Edge Federation: Towards an Integrated Service Provisioning Model

Xiaofeng Cao, Guoming Tang, *Member, IEEE,* Deke Guo, *Senior Member, IEEE,* Yan Li, and Weiming Zhang

***Abstract*—Edge computing is a promising computing paradigm for pushing the cloud service to the network edge. To this end, edge infrastructure providers (EIPs) need to bring computation and storage resources to the network edge, and allow edge service providers (ESPs) to provision latency-critical services to users. Currently, EIPs prefer to establish a series of private edge-computing environments to serve specific requirements of users. This kind of resource provisioning mechanism seriously limit the development and spread of edge computing for serving diverse user requirements. To this end, we propose an integrated resource provisioning model, named *edge federation*, to seamlessly realize the resource cooperation and service provisioning across standalone edge computing providers and clouds. To efficiently schedule and utilize the resources across multiple EIPs, we systematically characterize the provisioning process as a large-scale linear programming (LP) problem and transform it into an easily solved form. Accordingly, we design a dynamic algorithm to varying service demands from users. We conduct extensive experiments over the base station networks in Toronto city. The evaluation results demonstrate that our edge federation model proposed can reduce the overall cost of EIPs by 30.5% to 32.4%, compared with existing independent service provisioning model.**

***Index Terms*—Edge Federation, Resource integration, Optimal service provisioning solution.**

I. Introduction

The emergence of edge computing offers a new paradigm to deliver computation-intensive and latency-critical services to traditional cloud users [1]. The basic idea is to push the cloud service to the network edge (e.g., access point or base stations), which is closer to users than cloud. In this way, users can still exploit the power of cloud computing, while no longer suffer network congestion and long latency [2]. The prosperous development of edge computing offers an excellent opportunity for mobile service providers to rent resources from edge infrastructure providers (EIPs) for hosting their services. An EIP usually has to construct and maintain a set of distributed edge nodes at the network edge, where an edge node may consist of multiple edge servers and has specific computation and storage resources. EIPs are responsible for service provisioning and resource management.

However, compared with existing mobile cloud computing solution, edge computing is still constrained in the resource capacity and suffers the high cost due to maintaining the edge infrastructure widely [2], [3]. The main reason is that EIPs prefer to establish a series of private edge-computing environments to serve specific requirements of users from the aspect of their own viewpoint [4]. That is, each EIP only manages and uses its resources; hence, a standalone edge-computing environment is usually resource-constrained, especially in the scenario of serving the increasing amount of users. When a large number of services need to be deployed in broad geographic areas, the involved EIPs need to construct and maintain more edge nodes to cover them further, which leads to a huge cost. On the other hand, however, different EIPs may build edge nodes in the same place while without any cooperation, causing severe under-utilization of resources. To make matters worse, since the individual EIP has limited information about the whole edge-computing environment. It is tough to make a global optimization for efficiently delivering various services to different users. This dilemma may cause a low quality of service (QoS) for the edge service providers (ESPs) as well as the low quality of experience (QoE) for end users. In general, the existing resource-provisioning model falls in a triple-lose situation where the development and spread of the edge computing are severely restricted.

With the above challenges in mind, this paper presents *edge federation*, an integrated service provisioning model for the edge-computing paradigm. It aims at creating a cost-efficiency platform for EIPs and offering end users and ESPs a transparent resource management infrastructure by seamlessly integrating individual edge computing providers as well as clouds from two-dimensions, the horizontal dimension, and the vertical dimension.

Private or Public: In the horizontal dimension, EIPs independently construct and maintain their private resource infrastructures, which restricts the quick development and spread of edge computing. In the existing model, an EIP usually has a limited amount of edge servers to deploy services, and hence cannot cover broad areas and may cause long service delivery latency to those users outside the covered areas. This dilemma would severely limit the market size of each EIP. A straightforward method is to make each EIP build edge servers in more locations. This method, however, would cause a large amount of duplicated edge nodes across EIPs in many sites, leading to the enormous capital and operational expenditure. The similar problem has arisen in the construction of cellular towers by different mobile network operators. Therefore, it is urgent to enable the interoperability and realize resource sharing across EIPs.

Edge or Cloud: In the vertical dimension, cloud computing and edge computing both have their own advantages, but

X. Cao, G. Tang, D. Guo, Y. Li, W. Zhang are with the Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, Hunan, 410073, P. R. China (e-mail: {caoxiaofeng10, gmtang, dekeguo, liyan10, wmzhang}@nudt.edu.cn). D. Guo is also with the College of Intelligence and Computing, Tianjin University, Tianjin, 300350, P. R. China.

neither of them can meet the high latency requirement (a.k.a., low time delay) of services and low-cost resource provision simultaneously. Although edge computing can achieve much lower latency in service delivery than that in cloud computing, it also incurs high deployment cost of new computation and storage infrastructures [5], [6]. On the contrary, the low cost and sufficient resources are precisely the advantages of cloud computing. Moreover, as each edge node has a limited range of serving area, the cloud could be an essential complement for support end users in the areas not served by edge nodes [7]. In summary, edge computing and cloud computing can reasonably complement each other, but they need an effective mechanism to cooperate.

The edge federation proposed in this paper brings a new service-provisioning model for driving the next generation edge-computing network. To realize the designed vision of edge federation, we systematically tackle several essential challenges. First, an edge-computing network is very complex, which consists of a series of EIPs, diverse services and heterogeneous end devices. The edge federation designed in this paper should realize the targets of scalability, efficiency, and low-latency. Second, the edge federation should effectively model the joint service provisioning process across heterogeneous edge nodes, and across edge and cloud. Third, the service provisioning problem under the edge federation involves a large number of optimization variables and exhibits very high computation complexity. The efficient solutions with affordable complexity is needed to deal with the large-scale optimization problem.

We address all the above challenges in this work and make the following major contributions:

- We design the edge federation, an integrated edge-computing model, to realize the transparent service provisioning across independent EIPs and the cloud. The edge federation model can significantly improve the QoS to end users and save the cost of EIPs.
- We characterize the service provisioning process under our edge federation as a linear programming (LP) optimization model and propose a dimension-shrinking method to reformulate it into an easily solved model. Accordingly, we develop a service provisioning algorithm *SEE*, which dynamically enable an effective and efficient service deployment in an edge federation environment.
- We evaluate the proposed solution for edge federation under the base stations network of Toronto city with the real-world trace. The results indicate that the edge federation can help EIPs save the overall cost by 30.5% to 32.4%, compared with existing independent service provisioning model. We find that our solution can significantly help the EIP with latency-critical services save more cost, under the intensive service demands.

The rest of the paper is organized as follow. We introduce the background and the related challenges of the edge federation in Sec. II. Then, the detailed architecture is illustrated, and the contributions are also highlighted in Sec. III. Sec. IV formulates the cost minimization problem for EIPs with hard latency constraints. In Sec. V, the problem is transformed with the dimension-shrinking method and the dynamic provisioning algorithm is developed. We evaluate the performance of our solution via real-trace network service data and validates the effectiveness of the edge federation in Sec. VI. Sec. VII gives the discussion and future works. Sec. VIII and Sec. IX close the paper with related works and conclusions.

II. Edge Federation vs. Cloud Federation

The edge federation is the platform that spans the continuum of the resources in different EIPs, from cloud to the edge. With such the cross-EIP method, edge federation can bring the customized resources (e.g., computation, storage, and networking resources) for ESPs and end users in a broad, fast, and reliable geo-distributed manner.

The most similar idea to the edge federation is the cross-cloud cooperation architecture in previous works. These works attempted to establish the integrated cloud resources provisioning architecture, which could be named as cloud federation, such as the Joint Cloud [4], the Hybrid Cloud [8], etc.. The cloud federation tries to establish the environment that combines the public and the private resources, which can enable EIPs scale resources for handling short-term spikes (e.g., Black Friday in the Amazon, Single's day in the Taobao, etc.) in demand [4]. It can be seen as the horizontal integration mentioned before. Some works also considered vertical integration in the field of content caching or computation offloading. Most of these works construct the cloud-assisted [7] or edge-assisted [9] network structure, both of which try to solve two main problems: the limitation of the edge resource capacity and the long latency caused by the backhaul network from users to the cloud.

Compared with the works mentioned above, the edge federation is much different and even more challenging, which involves both horizontal and vertical integrations. The resource integration in the edge federation could be more complicated and urgent, mainly due to three aspects of the edge computing: the highly distributed, limited and heterogeneous edge resources; the high cost of edge resources; the latency-critical and computation-intensive edge services. Based on these characteristics, we have to solve several particular challenges in edge federation.

1) *The trade-off between the cloud and the edge:* As described in the previous section, the edge can achieve the lower service latency but with higher cost. In contrast, the cloud introduces the lower cost but with a higher latency. Neither of them can meet the high latency requirement of services and low-cost resource provision simultaneously. Thus, the goal of the edge federation is trying to strike a balance between the latency and the cost, either the trade-off between the cloud and the edge. How to take the least cost to fulfill the service requirements and achieve the best QoS is the most critical thing in the edge federation.
2) *The optimization of the service deployment on distributed and limited edge resources:* Compared with cloud nodes, edge nodes are much more scattered in geography with the very limited amount of resources.

Due to such the limitation, EIPs have to be very careful when they provide the resource to services. This severely restricts their capacity in size of the serving area and service demands. Thus, the cooperation of different EIPs and the optimization of resource provision in the edge federation are more urgent than them in the cloud federation. The challenge is how to maximize the resource provisioning efficiency in the highly distributed and limited edge resources.

3) *The contradiction between the computation-intensive edge services and the limited edge resources:* The resources in edge nodes are very limited. Worse still, most of the emerging services in edge scenario have high computation and strict latency requirements, which require significant and sufficient computation and storage resources. This dilemma makes the edge more likely to get into the "Spike" trouble and suffer from resource shortages, especially in the peak hours and the downtown areas.

For these challenges, an efficient service provisioning method is urgently needed. In the following section, we first design the architecture of the edge federation, under which sophisticated service provisioning methods can be developed.

III. THE ARCHITECTURE OF EDGE FEDERATION

We start with an initial example and an overview of the edge federation and then present the detailed architecture of the edge federation.

A. Rationale

As shown in the left of Fig. 1, existing network environment mainly has three layers: (i) the user layer consists of a large amount of heterogeneous smart devices (e.g., mobile phones, vehicles, etc.), which dynamically request for high-performance services from ESPs; (ii) the edge layer is formed by EIPs, which are responsible for resource provisioning for ESPs. They provide computation and storage resources for ESPs, as well as techniques (e.g., NFV and SDN) and platforms (e.g., Amazon Web Services, Microsoft Azure ,etc.) for services to run on; (iii) the cloud layer provides similar type but larger amount of resources to end users. In the current network environment, ESPs usually sign contracts and package their services content to given EIPs. An EIP can only manage their own resources and deliver the contracted services to corresponding end users. There is no interaction among the edge nodes of different EIPs.

The current service-provisioning model can make the individual edge node suffer resource limitation and extra-high cost. **From the perspective of resources**, EIPs independently deploy edge nodes at the edge of the network, where each edge node consisting of multiple edge servers provides computation and storage resources for accommodating diverse services. The capacity and the serving range of an individual edge node is much smaller than that of the cloud. Moreover, EIPs independently manage their resources without any cooperation in the current edge-computing model. Consequently, such

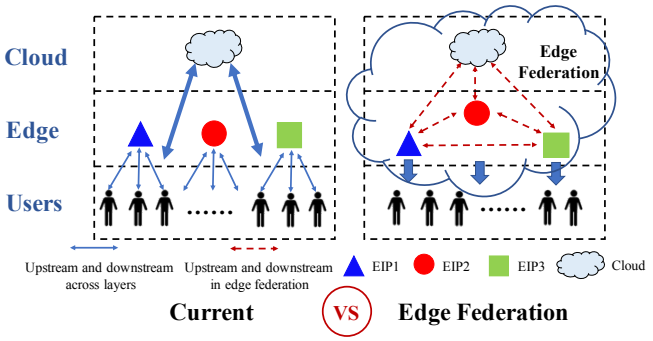


Fig. 1. The comparison between the current and the edge federation. In the future, the edge federation consortium realizes a transparent resource infrastructure over EIPs and cloud.

mechanism fails to enable globally optimal scheduling of resources and services, hence leading to the resource overloaded or under-utilization situation and resulting in the low QoS to end users. **From the perspective of cost**, each EIP tends to build more edge nodes in new locations for increasing the amount of resources and expanding the serving geographical range. Multiple EIPs even build edge nodes in the same location for the market competition. Such a method would definitely cause a huge overhead (e.g., the expenditure of infrastructure construction, maintenance cost, energy cost ,etc.) and severe resource waste; hence, it is not scalable. Eventually, such heavy burdens will be taken by EIPs, ESPs, and end users simultaneously in this triple-lose situation.

To overcome the above disadvantages, we propose the edge federation, a transparent service-provisioning model in the multi-EIP environment. It involves two-dimension integration for the service deployment, including the integration between edge and cloud, and the seamless cooperation among heterogeneous edge nodes of multiple EIPs. The basic idea of edge federation is shown on the right side of Fig. 1, where each EIP and cloud is a member of edge federation and the edge nodes of all EIPs and cloud nodes can share resources and interact with each other. Those edge nodes and cloud nodes are not necessary to be genuinely interconnected. They only disclose the details of edge nodes and cloud nodes to the authoritative and trusted consortium, which is the core of edge federation.

B. Architecture of Edge Federation

As shown in Fig. 2, the edge federation consortium mainly consists of three components, including the traffic analyzer, the central optimizer, and the dispatcher.

Traffic analyzer is a module that continuously analyzes the traffic pattern, based on the dynamic requests of different ESPs from end users at various locations. The traffic patterns can accurately characterize the service demands temporally and spatially and will serve as an essential input to the central optimizer. Consider that many proposals have devoted to traffic prediction and modeling. Thus, assumed that we use the existing methods (e.g., ARIMA [10]) in our traffic analyzer

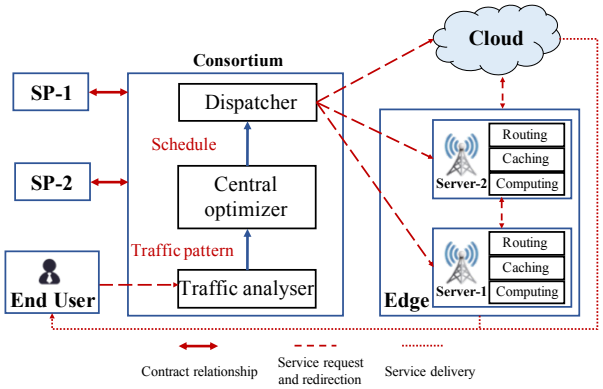


Fig. 2. The architecture of edge federation.

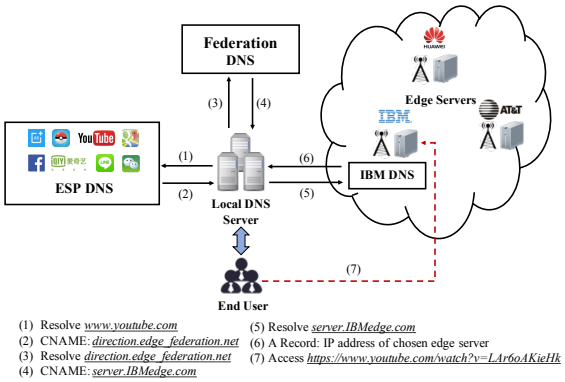


Fig. 3. An example of request direction.

to predict the traffic.[1] A comprehensive study of the traffic prediction and modeling is out of the main scope of this paper.

Central optimizer is the brain of the edge federation consortium. It computes the traffic redirection schedule based on the obtained traffic pattern, the information about end users (e.g., location area, time, type of service) and so on. Then the optimized solution will be sent to the dispatcher as the basic information of traffic redirection. Hence, EIPs deploy the corresponding services on the edge and cloud according to the optimization result.

The *dispatcher* redirects users' service requests to correct edge servers. Such redirection can be performed by the existing routing mechanisms (DNS CNAME record, A record). To ease the understanding, we present a detailed example of service redirection based on the DNS technique in Fig. 3. The end user at a specific location area requests a video of the *Youtube*. Compared to the traditional model, the EIP DNS modify its CNAME record to point to the domain of a federation DNS instead of the contracted EIP DNS domain. Based on another CNAME record, the consortium dispatcher will redirect the user's request to the optimal edge server. Thus the high-performance service can be achieved.

C. Benefits of Edge Federation

1) For the business relationship: Traditionally, ESPs will deploy services on the infrastructure of EIPs with the pay-as-you-go function. And different EIPs will manage their resources and deliver services to the end user individually. The difference between the money ESPs paid to EIPs, and the operation cost (e.g., storage cost, computation cost, communication cost, etc.) of EIPs is the revenue of the EIP. In the edge federation, ESPs will also deploy services on the EIP and pay the money to EIP with the same price as in the conventional method. However, these services will be deployed by the edge federation with a global view of the unified resource pool, which consists of edge nodes from different EIPs. Then, the edge node will deliver the corresponding service to the end user.

2) For EIPs: : EIPs in the traditional method can only manage the corresponding services delivery on their own edge nodes in limited areas. Compared with the old method, edge federation makes it possible that EIPs can operate the service more flexible among the unified resource pool. Such a method can help EIPs deliver the service to end users with the shorter distance, less infrastructure construction, and thus enable a more cost-efficiency service deployment and delivery with the reasonable edge cooperation and cloud assist. Therefore, the operation cost of the EIP can be reduced, and the revenue of the EIP can be improved.

3) For ESPs: : In existing method, due to the limited coverage area of edge nodes of the single EIP, the corresponding ESP only can spread their service in a considerable small region. This means that the ESP has a limited market size, while such the situation will no longer exist in the edge federation, attributed to the wide and dense distributed edge nodes of different EIPs in the unified resource pool. Moreover, with the same unit price, ESPs will get a higher QoS.

4) For end users: : The edge federation makes the ESPs deploy their services on demand over any parts and even all nodes of multiple EIPs. These edge nodes can distribute in a variety of geographical locations. As a result, end users can get the services from closer nodes with lower latency to some extent no matter where the user locates. Moreover, the reliability of service delivery is also considerably enhanced.

IV. Optimal Service Provision via Edge Federation

After designing the architecture of edge federation, we need to decide where and when the ESPs transparently launch various services to end users. We further derive how much resources (e.g., computation and storage) should be allocated to operate each edge service.

A. Modeling the Service Provision

1) Network Environment and Dynamic Service Demands

For an edge-computing network, there are various edge nodes, each of which may consist of multiple edge servers. We assume that end users are geographically distributed according

[1]The short-term prediction (e.g., conventional methods: ARIMA [10], etc., computation intelligence methods with off-line training and on-line prediction: LSTM [11], etc..) has been intensely developed and proven to be reliable with high prediction accuracy. They can provide proper input to the further optimization.

TABLE I
MAIN NOTATIONS

Notation	Description
T	A time period of n consecutive time slots
U	Set of end user clusters
P	Set of SPs
A	Set of cloud servers a in EIPs
E	Set of edge servers e in EIPs
$\alpha^e_{u,p}(t)$	Fraction of caching demands of SP p from end user location u assigned to edge server e at time slot t
$\beta^e_{u,p}(t)$	Fraction of computation demands of SP p from end user location u assigned to edge server e at time slot t
$\theta^s_{u,p}$	Fraction of edge federation store content of SP p on cloud
$\theta^c_{u,p}$	Fraction of edge federation deploy computation of SP p on cloud
$S_{u,p}(t)$	Amount of storage demands of SP p from end user cluster u at time slot t before computation
$S^{'}_{u,p}(t)$	Amount of delivery content of SP p from end user cluster u at time t after computation
$C_{u,p}(t)$	Amount of computation demands of SP p from end user cluster u at time slot t
S_a	Storage capacity of cloud a
C_a	Compuation capacity of cloud a
S_e	Storage capacity of edge e
C_e	Computation capacity of edge e
$S_E(t)$	Overall amount of storage demands deployed on edge at time slot t
$C_E(t)$	Overall amount of computation demands deployed on edge at time slot t
$l_{u,p}(t)$	Service latency of service p for end user cluster u at time slot t
h^a_u	Delivery distance between cloud server a and end user cluster u
h^e_u	Delivery distance between edge server e and end user cluster u
l_p	Latency requirement of service p
$m_{u,p}(t)$	Service satisfaction parameter indicates whether the service meets the latency requirement
l^{sat}_p	Satisfaction ratio of specific service p

to the locations of edge nodes. Generally, there are four roles in the entire edge-computing network. Define U as the set of all end users, A as the set of cloud nodes, E as the set of edge nodes, and P as the set of edge services, respectively. Let $u \in U$ represents a specific user, $a \in A$ represent a specific cloud node, $e \in E$ represents a specific edge node, while $p \in P$ represent a specific edge service. For simplicity, we assume that the topology of the designed edge federation is known in advance. The main notations are shown in Table. I.

The end users have time-varying service demands toward the storages and computations. The service demands within a time period T could be divided into n smaller equal time slots, e.g., 1 hour. Let the service demands p from end user u at time slot t is denoted as $K_{u,p}(t) = \{S_{u,p}(t), S^{'}_{u,p}(t), C_{u,p}(t)\}$ ($\forall t \in T, t = 1, 2, \ldots, n$). Here, $S_{u,p}(t)$ and $S^{'}_{u,p}(t)$ represent the amount of content before and after processing, respectively, while $C_{u,p}(t)$ denotes the computation demands for accomplishing the service. These terms can be captured by the following generation formulations:

$$\sum_{u \in U} S_{u,p}(t) := |U| \cdot q_p(t), \forall p \in P \tag{1}$$

$$S^{'}_{u,p}(t) := S_{u,p}(t) \cdot k_s, \forall u \in U, \forall p \in P, \forall t \in T \tag{2}$$

$$C_{u,p}(t) := S_{u,p}(t) \cdot k_c, \forall u \in U, \forall p \in P, \forall t \in T \tag{3}$$

where $|U|$ refers to the population in the target location area, $q_p(t)$ is the normalized traffic profile whose value is related to a specified service. k_s is the coefficient profile that describes the size of the content after processing, and k_c is the coefficient profile that describes the required computation resource for accomplishing the corresponding task. The traffic demands of the service p in the related area around edge server e at time t can be also captured by the following model:

$$d_{ep}(t) = |U|_e \cdot q_p(t), \forall p \in P \tag{4}$$

where $|U|_e$ represents the population of specified edge server location.

***Remark 1**: As the service demands of users are highly dynamic, the tailored time slot is necessary to the cost-efficiency schedule of the edge federation. Our implementation in later experiments shows that the time slot with a length of 1 hour is good enough to yield the better result than the existing method.*

2) Two-phase Resource Allocation

From the vertical, we assume that each EIP select to resolve part or all the storage demands $S_{u,p}(t)$ and computation demands $C_{u.p}(t)$ by cloud nodes. The two variables $\theta^{S,a}_{u,p}(t)$ and $\theta^{C,a}_{u,p}(t)$ represent the fraction of storage demands supplied by cloud node a at time slot t, respectively, while the other $(1 - \sum_{a \in A} \theta^{S,a}_{u,p}(t))$ and $(1 - \sum_{a \in A} \theta^{C,a}_{u,p}(t))$ demands served by the edge nodes. Obviously, the settings of these fractions are all between $[0, 1]$:

$$0 \leq \theta^{S,a}_{u,p}(t) \leq 1, \forall u \in U, \forall p \in P, \forall t \in T \tag{5}$$

$$0 \leq \theta^{C,a}_{u,p}(t) \leq 1, \forall u \in U, \forall p \in P, \forall t \in T \tag{6}$$

In any time slot t, the storage and computation demand should not exceed the capacity of the involved cloud nodes and edge nodes. Thus, we have the following two constraints:

$$\sum_{u \in U} \sum_{p \in P} S_{u,p}(t) \theta^{S,a}_{u,p}(t) \leq S_a, \forall t \in T \tag{7}$$

$$\sum_{u \in U} \sum_{p \in P} C_{u,p}(t) \theta^{C,a}_{u,p}(t) \leq C_a, \forall t \in T \tag{8}$$

where S_a and C_a are the storage and computation capacity of a specific cloud node a. Given the dynamic demands from each location area of end users, we assume that the least cloud capacities should satisfy the peak service demands, thus:

$$\sum_{a\in A} S_a = \max_{t\in T}\{\sum_{u\in U,p\in P,a\in A} S_{u,p}(t)\theta^{S,a}_{u,p}(t)\} \quad (9)$$

$$\sum_{a\in A} C_a = \max_{t\in T}\{\sum_{u\in U,p\in P,a\in A} C_{u,p}(t)\theta^{C,a}_{u,p}(t)\} \quad (10)$$

Noting that the service demands of end users may change over time, so the edge federation needs to manage available resources and charge resource consumption hourly without long-term commitments. Thus, the storage capacity $\sum_{a\in A} S_a$ and computation capacity $\sum_{a\in A} C_a$ are not fixed in our model. Moreover, compared to access an edge node, the latency of accessing a cloud node is higher, however, with lower resource cost. Therefore, we need to find the reasonable variables $\theta^{S,a}_{u,p}(t)$ and $\theta^{C,a}_{u,p}(t)$ to achieve a better trade-off between the cloud and the edge dynamically via the transparent optimization of the edge federation.

From the horizontal, overall storage demands and computation demands supplied by all edge nodes are:

$$S_E(t) = \sum_{u\in U}\sum_{p\in P} S_{u,p}(t)(1-\sum_{a\in A}\theta^{S,a}_{u,p}(t)) \quad (11)$$

$$C_E(t) = \sum_{u\in U}\sum_{p\in P} C_{u,p}(t)(1-\sum_{a\in A}\theta^{C,a}_{u,p}(t)) \quad (12)$$

Note that the edge federation prefers to first schedules and redirect those service demands from end users to involved edge nodes of multiple EIPs. To make such schedules, we use the two variables $\alpha^e_{u,p}(t)$ and $\beta^e_{u,p}(t)$ to denote the fraction of storage demands $S_{u,p}(t)$ and computation demands $C_{u,p}(t)$ supplied by edge node e at time slot t, respectively. Thus, we have the following constraints:

$$0 \le \alpha^e_{u,p}(t) \le 1, \forall u\in U, \forall p\in P, \forall e\in E, \forall t\in T \quad (13)$$

$$0 \le \beta^e_{u,p}(t) \le 1, \forall u\in U, \forall p\in P, \forall e\in E, \forall t\in T \quad (14)$$

We define the storage and computation capacity of an edge node as S_e and C_e, respectively. They represent the maximum demands the edge node can serve in a single time slot. We have the following constraints:

$$\sum_{u\in U}\sum_{p\in P} S_{u,p}(t)\alpha^e_{u,p}(t) \le S_e, \forall e\in E, \forall t\in T \quad (15)$$

$$\sum_{u\in U}\sum_{p\in P} C_{u,p}(t)\alpha^e_{u,p}(t) \le C_e, \forall e\in E, \forall t\in T \quad (16)$$

Formulas (15) and (16) mean that the storage and computation demands assigned to the edge node e should not exceed its capacity at any time slot.

Another important constraint is that those demands from all users should be completely accomplished. Thus, we have

$$\sum_{e\in E}\alpha^e_{u,p}(t) + \sum_{a\in A}\theta^{S,a}_{u,p}(t) = 1, \forall u\in U, \forall p\in P, \forall t\in T \quad (17)$$

$$\sum_{e\in E}\beta^e_{u,p}(t) + \sum_{u\in U}\theta^{C,a}_{u,p}(t) = 1, \forall u\in U, \forall p\in P, \forall t\in T \quad (18)$$

3) Cost Minimization for the Edge Federation

To serve a set of users' demand, the edge federation treats to the minimization of the overall cost (i.e., maximize revenue) as an important optimization goal. Under the edge-computing scenario, the overall cost of an edge federation, V, can be divided into three parts, including the computation cost, the storage cost, and the communication cost.

***Remark 2**: The cost in the edge server is similar to the cost in conventional cloud server [12], which consists of the cost of the servers, the infrastructure, the networking, and the power draw [13]. Since the infrastructures have been built. EIPs have to pay the costs to maintain them no matter they are used or not. Thus, the cost of infrastructures will not be considered in this paper. We mainly considered the server cost (e.g., the computation cost and the storage cost) and the networking cost (e.g. the communication cost) in this paper. And the related power or energy cost will be properly "absorbed" in the components modeled in the service data storage, service computation, and service delivery process.*

Therefore, during a time period T, the servers' cost on cloud nodes can be written as:

$$\begin{aligned} V^{cloud} &= V^{cloud}_S + V^{cloud}_C + V^{cloud}_M \\ &= \sum_{u\in U,p\in P,a\in A,t\in T} S_{u,p}(t)\theta^{S,a}_{u,p}(t)V_S \\ &+ \sum_{u\in U,p\in P,a\in A,t\in T} C_{u,p}(t)\theta^{C,a}_{u,p}(t)V_C \\ &+ \sum_{u\in U,p\in P,a\in A,t\in T} (S_{u,p}(t)+S'_{u,p}(t))\theta^{C,a}_{u,p}(t)V_M \end{aligned} \quad (19)$$

where V^{cloud}_S, V^{cloud}_C and V^{cloud}_C are the cost of storage, the cost of computation and the cost of communication in cloud nodes, respectively. V_S, V_C, and V_M are the cost of per storage unit, the cost of per computation unit and the cost of per communication unit, respectively. The servers' cost on edge nodes is:

$$\begin{aligned} V^{edge} &= V^{edge}_S + V^{edge}_C + V^{edge}_M \\ &= \sum_{u\in U,p\in P,e\in E,t\in T} S_{u,p}(t)\alpha^e_{u,p}(t)V^e_S \\ &+ \sum_{u\in U,p\in P,e\in E,t\in T} C_{u,p}(t)\beta^e_{u,p}(t)V^e_C \\ &+ \sum_{u\in U,p\in P,e\in E,t\in T} (S_{u,p}(t)+S'_{u,p}(t))\beta^e_{u,p}(t)V^e_M \end{aligned} \quad (20)$$

where V^{edge}_S, V^{edge}_C, V^{edge}_M are the cost of storage, the cost of computation and the cost of communication on edge nodes, respectively. V^e_S, V^e_C and V^e_M are the cost per storage unit, the cost per computation unit and the cost per communication unit of a specific edge node e, respectively.

***Remark 3**: The resource usage price is relatively stable in the current cloud computing market. Thus we set all cloud nodes with the same storage, computation and communication cost per unit. However, for the edge computing, the resource market is still in an initial and unstable stage, and the resource price of an edge node resources in each EIP is quite different [5], [6]. Therefore, the edge nodes of different EIPs can require different storage, computation, and communication price in our edge federation model.*

Thus, the total cost of all involved edge servers and cloud servers in an edge federation can be written as:

$$V = V^{cloud} + V^{edge} \tag{21}$$

The optimization goal is to minimize V in the network over a certain time period. It is worth note that the final optimization result should be subjected to the strict service latency requirements.

B. Guaranteeing the Service Performance

1) Modeling the Service Latency

The accessing latency is the key factor affecting service performance and can be roughly divided into two components, including computing latency and content delivery latency. The computing latency is the time consumption of completing the computation process of that service deployed on it. For an end user u, the computing latency of that service p on the cloud or edge servers could be measured by the followings formulas, respectively:

$$l_{u,p}^{cloud,C}(t) = \sum_{a\in A} C_{u,p}(t)\theta_{u,p}^{C,a}(t)\frac{r_p}{C_a}, \forall u \in U, \forall p \in P, \forall t \in T \tag{22}$$

$$l_{u,p}^{edge,C}(t) = \sum_{e\in E} C_{u,p}(t)\beta_{u,p}^{e}(t)\frac{r_p}{C_e}, \forall u \in U, \forall p \in P, \forall t \in T \tag{23}$$

where the parameter r_p represents the computation capacity of service p required by the end user u and is related to the service category (e.g., social networking, gaming, etc.). Note that the computation resources offered by the edge are still very limited, compared to the extra-large computation resources provided by the cloud. Thus, we have $C_a \gg C_e$ in general.

The delivery latency could be divided into the uploading delivery latency and the downloading delivery latency. Users usually get service through a one-hop transmission. Thus, we use the delivery distance instead of the hop distance to estimate the delivery latency in this model. We use h_u^a and h_u^e to denote the delivery distance from cloud node a and edge node e to end user u, respectively. First, the service data should be transferred from the users to the servers. The uploading delivery latency in the cloud and the edge in time slot t can be estimated as followings, respectively:

$$l_{u,p}^{cloud,up}(t) = \sum_{a\in A} S_{u,p}(t)\theta_{u,p}^{S,a}(t)h_u^a, \forall u \in U, \forall p \in P, \forall t \in T \tag{24}$$

$$l_{u,p}^{edge,up}(t) = \sum_{e\in E} S_{u,p}(t)\alpha_{u,p}^{e}(t)h_u^e, \forall u \in U, \forall p \in P, \forall t \in T \tag{25}$$

Then, after processed in the servers, the processed service data will be returned to the users. Thus, the downloading delivery latency in the cloud and the edge in time slot t can be desribed as followings, respectively:

$$l_{u,p}^{cloud,do}(t) = \sum_{a\in A} S_{u,p}^{'}(t)\theta_{u,p}^{S,a}(t)h_u^a, \forall u \in U, \forall p \in P, \forall t \in T \tag{26}$$

$$l_{u,p}^{edge,do}(t) = \sum_{e\in E} S_{u,p}^{'}(t)\alpha_{u,p}^{e}(t)h_u^e, \forall u \in U, \forall p \in P, \forall t \in T \tag{27}$$

2) The Constraint on the Service Latency

The service demands of services usually vary temporally and spatially for heterogeneous end users. Thus, we should make sure that the required performance of services (e.g., latency requirement) can be guaranteed by the schedule solution in the edge federation. Let l_p to denote the required latency of accessing service p. In any time slot t, only when the actual latency does not exceed l_p, the service can be regarded as satisfied in that time slot. Therefore, the relationship between the actual latency and required latency can be defined as followings:

$$\begin{aligned} l_{u,p}(t) &= l_{u,p}^{cloud}(t) + l_{u,p}^{edge}(t) \\ &= [l_{u,p}^{cloud,S}(t) + l_{u,p}^{cloud,C}(t)] + [l_{u,p}^{edge,S}(t) + l_{u,p}^{edge,C}(t)] \\ &\le l_p \end{aligned} \tag{28}$$

where $l_{u,p}(t)$ denotes the actual latency of service p from end user u in time slot t. Then, we use a satisfaction parameter $m_{u,p}(t)$ to represent whether a service demand of the user u is guaranteed, which can be defined as:

$$m_{u,p}(t) = \begin{cases} 1 & , \quad l_{u,p}(t) \le l_p, \\ 0 & , \quad l_{u,p}(t) > l_p. \end{cases} \tag{29}$$

Moreover, edge federation needs to keep the corresponding services at a high-level performance in the business environment to attract more users and improve revenues. The overall performance of service p in edge federation can be measured by the satisfaction ratio r_p^{sat}, which can be written as:

$$r_p^{sat} = \frac{\sum\limits_{t\in T}\sum\limits_{u\in U} S_{u,p}(t)m_{u,p}(t)}{\sum\limits_{t\in T}\sum\limits_{u\in U} S_{u,p}(t)} \tag{30}$$

According to the existing industry standards, the satisfaction ratio should reach following range:

$$l_1 \le r_p^{sat} \le l_2 \tag{31}$$

where l_2 could be 100%, and l_1 is usually larger than 99%.

***Remark 4**: The service satisfaction ratio of service p is evaluated by the satisfied service demands of each user in every time slot. Thus, we accumulate those service demands, whose requirement of the latency have been satisfied, to calculate the satisfaction ratio of a specific service. Noted that calculating the service satisfaction with a global measure is inaccurate, such as the average service latency, due to the*

potential uneven distribution (e.g., a bimodal distribution) of the service latency for each user.

Therefore, under the latency constraints, the problem that the central optimizer needs to solve for the edge federation can be formulated as the following optimization problem:

$$\min_{\{\theta^{S,a}_{u,p}(t),\theta^{C,a}_{u,p}(t),\alpha^{e}_{u,p}(t),\beta^{e}_{u,p}(t)\}} \quad V \tag{32a}$$

$$\text{s.t.} \quad (5) \sim (8), (13) \sim (18) \quad \text{and} \quad (31). \tag{32b}$$

Through solving this optimization problem, we can find the optimal resource assignment schedules (e.g. optimal caching and computing variables) of an edge federation in every time slot.

V. Problem Transformation and Dynamic Resolving Algorithm

In this section, we propose a dimension-shrinking method to reformulated the optimization problem into an easily solved model. Based on this method, we further develop a dynamic service provisioning algorithm to deal with varying service demands.

A. Problem Transformation

Due to those time-varying factors, i.e., $\alpha^{e}_{u,p}(t)$, $\beta^{e}_{u,p}(t)$, and $\forall t \in T$, the optimization problem mentioned in Formula (32) is very hard to be solved with existing solvers for the LP optimization. Therefore, we prefer to reformulate this problem as a low dimension optimization problem so that it can be solved efficiently using the off-the-shelf solvers.

We use V^{edge}_{S}, part of the V_O, as an example to illustrate the transforming process. To ease understanding, we begin with a simple scenario where only one service and a single time slot (i.e., $|P|$=1, $|T|$=1) are considered. Thus, the original three-dimensional variables of the caching in edge federation is converted into two dimensional variables, i.e., α^{e}_{u}, where $u \in U$, $e \in E$. Assume that $|U| = i$, $|E| = j$, the storage variable can be written as the matrix as follows:

$$\boldsymbol{\alpha} := \begin{bmatrix} \alpha^1_1 & \alpha^2_1 & \cdots & \alpha^j_1 \\ \alpha^1_2 & \alpha^2_2 & \cdots & \alpha^j_2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha^1_i & \alpha^2_i & \cdots & \alpha^j_i \end{bmatrix} \tag{33}$$

where each α^{e}_{u} means the fraction of storage demands, which result from user u and are assigned to edge node e. Let the vector $\boldsymbol{S}$ denote the amount of storage demands from each of these end user. The vector $\boldsymbol{V^E_S}$ represent the cost of per storage unit in different edge nodes. Therefore, the V^{edge}_{S} part can be formulated as:

$$V^{edge}_{S} := \|(\boldsymbol{S})^T \boldsymbol{\alpha} \boldsymbol{V^E_S}\|_1 \tag{34}$$

So far, we consider the solution for a more general case that includes multiple services and multiple time slots (i.e., $|P| = m$ and $|T| = n$). In this case, the storage variable can be converted into a super matrix that consists of $m * n$ aforementioned simple matrices (refer to (33)). Thus, the caching variable $\alpha^{e}_{u,p}(t)$ could be extended as following:

$$\widehat{\boldsymbol{\alpha}} := [\boldsymbol{\alpha}(1), \boldsymbol{\alpha}(2), \cdots, \boldsymbol{\alpha}(m*n)]^T \tag{35}$$

where each simple matrix $\alpha(l)$ represents the storage variable of a certain EIP b in a specific time slot t, and $m*(t-1)+b = l$.

The vector $\boldsymbol{S}$ and $\boldsymbol{V^E_S}$ could also be extended for the general case as follows:

$$\widehat{\boldsymbol{S}} := [\boldsymbol{S}(1), \boldsymbol{S}(2), \cdots, \boldsymbol{S}(m*n)]^T \tag{36}$$

$$\widehat{\boldsymbol{V^E_S}} := [\boldsymbol{V^E_S}(1), \boldsymbol{V^E_S}(2), \cdots, \boldsymbol{V^E_S}(m*n)] \tag{37}$$

in both of which each $\boldsymbol{S}(l)$ and $\boldsymbol{V^E_S}(l)$ represent the storage demand vector and the edge caching cost vector of EIP b in time slot t, respectively. Thus, $m*(t-1)+b = l$. The V^{edge}_{S} could be converted to:

$$V^{edge}_{S} := \|(\widehat{\boldsymbol{S}})^T \widehat{\boldsymbol{\alpha}} \widehat{\boldsymbol{V^E_S}}\|_1 \tag{38}$$

Similarly, the other parts of the cost of EIPs (e.g., V^{edge}_{C}, V^{cloud}_{S}, V^{cloud}_{C}) can be transformed into matrices and vectors using the similar procedure.

After the above transformation, the problem (32) could be solved efficiently with existing LP solvers such as CVX Gurobi solver.

B. Service Provisioning Algorithm

After the transformation mentioned above, we further develop a dynamic resolving algorithm, namely *SEE* (Service provision for Edge fEderation). It allows EIPs to achieve an efficient service provisioning solution in the edge federation environment.

To be more specific, as shown in **Algorithm 1**, our algorithm is developed under the dynamic service demands; thus the service provisioning should be rescheduled in each time slot. We take the storage and computation capacities of cloud nodes and edge nodes as (C_a, S_a, C_e, S_e). The profile of services' computation and latency requirements (r_p, l_p), and the transmission distances (h_e, h_a) act as the inputs of our algorithm.

Algorithm 1 *SEE* algorithm

Input : C_a, S_a, C_e, S_e, r_p, l_p, h_e, h_a

Output : edge storage variable $\alpha^{e}_{u,p}(t)$ d, edge computation variable $\beta^{e}_{u,p}(t)$, cloud storage variable $\theta^{S,a}_{u,p}(t)$, and cloud computation variable $\theta^{C,a}_{u,p}(t)$;

1: **for** t_1 to t_n **do**
2: Predict the service demands of different services $K_{u,p}(t)$=($S_{u,p}(t_i)$, $S^{'}_{u,p}(t_i)$, $C_{u,p}(t_i)$)
3: Update the $\alpha^{e}_{u,p}(t_i)$, $\beta^{e}_{u,p}(t_i)$, $\theta^{S,a}_{u,p}(t_i)$, $\theta^{C,a}_{u,p}(t_i)$ by solving the optimization problem (37)
4: Calculate the cost of EIPs at time slot t_i: $V(t_i) = (V^{edge}(t_i) + V^{cloud}(t_i))$
5: **end for**

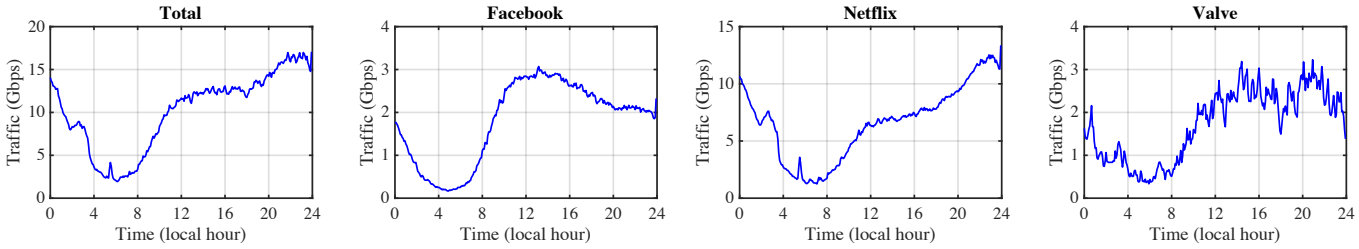


Fig. 4. Traffic demands of different services at NORDUnet nodes on May. 07, 2017.

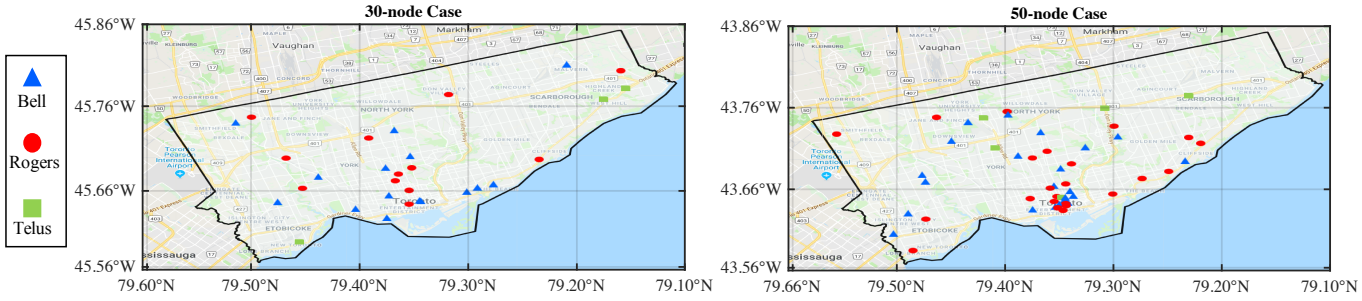


Fig. 5. The base station map of Toronto city.

In each time slot, we firstly predict the demands of services by well-studied method (e.g., ARIMA). This short-term prediction is conducted for the usage in the next time slot. Many prior studies show that such short-term prediction is more accurate than the long-term prediction. Based on the prediction results, the edge federation could calculate the schedule for the next time slot in advance, by solving the optimization problem (32). Such an optimization process is mainly executed by the consortium of edge federation for enabling dynamic optimal service provisioning. It decides how much workload retain at the edge or offload to the cloud, and how to deploy services among heterogeneous edge servers and cloud servers.

VI. Experimental Evaluation

In this section, we conduct trace-driven experiments over the base station network in Toronto and evaluate the performance of our service provisioning model under a multi-EIP network environment. We measure the performance of the edge federation in terms of the total cost of EIPs for serving a given set of edge services.

A. Experiment Setting

1) Designed Network

We first achieve the datasets about the edge-computing environment from the published data of Canada government [14], which provides the details of the location and brand of base stations all over Canada. The designed network of our experiments is constructed across the region of the Toronto city. As shown in Fig. 5, we carefully select the amount of 30 and 50 base stations as the potential edge nodes by fully considered the density of the population (e.g., the downtown area is more likely to have a higher density of population than the rural area, thus the higher density of edge nodes.) and the business condition in different areas of the city. In common sense, more edge nodes are needed in the prosperous and populous area.

All of such base stations are chosen from three popular telecommunication providers in Canada, including the Bells, Rogers ,and Telus. We then select the Amazon Datacenter in Montreal, the Google datacenters in the United States as the potential cloud nodes in our experiments.[2] We make use of the base station dataset for the following reasons: i) Upgrading base stations as edge nodes is a reasonable and accessible solution for the construction of future edge-computing environment. ii) The datasets have specific location information of BSs, and thus the content delivery latency could be accurately estimated. This is based on the fact that RTT between two nodes is approximately linear with their geo-distance [16]. Therefore, these selected edge nodes and cloud node make up the designed network in this paper.

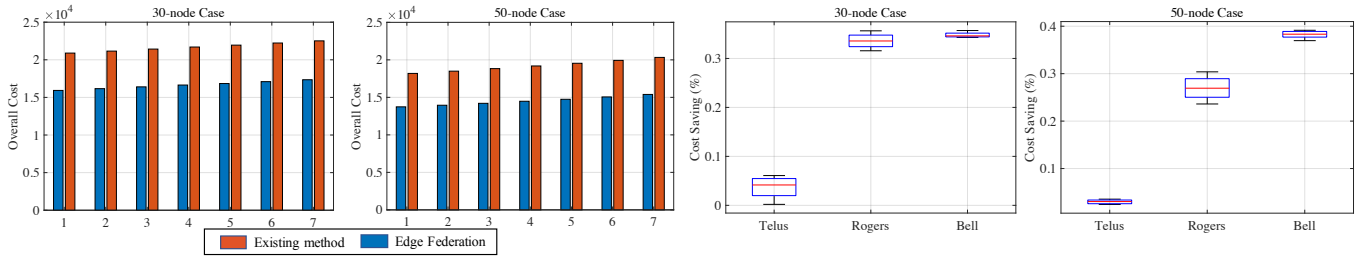
2) Service Demands of End Users

We collect the traffic data from the NORDUnet,[3] a research and education oriented network infrastructure that hosts cache servers at various peering points over Europe and North America. By using these real-world trace data, we generate synthetic service demands of each end user in our designed network.

Dynamic service demands: From the diverse requirements of latency, we mainly consider three types of services, including online gaming, online video, and social media. They represent the high, normal, and low latency requirements, respectively. Thus, we correspondingly select three representative services, including Valve, Netflix, and Facebook. Fig. 4

[2]Under the common situation, most of the users in the world are hundreds of kilometers away from the data center, and some of them even need to get the service from the data center continental distance away [15]. To make our experiment more representative, we carefully select the data centers far away from Toronto city as the potential cloud nodes.

[3]http://stats.nordu.net/connections.html



(a) The overall service provisioning cost of all EIPs under the 7 latency requirements, during the 24 hours.

(b) Ranges of savings with edge federation provisioning model with varying latency requirements over the 30-node case and 50-node case, repectively.

Fig. 6. The overall service provisioning performance of three EIPs in the fixed contract model and the edge federation model.

TABLE II
LATENCY REQUIREMENTS FOR THREE SERVICES.

Group / Service	1	2	3	4	5	6	7
Facebook	72	68	64	60	56	52	48
Valve	36	34	32	30	28	26	24
Netflix	54	51	48	45	42	39	36

shows the traffic curves in a 24-hour time window on May. 7, 2017. Finding some details about the differences among services is interesting. Netflix accounts for the most significant portion of traffic. The peak demands of Valve and Netflix appear at night and also the total demands, while the peak demands of Facebook appear in the daytime.

Synthetic Traffic Generation: Referring to the realistic service demand patterns as above, we generate synthetic traffic demands for further model evaluation. First, we normalize the traffic demand of each service as the traffic profile, i.e., $q_p(t), \forall t \in T$. Then, we collect the amount and density of the population of Toronto, from the online published data [17], [18]. Thus, we could generate the synthetic service demands and patterns in the location area of each end user by calculating Formula (1)-(3) for each type of service. These traffic patterns are treated as the result of *Traffic Analyser* in Fig. 2 and sent to the *Central Optimizer* for calculating the optimal service provisioning and request schedules.

B. Performance Evaluation

In this part, we analyze the service provisioning process of Telus, Rogers, and Bell in both the 30-node case and 50-node case. We mainly compare two service provisioning strategies: the edge federation model and the fixed one-one contract model (e.g., the existing model). For testing the performance of the fixed contract model, we assume that several fixed relationships: Telus contracts with Facebook, Rogers contracts with Valve, while Bell contracts with Netflix. In such a multi-EIP network environment, we evaluate the performance of the existing service provisioning model and our edge federation model by the cost of EIPs, under different latency requirements of services and the varying amount of service demands. More precisely, we consider not only the total cost but also the average cost, which could be defined as follow:

- The total cost: the overall cost in total 24 time slots for all EIPs, which can be calculated by Formula (21).
- The average cost: the average cost of each EIP for each end user at time slot t can be defined as

$$v_{u,p}(t) = [\sum_{u\in U}\sum_{e\in E} S_{u,p}(t)\alpha^e_{u,p}(t)V^e_S + \sum_{u\in U}\sum_{e\in E} S_{u,p}(t)\beta^e_{u,p}(t)V^e_C)]/n_{users}$$

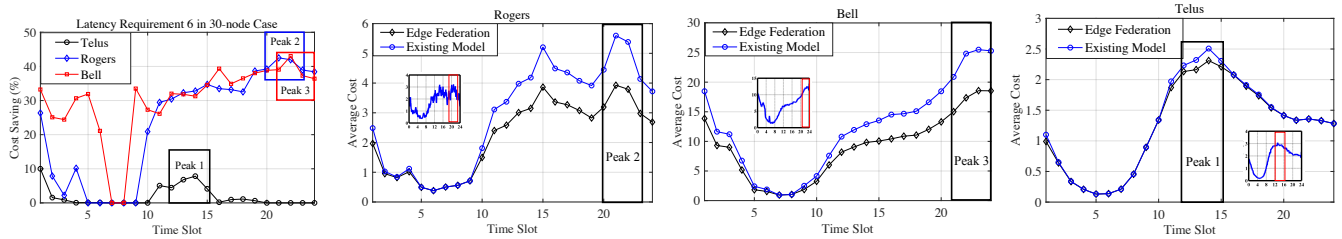
where n_{users} represents the number of users.

1) The Overall Performance Comparison

Fig. 6(a) illustrates the overall cost of the 50-node and 30-node cases in 24-time-slot dynamic traffic, with various latency requirements. The latency requirements of services are set from loose to strict, as shown in Table.II.

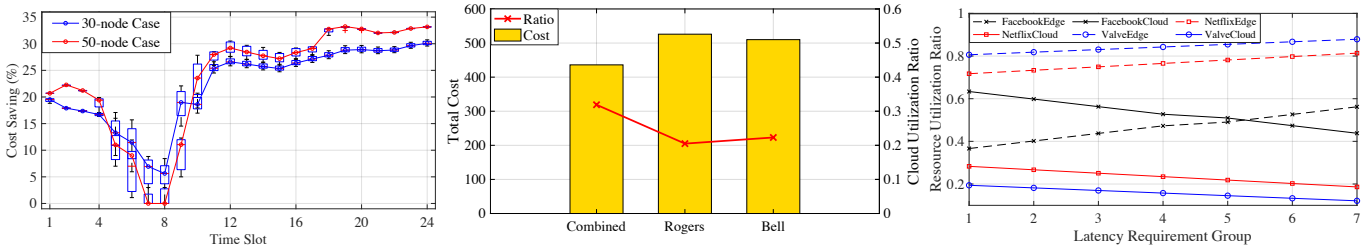
First, we can observe that, with the requirements loose to strict, the overall cost grows from low to high. This may due to the higher latency requirements (a.k.a., lower time delay) cause more massive edge resource utilization. The edge resource has a higher cost per unit than the cloud resource. Thus, the overall cost is growing. Then, from Fig. 6, we can find that the performance of edge federation is better than the existing service provisioning model, achieving the average cost saving of 32.4% and 30.5% in the 50-node case and 30-node case, respectively. This means, compared with the fixed contracted mode, edge federation model is more cost-efficiency for EIPs. Such saving will be significant for EIPs, especially that with a large number of edge nodes in the extensive coverage area (e.g., over a city or a country). There is also an interesting phenomenon that the total cost of 50-node is lower than the cost in the 30-node case in each latency requirement group. This is easy to be understood: compared with the 30-node case, when EIPs try to deploy services around the end users, the 50-node one has more and better (i.e., shorter distance to the end user) options in a specific geo-distance area for the EIPs to choose. Thus, EIPs avoid the remote service deployment, and the cost of service delivery could be saving.

Does Service Type Matter?: To investigate whether or not the type of service has a significant impact on the cost saving, we analyze the performances of each EIP individually under the constraints of varying latency requirements from latency requirement group 1 to 7. The corresponding results are shown in Fig. 6(b), where the range of cost savings for each EIP are demonstrated. We find that Telus, Rogers, Bell have savings around 3.7% and 3.0%, 33.6% and 26.0%, 34.8% and 38.1%



(a) The cost saving varation of three EIPs hourly in 30-node case. (b) The average cost varation of Rogers. (c) The average cost varation of Bell. (d) The average cost varation of Telus.

Fig. 7. The variation of the average service provisioning cost in three EIPs, under the fixed contract model and the edge federation model.



(a) The variance of the EIPs' hourly cost saving in the 30-node case and the 50-node case, over seven different latency requirements.

(b) The overall service provisioning cost of the Netflix and the cloud resource utilization ratio in different scenarios. Note that the total amount of allocated resource in Combined EIP, Rogers and Bell is equal.

(c) The variance of the edge and cloud resource utilization ratio in different SPs, under different latency requirement groups.

Fig. 8. Comparsion of the service provisioning cost and the reource utilization ratio.

in 30-node and 50-node case, respectively. The results indicate that the edge federation is advantageous for all kinds of EIPs, irrespective of the service they contracted and the number of nodes they have.

Dig deeper, we can find that the EIP contracted with the higher-latency-requirement service will receive even greater cost saving.[4] The reason for this result may be due to the fact that higher-latency-requirement services will use more edge resources. However, the highly distributed edge resources in an individual EIP makes it difficult for EIPs deploying and provisioning service with an efficient way (i.e., due to the considerable accumulated distance between different edge nodes, the service provisioning process will cause significant service delivery cost). The global resource integration and optimization in edge federation can significantly reduce the delivery overhead, especially for the high-latency-requirement services.

2) Variance of the Cost Saving over Dynamic Traffic

In the previous part, we accumulate the total cost over the whole period by ignoring the cost variation of each time slots. In this part, given the fixed latency constraints on the services (i.e., group six in Table.II), we consider the cost variation under dynamic service demands. Fig. 7(b), 7(c) and 7(d) show the varying average cost of Rogers, Bell, and Telus respectively.

Combined with Fig. 4 (i.e., the inset plot in Fig. 7(b), 7(c) and 7(d)), we can find that the curves of the average cost has similar changing trend with the amount of their corresponding service demands. This is in line with our general understanding that the larger the number of completed service requests, the higher cost of the EIPs. From the figure, we can also clearly see that the average cost of edge federation is less than existing service provisioning model in all time slots, and achieve average 1.8%, 23.6% and 30.3% cost saving for Telus, Rogers, and Bell, respectively.

Does Amount of Service Demands Matter?*:* Besides the similar trend between the average cost and service demands, we also find another meaningful result. That is, the higher cost saving is likely to appear at the situation with larger service demands. The cost-saving ranges of Telus, Rogers, and Bell are illustrated in Fig. 7(a). Fig. 7(a) and Fig. 4 together suggest that the cost saving and service demands have a strong correlation. As the amount of service demands grows up and down, there is a similar change in the amount of cost savings. To further explore this phenomenon, we use a 4-time-slot window to circle the peak cost of each service in Fig. 7. *Peak 1*, *Peak 2* and *Peak 3* represent the peak cost of Telus, Rogers, Bell, respectively. It is clear that the time windows of peak cost saving are perfectly matched with the peak service demands in each service. This means that the edge federation achieves even better performance in the case of the larger amount of service demands. It could be significantly helpful in the practical huge-traffic network environment.

3) Strength of Edge Federation

Resilient and Robust Service Provisioning*: Can edge federation achieve good performance all the time under varying requirements and dynamic service demands?* The question is critical to justify whether the edge federation can be reliable to the real network environment.

To answer this question, we mainly analysis the perfor-

[4]In this paper, for simplicity, we assume that the latency requirement of the Valve is more strict than the Netflix, and the Netflix is more strict than the Facebook.

mance from both the time dimension and the latency requirement dimension. From the view of time, compared with the existing solution, we can observe that the cost is reduced all the time, which means the performance is firm no matter how much service demands required. As for the varying requirements, the edge federation shows a steady saving behavior with slight fluctuation over different latency requirements. There is a very interesting phenomenon: the cost saving has the relative big fluctuations from the time slot 5 to the time slot 10, whereas the fluctuations in the other time slots are small. From Fig. 4, we can find that the traffic from the time slot 5 to the time slot 10 is much lower than the other time slots. Combined the fluctuation mentioned above, we can conclude that, compared with the low traffic situation, edge federation will achieve a more stable performance in the massive traffic scenario. This result once again proves that edge federation is suitable for the real huge-traffic environment.

Cost Efficiency Function with Horizontal Extending Edge Nodes: Edge federation enables the horizontal extension by integrating the edge nodes of EIPs. *Is this extending edge nodes function can reduce the cost of EIPs?* For validating the effectiveness of edge federation, we specially select two EIPs: Rogers and Bell. As shown in the map of the 50-node case in Fig. 5, Bell has better coverage in the West of Toronto while weak in the East. Rogers has a relatively balanced edge node locations. Then, we assume a virtual EIP owns all the edge nodes of Rogers and Bell (labeled as Combined EIP in Fig. 8(b)). Moreover, for fairness, we set all three EIPs to have the same amount of overall resource (i.e., same storage and computation capacity).

From Fig. 8(b) we can see that the extending edge nodes indeed improve the service provisioning performance. The Combined EIP achieve the least cost among the three EIPs, under the same service deploying. The red curve further illustrates this result with a cloud resource utlization ratio. The cloud utilization ratio of the Combined EIP is the highest, which means that the optimal provisioning schedule could be more efficient in the edge nodes extending scenario. Thus, more cloud resources will be utilized, and the overall cost will be reduced.

Adaptive Vertical Resource Allocation: To test the effectiveness of the dynamic service provisioning algorithm in edge federation, we calculate the resource utilization ratio of the services with seven different latency requirements (i.e., the latency requirement groups in Table.II). The results are shown in Fig. 8(c). We can see that, when the requirement becomes more and more strict, the edge resource utilization ratio of all the services are increasing. This indicates that, when facing the varying latency requirements, the algorithm truly realize the dynamic adjustment between edge and cloud resources.

The above all experimental results show that edge federation indeed solves the difficulties and challenges presented in Sec.II. It performs particularly effective under the heavy load and strict latency requirements, which fully match the needs of the latency-critical and resource-intensive smart services and show the value of our model in the real network environment.

VII. Discussion and Future Works

A. Determining the Optimal Controlling Scale

Rather than solving problems in the specific scenario, the edge federation is a general resource management model for the macro edge-computing scenario. The edge federation is operated in a centralized control manner, which could enable the most cost-efficiency service management for EIPs and provide the guaranteed QoS and QoE for the ESP and the end user, respectively.

One of the critical issue for the centralized management is the scale of the controlling area, which greatly determined by the factors in geography (e.g., different time zones may affect the prediction accuracy, different users in different areas may have different behavior patterns.), business environment (e.g., unique business policies in different countries and regions.), etc.. According to these factors, the centralized control in a city, a country or a strongly related region (e.g., the area of EU countries) can be more effective and robust. Traffic behaviors of these areas are more predictable and amenable to provide a mathematically well-grounded sizing solution.

B. Determining the Length of the Time Slot

The performance of the edge federation solution could be affected by the length of the time slot. In a practical scenario, this length should be determined by the multi-dimensional analysis of the real network environment, such as the general service completed time, end users' network behavior, network service traffic, etc.. A suitable length of the time slot can help users avoid the dynamical service migrate across servers, thus reducing the additional delays and costs. In this paper, we use traffic prediction technology as an important reference to determine the length of the time slot. We also leave it as a direction of future work.

C. Designing the Suitable Algorithm

The networking environment in this paper is quite complicated. We well formulated the optimization problem in the edge federation by mainly considering 1) resource factors (e.g., the heterogeneous resources of communication, storage, and computation); 2) geo factors (e.g., distributed edge nodes and users); 3) traffic factors (e.g., heterogeneous services, varying service demands, different latency requirements). Then, what we should do is finding the optimal analytical solution by solving the optimization problem. The primary purpose of this paper is to prove that edge federation is more cost-efficiency than the existing solution. Also, from other perspectives, one can design new algorithms or still exploit the advantages of the optimization techniques to solve problems (e.g., latency minimization, etc.) in edge federation. We leave this point as an open issue.

VIII. Related work

The related works can be roughly divided into two categories, including the service placement method and the service provisioning model.

Service placement is a popular topic in mobile edge computing (MEC), which involves the content caching and the computation offloading. The content caching has been studied extensively to place a large volume of content based on the popularity, to avoid frequent replication and enable a faster access [19]. The multimedia service is a representative field of the content caching. Many efforts have been made on the collaborative multi-bitrate video caching and processing in MEC network [20], [21]. The emerging concept of *In-Network Caching* has been proposed recently [22]. The basic idea is that servers tend to cache some content passing through them, according to the contents' popularity. Thus, each server may send the queried content directly to the end users with the small round-trip-time (RTT).

The computation offloading mainly focus on designing dedicated strategies for offloading partial even the entire task from an end device to edge servers. The major factors influence the offloading strategies including the characteristics of end devices and edge servers, such as the location [23], energy [24]), and different optimization objectives, such as minimizing the cost [7] or delay [25]. Liu et al. proposed a searching algorithm to find the optimal task scheduling policy to achieve the minimum average delay [25]. Mao et al. developed a LODCO algorithm to minimize the execution delay and addressed the task failure as the performance metric [26]. There are also some literature jointly consider the caching and offloading for maximizing the revenue of mobile network operator [27]. Different from these works mentioned above, our work considers the more general multi-EIP scenario.

Although the service provisioning is a crucial issue for edge computing, there still lack sufficient studies. The most involved literature focus on the integration between cloud and edge. Tong et al. designed a hierarchical edge cloud architecture to alleviate the peak workload from end users [28]. To minimize the cost of resources, Ma et al. proposed a cloud-assisted framework in MEC, named *CAME* [7], by combing the queueing network and convex optimization theories. Villari et al. also proposed the similar architecture call *Osmotic Computing*, which aims to decompose the applications into microservices and enhance seamless cooperation between cloud and edge resources [29]. It is true that literature [30] considered the cooperation between cells in a cellular network and [31] even studied the D2D collaboration among edge devices. However, there still lack many literature study the cooperation among edge servers, as pointed out in this paper.

Such a dilemma has already attracted considerable attention from industries, several organizations (e.g., OpenFog[5], EdgeComputingConsortium[6]) have been formed trying to find the effective network architecture and service provisioning model. To our best knowledge, this is the first step to consider the service provisioning model from the entire edge-computing environment. Our edge federation considers the service-provisioning problem among multiple EIPs and cloud with hard latency constraints.

[5]https://www.openfogconsortium.org/

[6]http://en.ecconsortium.org/

IX. Conclusion

In this paper, we proposed an integrated service provisioning model, named as *edge federation*, which considers a two-dimension integration between multiple EIPs, including the vertical and the horizontal. Over the edge federation, we formulated the provisioning process as an LP problem and took a variable dimension shrinking method to solve the large-scale optimization problem. Furthermore, for varying service demands, we proposed a dynamic service provisioning algorithm, *SEE*, which dynamically updates schedules to enable an efficient service deployment. Via the trace-driven experiments conducted on the real-world base station map of Toronto, we demonstrated that our edge federation model can help EIPs save the overall cost by 30.5% to 32.4%, compared with the existing model.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (No. 61802421, 61772544).

References

[1] "GSMA intelligence," https://www.gsmaintelligence.com/.
[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.
[3] "Sdxcentral," https://www.sdxcentral.com/mec/.
[4] H. Wang, P. Shi, and Y. Zhang, "Jointcloud: A cross-cloud cooperation architecture for integrated internet service customization," in *Proceedings of International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1846–1855.
[5] "Amazon Greengrass," https://aws.amazon.com/cn/greengrass/pricing/.
[6] "Google IoT Core," https://cloud.google.com/iot-core/.
[7] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin, and X. Shen, "Cost-efficient workload scheduling in cloud assisted mobile edge computing," in *Proceedings of International Symposium on Quality of Service (IWQoS)*. IEEE, 2017, pp. 1–10.
[8] "Microsoft hybrid cloud," https://azure.microsoft.com/en-us/overview/hybrid-cloud/.
[9] B. Yang, W. K. Chai, Z. Xu, K. V. Katsaros, and G. Pavlou, "Cost-efficient nfv-enabled mobile edge-cloud for low latency mobile applications," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 475–488, 2018.
[10] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using arima model and its impact on cloud applications' qos," *IEEE Transactions on Cloud Computing*, vol. 3, no. 4, pp. 449–458, 2015.
[11] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, "Time-series extreme event forecasting with neural networks at uber," in *Proceedings of International Conference on Machine Learning*, no. 34, 2017, pp. 1–5.
[12] "The cost of the micro data center," https://www.schneider-electric.com/en/download/document/apc_vavr-99x6svk_en/.
[13] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," vol. 39, no. 1. ACM, 2008, pp. 68–73.
[14] "The Broadcasting and Telecommunications Regulation of Canada," http://sms-sgs.ic.gc.ca/eic/site/sms-sgs-prod.nsf/eng/h_00010.html.
[15] "The location of the Google data centers," https://www.google.com/about/datacenters/inside/locations/.
[16] O. Krajsa and L. Fojtova, "Rtt measurement and its dependence on the real geographical distance," in *Proceedings of International Conference on Telecommunications and Signal Processing (TSP)*. IEEE, 2011, pp. 231–234.
[17] "Toronto population," http://www.citypopulation.de/.
[18] "Toronto population density," https://www.thestar.com/business/.
[19] G. Dán and N. Carlsson, "Dynamic content allocation for cloud-assisted service of periodic workloads," in *Proceedings of International Conference on Computer Communications (INFOCOM)*. IEEE, 2014, pp. 853–861.

[20] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, "Collaborative multi-bitrate video caching and processing in mobile-edge computing networks," in *Proceedings of Conference on Wireless On-demand Network Systems and Services (WONS)*. IEEE, 2017, pp. 165–172.
[21] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, "Qoe-driven mobile edge caching placement for adaptive video streaming," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 965–984, 2018.
[22] J. Llorca, A. M. Tulino, K. Guan, J. Esteban, M. Varvello, N. Choi, and D. C. Kilper, "Dynamic in-network caching for energy efficient content delivery," in *Proceedings of International Conference on Computer Communications (INFOCOM)*. IEEE, 2013, pp. 245–249.
[23] C. Wang, Y. Li, and D. Jin, "Mobility-assisted opportunistic computation offloading," *IEEE Communications Letters*, vol. 18, no. 10, pp. 1779–1782, 2014.
[24] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? the bandwidth and energy costs of mobile cloud computing," in *Proceedings of International Conference on Computer Communications (INFOCOM)*. IEEE, 2013, pp. 1285–1293.
[25] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proceedings of International Symposium on Information Theory (ISIT)*. IEEE, 2016, pp. 1451–1455.
[26] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.
[27] Y. Zhou, F. R. Yu, J. Chen, and Y. Kuo, "Resource allocation for information-centric virtualized heterogeneous networks with in-network caching and mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 11 339–11 351, 2017.
[28] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proceedings of International Conference on Computer Communications (INFOCOM)*. IEEE, 2016, pp. 1–9.
[29] M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan, "Osmotic computing: A new paradigm for edge/cloud integration," *IEEE Cloud Computing*, vol. 3, no. 6, pp. 76–83, 2016.
[30] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Joint computation offloading, resource allocation and content caching in cellular networks with mobile edge computing," in *Proceedings of International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.
[31] X. Chen, L. Pu, L. Gao, W. Wu, and D. Wu, "Exploiting massive d2d collaboration for energy-efficient mobile edge computing," *IEEE Wireless Communications*, vol. 24, no. 4, pp. 64–71, 2017.

Xiaofeng Cao received the Bachelor's and Master's degrees from the National University of Defense Technology, China, in 2014 and 2016, respectively. He is pursing his Ph.D. degree with the Department of System Engineering, National University of Defense Technology, Changsha, China. He is currently a visiting student in BBCR Lab of University of Waterloo, Waterloo, Canada. His research interests include edge computing and network traffic analysis.

Guoming Tang (S'12-M'17) received the Bachelor's and Master's degrees from the National University of Defense Technology, China, in 2010 and 2012, respectively, and the Ph.D. degree in Computer Science from the University of Victoria, Canada, in 2017. He joined the College of Systems Engineering at the National University of Defense Technology in 2017 where he is currently an Assistant Professor. Aided by machine learning and optimization techniques, his research mainly focuses on computational sustainability issues and distributed networking systems.

Deke Guo received the B.S. degree in industry engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001, and the Ph.D. degree in management science and engineering from the National University of Defense Technology, Changsha, China, in 2008. He is currently a Professor with the College of System Engineering, National University of Defense Technology, and is also with the College of Intelligence and Computing, Tianjin University. His research interests include distributed systems, software-defined networking, data center networking, wireless and mobile systems, and interconnection networks. He is a senior member of the IEEE and a member of the ACM.

Yan Li received the Bachelor's degree from the National University of Defense Technology, China, in 2014. She is pursing her Ph.D. degree with the Department of System Engineering, National University of Defense Technology, Changsha, China. She is currently a visiting student in BBCR Lab of Mcgill University, Montreal, Canada. Her research interests include edge computing and network traffic analysis.

Weiming Zhang received the Ph.D. degree in management science and engineering from National University of Defense Technology, Changsha, China, in 2001. He is currently the chief of the Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, China. His current research interests include command and control organizations.