Joint Caching and Computing Resource Reservation for Edge-Assisted Location-Aware Augmented Reality

Yingying Pei*, Mushu Li≀, Huaqing Wu†, Qiang Ye‡, Conghao Zhou*, Shisheng Hu*, and Xuemin (Sherman) Shen*
* Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada
≀ Department of Electrical, Computer, and Biomedical Engineering, Toronto Metropolitan University, Canada
† Department of Electrical and Software Engineering, University of Calgary, Calgary, AB, Canada
‡ Department of Computer Science, Memorial University of Newfoundland, St. John's, NL, Canada
Email: {y32pei, c89zhou, s97hu, sshen}@uwaterloo.ca, mushu1.li@ryerson.ca, huaqing.wu1@ucalgary.ca, qiangy@mun.ca

***Abstract*—In this paper, we investigate joint caching and computing resource reservation for supporting location-aware augmented reality (AR) applications in an edge-assisted two-tier radio access network. We aim at minimizing the caching and computing resource consumption while satisfying the AR service delay requirement. Specifically, to capture the spatio-temporal AR service dynamics, the resource consumption minimization problem is formulated as a long-term stochastic optimization problem. Due to the time-varying service demands and tightly coupled multi-resource reservation decisions, we propose a novel resource reservation algorithm based on the Lyapunov optimization technique to solve the problem. We first transform the original long-term problem into multiple one-shot optimization problems, each of which is then solved by our designed iterative algorithm in an online manner. Simulation results demonstrate that the proposed algorithm can significantly reduce the overall resource consumption compared to benchmark algorithms.**

I. Introduction

The sixth-generation (6G) networks are envisioned to support myriads of new applications, including Industrial Internet-of-Things, vehicle-to-everything, extended reality, etc. [1]. Augmented reality (AR), as one of the representative applications in 6G, provides users with interactive and immersive experiences by overlaying virtual information onto real-world environments. In AR applications, hardware sensors (e.g., cameras) first capture real-world environments and generate video frames [2]. Then, the captured video frames are integrated with virtual information (e.g., texts, 2D videos, or 3D models), in which real-time image processing (e.g., object detection, depth estimation) is required to create a truly immersive and engaging AR experience.

However, performing image processing tasks is challenging due to the following reasons. First, state-of-the-art image processing techniques for AR typically use large deep neural network (DNN) models. Running these computation-intensive models on resource-constrained mobile AR devices (e.g., AR glasses or handheld mobile phones) would generate excessive heat and quickly discharge their batteries. Second, the virtual information to be overlaid onto the captured frames is location-aware, i.e., highly dependent on the location where the frames are captured. For example, in AR tourism, a traveler can view a historical landmark with augmented information about its history by using AR glasses. The location-aware data are usually stored in the databases of computing servers [3], if the image processing tasks are processed on local devices, the location-aware data need to be retrieved from computing servers frequently, which results in significant communication delay and overheads.

To support real-time location-aware AR applications, a feasible way is to offload AR tasks to computing servers near the edge of the network, i.e., edge computing. Edge computing is a computing paradigm that deploys servers connected to wireless base stations or access points to process tasks offloaded from nearby mobile users. In edge-assisted AR, computing and caching resources at edge servers can be leveraged to process AR tasks (e.g., conducting DNN inference with low latency) and cache location-aware data. Despite the potential of edge computing for supporting AR applications, there are still some research challenges that need to be overcome.

First, the computing demands from AR devices can vary spatially and temporally. Correspondingly, the caching policies among edge servers should be adjusted dynamically to maximize resource utilization. Some existing works on edge caching determined content placement policies with a fixed caching size given the spatially distributed service demands and fixed content popularity, with the objective of maximizing system utility [4] or maximizing the total content hit rate [5]. In addition, works [6], [7] adjusted the content placement policies to adapt to time-varying content popularity under the limited caching size. These existing content placement policies with fixed caching sizes are not suitable for location-aware applications, which require dynamic caching size adjustment to adapt to the spatially and temporally varying service demands.

The second challenge is the complex decision-making in multi-resource management. The performance of AR applications is jointly affected by the task offloading, and the amount of multiple resources among edge servers. Obtaining globally optimized decisions every few milliseconds is not feasible due to the extensive signaling exchange among multiple edge servers [8]. Therefore, a scalable approach, i.e., resource reservation, can be used to tackle this challenge and support location-aware AR services. In this approach, network resources are proactively determined on large time scales (e.g., several minutes to hours) to support future service demands, thus simplifying the real-time resource allocation for individual AR devices. Existing works in [9], [10] focused on

resource reservation, e.g., joint radio and computing resource reservations, for satisfying the requirements of different network services. In location-aware AR applications, however, the caching resource reservation, i.e., cache size deployment, is highly related to the spatial distribution of service demands and task offloading decisions. Therefore, the coupling relations among caching, computing, and radio resource reservation require further investigation.

To address the aforementioned challenges, we investigate multi-resource reservation for location-aware AR applications while adapting to spatio-temporal service demands. The objective is to minimize the resource consumption for AR applications while meeting an average AR delay requirement. To capture the dynamic service demands from different locations, a stochastic optimization problem is formulated to determine the task offloading probability, caching, computing, and radio resource reservation over all time slots. The major contributions of this paper are as follows:

1) We tailor a resource reservation algorithm for AR applications, where the caching, computing, and radio resources are jointly reserved to adapt to the spatio-temporal varying location-aware service demands.
2) The proposed algorithm can solve the resource consumption minimization problem for AR applications in an online manner based on real-time service demands.

II. System Model

A. Network Model

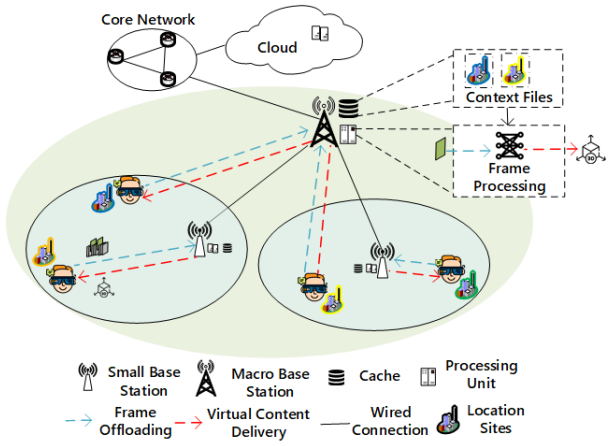

Fig. 1. A two-tier RAN to support location-aware AR services

As shown in Fig. 1, we consider a two-tier radio access network (RAN) to support location-aware AR applications (e.g., AR tourism) for multiple AR devices. A macro base station (MBS) is deployed in the upper network tier and underlaid by I non-overlapped small base stations (SBSs). Denote the set of all BSs by $\mathcal{I} = \{0, 1, 2, ..., I\}$, and let index 0 denote the MBS and index $i \in \mathcal{I}\backslash\{0\}$ denote SBS i. Each BS is co-located with an edge server which is capable of computing and caching. Under the service coverage of SBS i, a set of location sites (e.g., tourist attractions) is selected by the AR service provider and deployed with augmented services. Each AR device is covered by both the MBS and one SBS. The AR devices continuously generate video frames. Keyframes will be offloaded to either the MBS or the SBS that covers the corresponding device for real-time frame processing and virtual content generation. For brevity, a computing task refers to the process of processing and generating virtual content from a keyframe. Let the AR service delay be the total time from a keyframe leaving the AR device until the corresponding virtual content arrives, which should be less than a predefined threshold, denoted by τ, to ensure an immersive AR experience.

Processing the location-aware computing tasks requires location-aware context data, which can be stored at edge servers in advance. Due to the proximity to AR devices, SBSs are used as the primary place for computing AR tasks. Therefore, each SBS server stores whole context files related to the location sites under its coverage, while the MBS server stores partial context files to assist in task processing when traffic surges. To fully utilize the caching resources (we use caching and storage interchangeably) reserved at the MBS, both caching and computing resources should be reserved at the MBS server due to their coupling relation. Furthermore, due to the wide service coverage, the MBS has to process computing tasks for other applications. Therefore, we aim to minimize the total caching and computing resources reserved for the AR application at the MBS server.

A network controller is located at the MBS, making decisions in fixed-duration time slots, indexed by t and $t \in \mathcal{T} = \{1, 2, ..., T\}$. Decisions include radio resource reservation among BSs, computing and caching resource reservation at the MBS, and task offloading among SBSs. At each time slot t, the number of tasks generated from each AR device is assumed to follow a Poisson process. Thus, the aggregated number of generated tasks under the service coverage of SBS i also follows a Poisson process [9], and the generation rate is denoted by $\lambda_i(t)$. Furthermore, due to the stochastic task generation, a probabilistic task offloading policy is adopted, where $\alpha_i(t) \in [0, 1]$ is the probability of tasks generated under the service coverage of SBS i to be offloaded to the SBS i.

B. Communication Model

The total system spectrum, denoted by B, is divided into M orthogonal subchannels, and they are grouped into two parts:

$$M = S_0(t) + S_1(t), \tag{1}$$

where $S_0(t)$ is the number of subchannels reserved for the MBS at slot t, while all SBSs reuse the remaining $S_1(t)$ subchannels to exploit the resource multiplexing gain under acceptable inter-cell interference level [10]. Assume that each offloaded frame has a fixed and identical size, denoted by χ (in the unit of bits). Let $R_i(t)$ (in the unit of Mbps) represent the average uplink transmission rate from AR devices that are under the coverage of BS i to BS i. The average uplink

transmission time of one task offloaded to BS i, denoted by $d_i^T(t)$, is given by

$$d_i^T(t) = \frac{\chi \beta_i(t)}{S_i(t) R_i(t)}, \tag{2}$$

where $S_i(t) = S_1(t)$ if $i \in \{1, 2, ..., I\}$, and $\beta_i(t)$ denotes the task arrival rate at BS i, given by

$$\beta_i(t) = \begin{cases} \lambda_i(t)\alpha_i(t), & i = 1, 2, ..., I \\ \sum_{j=1}^{I} \lambda_j(t)(1 - \alpha_j(t)), & i = 0. \end{cases} \tag{3}$$

Here we do not consider the downlink result delivery since the generated virtual information has a small data size, and the downlink transmission rate is generally much higher than the uplink ones [11].

C. Caching Model

Let $\boldsymbol{C}_i = \{1, ..., C_i\}$ and $\boldsymbol{p^i}(t) = \{p_1^i(t), ..., p_{C_i}^i(t)\}$ be the set of context files related to the locations covered by SBS i and the corresponding file popularity, where c_i is the file index and $p_{c_i}^i(t)$ is the probability that the file c_i will be used for computing at slot t. Note that the files are stored in descending order according to popularity. Let $\boldsymbol{q}(t) = [q_1(t), q_2(t), ..., q_I(t)]$ denote the caching decision at the MBS, where $q_i(t) \in \mathbb{Z}$ is the number of context files selected from set $\boldsymbol{C}_i$. Given the caching decision $\boldsymbol{q}(t)$, the MBS always caches the most popular files for maximal resource utilization [6]. In other words, for the context files related to SBS i, the top $q_i(t)$ most popular context files are selected from set $\boldsymbol{C}_i$ and then stored at the MBS server. We consider that all files have the same file size (in the unit of bits), denoted by F. Then, the total caching space required at the MBS can be calculated as

$$U_c(t) = F \sum_{i=1}^{I} q_i(t). \tag{4}$$

When a computing task is generated under the coverage of SBS i and offloaded to the MBS, the corresponding context files may not be cached at the MBS in advance and need to be retrieved from SBS i upon needed. Let $P_i(t)$ denote the retrieval probability, which can be calculated as

$$P_i(t) = 1 - \sum_{c_i=1}^{q_i(t)} p_{c_i}^i(t). \tag{5}$$

Note that a retrieved context file will not be cached in the MBS after processing the corresponding task since the caching space and caching contents are proactively determined.

Denote the wired transmission delay of one context file from any SBS to the MBS by d_0, we can calculate the expected accumulated retrieval delay at slot t as

$$d^R(t) = \sum_{i=1}^{I} \lambda_i(t)(1 - \alpha_i(t)) P_i(t) d_0. \tag{6}$$

D. Computing Model

Let $L_i(t)$ (in the unit of CPU cycles per second) be the computing capacities of BS i, where $L_0(t)$ is a decision variable at slot t while $L_i(t)$ ($i \in \{1, 2, ..., I\}$) does not change with t. Given that each computing task requires O CPU cycles, the computing service rate at BS i, denoted by $\mu_i(t)$, is given by

$$\mu_i(t) = \frac{L_i(t)}{O}. \tag{7}$$

The arrived tasks at BS i are placed in a waiting queue until being processed. Since the computation intensity per task is identical and the task processing rate is deterministic, we model the task computing process at each edge server as an M/D/1 queue [12], and hence the average waiting and processing delay of one task at BS i is given by

$$d_i^C(t) = \frac{1}{\mu_i(t)} + \frac{\beta_i(t)}{2\mu_i(t)(\mu_i(t) - \beta_i(t))}. \tag{8}$$

Here the constraint of

$$\frac{\beta_i(t)}{\mu_i(t)} \leq 1, \forall i \in \mathcal{I}, \tag{9}$$

should hold to ensure the stability of the task computing queue at BS i.

E. Problem Formulation

With the average task transmission delay, i.e., $d_i^T(t)$, accumulated retrieval delay, i.e., $d^R(t)$, and average task waiting and processing delay, i.e., $d_i^C(t)$, we can obtain the average service delay of an AR task at time slot t, denoted by $d(t)$, as follows

$$d(t) = \frac{\sum_{i=0}^{I} \beta_i(t)(d_i^T(t) + d_i^C(t)) + d^R(t)}{\sum_{i=1}^{I} \lambda_i(t)}. \tag{10}$$

As mobile AR users may move across multiple tourist attractions, the service demands among BSs vary spatially and temporally. Therefore, it is crucial to adjust the multi-resource reservation decision dynamically to ensure a satisfactory quality of service for AR users. Our objective is to minimize the long-term resource consumption, i.e., the weighted sum of caching and computing resources reserved at the MBS, while ensuring that the average service delay does not exceed a predefined threshold τ. The problem is formulated as follows:

$$\begin{aligned} P_0: \min_{\boldsymbol{\alpha}(t), \boldsymbol{\mathcal{S}}(t), \boldsymbol{q}(t), L_0(t), t \in \mathcal{T}} \quad & \frac{1}{T} \sum_{t=1}^{T} (U_c(t) + \gamma L_0(t)) \\ \text{s.t.} \quad & (a)\ \frac{1}{T} \sum_{t=1}^{T} d(t) \leq \tau \\ & (b)\ 0 \leq \alpha_i(t) \leq 1, \forall i \in \mathcal{I} \setminus \{0\} \\ & (c)\ 0 \leq L_0(t) \leq L_0^{max} \\ & (d)\ 0 \leq q_i(t) \leq C_i, \forall i \in \mathcal{I} \setminus \{0\} \\ & (1), (9), \end{aligned} \tag{11}$$

where γ weights the relative cost of computing resource compared to the caching resource, $\mathcal{S}(t) = [S_0(t), S_1(t)]$ is the radio resource reservation decision at slot t, and L_0^{max} is the maximum computing capacity of the MBS server. We jointly determine caching and computing resource reservation at the MBS, i.e., $\boldsymbol{q}(t)$ and $L_0(t)$, the radio resources reserved among BSs, i.e., $\mathcal{S}(t)$, and task offloading probability, i.e., $\boldsymbol{\alpha}(t)$, to minimize the long-term resource consumption. Constraint (11a) ensures that the average long-term service delay should not exceed a predefined value, i.e., τ. Constraint (11b) ensures that the task offloading decision under each SBS is feasible. Constraints (11c), (11d), and (1) are resource capacity constraints to guarantee that the total amount of reserved subchannels, computing capacity, and caching space cannot exceed the amount of resources available at BSs. Constraint (9) ensures the stability of computing queues at all BSs.

III. Lyapunov-Based Online Resource Reservation

Due to the coupling relation among decisions at different time slots, finding the globally optimal solution to P_0 is challenging. To obtain the optimal solution, we first introduce the Lyapunov optimization technique to transform the original long-term problem into multiple one-shot optimization problems. Then, an iterative algorithm is designed to solve the one-shot optimization problem.

A. Problem Transformation

The key idea of the Lyapunov optimization technique is to satisfy the time-average constraints by maintaining the stability of queues.

1) Virtual Queues: We first define a virtual queue $\mathbf{Q}$, which evolves over time slots $t \in \{1, 2, ..., T\}$. The evolution of the virtual queue can be modeled as

$$Q(t+1) = \max\{Q(t) + d(t) - \tau, 0\}, \tag{12}$$

where $Q(0) = 0$. The virtual queue length $Q(t)$ indicates that until the current slot t, how good or bad the average delay performance is compared with the required delay τ. From [13], the virtual queue is *rate stable* if $\lim_{t\to\infty} \frac{Q(t)}{t} = 0$ with probability 1.

Lemma 1. *By enforcing the stability of $Q(t)$, constraint (11a) can be satisfied.*

Proof. According to Eq. (12), we have $Q(t+1) \geq Q(t) + d(t) - \tau$. Subtracting $Q(t)$ from both sides of the inequality, we have $Q(t+1) - Q(t) \geq d(t) - \tau$. Then, we sum up the equations among all slots and let both sides of the equation be divided by T. We have

$$\frac{Q(T) - Q(0)}{T} \geq \frac{1}{T}\sum_{t=0}^{T-1} d(t) - \tau. \tag{13}$$

By enforcing the stability of $Q(t)$, we have $\lim_{T\to\infty} \frac{Q(T)}{T} = 0$ with probability 1. Therefore, given that $Q(0) = 0$, we have $0 \geq \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} d(t) - \tau$. Thus, constraint (11a) can be satisfied by maintaining queue stability in the long run. □

2) Queue Stability: To maintain queue stability, we first introduce the Lyapunov function and Lyapunov drift. A quadratic Lyapunov function, i.e., $Y(Q(t))$, is defined as [13]

$$Y(Q(t)) = \frac{1}{2}Q^2(t). \tag{14}$$

This function is always non-negative, and it is equal to zero if and only if $Q(t)$ is zero. Next, we define the one slot conditional Lyapunov drift, denoted by $\Delta(t)$, as

$$\Delta(t) = \mathbb{E}[Y(Q(t+1)) - Y(Q(t)) \mid Q(t)], \tag{15}$$

which indicates the expected change in the Lyapunov function at slot t. According to [13], by minimizing Lyapunov drift $\Delta(t)$ at every slot t, the queue length is consistently reduced, which potentially maintains the queue stability to satisfy constraint (11a).

3) Minimizing the Drift-Plus-Penalty: According to the Lyapunov optimization theorem, we can greedily minimize the drift-plus-penalty to stabilize the queue while optimizing the objective. The drift-plus-penalty $\Pi(t)$ is given by

$$\Pi(t) = \Delta(t) + V\mathbb{E}[U_c(t) + \gamma L_0(t) \mid Q(t)] \tag{16}$$

where V is a non-negative parameter that balances the average service delay and the resource consumption trade-off. Specifically, the larger V we choose, the fewer computing and caching resources are reserved at the MBS, resulting in a longer average service delay. Next, we give an upper bound of the drift-plus-penalty. From Eq. (14) and (15), we have

$$\Delta(t) \leq \frac{1}{2}(d(t) - \tau)^2 + Q(t)(d(t) - \tau). \tag{17}$$

Let Λ be a positive constant that bounds the first term of the above inequality, i.e., $\frac{1}{2}(d(t) - \tau)^2$. Then, we have

$$\Lambda \geq \frac{1}{2}\mathbb{E}[(d(t) - \tau)^2 \mid Q(t)]. \tag{18}$$

Thus, the upper bound for the drift-plus-penalty is

$$\Pi(t) \leq \Lambda + Q(t)(d(t) - \tau) + V(U_c(t) + \gamma L_0(t)). \tag{19}$$

According to [13], minimizing the above right-hand-side expression is equivalent to minimizing $\Pi(t)$. Therefore, P_0 is equivalent to solving the following optimization problem at each time slot t:

$$\begin{aligned} P_1: \min_{\boldsymbol{\alpha}(t), \mathcal{S}(t), \boldsymbol{q}(t), L_0(t)} \quad & Q(t)(d(t) - \tau) + V(U_c(t) + \gamma L_0(t)) \\ \text{s.t.} \quad & (11b), (11c), (11d), (1), (9). \end{aligned}$$

B. Online Algorithm Design

Problem P_1 is still challenging due to the tightly-coupled integer and continuous variables. Given $\boldsymbol{\alpha}(t)$ and $\mathcal{S}(t)$, the computing and caching resource reservation decisions can be easily derived. Therefore, we decompose P_1 into the following two subproblems.

$$\begin{aligned} P_2: \min_{\boldsymbol{\alpha}(t), \mathcal{S}(t)} \quad & d(t) \\ \text{s.t.} \quad & (11b), (1), (9), \end{aligned} \tag{20}$$

and

$$P_3: \min_{\boldsymbol{q}(t), L_0(t)} \quad Q(t)(d(t)-\tau)+V(U_c(t)+\gamma L_0(t)) \quad \text{s.t.} \quad (11c), (11d). \tag{21}$$

P_2 is a radio resource reservation and task offloading subproblem, which can be solved with given $\boldsymbol{q}(t)$ and $L_0(t)$. P_3 is a caching and computing resource reservation subproblem, which can be solved with the result of P_2. Next, we will discuss how to solve P_2 and P_3 in detail and design an algorithm to solve them iteratively.

1) Radio Resource Reservation and Task Offloading: Each SBS i determines $\alpha_i(t)$ in a distributed manner to minimize its transmission delay $\overline{d}_i^T(t)$, given by

$$\overline{d}_i^T(t) = \alpha_i(t) d_i^T(t) + (1-\alpha_i(t)) d_0^T(t). \tag{22}$$

Note that the total number of subchannels is M. Therefore, there are M possible values of $\mathcal{S}(t)$. With any given value of $\mathcal{S}(t)$, $\boldsymbol{\alpha}(t)$ can be easily solved by minimizing Eq. (22).

2) Computing and Caching Resource Reservation: We rewrite P_3 as follows:

$$P_4: \min_{\boldsymbol{q}(t), L_0(t)} \quad h(\mathcal{S}(t), \boldsymbol{\alpha}(t)) + f(L_0(t)) + g(\boldsymbol{q}(t)) \quad \text{s.t.} \quad (11c), (11d). \tag{23}$$

where $h(\mathcal{S}(t), \boldsymbol{\alpha}(t)) = Q(t)\frac{\sum_{i=0}^{I} \beta_i(t) d_i^T(t)}{\sum_{i=1}^{I} \lambda_i(t)}$ represents the communication component in the objective function in P_4, and $f(L_0(t)) = \gamma V L_0(t) + Q(t)\frac{\sum_{i=0}^{I} \beta_i(t) d_i^C(t)}{\sum_{i=1}^{I} \lambda_i(t)}$ is the computing component, and $g(\boldsymbol{q}(t)) = VF\sum_{i=1}^{I} q_i(t) + Q(t)\frac{d^R(t)}{\sum_{i=1}^{I} \lambda_i(t)}$ refers to the caching component. The minimization of P_4 is equivalent to minimizing each component of the objective function separately.

In terms of the communication component, $h(\mathcal{S}(t), \boldsymbol{\alpha}(t))$ is a constant given the result of P_2. For the computing component, due to the convexity of $f(L_0(t))$, the optimal $L_0(t)$ can be obtained by the first-order derivation. We can calculate the first order derivation of $f(L_0(t))$, given by

$$\frac{\mathrm{d}f(L_0(t))}{\mathrm{d}L_0(t)} = V\gamma - \frac{Q(t)O^2}{\sum_{i=1}^{I} \lambda_i(t)} \left[\frac{1}{2L_0^2(t)} + \frac{1}{2(L_0(t)-O\lambda_0)^2} \right]. \tag{24}$$

The optimal value of $L_0(t)$ is discussed in the following two cases:

1) $Q(t) = 0$. The value of (24) is always positive, the optimal value is achieved at $L_0(t) = O\lambda_0$.
2) $Q(t) > 0$. Due to constraint (9), the value of (24) increases with $L_0(t)$ given $L_0(t) \in [O\lambda_0, L_0^{max}]$ when $Q(t) \geq 0$. If $\frac{\mathrm{d}f(L_0(t))}{\mathrm{d}L_0(t)} \mid_{L_0(t)=L_0^{max}} \leq 0$, the optimal $L_0(t)$ can be achieved at $L_0(t) = L_0^{max}$; if $\frac{\mathrm{d}f(L_0(t))}{\mathrm{d}L_0(t)} \mid_{L_0(t)=L_0^{max}} \leq 0$, the optimal $L_0(t)$ can be obtained by letting $\frac{\mathrm{d}f(L_0(t))}{\mathrm{d}L_0(t)} = 0$.

For the caching component, $VF\sum_{i=1}^{I} q_i(t)$ linearly increases as the caching size $F\sum_{i=1}^{I} q_i(t)$ increases. However, when adding a file c_i from SBS i, the benefits of reduced retrieval delay $M_{c_i}^i(t)$ is given by

$$M_{c_i}^i(t) = \frac{Q(t)\lambda_i(t)(1-\alpha_i(t))p_{c_i}^i(t)d_0}{\sum_{i=1}^{I} \lambda_i(t)}, \tag{25}$$

which diminishes as more files are added. Therefore, a file should not be cached if its caching cost is larger than the benefits of reduced retrieval delay. Based on this idea, we design Algorithm 1 to solve P_3. Based on the above analysis, we solve the above two subproblems iteratively to obtain the solution of P_1. Specifically, $L_0(t)$ and $\boldsymbol{q}(t)$ are first set to be 0 and $\mathbf{0}$, respectively, to solve P_2. After obtaining the radio resource reservation decision and the task offloading decision, the results are used to solve P_3 to update $L_0(t)$ and $\boldsymbol{q}(t)$. By iteratively solving P_2 and P_3 until convergence, P_1 can be optimized. The detailed process of the proposed Lyapunov-Based Online Resource Reservation algorithm to solve the original Problem P_0 is shown in Algorithm 2.

Algorithm 1 Computing and Caching Resource Optimization

Input: $\mathcal{S}(t), \boldsymbol{\alpha}(t)$
Compute $L_0(t)$ based on the Eq. (24).
Compute $\psi(t) = VF$.
for all $i \in \{1, 2, ..., I\}$ and $c_i \in \{1, 2, ..., C_i\}$ **do**
  Compute $M_{c_i}^i(t)$
  If $M_{c_i}^i(t) > \psi(t)$, let $a_{c_i}^i(t) = 1$; otherwise $a_{c_i}^i(t) = 0$.
end for
for all $i \in \{1, 2, ..., I\}$ **do**
  Calculate $q_i(t) = \sum_{c_i=1}^{C_i} a_{c_i}^i$.
end for
Output: $\boldsymbol{q}(t), L_0(t)$

Algorithm 2 Lyapunov-Based Online Resource Reservation

Input: $Q(t-1), \lambda_i(t), d(t-1)$
$\boldsymbol{q}(t) = \mathbf{0}$, $L_0(t) = 0$, update $Q(t)$ with Eq. (12).
while $\epsilon \geq 0.01$ **do**
  for $S_0(t) = 0 : M$ **do**
    Compute $\overline{d}_i^T(t)$ and $\boldsymbol{\alpha}(t)$ based on Eq. (22).
  end for
  Find $\{\boldsymbol{\alpha}(t), \mathcal{S}(t)\} = \arg\min_{\{\boldsymbol{\alpha}(t), \mathcal{S}(t)\}} \overline{d}_i^T(t)$.
  Compute $\boldsymbol{q}(t), L_0(t)$ using Algorithm 1.
  Update $d(t)$.
  Calculate the Euclidean distance ϵ between two consecutive delay values $\epsilon = \|d(t) - d(t-1)\|^2$.
end while
Output: $d(t), Q(t), \boldsymbol{q}(t), L_0(t), \boldsymbol{\alpha}(t), \mathcal{S}(t)$

IV. Simulation Results

In this section, we conduct simulations to evaluate the performance of the proposed algorithm. The simulation scenario includes one MBS and two SBSs for AR task processing. Under the coverage of each SBS, several AR devices continuously generate computing tasks, and the average delay requirement τ is 50 ms. The size and computing intensity of

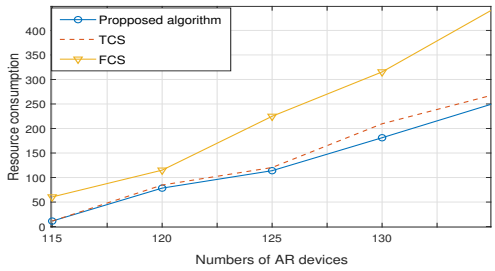

Fig. 2. Resource consumption with various numbers of AR devices in each SBS.

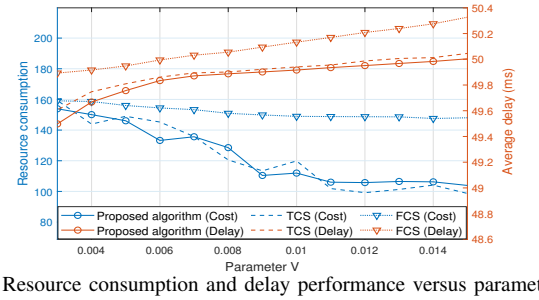

Fig. 3. Resource consumption and delay performance versus parameter V

each task is set to be $\chi = 160$ kbit and $O = 1.6 \times 10^7$ cycles, respectively. In terms of radio resources, there are $M = 10$ subchannels for task offloading, and the data rate for each subchannel follows a Gaussian distribution $\mathcal{N}(9, 3)$. In terms of the wired transmission, the retrieval delay for a file from each SBS to the MBS is $d_0 = 10$ ms. The processing rate for each SBS server is $L_i = 7 \times 10^8$ cycles per second, where $i \in \{1, 2\}$, and the maximum processing rate for the MBS is $L_0^{max} = 3.2 \times 10^9$ cycles per second. For each SBS, the number of associated context files is $C_i = 20$, where $i \in \{1, 2\}$, and the request probability for each file is randomly generated and normalized with a total probability of 1. We choose two benchmark algorithms. The first benchmark, referred to as FCS, keeps a fixed caching size and considers the spatially distributed computing demand. The second benchmark uses a time-varying caching size, which does not consider the spatial-varying computing demands, and we refer to it as TCS.

First, we evaluate resource consumption when using different algorithms. As shown in Fig. 2, the resource consumption increases with the number of AR users for all algorithms, and the proposed algorithm outperforms the benchmark algorithms. The reason is that the proposed algorithm can dynamically adjust the reserved caching size and computing resources based on the spatially and temporally varied service demands to maintain a low resource consumption. Specifically, the proposed algorithm is even more efficient when the number of AR devices increases. As more tasks are generated, dynamically changing the caching and computing resource reservation decisions become more important.

Next, we evaluate the resource consumption and the delay performance trade-off by tuning parameter V. Fig. 3 shows that for all algorithms, the resource consumption (blue lines) decreases as parameter V increases, whereas the average task delay (orange lines) increases. This is because, as V increases, a higher weight is applied to the resource consumption cost. Thus, the one-shot decision tends to use fewer resources while tolerating a longer task processing delay.

V. Conclusion

In this paper, we have studied a multi-resource reservation problem for a two-tier RAN to support AR applications. To adapt to the dynamic service demands, a long-term resource consumption minimization problem has been formulated and solved by our proposed Lyapunov-based online resource reservation algorithm. Simulation results have shown that the proposed algorithm reduces the overall resource consumption while satisfying the delay requirement for AR applications. The proposed solution can also be applied to other location-aware applications generating latency-critical computing tasks. For future work, we will investigate the interplay of resource management schemes in different timescales, where a large-timescale resource reservation correlates with a small-timescale adaptive task offloading for further reducing the average service delay.

References

[1] X. Shen, J. Gao, W. Wu, M. Li, C. Zhou, and W. Zhuang, "Holistic network virtualization and pervasive network intelligence for 6G," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 1–30, 2021.

[2] J. Ren, L. Gao, X. Wang, M. Ma, G. Qiu, H. Wang, J. Zheng, and Z. Wang, "Adaptive computation offloading for mobile augmented reality," *Proc. ACM IMWUT*, vol. 5, no. 4, pp. 1–30, 2021.

[3] Q. Liu, S. Huang, J. Opadere, and T. Han, "An edge network orchestrator for mobile augmented reality," in *Proc. IEEE INFOCOM*, Honolulu, HI, USA, Apr. 2018.

[4] L. Liu, Y. Zhou, J. Yuan, W. Zhuang, and Y. Wang, "Economically optimal MS association for multimedia content delivery in cache-enabled heterogeneous cloud radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 7, pp. 1584–1593, 2019.

[5] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor–critic deep reinforcement learning," *IEEE Internet of Things J.*, vol. 6, no. 2, pp. 2061–2073, 2018.

[6] C. Tang, C. Zhu, H. Wu, Q. Li, and J. J. Rodrigues, "Toward response time minimization considering energy consumption in caching-assisted vehicular edge computing," *IEEE Internet of Things J.*, vol. 9, no. 7, pp. 5051–5064, 2021.

[7] Z. Ning, K. Zhang, X. Wang, L. Guo, X. Hu, J. Huang, B. Hu, and R. Y. Kwok, "Intelligent edge computing in internet of vehicles: a joint computation offloading and caching solution," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2212–2225, 2020.

[8] H. Zhang and V. W. Wong, "A two-timescale approach for network slicing in C-RAN," *Trans. Veh. Technol.*, vol. 69, no. 6, pp. 6656–6669, 2020.

[9] W. Wu, N. Chen, C. Zhou, M. Li, X. Shen, W. Zhuang, and X. Li, "Dynamic RAN slicing for service-oriented vehicular networks via constrained learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2076–2089, 2020.

[10] Q. Ye, W. Shi, K. Qu, H. He, W. Zhuang, and X. Shen, "Joint RAN slicing and computation offloading for autonomous vehicular networks: a learning-assisted hierarchical approach," *IEEE Open J. Vehic. Tech.*, vol. 2, pp. 272–288, 2021.

[11] Y. Hao, M. Chen, L. Hu, M. S. Hossain, and A. Ghoneim, "Energy efficient task caching and offloading for mobile edge computing," *IEEE Access*, vol. 6, pp. 11 365–11 373, 2018.

[12] D. G. Kendall, "Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain," *Ann. Math. Stat.*, vol. 24, no. 3, pp. 338–354, 1953.

[13] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synth. Lect. Commun.*, vol. 3, no. 1, pp. 1–211, 2010.