Dynamical Resource Allocation in Edge for Trustable Internet-of-Things Systems: A Reinforcement Learning Method

Shuiguang Deng, *Senior Member, IEEE*, Zhengzhe Xiang, Peng Zhao, Javid Taheri, *Member, IEEE*, Honghao Gao, *Senior Member, IEEE*, Jianwei Yin, and Albert Y. Zomaya, *Fellow, IEEE*

***Abstract*—Edge computing (EC) is now emerging as a key paradigm to handle the increasing Internet-of-Things (IoT) devices connected to the edge of the network. By using the services deployed on the service provisioning system which is made up of edge servers nearby, these IoT devices are enabled to fulfill complex tasks effectively. Nevertheless, it also brings challenges in trustworthiness management. The volatile environment will make it difficult to comply with the service-level agreement (SLA), which is an important index of trustworthiness declared by these IoT services. In this article, by denoting the trustworthiness gain with how well the SLA can comply, we first encode the state of the service provisioning system and the resource allocation scheme and model the adjustment of allocated resources for services as a Markov decision process (MDP). Based on these, we get a trained resource allocating policy with the help of the reinforcement learning (RL) method. The trained policy can always maximize the services' trustworthiness gain by generating appropriate resource allocation schemes dynamically according to the system states. By conducting a series of experiments on the YouTube request dataset, we show that the edge service provisioning system using our approach has 21.72% better performance at least compared to baselines.**

Manuscript received January 29, 2020; accepted February 15, 2020. Date of publication February 18, 2020; date of current version May 26, 2020. This research was supported in part by the National Key Research and Development Program of China under Grant 2017YFB1400600, in part by the National Science Foundation of China under Grant 61772461 and Grant 61825205, in part by the Natural Science Foundation of Zhejiang Province under Grant LR18F020003, and in part by the Key Research Innovation Project of Hangzhou under Grant 20182011A06. Paper no. TII-20-0416. *(Corresponding authors: Peng Zhao; Honghao Gao.)*

Shuiguang Deng is with the First Affiliated Hospital, Zhejiang University School of Medicine, Hangzhou 310003, China, and also with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China (e-mail: dengsg@zju.edu.cn).

Zhengzhe Xiang and Jianwei Yin are with the College of Computer Science, Zhejiang University, Hangzhou 310027, China (e-mail: xiangzhengzhe@zju.edu.cn; zjuyjw@cs.zju.edu.cn).

Peng Zhao is with the First Affiliated Hospital, Zhejiang University School of Medicine, Hangzhou 310003, China (e-mail: zhaop@zju.edu.cn).

Javid Taheri is with the Department of Computer Science, Karlstad University, 65188 Karlstad, Sweden (e-mail: javid.taheri@kau.se).

Honghao Gao is with the Computing Center, Shanghai University, Shanghai 200444, China (e-mail: gaohonghao@shu.edu.cn).

Albert Y. Zomaya is with the School of Information Technologies, The University of Sydney, Sydney, NSW 2006, Australia (e-mail: albert.zomaya@sydney.edu.au).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2020.2974875

***Index Terms*—Edge computing, Internet-of-Things (IoT), trust management, resource allocation.**

I. Introduction

We are now embracing an era of Internet-of-Things (IoT). The amount of IoT devices proliferated rapidly with the popularization of mobile phones, wearable devices, and various kinds of sensors: There are 8.6 billion IoT connections established in the end of 2018, and the number is predicted to increase to 22.3 billion according to Ericsson's.[1] To face the challenge of trustworthy connection and low latency in the future, researchers pay their attention to a novel computing paradigm called edge computing (EC) [1]. In contrast to cloud computing, EC refers to decentralized tasks at the edge of the network. In EC paradigm, plenty of edge servers are established close to IoT devices to deal with the requests from these devices before they are routed to the core network [2]. Therefore, the computation and transmission between devices and cloud server can be partly migrated to edge servers or cloud server, like.[2] It enables IoT devices to fulfill complex analysis tasks with lower latency, higher performance, and less energy consumption [3] by taking advantage of the services deployed on edges. What is more, we can even establish a standalone cluster where the edge servers can work co-operatively [4] to get full control and improve the offline scheduling capability at edge side, like.[3] Besides these, with the help of edge servers in proximity, applications are enabled to learn from the mobile users' real-time context information [5] to improve their quality of experience [6].

However, simply establishing such a service provisioning system is not enough [7]. Because there will be another significant problem—trust management that should be considered in this scenario [8]. Except for the typical issues like key escrow and application distribution [9], we should also be careful about the business agreement complying. A service-level agreement (SLA) is a commitment between a service provider and a client, it can contain numerous service-performance metrics with corresponding service-level objectives like turnaround time (TAT)

[1][Online]. Available: https://www.ericsson.com/49d1d9/assets/local/mobility-report/documents/2019/ericsson-mobility-report-june-2019.pdf

[2][Online]. Available: https://docs.microsoft.com/en-us/azure/iot-edge/about-iot-edge

[3][Online]. Available: https://docs.kubeedge.io/en/latest/modules/edgesite.html

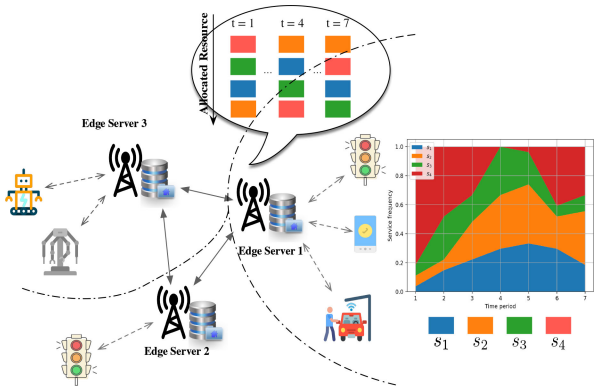

Fig. 1. Motivation scenario with multiple edge servers in smart city.

that describes the expected time to complete a certain task. Therefore, we can go a step further to find how to manage these services on edge servers so that the trustworthiness of them can be extremely improved. An effective but challenging way to do it is to keep the services running effectively by allocating appropriate resources to them. Therefore, we then need to explore the relationship between the service effectiveness and resource allocation scheme [10], [11]. And we need to find a policy to determine how to allocate resources for the services on edge servers. Particularly, the policy should be a dynamical one which can tell how to allocate resources in different time periods, because the requests produced by IoT devices may vary over time and we care about the trustworthiness all the time. These will be difficult because there is no existing model for the edge service provisioning system in IoT environment and most of the policies of existing works are static ones. Take the context of "Smart City" as an example. In this field, plenty of IoT sensors are distributed around the city to collect data [12]. Traditionally, the collected data will be uploaded to the cloud for analyzing. When decisions are made, some control data will be sent back to corresponding controllers for further actions. For example, Alibaba researchers use their City Brain [13] to help to reduce the traffic congestion: They collect the traffic information of different roads with webcams and upload to AliCloud, then they use the services on it to determine how to adjust the traffic lights. But things will be easier if we can take advantages of the EC paradigm: If we can divide the roadmap into several independent regions and use the edge servers in every region to establish a service provisioning system, we can then conduct the analyses locally and the traffic lights of this region can be adjusted in time.

However, there is not only one service in the smart city but another example is also shown in Fig. 1. In this case, there are three edge servers making up a mini service provisioning system to provide four services with different functions (we assume these services have the same parameters except the functionality for simplification) for the IoT devices. The four related smart city services are presented with four colored squares, where s_1–s_4 are [*TrafficLightService*, *NoiseService*, *AirQualityService*, *CriminalDetectService*] and these services all have declared their expected TATs. For edge server #1, the distribution of service requests over time is shown by a stacked area chart on right. As services can process faster with more resources, we should carefully allocate the limited resources: In time period $t = 1$, because most requests are about s_1, we can give s_1 more resource. Nevertheless, if we do not change the resource allocation scheme, the trustworthiness of the system will decay in $t = 4$. Because in this time period s_1 is not the most popular one, most requests are about s_2. Then the declared TAT may not be satisfied for these services so that the system will become not trustable. Therefore, we would better reallocate resources for a better result. Similarly, we would better reallocate resources in $t = 7$.

Though we have addressed the importance of dynamic resource allocation by a simple case, the relationship between the allocated resource and system performance may be more complex, and we should also consider the cooperation of edge servers and cloud. In summary, the contributions of this article are as follows.

1) We investigate the resource limitations of edge servers, resource consumptions of services, and consider the multiedge-cloud cooperation to model the trustable edge service provisioning.
2) We quantify the trustworthiness with the allocated resources for services so that we can explore how they can impact the provisioning.
3) We model the process of resource allocation with Markov decision process (MDP) and train the policy based on a reinforcement learning (RL)-based approach.
4) We conduct a series of experiments to evaluate the generated policy with the Youtube dataset, and compare it with some other approaches.

The rest of this article is organized as follows. Section II highlights the related work of EC and the corresponding approaches. Section III presents the system model. Section IV describes we have prepared to solve this problem like how we quantify the trustworthiness and how we model the entire process. Section V introduces the framework of the RL-based approach. Section VI shows the experimental results including the factors that affect our algorithms. Finally, Section VII concludes this article.

II. Related Work

With an increasing number of IoT devices connecting to the cloud, the demand for high-quality service becomes urgent. It drives more and more researchers to pay attention to issues of the EC paradigm which may affect the effectiveness of service provision. The resource allocation problem is one of the important issues. In this section, we will review the related works about how to improve it with appropriate resource allocation: By asking the question "*using what resource of what host at what time for what task ?*" (**Q1**), researchers find that an appropriate resource allocation scheme will contribute a lot in performance optimization.

For example, Li *et al.* [14] considered the energy consumption of IoT devices; they analyzed overheads of IoT devices and proposed an overhead-optimizing multidevice task scheduling strategy for ad hoc-based MEC systems. Stefania *et al.* [15] consider

a multi-input and multi-output (MIMO) multicell system where multiple mobile users ask for computation offloading to servers; they formulate the offloading problem as the joint optimization of the radio resources and the computational resources to satisfy the latency constraints. You *et al.* [16] considered incorporation with dense cellular networks; they proposed an online algorithm based on Lyapunov optimization which determined offloading and edge server sleeping policy to obtain good performance with low energy consumption. Zhang *et al.* [17] also considered the scenario in cellular networks, and they translated the resource allocation problem to replica deployment problem. They randomly distributed and stored the popular data contents in IoT devices and obtained high spectrum efficiency, and significantly reduced the duplicate data traffic in the core network and through base stations. In the system of Yang *et al.* [18], they mainly cared about the workload balance. By taking advantages of the mobility of users and service invocation pattern, they allocated resources for a service cache and used this cache in workload balancing. Chase *et al.* [19] considered the cost of virtual machines (VM) with high bandwidth requirements; they took advantage of software-defined network and VM technology to propose a novel allocation algorithm for minimizing the cost. Jia *et al.* [20] considered mobile service provision with cloudlets in a wireless metropolitan area network, in which the requests were imbalanced and bursty. Since cloudlets were resource limited, they proposed a scalable load-balancing algorithm to make full use of the cloudlet network resources. Tran *et al.* [21] considered the problem of joint task offloading; they studied the resource allocation in order to maximize the users' task offloading gains, which was measured by a weighted sum of reductions in task completion time and energy consumption.

These works show many approaches about performance optimization in EC paradigm. Based on these works, we go a step further by proposing a dynamic resource allocation policy to answer question **Q1** in our edge service provisioning system to ensure the SLA of involved services.

III. System Description and Model

In this trustable edge service provisioning system, there are mainly three types of entities. The first one is the IoT device: IoT devices can communicate with the edge servers by sending requests to and receiving responses from them if they are within the serving area of those edge servers; the second one is the service: services can help to fulfill some specific tasks; the last one is the host, a host is a machine where those services are deployed and executed. In this trustable service provisioning system, a host can be a cloud or an edge server. As the requests of IoT devices may be imbalanced and dynamic, and requested services may have different functionalities and processing capacities, here the edge servers are integrated into this system to make a geographically local cluster. The cluster will help to make full use of the edge resources, but some external complexity in trust management is also introduced. Therefore, we need to quantify the trustworthiness and explore the factors that may impact it, so that we can adjust the service resource allocation scheme in an appropriate way. In this section, we will give a brief description of the properties of these system entities and how they will interact with each other. Based on these, the resource allocation scheme is modeled. And as we mainly focus on the service average TAT of SLA in system trust management, it is formulated in this section to help to understand the definition of trustworthiness gain.

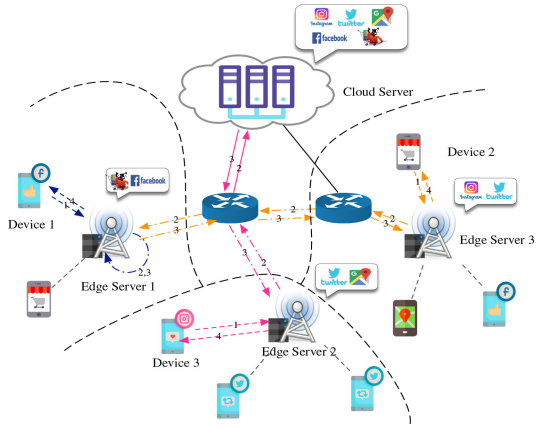

Fig. 2. Illustration of SRLC.

A. Average Turnaround Time Estimation

Suppose there is a cloud sever h_0 and n edge servers (h_1, ..., h_n) in this provisioning system for services s_0, s_1, ..., s_m (s_0 is a virtual service stands for "idle"). To explain the running of system clearly, here we introduce **service request life cycle (SRLC)**. A SRLC starts from its generation and ends with receiving the results by the invoker IoT device. Fig. 2 shows 3 typical SRLCs. In the blue one, the user with device #1 wants to request the content of Facebook. With the corresponding service on edge server #1, the request is easily handled by it; in the pink one, as the edge server #2 do not allocate resources to Instagram, it will dispatch the request from user with device #3 to the one which can handle it (the cloud server in this case); in the orange one, edges work together to provide services. An entire life cycle can be divided into four steps according to the state of request and hosts as following.

1) *Access step:* First, a request for service s_i is produced by the IoT device and the request will be sent to the nearest edge server h_j (called *access server* $\mathcal{H}_a$) via wireless link, then the time cost is

$$T_A = \frac{D_i^{in}}{v_u^j} \tag{1}$$

where D_i^{in} is the average input data size of s_i, and v_u^j is the average data transmission rate between h_j and the IoT devices in its serving area. It is worth noting that the principle of proximity is adopted here to help selecting appropriate edge servers. This is just one of the possible strategies in edge server selection, because the environment may become more complex in some extreme cases. The researchers will easily extend the model by using their own selection strategies.

2) *Routing step:* Second, access server h_j will select the appropriate server h_k to handle this request, and then send the request to it (called *executor server* $\mathcal{H}_e$), then the time cost is

$$T_R = \frac{D_i^{in}}{\mathcal{B}_{j,k}} \tag{2}$$

where $\mathcal{B}_{j,k}$ describes the topology and bandwidth between h_j and h_k. The elements of matrix $\mathcal{B}$ are nonnegative values, while $\mathcal{B}_{j,k}$ is set $+\infty$ if $j = k$, and is set 0 if h_k is not accessible for h_j. We can find that an appropriate routing policy is needed here to make decisions. Therefore, we use $p_{i,j,k}$ to denote the probability that the server h_j dispatches requests for service s_i to server h_k to describe routing policy. In reality, developers would like to use round-robin principle to balance the workload. With this principle, requests will be routed to different hosts in order, namely $p^t_{*,*,k} = \frac{1}{n+1}$. However, it implicitly assumes that the machines are homogeneous, and does not consider the system context. With consideration of different processing capacities of machines, we will give the host h_j a larger probability if it has better processing capacity for service s_i, like the weighted round-robin approach. And the weight here is the processing capacity. Namely, we can have $p_{i,j,k} = \frac{\mu_{i,k}}{\sum_{q=0}^{n} \mu_{i,q}}$.

3) *Execution step:* Then, as the executor server h_k has received the request, it will use the corresponding service to fulfill the task. Supposing the processing capacity (e.g., the ability to handle instructions measured by MIPS) for service s_i on host h_k is $\mu_{i,k}$, and the workload (e.g., the instruction number to run the program) of s_i is w_i, the time cost is

$$T_E = \frac{w_i}{\mu_{i,k}} \tag{3}$$

where we assume $\mu_{i,k}$ is proportional to $l_{i,k}$, the resource, e.g., CPU or memory, allocated to s_i on h_k like [22] (A more sophisticated model is shown in [23], here we can find that the linear assumption will be correct except for the extreme conditions). When $\mu^\star_k$ is the maximum processing capacity of h_k and the resource limitation of h_k is L_k, we have $\mu_{i,k} = \frac{l_{i,k}}{L_k}\mu^\star_k$ according to (13) in [22].

4) *Backhaul step:* Finally, the results will first go to the access server and finally go back to the IoT device to finish the whole life cycle, then the time cost is

$$T_B = \frac{D_i^{\text{out}}}{\mathcal{B}_{k,j}} + \frac{D_i^{\text{out}}}{v_u^j} \tag{4}$$

where D_i^{out} is the average output data size of s_i. In this way, when i, j, k are determined, so will SRLC ϕ. Then TAT for ϕ can be represented by

$$T_\phi = T_A + T_R + T_E + T_B. \tag{5}$$

With requests generated any time from IoT devices, the service provisioning system will repeat the above operations.

B. Dynamic Service Resource Allocation

The resource allocation scheme $\boldsymbol{P}$ can be represented with a matrix where the element $P_{j,i}$ means the resource quota (in %) for s_i on h_j. Without loss of generality, here we mainly focus on the computation resource like CPU and memory, because this kind of resource is more rare and expensive than the storage resource, and the followers can easily expand our model with some other pluralistic functions that consider more resources. Therefore, the allocated resource for service s_i on h_j can be calculated by $l_{i,j} = P_{j,i}L_j$. As mentioned in Section I, a significant task in the trustable service provisioning system is to ensure the SLA of services. But as the requests are time-varying, a fixed service resource allocation scheme cannot work well all the time. Therefore, it will be effective for the system to replan the resource allocation schemes in runtime. In our work, the allocation replanning for the resource on hosts is denoted with the matrix $\boldsymbol{A}^t \in \mathbb{R}^{(n+1)\times(n+1)}$, which describes how the service allocation scheme will be replanned at time period t. Then, the service resource allocation scheme of the next time period can be produced by

$$\boldsymbol{P}^{t+1} = \boldsymbol{A}^t \times \boldsymbol{P}^t \tag{6}$$

here we also have $\boldsymbol{P}^{t+1} \in \mathbb{R}^{(n+1)\times(m+1)}$. What is more, as $P_{j,i}$ is represented in percentage, we will have $\sum_{i=0}^{m} P_{j,i} = 1$ and $0 \le P_{j,i} \le 1$. This is a constraint that can be used in training of the RL-based approach [10] —for the neural network that generates $\boldsymbol{P}$, a $softmax$ layer will be added to ensure this constraint.

IV. Algorithm Design

In this trustable IoT service provisioning system, as we mainly care about the how SLA is complied with, especially the TAT of SLA. Thus, before developing of the **d**ynamic s**e**rvice **r**esource **a**llocation for **d**istributed **e**dges algorithm (DeraDE), we should clarify the expected TAT of services, and mine the relation between the system trustworthiness and it.

By denoting the request number of service s_i observed on h_j (or frequency) with $\omega^t_{i,j}$, the probability that h_j dispatches s_i to h_k with $p^t_{i,j,k}$, then the probability to get request path $\phi = (i, j, k)$ is

$$\begin{aligned} Pr\{\phi = (i,j,k)\} &= Pr\{s = s_i, \mathcal{H}_a = h_j, \mathcal{H}_e = h_k\} \\ &= Pr\{\mathcal{H}_e = h_k | s = s_i, \mathcal{H}_a = h_j\} \\ &\quad \times Pr\{s = s_i | \mathcal{H}_a = h_j\} Pr\{\mathcal{H}_a = h_j\} \\ &= p^t_{i,j,k} \cdot \frac{\omega^t_{i,j}}{\sum_{i=0}^{m} \omega^t_{i,j}} \cdot \frac{\sum_{i=0}^{m} \omega^t_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n} \omega^t_{i,j}}. \end{aligned} \tag{7}$$

As introduced in Section III-A, we can get $p^t_{i,j,k} = \mu^\star_k P^t_{k,i} / \sum_{q=0}^{n} \mu^\star_q P^t_{q,i}$ with the definition of $P_{j,i}$. Denote the total size of s_i's input and output with D_i ($D_i = D_i^{in} + D_i^{\text{out}}$) in time period t, the expected TAT $\mathbb{E}[T]$ of this time period can then be

represented by

$$\begin{aligned}\mathbb{E}[T] &= \sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{k=0}^{n} Pr\left\{\phi=(i,j,k)\right\}T_{\phi} \\ &= \frac{1}{\sum_{i=0}^{m}\sum_{j=0}^{n}\omega_{i,j}^{t}}\sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{k=0}^{n}\frac{\mu_k^{\star}\omega_{i,j}^{t}P_{k,i}^{t}}{\sum_{q=0}^{n}\mu_q^{\star}P_{q,i}^{t}} \\ &\quad\cdot\left(\frac{D_i}{v_u^j}+\frac{w_i}{P_{k,i}^{t}\mu_k^{\star}}+\frac{D_i^{in}}{\mathcal{B}_{j,k}}+\frac{D_i^{\text{out}}}{\mathcal{B}_{k,j}}\right). \end{aligned} \tag{8}$$

However, as there are more than one time periods in service provision, we should consider them synthetically. As mentioned in Section III-B, the resource allocation scheme of the next time period is determined by that of the previous time period, we can use MDP to describe this process.

MDP is a stochastic control process described with a 4-tuple (X, Y, P, R) where X is the state space, Y is the action space, $P_{y^t}(x^t, x^{t+1})$ is the probability that action y^t in state x^t will lead to x^{t+1}, and R_t is the immediate reward for applying action y^t when state is x^t. Besides these, we can use the policy $\boldsymbol{\pi}$ to describe the distribution for actions for different states

$$\pi\left(y^{'}\middle|x^{'}\right)=Pr\left[\boldsymbol{y}=y^{'}\middle|\boldsymbol{x}=x^{'}\right]. \tag{9}$$

In this way, the MDP provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision-maker. In our model, we denote $x^t = \boldsymbol{P}^t$ as the state of the provisioning system at the beginning of time period t. With observation x^t, the provisioning system strategically decides an action $y^t = \boldsymbol{A}^t$ following a resource replanning policy $\boldsymbol{\pi}$. What is more, because $\mathbb{E}^t[T]$ can be obtained at the end of this time period, the *Trustworthiness Gain* for applying action y^t for state x^t can be denoted with

$$R_t = T^{\star} - \mathbb{E}^t[T] \tag{10}$$

where $T^{\star}$ is a threshold to make a smaller $\mathbb{E}[T]$ be rewarded more trustworthiness gain than a larger one. Therefore, given the time period t, if the current state $x^t = x$ we can then generate a sequence like $seq = [x^t, y^t, x^{t+1}, y^{t+1}, \ldots]$ according to $\boldsymbol{\pi}$. The accumulative trustworthiness gain for seq can be represented with

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \tag{11}$$

where $0 \le \gamma \le 1$ is called discount factor to show how the subsequent trustworthiness gain will contribute to G_t. As these sequences may be diverse, we can use the expectation of G_t to evaluate the value for state $x^t = x$ under a policy $\boldsymbol{\pi}$.

Therefore, we can evaluate the global trustworthiness gain for applying action $y^t = y$ at state $x^t = x$ with an action-value function

$$Q_{\boldsymbol{\pi}}(x,y) = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{k=0}^{\infty}G_t|x^t=x, y^t=y\right]. \tag{12}$$

Thus, given a behavior policy $\boldsymbol{\beta}$ where $\beta(y|x)$ describes the probability for applying action y in state x and ρ_β the probability density function of β, the expected global trustworthiness gain for policy $\boldsymbol{\pi}$ can be represented with

$$\begin{aligned} J_\beta(\boldsymbol{\pi}) &= \int_{x\in X}\int_{y\in Y}\rho_\beta(x)Q_{\boldsymbol{\pi}}(x,y)dxdy \\ &= \mathbb{E}_{x\sim\rho_\beta, y\sim\beta}\left[Q_{\boldsymbol{\pi}}(x,y)\right]. \end{aligned} \tag{13}$$

The goal of our algorithm is clear—we need to maximize the global trustworthiness (which is equal to minimizing the average TAT), and it can be formulated with

$$\boldsymbol{\pi}^* = \arg\max_{\pi} \quad J_\beta(\boldsymbol{\pi}). \tag{14}$$

V. RL-Based Approach

In our model, as the resource allocation scheme $\boldsymbol{P}$ and replanning action $\boldsymbol{A}$ can be vectorized to vectors in $\mathbb{R}^{n^2+2n+1}$ and $\mathbb{R}^{nm+n+m+1}$ by concatenating their rows, we would prefer to represent the policy $\boldsymbol{\pi}$ with a determined function $f_\Pi : \mathbb{R}^{n^2+2n+1} \to \mathbb{R}^{nm+n+m+1}$ which generates actions for any given states. To meet this demand, we use a neural network Π with parameter $\boldsymbol{\theta}_\Pi$ to approximate this function

$$y^t = \Pi\left(x^t;\boldsymbol{\theta}_\Pi\right). \tag{15}$$

On the other hand, we can find that there is another item $Q_\pi(x,y)$ involved in (13). Similarly, we hope that it can also be represented with a function $f_Q : \mathbb{R}^{n^2+2n+1} \times \mathbb{R}^{nm+n+m+1} \to \mathbb{R}$. So we use another neural network Q whose parameters are denoted with θ_Q to approximate f_Q

$$Q_{\boldsymbol{\pi}}(x,y) = Q\left(x,y;\theta_Q\right). \tag{16}$$

Now we just need to clarify structures of Π and Q. To train them with adequate exploration, here we use an actor–critic structure [24] and Off-policy. The Off-policy means that the behavior policy $\boldsymbol{\beta}$ is different in actor module and critic module. In our work, we add a noise N^t from a random process in determining actions with Π in actor module to construct behavior policy β

$$y_\beta^t = \Pi\left(x^t;\boldsymbol{\theta}_\Pi\right) + N^t. \tag{17}$$

Thus, given state x^t at beginning of time period t, we can take the action y_β^t generated by β for (6). After replanning, the reward can be represented by R_t and the current state of the provisioning system is changed to x^{t+1}. Repeat this process, tuples like $(x^t, y_\beta^t, R_t, x^{t+1})$ will be stored in a replay memory buffer M for future training. Taking advantage of the following Bellman equation for $Q_{\boldsymbol{\pi}}(x^t, y^t)$derived from (12), we can use

$$\hat{Q}_{\boldsymbol{\pi}}\left(x^t,\Pi\left(x^t;\boldsymbol{\theta}_\Pi\right)\right) = \mathbb{E}\left[R_t + \gamma Q_{\boldsymbol{\pi}}\left(x^{t+1},\Pi\left(x^{t+1};\boldsymbol{\theta}_\Pi\right)\right)\right] \tag{18}$$

as the target of $Q_{\boldsymbol{\pi}}(x^t, \Pi(x^t;\boldsymbol{\theta}_\Pi))$. Then we can create a supervised learning task to train the network Q for data batches (batch size $= N$) sampled from M

$$L_Q = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{Q}_{\boldsymbol{\pi}}\left(x^t,\Pi\left(x^t;\boldsymbol{\theta}_\Pi\right)\right) - Q_{\boldsymbol{\pi}}^i\left(x^t,\Pi\left(x^t;\boldsymbol{\theta}_\Pi\right)\right)\right)^2 \tag{19}$$

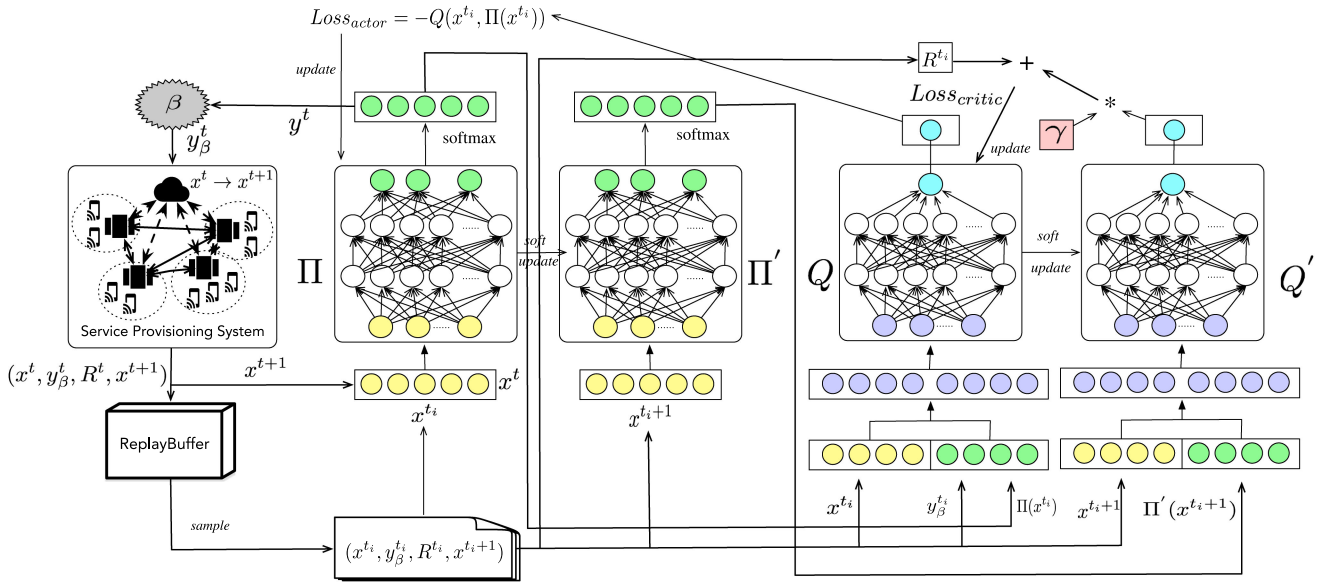

Fig. 3. Framework of the RL-based approach.

and the parameter for Q can be updated with

$$\theta_Q = \theta_Q - \eta_Q \nabla_{\theta_Q} L_Q. \tag{20}$$

On the other hand, the DeraDE is equivalent to searching for the best policy π that maximizes the accumulative reward. We can update θ_Π with $\nabla_{\theta_\Pi} J_\beta(\theta_\Pi)$, which can be computed by

$$\theta_\Pi = \theta_\Pi - \eta_\Pi \nabla_{\theta_\Pi} J_\beta(\theta_\Pi) \tag{21}$$

in which the gradient $\nabla_{\theta_\Pi} J_\beta(\theta_\Pi)$ can be approximated [25] with

$$\nabla_{\theta_\Pi} J_\beta(\theta_\Pi) \approx E_{x\sim\rho_\beta}\left[\nabla_{\theta_\Pi}\Pi(x;\theta_\Pi)\cdot\nabla_y Q(x,y;\theta_Q)\,|_{y=\Pi(x;\theta_\Pi)}\right]. \tag{22}$$

Simply using networks Π and Q will result in an unstable training process, because the parameters of Q will be updated frequently while they are involved in the gradients of both Q and Π, so we introduce target networks Π' and Q', which have the same structure and initial parameters with Π and Q but are updated softly with

$$\begin{aligned}\theta_{\Pi'} &= \tau\theta_\Pi + (1-\tau)\theta_{\Pi'}\\ \theta_{Q'} &= \tau\theta_Q + (1-\tau)\theta_{Q'}\end{aligned} \tag{23}$$

the framework of this process is shown in Fig. 3, and the details of the approach is shown in Algorithm 1.

Algorithm 1: DeraDE Algorithm.

Input: V: the batch size;
θ_Q^0: the initial parameters of Q;
θ_Π^0: the initial parameters of Π;
Output: Q: the action-value network;
Π: the action network;

for each episode **do**
  initialize the service provisioning system;
  for each time period t **do**
    select an action y^t with (17);
    get new state x^{t+1} with (6);
    get trustworthiness gain R_t with (10);
    store tuple (x^t, y^t, R_t, x^{t+1}) in buffer M;
    sample V tuples $\{(x^{n_i}, y^{n_i}, R_{n_i}, x^{n_i+1}) \mid 1 \le i \le V\} \subset M$ to compute $\nabla_{\theta_Q} L_Q$ and update Q;
    compute $\nabla_{\theta_\pi} J_\beta$ with (22) to update Π;
    softly update Q' and Π' with (23);
  end for
end for

VI. Experiments and Analysis

A. Preliminary

In this work, as there are training tasks involved, we should carefully choose representative datasets. The most important data is the requests of IoT devices, which describes when and where they use the services. Therefore, we turn to a network trace dataset of real-world and use it to simulate our service request. This dataset is a collection of traces from a campus network measurement on YouTube traffic in the University of Massachusetts [26]. The data covers a measurement period between June 2007 and March 2008 in the form of rows which are formatted with {Timestamp, YouTube server IP, Client IP, Request, Video ID, Content server IP}. It records when users visit YouTube.com for which video, and records how YouTube.com serve them as shown in Table I.

By classifying the videos to K classes as K different services with K–means or other clustering algorithms [27] and regarding the clients in the same subnet as the devices in the same area, we can easily reconstruct these access records as invocation records for these K services in the EC environment. At the same time,

TABLE I
EXAMPLE OF RECORD IN YOUTUBE TRAFFIC DATASET

Timestamp	YouTube Server IP	Client IP	Request	Video ID	Content Server IP
1189828805.208862	63.22.65.73	140.8.48.66	GET VIDEO	lML9dQN	158.12.125.12
1189909782.132296	63.22.65.77	102.15.54.63	LLNWD	iPrGtgmc	63.52.62.62
1189910238.327481	63.22.65.77	35.13.180.16	LLNWD	XS6QIs	218.11.53.57

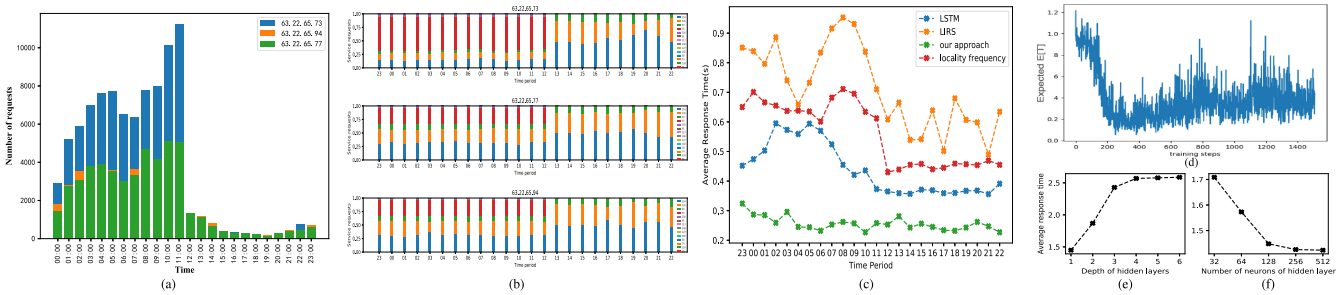

Fig. 4. (a) Running state of edge servers. (b) Service requests distribution. (c) Results of different approaches. (e) Training process of the approach (f) Impact of network structure.

due to the insensitivity and the lack of well adopted platforms and datasets, we generate the configurations of services and servers in a synthetic way. To make the data as close to reality as possible, we refer to the parameters of images on [4] for simulation data generation. Fig. 4(a) shows the service requests on different servers, we can find that the workload on different servers are quite imbalanced. If the resource of server 63.22.65.94 can be used to process service requests, as what we have modeled the cooperation of servers in this work, the performance will be dramatically improved.

B. Experiments and Analysis

1) Baselines Introduction: Traditionally, there are several approaches to solve this resource allocation problem. They mainly take advantage of the locality of service requests or try to get accurate request frequency of the next time period. Therefore, we choose the following representative algorithms as baselines:

1) *Locality frequency algorithm (LF):* This algorithm combines the ideas of least recently used algorithm and stable frequency assumption, which counts the frequency of service requests in different time periods and assumes the frequencies of requests for different services on different servers of the next time period are approximately equal to the former ones. Therefore, in this model we have

$$lf_{i,j}^{t+1} \approx \frac{\omega_{i,j}^{t}}{\sum_{p=0}^{m}\sum_{p=0}^{n}\omega_{q,p}^{t}} \tag{24}$$

2) *Low inter-reference recency set algorithm (LIRS) [28]:* In the LIRS algorithm, it utilizes the *reuse distance* of a page, which is the number of distinct pages accessed between two consecutive references of the page to quantify locality [28]. It maintains a hot set and a cold set to replace services in cache dynamically.

3) *Long shortterm memory algorithm (LSTM):* LSTM was first introduced for solving long-term dependency problems [29]. It has a form of a chain of repeating modules of neural networks, and it has the ability to remove or add information to the cell state, carefully regulated by structures called gates. By using the knowledge controlled by these gates, it works tremendously well on a large variety of problems, and are now widely used in time series analysis tasks like signal prediction, stock prediction, or even traffic prediction [30].

2) Performance Comparison: In this section, we will compare our approach with the baselines. As introduced in Section VI-B1, with the request data of the former N - 1 days in consecutive days as training data and that of the last day in the consecutive days as test data (e.g., we will use the data of March 13, 2008–March 17, 2008 as training data, and the data of March 18, 2008 as test data), we first use the LF to get the local service request frequency and allocate resources to services with this frequency percentage; and second, we use the LIRS to schedule the priority to be deployed of services of different time periods (1 h = 1 time period) and allocate more resource to services with higher priorities; third, we use the LSTM to predict the future service request frequency and allocate resource to services with their frequency percentage as what we do with the LF; finally, we use the trained dynamic service resource allocation policy with our approach to allocate resources.

By denoting $\text{imprv}^t(X)$ as the improvement of our approach compared with baseline X (X $\in$ {LF, LIRS, LSTM}) in time period t

$$\text{imprv}^t(X) = \frac{\mathbb{E}[T]_X^t - \mathbb{E}[T]_{\text{our}}^t}{\mathbb{E}[T]_X^t} \tag{25}$$

then we have $\text{imprv}^{\min}(X) = \min_{t=0\to T}\text{imprv}^t(X)$ to describe the how much can our approach can improve at least

[4][Online]. Available: https://hub.docker.com/DockerHub

TABLE II
PERFORMANCE COMPARISON

Method	Our approach			
	$imprv^{min}$	$imprv^{max}$	$imprv^{med}$	$imprv^{avg}$
LIRS	49.38%	72.86%	63.33%	63.01%
LSTM	21.72%	59.29%	37.26%	39.86%
LF	35.99%	64.19%	51.84%	52.98%

in Table II, while $\text{imprv}^{\max}(X)$ shows the maximum improvement, $\text{imprv}^{\text{median}}(X)$ shows the median improvement, and $\text{imprv}^{\text{avg}}(X)$ shows the average improvement.

In this table, we can find that our approach performs better than the baselines. To go deep for analyzing the result, we can conduct a case study. In Fig. 4(b), we show the service popularities in different time periods in March 18, 2008. We can find that the most popular services are [254, 105, 63, 55, 250]—96% of the requests on the edge servers are about these services. On each edge server, the service request distribution keeps stable in short continuous time periods, except time period 12 and 13. Because in these time periods, service 55 loses its popularity while the requests of service 254 and 105 become active. Fig. 4(c) shows the comparison results of the aforementioned approaches. Affected by the invocation order, the LIRS may lose accuracy in estimating the service request frequency, so it performs worse than other approaches. The locality frequency algorithm performs better than the LIRS, and it can find the changing rule of service request distribution: the average response times result from this algorithm keep similar to each other in time periods 23–12 and time periods 13–22. However, because the frequency of time period $t+1$ is predicted with data in time period t, there will be a lag problem that may affect the accuracy. This lag problem may be partly eliminated in LSTM because it performs better in frequency prediction. However, as they all do not consider the workloads and processing capability of services and servers, they cannot give a better deployment scheme than our approach, which takes advantage of all these factors. The training process of ours is shown in Fig. 4(e). With the replanning service resource allocation scheme in every time period, the service provisioning system can work with good trustworthiness.

3) Impact Factors Exploration: As people may have their own systems in different scenarios (e.g., people may establish edge service provisioning system in their own environments), it will be necessary to explore what factors will impact the results of the resource allocation scheme generated by the approach. Therefore, in this section, we will investigate the results of the generated resource allocation scheme when changing the attributes of the system, and visualize the impacts of these attributes with line charts. What is more, we use another ten heat-maps for deep inspection on the right side of every line chart in Figs. 5–7, where every two near ones in the same row make a group to show the resource allocation scheme and the host properties individually. For every heat-map pair, each color block in the ith row and jth column of the left heat-map shows the resource allocated to s_j by h_i. And in the right heat-map, it shows the ratios of data throughput (dt), request flow (λ),

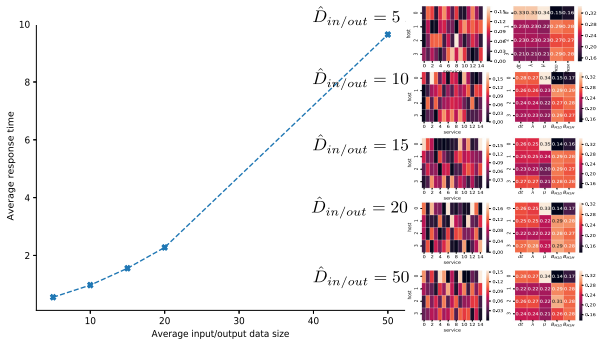

Fig. 5. Impact of data throughput.

processing capacity (μ), machine-to-device (M2D) communication quality (B_{M2D}) and machine-to-machine communication quality (B_{M2M}) of each host. Here, dt describes how much data a host can handle in every second, λ describes how many requests a host can handle in every second, μ describes how much workload a host can handle in every second, $B_{\text{M2D}} = v_u$ and $B_{\text{M2M}}[h_j] = \text{mean}_{k \neq j} B_{j,k}$ describe the communication qualities.

Neural network structures: As the structure of neural networks will affect the accuracy, it must be careful for the developers to select appropriate network parameters. In our work, as the neural networks are fully connected, we need to determine the number of hidden layers and the number of neurons in hidden layers.

Fig. 4(e) and (f) show the results with different hidden layer and neuron numbers. With the neuron number fixed (=128), we can find that the performance becomes worse with the increasing of hidden layer number N_{hidden}. This is because the increasing of N_{hidden} will make the network more complex and will cause more training errors in the fully connected neural networks [31]. On the other hand, the performance becomes better, and the improvement slows down when N_{neuron} is larger than 128 when increasing N_{neuron}. In this way, we choose this wide but not so deep neural network for approximation.

Configurations of the service provisioning system: As the service provisioning system receives the requests from IoT devices and sends the result of the service back to them, the input and output are the main data that flows in the system—not only in the communication of IoT devices and edge servers but also in the transmission between servers. Therefore, we will first explore how the data size will impact the result of the scheme. By changing the average data size of input and output $\hat{D}_{\text{in/out}}$ from 5 to 25, we can get the curve in Fig. 5. This figure shows that the average response time of services will increase with the increasing of $\hat{D}_{\text{in/out}}$, it is easy to follow because it will cost more in data transmission if there is more data in the input and output of the services. But things become interesting when we look at the heat-maps on the right side of the line chart. When $\hat{D}_{\text{in/out}}$ is not big ($\hat{D}_{\text{in/out}} = 5$), most of the requests will go to host h_0 or the cloud server even the communication qualities of h_0 are not good. This is because when the data size of input and output is small, the transmission time will not act as the key role that impacts the average response time, and because the cloud

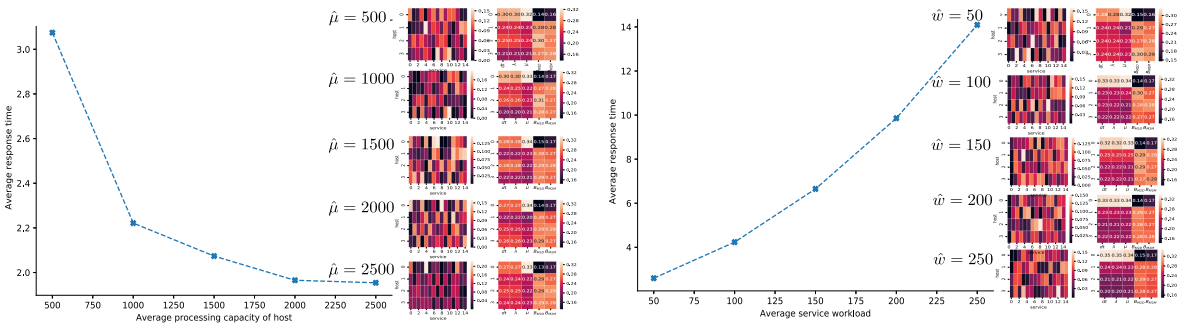

Fig. 6. (a) Impact of processing capacity. (b) Impact of service workload.

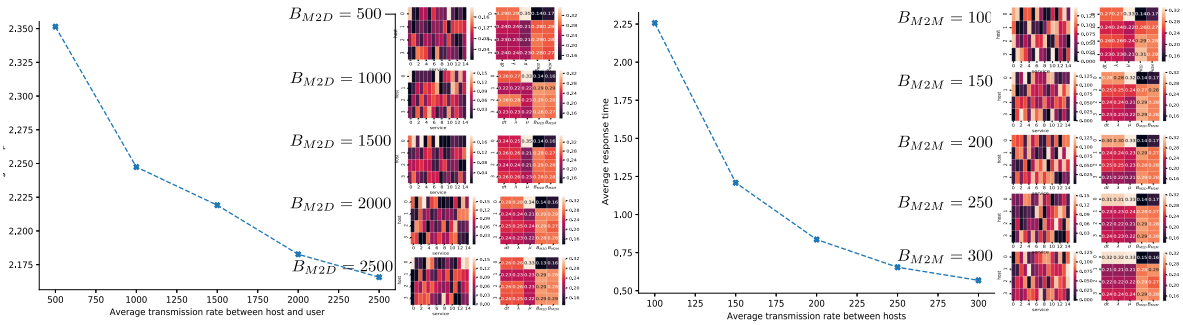

Fig. 7. (a) Impact of M2D transmission rate. (b) Impact of M2M transmission rate.

server has a better processing capacity than the edge servers, the execution time will be optimized if requests are routed to h_0. However, when $\hat{D}_{\text{in/out}}$ becomes bigger, the advantage of processing capacity will not help h_0 because the transmission cost starts to dominate the result. And we can also find that some of the resources in h_0 will become idle with the increasing of $\hat{D}_{\text{in/out}}$. The processing capacity and service workload determine the computation cost of service requests, because the former one describes how fast can a host to fulfill a task and the latter one describes how complex the task of service can be. By changing the average processing capacity of hosts ($\hat{\mu}$) from 500 to 2500, we can get the curve in Fig. 6. We can find that the bigger the $\hat{\mu}$ is, the better the performance will be. We can also find that with the increasing of $\hat{\mu}$, the data throughput and request flow reduce in h_0 while those of edge servers increase. This is because the value of $\frac{1}{\mu_{\text{edge}}} - \frac{1}{\mu_{\text{cloud}}}$ becomes small with the increasing of $\hat{\mu}$, so that the difference of execution cost on the cloud server and edge server becomes not important. On the other hand, by changing the average service workload ($\hat{w}$) from 50 to 250, we can find that the average response time increases. This is reasonable because when the task becomes more complex, it will cost more time in execution. Besides this, we can find the data throughput and request flow of h_0 becomes bigger. This is because the increasing workload makes edge servers overloaded, and they have to dispatch the requests to the cloud server. The conclusion of this result will be quite valuable, because it teaches the developers that they must not make the service too complex if they want to make full use of the resources on edges. If they have to fulfill complex tasks on edges, they would better keep them running with good processing capacities.

Quality of network communication: As mentioned in the former section, when service input and output data size becomes large, the transmission cost will become the main factor that may impact the performance. Therefore, we will explore two types of communications involved in this system. The first one is M2D communication—the IoT devices will connect to the edge servers via the wireless network. By changing B_{M2D} from 100 to 300, we can find that the average response time decreases. At the same time, the resource allocation schemes do not change much. This is because the request comes from and goes back to the IoT device, simply changing B_{M2D} will only impact the transmission time of M2D communication, and the M2D communication of different hosts and IoT devices are similar. However, things will be different if we focus on the value of B_{M2M}. By changing B_{M2M} from 100 to 300, we can find that the average response time decreases but the resource allocation schemes and properties of hosts vary with the changing. Though the cloud server still has the worst communication qualities, but the data throughput and request flow of h_0 increases with the decreasing of $\frac{1}{B_{\text{edge-edge}}} - \frac{1}{B_{\text{edge-cloud}}}$. This is because the cost of dispatching requests to cloud server decreases with the increasing of B_{M2M}—if the long-distance connection is not a problem anymore, the cloud server will absolutely be the best choice.

VII. Conclusion

In this article, we tried to improve the trustworthiness of services in IoT environment with EC paradigm from the perspective of how the SLA was complied with. We investigated the resource limitations of edge servers and resource consumptions of services, and explored the relationship between the allocated resources for service and their expected trustworthiness gain. Then, we highlighted the dynamical service resource allocation problem, and proposed a RL-based approach to determine the service resource allocation scheme in different time periods by modeling the resource allocation problem with a MDP. Finally, we conducted a series of experiments to compare this approach with other baselines and showed the factors that may affect the performance of the service provisioning system. However, as the modeling of the service provisioning system in this article is simple and idealized, there are still lots of issues that deserve further investigation in this problem. For example, we will consider the cyclical fluctuation and seasonal fluctuation in describing the state of the system, and we will try to use some more sophisticated service running model, and consider other useful SLA attributes. Besides these, because services can work together to make a composite service, the cooperation of services is also important in further research.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[2] H. Wu, S. Deng, W. Li, M. Fu, J. Yin, and A. Y. Zomaya, "Service selection for composition in mobile edge computing systems," in *Proc. IEEE Int. Conf. Web Services*, 2018, pp. 355–358.

[3] S. Deng, H. Wu, W. Tan, Z. Xiang, and Z. Wu, "Mobile service selection for composition: An energy consumption perspective," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 3, pp. 1478–1490, Jul. 2017.

[4] S. Deng, Z. Xiang, J. Yin, J. Taheri, and A. Y. Zomaya, "Composition-driven IoT service provisioning in distributed edges," *IEEE Access*, vol. 6, pp. 54258–54269, 2018.

[5] Z. Xiang, S. Deng, S. Liu, B. Cao, and J. Yin, "Camer: A context-aware mobile service recommendation system," in *Proc. IEEE Int. Conf. Web Services*, 2016, pp. 292–299.

[6] X. He, K. Wang, H. Huang, and B. Liu, "QoE-driven big data architecture for smart city," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 88–93, Feb. 2018.

[7] T. Wang, G. Zhang, A. Liu, M. Z. A. Bhuiyan, and Q. Jin, "A secure IoT service architecture with an efficient balance dynamics based on cloud and edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4831–4843, Jun. 2019.

[8] H. Tao *et al.*, "TrustData: Trustworthy and secured data collection for event detection in industrial cyber-physical system," *IEEE Trans. Ind. Informat.*, vol. 16, no. 5, pp. 3311–3321, May 2020.

[9] U. Jayasinghe, G. M. Lee, T.-W. Um, and Q. Shi, "Machine learning based trust computational model for IoT services," *IEEE Trans. Sustain. Comput.*, vol. 4, no. 1, pp. 39–52, Jan./Mar. 2018.

[10] Z. Xu, Y. Wang, J. Tang, J. Wang, and M. C. Gursoy, "A deep reinforcement learning based framework for power-efficient resource allocation in cloud rans," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.

[11] X. He, K. Wang, H. Huang, T. Miyazaki, Y. Wang, and S. Guo, "Green resource allocation based on deep reinforcement learning in content-centric IoT," *IEEE Trans. Emerg. Topics Comput.*, vol. PP, no. 99, 2018, doi: 10.1109/TETC.2018.2805718.

[12] L. Qi, M. Zhou, and W. Luan, "A two-level traffic light control strategy for preventing incident-based urban traffic congestion," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 13–24, Jan. 2018.

[13] X.-S. Hua, "The city brain: Towards real-time search for the real-world," in *Proc. 41st Int. Assoc. Comput. Machinery's Special Interest Group. Inf. Retrieval Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 1343–1344.

[14] L. Tianze, W. Muqing, Z. Min, and L. Wenxing, "An overhead-optimizing task scheduling strategy for ad-hoc based mobile edge computing," *IEEE Access*, vol. 5, pp. 5609–5622, 2017.

[15] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[16] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[17] X. Zhang and Q. Zhu, "Spectrum efficiency maximization using primal-dual adaptive algorithm for distributed mobile devices caching over edge computing networks," in *Proc. 51st Annu. Conf. Inf. Sci. Syst.*, 2017, pp. 1–6.

[18] L. Yang, J. Cao, G. Liang, and X. Han, "Cost aware service placement and load dispatching in mobile cloud systems," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1440–1452, May 2016.

[19] J. Chase and D. Niyato, "Joint optimization of resource provisioning in cloud computing," *IEEE Trans. Services Comput.*, vol. 10, no. 3, pp. 396–409, May/Jun. 2017.

[20] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.,* 2016, pp. 1–9.

[21] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[22] Y. Chen, C. Lin, J. Huang, X. Xiang, and X. S. Shen, "Energy efficient scheduling and management for large-scale services computing systems," *IEEE Trans. Services Comput.*, vol. 10, no. 2, pp. 217–230, Mar./Apr. 2015.

[23] S. R. Das and R. M. Fujimoto, "An empirical evaluation of performance-memory trade-offs in time warp," *IEEE Trans. Parallel Distrib. Syst.*, vol. 8, no. 2, pp. 210–224, Feb. 1997.

[24] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Advances Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.

[25] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Advances Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[26] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Watch global, cache local: YouTube network traffic at a campus network: Measurements and implications," in *Proc. SPIE - Int. Soc. Opt. Eng.*, vol. 6818, Jan. 2008, doi: 10.1117/12.774903.

[27] Y.-Y. Fanjiang, Y. Syu, S.-P. Ma, and J.-Y. Kuo, "An overview and classification of service description approaches in automated service composition research," *IEEE Trans. Services Comput.*, vol. 10, no. 2, pp. 176–189, Mar./Apr. 2017.

[28] W. Liu, F. Shi, and W. Du, "An LIRS-based replica replacement strategy for data-intensive applications," in *Proc. IEEE 10th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, 2011, pp. 1381–1386.

[29] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[30] J. Mackenzie, J. F. Roddick, and R. Zito, "An evaluation of HTM and LSTM for short-term arterial traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 5, pp. 1847–1857, May 2018.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

Shuiguang Deng (Senior Member, IEEE) received the B.S. and Ph.D. degrees, both in computer science, from the College of Computer Science and Technology, Zhejiang University, Hangzhou, China, in 2002 and 2007, respectively.

He worked with the Massachusetts Institute of Technology in 2014 and Stanford University in 2015 as a Visiting Scholar. He is currently a Full Professor of Computer Science with the College of Computer Science and Technology, Zhejiang University. Up till now, he has authored or coauthored more than 100 papers in journals and refereed conferences. His research interests include edge computing, service computing, mobile computing, and business process management.

Dr. Deng serves as the Associate Editor for the journal IEEE Access and *IET Cyber-Physical Systems: Theory & Applications.* In 2018, he was granted the Rising Star Award by IEEE TCSVC. He is a Fellow of IET.

Zhengzhe Xiang received the bachelor's degree in computer science and technology in 2013 from Zhejiang University, Hangzhou, China, where he is currently working toward the Ph.D. degree in computer science with the College of Computer Science.

His research interests include the fields of service computing, cloud computing, and edge computing.

Peng Zhao received the Ph.D. degree in oncology from Sun Yat-sen university, Guangzhou, China, in 2007.

He works with the Department of Medical Oncology, the First Affiliated Hospital, Zhejiang University School of Medicine, Hangzhou, China. He is currently leading some research projects supported by the National Natural Science Foundation of China. His research interests include artificial intelligence and edge computing in medical oncology.

Javid Taheri (Member, IEEE) received the bachelor's and master's degrees in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 1998 and 2000, respectively. He received the Ph.D. degree in the field of mobile computing from the School of Information Technologies at the University of Sydney, Sydney, NSW, Australia, in 2007.

Since 2006, he has been actively working in several fields, including: networking, optimization, parallel/distributed computing, and cloud computing. He is currently working as an Associate Professor with the Department of Computer Science, Karlstad University, Karlstad, Sweden. His major areas of interest include: profiling, modeling, and optimization techniques for private and public cloud infrastructures; profiling, modeling and optimization techniques for software defined networks; and network-aware scheduling algorithms for cloud and green computing.

Because of Dr. Taheri's contribution to the vibrant area of cloud computing, he was selected among 200 top young rehearses by the Heidelberg Forum in 2013. He also holds several cloud/networking-related industrial certification from VMware (VCP-DCV, VCP-DT, and VCP-Cloud), Cisco (CCNA), Microsoft, etc.

Honghao Gao (Senior Member, IEEE) received the Ph.D. degree in computer science, in 2012.

He started his academic career with Shanghai University in 2012. He is currently a Distinguished Professor in computer science with the Key Laboratory of Complex Systems Modeling and Simulation, Ministry of Education, China. His research interests include service computing, model checking-based software verification, wireless network, and intelligent medical image processing, Hangzhou, China.

Dr. Gao is an Institution of Engineering and Technology (IET) Fellow, British Computer Society (BCS) Fellow, European Alliance for Innovation (EAI) Fellow, China Computer Federation (CCF) Senior Member, and Chinese Association For Artificial Intelligence (CAAI) Senior Member.

Jianwei Yin received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2001.

He is currently a Full Professor with the College of Computer Science and Technology, Zhejiang University. He has authored or coauthored more than 120 research papers in major peer-reviewed international journals and conference proceedings in the areas of his research interests, which include cloud computing, performance evaluation, service computing, etc.

Dr. Yin is the Associate Editor for the IEEE TRANSACTIONS ON SERVICES COMPUTING.

Albert Y. Zomaya (Fellow, IEEE) received the Ph.D. degree in automatic control and systems engineering from Sheffield University, Sheffield, U.K., in 1990. He is currently the Chair Professor of High Performance Computing & Networking with the School of Computer Science, University of Sydney. He is also the Director of the Centre for Distributed and High Performance Computing. He authored or coauthored more than 600 scientific papers and articles and is author, coauthor, or editor of more than 25 books. His research interests include parallel and distributed computing, networking, and complex systems.

Prof. Zomaya served as Editor in Chief for the IEEE TRANSACTIONS ON COMPUTERS, from 2011 to 2014. He is the Founding Editor in Chief for the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING and the Editor in Chief for the *ACM Computing Surveys*. Currently, He serves as an Associate Editor for several leading journals. He delivered more than 190 keynote addresses, invited seminars, and media briefings and has been actively involved, in a variety of capacities, in the organization of more than 700 conferences. He is the recipient of the IEEE Technical Committee on Parallel Processing Outstanding Service Award in 2011, IEEE Technical Committee on Scalable Computing Medal for Excellence in Scalable Computing in 2011, IEEE Computer Society Technical Achievement Award in 2014, the ACM MSWIM Reginald A. Fessenden Award in 2017, and the New South Wales Premier's Prize of Excellence in Engineering and Information and Communications Technology in 2019. He is a Chartered Engineer, a Fellow of AAAS, IET (U.K.), an Elected Member of Academia Europaea, and an IEEE Computer Society's Golden Core Member.