DIRECT: Distributed Cross-Domain Resource Orchestration in Cellular Edge Computing

Qiang Liu
The University of North Carolina at Charlotte
Charlotte, NC, United States
qliu12@uncc.edu

Tao Han
The University of North Carolina at Charlotte
Charlotte, NC, United States
Tao.Han@uncc.edu

ABSTRACT

Network slicing and edge computing are key technologies to enable compute-intensive applications for vertical industries in 5G. We define cellular networks with edge computing capabilities as cellular edge computing. In this paper, we study the cross-domain resource orchestration solution for dynamic network slicing in cellular edge computing. The fundamental research challenge is from the difficulty in modeling the relationship between the slice performance and resources from multiple technical domains across the network with many base stations and distributed edge servers. To address this challenge, we develop a distributed cross-domain resource orchestration (DIRECT) protocol which optimizes the cross-domain resource orchestration while providing the performance and functional isolations among network slices. The main component of DIRECT is a distributed cross-domain resource orchestration algorithm which is designed by integrating the ADMM method and a new learning-assisted optimization approach. The proposed resource orchestration algorithm efficiently orchestrates multi-domain resources without requiring the performance model of the network slices. We develop and implement the DIRECT protocol in a small-scale prototype of cellular edge computing which is designed based on OpenAirInterface LTE and CUDA GPU computing platforms. The performance of DIRECT is validated through both experiments and network simulations.

CCS CONCEPTS

• **Computing methodologies → Distributed algorithms**;
• **Networks → Network management**; *Mobile networks*;

Mobihoc '19, July 2–5, 2019, Catania, Italy
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6764-6/19/07...$15.00
https://doi.org/10.1145/3323679.3326516

KEYWORDS

Edge Computing, Radio Access Networks, Network Slicing

ACM Reference Format:
Qiang Liu and Tao Han. 2019. DIRECT: Distributed Cross-Domain Resource Orchestration in Cellular Edge Computing. In *The Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc '19), July 2–5, 2019, Catania, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3323679.3326516

1 INTRODUCTION

As the emergence of Internet of Things (IoT) and artificial intelligence (AI), 5G needs to support a multitude of new use cases and services that will place different requirements on the network in terms of functionality, performance, and cost [16, 23]. To serve the new demands, cellular networks have to offer flexible data transmissions as well as computing capabilities. We name cellular networks with edge computing capabilities as cellular edge computing [12, 13]. The main challenge of designing cellular edge computing is how to efficiently meet the diverse requirements imposed by new use cases and services [27].

Network slicing, which enables mobile network operators to create multiple logical networks on top of a common physical network infrastructure, is a promising technology for addressing this challenge [9]. The logical networks can be customized to meet a wide variety of network requirements on performance and functionality. For example, a logical network can be customized to support IoT services that have a large number of devices with low data rates. At the same time, another logical network can be tailored to latency-critical services such as vehicle-to-vehicle communications and smart grid controls. The goal of network slicing is to efficiently utilize the physical network and computing resources to meet the diverse requirements of applications while ensuring functional and performance isolations among network slices. Here, the functional isolation ensures that each network slice can fully control the slice operation; the performance isolation makes sure that the performance of a network slice is independent of other network slices which share the same physical infrastructure.

As a critical technology of 5G, network slicing attracts many research efforts from both academia and industry.

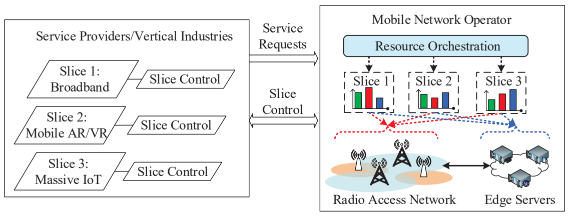

Figure 1: Network slicing in cellular edge computing.

However, most of the existing works present conceptual network slicing frameworks and lack the in-depth discussion of resource orchestration algorithms and realizable system designs [9, 13, 16, 23, 27]. Only a few technical papers discuss resource orchestration solutions [4, 14, 15] and system designs [6, 7] for radio access network slicing and virtualization. These papers provide insights and lay foundations on network slicing in radio access networks. However, these papers only focus on radio resource virtualization in radio access networks and did not investigate how to orchestrate resources from multiple technical domains such as communication and computing resources. In addition, none of these papers discuss how to optimize the utilization of the physical infrastructure under the scenarios of multiple base stations and edge servers with the constraint of performance and functional isolations. In this paper, we propose a distributed cross-domain resource orchestration (DIRECT) protocol to bridge this research gap.

Fig. 1 illustrates the network slicing in cellular edge computing. The service providers or vertical industries make the service requests to create network slices and manage the network slices once they are created. The mobile network operator manages the physical network infrastructure and is responsible to create network slices to support diverse requirements from its customers. Since a network slice in cellular edge computing consists of multiple radio cells and edge servers, the resources allocated to a network slice should be properly distributed among base stations and edge servers to ensure the performance of a network slice and support seamless mobility.

The fundamental research challenge of the cross-domain source orchestration is from the difficulty in modeling the relationship between the slice performance and the multi-domain resource allocations. On the one hand, the performance of a network slice depends on resources from multiple technical domains and the characteristics of the applications supported in the slice. For example, mobile augmented reality (MAR) not only needs communication resources to upload sensed information and download corresponding digital augments but also requires computing resources on the edge server to execute compute-intensive machine learning algorithms to process the sensed information. Since network slices host applications with diverse requirements on different resources, it is impossible to have a slice performance model that correctly reflects the relationship between the slice performance and the resource allocations for all slices. On the other hand, with the functional isolation, each network slice can fully customize the slice operations such as the user scheduling and traffic load balancing. The slice customized control strategies may change the resource demands of a network slice on base stations and edge servers. Hence, the functional isolation makes it impossible for the mobile network operator to model the slice performance without knowing the customized slice control strategies and user profiles of the network slices.

To address this challenge, we design the DIRECT protocol based on the alternating direction method of multipliers (ADMM) method and a new learning-assisted optimization (LAO) approach. We introduce the edge node which is a logic network unit consisting of a physical base station and a certain amount of computing resources, e.g., an edge server. Then, cellular edge computing can be recognized as a collection of interconnected edge nodes. We apply ADMM to decompose the cross-domain resource orchestration into two subproblems. The first subproblem handles the resource orchestrations within an edge node while the second subproblem coordinates the resource allocation among edge nodes. Since the slice performance model is not available, we represent the performance of a slice using a black-box function. Therefore, the first subproblem becomes a black-box optimization problem. We solve this problem by designing a new learning-assisted optimization algorithm that constructs a probabilistic model for the black-box function and iteratively learn the gradients of the function with the observed data for the optimization. The second subproblem is a standard quadratic programming problem, and we solve it based on convex optimization. We implemented the DIRECT protocol in a small-scale system prototype of cellular edge computing that is developed with the OpenAirInterface (OAI) LTE platform [20] and CUDA GPU programming platform [10].

The contributions of this paper are summarized as follows:

- We develop the DIRECT protocol for network slicing in cellular edge computing. DIRECT is a realizable cross-domain resource orchestration protocol that optimizes the performance of network slices while maintaining the functional and performance isolations.
- We design a distributed algorithm based on ADMM and LAO to optimize the network-wide cross-domain resource orchestration in a distributed fashion. By integrating ADMM and LAO, this algorithm efficiently addresses the challenge of the unknown performance model of network slices in the resource orchestration.
- We implement and validate the DIRECT protocol in a small-scale system prototype developed based on the OpenAirInterface (OAI) LTE platform [20] and CUDA

GPU programming platform [10]. We also evaluate the performance of the protocol in network simulations.

2 SYSTEM MODEL

In cellular edge computing, the performance of a network slice depends on resources from multiple technical domains. In this paper, we mainly consider the radio resources and computing resources from a radio access network and edge servers, respectively. On modeling the system, we introduce a logic network unit, edge node, which is composed of a cellular base station and a certain amount of computing resources. Then, cellular edge computing is composed of a collection of edge nodes. To provide seamless mobility and continuous services, network slices need resources from all edge nodes.

Let $\mathcal{I}$, $\mathcal{J}$ and $\mathcal{K}$ be the set of network slices, edge nodes and resources, respectively. Denote $x_{i,j,k}$ as the amount of the kth resource allocated to the ith network slice on the jth edge node, and let $\mathcal{X} = \{x_{i,j,k} | \forall i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{K}\}$ be the set of the resource allocations. Denote $\mathbf{x}_{i,j} = \{x_{i,j,k} | \forall k \in \mathcal{K}\}$ as the set of resources allocated to the ith network slice on the jth edge node. We define the utility function, i.e., performance, of the ith network slice on the jth edge node as $f_{i,j}(\mathbf{x}_{i,j})$. Denote $R^{tot}_{j,k}$ as the total amount of the kth resource on the jth edge node. The service provider pays the mobile network operator for running the network slice according to service level agreement. Denote $\gamma_{j,k}$ and Q^{tot}_i as the unit price of the kth resource on the jth edge node and the total payment of the ith network slice, respectively.

The objective of the mobile network operator is to maximize the sum utility of network slices on all edge nodes. Therefore, the cross-domain resource orchestration problem is formulated as

$$\begin{aligned} &\max_{\{x_{i,j,k} \geq 0\}} && \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} f_{i,j}(\mathbf{x}_{i,j}) \\ &s.t. && \\ &C_1: && \textstyle\sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} x_{i,j,k}\gamma_{j,k} \leq Q^{tot}_i, \forall i \in \mathcal{I}, \\ &C_2: && \textstyle\sum_{i\in\mathcal{I}} x_{i,j,k} \leq R^{tot}_{j,k}, \forall j \in \mathcal{J}, k \in \mathcal{K}. \end{aligned} \tag{1}$$

Here, constraints C_1 ensure that the cost of the resources allocated to a network slice do not exceed the network slice's payment; constraints C_2 restrict that the amount of the kth resource allocated to network slices on the jth edge node should be less than the total amount of the kth resource on the jth edge node for all $j \in \mathcal{J}$ and $k \in \mathcal{K}$. For the sake of simplicity, we let $\gamma_{j,k} = 1, \forall j \in \mathcal{J}, k \in \mathcal{K}$ and simplify constraints C_1 as $\sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} x_{i,j,k} \leq Q^{tot}_i, \forall i \in \mathcal{I}$.

3 DISTRIBUTED RESOURCE ORCHESTRATION

In this section, we present the distributed resource orchestration algorithm that solves the problem in Eq.1. This problem is difficult to solve because all utility functions $f_{i,j}(\mathbf{x}_{i,j})$, $\forall i \in \mathcal{I}, j \in \mathcal{J}$ are unknown, have various mathematical properties, and are coupled by the constraints. To solve this problem, we decompose it to two subproblems using the ADMM method. The first subproblem is to optimize the resource orchestration within an edge node, where the utility functions of the network slices are unknown. This falls into the realm of black box optimization. However, the classic black box optimization methods, e.g., genetic algorithms and pattern search methods, have very high computation complexity and are not appropriate for solving the resource orchestration problem. Therefore, we design a new learning-assisted resource orchestration algorithm to solve the subproblem. The second subproblem is to coordinate the resource orchestration among network slices and can be efficiently solved by convex optimization methods.

3.1 Problem Decomposition

To decompose the problem, we introduce an auxiliary variable $z_{i,j,k}$ and let $\mathcal{Z} = \{z_{i,j,k} | \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall k \in \mathcal{K}\}$. Then, the problem in Eq.1 is equivalent to

$$\begin{aligned} &\max_{\{x_{i,j,k}, z_{i,j,k}\}} && \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} f_{i,j}(\mathbf{x}_{i,j}) \\ &s.t. && \\ &C_1: && \textstyle\sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} z_{i,j,k} \leq Q^{tot}_i, \forall i \in \mathcal{I}, \\ &C_2: && \textstyle\sum_{i\in\mathcal{I}} x_{i,j,k} \leq R^{tot}_{j,k}, \forall j \in \mathcal{J}, k \in \mathcal{K}, \\ &C_3: && x_{i,j,k} = z_{i,j,k}, \forall i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{K}. \end{aligned} \tag{2}$$

After the transformation, the problem has two set of variables, i.e., $\mathcal{X}$ and $\mathcal{Z}$, which are coupled by constraints C_3. Based on the ADMM method, we decompose the problem in Eq. 2 into two subproblems with variables $\mathcal{X}$ and $\mathcal{Z}$, respectively. Here, constraints C_2 apply to the first subproblem whose variables are $\mathcal{X}$, while constraints C_1 apply to the second subproblem whose variables are $\mathcal{Z}$. Hence, we derive the augmented Lagrangian of the problem as

$$\begin{aligned} \mathcal{L}_u = &\sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} f_{i,j}\left(\mathbf{x}_{i,j}\right) \\ &- \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} \frac{\rho}{2} \left\| x_{i,j,k} - z_{i,j,k} + u_{i,j,k} \right\|_2^2, \end{aligned} \tag{3}$$

where $\rho \geq 0$ is a positive constant, and $u_{i,j,k}$ is the scaled dual variable. Here, the augmented Lagrangian incorporates the coupling constraints C_3, and constraints C_1 and C_2 apply when the corresponding subproblems are solved. According to the ADMM method, the above problem is solved by alternatively solving the following subproblems

$$\mathcal{SP}1: \quad x^{(n+1)}_{i,j,k} = \arg \max_{x_{i,j,k} \in C_2} \mathcal{L}_u(x_{i,j,k}, z^{(n)}_{i,j,k}, u^{(n)}_{i,j,k}), \tag{4}$$

$$\mathcal{SP}2: \quad z^{(n+1)}_{i,j,k} = \arg \max_{z_{i,j,k} \in C_1} \mathcal{L}_u(x^{(n+1)}_{i,j,k}, z_{i,j,k}, u^{(n)}_{i,j,k}), \tag{5}$$

and updating the dual variables

$$u^{(n+1)}_{i,j,k} = u^{(n)}_{i,j,k} + (x^{(n+1)}_{i,j,k} - z^{(n+1)}_{i,j,k}). \tag{6}$$

3.2 Resource Orchestration on Edge Node

Since the constraints to $\mathcal{SP}1$ (constraints C_2) only restrict the resource allocation within an edge node, $\mathcal{SP}1$ can be solved independently on each edge node. Therefore, to solve $\mathcal{SP}1$, each edge node addresses the following problem:

$$\begin{aligned} \max_{\{x_{i,j,k}\}} \quad & \sum_{i\in\mathcal{I}} f_{i,j}(\mathbf{x}_{i,j}) \\ & - \sum_{i\in\mathcal{I}} \sum_{k\in\mathcal{K}} \frac{\rho}{2} \left\| x_{i,j,k} - z_{i,j,k} + u_{i,j,k} \right\|_2^2 \\ s.t. \quad C_2: \quad & \textstyle\sum_{i\in\mathcal{I}} x_{i,j,k} \le R_{j,k}^{tot}, \forall k \in \mathcal{K} \end{aligned} \tag{7}$$

where $z_{i,j,k}$ is assumed known. Since $f_{i,j}(\mathbf{x}_{i,j})$, which is the utility of the ith network slice on the jth edge node, is an unknown function, we develop a probabilistic model to represent it and iteratively learn its properties from observed data. Based on the properties, we design a gradient-based optimization algorithm to solve the problem [24].

Since the problem is solved within an edge node, for the sake of simplicity, we omit the subscript j in the math expression when deriving the solution in this section. Hence, $\mathbf{x}_i = \{x_{i,j,k}|\forall k \in \mathcal{K}\}$ and $f_i(\mathbf{x}_i)$ equal to $\mathbf{x}_{i,j} = \{x_{i,j,k}|\forall k \in \mathcal{K}\}$ and $f_{i,j}(\mathbf{x}_{i,j})$, respectively. Given $\mathbf{x}_i$, mobile network operator can observe $y_i = f_i(\mathbf{x}_i) + \epsilon$ which contains Gaussian noises $\epsilon \sim \mathcal{N}(0, \delta^2)$. Let $\mathbf{x}_i^{1:t}$ and $y_i^{1:t}$ as the set of resource allocations and the corresponding observation in t iterations. With the observations $\mathcal{D}_i^{1:t} = \{\mathbf{x}_i^{1:t}, y_i^{1:t}\}$, the posterior distribution of $f_i(\mathbf{x}_i)$ can be expressed as

$$P(f_i(\mathbf{x}_i)|\mathcal{D}_i^{1:t}) \propto P(\mathcal{D}_i^{1:t}|f_i(\mathbf{x}_i))P(f_i(\mathbf{x}_i)). \tag{8}$$

We adopt the Gaussian process (GP) to model the prior distribution of $f_i(\mathbf{x}_i)$ [21]. Hence, $f_i(\mathbf{x}_i)$ can be described as $f_i(\mathbf{x}_i) \sim \mathcal{GP}(\mu(\mathbf{x}_i), c(\mathbf{x}_i, \mathbf{x}_i'))$ where $\mu(\mathbf{x}_i)$ and $c(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2}||\mathbf{x}_i - \mathbf{x}_j||^2)$ are the mean function and covariance function, respectively.

Given a resource allocation $\mathbf{x}_i^*$, the posterior distribution at the tth iteration can be derived as

$$P(f_i(\mathbf{x}_i^*)|\mathcal{D}_i^{1:t}, \mathbf{x}_i^*) \sim \mathcal{N}(\mu(\mathbf{x}_i^*), \sigma_i^2(\mathbf{x}_i^*)), \tag{9}$$

where

$$\mu(\mathbf{x}_i^*) = \mathbf{c}^T[\mathbf{C} + \delta^2 I]^{-1} y_i^{1:t}. \tag{10}$$

$\sigma_i^2(\mathbf{x}_i^*)$ can be derived as

$$\sigma_i^2(\mathbf{x}_i^*) = c(\mathbf{x}_i^*, \mathbf{x}_i^*) - \mathbf{c}^T[\mathbf{C} + \delta^2 I]^{-1}\mathbf{c} \tag{11}$$

where $\mathbf{c} = [c(\mathbf{x}_i^*, \mathbf{x}_i^1), c(\mathbf{x}_i^*, \mathbf{x}_i^2), \cdots, c(\mathbf{x}_i^*, \mathbf{x}_i^t)]$ and

$$\mathbf{C} = \begin{bmatrix} c(\mathbf{x}_i^1, \mathbf{x}_i^1) & \cdots & c(\mathbf{x}_i^1, \mathbf{x}_i^t) \\ \vdots & \ddots & \vdots \\ c(\mathbf{x}_i^t, \mathbf{x}_i^1) & \cdots & c(\mathbf{x}_i^t, \mathbf{x}_i^t) \end{bmatrix}. \tag{12}$$

Based on the posterior distribution, we define the predictive gradient of $f_i(\mathbf{x}_i^t)$ as

$$\nabla \bar{f}_i(\mathbf{x}_i^t) := \frac{1}{\tau} \begin{bmatrix} \mu([\mathbf{x}_{i,1}^t + \tau, \mathbf{x}_{i,2}^t, ..., \mathbf{x}_{i,K}^t]) - y_i^t \\ \mu([\mathbf{x}_{i,1}^t, \mathbf{x}_{i,2}^t + \tau, ..., \mathbf{x}_{i,K}^t]) - y_i^t \\ \vdots \\ \mu([\mathbf{x}_{i,1}^t, \mathbf{x}_{i,2}^t, ..., \mathbf{x}_{i,K}^t + \tau]) - y_i^t \end{bmatrix}, \tag{13}$$

where $\mathbf{x}_{i,k}^t$ is the amount of the kth resource allocated to the ith network slice at the tth iteration, and τ is small positive constant.

With the predictive gradient, we solve the resource orchestration problem on the edge node using the proximal gradient descent method [3]. According to this method, the resource allocation to the ith slice is updated as follow

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \lambda \cdot (\nabla \bar{f}_i(\mathbf{x}_i^t) - \rho(\mathbf{x}_i^t - \mathbf{z}_i^t + \mathbf{u}_i^t)), \tag{14}$$

where $\lambda = [\lambda_1, \lambda_2, ..., \lambda_K]^T$ are appropriate step sizes, and $\cdot$ is the dot product operator. To prevent the updated resource allocation $\mathbf{x}_i^{t+1}, \forall i \in \mathcal{I}$ from violating resource constraints on the edge node, we project $\mathbf{x}_i^{t+1}, \forall i \in \mathcal{I}$ into a bounded domain that satisfies the constraint of each type resource, i.e., constraints C_2 in the problem. We define $P_\Omega(x) = \arg\min_{y\in\Omega} ||x - y||^2$ as the Euclidean projection of x on bounded domain Ω. Denote $\mathbf{x}_k^{t+1} = \{\mathbf{x}_{i,k}^{t+1}|\forall i \in \mathcal{I}\}$ as the amount of the kth resource allocated to all network slices on the edge node. Then, $\mathbf{x}_k^{t+1}$ is projected as

$$\mathbf{x}_k^{t+1} = P_{\Omega_k}\left(\mathbf{x}_k^{t+1}\right), \tag{15}$$

where Ω_k is the bounded domain of the kth resource. The algorithm stops when the resource allocations satisfy $||\mathbf{x}_i^{t+1} - \mathbf{x}_i^t|| \le \eta, \forall i \in \mathcal{I}$. The pseudo code of the algorithm on the edge node is shown in Alg. 1.

3.3 Resource Orchestration on Controller

The controller is responsible to solve $\mathcal{SP}2$. Since $\mathcal{X}$ is assumed known, $\mathcal{SP}2$ can be equivalently transformed to

$$\begin{aligned} \min_{\{z_{i,j,k}\}} \quad & \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} \frac{\rho}{2} \left\| x_{i,j,k} - z_{i,j,k} + u_{i,j,k} \right\|_2^2 \\ s.t. \quad C_1: \quad & \textstyle\sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} z_{i,j,k} \le Q_i^{tot}, \forall i \in \mathcal{I}. \end{aligned} \tag{16}$$

This is a standard quadratic programming problem, and we solve it using convex optimization tools, e.g., CVX [3]. After solving the problem, the controller updates the dual variables $u_{i,j,k}^{(n+1)}$ according to Eq. 6. The pseudo code of the controller side algorithm is shown in Alg. 2.

3.4 Algorithm Analysis

Fig. 2 illustrates the distributed resource orchestration algorithm. At the beginning, each edge node derives an initial resource allocation to the network slices and sends the resource allocation and the dual variables to the controller.

Algorithm 1: Resource Orchestration: Edge Node

```
Input: $R^{tot}_{j,k}$, $u_{i,j,k}$ and $z_{i,j,k}$, $\forall i \in \mathcal{I}, k \in \mathcal{K}$.
Output: $\mathbf{x}_i, \forall i \in \mathcal{I}$, $u_{i,j,k}, \forall i \in \mathcal{I}, k \in \mathcal{K}$.
1  Initialize $\mathbf{x}_i, \mathcal{D}_i, \forall i \in \mathcal{I}$, and set $t \leftarrow 1$;
2  while True do
3      /** query the utility of slices **/;
4      Given $\mathbf{x}_i$, query to observe $f_i(\mathbf{x}_i), \forall i \in \mathcal{I}$ ;
5      /** update the observation set **/;
6      $\mathcal{D}_i^{1:t} = \{\mathbf{x}_i^{1:t}, y_i^{1:t}\}, \forall i \in \mathcal{I}$;
7      /** update prediction for slices **/;
8      $\mu(\mathbf{x}_i^t) \leftarrow Eq.\ 10, \sigma_i^2(\mathbf{x}_i^t) \leftarrow Eq.\ 11, \forall i \in \mathcal{I}$;
9      /** compute predictive gradients **/;
10     $\nabla \bar{f}_i(\mathbf{x}_i^t) \leftarrow Eq.\ 13, \forall i \in \mathcal{I}$;
11     /** update allocation based on gradients **/;
12     $\mathbf{x}_i^{t+1} \leftarrow Eq.\ 14, \forall i \in \mathcal{I}$;
13     /** project allocation with constraints **/;
14     $\mathbf{x}_k^{t+1} = P_{\Omega_k}\left(\mathbf{x}_k^{t+1}\right), \forall k \in \mathcal{K}$;
15     if $||\mathbf{x}_i^{t+1} - \mathbf{x}_i^t|| \leq \eta, \forall i \in \mathcal{I}$ then
16         break;
17     $t \leftarrow t + 1$;
18 return $\mathbf{x}_i, \forall i \in \mathcal{I}$, $u_{i,j,k}, \forall i \in \mathcal{I}, k \in \mathcal{K}$.
```

After receiving the initial resource allocations from all edge nodes, the controller coordinates the resource orchestration by updating the auxiliary variables, $z_{i,j,k}$, and the dual variables. The updated variables are fed back to edge nodes for the next round of resource allocation. The orchestration process stops when the resource allocations converge. We prove the convergence in Corollary 1.

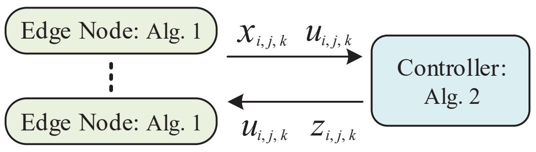

Figure 2: The distributed resource orchestration.

Proposition 1. *$\nabla \bar{f}_i(\mathbf{x}_i), \forall i \in \mathcal{I}$ are the controllably accurate gradient approximations of $\nabla f_i(\mathbf{x}_i), \forall i \in \mathcal{I}$ if $f_i(\mathbf{x}_i), \forall i \in \mathcal{I}$ are Lipschitz continuous.*

Proof. Denote $f_i(\mathbf{x}_i)$ and $\bar{f}_i(\mathbf{x}_i)$ as the utility and predicted utility of the ith slice at $\mathbf{x}_i$, respectively. The predictive gradients are defined as $\nabla \bar{f}_i(\mathbf{x}_i) = (\bar{f}_i(\mathbf{x}_i + \tau) - f_i(\mathbf{x}_i))/\tau = (\mu(\mathbf{x}_i + \tau) - f_i(\mathbf{x}_i))/\tau, \forall i \in \mathcal{I}$ (Eq. 13). Then,

$$|\nabla \bar{f}_i(\mathbf{x}_i) - \nabla f_i(\mathbf{x}_i)| = \frac{1}{\tau}|(\mu(\mathbf{x}_i + \tau) - f_i(\mathbf{x}_i + \tau))| \tag{17}$$
$$= \frac{1}{\tau}(|\mathbf{c}^T[\mathrm{C} + \delta^2 I]^{-1} y_i^{1:t} - f_i(\mathbf{x}_i + \tau)|),$$

where $y_i = f_i(\mathbf{x}_i) + \epsilon$. Since the utility functions $f_i(\mathbf{x}_i), \forall i \in \mathcal{I}$ are Lipschitz continuous [3], there exists a positive real constant κ such that, for all real $\mathbf{x}_i^1$ and $\mathbf{x}_i^2$,

$$|f_i(\mathbf{x}_i^1) - f_i(\mathbf{x}_i^2)| \leq \kappa|\mathbf{x}_i^1 - \mathbf{x}_i^2|, \forall i \in \mathcal{I}. \tag{18}$$

Hence, we obtain

$$|\nabla \bar{f}_i(\mathbf{x}_i) - \nabla f_i(\mathbf{x}_i)| \leq \Delta, \tag{19}$$

where Δ is a constant. According to Definition 10.1 in [2], the $\nabla \bar{f}_i(\mathbf{x}_i), \forall i \in \mathcal{I}$ are the controllably accurate gradient approximations of $\nabla f_i(\mathbf{x}_i), \forall i \in \mathcal{I}$. □

Algorithm 2: Resource Orchestration: Controller

```
Input: $Q_i^{tot}$, $x_{i,j,k}$ and $u_{i,j,k}$, $\forall i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{K}$.
Output: $x_{i,j,k}$, $u_{i,j,k}$, $z_{i,j,k}$, $\forall i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{K}$.
1  /** determine algorithm convergence **/;
2  if $|| \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} x_{i,j,k} - Q_i^{tot} || \leq \eta, \forall i \in \mathcal{I}$ then
3      return $x_{i,j,k}, \forall i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{K}$;
4  else
5      /** update z in the controller **/;
6      $z_{i,j,k} \leftarrow \arg\max_{z_{i,j,k}\in C_1} \mathcal{L}_u(x_{i,j,k}, z_{i,j,k}, u_{i,j,k})$;
7      /** update u in the controller **/;
8      $u_{i,j,k} \leftarrow u_{i,j,k} + (x_{i,j,k} - z_{i,j,k})$;
9      return $z_{i,j,k}$ and $u_{i,j,k}, \forall i \in \mathcal{I}, j \in \mathcal{J}, k \in \mathcal{K}$.
```

Corollary 1. *The distributed resource orchestration algorithm converges to a local optimum if $f_i(\mathbf{x}_i), \forall i \in \mathcal{I}$ are non-decreasing and Lipschitz continuous.*

Proof. The distributed resource orchestration algorithm consists of the controller-side and edge-node-side algorithms which iteratively exchanges information. The controller-side algorithm, i.e., Alg. 2, simply updates the auxiliary and dual variables, and there is no need to prove its convergence. Hence, to show the convergence of the distributed resource orchestration algorithm, we prove that the edge-node-side algorithm, i.e., Alg. 1, converges, and the iterative information exchanges lead to a converged resource allocation.

Convergence of Alg. 1: This algorithm is designed based on gradient-based optimization. According to Theorem 10.6 in [2], the convergence of the algorithm can be proved by showing 1) the predictive gradients $\nabla \bar{f}_i(\mathbf{x}_i), \forall i \in \mathcal{I}$ are the controllably accurate gradient approximations of $\nabla f_i(\mathbf{x}_i), \forall i \in \mathcal{I}$; and 2) the step size is positive in the gradient-based descent method. The first condition has been proved in *Proposition* 1. Since the step size in the algorithm is positive, the algorithm satisfies both conditions required for the convergence. Therefore, the convergence of Alg. 1 is proved.

Convergence of iterative information exchanges: The iterative information exchanges are designed based on the ADMM method. Therefore, we prove the convergence of the iterative information exchanges based on the convergence proofs of the ADMM method [1, 26]. First, we prove the intermediate variables, $\mathcal{X}$, $\mathcal{Z}$ and $\mathcal{U}$, and augmented Lagrangian $\mathcal{L}_u$ are bounded. Second, we prove that there

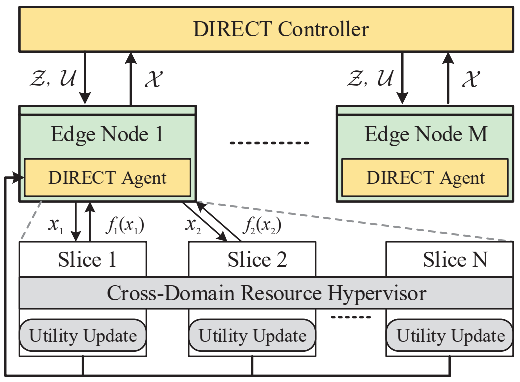

Figure 3: The design of DIRECT protocol.

exists positive constants α_x and α_z such that $\mathcal{L}_u^{t+1} - \mathcal{L}_u^t \leq \alpha_x|\mathbf{x}^{t+1} - \mathbf{x}^t| + \alpha_z|\mathbf{z}^{t+1} - \mathbf{z}^t|$; Thus, $\mathcal{L}_u$ is monotonically non-decreasing and lower bounded. Third, we prove that there exists positive constants α_x, α_z and $d^* \in \partial\mathcal{L}$ such that $|d^*| \leq \alpha_x|\mathbf{x}^{t+1} - \mathbf{x}^t| + \alpha_z|\mathbf{z}^{t+1} - \mathbf{z}^t|$. Therefore, when $t \to \infty$, $|d^*| \to 0$. Fourth, we prove that if $(\mathcal{X}^*, \mathcal{Z}^*, \mathcal{U}^*)$ is the limit point of generated sequence $(\mathcal{X}^{1:t}, \mathcal{Z}^{1:t}, \mathcal{U}^{1:t})$, we obtain $\mathcal{L}_u(\mathcal{X}^*, \mathcal{Z}^*, \mathcal{U}^*) = \lim_{t\to\infty} \mathcal{L}_u(\mathcal{X}^t, \mathcal{Z}^t, \mathcal{U}^t)$. Therefore, the iterative information exchanges between edge nodes and the controller converge. Due to the space limitation, we omit the detail convergence proof.

Since both Alg. 1 and the iterative information exchanges are convergent, the convergence of the distributed resource orchestration algorithm is proved. □

4 PROTOCOL DESIGN AND IMPLEMENTATION

In this section, we design the DIRECT protocol according to the distributed resource orchestration algorithm and implement the protocol in a small-scale testbed developed based on the LTE and GPU computing platforms [10, 20].

4.1 Protocol Design

Fig. 3 illustrates the design of the DIRECT protocol which mainly consists of a DIRECT controller and multiple DIRECT agents on edge nodes. To realize the protocol, we also develop a cross-domain resource hypervisor to manage the resource virtualization. The DIRECT controller runs the controller-side algorithm, i.e., Alg. 2, to coordinate the cross-domain resource orchestration among network slices and ensure that each network slice is properly served according to its service level agreement. It coordinates the resource orchestration by controlling the auxiliary variables, $\mathcal{Z}$, and the dual variables, $\mathcal{U}$, which are fed back to the DIRECT agent. On the edge node, the DIRECT agent runs the node-side algorithm, i.e., Alg. 1, to allocate the multiple resources to network slices for maximizing the sum utility under resource constraints of the edge node. We develop a cross-domain resource hypervisor on each edge node to enable the coexistence of multiple network slices on the same physical infrastructure, e.g., base station and edge server. Hence, the agent can maintain network slices operations, such as creation, suspension and deletion. The DIRECT agent manages the hypervisor for the resource virtualization as well as the slice creation, suspension and deletion. On the edge node, the resources allocated to network slices are mapped to physical infrastructure at runtime, and the utility of network slices are reported to the DIRECT agent.

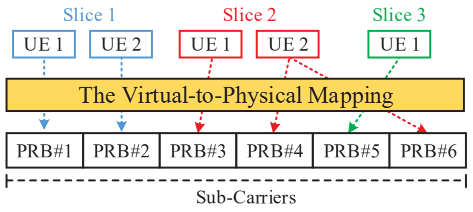

Figure 4: The radio resource virtualization.

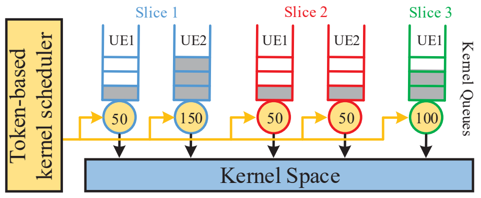

Figure 5: The computing resource virtualization.

4.2 Protocol Implementation

To realize the DIRECT protocol, we develop a cross-domain resource hypervisor to dynamically manage the physical resource according to the resource allocation. Meanwhile, the hypervisor ensures the performance and functional isolations among network slices during the resource orchestration. In the protocol implementation, we consider the radio and computing resources in the context of LTE and CUDA GPU programming, respectively.

4.2.1 Radio Resource Hypervisor. In the implementation, the network slices on an edge node share the same control plane operations following standard LTE protocols. To differentiate users among network slices, we implement a user-association function in the control plane to associate users to their corresponding network slices. The radio resource hypervisor focuses on managing the uplink/downlink resources (URs/DRs), i.e., physical resource blocks (PRBs) of PUSCH/PDSCH, in the LTE user plane. We define the resource allocated to users by network slices as the virtual resource. As shown in Fig. 4, the radio resource hypervisor maps the virtual radio resources that allocated to users by network slices to PRBs. During the resource mapping, we maximize the network throughput by selecting the user with the best channel condition for each PRB. After all virtual resources are mapped, the surplus PRBs are allocated to the users who have the best channel condition. Since the single carrier-FDMA (SC-FDMA) is the access method in LTE uplink, the PRBs allocated to a single user must be contiguous in frequency domain [17]. Hence, the hypervisor allocates the surplus PRBs to the users whose current PRBs and the surplus PRBs are contiguous.

4.2.2 Computing Resource Hypervisor. In the CUDA programming model, an application can launch multiple *kernels* that can be concurrently executed by massive CUDA *threads*. To realize dynamic computing resource management, we develop a token-based kernel scheduler to control the execution of *kernels*. As illustrated in Fig. 5, the kernel scheduler dispatches the *kernels* commands according to the computing tokens of network slices. In the user space, we add a *KernelSpawn* function to push the *kernel* commands into a FIFO queue. A user's *kernel* commands is pulled out of the queue and enter the kernel space if the user has sufficient tokens. Here, the DIRECT agent allocates virtual resources to a network slice, and the network slice distributes the resources to its users. The computing resource hypervisor covers a user's virtual computing resources into tokens. The *KernelSpawn* function is running in a system thread with non-blocking property to ensure the asynchronous executions of *kernels*.

5 PERFORMANCE EVALUATION

In this section, we validate the performance of the DIRECT protocol through experiments in a small-scale system prototype and network simulations.

5.1 System Prototype

We develop a small-scale prototype as shown in Fig. 6 to evaluate the performance of the DIRECT protocol. In the prototype, we consider three resources from two technical domains: for the radio access network, we consider uplink and downlink radio resources; for edge computation, we consider the GPU resources for hardware-accelerated computing. The radio and computing resources are essential for killer applications in 5G such as mobile cross reality and autonomous driving [9]. The prototype consists of two edge nodes, and each edge node is composed of an eNodeB and a GPU. We place two eNodeBs in different room to emulate a cellular network with the limited co-channel interference. The radio access network is implemented based on the OpenAirInterface (OAI) LTE platform [20], and the core network is built with openair-cn [19]. The computing platform is NVIDIA CUDA-enable GPU [10]. We use a dell alienware desktop (Intel i7 8700 @3.2GHz, 64GB RAM) with two NVIDIA GEFORCE GTX 1080Ti (3584 CUDA cores, 11G RAM) for deploying the DIRECT controller, DIRECT agents and mobile core network. We use two desktop computers with low-latency kernel (Intel i5 4590@3.3GHz, 16 RAM) to deploy the eNodeBs and emulate mobile users with Huawei E3372h LTE dongles. Two Ettus USRP B210 SDR boards are used as the RF front-end of

Table 1: User Association

	Slice 1	Slice 2	Slice 3
Edge Node 1	1	0	1
Edge Node 2	0	1	1

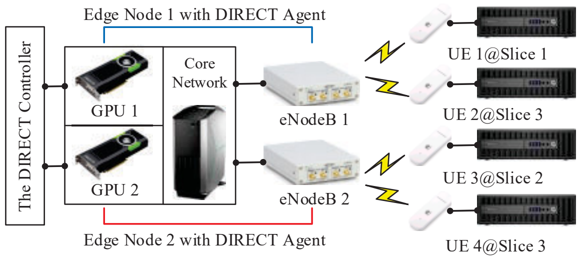

Figure 6: The testbed implementation of DIRECT protocol.

eNodeBs. The uplink and downlink carrier frequencies are 2.655GHz and 2.535GHz, respectively (LTE Band 7). Both the uplink and downlink bandwidths are 5MHz (25 PRBs).

In the experiment, we create three network slices to serve four mobile users on the edge nodes[1]. The user-slice-edge node association is listed in Table 1.

We adopt the negative slice-latency as the utility of a slice, i.e., $f_{i,j}(\mathbf{x}_{i,j}), \forall i \in \mathcal{I}, j \in \mathcal{J}$. Here, the slice-latency is defined as the sum latency of all users in a slice across all edge nodes. The edge-latency is defined as the summation of the weighted slice-latency of all network slices in an edge node. The system-latency is the summation of the slice-latency of all slices. Hence, maximizing the utility of slices is equivalent to minimizing the slice-latency of slices. In this paper, we do not study the user scheduling algorithm within a network slice and thus assume that a network slice evenly allocates resources to its users.

5.2 Compute-intensive Applications

In the experiments, we implement two compute-intensive mobile applications based on the YOLO object detection algorithm to evaluate the performance of the DIRECT protocol [22]. These applications are mobile augmented reality (MAR) and video analytics and streaming (VAS). The first and second network slices support the MAR application, and the third slice supports the VAS application.

Mobile Augmented Reality (MAR): A client continuously sends video frames with the resolution of 1280x720 to server and receives the detection results. The server receives the frames, executes the YOLO 608x608 algorithm, and sends the detection results back to client. MAR represents the type of applications that have heavy uplink traffic loads and intensive computing workloads. Here, YOLO 608x608 means the YOLO algorithm tuned at the image resolution of 608x608.

Video Analytics and Streaming (VAS): A client sends a streaming request to the server. The server retrieves the

[1]Since the numbers of network slices and users are small in the system prototype, we evaluate the scalability of the DIRECT protocol in simulations.

real-time camera frames with the resolution of 1280x720, processes it with the YOLO 416x416 algorithm, and sends the frames with detection results back to client. VAS represents the type of applications that have heavy downlink traffic loads and moderate computing workloads. Here, YOLO 416x416 is less compute-intensive than YOLO 608x608.

5.3 Comparison Protocols

We compare DIRECT with following protocols:

- **Static:** With Static, a network slice evenly distributes its payment, Q_i^{tot}, to acquire resources from all technical domains on all edge nodes.
- **PSwarm:** PSwarm is a global optimization solver [25] for bounded and linear constrained derivative-free problems. In this protocol, PSwarm replaces Alg. 1 in the distributed resource orchestration.
- **TOMLAB:** The TOMLAB with *glcSolve* is a optimization solver [11] that handles the global mixed-integer nonlinear programming problems. In this protocol, TOMLAB replaces Alg. 1 in the distributed resource orchestration.

PSwarm and TOMLAB have very high computation complexity and communication overhead and are impractical in a real system. Therefore, we compare the performance of DIRECT with those protocols in network simulations.

5.4 Experimental Evaluation

Convergence: Figs. 7 (a) and (b) show the system-latency and resource allocation gap versus the number of iterations. In the experiment, the resource allocation gap is defined as $\sum_{i\in\mathcal{I}} |\sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} x_{i,j,k} - Q_i^{tot}|$. When the resource allocation gap reaches zero, the resource orchestration algorithm converges. The experiment result shows that DIRECT converges after 5 iterations. DIRECT reduces about 21% system-latency as compared to the Static protocol which has an almost constant system-latency because of its static resource allocation. The DIRECT protocol has a small fluctuation after the convergence because of the dynamic wireless channel conditions. Fig. 7 (c) shows the edge-latency of edge nodes versus the number of iterations and reflects the convergence of Alg. 1. It shows that Alg. 1 converges after 15 iterations and significantly reduces the edge-latency.

Resource allocation: Fig. 8 (a) shows resource allocations of network slices on different edge nodes. We can observe that all resources of the first and second slices are allocated to the first and second edge nodes, respectively. This resource allocation matches the traffic workload in the experiment where the first and second slices do not have users in the second and first edge node, respectively. Fig. 8 (b) shows the resource utilization in different technical domains of network slices. It can be seen that the third slice utilizes more downlink radio resources than other types of resources because the slice supports VAS applications which

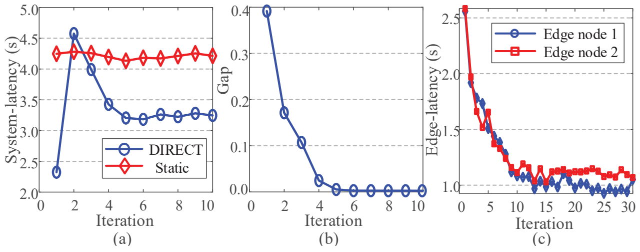

Figure 7: The convergence of the DIRECT protocol.

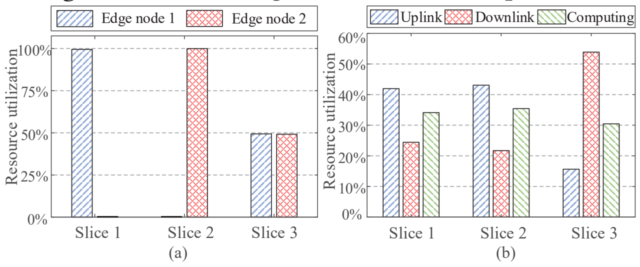

Figure 8: The cross-domain resource allocation.

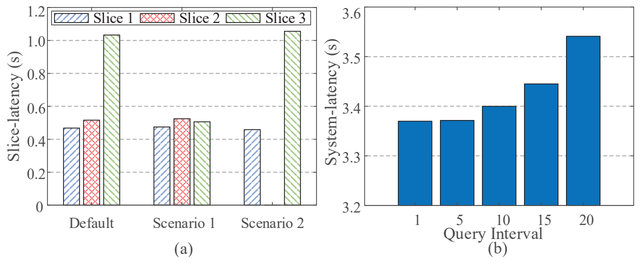

Figure 9: The query interval and isolation performance.

have heavier downlink traffic loads These results indicate that DIRECT is able to allocate multi-domain resources according to the traffic workload among edge nodes and the characteristics of applications served by the slices.

Performance Isolation: Fig. 9 (a) shows the slice-latency under different scenarios with the DIRECT protocol. In the first scenario, we remove a user from slice 3. The slice-latency of slice 3 is reduced since the DIRECT protocol adapts to the traffic workload of slice among edge nodes and allocates all its resources to the single user. In the second scenario, we remove a user from slice 2. As a result, the utility of slice 2 is zero since there are no associated users. From the experiment, it can be observed that the slice-latency of a network slice will not affect or be affected by the traffic variations of other slices, which means that DIRECT ensures the performance isolation among network slices.

Query Interval: In the DIRECT protocol, the resource orchestration algorithm on edge nodes, i.e., Alg. 1, queries the utility of network slices, i.e., $f_i(\mathbf{x}_i), \forall i \in \mathcal{I}$, for the resource allocation. The frequency of the utility queries impacts the system latency and the convergence speed of the algorithm. More utility queries lead to a lower system latency but a slower convergence speed. Fig. 9 (b) shows the system-latency versus different query intervals. A large query interval means a lower query frequency. It can be seen from the figure that a larger query interval leads to a longer system-latency. This because a larger query interval may incur inaccurate predictions of the predictive gradients.

5.5 Simulation Evaluation

Here, we aim to evaluate the scalability of the DIRECT protocol through network simulations. In the simulation, there are 4 edge nodes and 6 network slices with 3 types of resources. The number of users in each network slice and edge node is random with a range of one to five. The utility function of the ith slice in jth edge node is defined as

$$f_{i,j}(\mathbf{x}_{i,j}) = \sum\nolimits_{k \in \mathcal{K}} A_k \cdot (x_{i,j,k})^{\beta}, \tag{20}$$

where $x_{i,j,k}$ is the kth resource of the ith slice in the jth edge node, A_k is the weight for the kth resource, and β is a parameter controls the property of the utility function. The utility functions are used by individual slices to calculate their utilities given resource allocations in the simulation. The resource orchestrator does not know the utility functions. When β is positive, the utility of slices $f_{i,j}$ is positively correlated to the allocated resources $x_{i,j,k}$, e.g., more resources lead to a larger utility. For example, the network throughput can be such a utility. The network aims to maximize the throughput. When β is negative, the utility of slices $f_{i,j}$ is negatively correlated to the allocated resources $x_{i,j,k}$, e.g., more resources result in a smaller utility. For example, the network latency can be such a utility. The network aims to minimize the latency. In the simulations, the default value of β is -1. The total amount of the kth resource on the jth edge nodes is constrained by $R^{tot}_{j,k} = 100, \forall j \in \mathcal{J}, k \in \mathcal{K}$. The weights, $A_k, \forall k \in \mathcal{K}$, are generated according to a uniform distribution between one and ten.

Convergence: Fig. 10 validates the convergence of DIRECT under different number of slices and edge nodes. In the simulation, the system latency under a simulation setting is normalized with respect to the optimal system latency derived by DIRECT under the same simulation setting. The DIRECT protocol converges in about ten iterations with nearly zero gap, for all simulation settings. Meanwhile, the convergence of the Alg. 1 and service latency in edge nodes are also shown in Fig. 10 (c). The simulation results show that the properties of black-box function can be learned in several iterations, and the proposed algorithm gradually converges and significantly reduces the edge-latency of edge nodes.

Scalability: Fig. 11 evaluates the performance of the DIRECT protocol under different number of slices and edge nodes. The performance gap between the DIRECT and Static protocol enlarges with the increment of number of slices and edge nodes. This is because the Static protocol cannot adapt to the resource requirements of slice service that results in a high system-latency. The simulation results also show that the DIRECT protocol outperforms both the PSwarm and TOMLAB. Although the performance of the DIRECT is only slightly better than that of the PSwarm, both PSwarm and TOMLAB have a very high communication overhead and complexity and cannot be implemented in a practical system.

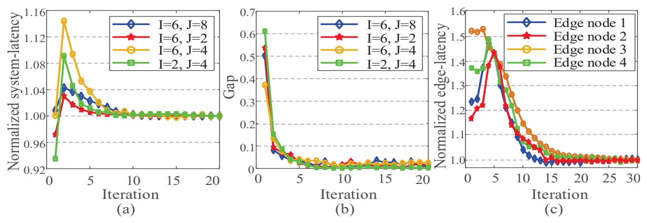

Figure 10: The convergence of the DIRECT protocol.

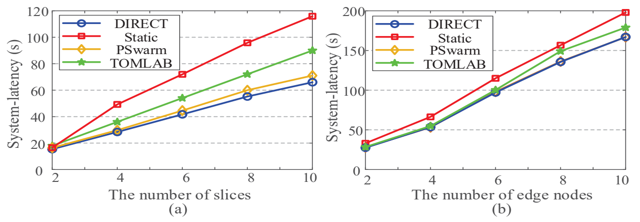

Figure 11: The scalability of the DIRECT protocol.

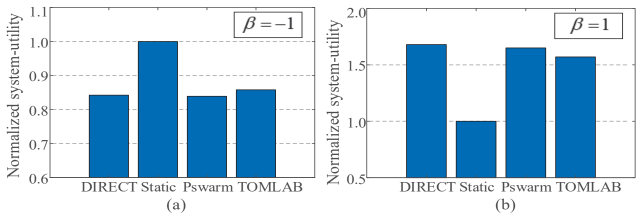

Figure 12: The impact of utility function.

Utility functions: Fig. 12 shows the performance of the DIRECT protocol with different utility functions. When $\beta = -1$, i.e., lower utility is preferred, the DIRECT protocol reduces the utility of system by 17% as compared to the Static protocol. When $\beta = 1$, i.e., higher utility is preferred, the DIRECT protocol improves 70% utility of system as compared to the Static protocol.

6 RELATED WORK

This work is related to the radio access network slicing and multiple resource management.

Radio Access Network Slicing: Network slicing has attracted extensive research attentions. Kokku *et. al.* [15] proposed a network virtualization substrate (NVS) which introduces a slice scheduler to control the frame scheduling in the MAC layer in WiMAX networks. Foukas *et. al.* designed FlexRAN [7] to decouple the control and user plane of LTE. FlexRAN provides a customized API and programmable control plane for the radio access network management. They also developed the Orion system [6] that is able to perform network slicing at runtime and provide the functional and performance isolation among network slices. However, these works do not investigate the jointly orchestration of radio and computing resources. Moreover, none of them discuss the radio access network slicing over multiple base stations and edge servers.

Multi-Domain Resource Management: Multi-domain resource management has been well investigated in cloud computing where heterogeneous applications require diverse computing, memory and storage resources. Ghodsi *et. al.* [8] developed a dominant resource fairness (DRF) algorithm to maintain the max-min fairness among tenants. Chowdhury *et. al.* [5] proposed a high utilization with guarantees (HUG) algorithm that is an extension of DRF for handling elastic resource demands. Liu *et. al.* [18] designed a multi-domain resource allocation algorithm which maximizes the utility of system in mobile cloud computing using a Markov decision process. However, almost all existing works need the system performance model for the resource management. In the context of resource orchestration in cellular edge computing, the performance models of network slices are unknown.

7 CONCLUSION

We have presented the DIRECT protocol that realizes the cross-domain resource orchestration for cellular edge computing. As a key component of the protocol, a new distributed resource orchestration algorithm is developed by integrating the ADMM method and LAO. The proposed algorithm addresses the challenge of unknown performance model of network slices and efficiently orchestrates multi-domain resources among network slices across edge nodes. We implemented and validated the DIRECT protocol in a small-scale system prototype based on the OpenAirInterface LTE and CUDA GPU computing platforms. We also evaluated the DIRECT protocol in network simulations.

ACKNOWLEDGMENTS

We appreciate insightful comments and suggestions from the shepherd, Dr. Stratis Ioannidis, and the anonymous reviewers. This work is partially supported by the US National Science Foundation under Grant No. 1731675 and No. 1810174.

REFERENCES

[1] Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter. 2013. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods. *Mathematical Programming* 137, 1-2 (2013), 91–129.

[2] Charles Audet and Warren Hare. 2017. *Derivative-free and blackbox optimization.* Springer.

[3] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization.* Cambridge university press.

[4] P. Caballero, A. Banchs, G. de Veciana, and X. Costa-PÃĩrez. 2017. Network slicing games: Enabling customization in multi-tenant networks. In *IEEE INFOCOM.* Atlanta, GA, 1–9.

[5] Mosharaf Chowdhury, Zhenhua Liu, Ali Ghodsi, and Ion Stoica. 2016. HUG: Multi-Resource Fairness for Correlated and Elastic Demands. In *NSDI.* 407–424.

[6] Xenofon Foukas, Mahesh K Marina, and Kimon Kontovasilis. 2017. Orion: RAN Slicing for a Flexible and Cost-Effective Multi-Service Mobile Network Architecture. In *ACM MobiCom.* 127–140.

[7] Xenofon Foukas, Navid Nikaein, Mohamed M Kassem, Mahesh K Marina, and Kimon Kontovasilis. 2016. FlexRAN: A flexible and programmable platform for software-defined radio access networks. In *ACM CoNEXT.* 427–441.

[8] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. 2011. Dominant Resource Fairness: Fair Allocation of Multiple Resource Types. In *NSDI*, Vol. 11. 24–24.

[9] Global Mobile Suppliers Association. 2017. 5G Network Slicing for Vertical Industries. http://www.huawei.com/minisite/5g/img/5g-network-slicing-for-vertical-industries-en.pdf.

[10] Mark Harris. 2017. An Even Easier Introduction to CUDA.

[11] Kenneth Holmström. 1999. The TOMLAB optimization environment in Matlab. (1999).

[12] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. 2015. Mobile Edge Computing: A key technology towards 5G. *ETSI White Paper* 11 (September 2015).

[13] K. Katsalis, N. Nikaein, E. Schiller, A. Ksentini, and T. Braun. 2017. Network Slices toward 5G Communications: Slicing the LTE Network. *IEEE Communications Magazine* 55, 8 (2017), 146–154.

[14] Ravi Kokku, Rajesh Mahindra, Honghai Zhang, and Sampath Rangarajan. [n. d.]. CellSlice: Cellular wireless resource slicing for active RAN sharing. In *IEEE COMSNETS 2013.* 1–10.

[15] Ravi Kokku, Rajesh Mahindra, Honghai Zhang, and Sampath Rangarajan. 2012. NVS: A substrate for virtualizing wireless resources in cellular networks. *IEEE/ACM Transactions on Networking (TON)* 20, 5 (2012), 1333–1346.

[16] Adlen Ksentini and Navid Nikaein. 2017. Toward Enforcing Network Slicing on RAN: Flexibility and Resources Abstraction. *IEEE Communications Magazine* 55, 6 (2017), 102–108.

[17] S-B Lee, Ioannis Pefkianakis, Adam Meyerson, Shugong Xu, and Songwu Lu. 2009. Proportional fair frequency-domain packet scheduling for 3GPP LTE uplink. In *IEEE INFOCOM.* 2611–2615.

[18] Yanchen Liu, Myung J Lee, and Yanyan Zheng. 2016. Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system. *IEEE Transactions on Mobile Computing* 15, 10 (2016), 2398–2410.

[19] OpenAirInterface Software Alliance. Openair-cn repository. https:gitlab.eurecom.fr/oai/openair-cn. 2017.

[20] OpenAirInterface Software Alliance. OpenAirInterface repository. https:gitlab.eurecom.fr/oai/openairinterface5g. 2017.

[21] Carl Edward Rasmussen. 2004. Gaussian processes in machine learning. In *Advanced lectures on machine learning.* Springer, 63–71.

[22] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of IEEE conference on computer vision and pattern recognition.* 779–788.

[23] Peter Rost, Christian Mannweiler, Diomidis S Michalopoulos, Cinzia Sartori, Vincenzo Sciancalepore, Nishanth Sastry, Oliver Holland, Shreya Tayade, Bin Han, Dario Bega, et al. 2017. Network slicing to enable scalability and flexibility in 5G mobile networks. *IEEE Communications magazine* 55, 5 (2017), 72–79.

[24] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems.* 2951–2959.

[25] A Ismael F Vaz and Luís N Vicente. 2009. PSwarm: a hybrid solver for linearly constrained global derivative-free optimization. *Optimization Methods & Software* 24, 4-5 (2009), 669–685.

[26] Yu Wang, Wotao Yin, and Jinshan Zeng. 2015. Global convergence of ADMM in nonconvex nonsmooth optimization. *Journal of Scientific Computing* (2015), 1–35.

[27] Haijun Zhang, Na Liu, Xiaoli Chu, Keping Long, Abdol-Hamid Aghvami, and Victor CM Leung. 2017. Network slicing based 5G and future mobile networks: mobility, resource management, and challenges. *IEEE Communications Magazine* 55, 8 (2017), 138–145.