This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2875753, IEEE Internet of Things Journal

Resource Provisioning in the Edge for IoT Applications with Multi-Level Services

Xu Zhang, Haojun Huang, Hao Yin, Dapeng Oliver Wu, *Fellow, IEEE*,
Geyong Min, Zhan Ma

***Abstract*— As the prevalence of computing-intensive and delay-sensitive IoT applications, IoT Service Providers begin to deploy micro data centers in the edge and offload functions to them. However, more and more complex IoT applications require an ordered sequence of services across geographically distributed infrastructure to fulfil their functions, which poses grand challenges for IoT Service Providers to deploy applications with low costs and high efficiency. To the best of our knowledge, no existing works have studied the deployment for an application with multi-level services (referred to as *ADMS* problem). To fill in the gap, we formulate the *ADMS* problem as an optimization problem with the aim of minimizing the overall deployment cost under the latency/computation/storage/bandwidth requirements and the infrastructure capacity limitations. We design a workflow-based heuristic algorithm called *AMS*, which can determine how many Virtual Machines (VMs) should be placed for each type of service and where to place them. *AMS* supports the services to scale up or scale down on demand in real time. Simulation experiments based on real network measurement demonstrate that *AMS* can reduce the number of deployed VMs by 28.4% and the deployment cost by 33.9% subject to comparable satisfied user ratio.**

I. Introduction

With the development of software and hardware technology, IoT devices are becoming more and more intelligent. These lightweight devices are widely used in many fields, such as smart homes and smart cities [25]. As predicted by Statista, there will be 8.4 billion IoT devices in use by 2020 [9]. Moreover, the global market for IoT is expanding dramatically and projected to reach $457B by 2020, with a Compound Annual Growth Rate (CAGR) of 28.5% [7]. However, with the prevalence of computing-intensive and delay-sensitive applications, such as augmented/virtual realities and spectral applications [13], existing IoT devices with limited storage and battery capacity cannot satisfy the requirements of these applications any longer [31]. To this end, more and more IoT Service Providers (SP) begin to deploy much more powerful servers, which is also called micro data centers, in the edge to assist the lightweight IoT devices [28]. Then the SP offloads the computing-intensive functions, which is hosted by specially-designed Virtual Machines (VMs), strategically among the micro data centers in the edge, while these VMs can provide services to multiple IoT devices.

The placement of these VMs is of great significance to the performance of applications and the operational cost of a SP. Note that an application may consist of several types of services and different types of services are queried in turn. For example, in order to support GPS navigations, a smart glass may send a query to an edge device such as a smartphone, and then the device fetches the navigation results from a cloud server. These applications requiring an ordered sequence of services are referred as to applications with muti-level services. One fundamental question is how to place the VMs for the application with multi-level services, or more precisely, given the performance requirement, how does a SP deploy the application among its infrastructure, aiming at minimizing the operational cost under the performance Service Level Agreement (SLA)? This problem is called Application Deployment with Multi-level Services (*ADMS* problem). Nevertheless, due to the inherently widely-distributed architecture of infrastructure, people are facing many pragmatic complexities in the application deployment:

Complexity of Infrastructure: nowadays, more and more SPs not only deploy data centers globally [2, 3], but also place micro data centers in the network edge [4]. The infrastructure in different regions have different costs in terms of the resources, *e.g.* the computation resource, the storage resource, and the bandwidth resource [11]. As a result, there exist preferences to the type of service for given infrastructure. For example, SP would like to deploy a service with high bandwidth consumption to the infrastructure with lower bandwidth price. Moreover, the cost function in a single location for a specific resource is usually non-linear.

Complexity of Application: different customers have different applications, and these applications may have different features and different performance requirements. For example, some applications are latency-sensitive, such as augmented/virtual realities, while others are not. In addition, some applications are composed of more than one services. A typical example is living streaming applications that include video transcoder VM for video format transcoding, and video

This work was supported in part by the National Key Research and Development Program under Grant No. 2016YFB1000102, in part by the Natural Science Foundation of China under Grant No. 61571215 and No. 61672318, in part by the National Science Foundation under NSF ECCS-1509212, and in part by the Fundamental Research Funds for the Central Universities under Grant No. 021014380053 and No. 021014380099.

X. Zhang and Z. Ma are with the School of Electronic Science and Engineering in Nanjing University, China. Email: (xzhang17, mazhan)@nju.edu.cn;

H. Huang is with the School of Electronic Information and Communications, Huazhong University of Science and Technology, China. Email: hhj0704@hotmail.com.

H. Yin is with the Research Institute of Information Technology (RIIT) in Tsinghua University, China. Email:h-yin@mail.tsinghua.edu.cn.

D. Wu is with Department of Electrical & Computer Engineering at University of Florida, USA. Email: wu@ece.ufl.edu.

G. Min is with the Department of Computer Science, College of Engineering, Mathematics, and Physical Sciences, University of Exeter, Exeter, EX4 4QF, U.K. Email: g.min@exeter.ac.uk.

Corresponding authors: Zhan Ma, Hao Yin and Haojun Huang.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2875753, IEEE Internet of Things Journal

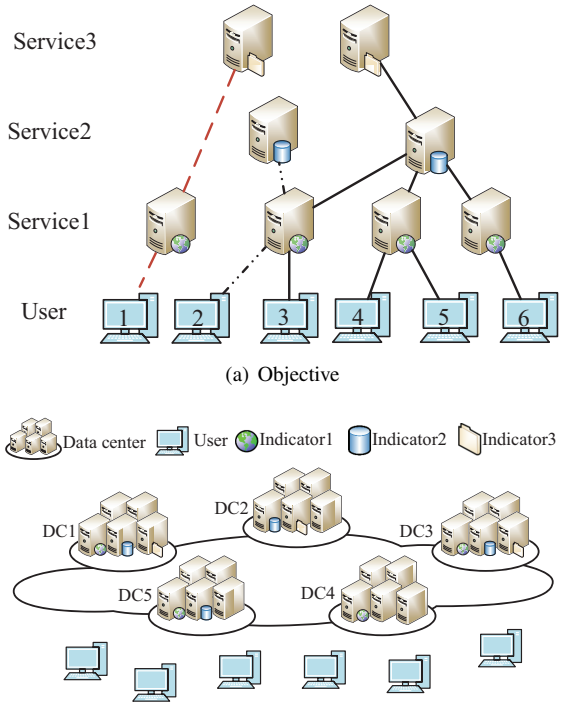
compressor VM for video compressing. Moreover, the access to different VMs should comply with a specific order. In other words, the request of a user should be handled by various types of VMs in sequence.

Complexity of Service: for different usages, the VM for a service may be designed to be computing-intensive, storage-intensive or bandwidth-intensive. As a result, different types of VMs have different consumption on computation resources, storage resources, and bandwidth resources. Moreover, these VMs may be located in the same infrastructure and compete for the same resources in the facility. In addition, some VMs should be deployed at specific locations due to policy restrictions or service features. For example, Session Initiation Protocol (SIP) VMs for IP Multimedia Subsystem (IMS) applications must be performed at the edge of the operator network.

To the best of our knowledge, no existing works take all these pragmatic factors into consideration. The most related work from the literature are the multi-Level capacitated and uncapacitated facility location problems [16, 20, 21]. These works aim at minimizing the sum of the fixed costs of the open facilities, plus the transportation cost of the clients' assignment, where each client's transportation cost is the sum of transportation cost from itself to the first facility of its sequence, plus the transportation cost between successive facilities of its sequence. ADMS *Problem* differs from them in the following aspects: 1) Different types of VMs may locate at the same facility and they share the facility's resources (*e.g.* computation resource, storage resource and bandwidth resource) in common. 2) The opening cost of a facility is not fixed, but is related to the type of opening VMs and the copy number of each type of VM. Moreover, the cost functions are always non-linear to the number of consumed resources. 3) One type of VM customized for a given infrastructure always consumes the same number of resource. In other words, the number of allocated resource for one type of VM in an infrastructure is a step function of the number of requested resource.

In order to solve the ADMS *problem*, this paper makes the following contributions.

- We formulate it as an optimization problem with the aim of minimizing the overall deployment cost under the latency/computaion/storage/bandwidth requirements and the infrastructure capacity limitations. Moreover, the formulation jointly considers the customer's and SP's preferences, the resource allocation model, and the resource cost model.
- As the ADMS *problem* is proved to be NP-hard, we design a workflow-based heuristic algorithm called *AMS*. *AMS* can determine how many VMs for each type of service should be placed, and where to place them. Moreover, *AMS* supports the services to scale up or scale down on demand in real time.
- We simulate *AMS* using the real Internet data from iPlane [5]. Simulations based on real network measurement demonstrate that *AMS* can reduce the number of deployed VMs by 28.4% and the deployment cost by 33.9% with comparable satisfied user ratio.



(a) Objective

(b) Constraints

Fig. 1: An example for deploying an application with multi-level services.

The rest of this paper is organized as follows. Section II gives an example of ADMS *Problem* to help better understanding and then formulates the problem taking various pragmatic factors into consideration. Section III presents our heuristical algorithm, which supports the services to scale up or scale down on demand in real time. The proposed method is evaluated in Section IV, and the related work is reviewed in Section V. Finally, Section VI summarizes the paper.

II. Problem Formulation

A. Problem Description

In general, a customer (an enterprise or an individual) would negotiate with a IoT Service Provider about the expected performance of its applications. Under the SLA with the customer, the SP would determine how to place the application in its infrastructure in order to minimize the resource consumption while satisfying the required performance of the application and the capacity constraints of its infrastructure, where the infrastructure includes the central data centers and micro data centers in the edge.

Actually an application may consist of more than one type of services. Fig. 1(a) shows an example that helps understanding the problem. In the example, the application is composed of three services: Service1-Service3. The target users of the application are usually distributed geographically. In the example, there exist 6 users (*i.e.*User1-User6).

When accessing the application, a user requires a subset of services by order. Without loss of generality, the request of User1 would follow the sequence of User1 $\leftrightarrow$ Service1 $\leftrightarrow$ Service3, the request of User 2 would follow the sequence

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2875753, IEEE Internet of Things Journal

of User2 $\leftrightarrow$ Service1 $\leftrightarrow$ Service2, while the request of User3-User6 should be served by User3/4/5/6 $\leftrightarrow$ Service1 $\leftrightarrow$ Service2 $\leftrightarrow$ Service3.

For latency-sensitive applications, the latency perceived by users from sending the request to getting the content should be less than a given threshold D_{max}. Without loss of generality, there exists $d_{\text{user1; service1}} + d_{\text{service1;service3}} < D_{max}$ for User1, where $d_{\text{user1; service1}}$ is the Round Trip Time (RTT) between User1 and the VM hosting Service1, $d_{\text{service1;service3}}$ is the Round Trip Time (RTT) between the VM holding Service1 and the VM holding Service3. The processing time within a is neglected comparing the the propagation latency between the user and data centers and the propagation latency between two data centers. If Service1 and Service3 are located in the same data center, $d_{\text{service1,service3}} = 0$. For compute/storage/bandwidth-intensive applications, the SP should provision enough resources related to the application in order to guarantee the SLA.

The infrastructure is always distributed geographically in order to provide better service and tolerate the failure of any data center. As shown in Fig. 1(b), the SP has 5 data centers in the network, that is, DC1-DC5. Each data center has resource capacity for computation/storage/bandwidth. Without loss of generality, DC1 has capacity of $\{U_1^c, U_1^s, U_1^b\}$, where U_1^c denotes the maximum computation resource (in a unit of CPU-hours), U_1^s denotes the maximum storage resource (in a unit of GB), and U_1^b denotes the maximum bandwidth (in a unit of MB/s).

Due to the resource price difference in different locations, SP may have special preferences to the services for the infrastructure. For example, the provider prefers to place bandwidth-intensive services on infrastructure with low bandwidth price. Without loss of generality, The cost when x units of computation resource, y units of storage resource, and z units of bandwidth resource are rented at the data center DC_1 is $F_1(x, y, z)$.

Moreover, some types of VM can only be placed at specific locations in the network, such as the SIP VM for IMS services. As shown in Fig. 1(b), a data center with the Indicator1 has the ability to accommodate the VMs of Service1. Thus the data center candidate sets for different services are different but may have intersections with each other. For example, the data center candidate set for Service1 is {DC1, DC3, DC4, DC5}, the data center candidate set for Service3 is {DC1, DC2, DC3}. The two candidate sets have {DC1, DC3} as the intersections, which means the two types of services will compete for the resources in {DC1, DC3}.

Then *ADMS* problem in the above simple example is defined as follows: given the distribution of users (User1-User6), the candidate infrastructure (DC1-DC5), the resource capacities in each infrastructure (*e.g.* $\{U_1^c, U_1^s, U_1^b\}$ for DC1), the resource cost function in each infrastructure (*e.g.* $F_1(x, y, z)$ for DC1), the type of services (Service1-Service3), the service sequence for each user (*e.g.* User1 $\leftrightarrow$ Service1 $\leftrightarrow$ Service3 for User1), the data center candidate set for each service (*e.g.* {DC1, DC3, DC4, DC5} for Service1), how many VMs for each service should be placed and where to place them.

TABLE I: The main terminologies used in the formulation.

Notation	Semantics
D	Set of the service provider's infrastructure, $D = \{D_1, D_2, ..., D_M\}$
M	Number of the service provider's infrastructure
U_i^c, U_i^s, U_i^b	Maximum available computation resource (in a unit of CPU-hours), storage resource (in a unit of GB), bandwidth (in a unit of MB/s) at D_i, respectively
V	Types of services required for the application, $V = \{V_1, V_2, ..., V_N\}$
N	Type number of the services for the application
P_i	Candidate data center set for the service V_i, $P_i = \{p_{i,j} \vert 1 \le i \le N, 1 \le j \le N_i\} \subseteq D$
$x_{i,j,k}$	Indicator variable, which is equal to 1 if $p_{i,j}$ is the data center D_k, and equal to 0 otherwise
S_i	VMs of the service V_i distributed among data centers P_i, $S_i = \{s_{i,j} \vert 1 \le i \le N, 1 \le j \le n_i\}$
$y_{i,j,k}$	Indicator variable, which is equal to 1 if $s_{i,j}$ is placed at data center $p_{i,k}$, and equal to 0 otherwise
$u_{i,j,k}^c, u_{i,j,k}^s, u_{i,j,k}^b$	Allocated computation resource, storage resource, and bandwidth resource for $s_{i,j}$ at $p_{i,k}$, respectively
U	User set for the application, $U = \{u_1, u_2, ..., u_R\}$
R	User number for the application
W_i	u_i's request is served following the sequence $W_i = \{w_{i,0}, w_{i,1}, ..., w_{i,r_i}\}$, where $w_{i,0}$ represent the user u_i itself, $w_{i,j}$ is a service for $1 \le j \le r_i$
$z_{i,j,k,l}$	Indicator variable, which is equal to 1 if $w_{i,j}$ belongs to the VM $s_{k,l}$, and equal to 0 otherwise
$R_{i,j}^c, R_{i,j}^s, R_{i,j}^b$	Number of the requested computation/storage/bandwidth resource for $s_{k,l}$ by U, respectively
$u_{i,j,k}^c, u_{i,j,k}^s, u_{i,j,k}^b$	Number of allocated computation/storage/bandwidth resources, respectively
$f_{i,j}(\cdot)$	Resource allocation function for service V_i at data center $p_{i,j}$
$F_i(x, y, z)$	Resource cost function when x units of computation resource, y units of storage resource, and z units of bandwidth resource are rented at the data center D_i

B. Problem Formulation

Table I lists the main terminologies in the formulation.

Given the set of a service provider's infrastructure $D = \{D_1, D_2, ..., D_M\}$, each element $D_i \in D, 1 \le i \le M$ represents a data center of the SP. Moreover, the capacity the data center of D_i can provide to the customer is $\{U_i^c, U_i^s, U_i^b\}$, where U_i^c denotes the maximum computation resource (in a unit of CPU-hours), U_i^s denotes the maximum storage resource (in a unit of GB), and U_i^b denotes the maximum bandwidth (in a unit of MB/s) to the customer.

Assume that there exist N types of services required for an application, *i.e.*, $V = \{V_1, V_2, ..., V_N\}$. For each kind of service $V_i \in V, 1 \le i \le N$, its candidate data center set is $P_i = \{p_{i,j} \vert 1 \le i \le N, 1 \le j \le N_i\} \subseteq D$, where N_i is the number of candidate data centers for the services= of type V_i, and $p_{i,j}$ is a data center in D, that is $p_{i,j} \in D$.

Let $x_{i,j,k}$ be an indicator variable, which is equal to 1 if $p_{i,j}$ is the data center D_k, and 0 otherwise.

$$x_{i,j,k} \in \{0, 1\} \quad \text{for all } i, j, k \tag{1}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2875753, IEEE Internet of Things Journal

Note that $p_{i,j} \in D$, there exists

$$\sum_{k} x_{i,j,k} = 1 \quad \text{for all } i,j \tag{2}$$

For the service V_i, it may consist of VMs in numerous sites distributed among data centers P_i, *i.e.*, $S_i = \{s_{i,j} | 1 \leq i \leq N, 1 \leq j \leq n_i\}$, where $s_{i,j}$ denotes the VMs of the service V_i at the same data center, and $n_i \leq N_i$. Note that it is possible that multi-VMs of the same service are placed at the same data center. Then the set of all VMs of all types of services is $S = \{S_i | 1 \leq i \leq N\}$.

Let $y_{i,j,k}$ be an indicator variable, which is equal to 1 if $s_{i,j}$ is placed at data center $p_{i,k}$, and 0 otherwise.

$$y_{i,j,k} \in \{0,1\} \quad \text{for all } i,j,k \tag{3}$$

Note that $s_{i,j}$ is placed in P_i, thus

$$\sum_{k} y_{i,j,k} = 1 \quad \text{for all } i,j \tag{4}$$

If $s_{i,j}$ is placed at data center $p_{i,k}$, let $\{u^c_{i,j,k}, u^s_{i,j,k}, u^b_{i,j,k}\}$ denote the allocated computation resource, storage resource, and bandwidth resource for $s_{i,j}$, respectively, then we have $\{u^c_{i,j,k}, u^s_{i,j,k}, u^b_{i,j,k}\} \leq \{U^c_k, U^s_k, U^b_k\}$.

Assume that there exist R users for the application, *i.e.*, $U = \{u_1, u_2, ..., u_R\}$. For each user $u_i \in U$, its request is served following the sequence $W_i = \{w_{i,0}, w_{i,1}, ..., w_{i,r_i}\}$, where $w_{i,0}$ represents the user u_i itself, r_i is the number of services for u_i's request, and the service $w_{i,j} \in S, \forall 1 \leq j \leq r_i$.

Let $z_{i,j,k,l}$ be an indicator variable, which is equal to 1 if $w_{i,j}$ belongs to the VMs $s_{k,l}$, and 0 otherwise.

$$z_{i,j,k,l} \in \{0,1\} \quad \text{for all } i,j,k,l \tag{5}$$

Note that there exist services to which $w_{i,j}$ belongs, thus

$$\sum_{k} \sum_{l} z_{i,j,k,l} = 1 \quad \text{for all } i,j \tag{6}$$

If $w_{i,j}$ belongs to $s_{k,l}$, where $1 \leq j \leq r_i$, the required computation resource, storage resource, and bandwidth resource at $w_{i,j}$ for u_i is $\{c^c_{i,j,k,l}, c^s_{i,j,k,l}, c^b_{i,j,k,l}\}$, respectively.

The number of the computation resource requested for $s_{k,l}$ by u_i is

$$\sum_{1 \leq j \leq r_i, j: z_{i,j,k,l}=1} c^c_{i,j,k,l} = \sum_{1 \leq j \leq r_i} c^c_{i,j,k,l} z_{i,j,k,l}$$

thus the overall number of the computation resource for $s_{k,l}$ requested by U is

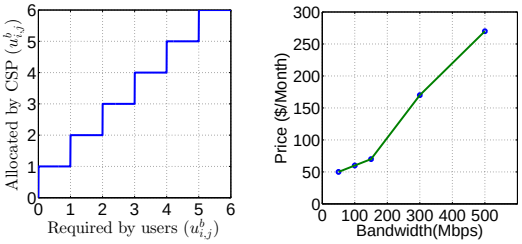
$$R^c_{k,l} = \sum_{1 \leq i \leq R} \sum_{1 \leq j \leq r_i} c^c_{i,j,k,l} z_{i,j,k,l} \tag{7}$$

Similarly, the overall number of the bandwidth resource for $s_{k,l}$ requested by U is

$$R^b_{k,l} = \sum_{1 \leq i \leq R} \sum_{1 \leq j \leq r_i} c^b_{i,j,k,l} z_{i,j,k,l} \tag{8}$$

and the overall number of the storage resource for $s_{k,l}$ requested by U is

$$R^s_{k,l} = \sum_{1 \leq i \leq R} \sum_{1 \leq j \leq r_i} c^c_{i,j,k,l} z_{i,j,k,l} \tag{9}$$



(a) Bandwidth Allocation (b) Bandwidth Pricing [6]

Fig. 2: An example for bandwidth allocation function and bandwidth pricing function.

If $s_{i,j}$ is placed at data center $p_{i,k}$, that is, $y_{i,j,k} = 1$, the number of computation/storage/bandwidth resources $\{u^c_{i,j,k}, u^s_{i,j,k}, u^b_{i,j,k}\}$ allocated by SP is determined by the number of computation/storage/bandwidth resources $\{R^c_{i,j}, R^s_{i,j}, R^b_{i,j}\}$ for $s_{i,j}$ requested by U:

$$\forall\ y_{i,j,k} = 1: \quad \{u^c_{i,j,k}, u^s_{i,j,k}, u^b_{i,j,k}\} = f_k(R^c_{i,j}, R^s_{i,j}, R^b_{i,j})$$

In other words,

$$\{u^c_{i,j,k}, u^s_{i,j,k}, u^b_{i,j,k}\} = f_{i,k}(R^c_{i,j}, R^s_{i,j}, R^b_{i,j}) y_{i,j,k} \tag{10}$$

where $f_{i,k}(\cdot)$ is the resource allocation function for service V_i at data center $p_{i,k}$.

If the number of computation/storage/bandwidth resources consumed by each copy of service V_i at data center $p_{i,k}$ is $\{u^c_{i,j}, u^s_{i,j}, u^b_{i,j}\}$, respectively, then a kind of resource allocation function is a step function, as shown in Fig. 2(a).

$$f_{i,k}(R^c_{i,j}, R^s_{i,j}, R^b_{i,j}) = max\{\lceil \frac{R^c_{i,j}}{u^c_{i,j}} \rceil, \lceil \frac{R^s_{i,j}}{u^s_{i,j}} \rceil, \lceil \frac{R^b_{i,j}}{u^b_{i,j}} \rceil\} * \{u^c_{i,j}, u^s_{i,j}, u^b_{i,j}\}$$

where $max\{\lceil \frac{R^c_{i,j}}{u^c_{i,j}} \rceil, \lceil \frac{R^s_{i,j}}{u^s_{i,j}} \rceil, \lceil \frac{R^b_{i,j}}{u^b_{i,j}} \rceil\}$ is the minimum number of VMs for service V_i at data center $p_{i,k}$ in order to provide the number of requested resources.

Note that the overall number of the computation resource consumed at D_k should not be more than the computation capacity of D_k, that is,

$$\sum_{1 \leq i \leq N} \sum_{1 \leq j \leq N_i} u^c_{i,j,k} x_{i,j,k} \leq U^c_k \quad \text{for all } k \tag{11}$$

Similarly, the overall number of the storage resource consumed at D_k should not be more than the storage capacity of D_k, that is,

$$\sum_{1 \leq i \leq N} \sum_{1 \leq j \leq N_i} u^s_{i,j,k} x_{i,j,k} \leq U^s_k \quad \text{for all } k \tag{12}$$

and the overall number of the bandwidth resource consumed at D_k should not be more than the bandwidth capacity of D_k, that is,

$$\sum_{1 \leq i \leq N} \sum_{1 \leq j \leq N_i} u^b_{i,j,k} x_{i,j,k} \leq U^b_k \quad \text{for all } k \tag{13}$$

Considering that many applications have requirements on the latency users perceived. The latency perceived by user u_i

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2875753, IEEE Internet of Things Journal

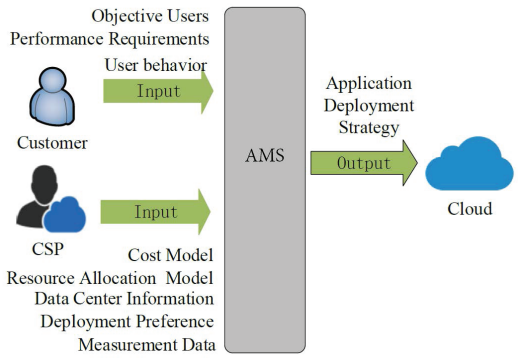


Fig. 3: The input and output of *AMS*.

should not be more than the expected maximum latency D_{max}, that is,

$$\sum_{1\le j\le r_i} d_{w_{i,j-1};w_{i,j}} \le D_{max} \quad \text{for all } i \tag{14}$$

where $d_{x;y}$ is the latency between entity x and y.

For the latency $d_{w_{i,j};w_{i',j'}}$ between $w_{i,j}$ and $w_{i',j'}$, the following equations hold true according to Eqs.(1)-(6):

$$\begin{aligned} \exists k,l,g,h, \quad s.t. \quad & z_{i,j,k,l} = 1 \\ & y_{k,l,g} = 1 \\ & x_{k,g,h} = 1 \end{aligned}$$

$$\begin{aligned} \exists k',l',g',h', \quad s.t. \quad & z_{i',j',k',l'} = 1 \\ & y_{k',l',g'} = 1 \\ & x_{k',g',h'} = 1 \end{aligned} \tag{15}$$

$$d_{w_{i,j};w_{i',j'}} = d_{D_h;D_{h'}}$$

where $w_{i,j}$ is located at Data center D_h while $w_{i',j'}$ is located at data center $D_{h'}$.

The overall cost for the consumed resources is as follows:

$$\sum_{1\le k\le M} F_k\Big(\sum_{1\le i\le N}\sum_{1\le j\le N_i} u^c_{i,j,k}x_{i,j,k}, \sum_{1\le i\le N}\sum_{1\le j\le N_i} u^s_{i,j,k}x_{i,j,k}, \sum_{1\le i\le N}\sum_{1\le j\le N_i} u^b_{i,j,k}x_{i,j,k}\Big) \tag{16}$$

where $F_k(x,y,z)$ is the cost when x units of computation resource, y units of storage resource, and z units of bandwidth resource are rented at the data center D_k. The cost function is always non-linear, as Verizon's bandwidth pricing function [6] shown in Fig. 2(b).

Our objective is to minimize the overall placement cost given the latency expectation and resource capacities constraints. Then the *ADMS* problem can be formulated to minimize the above cost function given the constraints (1)-(15).

III. Methodology

In this section, we design a heuristic algorithm, named *AMS* to solve the formulated problem.

As shown in Fig. 3, *AMS* takes the application requirements from the customer and the status of infrastructure from the SP to compute the optimized application deployment strategy among the infrastructure, which is feasible when *AMS* is adopted by the SP. In detail, the customer should provide the objective users ($U = \{u_1, u_2, ..., u_R\}$), the performance requirements (*e.g.* the expected maximum average latency D_{max}) and the user behavior (*e.g.* the accessing sequence to each type of services W); while the SP should have awareness to its own infrastructure, including the cost model (*e.g.* the set of resource cost functions for each data center F), the resource allocation model (*e.g.* the set of resource allocation functions for each type of service at each data center f), the data center information (*e.g.* the locations and the capacities of its data centers D), the deployment preference (set of data center preferences for each type of service P) [33], and the measurement data (the latencies between data centers and the latencies between data centers and objective users). There exist many works that provide lightweight methods to measure the latency between arbitrary two hosts in the large-scale network, such as GNP [22] and SSL [33]. *AMS* can take advantage of these methods to measure the latencies between data centers and the latencies between data centers and objective users. The selection of measuring method is out of scope in this paper. Taking all these factors into consideration, *AMS* aims at reducing the deployment cost while satisfying the application's performance requirements and the capacity limitation of infrastructure. The overall algorithm is shown in Algorithm 1.

In order to avoid bias, *AMS* randomly shuffles the order of the users in U at first, and then determines whether to open VMs and where to place the open VMs for each service following a greedy mode. In detail, for each user $u_i \in U$, the service sequence that its request should follow is $W_i = \{w_{i,0}, w_{i,1}, ..., w_{i,r_i}\}$, where $w_{i,0}$ is the user itself. For example, in the case of Fig. 1, the service sequence for User1 is $W_1 = \{$User1, Service1, Service3$\}$. Then *AMS* finds the candidate data center set $P_{k_{i,j}}$ for each $w_{i,j} \in W_i \backslash \{w_{i,0}\}$. Based on the candidate data center set for each service in the sequence, *AMS* can enumerate all the possible paths $P_{k_{i,j}} \times ... \times P_{k_{i,r_i}}$ that can provide the sequence of services. For example, in the case of Fig. 2, $P_{k_{1,1}}$ for $w_{1,1}$ is the candidate data center set for Service1, that is, $P_{k_{1,1}} = P_1 = \{$DC1, DC3, DC4, DC5$\}$; $P_{k_{1,2}}$ for $w_{1,2}$ is the candidate data center set for Service3, that is, $P_{k_{1,2}} = P_3 = \{$DC1, DC2, DC3$\}$; and the element in $P_{k_{1,1}} \times P_{k_{1,2}}$ is a pair of data centers, which contains an element in $P_{k_{1,1}}$ and an element in $P_{k_{1,2}}$. Thus the number of possible paths for User1 is 12. Among the possible serving paths of u_i, *AMS* chooses all the feasible paths that satisfy the latency requirements ($d_{u_i;L} \le D_{max}$) and resource limitation constraints (*checkPath*(u_i, L, f, D) is *true*) into set $\mathbb{L}$. If there exists any feasible path ($\mathbb{P} \neq \emptyset$) for user u_i, *AMS* chooses the path $L^* \in \mathbb{P}$ with the lowest incremental cost, that is,

$$pathCost(u_i, L^*, f, F, D) \le pathCost(u_i, L, f, F, D), \forall L \in \mathbb{L}$$

to serve the user. In addition, *AMS* updates the requested computation/storage/bandwidth resources for each type of service and their locations according to L^* (*updateRequestResource*(u_i, L^*, f, D)), and records the assignment to S. For the user u_i that there does not exist feasible paths ($\mathbb{L} = \emptyset$), *AMS* records them into U'.

AMS can determine how many VMs for each type of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2875753, IEEE Internet of Things Journal

Algorithm 1: *AMS* (U, W, D_{max}, D, P, f, F)

// U: User set for the application, $U = \{u_1, u_2, ..., u_R\}$
// W: Set of the service sequence for each user
// D_{max}: Expected maximum latency user perceived when using the application
// D: Set of the service provider's infrastructure
// P: Set of data center preferences for each type of service
// f: Set of resource allocation functions for each type of service at each data center
// F: Set of resource cost functions for each data center
$Q \leftarrow \emptyset$; // requested resources for each type of services and their locations
$S \leftarrow \emptyset$; // user assignment strategy
$U^{'} \leftarrow \emptyset$; // users that are not unassigned
$U \leftarrow U$.shuffle();
foreach $u_i \in U$ **do**
  //Serving sequence of u_i, where $w_{i,0} = u_i$
  $W_i \leftarrow \{w_{i,0}, w_{i,1}, ..., w_{i,r_i}\}$;
  foreach $w_{i,j} \in W_i \backslash \{w_{i,0}\}$ **do**
    //$w_{i,j}$ belongs to the service $V_{k_{i,j}}$
    $P_{k_{i,j}} \leftarrow$ Candidate data center set for $w_{i,j}$;
  //Find all feasible paths for u_i
  $\mathbb{L} \leftarrow \emptyset$
  foreach $L \in P_{k_{i,1}} \times ... \times P_{k_{i,r_i}}$ **do**
    if $d_{u_i;L} \leq D_{max}$ **then**
      if *checkPath*(u_i, L, f, D) **then**
        $\mathbb{L} \bigcup \{L\}$;
  //Choosing the path with the lowest incremental cost to serve u_i
  if $\mathbb{L} \neq \emptyset$ **then**
    $L^* \leftarrow L$ with the lowest *pathCost*(u_i, L, f, F, D) in $\mathbb{P}$;
    $Q \leftarrow$ *updateRequestResource*(u_i, L^*, f, D);
    $S \bigcup \{\{u_i, L^*\}\}$;
  else
    $U^{'} \bigcup \{u_i\}$
return Q, S, $U^{'}$;

Algorithm 2: checkPath(u_i, L, f, D)

// u_i: User who requests the application
// L: Service sequence for user u_i, $L \in P_{k_{i,1}} \times ... \times P_{k_{i,r_i}}$
// f: Set of resource allocation functions for each type of service at each data center
// D: Set of the service provider's infrastructure
$\{\alpha^c_j, \alpha^s_j, \alpha^b_j\} \leftarrow \{A^c_j, A^s_j, A^b_j\}$ for each D_j //Temporal variables storing the unallocated number of computation/storage/bandwidth resource at each data center
$\{\beta^c_{j,l}, \beta^s_{j,l}, \beta^b_{j,l}\} \leftarrow \{B^c_{j,l}, B^s_{j,l}, B^b_{j,l}\}$ for each D_j and each service V_l //Temporal variables storing the requested number of computation/storage/bandwidth resource at each data center for each type of service
$flag \leftarrow true$;
foreach $l_j \in L$ **do**
  //The number of extra resources provisioned for l_j, which is hosted by D_h
  $\Delta R_j = f_{k_{i,j},g}(\beta^c_{h,k_{i,j}} + c^c_{k_{i,j},g}, \beta^s_{h,k_{i,j}} + c^s_{k_{i,j},g}, \beta^b_{h,k_{i,j}} + c^b_{k_{i,j},g}) - f_{k_{i,j},g}(\beta^c_{h,k_{i,j}}, \beta^s_{h,k_{i,j}}, \beta^b_{h,k_{i,j}})$;
  if $\Delta R_j \leq \{\alpha^c_h, \alpha^s_h, \alpha^b_h\}$ **then**
    //update the temporal variable of the requested resource number at D_h for service $V_{k_{i,j}}$
    $\{\beta^c_{h,k_{i,j}}, \beta^s_{h,k_{i,j}}, \beta^b_{h,k_{i,j}}\} = \{\beta^c_{h,k_{i,j}} + c^c_{k_{i,j},g}, \beta^s_{h,k_{i,j}} + c^s_{k_{i,j},g}, \beta^b_{h,k_{i,j}} + c^b_{k_{i,j},g}\}$;
    //update the temporal variable of the unallocated resource number at D_h
    $\{\alpha^c_h, \alpha^s_h, \alpha^b_h\} = \{\alpha^c_h, \alpha^s_h, \alpha^b_h\} - \Delta R_j$;
    continue;
  $flag \leftarrow false$;
  break;
return $flag$;

service should be placed, and where these VMs be placed in $O(RM^K)$ time complexity, where R is the number of users, M is the number of Dater Centers, and K is the maximum number of different services in a serve sequence, that is, $K = max\{r_i | 1 \leq i \leq R\}$.

If there are residual users after the process ($U^{'} \neq \emptyset$), *AMS* will relax the performance requirements D_{max} for them, and go through the basic process until they are served or the resources in the infrastructure are exhausted.

lemma III.1. *If there exists only one user to be served, that is, $|U| = 1$, the service deployed by* AMS *is pareto-optimal.*

Proof. As described by Algorithm 1, *AMS* enumerates all the paths which satisfies the latency requirements ($d_{u_i;L} \leq D_{max}$) and the resource capacity requirements ($checkPath(u_i, L, f, D)$ returns $True$), and selects the path L^* with the lowest cost to serve the user.

It's obvious that we cannot find another strategy that satisfies the latency requirement but with lower cost to serve the user. So, the service deployed by *AMS* is pareto-optimal. □

In Algorithm 1, we invoke the *checkPath* function, which checks whether the infrastructures carrying the path L have enough resources for the user u_i. During the checking process, we do not really open the path, thus using a set of temporal variables $\{\alpha^c_j, \alpha^s_j, \alpha^b_j\}$ to represent the unallocated amount of computation/storage/bandwidth resources at data center D_j and a set of temporal variables $\{\beta^c_{j,l}, \beta^s_{j,l}, \beta^b_{j,l}\}$ to represent the requested number at data center D_j for the service V_l.

For each node $l_j \in P_{k_{i,j}}$ along the path L, it provides the service $V_{k_{i,j}}$ and is located at the data center $p_{k_{i,j},g}$, that is, D_h. The resource allocation function for service $V_{k_{i,j}}$ at data center $p_{k_{i,j},g}$ is $f_{k_{i,j},g}(\cdot)$. Note that the requested resource number by user u_i at D_h for the service $V_{k_{i,j}}$ is $c^c_{k_{i,j},g}, c^s_{k_{i,j},g}, c^b_{k_{i,j},g}$. Thus, the total requested resource number at D_h for the service $V_{k_{i,j}}$ is

$$\{\beta^c_{h,k_{i,j}}, \beta^s_{h,k_{i,j}}, \beta^b_{h,k_{i,j}}\} \leftarrow \{\beta^c_{h,k_{i,j}} + c^c_{k_{i,j},g}, \beta^s_{h,k_{i,j}} + c^s_{k_{i,j},g}, \beta^b_{h,k_{i,j}} + c^b_{k_{i,j},g}\}$$

Considering the resource allocation function, the number of allocated resources should be

$$f_{k_{i,j},g}(\beta^c_{h,k_{i,j}} + c^c_{k_{i,j},g}, \beta^s_{h,k_{i,j}} + c^s_{k_{i,j},g}, \beta^b_{h,k_{i,j}} + c^b_{k_{i,j},g}).$$

The number of extra resources provisioned for l_j is

$$\Delta R = f_{k_{i,j},g}(\beta^c_{h,k_{i,j}} + c^c_{k_{i,j},g}, \beta^s_{h,k_{i,j}} + c^s_{k_{i,j},g}, \beta^b_{h,k_{i,j}} + c^b_{k_{i,j},g}) - f_{k_{i,j},g}(B^c_{h,k_{i,j}}, B^s_{h,k_{i,j}}, B^b_{h,k_{i,j}})$$

If these resources are allocated to L, the left unallocated resource number at D_h would be

$$\{\alpha^c_h, \alpha^s_h, \alpha^b_h\} \leftarrow \{\alpha^c_h, \alpha^s_h, \alpha^b_h\} - \Delta R$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2875753, IEEE Internet of Things Journal

Algorithm 3: pathCost(u_i, L, f, F, D)

// u_i: User who requests the application
// L: Service sequence for user u_i, $L \in P_{k_{i,1}} \times ... \times P_{k_{i,r_i}}$
// f: Set of resource allocation functions for each type of service at each data center
// F: Set of resource cost functions for each data center
// D: Set of the service provider's infrastructure

$\{\beta^c_{j,l}, \beta^s_{j,l}, \beta^b_{j,l}\} \leftarrow \{B^c_{j,l}, B^s_{j,l}, B^b_{j,l}\}$ for each D_j and service V_l //Temporal variables storing the requested number of computation/storage/bandwidth resource at each data center for each type of service

foreach $l_j \in L$ **do**
    //update the temporal variable of the requested resource number at D_h for service $V_{k_{i,j}}$
    $\{\beta^c_{h,k_{i,j}}, \beta^s_{h,k_{i,j}}, \beta^b_{h,k_{i,j}}\} =$
    $\{\beta^c_{h,k_{i,j}} + c^c_{k_{i,j},g}, \beta^s_{h,k_{i,j}} + c^s_{k_{i,j},g}, \beta^b_{h,k_{i,j}} + c^b_{k_{i,j},g}\}$;

return $\sum_j F_j(\sum_l f_{l,j}(\beta^c_{j,l}, \beta^s_{j,l}, \beta^b_{j,l})$;

Algorithm 4: updateRequestResource(u_i, L, f, D)

// u_i: User who requests the application
// L: Service sequence for user u_i, $L \in P_{k_{i,1}} \times ... \times P_{k_{i,r_i}}$
// f: Set of resource allocation functions for each type of service at each data center
// D: Set of the service provider's infrastructure

global $\{A^c_j, A^s_j, A^b_j\}$ for each D_j //The unallocated number of computation/storage/bandwidth resource at each data center
global $\{B^c_{j,l}, B^s_{j,l}, B^b_{j,l}\}$ for each D_j and service V_l //The requested number of computation/storage/bandwidth resource at each data center for each type of service

foreach $l_j \in L$ **do**
    //update the requested resource number at D_h for service $V_{k_{i,j}}$
    $\{B^c_{h,k_{i,j}}, B^s_{h,k_{i,j}}, B^b_{h,k_{i,j}}\} =$
    $\{B^c_{h,k_{i,j}} + c^c_{k_{i,j},g}, B^s_{h,k_{i,j}} + c^s_{k_{i,j},g}, B^b_{h,k_{i,j}} + c^b_{k_{i,j},g}\}$;
    //update the unallocated resource number at D_h
    $\{A^c_h, A^s_h, A^b_h\} =$
    $\{A^c_h, A^s_h, A^b_h\} - f_{k_{i,j},g}(B^c_{h,k_{i,j}} + c^c_{k_{i,j},g}, B^s_{h,k_{i,j}} +$
    $c^s_{k_{i,j},g}, B^b_{h,k_{i,j}} + c^b_{k_{i,j},g}) + f_{k_{i,j},g}(B^c_{h,k_{i,j}}, B^s_{h,k_{i,j}}, B^b_{h,k_{i,j}})$;

return $\{B^c_{j,l}, B^s_{j,l}, B^b_{j,l}\}$;

checkPath function goes over every nodes in L to check whether the extra number of resources for the whole path L can be allocated, as described in Algorithm 2.

The *pathCost* function invoked in Algorithm 1 computes the overall cost if a path is opened for a user. Similar to the *checkPath* function, the *pathCost* function does not really open the path and uses a set of temporal variables to represent the requested number at each data center for each type of service. Then the function adds the resource according to the requested computation/storage/bandwidth resources when the user is served by the path. After that, the function traverses all types of services at all data centers, computes the cost based on the resource allocation function and the cost model, as described in Algorithm 3, where $(\beta^c_{j,l}, \beta^s_{j,l}, \beta^b_{j,l})$ is the requested number of computation/storage/bandwidth resources at data center D_j for service V_l, $f_{i,j}(\beta^c_{j,l}, \beta^s_{j,l}, \beta^b_{j,l})$ is the allocated number of resources at data center D_j.

After choosing the path which incurs the lowest extra resource cost, *AMS* will invoke $updateRequestResource$ to update the requested resources for the application, including

Fig. 4: The geographical distribution of the simulated data centers.

increasing the real number of the requested resources and decreasing the unallocated resource number at related infrastructure. In detail, Algorithm 4 maintains the set of values $\{A^c_j, A^s_j, A^b_j\}$ to represent the unallocated amount of computation/storage/bandwidth resources at data center D_j and the set of values $\{B^c_{j,l}, B^s_{j,l}, B^b_{j,l}\}$ to represent the requested number at data center D_j for the service V_l. When opening the path L for user u_i, Algorithm 4 goes over every nodes in L to allocate the extra number of resources for the whole path L. The process is described in Algorithm 4.

In the application deployment scenario, the ability to scale up or scale down the services on demand is critical both to the customer and the SP. It's feasible for *AMS* to handle this.

When there exist users joining or quitting to use the application, their requested numbers of computation/storage/bandwidth resources would vary accordingly. What *AMS* would do is recompute the allocated number of resources for each type of service at each data center based on the requested numbers. After that, *AMS* updates the number of VMs for each type of service at each data center. If the updated number of VMs increases, it indicates that SP should scale up the service to accommodate the increasing population; while reducing the updated number of VMs when scaling down the service to reduce cost. Note that when the updated number of VMs reduces, SP should shut down some VMs of the service and migrate the users on the shut-down VMs to the open VMs in the same data center.

IV. Evaluation

In order to evaluate *AMS*, we simulate *AMS* using the real measurement data in the global Internet and compare it with a revised algorithm in the literature.

We use the iPlane [19] nodes to simulate the data centers for a SP, the latencies between two arbitrary iPlane nodes to represent the latencies between two arbitrary data centers, and the latencies between the iPlane nodes and the objective users to represent the latencies between data centers and the objective users. The latencies were measured during 2016/2/10-2016/2/14, and the overall measurement data was downloaded from [5]. To guarantee the completeness of the measurement data, we pre-process the data set and get 26 nodes and 84153 users that had valid measurement latencies between arbitrary two nodes and latencies between the nodes and the users, that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2018.2875753, IEEE Internet of Things Journal

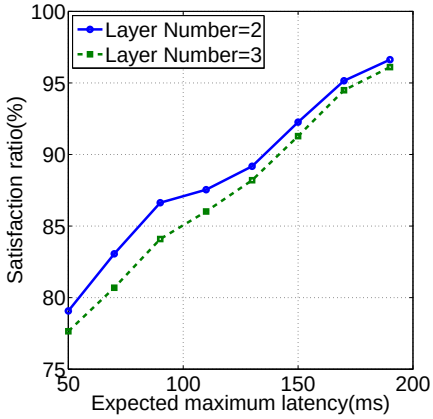


Fig. 5: The relationship between the expected maximum latency and the satisfied user ratio of *AMS*.

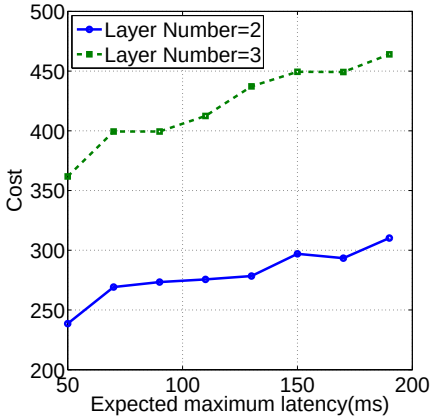


Fig. 7: The relationship between the expected maximum latency and the overall deployment cost of *AMS*.

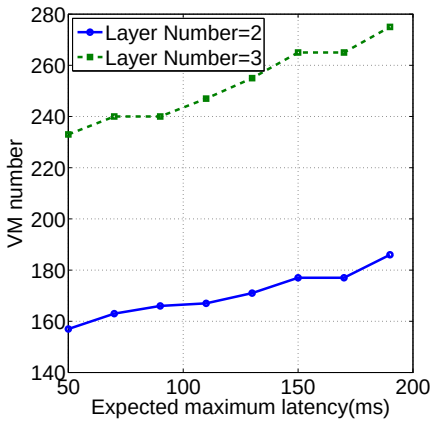


Fig. 6: The relationship between the expected maximum latency and the total number of VM deployed by *AMS*.

is, there exist 26 data centers for the SP and 84153 objective users for the application in our simulation. The simulated data centers and the objective users are both distributed across the global Internet. As shown in Fig. 4, the simulated data centers are mainly distributed in North America and Europe. To simplify the simulation, we only consider the bandwidth resource, and use the relative bandwidth prices around the world [1] to represent the relative VM cost at different data centers.

We explore the ratio of satisfied users, the number of VM, and the deployment cost for *AMS* under different layer numbers and different expected maximum latencies, where the layer number is defined as the length of users' service sequences. As shown in Fig. 5, the ratio of satisfied users increases as the expected maximum latency increases. However, increasing the layer number would make the satisfied user ratio decrease. This follows our intuition that the lower our expectation is, the higher satisfaction ratio will be. Fig. 6 shows the relationship between the needed VM number and the expected maximum latency. As the expected maximum latency increases, the satisfied user ratio increases. In this simulation, we do not relax the expected maximum latency for the unsatisfied users, and just deploy VMs for the satisfied users. Due to this, the number of deployed VMs is positively related to the satisfied user ratio and negatively related to the expected maximum latency.

Fig. 7 shows the relationship between the deployment cost and the expected maximum latency. We can see that the deployment cost is basically consistent with the number of deployed VMs, but not totally in direct proportional due to the price differences in different regions.

In order to evaluate *AMS*, we compare it with the Dynamic Vogel Approximation Method (*DVAM*) in the literature [16], which has been revised to solve the *ADMS* problem and referred to as *rDVAM*. In the *ADMS* problem, different types of services may locate at the same facility and they share the facility's resources in common, while in [16] different types of services must locate at different facilities. The crucial revision to the DVAM is the way to update the spare capacity of facilities. *rDVAM* updates the facility's spare capacity based on all the located services in a path. The key idea of *rDVAM* is to assign the user with the highest regret value at first, which is defined as the arithmetic difference between the smallest and next-to-the-smallest incremental cost.

Moreover, we compare *AMS* with the strategy of surrogating [15], where a user will request the first service from the closest candidate server. Afterwards, the server will act as the surrogate for the user, requesting the followed services and feeding back the results to the user.

Due to the high time complexity of *rDVAM*, we compare *AMS* with the other two methods using relative small scale user set. Table II lists the ratio of satisfied users, the number of deployed VMs, and the deployment cost for the three methods.

Note that *rDVAM* is much slower than *AMS*, especially in the large-scale network, as its time complexity of *rDVAM* is $O(R^2M^K)$. Moreover, the high complexity makes *rDVAM* hard to be used for services in real time scale-up or scale-down. Although *rDVAM* outperforms the other two methods in terms of satisfied users, the number of deployed VMs and the deployment cost, its complexity limits the usability in large-scale networks. Compared to *Surrogating*, *AMS* can reduce the VM number by 28.4% and the deployment cost by 33.9% with comparable satisfied user ratio.

TABLE II: Performance Comparison when the expected maximum latency for users is 70ms with the layer number equals to 3.

Method	Satisfaction Ratio	VM number	Cost
AMS	97.2%	73	107.8
rDVAM	99.9%	62	87.4
Surrogating	97.5%	102	163.2

V. Related Work

IoT and the computation offloading from the IoT devices has attracted many attentions from the literature [12, 23, 26, 29, 30] and the Industry [8]. Zhou *et al.* [34] integrated IoT and Cloud to accelerate IoT application, development, and management. Vogler *et al.* [27] proposed to take advantage of the limited execution environments in IoT devices and designed a framework for the separation of independently executable application components. Ren *et al.* [24] proposed a transparent computing based IoT architecture to move service provisioning from the cloud to the edge. Lyu *et al.* [18] presented an integration architecture of the cloud, MEC, and IoT to resolve the scalability problem caused by the massive number of IoT devices. However, these works mainly focus on the offloading for applications with a single service.

To the best of our knowledge, the service deployment for an application with multi-level services has not been investigated before. The most related work in the literature is the Multi-Level Capacitated Facility Location Problems (MCFLP). These works aim at minimizing the sum of the fixed costs of the open facilities, plus the transportation cost of the clients' assignment, where each client's transportation cost is the sum of transportation cost from itself to the first facility of its sequence, plus the transportation cost between successive facilities of its sequence. Chen *et al.* [16] proposed an algorithm based on the concept of Vogels Approximation Method with dynamic cost at each iteration. Another variant is Multi-Level Uncapacitated Facility Location Problems (MCFLP), with capacity constraints in each facility. Marić *et al.* proposed a Genetic Algorithm [20] in 2010 and a memetic algorithm [21] in 2014 to solve the MCFLP.

There also exist some work aiming at deploying VMs at multi datacenters. Chaisiri *et al.* [14] proposed an optimal algorithm to minimize the total cost due to buying reservation and on-demand plans of resource provisioning. Agarwal *et al.* [10] presented an automated mechanism to place application data across geo-distributed datacenters. Zhang *et al.* [32] solved the dynamic service placement problems based on control- and game-theoretic models. Cohen *et al.* [17] formulated the VM placement problem into the combination of the facility location problem and the generalized assignment problem. *AMS* differs from these prior works by deploying VMs for applications with multi-level services.

VI. Conclusion

In this paper, we formulate the service placement problem for IoT applications with multi-level services as an optimization problem with the aim of minimizing the overall deployment cost under the latency/computaion/storage/bandwidth requirements and the infrastructure capacity limitations. Moreover, the formulation jointly considers the customer's and SP's preferences, the resource allocation model, and the resource cost model. To the best of our knowledge, this is the first time that service placement for IoT applications with multi-level services is studied. Due to its hardness, we designed a workflow-based heuristic algorithm called *AMS*, which supports the scale-up and scale-down of services on demand. What's more, we simulated *AMS* using the real measurement data in the global network and compared it with revised algorithm in the literature. Simulations based on real network measurement demonstrate that *AMS* can reduce the number of deployed VMs by 28.4% and the deployment cost by 33.9% with comparable satisfied user ratio.

References

[1] The Relative Cost of Bandwidth Around the World. https://blog.cloudflare.com/the-relative-cost-of-bandwidth-around-the-world/, 2014.

[2] AWS Global Infrastructure. https://aws.amazon.com/about-aws/global-infrastructure/, 2016.

[3] Azure Regions, Microsoft Azure. https://azure.microsoft.com/en-us/regions/, 2016.

[4] Cloud Locations, Goolge CloudPlatform. https://cloud.google.com/about/locations/, 2016.

[5] iPlane: Datasets. http://iplane.cs.washington.edu/data/data.html, 2016.

[6] Pick Your Fios Home Internet Plan. http://www.verizon.com/home/fios-fastest-internet/, 2016.

[7] Market Pulse Report, Internet of Things (IoT). https://growthenabler.com/flipbook/pdf/IOT\%20Report.pdf, 2017.

[8] AWS IoT, A system of ubiquitous devices connecting the physical world to the cloud. https://aws.amazon.com/iot/, 2018.

[9] IoT device installed base worldwide 2009-2020. https://www.statista.com/statistics/764026/number-of-iot-devices-in-use-worldwide/, 2018.

[10] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan. Volley: Automated Data Placement for Geo-distributed Cloud Services. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, pages 2–2, Berkeley, CA, USA, 2010. USENIX Association.

[11] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. The price is right: Towards location-independent costs in datacenters. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, pages 23:1–23:6, New York, NY, USA, 2011. ACM.

[12] A. Botta, W. de Donato, V. Persico, and A. Pescap. On the integration of cloud computing and internet of things. In *2014 International Conference on Future Internet of Things and Cloud*, pages 23–30, Aug 2014.

[13] X. Cao, T. Yue, X. Lin, S. Lin, X. Yuan, Q. Dai, L. Carin, and D. J. Brady. Computational Snapshot Multispectral Cameras: Toward dynamic capture of the spectral world. *IEEE Signal Processing Magazine*, 33(5):95–108, Sept 2016.

[14] S. Chaisiri, B.-S. Lee, and D. Niyato. Optimal Virtual Machine Placement across Multiple Cloud Providers. In *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, pages 103–110, Dec 2009.

[15] F. Chen, K. Guo, J. Lin, and T. L. Porta. Intra-cloud lightning: Building cdns in the cloud. In *INFOCOM, 2012 Proceedings IEEE*, pages 433–441, March 2012.

[16] Y.-Y. Chen and H.-F. Wang. Applying a Revised VAM to a Multi-level Capacitated Facility Location Problem. In *2007 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 337–341, Dec 2007.

[17] R. Cohen, L. Lewin-Eytan, J. Naor, and D. Raz. Near Optimal Placement of Virtual Network Functions. In *Proceedings of 2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 1346–1354, April 2015.

[18] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, and Y. Zhang. Selective offloading in mobile edge computing for the green internet of things. *IEEE Network*, 32(1):54–60, Jan 2018.

[19] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An Information Plane for Distributed Services. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*, OSDI '06, pages 367–380, Berkeley, CA, USA, 2006. USENIX Association.

[20] M. Marić. An Efficient Genetic Algorithm for Solving Muti-level Uncapacitated Facility Location Problem. *Computing and Informatics*, 29:183–201, 2010.

[21] M. Marić, Z. Stanimirović, A. Djenić, and P. Stanojević. Memetic Algorithm for Solving the Multilevel Uncapacitated Facility Location Problem. *Informatica*, 25(3):439–466, July 2014.

[22] T. S. E. Ng and H. Zhang. Towards global network positioning. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW '01, pages 25–29, New York, NY, USA, 2001. ACM.

[23] X. Peng, J. Ren, L. She, D. Zhang, J. Li, and Y. Zhang. BOAT: A Block-Streaming App Execution Scheme for Lightweight IoT Devices. *IEEE Internet of Things Journal*, 5(3):1816–1829, June 2018.

[24] J. Ren, H. Guo, C. Xu, and Y. Zhang. Serving at the Edge: A Scalable IoT Architecture Based on Transparent Computing. *IEEE Network*, 31(5):96–105, 2017.

[25] J. Santos, T. Wauters, B. Volckaert, and F. D. Turck. Resource provisioning for IoT application services in smart cities. In *2017 13th International Conference on Network and Service Management (CNSM)*, pages 1–9, Nov 2017.

[26] R. M. Shukla and A. Munir. A computation offloading scheme leveraging parameter tuning for real-time iot devices. In *2016 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)*, pages 208–209, Dec 2016.

[27] M. Vogler, J. Schleicher, C. Inzinger, and S. Dustdar. Optimizing elastic iot application deployments. *IEEE Transactions on Services Computing*, pages 1–1, 2016.

[28] H. Yin, X. Zhang, H. H. Liu, Y. Luo, C. Tian, S. Zhao, and F. Li. Edge Provisioning with Flexible Server Placement. *IEEE Transactions on Parallel and Distributed Systems*, 28(4):1031–1045, April 2017.

[29] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue. On Reducing IoT Service Delay via Fog Offloading. *IEEE Internet of Things Journal*, 5(2):998–1010, April 2018.

[30] D. Zhang, Y. Qiao, L. She, R. Shen, J. Ren, and Y. Zhang. Two time-scale resource management for green internet of things networks. *IEEE Internet of Things Journal*, pages 1–1, 2018.

[31] D. Zhang, R. Shen, J. Ren, and Y. Zhang. Delay-optimal proactive service framework for block-stream as a service. *IEEE Wireless Communications Letters*, pages 1–1, 2018.

[32] Q. Zhang, Q. Zhu, M. F. Zhani, R. Boutaba, and J. L. Hellerstein. Dynamic Service Placement in Geographically Distributed Clouds. *IEEE Journal on Selected Areas in Communications*, 31(12):762–772, December 2013.

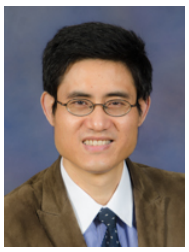
[33] X. Zhang, H. Yin, D. O. Wu, G. Min, H. Huang, and Y. Zhang. SSL: A Surrogate-based Method for Large-scale Statistical Latency Measurement. *IEEE Transactions on Services Computing*, pages 1–1, 2017.

[34] J. Zhou, T. Leppanen, E. Harjula, M. Ylianttila, T. Ojala, C. Yu, and H. Jin. Cloudthings: A common architecture for integrating the internet of things with cloud computing. In *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 651–657, June 2013.

Xu Zhang is an Associate Professor in the School of Electronic Science and Engineering, Nanjing University, China. He received the B.Sc. degree in communication engineering from Beijing University of Posts and Telecommunications, China, in 2012, and the Ph.D. degree from the Department of Computer Science and Technology at Tsinghua University, China, in 2017. He was a visiting student at the Department of Electrical and Computer Engineering, University of Florida, USA, from 2015 to 2016. His research interests include content delivery networks, network measurement and cloud gaming.

Haojun Huang is an associate professor in Communication Engineering within the School of Electronic Information and Communications at Huazhong University of Science and Technology. He received his BS degree in School of Computer Science and Technology from Wuhan University of Technology in 2005, and his PhD degree in School of Communication and Information Engineering from the University of Electronic Science and Technology of China in 2012. He worked as a postdoctor in the Research Institute of Information Technology at Tsinghua University from 2012 to 2015. His current research interests include wireless communication, Ad Hoc networks, Big Data, and Software-Defined Networking.

Hao Yin is a Professor in the Research Institute of Information Technology (RIIT) at Tsinghua University. He received the B.S., M.E., and Ph.D. degrees from Huazhong University of Science and Technology, Wuhan, China, in 1996, 1999, and 2002,respectively, all in electrical engineering. He was elected as the New Century Excellent Talent of the Chinese Ministry of Education in 2009, and won the Chinese National Science Foundation for Excellent Young Scholars in 2012. His research interests span broad aspects of Multimedia Communication and Computer Networks.



Dapeng Oliver Wu (S'98–M'04–SM'06–F'13) received a Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 2003. He is a professor at the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL. His research interests are in the areas of networking, communications, signal processing, computer vision, machine learning, smart grid, and information and network security. Currently, he serves as Editor in Chief of IEEE Transactions on Network Science and Engineering, and Associate Editor of IEEE Transactions on Communications.

Geyong Min is a Professor of High Performance Computing and Networking in the Department of Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. He received the PhD degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the B.Sc. degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests include Future Internet, Computer Networks, Wireless Communications, Multimedia Systems, Information Security, High Performance Computing, Ubiquitous Computing, Modelling and Performance Engineering.

Zhan Ma (S'06–M'11) received the B.S. and M.S. degrees from Huazhong University of Science and Technology, Wuhan, China, in 2004 and 2006, respectively, and the Ph.D. degree from the Tandon School of Engineering of New York University (formerly Polytechnic University, Brooklyn, NY, USA), New York, NY, USA, in 2011. He is currently on the faculty of Electronic Science and Engineering School, Nanjing University, Nanjing, China. From 2011 to 2014, he was with Samsung Research America, Dallas, TX, USA, and Futurewei Technologies, Inc., Santa Clara, CA, USA, respectively. His current research interests include the video compression, gigapixel streaming, and multispectral signal processing.