordi. Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–14, 2022.

[10] Junhuai Li, Ruijie Wang, and Kan Wang. Service function chaining in industrial internet of things with edge intelligence: A natural actor-critic approach. *IEEE Transactions on Industrial Informatics*, 19(1):491–502, 2023.

[11] Lei Liu, Jie Feng, Qingqi Pei, Chen Chen, Yang Ming, Bodong Shang, and Mianxiong Dong. Blockchain-enabled secure data sharing scheme in mobile-edge computing: An asynchronous advantage actorcritic learning approach. *IEEE Internet of Things Journal*, 8(4):2342–2353, 2021.

[12] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski. Sndlib 1.0survivable network design library. *Netw.*, 55(3):276C286, may 2010.

Authorized licensed use limited to: Beijing Jiaotong University. Downloaded on October 13,2023 at 09:38:04 UTC from IEEE Xplore. Restrictions apply.