Towards QoS-aware Fog Service Placement

Olena Skarlat*, Matteo Nardelli†, Stefan Schulte* and Schahram Dustdar*
*Distributed Systems Group, TU Wien, Austria
Email: {o.skarlat, s.schulte, s.dustdar}@infosys.tuwien.ac.at
†Department of Civil Engineering and Computer Science Engineering, University of Rome Tor Vergata, Italy
Email: nardelli@ing.uniroma2.it

***Abstract*—Fog computing provides a decentralized approach to data processing and resource provisioning in the Internet of Things (IoT). Particular challenges of adopting fog-based computational resources are the adherence to geographical distribution of IoT data sources, the delay sensitivity of IoT services, and the potentially very large amounts of data emitted and consumed by IoT devices.**

Despite existing foundations, research on fog computing is still at its very beginning. A major research question is how to exploit the ubiquitous presence of small and cheap computing devices at the edge of the network in order to successfully execute IoT services. Therefore, in this paper, we study the placement of IoT services on fog resources, taking into account their QoS requirements. We show that our optimization model prevents QoS violations and leads to 35% less cost of execution if compared to a purely cloud-based approach.

1. Introduction

The integration of the Internet of Things (IoT) and information systems which support various industrial and private domains, e.g., in smart cities [1] or in smart factories [2], creates new requirements for system infrastructures [3]. Currently, a vast amount of IoT data emitted by distributed devices are sent to the cloud for centralized processing, and afterwards are sent back from the cloud to data consumers, which are very often located near the initial data sources. This leads to high delays and considerable cost for the usage of cloud-based computational resources [4]. To prevent this, the decentralized processing of IoT data has been identified as a promising approach [5].

As a matter of fact, IoT devices (e.g., gateways, sensors, or embedded systems) offer computational, storage, and networking resources [6]. The presence of these resources allows to move the execution of IoT applications to the edge of the network [7]. This approach is known as *fog computing* (or *edge computing*) [4].

To enable fog computing and use available IoT-based resources, those resources need to be virtualized. Also, it is necessary to allow the distributed execution of applications on different devices, networks, and domains, e.g., using the *Distributed Data Flow* model [8]. Notably, applications should be able to run in both the fog and the cloud. Especially in IoT scenarios, a *sense-process-actuate* application model is needed [9]. This model describes IoT applications where sensors emit data, which are then processed to activate actuators that perform necessary actions. Applications following this model consist of a set of services, which interact in a sequential way.

Existing conceptual approaches to realize fog computing deal with basic fog architectures, Application Programming Interfaces (APIs), and data communication [5], [7]. In contrast, the question of how to effectively distribute IoT services (e.g., following the sense-process-actuate model) among fog resources has gained very little attention so far [7], [10]. Therefore, within this paper, we propose an approach on how to optimally place IoT services on fog resources.

For this purpose, we use the notion of *fog colonies* (see Figure 1) [10], which act as micro data centers constructed from an arbitrary number of *fog cells*. Fog cells are software components running on IoT devices, which control and monitor virtualized resources of these devices. Within a fog colony, a *control node* performs resource provisioning by orchestrating fog cells, and by communicating with other colonies if additional resources are needed. Similarly to data centers in the cloud, fog colonies share services between their fog cells and between each other. Fog colonies are connected to a *fog computing management system* that is a middleware running in the cloud. Such a system provides additional resources if needed, performs reconfigurations of resources, and optimizes resources. We call an environment that is made up from fog cells, fog colonies, and a fog computing management system a *fog landscape*.

Based on this system model, we are able to establish an approach for the optimal sharing of resources among IoT services. We define a formal system model for fog landscapes. Afterwards, we present the Fog Service Placement Problem (FSPP), which allows to place IoT services on virtualized fog resources while taking into account Quality of Service (QoS) constraints like deadlines on the execution time of applications. Solutions to the FSPP describe execution plans in terms of resource provisioning and service placement. We implement the FSPP as an Integer Linear Programming problem, solve it using the CPLEX solver, and evaluate the placement solutions in terms of the cost of execution and QoS adherence.

The remainder of this paper is organized as follows:

978-1-5090-3047-7/17 $31.00 © 2017 IEEE
DOI 10.1109/ICFEC.2017.12

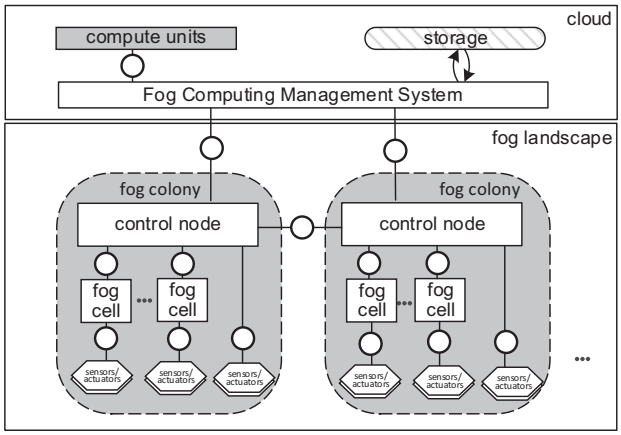

Figure 1. High-level View on Fog Computing Architecture

First, we elaborate on the problem statement and according resource and application models in Section 2. Next, in Section 3, we formalize the envisioned optimization model, i.e., the FSPP. We evaluate the model extensively and discuss results in Section 4. In Section 5 we elaborate on the state-of-the-art work in the area of the fog computing and resource provisioning. Finally, we conclude the paper and highlight our future work.

2. Problem Statement

In order to determine QoS-aware fog service placement, we need a suitable representation of the involved entities. For this, we provide a model of an IoT application which comprises the related set of IoT services, as well as QoS requirements of the application in Section 2.1. Then, in Section 2.2, we describe computational resources which collectively compose the fog landscape and the communication links between them. Finally, in Section 2.3, we formulate the FSPP as a mapping between applications and resources that takes into account QoS requirements. For the sake of clarity, in Table 1 we summarize the notation used throughout the paper.

2.1. Application Model

An IoT application consists of several services, which run on virtualized IoT devices (i.e., fog cells) and interact with each other to provide a well-defined functionality. Hence, an application is distributed in a fog landscape by placing the services onto particular fog cells. Let $A = \cup_{i=1}^{n} A_i$ be a set of applications to be placed in a fog landscape. Each application consists of a set of services. A service $a \in A_i$ is characterized by its demands for computational resources, i.e., CPU c_a^C, RAM c_a^M, storage c_a^S, and by a service type T_a. The type T_a specifies the requirement of a service to be executed on a specific kind of resource, e.g., which is equipped with a specific sensor. Without loss of generality, we consider three types of services: sensing, processing, and actuating services. A service has an estimated makespan duration m_a.

An application A_i is characterized by a user-defined deadline D_{A_i} which defines the maximum amount of time allowed for the application execution. The parameter w_{A_i} keeps record of the current deployment time for an application A_i. The application response time r_{A_i}, with $A_i \in A$, depends on the makespans of its services $a \in A_i$, on the communication delays among the services, and on the deployment time. The deployment time w_{A_i}, with $A_i \in A$, comprises the time interval needed to compute and enact the fog service placement.

2.2. Resource Model

A *fog cell* f is a virtualized single IoT device coordinating a group of other IoT devices, i.e., sensors and actuators. Fog cells run on IoT devices with computational, network, and storage capacities, e.g., gateways or Cyber-Physical Systems (CPS), while "pure" sensors and actuators are IoT devices without such resources. A fog cell is a software environment that provides access to, control of, and monitoring of an underlying physical IoT device. Computational resources of a fog cell are less powerful than those provided by the cloud. A fog cell is characterized by its capacities CPU C_f^C, RAM C_f^M, and storage C_f^S. There is no direct communication between fog cells, instead fog cells communicate via the control node in the colony.

Fog cells are organized in a fog colony. Each colony is identified and managed by its *control node* F. We denote $Res(F)$ the set of fog cells which are managed by F. A colony acts as a micro data center with resources scattered in a certain geographical area, i.e., each fog cell belongs to the resources of a colony $f \in Res(F)$. The communication link between a fog cell f and a control node F is characterized by a not negligible delay. Under the assumption of symmetric links and considering that a fog cell f can communicate only with the control node F of its own colony, we denote d_f as the communication link delay between f and F with $f \in Res(F)$. Colonies communicate with each other via their control nodes.

A control node is a more powerful fog cell, but, unlike a fog cell, it does not directly manage IoT devices (see Figure 1). An example of such a control node can be a proxy server or a gateway. Usually, a control node F is supplemented with faster and more expensive computational resources than resources of fog cells, however, at the same time, slower and cheaper than cloud resources. F is characterized by its capacities CPU C_F^C, RAM C_F^M, and storage C_F^S. Each control node has to perform resource provisioning by exploiting the computational resources of the colony for the placement of services. Apart from resource provisioning, the purposes of control nodes are (i) to perform infrastructural changes in their fog colonies, (ii) to analyze resource utilization in the colonies, and (iii) to monitor fog cells.

A cloud R has theoretically unlimited resources. The interaction between control nodes of colonies and the cloud

TABLE 1. PROBLEM NOTATION

	Parameter	Description
Time	t	Current time
	τ	Interval between two solutions of the FSPP (in sec)
Applications	A	Set of applications to be executed
	A_i	Application
	D_{A_i}	Deadline for an application (in sec)
	w_{A_i}	Current deployment time of an application (in sec)
	r_{A_i}	Response time of an application (in sec)
	a	Service in an application
	c_a^C	CPU demand of a service (in MIPS)
	c_a^M	RAM demand of a service (in MB)
	c_a^S	Storage demand of a service (in MB)
	m_a	Makespan of a service (in sec)
	T_a	Type of a service
Fog Landscape	R	Cloud
	F	Control node
	N	Closest neighbor control node
	C_F^C	CPU capacity of F (in MIPS)
	C_F^M	RAM capacity of F (in MB)
	C_F^S	Storage capacity of F (in MB)
	$Res(F)$	Fog cells in a colony
	f	Fog cell
	C_f^C	CPU capacity of f (in MIPS)
	C_f^M	RAM capacity of f (in MB)
	C_f^S	Storage capacity of f (in MB)
	$Res^a(F)$	Fog cells that can host a service a
	d_f	Link delay between F and f (in sec)
	d_R	Link delay between F and R (in sec)
	d_N	Link delay between F and N (in sec)

is performed via a *fog computing management system*. The logical link between a control node F and the cloud R has a not negligible delay d_R. A fog computing management system is a central unit that manages the execution of services in the cloud and supports the underlying fog landscape. Importantly, a fog computing management system is able to overrule control nodes in fog colonies, but the latter may also act autonomously in the case that no fog computing management system is available.

2.3. Fog Service Placement Problem

The FSPP aims to determine an optimal mapping between IoT applications and computational resources with the objective of optimizing the fog landscape utilization while satisfying QoS requirements of applications. In the FSPP, we consider a proactive scenario of service placement, where the placement of services is calculated and applied periodically. Such a proactive scenario fits the volatility of the IoT (with regard to data to be processed and also with regard to IoT devices leaving and entering the system) better than static resource provisioning approaches [11]. To account for QoS requirements of applications, i.e., deadlines, historical data about deployment time in fog colonies has to be preserved. This data allows to receive an upper bound (i.e., worst-case estimation) of the response time of applications and to prioritize applications which are closer to the deadline.

We define the FSPP as a decentralized optimization problem, i.e., when an IoT application request is submitted to a control node, the latter solves an instance of the FSPP and performs the service placement. For that, every control node considers a local view on the colony and accounts for the closest neighbor colony and the cloud. The control node F places the application services on the computational resources available in its colony (i.e., fog cells, the control node itself). If there are not enough resources, it sends the services either to the closest neighbor colony, or to the cloud. It is a matter of future work to find mechanisms not only to choose the closest neighbor colony, but also to account for the most effective colony among the neighbors. With respect to the control node F, we denote N as the control node of the closest neighbor colony and d_N as the communication link delay between F and N.

Specifically, the control node F manages fog cells which are resources of the colony $Res(F)$. Recalling the application model, we observe that a service $a \in A_i$ cannot be placed on every fog cell $f \in Res(F)$ because of its type T_a that may require specific resources (e.g., sensors). Therefore, for each $a \in A_i$ we consider a subset of fog cells, where it can be deployed, i.e., $Res^a(F) \subseteq Res(F)$.

To sum up, F determines the following subsets of placements: (1) Which services have to be executed in its colony? (2) Which services have to be executed locally on F's own resources? (3) Which services have to be propagated to the closest neighbor colony N? (4) Which services have to be propagated to the cloud R? Afterwards, the control node is able to perform placement of services on the identified fog resources. The concrete model on how services are placed on fog resources is presented in the next section.

3. Optimization Model

The FSPP is solved by the control node F periodically, every τ time units, i.e., seconds. The application requests submitted between $t-\tau$ and t are scheduled for placement in t. We provide a formulation of the FSPP considering t as the current time. After defining the preliminaries and problem notation, we formulate the FSPP as an Integer Linear Programming problem.

The variables of the optimization model are binary and indicate a placement of a specific service on specific fog resources. The binary variable $x_{a,f}$ indicates whether a service a is placed on a fog cell f in the current colony. y_a defines that a service a is placed on the control node. The binary variable z_a defines whether a service a is propagated to the cloud, and n_a indicates whether a service a is propagated to the closest neighbor colony. Variables $x_{a,f}$ implicitly account for the types of services and types of IoT devices through $Res^a(F)$. Analogously, variables y_a and n_a indicate that the control node and the neighbor control node N can host a service a.

The goal function of the optimization model is expressed by (1) and maximizes the utilization of the fog landscape

while satisfying application requirements.

$$\max \sum_{A_i}^{A} \Big(\frac{1}{D_{A_i} - w_{A_i}} \cdot \cdot \sum_{a}^{A_i} \Big(\sum_{f}^{Res^a(F)} x_{a,f} + y_a + n_a \Big) \Big) \quad (1)$$

To account for deadlines, the goal function prioritizes those applications which have a closer deadline. For that, we use the coefficient $\frac{1}{D_{A_i} - w_{A_i}}$ which depends on the deadline and the deployment time of an application. This coefficient becomes more influential when the deadline is close or the deployment time of the application is significant.

As a first constraint, each service a has to be placed either in the fog or in the cloud:

$$\sum_{f}^{Res^a(F)} x_{a,f} + y_a + z_a + n_a = 1, \forall a \in A_i, \forall A_i \in A \quad (2)$$

Second, placed services must not exceed the capacities of CPU, RAM, and storage of a corresponding fog resource. The equations (3)-(4) guarantee that the services placed on fog cells or on the control node do not exceed a given percentage μ of available resources.

$$\sum_{a}^{A_i} c_a^\gamma \cdot x_{a,f} \leq \mu C_f^\gamma, \gamma = \{C, M, S\}, \forall f \in Res(F) \quad (3)$$

$$\sum_{a}^{A_i} c_a^\gamma \cdot y_a \leq \mu C_F^\gamma, \gamma = \{C, M, S\} \quad (4)$$

As a third constraint, we account for the response time r_{A_i} of an application, so that it does not violate the application deadline as in (5).

$$r_{A_i} \leq D_{A_i}, \forall A_i \in A \quad (5)$$

The response time has to account for the time until the next optimization period and for an average deployment time of services in the closest neighbor colony. The calculation of r_{A_i} is done from the central point of view of the control node in the colony. The application response time r_{A_i} is defined in (6) and depends on the overall makespan m_{A_i} and the overall deployment time w_{A_i}.

$$r_{A_i} = m_{A_i} + w_{A_i}, \forall A_i \in A \quad (6)$$

The overall makespan m_{A_i} accounts for the time needed to transfer services to computational resources, execute them, and retrieve the results, considering that the communications are coordinated by the control node. The makespan m_{A_i} depends on the execution time of each service according to its placement as in (7)–(11). We define m_{A_i} as:

$$m_{A_i} = \sum_{a}^{A_i} \Big(\sum_{f}^{Res^a(F)} d(a,f) \cdot x_{a,f} + d(a,F) \cdot y_a + + d(a,R) \cdot z_a + d(a,N) \cdot n_a \Big) \quad (7)$$

where $d(a,f)$, $d(a,F)$, $d(a,R)$, and $d(a,N)$ represent the makespan of a when it is executed on the fog cell f, the control node F, the cloud R, and the neighbor colony N respectively. These variables are defined in 8-11:

$$d(a,f) = 2 \cdot d_f + m_a, \forall a \in A_i, \forall f \in Res^a(f) \quad (8)$$

$$d(a,F) = m_a, \forall a \in A_i \quad (9)$$

$$d(a,R) = 2 \cdot d_R + m_a, \forall a \in A_i \quad (10)$$

$$d(a,N) = 2 \cdot d_N + m_a, \forall a \in A_i \quad (11)$$

The overall deployment time w_{A_i} depends on the already experienced deployment time at the time $t - \tau$, and the possibly additional time if any $a \in A_i$ is propagated to the closest neighbor colony for execution. We define w_{A_i} in 12:

$$w_{A_i} = w_{A_i}^\tau + \tau \cdot n_{A_i} + \overline{T}_{w_N} \cdot n_{A_i} \quad (12)$$

where the first term $w_{A_i}^\tau$ is an already experienced deployment time of A_i, the second term $n_{A_i} \cdot \tau$ considers that if a service is propagated to the closest neighbor colony, the placement will be performed in the next round of the FSPP, i.e., the application has to wait at least τ time units before its execution. The term $\overline{T}_{w_N} \cdot n_{A_i}$ models the additional deployment time spent in the closest neighbor colony before the application execution. The calculation of the real value of average deployment time in the closest neighbor colony $\overline{T}_{w_N}$ requires to look ahead in time. To simplify the formulation, we rely on the estimation of $\overline{T}_{w_N}$ based on historical data. To this end, we define $\overline{T}_{w_N}$ as a moving average of parameter α of T_{w_N}, i.e., the sampled deployment time per each service propagated to N:

$$\overline{T}_{w_N} = \alpha T_{w_N} + (1 - \alpha) \overline{T}_{w_N}^\tau, \alpha \in [0,1] \quad (13)$$

where $\overline{T}_{w_N}^\tau$ is the average deployment time in the closest neighbor colony in the previous round of FSPP.

Finally, we introduce variables n_{A_i} that charge a supplementary deployment time per application A_i if at least one of its services $a \in A_i$ is propagated to the closest neighbor colony N given that $|A_i|$ is the cardinality of A_i:

$$n_{A_i} \leq \sum_{a}^{A_i} n_a \quad (14)$$

$$n_{A_i} \geq \frac{\sum_{a}^{A_i} n_a}{|A_i|} \quad (15)$$

Equations (16)–(20) specify the domain definitions for the optimization model:

$$x_{a,f} \in \{0,1\},\ \forall a \in A_i,\ \forall A_i \in A,\ \forall f \in Res^a(F) \quad (16)$$

$$y_a \in \{0,1\},\ \forall a \in A_i,\ \forall A_i \in A \quad (17)$$

$$z_a \in \{0,1\},\ \forall a \in A_i,\ \forall A_i \in A \quad (18)$$

$$n_a \in \{0,1\},\ \forall a \in A_i,\ \forall A_i \in A \quad (19)$$

$$n_{A_i} \in \{0,1\},\ \forall A_i \in A \quad (20)$$

4. Evaluation

In the following evaluation, we show the benefits of the FSPP in terms of processing cost, response times, and deadline violations with respect to a baseline approach and to the execution in the cloud. For this, we use and extend *iFogSim*, which is a modeling and simulation toolkit for IoT and fog computing environments [9].

4.1. Evaluation Environment

We extend iFogSim with the means to account for the structure of the fog landscape introduced in Section 2. The *Application* and *AppModule* classes are modified to account for deadlines and deployment times of applications. The *FogDevice* class is extended by *FogLandscapeDevice* to act as a fog cell and a control node. The *ModulePlacement-Optimization* API is defined to ease the implementation of various resource provisioning approaches, to unify the access to the simulation environment, and to interpret and activate the obtained placement results. The baseline scenario is represented by the *ModulePlacementFirstFit* class, which implements the API and performs service placement by searching a *first fit* fog resource [10].

The optimization model is implemented by means of the IBM CPLEX solver[1] and the open source Java ILP library[2]. The implemented *FogSolverCPLEX* class performs the mapping from Java ILP to IBM CPLEX. The *Module-PlacementExact* class implements the API and solves the model. It has to be noted that in order to use IBM CPLEX to solve the described optimization problem, the constraints are transformed to their canonical form.

[1]https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

[2]https://sourceforge.net/projects/javailp/

TABLE 2. APPLICATION DETAILS

Services	c_a^C (MIPS)	c_a^M (MB)	m_a (sec)
Application A_1, D_{A_1} = 300 sec, w_{A_1} = 0 sec			
SenseMotion	50	30	0.10
ProcessMotion1	100	10	0.10
ProcessMotion2	100	20	0.10
ProcessMotion3	200	30	0.08
ActuateMotion	50	20	0.80
Application A_2, D_{A_2} = 240 sec, w_{A_2} = 60 sec			
SenseVideo	50	30	0.90
ProcessVideo1	200	100	0.10
ProcessVideo2	200	20	0.10
ProcessVideo3	100	30	0.25
ActuateVideo	50	20	0.50
Application A_3, D_{A_3} = 360 sec, w_{A_3} = 60 sec			
SenseSound	50	30	0.40
ProcessSound1	100	10	0.10
ProcessSound2	200	20	0.07
ProcessSound3	200	30	0.35
ActuateSound	50	20	0.40
Application A_4, D_{A_4} = 360 sec, w_{A_4} = 0 sec			
SenseTemperature	50	30	0.40
ProcessTemperature1	200	10	0.70
ProcessTemperature2	200	20	0.10
ProcessTemperature3	200	30	0.90
ActuateTemperature	50	20	0.30

4.2. Experimental Setup

To evaluate the efficiency of the FSPP, we apply a first fit placement (called the "Baseline" scenario) and the optimization model of the FSPP as introduced in Section 3 (called the "Optimization" scenario) in the fog landscape. Furthermore, we perform an execution of all applications in the cloud for the case the fog landscape is not available (called the "Cloud" scenario).

Four different sense-process-actuate application requests are submitted for execution in the fog colony. Each fog cell is connected to four different sensors and actuators corresponding to each of the applications, i.e., motion, video, sound, and temperature sensors and actuators. Each application consists of several services. A summary of the setup is provided in Table 2.

As a fog landscape, we consider several fog colonies. One colony is the main colony, where service placements are observed. From the remaining neighbor colonies, the closest neighbor is identified according to the least communication link delay (see Section 2). The control node can host processing and actuating services. The neighbor colony can host only processing services, and the cloud can host all services. The communication link delay between the control node and the cloud is set to 2 seconds. The cost per processing in the cloud is set to $ 0.30 per Billing Time Unit (BTU), i.e., one hour. The communication link delay between the control node and the neighbor is set to 0.5 seconds. The average deployment time of applications in the neighbor colony is set to 3 minutes, the average deployment time of applications in the neighbor colony of the previous optimization period is set to 2 minutes. A control node in a colony has 500 MIPS of CPU power, 512 MB of RAM, and 8 GB of storage. 10 fog cells are connected to the control

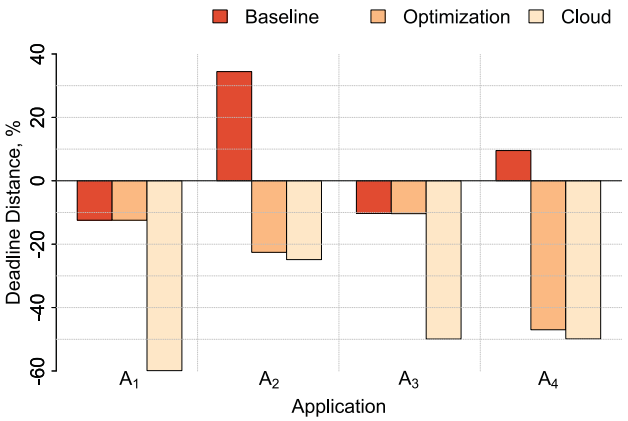

Figure 2. Deadline Distance

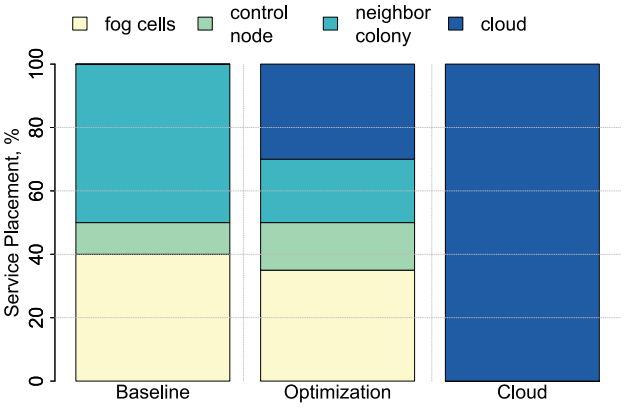

Figure 3. Service Placement

node. These fog cells are of less computational and storage power than the control node, i.e., 250 MIPS of CPU power, 256 MB of RAM, and 4 GB of storage. These fog cells can host sensing and actuating services. The communication link delay between fog cells and the control node is set to 0.5 seconds.

4.3. Discussion

This experiment aims to show how the model behaves depending on various deadlines and deployment times of the given applications. In the Baseline scenario, for applications A_1, A_3, and A_4 sensing and actuating services are placed on different fog cells, and processing services are propagated to the closest neighbor colony. For application A_2, two processing services are placed on the control node, and one processing service is propagated to the neighbor colony. In the course of experiment, the Baseline scenario results in deadline violations for A_2 and A_4 by 82.65 and 22.87 seconds respectively.

In the Optimization scenario, if it is not advantageous for an application to wait for the deployment in the closest

TABLE 3. SCENARIO COMPARISON

Metrics	Baseline	Optimization	Cloud
Application A_1, D_{A_1} = 300 sec, w_{A_1} = 0 sec			
Response time (sec)	262.66	262.66	120.31
Deadline distance (sec)	-37.34	-37.34	-179.69
Application A_2, D_{A_2} = 240 sec, w_{A_2} = 60 sec			
Response time (sec)	322.65	185.82	180.31
Deadline distance (sec)	82.65	-54.18	-59.69
Application A_3, D_{A_3} = 360 sec, w_{A_3} = 60 sec			
Response time (sec)	322.90	322.66	180.31
Deadline distance (sec)	-37.10	-37.34	-179.69
Application A_4, D_{A_4} = 360 sec, w_{A_4} = 0 sec			
Response time (sec)	262.87	126.24	120.31
Deadline distance (sec)	22.87	-113.76	-119.69
Service placement on resources, (% of all services)			
Fog cells	40	35	00
Control node	10	15	00
Neighbor node	50	20	00
Cloud	00	30	100
Cost of execution ($)	0.000	0.069	0.107

neighbor colony because of the close deadline, the optimization model places the services in the cloud. If the deadline is not critically close, and the estimated response time of the application conforms to the constraints of the optimization model, the services are propagated to the closest neighbor colony. For application A_2, the optimization model places the sensing service on the fog cell, two processing and one actuating services onto the control node, and one processing service in the cloud. The placement of any service from A_2 to the closest neighbor colony would violate the deadline because of the additional deployment time that appears in the neighbor colony. In the case of A_4, the sensing and actuating services are placed on one fog cell and three services are propagated to the cloud because the control node is already busy with A_2. For A_1 and A_3, sensing and actuating services are placed on separate fog cells, and processing services are propagated to the closest neighbor colony. To summarize, the service placement in the Optimization scenario depends on the time interval τ between two subsequent FSPP optimizations in the system and on the average deployment time $\overline{T}_{w_N}$ in the closest neighbor colony, and the solution of the FSPP of the Optimization scenario circumvent deadline violations of the applications. If τ and $\overline{T}_{w_N}$ are small enough, the FSPP optimization model propagates the services to the closest neighbor colony. The deadline distances for the three scenarios are depicted in Figure 2.

The utilization of the fog landscape in the three scenarios is shown in Figure 3. In the Baseline scenario, services are placed only on fog resources, i.e, 40% of services are placed on fog cells, 10% of services are placed on the control node itself, and 50% services are propagated to the closest neighbor colony. In the Optimization scenario, these percentages are respectively 35%, 15% and 20%, and also 30% of all services are propagated to the cloud. As a result, the cloud resource is utilized only by 30%. The use of the control node's own resources is performed more efficiently in the Optimization scenario. More services with

lesser resource demands are placed on the control node rather than less services with bigger demands compared to the Baseline scenario. Such behavior is provided by the prioritization of applications in the goal function according to the deadlines and deployment times, and strict adherence to the deadlines in the constraints of the optimization model.

As discussed in Section 1, we assume ownership of the fog, hence the cost of execution in the fog landscape can be neglected. The cost of execution in the cloud is calculated as the sum of cost per processing second in the cloud of each service divided by the amount of seconds in 1 BTU. In the Optimization scenario, services are propagated either to the closest neighbor colony, or to the cloud, yet at the same time satisfying deadlines of applications. Therefore, execution cost accrue. These cost are around and it is around 35% less than the cost of execution in the Cloud scenario, yet the Optimization scenario still satisfies the application deadlines. A summary of all evaluation results is shown in Table 3.

FSPP Computational Time Additionally, we conducted another experiment to calculate the *computational time* of producing a solution of the FSPP depending on the amount of services to be placed. The amount of services affects the amount of decision variables and constraints in the model, and therefore, influences the computational time. In Figure 4, the distribution of the computational time is shown based on minimum and maximum values, first and third quartiles, and the median of the computational times. The individual outlying points of computational times are displayed as unfilled circles in the first two boxes. The measurements of computational time show that FSPP is solved in less than a second even in the case of submitted applications with large amounts of services.

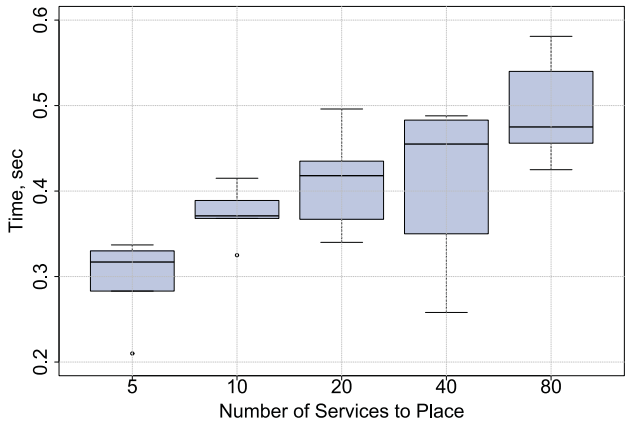

Figure 4. Computational Time of FSPP

5. Related Work

As a matter of fact, there are only few resource provisioning approaches specifically aiming at fog computing. However, there is some work in the area of resource provisioning in distributed environments which needs to be discussed. Resource provisioning is an extensively researched topic in distributed systems, e.g., for cloud computing [12], [13] and mobile cloud computing [14], [15]. These works describe resource provisioning approaches, yet cannot be directly adopted for the usage in fog computing, since fog landscapes are usually more volatile than cloud environments and the number of nodes necessary to provide a particular service is usually larger.

Having in mind that storage, computational resources, and network demands are volatile, a resource provisioning approach for fog computing has to account for peak resource demands and utilization gaps. The basis for such a smart resource provisioning that eliminates over-provisioning lays in the characteristic of the elasticity of the cloud, i.e., minimizing cost, latency and delays, while maximizing QoS. Some work over the elasticity in the cloud has been introduced [12], however, fog computing environments have not been considered yet. Mobile cloud computing is mostly based on a rather simple network topology with direct communication between mobile devices and the cloud. Therefore, the according resource provisioning approaches offer interesting insights and ideas, but cannot be directly applied to fog computing.

In the field of fog computing, Aazam and Huh [16] propose an approach for the prediction of future resource demands by means of historical access data and pricing models. In another paper, the authors propose an improvement of the theoretical resource management model in terms of specifications of utilization and QoS in the context of multimedia IoT devices [17]. However, the proposed approaches are not reactive to dynamic changes in the fog landscape. In the work by Hong et al. [18] a high level programming model is presented, including APIs and a simple resource provisioning approach based on utility thresholds of fog resources. In the work of Saurez et al. [19] the approach of Hong et al. is extended by implementing the envisioned APIs and adding algorithms regarding the discovery, migration, and deployment of fog cells and services. Furthermore, an experimental real-world fog landscape was set up using docker containers deployed on several servers.

Urgaonkar et al. [20] present an approach on how to distribute services between edge clouds, which resemble our notion of fog colonies. The placement of services onto resources is formulated as a Markov Decision Problem and solved by means of a control algorithm based on Lyapunov optimization, minimizing the cost of execution and accounting for delays and location in constraints. In contrast, in our work, fog resource and application models are explicitly formulated. The fog landscape is considered as an extension of the cloud, and the optimization model takes into account QoS metrics of applications.

In our former work, a simple optimization problem for resource provisioning in fog landscape is introduced [10]. The optimization model maximizes the utilization of the fog landscape and minimizes link delays. Compared to our former work, in this current paper the optimization model is now considerably enhanced. Apart from the maximization

of the utilization of the fog landscape, we take into account application QoS metrics, i.e., deadlines of applications.

6. Conclusions

Fog computing provides means to utilize available resources close to the edge of the network. A particular research challenge is the optimal resource provisioning and service placement.

After having motivated our work, we provide a model for an IoT application and a resource model for the fog landscape. Afterwards, the FSPP is described, and an according optimization model is formalized. The experimental evaluation has shown that the FSPP utilizes the fog landscape for 70% of services, thus reducing the execution cost by about 35% with respect to the execution in the cloud. The solution of the FSPP does not violate deadlines of applications unlike the solution of the baseline approach.

In our future work, we plan to improve the optimization model by adding constraints about the availability of resources, the reliability of services, and the cost of resources. Another challenge is to not only find the closest neighbor colony in a fog landscape for service propagation, but also to find the most efficient neighbor colony. Last but not least, it is necessary to obtain realistic network data, e.g., communication link delays, for service placement evaluations in a fog landscape. For this, we currently develop a real-world testbed.

Acknowledgments

This paper is supported by TU Wien research funds. This work is partially supported by the Commission of the European Union within the CREMA H2020-RIA project (Grant agreement no. 637066).

The authors thank IBM Academic Initiative for providing an academic license for IBM CPLEX.

References

[1] M. Vögler, J. M. Schleicher, C. Inzinger, and S. Dustdar, "A Scalable Framework for Provisioning Large-scale IoT Deployments," *ACM Transactions on Internet Technology*, vol. 16, no. 2, 2016.

[2] S. Schulte, P. Hoenisch, C. Hochreiner, S. Dustdar, M. Klusch, and D. Schuller, "Towards Process Support for Cloud Manufacturing," in *18th IEEE International Enterprise Distributed Object Computing Conference*. IEEE, 2014, pp. 142–149.

[3] A. Botta, W. de Donato, V. Persico, and A. Pescape, "Integration of Cloud Computing and Internet of Things: a Survey," *Future Generation Computer Systems*, vol. 56, pp. 684–700, 2016.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in *MCC Workshop on Mobile Cloud Computing*. ACM, 2012, pp. 13–16.

[5] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog Computing: A Platform for Internet of Things and Analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*, ser. Studies in Computational Intelligence. Springer, 2014, vol. 546, pp. 169–186.

[6] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, pp. 2787–2805, 2010.

[7] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, "Fog Computing: Principles, Architectures, and Applications," in *Internet of Things: Principles and Paradigms*. Morgan Kaufmann, 2016, ch. 4, pp. 61–75.

[8] N. K. Giang, M. Blackstock, R. Lea, and V. C. Leung, "Developing IoT Applications in the Fog: a Distributed Dataflow Approach," in *5th International Conference on the Internet of Things (IoT)*. IEEE, 2015, pp. 155–162.

[9] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments," Cloud Computing and Distributed Systems Laboratory, The University of Melbourne, Tech. Rep. CLOUDS-TR-2016-2, 2016. [Online]. Available: arXiv:1606.02007

[10] O. Skarlat, S. Schulte, M. Borkowski, and P. Leitner, "Resource Provisioning for IoT Services in the Fog," in *9th IEEE International Conference on Service Oriented Computing and Applications (SOCA 2016)*. IEEE, 2016, pp. 32–39.

[11] D. Georgakopoulos, P. P. Jayaraman, M. Fazia, M. Villari, and R. Ranjan, "Internet of Things and Edge Cloud Computing Roadmap for Manufacturing," *IEEE Cloud Computing*, vol. 3, no. 4, pp. 66–73, 2016.

[12] P. Hoenisch, I. Weber, S. Schulte, L. Zhu, and A. Fekete, "Four-fold Auto-Scaling on a Contemporary Deployment Platform using Docker Containers," in *13th International Conference on Service Oriented Computing (ICSOC 2015)*, ser. LNCS, vol. 9435. Springer, 2015, pp. 316–323.

[13] P. Leitner, W. Hummer, B. Satzger, C. Inzinger, and S. Dustdar, "Cost-Efficient and Application SLA-Aware Client Side Request Scheduling in an Infrastructure-as-a-Service Cloud," in *5th International Conference on Cloud Computing*. IEEE, 2012, pp. 213–220.

[14] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches," *Wireless Communications and Mobile Computing*, vol. 13, pp. 1587–1611, 2013.

[15] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generations Computer Systems*, vol. 29, pp. 84–106, 2013.

[16] M. Aazam and E.-N. Huh, "Fog Computing Micro Datacenter Based Dynamic Resource Estimation and Pricing Model for IoT," in *29th IEEE International Conference on Advanced Information Networking and Applications*. IEEE, 2015, pp. 687–694.

[17] M. Aazam, M. St-Hilaire, C.-H. Lung, and I. Lambadaris, "MeFoRE: QoE based resource estimation at Fog to enhance QoS in IoT," in *23rd International Conference on Telecommunications (ICT)*. IEEE, 2016, pp. 1–5.

[18] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwlder, and B. Koldehofe, "Mobile Fog: A Programming Model for LargeScale Applications on the Internet of Things," in *1st ACM SIGCOMM Workshop on Mobile Cloud Computing*. ACM, 2013, pp. 15–20.

[19] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, and B. Ottenwälder, "Incremental Deployment and Migration of Geo-distributed Situation Awareness Applications in the Fog," in *10th ACM International Conference on Distributed and Event-based Systems (DEBS'16)*. ACM, 2016, pp. 258–269.

[20] R. Urgaonkar, S. Wang, T. Hea, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Performance Evaluation*, vol. 91, pp. 205–228, 2015.