UNIVERSITY OF SOUTHAMPTON

Online Mechanism Design for Resource Allocation in Fog Computing

by

Fan Bi

A mini-thesis submitted for transfer from MPhil to PhD

Supervisors:
Doctor Sebastian Stein
Doctor Enrico Gerding
Professor Nicholas R. Jennings

Examiners:
Professor Geoff Merrett
Doctor Jie Zhang

June 2019

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

A progress report submitted for continuation towards a PhD

Fan Bi

Fog computing is becoming more and more popular as an appropriate computing paradigm for the Internet of Things (IoT). It is a virtualised platform that lies between IoT devices and centralised cloud computing. The characteristics of fog computing include closeness to the IoT devices, low latency, geo-distributed, large number of fog nodes, and real-time interaction. However, most existing resource allocation mechanisms for fog computing are not truthful, which means that users are not incentivised to always provide their true information. Hence, an efficient resource allocation paradigm for this computing paradigm, which can be used in a strategic environment is in need. To this end, we consider two challenges: (1) near-optimal resource allocatioin in a fog system; (2) incentivising self-interested fog users to report their tasks truthfully.

In this thesis, we examine relevant literature and describe its achievements and shortcomings. Currently, most of the resource allocation mechanisms proposed are for cloud computing or other resource allocation problems like electric vehicle (EV) charging. However, there is little work that studies truthful and online fog computing resource allocation mechanisms. Furthermore, they only address a subset of the challenges in our fog computing resource allocation problem.

Additionally, we design our resource allocation model in detail and choose the benchmark mechanisms for this model. We also develop an efficient and truthful mechanism called flexible online greedy and evaluated its performance by simulations, the simulations show that our mechanism can achieve better social welfare than other truthful benchmark mechanisms by up to 10% and achieves a social welfare of about 90% of the theoretical uppper bound. Finally, we outline possible future work and the plan for the next phase of work.

Contents

Nomenclature

$\Gamma(\mathcal{M})$	the game induced by mechanism $\mathcal{M}$ Γ_l^i
Θ_i	the set of possible types of agent i
Θ_{-i}	$\Theta_1 \times \cdots \times \Theta_{i-1} \times \Theta_{i+1} \times \cdots \times \Theta_n$
ϵ	the approximation error
λ_i	the resource allocation scheme for task i
$\kappa_{m,t}$	the load factor of resource m at time step t
μ	$\mu = 2MF + 2$
$\succ_i$	the preference of agent i
$\hat{\theta}^{\langle t \rangle}$	the set of all reported types until and including time t
θ_i	the type/private information of agent i
$\hat{\theta}_i$	the reported type/private information of agent i
A	the set of results
$A_{w,r}$	the capacity of type r resource at FN w
$C(\theta_i)$	the limited misreports that agent i can make
$\mathcal{E}$	a mechanism environment
E	the set of IoT devices
E_i	the set of IoT devices of task i
E_l	the set of IoT devices in location l
$\mathbb{E}$	the set of all data links in the fog
F	a constant in SWMOA mechanism
H^t	the set of mechanism states
I	$\{1, 2, \ldots, n\}$the set of tasks/agents
L	the set of locations
$\mathcal{M}$	a mechanism
M	the set of all resource types
N_t	the set of tasks running at time t
$Q(h^t)$	all possible decisions in state h^t
R	the set of computational resource types
S	the set of all possible vectors of strategies
S_i	the set of all strategies of agent i
S_{-i}	the set of all vectors of strategies expect agent i's
T	$\{1, 2, \ldots, t\}$ the set of discrete time steps

T_i^a	the arrival time of task i
T_i^d	the deadline of task i
T_i^s	the (earliest) start time of task i
$\hat{T}_i^a$	the reported arrival time of task i
$\hat{T}_i^f$	the reported deadline of task i
$\hat{T}_i^s$	the reported (earliest) start time of task i
W	the set of FNs
X_i	the set of actions for agent i
$\mathcal{Y}_i$	the set of interesting decisions for agent i
Y	an interesting decision
Z	the set of resources
a	the outcome function
$a_{i,r}$	the amount of resource r required by task i
$b_{j,k}$	the bandwidth capacity of link (j,k)
c_i	the virtual cost of task i
$c_{m,t}$	the virtual cost of resource m at time step t
d	$(d^1, d^2, \ldots)$ the sequence of decisions made by the mechanism
e	an IoT device
f	the social choice function
f^t	the decision made in time time step t
$f^i_{l,p,j,k,t}$	traffic from the location l of task i to FN w on link (j,k) at time step t (from node j to node k)
g_i	task i's valuation coefficient (if the valuation function is linear)
h^t	the mechanism state in time period t
i	one task/agent
k	resource capacity coefficient
l	one location of the fog
m	one type of resource
n	the total number of agents
o	the total operational cost
o_i	the total operational cost of task i
$o_{j,k}$	unit operational cost of link (j,k)
$o_{w,r}$	unit operational cost of resource r on FN w
p_i	the payment function for agent i
p_i^t	the payment for agent i in time step t
q	the sequence of decisions made over time
q^t	decision made in time period t
r	a type of computational resource
r_i	the value of agent i for any one of the decisions during the desired period
s	the vector of strategies played by all agents
s^*	the dominant strategy equilibrium
s_i	the strategy played by agent i

s_i^*	the dominant strategy for agent i
s_{-i}	the strategy vector of other players except agent i
t_i	usage time needed for task i
$\hat{t}_i$	reported usage time for task i
$\tilde{t}_i$	usage time allocated to task i
u_i	the utility of task i
$u_{e,l}^i$	whether IoT devices e of task i is at location l
v^c	critical value
v_i	the set of possible valuation functions for task i
v_i	task i's valuation function
$\hat{v}_i$	task i's reported valuation function
$v_{i,t}$	task i's valuation when it gets t usage time
w	one FN of the fog
x_i	the action taken by agent i
z	the number of items in a combinatorial auction
$z_{w,t}^i$	whether the VM of task i is placed in FN w at time step t

Acronyms

AI	Artificial intelligence
ADMM	Alternating direction method of multipliers
ANN	Artificial neural network
AR	Augmented reality
AV	Autonomous vehicle
AWS	Amazon Web Services
BB	Branch-and-bound
BoB	Branch-on-bids
BoI	Branch-on-items
CABoB	Combinatorial auction BoB
DSIC	Dominant-strategy incentive compatible
DSO	Data service operator
DSS	Data service subscriber
EC2	Elastic Compute Cloud
EV	Electric vehicle
FCFS	First-come-first-served
FlexOG	Flexible online greedy
FN	Fog node
IaaS	Infrastructure-as-a-Service
IoT	Internet of Things
IR	Individually rational
ISP	Internet service provider
LISP	Locator ID separation protocol
FN	Fog node
MINLP	Mixed-integer nonlinear programming
MIPv6	Mobile IPv6
MARA	Multiagent resource allocation
HMIPv6	Hierarchical Mobile IPv6
OG	Online greedy
PayEx	Pay externality
PTAS	Polynomial-time approximation scheme
PoA	Price of anarchy

QoE	Quality of Experience
QoS	Quality of Service
RAM	Random access memory
RACC	Resource allocation in cloud computing
RAFC	Resource allocation in fog computing
SDN	software-defined networking
SLA	Service-level agreement
SLS	Stochastic local search
PRMOA	Provider revenue maximization online auction
SWMOA	Social welfare maximisation online auction
VCG	Vickrey-Clarke-Groves
VC	Virtual cluster
VM	Virtual machine
VR	Virtual reality
WBB	Weak budget balance
WDP	Winner determination problem
WMON	Weak monotonicity

Chapter 1

Introduction

To extend the traditional Internet, which only connects computers and smartphones, the Internet of Things (IoT) is about connecting all kinds of physical devices in the world to the Internet, and we call those things IoT devices. The IoT is developing rapidly, and it is estimated that by 2025, there will be 22 billion active devices in the IoT (Lueth, 2018). The reason why the IoT is fast-developing is that it makes things smart by giving them the ability to receive and send information to the internet. IoT applications such as smart homes (Ricquebourg *et al.*, 2006), smart cities (Cocchia, 2014), industry 4.0 (Roblek *et al.*, 2016), smart agriculture (TongKe, 2013), and the smart grid (Tuballa and Abundo, 2016) can significantly improve work efficiency, make life more convenient and improve our health. For example, a smart city can have a traffic management system, which take advantage of widely distributed sensors such as video cameras, Bluetooth sensors (i.e., sensors that detects how many smartphones with their Bluetooth on near it.), and loop detectors (detect vehicles arriving or passing it) to reduces traffic congestions (Su *et al.*, 2011), and a smart home may allow a refrigerator to buy food automatically according to its stock (Ricquebourg *et al.*, 2006).

However, IoT devices usually have very limited computing power because they need to have extremely low costs and very low power consumption (Chen *et al.*, 2014). For instance, to reduce cost many IoT devices only use energy harvested from the environment, which is very limited. Many IoT devices must have low costs so that people are willing to buy and use them. For this reason, many IoT devices cannot process their computational tasks or store their data locally. For example, as a surveillance camera in a traffic management system generates a vast amount of video data every day, its limited storage will fill up quickly if it stores the data locally. Another example, the Amazon Echo, which is a smart speaker developed by Amazon, performs speech recognition by sending the voice to Amazon's servers because the Echo does not have enough computational power to perform this locally (Marr, 2018). Similarly, the data generated by weather sensors in smart agriculture need to be gathered for analysis, because weather

sensors can only collect and send weather data and overall data is needed to predict weather (Mekala and Viswanathan, 2017).

One way to solve this problem is by combining IoT with cloud computing (Chen *et al.*, 2014). For example, the cloud offers virtually unlimited computational power and storage from its data centres to IoT devices (Botta *et al.*, 2016). In more detail, the data centre in the cloud is mainly a collection of computing facilities (e.g., servers, routers and switches) that is used to create cloud services (Greenberg *et al.*, 2008). However, this approach also has three main deficiencies. Firstly, cloud computing often has a high latency, and thus, latency-sensitive applications are not feasible to be deployed in the cloud (Bonomi *et al.*, 2012). For instance, a study of two cloud gaming platforms shows that the latency for their games is between 135 500ms (Chen *et al.*, 2011). Secondly, the bandwidth to and from the cloud provider is a bottleneck of cloud computing, especially when more and more IoT devices will connect to the internet in the future (Sarkar *et al.*, 2018). Finally, the data privacy and security of cloud computing faces many risks (Takabi *et al.*, 2010). For example, many IoT devices do not have the capability to encrypt their data before sending it to the cloud (Alrawais *et al.*, 2017). Hence, cloud computing is not suitable for many IoT applications that are latency-sensitive, generating a large volume of traffic or requiring high-security level (e.g., smart homes, smart cities and smart grid) (Sen, 2015).

Thus, we need to turn to a new computing paradigm to make up for these deficiencies of cloud computing. Against this background, fog computing is proposed as a promising complement to cloud computing for IoT applications (Bonomi *et al.*, 2012). Fog computing is similar to cloud computing that provides computing, networking and storage services between IoT devices and the cloud (Bonomi *et al.*, 2012). Fog computing reduces latency and saves network bandwidth requirements by processing data close to IoT devices. Furthermore, fog computing makes IoT systems more secure. For instance, it reduces the chances of eavesdropping by processing data locally and reducing the amount of sensitive data being transmitted to the cloud (Bonomi *et al.*, 2012). For example, fog computing can process the video stream collected from a video surveillance system, which is latency sensitive, privacy sensitive and generates a log of data, to track objects (Liu *et al.*, 2018). A new secure data storage and searching framework, where raw data are first processed and stored in the fog, and then non-time-sensitive data are sent to the cloud, is proposed and verified to significantly improve the data security in the IoT because the amount of data transmitted in the network are deceased (Fu *et al.*, 2018).

Besides the benefits of fog computing mentioned above, it is clear that the fog needs a system to make decisions on how to schedule its resources in order to provide high-quality services to its users (see chapter 1.3). In this report, we focus on resource allocation in fog computing (RAFC) in our report because resource allocation is critical to making full use of resources in the fog and improving the Quality of service (QoS)

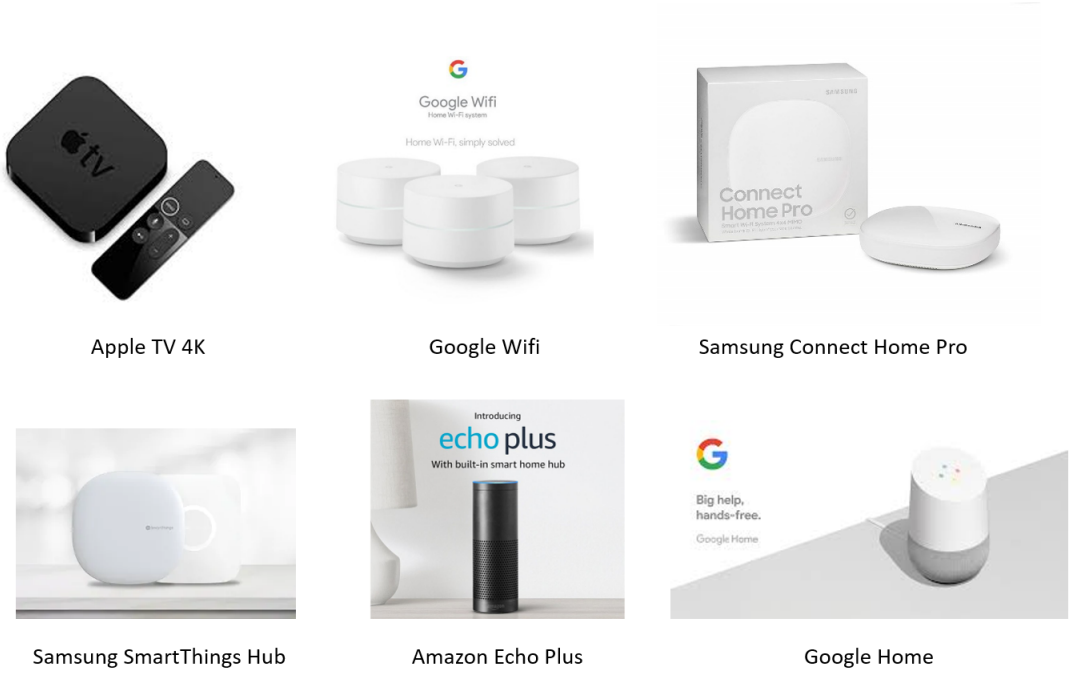

FIGURE 1.1: Some existing products that can serve as fog nodes

of the fog (Mahmud *et al.*, 2018). In particular, we study RAFC in a strategic setting (i.e., assuming fog users are rational) due to the fact that the information of fog users' tasks is usually private, and fog users are not necessarily truthful in reality.

In the following, we introduce the fog computing paradigm in more detail in Section 1.1 1.3 and present the research challenges of computational resources allocation in fog computing in Section 1.4. Moreover, we highlight our research contributions and give a brief outline of the whole report in Sections 1.5 and 1.6 respectively.

1.1 Fog Computing Overview

Fog computing is formally defined as a virtualised[1] computing platform, which lies between IoT devices and cloud computing data centres, providing storage, computing and networking services, which is typically located at the edge of the network (i.e., the periphery of the network) (Bonomi *et al.*, 2012). In particular, the main components of the fog are fog nodes (FNs), which are devices or facilities that can provide computing resources to IoT devices at the edge of the network, such as routers, switches, smartphones, laptops and base stations (Yi *et al.*, 2015). Some commercial offerings of FNs are illustrated in Figure 1.1. Compared with centralised data centres of the cloud, FNs have limited computing power and are distributed, heterogeneous, and closer to IoT

[1]Virtualisation is a technique to create virtual versions of resources such as an operating system or a storage device. It allows many users to share a single physical server and different operating systems, and applications can run on the same hardware at the same time.

devices (Tordera *et al.*, 2016). For instance, Google data centres, which provide services for Google users, are only deployed at 15 locations around the globe[2], and only two of them are in Asia. In contrast, many FNs would need to be deployed just along a motorway to provide low-latency services for autonomous vehicles (AVs) (Xiao and Zhu, 2017). Figure 1.2 shows the architecture of IoT and the position of fog computing in this architecture (Bonomi *et al.*, 2012).

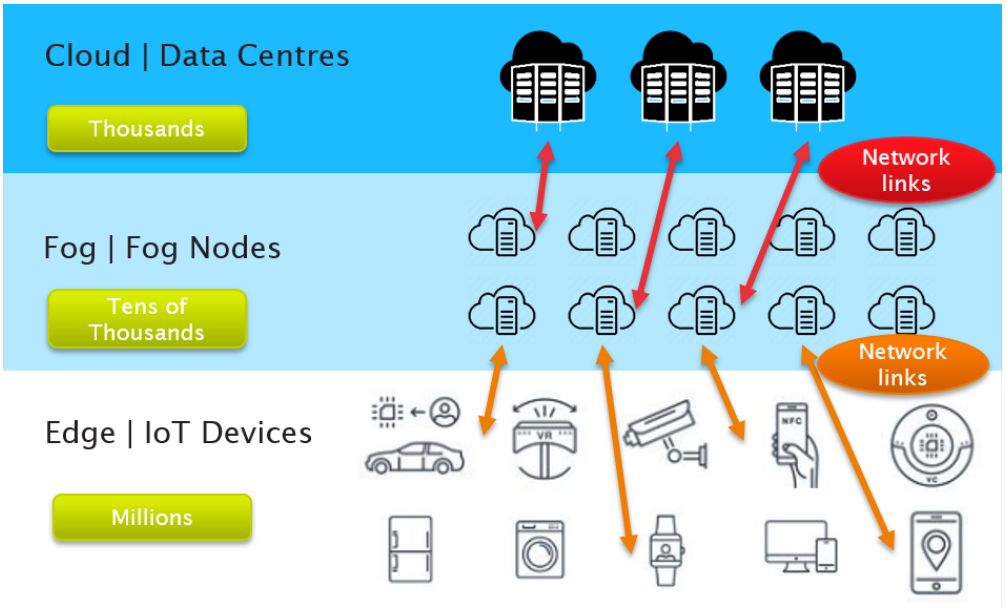

FIGURE 1.2: The architecture of the IoT, fog computing and cloud computing[3]

Note that there are several computing paradigms that are similar to fog computing, such as geo-distributed clouds (Narayanan *et al.*, 2014), edge computing (Garcia Lopez *et al.*, 2015), mobile cloud computing (Fernando *et al.*, 2013) and cloudlets (Satyanarayanan *et al.*, 2009). Although there are many differences between these computing paradigms, in essence, they all places computational resources close to the users and they all have similar resource allocation models. As the name suggests, a geo-distributed cloud consists of several geo-distributed cloud data centres, which offer many advantages such as low latency, the ability to safeguard against failures, and exploitation of different regional energy prices. However, compared with fog nodes, the data centres in a geo-distributed cloud is still far away from IoT devices (Narayanan *et al.*, 2014). Furthermore, edge computing pushes the computing power directly into the edge of the Internet, that is, applications are processed on IoT devices that have sufficient computing power (Garcia Lopez *et al.*, 2015), and mobile cloud computing is about offloading computing tasks from mobile devices to computing resource providers such as cloud or fog (Fernando *et al.*, 2013). In addition, cloudlets is just an alternative name for fog (Satyanarayanan *et al.*, 2009). Given this, in the rest of the report, we refer to fog computing because it is

[2]https://www.google.com/about/datacenters/inside/locations/index.html

[3]source of icons: https://www.colourbox.com/vector/set-vector-flat-line-icons-internet-of-things-vector-18757991

the most appropriate platform for many IoT applications (Bonomi *et al.*, 2012), but our discussion and solutions are equally applicable to the other types of systems mentioned.

Similar to the cloud, the primary function of the fog is to provide computing and networking resources, and fog computing is rather a complement to cloud computing than a substitute (Matt, 2018). The differences between them make fog computing more suitable for many IoT application scenarios. In what follows we highlight important features that fog computing should have (Bonomi *et al.*, 2012). Note that, fog computing is recently proposed and have not yet been widely used, and some of these features are shared by both fog and cloud computing.

First, the fog should have the ability to execute low latency computational tasks for applications such as AVs, real-time video analytics and online games. For example, at high speed, the response time for the autopilot system of an AV to avoid an accident can be just several milliseconds. Fog computing can reduce latency and network traffic because the data collected by IoT devices will be sent to FNs nearby for processing and storage, instead of sending them to often far-away cloud data centres (Atlam *et al.*, 2018).

Second, since many IoT devices are mobile (e.g., smartphones, laptops, AVs), the fog should support mobility through wireless access. It can use techniques like Locator ID Separation Protocol (LISP)[4], Mobile IPv6 (MIPv6) and Hierarchical Mobile IPv6 (HMIPv6) (Liu *et al.*, 2015). For example, LISP separates the address space of locators and identifiers and has the advantage of mobility. In addition, the mobility of IoT devices brings new challenges to resource allocation such as VM migration and dynamic traffic routing.

Third, the fog also needs to support real-time interactions besides batch processing because real-time interactions are demanded by many IoT applications such as virtual reality (VR) and augmented reality (AR).

Finally, some data of the IoT needs to be processed in the fog and others are more suitable to be processed in the cloud because the cloud has data that the fog does not have or has a greater computing power (Bonomi *et al.*, 2012). As a result, the fog should also support interplay with the cloud, which means that the fog can send data to the cloud to process. For example, the big data generated by a smart grid is first processed in the fog and then the results are sent to the cloud for latency-insensitive analysis (Borylo *et al.*, 2016).

[4]http://www.lispmob.org

1.2 Fog Computing Application Scenarios

Next, we will introduce the application scenarios that are especially suitable for fog computing, such as AR, VR, real-time video analytics, smart grid and autonomous vehicles.

- **AR and VR:** AR can "augment" real-world scenes by adding computer-generated graphics to the real world, and VR constructs an environment of computer-generated feedback of video and sound that makes people have the feeling that the generated world is immersive and realistic. Examples of AR include Microsoft HoloLens (Gabriel Evans, 2017) and Google Glass (Muensterer *et al.*, 2014), while HTC Vive (Dempsey, 2016) and Oculus Rift (Desai *et al.*, 2014) are representatives of VR. Since the computer-generated environment should be able to deceive the senses and humans are very sensitive to feedback delays (latencies greater than 50ms are perceptible) (Brooks, 1999), the video must have a high resolution to feel real as well as a high frame rate and very low latency for sensitivity. Consequently, both AR and VR need immense computing power to run properly. However, mobile devices like smartphones often have limited computing power (Yang *et al.*, 2018). A possible solution is to combine fog computing and mobile AR or VR devices and put intensive computations in the fog. For example, a startup called GridRaster is developing a VR/AR software platform on Saguna's fog computing solution Open-RAN (Alto, 2018) to offer an immersive VR/AR experience on mobile devices by leveraging fog computing. Furthermore, a large shipbuilder Navantia is starting to use an AR system, which is also based on fog computing. A study of this system shows that fog computing based AR system respond clearly faster than cloud-based AR system (Fernández-Caramés *et al.*, 2018).
- **Real-Time Video Analytics:** Real-time video analytics uses artificial intelligence to analyse video contents generated by huge numbers of cameras deployed in buildings, along the streets or in cars in real-time. Fog computing can help many video analytics applications, such as video surveillance, object/face recognition, and smart traffic lights, which typically require low latency, immense computing power and large bandwidth. For example, a bandwidth of 25 Mbps is required to stream a 4K video (Ananthanarayanan *et al.*, 2017). A prototype of a dynamic urban surveillance system based on fog computing, which uses three drones to monitor vehicles and a laptop as the FN, was built (Chen *et al.*, 2016). Evaluations showed that this scheme is with great promise for smart urban surveillance applications, as it can track the target all the time in a noisy environment and the estimated speed is close to the actual speed of the target. Ali *et al.* (2018) show that the efficiency in the throughput of a facial recognition system using deep learning can be considerably improved by putting initial processing of the data at fog nodes compared to a cloud-only system.

- **Smart Grid:** The smart grid is a virtual network which can control the electricity grid including energy load, clean energy and grid safety. Many grid control applications are run on the edge of the network, like smart meters and micro-grids (Wei *et al.*, 2014). As grid networks are distributed widely, it is impractical to offload all computational tasks to the cloud. However, the fog can gather and process data from the grid and do real-time analytics and control. For instance, Singh and Yassine (2018) propose and validate an IoT big data analytics system, which uses fog computing, to store, process and analyse energy consumption data from smart homes. In addition, Okay and Ozdemir (2016) propose a three-tier (i.e., smart meters, fog nodes and cloud data centres) smart grid model and show that their model improves the cloud computing based smart grid in terms of privacy, latency, and locality.

- **AVs:** An AV is a vehicle that can drive by itself without human interference, which is the trend of the car industry. For example, all Tesla cars have autopilot features such as lane centring, adaptive cruise control, and self-parking (Endsley, 2017), and Google and Apple are also developing their own AVs. An AV uses cameras, radars and other sensors to capture information from the surrounding environment, which must be processed in real time to make proper driving decisions. Since AV applications also need very short latency and high bandwidth to send the data, fog computing is particularly suited to run them (Peter, 2015). Here, fog nodes can be AVs or static fog nodes such as mart base stations or smart routers. With fog computing, vehicles without sufficient computing power can utilise these fog noes to run compute-intensive applications (e.g., AR driving assistance and VR gaming), and spontaneous exchange of information between AVs is made possible (Xiao and Zhu, 2017). For example, a two-level architecture (i.e., AVs and fog nodes) is introduced to cope with the challenges of automated driving services such as large content volume, location-dependence and delay-sensitivity (Yuan *et al.*, 2018). Another framework called vehicular fog computing, which uses moving vehicles such as taxis and buses as mobile fog nodes, is proposed in order to provide cost-effective and on-demand fog computing service for AVs (Xiao and Zhu, 2017).

1.3 Fog Computing Resource Allocation

A typical fog has computing resources in its FNs and networking resources in its network. Computing resources, which generally include processors, random access memory (RAM) and disk storage, are capable of processing computational tasks. Additionally, IoT tasks are processed by virtual machines (VMs) generated in FNs. Here, VMs are emulations of real computers that contain all necessary elements to run fog tasks. Moreover, there are two different types of networking resources. One is the bandwidth among the FNs; the other is the bandwidth between FNs and IoT devices. Specifically, fog systems are

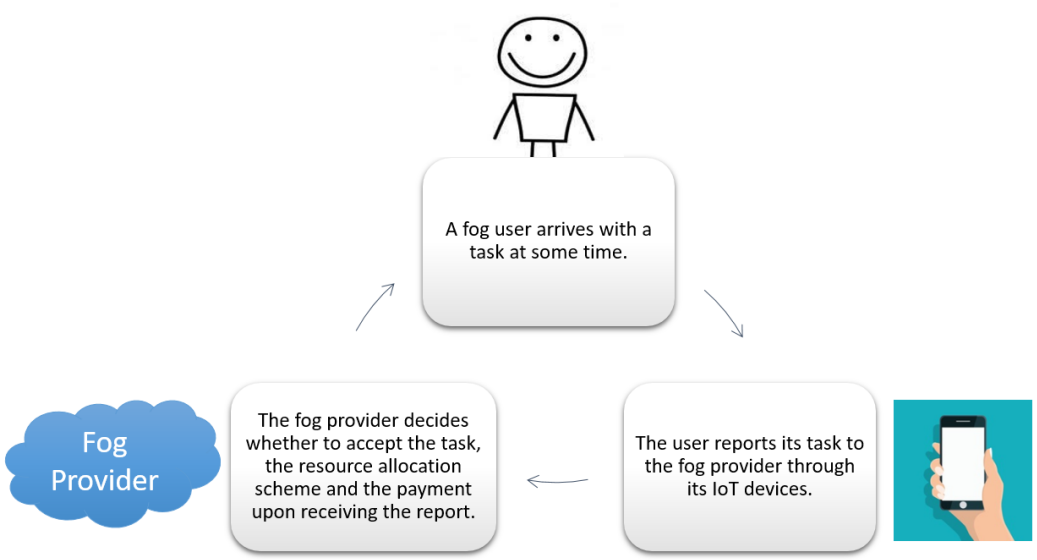

FIGURE 1.3: The procedure of fog computing resource allocation[5]

owned and operated by fog providers, which can be Internet service providers, wireless carriers, cloud service providers and even fog users who are willing to trade their spare computing resources (Yi *et al.*, 2015). Finally, the main business model of fog computing is the pay-per-use business model, i.e., fog users pay for the costs of fog resources when they use it rather than before or afterwards (Yi *et al.*, 2015).

The RAFC mainly involves three components, which are task scheduling, VM placement and traffic routing (Gu *et al.*, 2018). In detail, task scheduling is deciding when to process each task, and VM placement is assigning VMs, which process fog tasks, to appropriate FNs. Furthermore, traffic routing is finding the paths to send data between IoT devices and FNs. Notably, the latter two components are unique to fog computing and do not apply to centralised cloud computing.

Against this background, the process of the resource allocation is briefly introduced below (Shi *et al.*, 2017) (Figure 1.3). First, the fog users report the information about their tasks (e.g., value, demanded resource, processing time and deadline) to the fog provider over time. Then, after receiving the report, the fog provider will immediately decide whether to run the task, how much to charge and how to allocate resource for the task. Finally, the fog provider will send these decisions to the user, process the tasks accordingly and wait for new task requirements.

[5]source of images: https://www.shutterstock.com/image-vector/hand-holding-smart-phone-vector-illustration-1084761833

In addition, in a strategic setting, self-interested users can misreport their tasks such as delaying the report, expanding their processing time, declaring a higher value or an earlier deadline for them. In order to prevent misreporting, we can impose regulations and policies and penalise any users who misreport, or we can take the approach of online mechanism design which incentivise fog users to report truthfully. In detail, online mechanism design is a subfield of game theory that studies the problem of how to design mechanisms, which decides allocations of resource and monetary transfers based on received reports, to get desired objectives when rational users arrive over time (Nisan *et al.*, 2007). We choose to adopt the approach of online mechanism design to deal with this problem because imposing regulations has been criticised as being costly and slow (Demougin and Fluet, 2001).

1.4 Research Challenges

After the introduction of fog computing and the fog computing resource allocation problem, we now discuss the research challenges that we intend to address.

1. **Challenge 1** *Limited bandwidth resources.* If bandwidth in the fog is unlimited and free, it is feasible to just consider allocating computing resources for fog tasks. However, in actual situations, bandwidth is limited and costly. Thus, resource allocation mechanisms in the fog need to take bandwidth into consideration and try to make the best of it to reduce cost or latency. By contrast, bandwidth is usually ignored in cloud computing resource allocation (Shi *et al.*, 2016) because the traffic routing is not controlled by the cloud provider (cloud providers just buy bandwidth from Internet service providers (ISPs)) (He and Walrand, 2005). However, the emerging software-defined networking (SDN) (i.e., a network approach that enables the network to be centrally controlled using software) (Kreutz *et al.*, 2015) makes centralised network control in fog computing feasible.

2. **Challenge 2** *Dynamic VM allocation.* Another challenge that is unique to fog computing is the dynamic allocation of VMs. Firstly, in our model, VMs are not limited to several types, instead they can be dynamically assembled according to users demands. Secondly, unlike cloud providers, the fog provider needs to choose an FN to generate the VM for newly arrived tasks. Furthermore, in order to keep a low latency to a moving IoT device, the VM may need to be moved from one FN to another every now and then (Bittencourt *et al.*, 2015). Beyond that, VMs in the fog also need to move among FNs to serve new tasks efficiently.

3. **Challenge 3** *Time-oriented tasks.* How to allocate time-oriented tasks is another challenge to fog computing. This is because many IoT tasks are time-oriented, which means that they need a certain amount of computational time (between the

earlies start time and the deadline of the task) to achieve their maximum value, but they can still achieve part of the value if they are allocated less time. For example, suppose a user wants to run a video surveillance application with facial recognition to surveil their shops for 24 hours. Then, it is still of value to them if the surveillance lasts less than 24 hours, say, 18 hours.

4. **Challenge 4** *High social welfare resource allocation.* This challenge is about how to maximise the social welfare (i.e., the overall utility) of fog users given the limited computing and networking resources of the fog, especially when there are a lot of users demanding diverse resources dynamically. In more detail, the utility of a computing task represents the value obtained by the fog user from processing that task. Maximising social welfare is a typical mechanism-design goal because social welfare represents the aggregated satisfaction of users. Note that efficiency of resource allocation mechanisms means efficiency in terms of social welfare in this thesis.

5. **Challenge 5** *Resource allocation in a strategic setting.* In practice, fog users may be strategic, which means that users can misreport their tasks for their own benefit. In general, such behaviours are detrimental to the overall efficiency of the fog, and possibly lead to much worse system performance than expected. To address this problem, the fog provider may use a truthful resource allocation mechanism. Here, truthfulness (or strategyproofness) means that fog users cannot get a better utility by misreporting their tasks to the fog provider (Nisan *et al.*, 2007). This challenge is composed of two parts. The first part is how to design a mechanism that achieves truthfulness while only degrading a little proportion of the efficiency or without a loss compared to the best non-truthful mechanism in a non-strategic setting (i.e., users always report their information truthfully unconditionally). The second part is that this truthful mechanism should also outperform state-of-the-art non-truthful mechanisms in social welfare in a strategic setting.

6. **Challenge 6** *Resource allocation in online settings.* Furthermore, the online nature of the RAFC, which means that allocation decisions have to be made as information of new task reported over time and without any knowledge of the future, makes social welfare maximisation even more challenging. For example, myopic poor decisions can make high-value tasks in the future unable to be schedules, which lead to big losses in social welfare. What's more, it also makes designing truthful mechanisms more challenging. For example, a classical truthful mechanism—VCG mechanism (see Section 2.1.2) does not work in online scenarios. However, a large amount of literature just focus on offline settings, in which all users report their information simultaneously.

7. **Challenge 7** *Cost-aware resource allocation.* In contrast to most existing literature, which only treat social welfare as the total value of finished tasks, we take

resource costs into consideration in our model. Therefore, our problem, which contains a mixture of packing and covering constraints (packing task demands within resource capacities, and covering accepted tasks by paying operational costs of resources) is more challenging than problems with only packing constraints (Azar *et al.*, 2013).

8. **Challenge 8** *Timely resource allocation.* Since it is common to have ad hoc fog tasks that request to process immediately after the report. The resource allocation mechanism should be computationally efficient so that it can make allocation and pricing decisions right away when it receives new reports of fog tasks. However, the resource allocation problem for fog computing has an NP-hard nature (explained in Section 3.1), and the number of fog tasks, fog nodes and IoT devices can be really big. So it is both important and challenging to design very computationally efficient resource allocation mechanisms.

1.5 Research Contributions

In order to address the research challenges listed above, this report takes the approaches of constrained optimisation and online mechanism design to study the problem of RAFC with the aim of maximising social welfare. In more detail, constrained optimisation is a domain in mathematical optimisation whose goal is to find the values of related variables to optimise an objective function given some constraints of these variables (Bertsekas, 2014). In our problem, the objective function, which is to be maximised, calculates the total social welfare, and constraints consist of both resource constraints and time constraints. Furthermore, we take the approach of online mechanism design because we intend to design a truthful online mechanism and in our problem users arrive over time and thus resource allocation decisions must be made online.

The main contribution of this report is proposing a truthful mechanism that can also achieve near-optimal social welfare of fog users. By designing, implementing and evaluating our resource allocation mechanism, we show that it achieves better social welfare than benchmarks and achieves social welfare close to the optimal (around 90%) in a strategic setting. Specifically, we make the following three contributions to address the challenges in Section 1.4.

1. **We formulate the fog computing resource allocation problem as a constraint optimisation problem.** In order to address challenges 1–3 mentioned above, we formulate the fog computing resource allocation problem, which considers the bandwidth constraints and traffic routing as well as allows flexible allocation of VMs, as a constraint optimisation problem. We also model it as an online mechanism design problem where a fog task requests an amount of processing time with specific resource demands.

2. **We design a truthful mechanism and evaluate its performance in social welfare.** We design a new truthful and individually rational (IR) mechanism called flexible online greedy (FlexOG). A mechanism is called IR if no participants can get a negative utility by participation (Nisan *et al.*, 2007). Thus, under our mechanism, fog users will truthfully report their tasks as soon as they get them. By extensive simulations, we show that FlexOG achieves social welfare better than that achieved by the state-of-the-art benchmarks (up to 10%) and is close to the optimal value (around 90%). Therefore, we have addressed challenge 4,6,7 and part one of challenge 5, which is designing a truthful mechanism that outperforms state-of-the-art truthful mechanisms.

3. **We conduct simulations to show that our truthful mechanism performs better than online optimal, which is a non-truthful mechanism, in a strategic setting.** To fully address challenge 5, we show that FlexOG achieves better social welfare than the online optimal mechanism, which optimally allocates resources over time with the information it has received, in a strategic setting by simulations. This is surprising because online optimal achieves a better social welfare if all users report truthfully. We choose online optimal as the benchmark here because online optimal is a greedy algorithm, which is good enough for resource allocation under uncertainty (Gupta *et al.*, 2017b). First, we show that on average, non-truthful users have a higher utility than truthful users under the online optimal mechanism, so that fog users indeed have incentives to misreport their tasks. Then, we show that when even a small proportion of users misreport, the social welfare achieved by online optimal declines significantly and is lower than that achieved by FlexOG. This is interesting because in literature Price of Anarchy (PoA) (i.e., the ratio between the optimal centralised solution and the worst Nash equilibrium[6]) is often studied to measure how the efficiency (e.g., social welfare, revenue, and average latency) of a system degrade when its users are self-interested and rational (Koutsoupias and Papadimitriou, 1999). Another way to measure this is using the price of sinking, which is the biggest ratio between the value of the optimal solution and the value of a sink equilibrium (i.e., an equilibrium repeated selfish behaviours converge) (Goemans *et al.*, 2005). However, PoA cannot apply to our model because often pure strategy Nash equilibria do not exist and randomised strategies are unrealistic in our case. Furthermore, the price of sinking is also not suitable here because in our model fog users are not fixed, and they do not always acquire fog services repeatedly. So we study how a truthful mechanism actually performs compared with a non-truthful one in a strategic setting by simulations.

[6]In a Nash equilibrium, no user can benefit by changing their strategy unilaterally.

1.6 Outline of the Report

This report consists of five chapters. Specifically, in Chapter 2 we outline the literature which is related to our research and the techniques they use to address the challenges listed in Section 1.4. It includes the related theory of game theory and mechanism design (Section 2.1) and the related resource allocation mechanisms proposed for MARA (Section 2.2), RACC (Section 2.3) and RAFC (Section 2.4). In Chapter 3 and Chapter 4, we demonstrate the contributions of our research, including our model of RAFC (Section 3.1), the descriptions and properties of benchmarks and our proposed mechanism (Section 3.3 and Section 3.4), the experimental setup for simulations (Section 4.1) and results and analysis of the simulations (Section 4.2). Finally, we draw conclusions from the research so far and detail our planned future work in Chapter 5

Chapter 2

Literature Review

Since the main aim of our work is to design a truthful mechanism that is efficient in terms of social welfare as described in the previous chapter. In this chapter, we provide the necessary background for our research and relevant literature on resource allocation mechanisms for non-strategic settings and truthful resource allocation mechanisms for strategic settings. Our main aim is to explore the progress of current research, their advantages and disadvantages and evaluate the extent to which the research challenges in Section 1.4 have already been addressed and what research gaps are still left open.

We begin our literature review with the relevant knowledge in algorithmic game theory (Section 2.1) including the pertinent concepts of game theory, mechanism design and online mechanism design because it is the theory of designing mechanisms that achieve a specific outcome including truthfulness (Challenge 5). Then in Sections 2.2, 2.3 and 2.4, we examine the work on general multiagent resource allocation (MARA), resource allocation in cloud computing (RACC) and resource allocation in fog computing (RAFC) respectively, and discuss what challenges they address and what challenges are still open. Finally, Section 2.5 summarises our findings and the main research gaps according to our research challenges.

2.1 Preliminaries

Given that mechanism design is the theory we use to deal with the problem of resource allocation in strategic settings (Challenge 5), in what follows we explain the main concepts before proceeding to related work. If the reader is familiar with the basics of game theory and mechanism design, they can choose to skip this section. Mechanism design is a domain in game theory (i.e., the study of participants' strategies and behaviours in a certain game, which means a formal model of an interactive situation). It studies how to design the mechanism, which makes decisions of resource allocation and money transfer,

of a game to achieve certain desirable properties or outcomes under the assumption that all participants are rational (Hurwicz, 1973).

Traditionally, game theory and mechanism design are widely used in the field of microeconomics as important analysis tools. Furthermore, they are also used in some computer science problems such as sponsored search auctions (Zhu *et al.*, 2012), spectrum auctions (Qin *et al.*, 2015) and bandwidth allocation (Wu *et al.*, 2012). Particularly, online mechanism design is an extension of mechanism design which deals with problems in a dynamic environment (Challenge 6) where agents are allowed to come and go at any time, such as the smart grid, cloud computing or fog computing. In contrast, classic mechanism design only deals with static environments. For example, in a spectrum auction, telecommunications suppliers submit their bids before the deadline and the spectrum allocation decisions are made all at once. Similarly, in sponsored search auctions, once someone searches online, the advertisers bid for advertisement slots simultaneously, and then a decision is made to decide how all the slots will be assigned.

The structure of this section is as follows. In Section 2.1.1, we introduce the basic concepts of game theory. In Section 2.1.2, we examine the relevant concepts and theorems of mechanism design and show a representative example. Finally, we introduce online mechanism design and discuss some results in a special online domain called single-valued online domain in Section 2.1.3

2.1.1 Game Theory

Game theory is "the study of mathematical models of conflict and cooperation between intelligent rational decision-makers" (Roger, 1991). In more details, in game theory decision makers are often called agents, players or participants, and we use the term agents in this thesis. To analyse agents' behaviours, we must make assumptions about how they behave. In game theory, this assumption is that all agents are rational, which means they always strive to maximise their utility. The utility of an agent is the quantification of its preference over all outcomes of a game, that is, a real number is assigned to each outcome, and this is defined as its utility. If the utility of outcome a is greater than that of outcome b for the agent, then it prefers outcome a to outcome b. In the game, every agent tries to get the best outcome for itself and also knows that other agents similarly try to achieve the best outcomes for themselves. In order to achieve this goal, agents follow strategies that decide their behaviours in all situations. Researchers in game theory have applied these mathematical models to understand decision making in economics (Gu *et al.*, 2005; Kelly, 2003), biology (Hammerstein and Selten, 1994; Dugatkin and Reeve, 2000) and multi-agent systems (Parsons and Wooldridge, 2002; Semsar-Kazerooni and Khorasani, 2009). Furthermore, RAFC can be modelled as a game, where fog users are players of the game, and they act to compete for fog resources (see the details in Section 3.1).

In game theory, a simultaneous game (e.g., rock-paper-scissors) is a game where each agent chooses its actions without any knowledge of the actions of other agents. Our RAFC problem belongs to this type of game because fog users are assumed to not have any information on other users' actions, although they do not take actions at the same time. First of all, we present a rigorous mathematical definition of a simultaneous game, so that we can introduce other concepts based on it. Formally, we define a simultaneous game as consisting of a finite set I of agents, $I = \{1, 2, ..., n\}$. The relevent traits of agent i (e.g., preferences and time constraints) are summarised as the type $\theta_i \in \Theta_i$ of that agent, and the type profile $\theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$ is the set of all agents' types. The behaviour of agent i is captured by a function that maps from all its possible types to its set of actions: $\Theta_i \to X_i$, which is called the strategy of agent i. S_i is the set of all strategies of agent i. The strategy of agent i and the strategies vector of other agents are s_i and s_{-i}, respectively. Then the vector of strategies selected by all agents is $s = (s_1, s_2..., s_n)$. The set of all possible vectors of strategies is denoted as S, and the set of all possible vectors of strategies except agent i's is S_{-i}. The utility of agent i when it plays strategy s_i and others play strategy s_{-i} is $u_i(s_i, s_{-i}) \in \mathbb{R}$.

In addition, the strategies of agents can be categorised into pure strategies and mixed strategies. A pure strategy uniquely determines the behaviour of an agent for any situation (the state of the game) it can face. In other words, an agent using a pure strategy will always take the same action when it faces the same situation. On the other hand, a mixed strategy is an assignment of a probability to each pure strategy. This allows an agent to randomly choose a pure strategy when it needs to take actions. In this thesis, we focus on pure strategies because we aim to incentivise agents to use a pure strategy, which is to reveal their types truthfully.

Furthermore, given a game, there are different ways to predict its result. We use solution concepts to describe how we predict game results, and these predictions are called solutions. There are two basic solution concepts needed in this thesis: dominant strategy equilibrium and Nash equilibrium. Here, an equilibrium means a strategy profile (i.e., a set of strategies for all players) which is stable enough to be predicted as the actual result of the game.

To present dominant strategy equilibrium, we first introduce dominant strategies of agents. If an agent has a unique best strategy in the game regardless of other agents' choices, then we call it a dominant strategy for that agent. Namely, $s_i^* \in S_i$ is the dominant strategy for agent i, if:

$$u_i(s_i^*, s_{-i}) > u_i(s_i, s_{-i}), \quad \forall s_i \neq s_i^*,\ s_{-i} \in S_{-i}.$$

Furthermore, a weakly dominant strategy for agent i is a strategy such that no other strategies are better than it. Formally, $s_i^* \in S_i$ is a weakly dominant strategy for agent

i, if:

$$u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i}), \quad \forall s_i \neq s_i^*,\ s_{-i} \in S_{-i}.$$

Then, a strategy vector $s \in S$ is called a dominant strategy equilibrium, if every strategy in it is a weakly dominant strategy:

$$u_i(s_i, s'_{-i}) \geq u_i(s'_i, s'_{-i}), \quad \forall i \in I,\ s'_i \in S_i,\ s'_{-i} \in S_{-i}$$

It is important to note that if a game has a dominant strategy equilibrium, each agent has a best strategy regardless of other agents' strategies. This property is very useful when we design mechanisms because if the dominant strategy equilibrium of a mechanism is desirable, this mechanism can be used in strategic settings and still get this desirable equilibrium. For example, if we hope that the strategy of every agent is simply to reveal its type truthfully, then the corresponding equilibrium is desirable in this case. Thus, it is the most widely used solution concept in mechanism design, and we focus on designing mechanisms that have dominant strategy equilibria in this thesis.

However, in most games, there is no dominant strategy equilibrium, because, in these games, an agent's best strategy changes with other agents' strategies. The stable equilibria in these games are called Nash equilibrium, which is one of the most important concepts in game theory and has widespread applications. In a Nash equilibrium, no agent can benefit from changing its behaviour unilaterally. Formally, a strategy vector $s \in S$ is called a Nash equilibrium if:

$$u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i}), \quad \forall i \in I,\ s'_i \in S_i.$$

Nash equilibria can exist in a stable manner because no agent in the Nash equilibrium has an incentive to change its strategy. So normally games will end up in one of its Nash equilibria if it has one. This solution concept is not very convincing in predicting agents' strategies if there is more than one Nash equilibrium in a game (Nisan *et al.*, 2007). Furthermore, the Nash equilibria are not guaranteed to be reached even when it is unique because an agent's belief of other agents' strategies may be incorrect.

2.1.2 Mechanism Design

We now introduce the core concept of this thesis–mechanism design, which can be viewed as reverse game theory. For game theory, the goal is to analyse the strategies and behaviours of agents in a certain game (as described in Section 2.1.1), while for mechanism design, the goal is to design the mechanism of the game to achieve a certain result or social choice (i.e., an aggregation of different types of agents toward a single collective decision) in a strategic setting (Nisan *et al.*, 2007). Since the types (or part of the types)

of the agents are usually private (i.e., not common knowledge), mechanism design is necessary under these circumstances. In particular, the designer of the mechanism does not participate in the game. Instead, it designs the mechanism, or rules of the game, to achieve certain goals (e.g., social welfare maximisation or revenue maximisation). Since the mechanism designer needs to make decisions based on agents' types, the difficulty of mechanism design is how to incentivise agents to reveal their true types.

In order to model strategic behaviours of the agents, a model called independent private values, and strict incomplete information game (Nisan *et al.*, 2007, Chapter 9) is adopted in this thesis. Independent private values mean that the valuation of an agent is not affected by any other agents' information. Here, the valuation function of an agent represents its valuation of the result it gets. Unlike the utility function, this does not include any payments (see Definition 2.2). Strict incomplete information means that there is no probabilistic information in the model. For example, we do not know the distribution of valuation functions. This model is reasonable in our RAFC scenario, in which users are assumed to have independent valuation functions and have no knowledge about the types of others, and the mechanism designer has no probabilistic information about the agents either (Challenge 5). Here, a game comprises a mechanism environment and a mechanism. We first introduce the mechanism environment.

Definition 2.1 (mechanism environment)**.** A mechanism environment $\mathcal{E} = (I, A, \{X_i\}_{i\in I}, \{\Theta_i\}_{i\in I}, \{v_i\}_{i\in I})$ contains the following ingredients:

1. A set of n agents: I
2. A set of results: A
3. For every agent i, a set of actions X_i
4. For every agent i, a set of possible types Θ_i. The type of agent i is $\theta_i \in \Theta_i$
5. For every agent i, a valuation function $v_i : \Theta_i \times X_1 \times \cdots \times X_n \to \mathbb{R}$, which maps from the type of agent i (θ_i) and the actions taken by all agents $(x_1, x_2, \ldots, x_n)$ to a real number (Nisan *et al.*, 2007, Definition 9.40)

In the following sections, we introduce some fundamental concepts and theories in mechanism design. Specifically, we introduce the definitions of social choice function and mechanism (Section 2.1.2.1), and we describe a special type of mechanisms called truthful direct-revelation mechanisms along with an important theory called the revelation principle (Section 2.1.2.2).

2.1.2.1 Social Choices and Mechanisms

Building on the general definitions, we next introduce the notion of a general (nondirect-revelation) mechanism. A general mechanism is composed of two components: an outcome function that chooses a result based on the profile of agents' actions, and a payment function that decides a payment for every agent also based on the profile of actions.

Definition 2.2 (mechanism)**.** A general mechanism $\mathcal{M} = (a, \{p_i\}_{i \in I})$ comprises:

1. an outcome function $a : X_1 \times \cdots \times X_n \to A$
2. payment functions $p_1, \ldots, p_n$, where $p_i : X_1 \times \cdots \times X_n \to \mathbb{R}$ (Nisan *et al.*, 2007, Definition 9.24).

Then, the game induced by a mechanism over some mechanism environment is formally defined below.

Definition 2.3 (games of mechanisms)**.** The game $\Gamma(\mathcal{M})$ induced by a mechanism $\mathcal{M} = (a, \{p_i\}_{i \in I})$ over mechanism environment $\mathcal{E} = (I, A, \{X_i\}_{i \in I}, \{\Theta_i\}_{i \in I}, \{v_i\}_{i \in I})$ is the strict incomplete information game $\Gamma(\mathcal{M}) = (I, \{\Theta_i\}_{i \in I}, \{X_i\}_{i \in I}, \{u_i\}_{i \in I})$ where the agents' utility functions $u_i(\theta_i, x_1, \ldots, x_n) = v_i(\theta_i, a(x_1, \ldots, x_n)) - p_i(x_1, \ldots, x_n)$ (Nisan *et al.*, 2007, Definition 9.24).

Definition 2.4 (social choice function)**.** The social choice function f maps the types of agents to the results (allocation of goods):

$$f : \Theta_1 \times \cdots \times \Theta_n \to A$$

In general, we want to design a mechanism that can get the same outcome of a desired social choice function (e.g., a social choice function that maximises the total utilities of the agents). So if the equilibrium under a mechanism maps to the same result as a social choice function, we say that this mechanism implements this social choice function.

Definition 2.5 (implementation)**.** The mechanism implements a social choice function $f : \Theta_1 \times \cdots \times \Theta_n \to A$ in dominant strategies if for some dominant strategy equilibrium $s_1, \ldots, s_n$ of the induced game, where $s_i : \Theta_i \to X_i$ we have that

$$f(\theta_1, \ldots, \theta_n) = a(s_1(\theta_1), \ldots, s_n(\theta_n)), \quad \forall \theta_1, \ldots, \theta_n \in \Theta$$

(Nisan *et al.*, 2007, Definition 9.24).

In the following section, we introduce a special type of mechanism called direct-revelation mechanisms and a property of mechanisms called truthfulness. Then we introduce the revelation principle, which shows that we can just focus on direct-revelation truthful mechanisms when designing mechanisms.

2.1.2.2 Direct-Revelation Mechanisms and Truthfulness

There is a special type of mechanism called direct-revelation mechanisms, which is very important in mechanism design because of the revelation principle (see Proposition 2.8). In a direct-revelation mechanism, the set of actions of agent i, X_i, is the set of its possible types Θ_i, which means what an agent can do is just to reveal its type to the mechanism. To formally introduce this definition below, we first introduce some notation. We denote the set of all possible valuation functions for agent i as V_i, and the vector of all types except agent i's as $\theta_{-i} = (\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_n)$. Similarly, we define $\Theta_{-i} = \Theta_1 \times \cdots \times \Theta_{i-1} \times \Theta_{i+1} \times \cdots \times \Theta_n$.

Definition 2.6 (direct-revelation mechanisms)**.** A direct-revelation mechanism $(f, p_1, ..., p_n)$ is composed of a social choice function f and a vector of payment function $(p_1, ..., p_n)$. The payment functions $(p_1, ..., p_n)$ maps the reported types of agents to the payments of agents, where

$$p_i : \Theta_1 \times \cdots \times \Theta_n \to \mathbb{R}$$

(Nisan *et al.*, 2007, Definition 9.14).

Now we introduce the definition of truthfulness, which is a key concept in this thesis. It is also called strategy-proofness or dominant-strategy incentive-compatibility (DSIC). Under a truthful mechanism, agent i would prefer providing its true type to the mechanism regardless of other agents' strategies, because by providing false information agent i cannot increase its utility. In other words, truthful mechanisms can prevent agents from misreporting and manipulating.

Definition 2.7 (truthfulness)**.** A mechanism $(f, p_1, ..., p_n)$ is called truthful or DSIC if $v_i(f(\theta_i, \theta_{-i})) - p_i(\theta_i, \theta_{-i}) \geq v_i(f(\theta'_i, \theta_{-i})) - p_i(\theta'_i, \theta_{-i}), \quad \forall \theta_i, \theta'_i \in \Theta_i,\ \theta_{-i} \in \Theta_{-i}$ (Nisan *et al.*, 2007, Definition 9.15).

We can now introduce a fundamental principle in mechanism design called the revelation principle. It is fundamental because this principle shows that it is sufficient just to study direct-revelation truthful mechanisms to find out if it is possible to implement social choice function f in dominant strategies. This principle makes mechanism design much easier by narrowing the space of mechanisms that need to be searched.

Proposition 2.8. *(Revelation principle) Any social choice functions f that can be implemented in dominant strategies by a general mechanism can also be implemented by a direct-revelation truthful mechanism. Moreover, the payments of the agents in the truthful mechanism are the same as those in the equilibrium of the original mechanism (Nisan* et al.*, 2007, Proposition 9.25).*

In the following, we introduce two desirable properties of a mechanism besides truthfulness in this thesis. The first desirable property is called individual rationality (IR),

which guarantees that all (rational) agents are willing to participate. This is required because agents in our RAFC problem cannot be forced to participate in the mechanism. To satisfy this property, the mechanism should guarantee that no agents will get negative utility by joining the game.

Definition 2.9 (IR)**.** A mechanism is individually rational if no agent gets negative utility by participating the game (i.e., $v_i(f(\theta_1, ..., \theta_n)) - p_i(\theta_1, ..., \theta_n) \geq 0, \quad \forall \theta_1, \ldots, \theta_n \in \Theta_n$, $i \in I$) (Nisan *et al.*, 2007, Definition 9.18)

The second desired property is weak budget balance (WBB), which means that the total payment from agents is not lower than the overall cost. Here, the cost function $o : f(\theta_1, ..., \theta_n) \to \mathbb{R}$ of the mechanism maps from resource allocation decisions to the overall operational cost of the resource (Challenge 7). This property is desired because normally, fog providers try to avoid losses. So they are not willing to adopt a mechanism that is not WBB.

Definition 2.10. A mechanism is weakly budget balanced if it will never run a deficit. Formally if for every $\theta_1, \ldots, \theta_n$ we have that $\sum_{i \in I} p_i(\theta_1, ..., \theta_n) \geq \sum_{i \in I} o(f(\theta_1, ..., \theta_n))$.

To design truthful mechanisms, we define important properties of truthful mechanisms in the following. These properties can be used to help design mechanisms or judge if a mechanism is truthful.

Proposition 2.11. *A mechanism is truthful if and only if it satisfies the following conditions* $\forall i \in I,\ \theta_{-i} \in \Theta_{-i}$*:*

1. *The payment* p_i *only depends on the outcome of the game* $f(\theta_i, \theta_{-i})$*, namely, for every* θ_{-i}*, there exist prices* $p_a \in \mathbb{R}$, $\forall a \in A$*, such that for all* θ_i *with* $f(\theta_i, \theta_{-i}) = a$ *we have that* $p_i(\theta_i, \theta_{-i}) = p_a$
2. *The mechanism optimises for each agent (i.e.,* $f(\theta_i, \theta_{-i}) \in argmax_a(v_i(a) - p_a), \quad \forall \theta_i \in \Theta_i$*) (Nisan* et al.*, 2007, Proposition 9.27).*

Then, the following definition describes the property a social choice function must have in a truthful mechanism. This property is called weak monotonicity (WMON), and it means that if the result changes when an agent changes its valuation function, then the agent's value of the new result relative to its value of the old result should also be increased.

Definition 2.12. A social choice function f satisfies Weak Monotonicity if $f(\theta_i, \theta_{-i}) = a \neq b = f(\theta'_i, \theta_{-i}) \implies v_i(a) - v_i(b) \geq v'_i(a) - v'_i(b) \quad \forall i \in I,\ \theta_i, \theta'_i \in \Theta_i,\ \theta_{-i} \in \Theta_{-i}$ (Nisan *et al.*, 2007, Definition 9.28).

Then, the following theorem means that WMON is a necessary condition for truthfulness but becomes a sufficient condition only if all domains of types are convex sets. More specifically, a convex set is a set of points, for every pair of points within it, every point on the straight line that joins them also lies within the set.

Theorem 2.13. *If a mechanism $(f, p_1, ..., p_n)$ is truthful, f satisfies Weak Monotonicity (WMON). In addition, for every WMON f there exists payment function $p_1, ..., p_n$ such that the mechanism $(f, p_1, ..., p_n)$ is truthful only if all domains of types Θ_i are convex sets (Nisan* et al.*, 2007, Theorem 9.29).*

2.1.3 Online Mechanism Design

In this section, we extend the framework of classic mechanism design to online mechanism design, which deals with time-dynamic scenarios where agents' types are reported over time, and decisions must be made without future information. So unlike an offline mechanism, which just makes one overall decision, an online mechanism must make a sequence of decisions over time. Many real-world scenarios are dynamic, such as allocating computational resources to tasks being submitted over time, allocating electrical power to electric cars arriving over time or selling seats on an aeroplane to passengers. The RAFC problem we study in this thesis is also dynamic because fog users report their tasks over time and allocation decisions must be made at the same time of reports.

Online mechanism design faces challenges that are different from (offline) mechanism design because the environment is dynamic (Nisan *et al.*, 2007, Chapter 16). In particular:

1. Decisions must be made without future information.
2. Other than their valuation functions, agents can also misreport their private information about time, such as the arrival times and deadlines of their tasks (Challenge 3)
3. (limited misreports) In practice, an agent may not be able to misreport their type without limitations. For example, in the RAFC scenario, it is impossible to report a task before the agent knows it needs to run this task. So an agent can only misreport a later arrival time of its task.

Next, we discuss the component of a dynamic environment in this thesis. $T = \{1, 2, \ldots, t\}$ denotes discrete (possibly infinite) time steps. Let $q = (q^1, q^2, \ldots)$ denote the sequence of decisions (e.g., accepting/rejecting an agent, the payment of an agent, and when to serve an agent) which can be made when receiving reports from agents. The type of agent i, $\theta_i = (T_i^a, T_i^d, v_i) \in \Theta$, where $T_i^a, T_i^d \in T$ are its arrival time and departure time (deadline), and $v_i(k) \in \mathbb{R}$ is its valuation function. On the

other hand, the reported type of agent i is denoted as $\hat{\theta}_i = (\hat{T}_i^a, \hat{T}_i^d, \hat{v}_i) \in \Theta$, which may be different from θ_i because agents can misreport in our setting. The mechanism state $h^t = (\theta^1, \ldots, \theta^t; q^1, \ldots, q^{t-1}) \in H^t$ denotes the mechanism state in time step t, where H^t is the set of possible mechanism states in time step t. It contains the information of agent types $(\theta^1, \ldots, \theta^t)$, and decisions $(q^1, \ldots, q^{t-1})$ that have been made. Furthermore, we use $Q(h^t)$ to denote all possible decisions in state h^t. (Nisan *et al.*, 2007, Section 16.2)

In the following section, we extend the direct-revelation mechanisms and revelation principle to the online domain. Furthermore, we give a formal definition of limited misreports and the definition of truthfulness under limited misreports.

2.1.3.1 Direct-Revelation Online Mechanisms

Now we are ready to introduce the definition of a direct-revelation online mechanism. A direct-revelation online mechanism is an extension of a direct-revelation offline mechanism, but it has more restrictions on agents' behaviours. Note that the mechanism we design for RAFC in Section 3.4 belongs to this type of mechanism because the only actions fog users can take are to report their types, and they report over time instead of reporting simultaneously.

Definition 2.14 (direct-revelation online mechanism)**.** In a direct-revelation online mechanism, each agent only sends a single report about its type to the mechanism. The mechanism consists a social choice policy $f = \{f^t\}^{t \in T}$ and a payment policy $p = \{p^t\}^{t \in T}$, where decision $f^t(h^t) \in Q(h^t)$ is made in mechanism state h^t and payment $p_i^t(h^t)$ is the money that agent i needs to pay to the mechanism. For convenience, we denote the sequence of decisions as $f(\theta) = (q^1, q^2, \ldots)$ and overall payment collected from agent i as $p_i(\theta) \in \mathbb{R}$, given type profile θ (Nisan *et al.*, 2007, Definition 16.2).

Unlike offline scenarios, in certain online scenarios, it is reasonable to assume some limitations on the domain of the misreports, which can also simplify the mechanism design problem. For example, agent i cannot report an earlier arrival time than its true arrival time ($\hat{T}_i^a < T_i^a$) or report a later deadline ($\hat{T}_i^d > T_i^d$). This is a very natural assumption because agent i has no idea of its type before its true arrival time T_i^a, and reporting a later deadline will make the task miss its deadline.

Definition 2.15 (limited misreports)**.** The limited misreports that an agent can make is denoted as $C(\theta_i) \subseteq \Theta_i$, where θ_i is the type of agent i (Nisan *et al.*, 2007, Definition 16.4).

Then, we introduce the definition of a truthful online mechanism given limited misreports, which is similar to the definition of a truthful (offline) mechanism.

Definition 2.16 (truthful)**.** Given limited misreports C, a truthful or DSIC online mechanism satisfies:

$$v_i(\theta_i, f(\theta_i, \theta_{-i})) - p_i(\theta_i, \theta_{-i}) \geq v_i(\theta_i, f(\hat{\theta}_i, \theta_{-i}, \omega)) - p_i(\hat{\theta}_i, \theta_{-i}),$$

$$\forall \hat{\theta}_i \in C(\theta_i),\ \theta_i \in \Theta_i,\ \theta_{-i} \in \Theta_{-i}$$

(Nisan *et al.*, 2007, Definition 16.5).

Unfortunately, unlike (offline) mechanism design, only studying direct-revelation truthful online mechanism is not without loss of generality because the online revelation principle does not always hold. However, if the agents are assumed to have no-early arrivals ($\hat{T}_i^a \geq T_i^a$) and no-late departures ($\hat{T}_i^d \leq T_i^d$), then the online revelation principle still holds. Alternatively, revelation principle can be recovered by demanding each agent to send a non-informative "heartbeat" message in every time step $t \in [\hat{T}_i^a, \hat{T}_i^d]$ besides the report of its type (Nisan *et al.*, 2007, Section 16.22). In our RAFC model, although fog users can report late departures, it is reasonable to ask them to send a "heartbeat" message in every time step. So we focus on truthful direct-revelation online mechanisms in our work.

In the following sections, we examine related work on MARA (Section 2.2), RACC (Section 2.3), and RAFC (Section 2.4). Specifically, RAFC is the domain we study in this thesis, RACC is a close domain with many similarities, and MARA is the big domain that RAFC and RACC belong to. We analyse the different resource allocation models they deal with, the techniques they use, and why their approach cannot directly apply to our RAFC problem.

2.2 General Multiagent Resource Allocation

In this Section, we discuss recent work on general MARA, in which resources are distributed among several agents and these agents may have an impact on the allocation results (Chevaleyre *et al.*, 2006). MARA is relevant to a wide range of applications such as resource allocation in electricity grids (Gradwell and Padget, 2005), network routing (Feldmann *et al.*, 2003), RACC (Wang *et al.*, 2017) and RAFC (Yi *et al.*, 2015), which is the problem we focus on in the thesis.

Formally, MARA models the following resource allocation problems. There is a set of resources Z and a set of agents I. The resources may be indivisible or divisible (e.g., electricity, RAM and storage). Similar to what is described in Section 2.1.2.2, agents report their preferences over the bundles of resources (i.e., collections of resources that are wrapped together) by means of utility functions (i.e., functions that assign a real

number (utility) to each bundle to represent the agents' preferences) The goal is to find the resource allocation plan that maximises the social welfare.

Now, there are different types of social welfare, some common ones are utilitarian, egalitarian, and Nash product social welfare. Utilitarian social welfare is the most common one, which is the total sum of the utilities achieved by all agents, and it is also the social welfare we try to maximise in this thesis (Challenge 4). This is because we assume that the fog provider is a non-profit organisation and aims to improve the overall utilities.

In terms of utility functions of the MARA problem, some properties (e.g., monotonicity, submodularity, subadditivity, and fractional subadditivity) are to hold because they are realistic for many applications (Nisan). In the following, we give a formal definition of these properties. Let $u : 2^Z \to \mathbb{R}$ be a utility function.

- u is monotonic if $u(X) \leq u(Y)\ \forall\ X, Y$ with $X \subseteq Y \subseteq Z$.
- u is submodular if $\forall\ X, Y$ with $X \subseteq Y \subseteq Z$ and $\forall x \in Z \backslash Y$ we have that $u(X \cup \{x\}) - u(X) \geq u(Y \cup \{x\}) - u(Y)$.
- u is subadditive if $\forall X, Y \subseteq Z$ we have that $u(X \cup Y) \leq u(X) + u(Y)$.
- u is a fractionally-subaddtive function if there exists a collection of set functions, $\{a_1, a_2, \ldots, a_l\}$ such that each a_j is additive (i.e., the value of $X \subseteq Z$ is the sum of the values of the items in X), and $u(X) = \max_{j=1}^{l} a_j(X)$.

In our problem, the utility function of the agents satisfies three properties of them, except for submodularity (see Section 3.1). This is reasonable for our model, because often the processing time near the accomplishment of a fog task is much more valuable than the previous processing time, which causes the utility function to not be submodular.

Furthermore, resource allocation mechanisms can generally be categorised into two main classes, namely, centralised mechanisms and distributed mechanisms. For centralised mechanisms, there is a central authority in charge of allocating the resources to the agents, based on the utility functions of the agents. A typical example is a combinatorial auction, where bidders bid for bundles of resources and the auctioneer is the central authority. However, centralised mechanisms have the following disadvantages: they require communication between agents and the central authority always to be stable and reliable; the optimal allocation is often computationally hard; the procedure is not very flexible due to its centralised nature (Friedlander, 1982). To address these disadvantages, many fully or partially distributed allocation mechanisms (Herreiner and Puppe, 2002; Bouveret and Lang, 2011) are proposed. Under a fully distributed mechanism, agents execute the mechanism totally by themselves, while under a partially distributed mechanism, agents execute the mechanism with the help of a central authority. However,

distributed allocation mechanisms cannot guarantee the optimality of the allocation results in most situations (Rothe, 2015). Thus, we concentrate on centralised mechanisms in this thesis, although centralised or distributed mechanisms can both be used in RAFC.

2.2.1 Online Mechanisms for General Multiagent Resource Allocation

We next discuss work on truthful online mechanisms for MARA, which address Challenges 4, 5 and 6. In particular, we mainly look at resource allocation in EV charging because this problem is similar to RAFC and some work in this field also inspires the techniques used in our work. For example, the users all arrive over time, requiring some resource and have deadlines for this resource. However, there are also many differences between them such as an EV can charge the grid while IoT devices cannot send the resource back to the fog and the resource in the grid is homogeneous while the resources in the fog are heterogeneous. However, the work we discuss in this section does not proactively address the dynamic VM allocation challenge (Challenge 2) and the limited bandwidth challenge (Challenge 1).

First, model-free mechanisms are adopted by a large part of the literature. Here, model-free mechanisms are those that do not predict the demand and supply of resources in the future. The advantages of model-free mechanisms are that they need fewer assumptions and are usually easier to compute. Although these mechanisms are myopic (i.e., they make resource allocation decisions without considering possible future situations), their efficiency of resource allocation can still be near-optimal in certain scenarios. For example, Gerding *et al.* (2011) consider the problem of hybrid electric vehicles charging. Since hybrid vehicles can use both electricity and petrol, they have marginal non-increasing valuations for the electricity charged and have no lower limit of the power charged. To make the mechanism truthful, they combine a greedy algorithm with a technique called "burning" which leaves some units of electricity unallocated when there is still demand unsatisfied. The timing of the "burning" can be at the allocation of the resource or the departure of the users. In simulations, this mechanism's performance is close to a non-truthful scheduling algorithm and much better than a fixed price mechanism. However, the "burning" technique may reduce the efficiency of the mechanism significantly. To deal with the hybrid electric vehicle charging problem with various charging speeds, Robu *et al.* (2011) present two greedy-algorithm-based online mechanisms that incentivise users to truthfully report not only their valuations for the electricity but also their maximum charging rate. The "burning" technique is also used in this literature to achieve truthfulness. Furthermore, a two-sided pricing mechanism, in which both the EVs and charging stations report their private information is presented (Gerding *et al.*, 2013). However, this mechanism is only truthful on the buyer side (i.e., EVs have no incentive to misreport their types).

Model-based mechanisms, which consider possible future arrivals, are also used by some work to improve the performance of the mechanisms (Challenge 4). Stein *et al.* (2012) studied the problem of pure electric vehicles charging and proposed a model-based truthful online mechanism to allocate electricity. The mechanism modified the consensus algorithm (Bent and Van Hentenryck, 2004) by a technique called pre-commitment to achieve truthfulness. In the consensus algorithm, some future scenarios are sampled, and then the scheduling is solved for every one of them. Then, these scenarios vote to decide whether to accept a job (in their case, an EV charging job) or not. A mechanism with pre-commitment will only commit to an agent that the requested resource will be allocated before its departure, but when and how the resource will be allocated remains flexible. Real data simulations showed that this mechanism significantly outperforms a model-free algorithm and is nearly as efficient as an offline optimal algorithm. Moreover, Ströhle *et al.* (2014) considered the scenario where both the supply and demand for electricity are uncertain. They also developed a truthful online mechanism based on the consensus algorithm, which achieves near-optimal efficiency. In this thesis, we also use the technique of pre-commitment to improve the efficiency of our resource allocation mechanism in terms of social welfare.

Most of the above literature find truthful resource allocation mechanisms by looking for an allocation policy that satisfies WMON and then coupling it with a critical-value payment function. However, some literature takes the approach of price-based mechanisms which decide the resource price before allocation. For instance, Hayakawa *et al.* (2015) presented a class of price-based truthful online mechanisms which can be used in scenarios where electricity has various marginal generation costs, and users have multi-dimensional preferences by using carefully designed pricing functions and scheduling algorithms. They showed that their mechanism has near optimal efficiency in a realistic setting with different marginal costs. Moreover, they also present a class of price-based mechanisms allowing increasing marginal valuations, which is DSIC and IR in settings with uncertain procurement costs, multi-unit demand and multi-minded bidders (Challenge 3) (Hayakawa *et al.*, 2018). Although their system model is similar to ours (see Section 3) and our work is based on this class of mechanisms, they only assume homogeneous resource. Specifically, they do not consider limited bandwidth resource (Challenge 1) or deal with the dynamical assembly of VMs from several different types of resource (Challenge 2). Therefore, their resource allocation framework has been modified for our RAFC problem in this thesis.

2.3 Resource Allocation in Cloud Computing

In this section, we examine work on RACC, which is a subdomain of MARA and very similar to the RAFC problem we study in this thesis. For example, they all have heterogeneous resources such as CPU, RAM and storage, and users arrive over time to

request VM usage. We mainly focus on the Infrastructure-as-a-Service (IaaS) clouds in this section because the service model of fog computing in this thesis is IaaS (see Section 3.1). Here, IaaS is a service model that serve users by directly providing computer infrastructures. More specifically, cloud providers pack computational resources (e.g., CPU, RAM and storage) into VMs and provide these VMs to their users. Now, most IaaS cloud providers typically offer pre-configured VM instances of fixed types instead of dynamic VM instances with arbitrary VM configurations (Challenge 2). For example, Amazon EC2 currently offers five categories and 24 types of VMs, and each type has one or more instance sizes[1]. In terms of resource allocation mechanisms, most cloud providers, such as Amazon Web Services (AWS)[2], Microsoft Azure[3], and Google Cloud[4] provide long-term reservation plans or short-range on-demand fixed-price plans to their users. However, long-term reservation plans are obviously not suitable for users with uncertain tasks, which is common in IoT scenarios. Although fixed-price mechanisms are easy to implement and are obviously truthful, they are not very efficient in social welfare (Challenge 4) or revenue because they fail to discriminate different types of tasks (Al-Roomi *et al.*, 2013). For example, a high-value task would not get a high priority in a fixed-price mechanism. Furthermore, they also fail to cater to the volatility of the market. This leads to underpricing or overpricing. To improve efficiency in revenue, spot pricing is also adopted by these cloud providers such as Amazon EC2[5], Azure Low-Priority VM[6], and Google Preemptible VM Instances[7]. With spot pricing, a cloud user bids a price for its task, the task will run when the spot price is lower than its price, and the user pays the spot price. However, this mechanism has no guarantee of service-level agreement (SLA). Specifically, the spot price is volatile, and thus, tasks can be interrupted frequently, and there is no guarantee of the finish time of the task. In addition, the spot pricing mechanism is not truthful (Challenge 5) either (Wang *et al.*, 2012b).

2.3.1 Online Mechanisms for RACC

To address Challenge 6, there has been a lot of work on designing mechanisms for RACC in an online manner. For example, Wang *et al.* (2013); Zhang *et al.* (2016) propose online auctions for RACC, but they only consider VM instances of a single type and neglect the dynamic allocation of different VMs (Challenge 2). Moreover, a posted pricing mechanism is proposed by Zhang *et al.* (2017), under which the cloud provider publishes dynamic unit prices for different resource types, and cloud users either accept the current rates or give up running their jobs. A threshold-based mechanism is also proposed by

[1]https://aws.amazon.com/ec2/instance-types/
[2]https://aws.amazon.com/
[3]https://azure.microsoft.com/en-gb/
[4]https://cloud.google.com/
[5]https://aws.amazon.com/ec2/spot/pricing/
[6]https://azure.microsoft.com/en-gb/blog/low-priority-scale-sets/
[7]https://cloud.google.com/compute/docs/instances/preemptible

Farooq and Zhu (2018) to maximise the revenue of the cloud provider. In their model, the cloud provider has a limited number of VMs with different computational efficiency, and tasks with different complexity that arrive over time. However, these mechanisms do not work in a strategic setting (Challenge 5).

Therefore, there is a growing body of work looking at how to efficiently allocate cloud resources to users in an online and strategic setting, and many truthful mechanisms have been proposed in recent years. The early mechanisms only apply to very simple settings. For example, Wang *et al.* (2012a); Zhang *et al.* (2016) propose truthful mechanisms in the setting of single type VMs, and Lin *et al.* (2010) treat cloud computing resources as homogeneous. However, this is quite impractical because cloud providers usually have more than one type of VM and more than one type of resource (Challenge 2).

For this reason, dynamic virtual machine allocation where cloud providers can decide how to generate the type and number of VMs is also studied by some researchers. For example, Shi *et al.* (2014a) look at the setting of multi-type VMs and multi-type users. They use a non-decreasing pricing curve to decide the allocation of the resources and the payments of users and claim that their truthful mechanism RSMOA is both efficient in the social welfare of the system and the revenue of the cloud provider. Additionally, an online combinatorial auction framework for dynamic resource allocation in cloud computing is proposed by Shi *et al.* (2014b). This framework is able to model dynamic allocations of heterogeneous virtual machines and optimise system social welfare over a period of time. Primal-dual optimisation is used to transform a centralised approximation algorithm to a truthful auction mechanism. They showed that this auction framework is also computationally efficient, truthful and has a guaranteed competitive ratio [8]. However, their work does not study the temporal correlation in decision making, owing to the fact that tasks can span several time slots (Challenge 3).

Other researchers study truthful resource allocation mechanisms for tasks with deadlines (e.g., financial companies need data analysis results before the next market open time, or AVs need to identify the object in front of them in a very short time to stay safe). For instance, Lucier *et al.* (2013) propose two deadline-aware algorithms for homogeneous resource (i.e., CPU time) allocation in two settings. In one setting, the tasks can be paused and resumed (preemptible), but in the other setting, once a task is started, it must be processed until completion. They design a mechanism using the dual fitting technique for the first setting and prove its competitive ratio as well as showing that it significantly outperforms other heuristics used in practice. For the other setting, they prove that no performance guarantee can be given, and propose an efficient heuristic called Committed for resource allocation. Finally, they prove that their Committed algorithm can be made truthful very easily. Additionally, preemptive scheduling with the assumption of deadline slackness (i.e., the lower bound on the ratio between the

[8] Here, the competitive ratio is the worst ratio between the social welfare incurred by an online mechanism and the optimal social welfare.

requested time of a task and the time window in which it can be processed) has also been studied (Azar *et al.*, 2015). A truthful online mechanism with a guaranteed competitive ratio for the overall utility of jobs is presented in their work, and another truthful online mechanism is developed that can commit whether a job will complete before its deadline if the deadline slackness is large enough. In similar work (Zhou *et al.*, 2017), cloud users not only specify their deadlines of the job but also provide penalty functions for the situation that the deadlines are violated. An auction framework with posted prices is used to incentivise truthful reports, and a new technique of compact exponential-size LPs coupled with dual separation oracles is introduced to deal with the soft deadline constraints. Moreover, the classic primal-dual framework is used to develop a social welfare approximation algorithm, and their mechanism is shown to be efficient by simulations using Google cluster data.

However, all the above work considers the setting of cloud computing, in which a single centralised data centre provides resources, so their techniques do not deal with bandwidth resources (Challenge 1) or dynamic VM allocation (Challenge 2). Furthermore, they do not consider the operational cost of cloud providers, which is also different from our setting (Challenge 7).

2.4 Resource Allocation in Fog Computing

Although fog computing is a relatively new concept, some research has been carried out on resource allocation in this domain. As mentioned in Section 1.1, the fog computing structure is three-tier, which consists the IoT devices, the fog, and the cloud. Some work on resource allocation only considers the fog (Oueis *et al.*, 2015; Aazam and Huh, 2015), while others consider the fog and IoT devices (Zeng *et al.*, 2016) or the fog and cloud simultaneously (Deng *et al.*, 2015; Agarwal *et al.*, 2016). However, we only focus on the resource allocation in the fog in this thesis.

2.4.1 Online Mechanisms for RAFC

In this section, we discuss work that considers RAFC in an online setting (Challenge 6). Aazam and Huh (2015) introduce an efficient resource allocation framework which predicts and reserves resources based on users' historical behaviour and offers prices based on users' characteristics. Their simulations show that their framework can allocate resources adaptively and avoid resource wastage. For joint resource allocation in fog and cloud computing, Agarwal *et al.* (2016) design an efficient architecture and algorithm in terms of the total cost, processing time, overall response time and data transferred over the Internet. In their architecture, a fog provider receives all the requests from fog users and decides whether to process them locally or to send some of them to the

cloud depending on its available resources. Moreover, in the work from Alsaffar *et al.* (2016), an architecture of IoT resource allocation in a hybridisation of cloud and fog computing systems is proposed. Apart from the architecture, efficient algorithms have been developed to allocate resources, balance workload and distribute data among the system. However, the above work does not consider the strategic behaviour of agents (Challenge 5). Thus, in a strategic setting where agents act according to their utilities, these mechanisms can no longer guarantee their efficiency.

Although there is little work that directly studies the truthful online mechanism for RAFC, some literature has looked at related domains such as distributed cloud computing and edge computing, which have similar resource allocation models. First of all, we discuss the work on resource allocation in distributed clouds. Similar to RAFC, resource allocation mechanisms in distributed clouds need to decide not only when but also where to put the VMs (Challenge 2) and the bandwidth between VMs (Challenge 1). Shi *et al.* (2015) have studied the problem of bandwidth allocation in a distributed cloud. They developed truthful offline and online mechanisms from Shapley value based auctions. Although real data simulations show the efficacy of their mechanisms, their mechanism is only for bandwidth allocation. Moreover, Zhang *et al.* (2015a) design truthful online auctions where users bid for VMs for a fixed time in the future for social welfare and revenue maximisation. This work used an online primal-dual optimisation framework to maximise the social welfare and used a randomised reduction algorithm to convert the social welfare maximisation auction to a revenue maximisation one. They showed that their mechanisms are polynomial-time and have better performance than existing ones using Google cluster data (Reiss *et al.*, 2011). However, they did not consider the bandwidth between servers and between server and agents (Challenge 1), and their agents are not multi-minded (Challenge 3).

Furthermore, two truthful online mechanisms are proposed by Shi *et al.* (2017) for dynamic virtual cluster (VC) provisioning. VCs are assemblies of several VMs and the communication resources (bandwidths) between them. They designed the Social Welfare Maximizing Online Mechanism (SWMOA) to maximise the social welfare and Provider Revenue Maximization Online Auction (PRMOA) to maximise the cloud providers' revenue. The main idea of SWMOA is to maintain a dynamic virtual unit cost for each type of resource and accept a requested VC scheme (i.e., the placement of VMs) only if its total virtual cost is less than its valuation. PRMOA, on the other hand, first determines a provisional VC allocation and payments for users using SWMOA and then charges each accepted bid with a randomised boosted payment, which is still below its valuation, to increase the cloud providers' revenue. However, under their mechanisms, cloud users need to specify their desired schemes of virtual cluster placement, which is not practical because cloud users usually have no knowledge about the following things: the physical topology of the distributed cloud and the desirable VC schemes.

2.5 Summary

In this chapter, we have presented the concepts and theories in game theory and mechanism design as well as the literature that is related to our work. As discussed beforehand, the research community has proposed many effective approaches for RAFC and other related resource allocation problems. However, they only address a subset of our challenges.

Firstly, we examined related knowledge in game theory and mechanism design as a background in Section 2.1. In particular, we introduced important concepts such as game, solution concept, mechanism, and we gave the definition of a special type of mechanism called direct-revelation truthful mechanisms. Then we showed that we can just focus on this type of mechanisms because of the revelation principle. Furthermore, we listed desirable properties for our mechanism in this thesis, such as truthfulness, IR and WBB.

Secondly, we presented background and related work in general MARA in Section 2.2. In the background part, we introduced the general model of MARA and different types of utility functions, objective functions and resource allocation mechanisms and clarified, which type our work focuses on. Then we examined online mechanisms proposed for MARA. Although they address parts of our challenges in isolation, they typically fail to consider heterogeneous resources (Challenge 1 and Challenge 2).

Finally, we examined the related work in RACC or RAFC, which also addresses a subset of our challenges. The main problem of work in RACC is that it usually does not deal with bandwidth resources (Challenge 1) or location-aware VM allocation (Challenge 2). Some work in RAFC considers most of our challenges, but they still have some deficiencies like requiring users to specify all the details of the allocation. Hence, no work has been done to design a truthful online mechanism that addresses all our challenges in RAFC. This is the research gap that we intend to bridge.

In particular, some insights from the examined work are integrated into our work. Firstly, we adapt the price-based mechanisms (Hayakawa *et al.*, 2018) to our RAFC problem and design our mechanism belonging to this class of mechanisms because price-based mechanisms are guaranteed to be DSIC and IR. Secondly, our mechanism is based on the greedy allocation (Gerding *et al.*, 2011), which means that at each time step resources are allocated to maximise the marginal valuations. Finally, the pre-commitment technique (Stein *et al.*, 2012) is used in our work to improve the efficiency of our mechanism.

To evaluate our work, we choose four benchmark mechanisms: the offline optimal mechanism represents the upper limit of social welfare efficiency; the offline optimal indicates the social welfare could be achieved if all users report truthfully; the online greedy

algorithm is a simple heuristic truthful mechanism; SWMOA2 is a variant of a state-of-the-art truthful mechanism called SWMOA (Shi *et al.*, 2017). The details of these benchmarks are in Section 3.3.

Chapter 3

Problem Model and Resource Allocation Mechanisms

In this chapter, we present our work on designing a truthful online mechanism for resource allocation in fog computing (RAFC), which addresses our Challenges 1-7 in Section 1.4. In Section 3.1, we describe the model of our RAFC problem. Then, in Section 3.2, we introduce a class of truthful online resource allocation mechanisms call price-based mechanisms to which our proposed mechanism belongs. After that, we present the algorithms of benchmark mechanisms in Section 3.3 and the algorithm and properties of our mechanism in Section 3.4. Finally, we summarise the whole chapter in Section 3.5.

3.1 Model of RAFC

In this section, we present the RAFC model used in this thesis. We first give a brief overview of this model in Section 3.1.1, and then formally describe it in detail in Section 3.1.2.

3.1.1 Overview of the Model

In our model, a single fog provider owns a fog computing system with several geographically distributed fog nodes (FNs) and data links interconnecting them, as shown in Figure 3.1. Here, FNs have computational resources (such as CPU, RAM and storage) and data links have bandwidth resources, and different resources have different fixed operational costs. The operational costs of computational resources comprise electricity costs and the depreciation charge of the resources, and the operational costs of bandwidth are the costs charged by Internet service providers (ISPs). FNs and data links

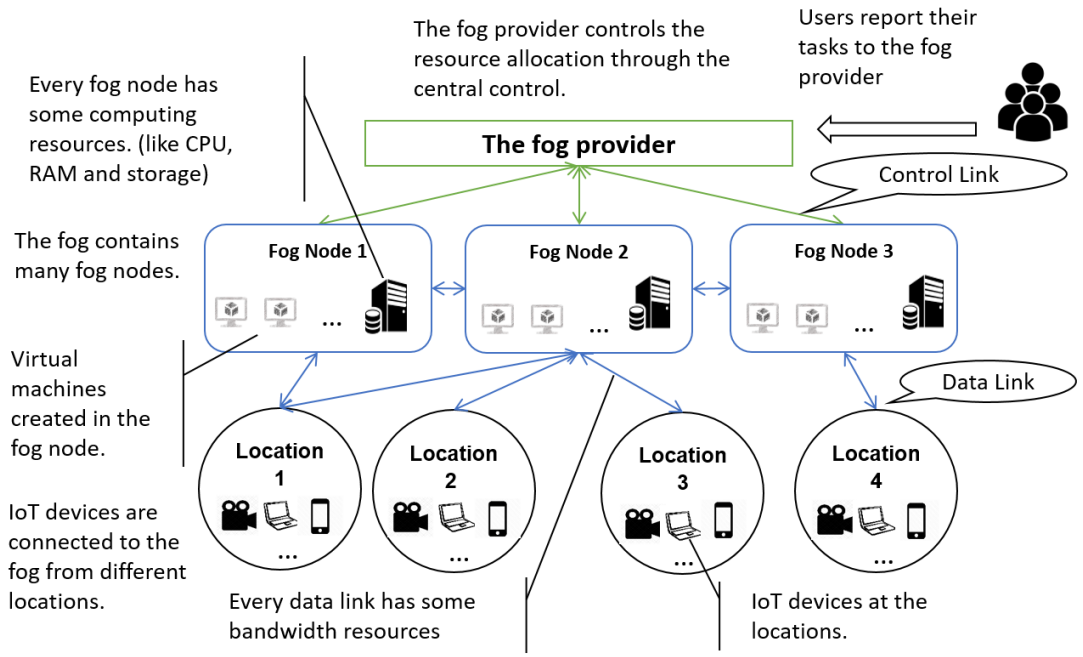

FIGURE 3.1: General view of a fog computing system.

together offer these resources to satisfy the needs of fog users (i.e., agents in the previous chapter) by means of processing tasks using VMs. IoT devices at different locations are connected to FNs of the fog through data links. They can be stationary IoT devices in the fog computing system or portable IoT devices carried by fog users. For example, stationary IoT devices can be smart TVs, surveillance cameras and smart speakers. While IoT devices carried by users can be smartphones and AVs. Furthermore, the fog provider controls the resource allocation of the fog through a central point of control and control links. More specifically, the central point of control is a server that receives reports of tasks from fog users and decides how to allocate resources to satisfy these users, and these decisions are sent to FNs to execute through control links.

Another essential part of our model are fog users. They report their tasks (e.g., a video surveillance task or a picture processing task) to the fog provider through their IoT devices over time (Figure 3.2), which includes the resource requirements, the time constraints and the valuation functions of the tasks. For example, a picture processing application can have computing tasks like adding filters or compressing photos. To finish a task of adding filters to 100 photos, bandwidth resources are needed to send the images to and from the FN, and computational resources are needed to process the image. The time constraints of this task could be like this: the task can start right now, and the result should be sent to the user in 30 seconds because users usually do not want to wait for too long to add a filter to their photos. If the task is completely finished before the deadline, the user can get all the value of this task. However, if filters have only been

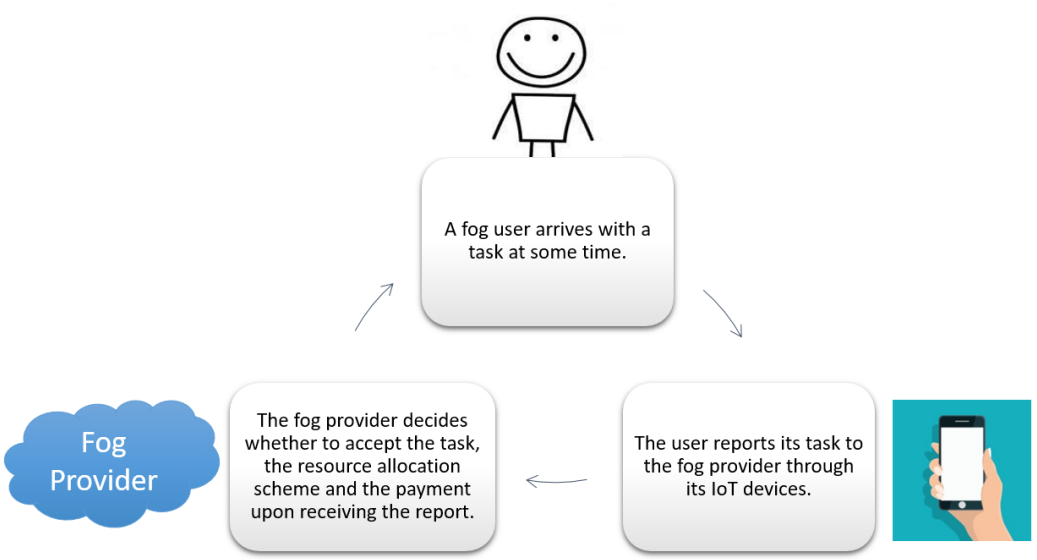

FIGURE 3.2: The procedure of RAFC.

added to 50 photos at the deadline, it is reasonable that the user can still get part of the value of this task.

When receiving the reports from the applications, the fog provider decides whether to accept them, how to allocate resources to satisfy the demands of the accepted tasks and how much is the corresponding payment through an online mechanism (Figure 3.2). Finally, the social welfare of the allocation is the sum of value fog users get by processing tasks minus the sum of the fog provider's operational costs, while the revenue is the sum of the fog provider's income minus its operational costs. We focus on social welfare because we assume that the fog provider is a non-profit organisation, and improving overall social welfare is its primary objective. Therefore, we leave the objective of maximising the fog provider's revenue to future work.

3.1.2 Formal Model

We now present the model described in Section 3.1.1 in detail and formally. Consider a fog provider with a set W of geo-distributed FNs and a set L of locations, which are interconnected through a set $\mathbb{E}$ of data links, as shown in Figure 3.1.

Furthermore, there is a set E_l of IoT devices in each location l, and $e \in E_l$ is an IoT device (e.g., a smart TV, surveillance camera, smart speaker or AV). Every FN $w \in W$ has a set R of limited computational resources (e.g., CPU, RAM and storage). Moreover, there are $A_{w,r}$ units of type $r \in R$ resources in FN w, and the unit operational

cost of resource r in FN w is $o_{w,r}$. In addition, the bandwidth capacity and the unit operational cost of link $(j,k) \in \mathbb{E}$ are $b_{j,k}$ and $o_{j,k}$ respectively. For simplicity, we assume that the bandwidth capacity and unit bandwidth costs are symmetrical for all links (i.e., $b_{j,k} = b_{k,j},\ o_{j,k} = o_{k,j},\ \forall (j,k) \in \mathbb{E}$). FNs and data links together offer their resources to satisfy the needs of fog users. In particular, we assume that VMs can be created in an FN to run computing tasks as long as there are enough computational resources in that FN, and the total resource requirements of several virtual machines are just the sum of their individual resource requirement for simplicity, although, in reality, they may need fewer resources because they can share resource with each other. Furthermore, the fog provider uses a centralised online resource allocation mechanism to make resource allocation decisions.

Fog users with tasks arrive over time, and I denotes the set of all tasks. Note that we adopt a continuous time system, but the tasks can only start execution at discrete time steps, denoted by the set $T = \{1, 2, \ldots, |T|\}$. Each task $i \in I$ is owned by a user, which is also denoted as i for simplicity because we assume that each user has one task. In addition, the arrival time of task i is $T_i^a \in [0, |T|]$, which is the time when user i becomes aware of its task i, and the time window that the task can be processed is from its start time T_i^s to its deadline T_i^d. Here, we assume that no tasks arrive at the exact same time. User i reports its task's type $\hat{\theta}_i$ (as defined in the following) at time $\hat{T}_i^a$ to run a certain application (e.g., a video surveillance application or a picture processing application). We assume that user i wants to know the number of time steps $\tilde{t}_i$ it will get and the payment p_i for its task also by time $\hat{T}_i^a$ because users want to run the tasks locally or elsewhere if their tasks get rejected. The operational cost of task i is denoted as o_i, which is the sum of costs of all resources allocated to task i, including the cost of bandwidth. Furthermore, we also assume that every task only requires one VM to run but may require connections to several endpoints $e \in E$ (in the same location or in different locations) because this is common in an IoT system (Du *et al.*, 2018). Users are also assumed to be stationary, which means that the endpoints of users do not change locations over time. Furthermore, we also assume VMs can migrate between FNs and the migration costs are negligible, and all tasks are preemptible, which means that they can always be paused and resumed. Finally, we focus on one type of task called time-oriented tasks (e.g., video surveillance and video processing tasks), which are common in fog computing. Such a task i needs a certain configuration of resources for a time length t_i to get its full value, but can still get part of the value if the processing time is less than t_i. Formally, the type of task i is a tuple $\theta_i = (T_i^a, T_i^s, T_i^d, v_i, \{a_{i,r}\}_{r \in R}, \{\Gamma_l^i\}_{l \in L})$, where $a_{i,r}$ denotes the amount of resource $r \in R$ required, and Γ_l^i denotes the bandwidth demand between its VM and location $l \in L$. For simplicity, bandwidth demands are symmetrical. That is, Γ_l^i denotes both the bandwidth demands to and from location $l \in L$. In this paper, the valuation function is given by $v_i = \{v_{i,0}, v_{i,1}, \ldots, v_{i,t_i}\}$, where $v_{i,t}$ is the value when task i gets a usage time of t time steps within its time window ($[T_i^s, T_i^d]$) and t_i denotes the usage time needed to get the full value of the task. We

make the reasonable assumption that the valuation function monotonically increases with usage time (i.e., $v_{i,t''} \geq v_{i,t'} \quad \forall t'' \geq t'$). For example, suppose a user wants to run a real-time video analytics application with facial recognition to surveil their shops for 24 hours. It is intuitive that they will not get additional value for a surveillance time of more than 24 hours, and it is still of value to them if the surveillance lasts less than 24 hours, say, 18 hours. We choose this type of valuation function because it corresponds to many applications in the fog, which achieve better results as processing time increases. Moreover, the reported type of task i is a tuple $\hat{\theta}_i = (\hat{T}_i^a, \hat{T}_i^s, \hat{T}_i^d, \hat{v}_i, \{\hat{a}_{i,r}\}_{r \in R}, \{\hat{\Gamma}_l^i\}_{l \in L})$, and $\hat{\theta}^{\langle t \rangle}$ denotes the set of all reported types until and including time step t.

Now, a key assumption in our work is that users are strategic, so $\hat{\theta}_i$ may not be the same as θ_i. Moreover, we assume limited misreports (see Section 2.1.3.1) based on the nature of our problem (i.e., $\hat{T}_i^a \geq T_i^a, \hat{T}_i^s \geq T_i^s, \hat{T}_i^d \leq T_i^d, \hat{a}_{i,r} \geq a_{i,r}\, r \in R, \hat{\Gamma}_l^i \geq \Gamma_l^i\, l \in L$). This is reasonable because a user cannot report a task before it becomes aware of it (i.e., $\hat{T}_i^a < T_i^a$), and cannot report a looser time window (i.e., $\hat{T}_i^s < T_i^s$ or $\hat{T}_i^d > T_i^d$) because the fog provider can check whether task i is ready to be processed at $\hat{T}_i^s$ and withhold the results for i until $\hat{T}_i^d$. So reporting $\hat{T}_i^s < T_i^s$ will be detected and penalised by cancelling the task and reporting $\hat{T}_i^d > T_i^d$ will result in no value. Finally, user i will not misreport a lower resource requirement because its task cannot be processed in that case. However, for some tasks, the results cannot be withheld until the deadline of the tasks such as video surveillance tasks and autopilot tasks. This situation is discussed in Section 3.4.1.

Next, when receiving the bid $\hat{\theta}_i$ for task i, the fog provider will decide the resource allocation scheme λ_i, which is a tuple formally defined in the later part of this paragraph, right away to this task including how much usage time $\tilde{t}_i$ will be allocated, and the payment p_i because of the assumption we made earlier that all users want to know the allocation results at their arrival times. To provide an upper bound on the social welfare, we find the optimal social welfare in an offline setting by solving a constraint optimisation problem, and the decision variables are: (1) $\{z_{w,t}^i \in \{0,1\}\}_{i \in I, w \in W, t \in T}$, indicating that the VM of task i is placed in FN w ($z_{w,t}^i = 1$), or not ($z_{w,t}^i = 0$) at time step t. (2) $\{f_{l,w,j,k,t}^i \in \mathbb{R}^+\}_{i \in I, l \in L, w \in W, (j,k) \in \mathbb{E}, t \in T}$, indicating allocation of the bandwidth on link (j,k) for traffic from location l to FN w for task i at time step t. So, for task i, its usage time $\tilde{t}_i = \sum_{w \in W, t \in T} z_{w,t}^i$ and resource allocation scheme $\lambda_i = (\{z_{w,t}^i\}_{i \in I, w \in W, t \in T}, \{f_{l,w,j,k,t}^i\}_{i \in I, l \in L, w \in W, (j,k) \in \mathbb{E}, t \in T})$. The objective function (3.1a) of this optimisation problem maximises the social welfare, which is subject to resource and time

constraints:

$$\max_{\lambda_i} \quad \sum_{i\in I} v_i\Big(\sum_{w\in W,t\in T} z^i_{w,t}\Big) - \Big(\sum_{i\in I,r\in R,w\in W,t\in T} a_{i,r} z^i_{w,t} o_{w,r} + \sum_{i\in I,l\in L,w\in W,(j,k)\in\mathbb{E},t\in T} 2o_{j,k} f^i_{l,w,j,k,t}\Big) \tag{3.1a}$$

$$\text{s.t.} \quad \sum_{w\in W} z^i_{w,t} \le 1 \quad \forall i \in I, t \in T \tag{3.1b}$$

$$\sum_{i\in I} z^i_{w,t} a_{i,r} \le A_{w,r} \quad \forall w \in W, r \in R, t \in T \tag{3.1c}$$

$$z^i_{w,t} = 0 \quad \forall i \in I, w \in W, t < T^s_i \; or \; t > T^d_i \tag{3.1d}$$

$$\sum_{j:(j,w)\in\mathbb{E}} f^i_{l,w,j,p,t} = \Gamma^i_l z^i_{w,t} \quad \forall w \in W, i \in I, l \in L, t \in T \tag{3.1e}$$

$$\sum_{k:(l,k)\in\mathbb{E}} f^i_{l,w,l,k,t} = \Gamma^i_l z^i_{w,t} \quad \forall w \in W, i \in I, l \in L, t \in T \tag{3.1f}$$

$$\sum_{j:(j,k)\in\mathbb{E}} f^i_{l,w,j,k,t} = \sum_{j:(k,j)\in\mathbb{E}} f^i_{l,w,k,j,t} \quad \forall w \in W, k \in W, i \in I, l \in L, t \in T \tag{3.1g}$$

$$\sum_{i\in I} f^i_{l,w,j,k,t} \le b_{j,k} \quad \forall (j,k) \in \mathbb{E}, t \in T \tag{3.1h}$$

$$f^i_{l,w,j,k,t} \ge 0 \quad \forall i \in I, l \in L, w \in W, (j,k) \in \mathbb{E}, t \in T \tag{3.1i}$$

To explain the above constraints in detail, constraint (3.1b) represents that every task only needs one VM. Constraint (3.1c) guarantees that at each time step the allocated resources at any FN do not exceed its resource capacities. Constraint (3.1d) means that the VM is created within the start time and deadline of the task. Constraint (3.1e) requires that the total inbound traffic to FN w for the traffic from task i's location l equals its corresponding bandwidth demand at each time step if the VM for task i is placed in FN w. Since the bandwidth demands and the bandwidth costs are both symmetrical. It is sufficient to just consider the traffic from L to W. Constraint (3.1f) indicates that the outbound traffic from task i's location l is equal to its corresponding bandwidth demand at each time step. Constraint (3.1g) represents that the inbound and outbound traffic of intermediate nodes for task i should be equal. Constraint (3.1h) guarantees that the aggregated traffic on each link does not exceed its bandwidth capacity at each time step. Finally, constraint (3.1i) ensures that the allocated bandwidth in each data link is not negative, which is impossible in practice.

This is a mixed integer linear programming problem (MILP) because the objective function and the constraints are linear and one of the decision variables $z^i_{w,t}$ is discrete. Unfortunately, this problem is NP-hard, which can be proved by reducing a 0-1 knapsack problem to it (see detailed proof below). In practice, the optimisation problem (3.1a) can be solved using linear programming solvers. In particular, we use the IBM ILOG CPLEX Optimization Studio to solve it. However, solving this problem can be time-consuming because the problem is hard per se.

Theorem 3.1. *The optimisation problem (3.1a) is NP-hard.*

Proof. Suppose we have a 0-1 knapsack problem with a maximum weight capacity A_w and $|I|$ items with weights $\{a_i\}_{i\in I}$ and values $\{v_i\}_{i\in I}$. We can design a fog computing resource allocation problem as follows. There is one FN with CPU resource A_w and no other resources. All $|I|$ tasks arrives at $t = 0$ and task i requires only a_i CPU resource with start time $T_i^s = 0$, deadline $T_i^d = 1$, and value $v_i, i \in I$ for $t_i = 1$. Then, we can solve the above 0-1 knapsack problem by just solving the above fog computing resource allocation problem. Ergo, the offline optimal problem is at least as hard as the 0-1 knapsack problem. Since the optimisation problem of a 0-1 knapsack problem is NP-hard, the optimisation problem (3.1a) is NP-hard too. □

Finally, mechanisms need to make both resource allocation decisions and payment decisions for fog users. We use $p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle}) \in \mathbb{R}^+$ to denote the payment of task i, which is a function of the allocation (λ_i) and all information received by $\hat{T}_i^a$ ($\hat{\theta}^{\langle \hat{T}_i^a \rangle}$). Thus the utility of user i is $u_i = v_i(\tilde{t}_i) - p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle})$, and this is what user i tries to maximise.

3.2 Price-based Mechanisms

First, we introduce a class of online resource allocation mechanisms called price-based mechanisms that guarantee DSIC and IR for our resource allocation problem , and it includes our proposed mechanism. Note that this class of mechanisms is an adaptation from the price-based mechanisms characterised by Hayakawa *et al.* (2018). In the following, we characterise the properties that this class of mechanisms should have, in order to guarantee DSIC and IR.

Definition 3.2. A monotonic payment function is (weakly) monotonically increasing over $\hat{T}_i^a$, $\hat{T}_i^s$, $\hat{t}_i$, $\hat{a}_{i,r}, r \in R$ and $\hat{\Gamma}_l^i, l \in L$, and (weakly) monotonically decreasing over $\hat{T}_i^d$.

Then, we define the class of price-based mechanisms for our RAFC problem as follows:

Definition 3.3. An online mechanism belongs to the class of price-based mechanisms if it has the following properties:

1. The mechanism computes the payment p_i for any possible allocation λ_i to task i by using a payment function $p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle})$ that is independent of $\hat{v}_i$ and monotonic.
2. The resource allocation scheme λ_i for task i maximises $\hat{v}_i(\lambda_i) - p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle})$ (over all λ_i that can be made to task i for any $\hat{v}_i$).
3. The payment for tasks with no resource allocated is zero.

Then, the following theorem guarantees that any mechanism in the class of price-based mechanisms is DSIC and IR.

Theorem 3.4. *Any online mechanism that satisfies Definition 3.3 is DSIC and IR.*

Proof. The proof is shown below. □

To prove a mechanism that satisfies Definition 3.3 is DSIC, we first use the sufficient and necessary characterisation of incentive compatible mechanisms for a setting where users can only misreport their valuation function (i.e., where the misreports of $\hat{T}_i^a$, $\hat{T}_i^s$, $\hat{T}_i^d$, $\hat{t}_i$, $\{\hat{a}_{i,r}\}_{r\in R}$ and $\{\hat{\Gamma}_l^i\}_{l\in L}$ are not considered) proposed by Bartal *et al.* (2003) .

Lemma 3.5. *In our setting, assuming the parameters $\hat{T}_i^a$, $\hat{T}_i^s$, $\hat{T}_i^d$, $\hat{t}_i$, $\{\hat{a}_{i,r}\}_{r\in R}$, $\{\hat{\Gamma}_l^i\}_{l\in L}$ in user i's bid $\hat{\theta}_i$ are truthful, a direct revelation mechanism is DSIC if and only if*

1. *The payment function $p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle})$ is computed for every possible allocation $\tilde{t}_i(\lambda_i)$ for task i and does not depend on $\hat{v}_i$.*
2. *The allocation function allocates $\tilde{t}_i$ for task i such that the value of $\hat{v}_i(\tilde{t}_i) - p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle})$ is maximised (over all $\tilde{t}_i$ that can be allocated to i for any choice of $\hat{v}_i$).*

Lemma 3.5 is a simple extension of Theorem 1 in (Bartal *et al.*, 2003).

Then, we use this Lemma to prove the following theorem that is in settings where users can misreport $\hat{T}_i^a, \hat{T}_i^s, \hat{T}_i^d$, $\hat{t}_i$, $\{\hat{a}_{i,r}\}_{r\in R}$, $\{\hat{\Gamma}_l^i\}_{l\in L}$ besides $\hat{v}_i$ (assuming limited misreports).

Theorem 3.6. *In our setting, a direct revelation mechanism is DSIC if and only if*

1. *The payment function $p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle})$ for every possible allocation $\tilde{t}_i(\lambda_i)$to task i is monotonic and does not depend on $\hat{v}_i$.*
2. *The allocation function allocates $\tilde{t}_i$ for task i such that the value of $\hat{v}_i(\tilde{t}_i) - p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}_i^a \rangle})$ is maximised (over all $\tilde{t}_i$ that can be allocated to i for any choice of $\hat{v}_i$).*

Proof. At first, we show that the conditions in Theorem 3.6 are sufficient. According to Lemma 3.5, a user i cannot increase its utility by manipulating $\hat{v}_i$. Therefore, we can assume that it truthfully reports its valuation coefficient $\hat{v}_i$, and the only way to increase its utility is by decreasing the payment function. Since the payment function is monotonic, only misreporting $\hat{T}_i^a < T_i^a, \hat{T}_i^s < T_i^s, \hat{T}_i^d > T_i^d$, $\hat{t}_i < t_i$, $\hat{a}_{i,r} < a_{i,r}, r \in R$ or $\hat{\Gamma}_l^i < \Gamma_l^i, l \in L$ can reduce it. First of all, misreporting $\hat{T}_i^a < T_i^a, \hat{T}_i^s < T_i^s, \hat{T}_i^d > T_i^d$ is impossible because the limited misreports assumption we made earlier. Then,

misreporting $\hat{t}_i < t_i$ cannot increase u_i because this only reduces the domain of $\tilde{t}_i$ and the allocation function allocates $\tilde{t}_i$ to maximise u_i (condition two of the above theorem). Finally, misreporting $\hat{a}_{i,r} < a_{i,r},\ r \in R$ or $\hat{\Gamma}^i_l < \Gamma^i_l\ l \in L$ will lead to the failure of the task, which reduces u_i to negative. Hence, user i has no incentive to submit a non-truthful bid (i.e., where $(\hat{T}^a_i, \hat{T}^s_i, \hat{T}^d_i, \hat{v}_i, \{\hat{a}_{i,r}\}_{r\in R}, \{\hat{\Gamma}^i_l\}_{l\in L}) \neq (\ T^a_i, T^s_i, T^d_i, v_i, \{a_{i,r}\}_{r\in R}, \{\Gamma^i_l\}_{l\in L})$).

Then, we show that the conditions in Theorem 3.6 are also necessary. We first assume to the contrary that the first condition does not hold, i.e., the payment function is not independent of $\hat{v}_i$ or the payment function is not *monotonic*. In the former case, the mechanism is not DSIC according to Lemma 3.5. In the latter case, that is, there is some $T^{a\prime}_i < T^{a\prime\prime}_i$ such that the resource allocation is the same ($\lambda' = \lambda''$) but the payment satisfies $p_i(\lambda', T^{a\prime}_i) > p_i(\lambda'', T^{a\prime\prime}_i)$, while $\hat{T}^s_i$ and $\hat{T}^d_i$ remain unchanged. On this occasion, a user whose true arrival time is $T^{a\prime}_i$ and who gets allocation λ' when reporting truthfully is incentivised to misreport $T^{a\prime}_i$ as $T^{a\prime\prime}_i$ because it can get the same allocation with less payment. Since this is contrary to the definition of DSIC, the first condition must be necessary.

Then, we assume that the first condition holds, but the second condition does not. For instance, for some user i with $v_i = v'_i$, there exists v''_i, the mechanism allocates λ'_i and λ''_i respectively such that $v'_i(\tilde{t}'_i) - p_i(\lambda'_i, \hat{\theta}^{\langle \hat{T}^a_i \rangle}) < v''_i(\tilde{t}''_i) - p_i(\lambda''_i, \hat{\theta}^{\langle \hat{T}^a_i \rangle})$. On this occasion, this user is incentivised to misreport v'_i as v''_i. Hence, the second condition is also necessary.

Finally, a mechanism that satisfies the Definition 3.3 is also IR for the following reasons. Since user i will always bid truthfully (i.e., $\hat{v}(t) = v(t)$), the final allocation actually maximises the utility of user i: $u_i = v_i(\tilde{t}_i) - p_i$. In addition, the maximum of u_i should be greater or equal to zero as i can always get a utility of zero with no resource allocated according to condition 3 in Definition 3.3. From the above discussion, it is clear that i will never get a negative utility under such mechanisms.

From the above, Theorem 3.6 is proven, and so is Theorem 3.4. □

3.3 Resource Allocation Benchmark Mechanisms

Against this background, we present the benchmark mechanisms and the mechanism we proposed for RAFC. We choose four benchmark mechanisms: the offline optimal mechanism represents the upper bound of social welfare efficiency; the online optimal mechanism is a representative online heuristic mechanism; the online greedy mechanism is a representative truthful online heuristic mechanism; and Social Welfare Maximisation Online Auction 2 (SWMOA2) is a variant of a state-of-the-art truthful online mechanism called Social Welfare Maximisation Online Auction (SWMOA) (Shi *et al.*, 2017).

3.3.1 Offline Optimal Mechanism

Under this mechanism, we assume that we know all the information about future tasks and allocate resources to optimise the social welfare with no need to incentivise fog users to report their tasks truthfully. This theoretical and idealised case can be achieved by solving the constraint optimisation problem 3.1a in Section 3.1.2.

3.3.2 Online Optimal Mechanism

This mechanism is similar to the offline optimal mechanism except that the optimisation problem is solved at each time step with knowledge only of the tasks that have arrived so far (and not of future tasks). Note that this mechanism is not truthful, but we use this to determine the social welfare that could be achieved in an online setting if all users report truthfully. In Section 4.2.4, we also evaluate this mechanism's performance in social welfare in settings where part of the users misreport. The details of this mechanism is given in Algorithm 1 below.

3.3.3 Online Greedy (OG) Mechanism

This mechanism is an extension of the greedy algorithm from Wang *et al.* (2012b), which greedily make an allocation decision to maximise the utility of a task when it arrives and commits to the decision henceforth. Furthermore, it computes the payment as this task's corresponding operational costs ($p_i = o_i = \sum_{r\in R, w\in W, t\in T}(a_{i,r} z^i_{w,t} o_{w,r}) + \sum_{l\in L, w\in W,(j,k)\in\mathbb{E},t\in T}(2o_{j,k} f^i_{l,w,j,k,t})$). This mechanism is DSIC and IR, and the details of this mechanism are given in Algorithm 2:

Theorem 3.7. *The online greedy mechanism is DSIC, IR and WBB.*

Proof. Under the online greedy mechanism, any user who gets allocated nothing has to pay zero because the corresponding operational cost is zero. So it satisfies the third condition of Definition 3.3. Furthermore, for every possible $\tilde{t}_i$ allocated to task i, the mechanism chooses the allocation that has the lowest o_i according to $\hat{\theta}_i$. This is because $\hat{v}_i(\tilde{t}_i)$ is independent of the allocation details, and the mechanism maximises $\hat{v}_i(\tilde{t}_i) - o_i$. Since $p_i = o_i$, the payment p_i is also independent of $\hat{v}_i$. In addition, increasing $\hat{T}^a_i, \hat{T}^s_i$ or decreasing $\hat{T}^d_i$ can only increase o_i by reducing the space of possible allocations (i.e., reducing the available time steps (increasing $\hat{T}^s_i$ or decreasing $\hat{T}^d_i$) or causing more resource to be allocated to other users (increasing $\hat{T}^a_i$)), and increasing $\hat{t}_i$, $\{\hat{a}_{i,r}\}_{r\in R}$ or $\{\hat{\Gamma}^i_l\}_{l\in L}$ can only increase o_i too because this increases the resource demands. Due to the fact that $p_i = o_i$, $p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}^a_i \rangle})$ is monotonic according to Definition 3.2. Hence, the mechanism satisfies the condition 1 in Definition 3.3. Finally, the mechanism also

Algorithm 1: The online optimal mechanism

$\theta_{arrived} \leftarrow \emptyset$ ▷ The set of arrived tasks
$\theta_{flex} \leftarrow \emptyset$ ▷ The set of flexible tasks
for t *in* T **do**
  while *new tasks arrive within* t **do**
    When a new task i arrives ▷ Tasks arrive over time
    $\theta_{arrived} \leftarrow \theta_{arrived} \cup \{i\}$ ▷ Update the set of arrived tasks
    $\theta_{flex} \leftarrow \theta_{flex} \cup \{i\}$ ▷ Update the set of flexible tasks
  end
  Solve the maximum utility allocation for tasks in θ_{flex} (i.e., $\arg\max_{\lambda_j} \sum_{j \in \theta_{flex}} (\hat{v}_j(\lambda_j) - o_j(\lambda_j))$) ▷ Find the allocation for tasks in θ_{flex} that maximise their social welfare
  $p_i \leftarrow 0$ ▷ Payment for task i is zero
  for i *in* θ_{flex} **do**
    Allocate resources for the next time step $(t+1)$ according to λ_i
    $t_i \leftarrow t_i - \sum_{w \in W} z^i_{w,t+1}$ ▷ Update the remaining processing time of task i
    for $j = 1; j \leq t_i; j{+}{+}$ **do**
      ▷ Update the valuation function of task i
      $\hat{v}_i(j) \leftarrow \hat{v}_i(j + \sum_{w \in W} z^i_{w,t+1})$
    end
    if $t_i = 0$ **then**
      $\theta_{flex} \leftarrow \theta_{flex} \setminus \{i\}$ ▷ Delete task i from flexible tasks if it gets its required usage time
    end
    if $t = \hat{T}^d_i$ **then**
      $\theta_{flex} \leftarrow \theta_{flex} \setminus \{i\}$ ▷ Delete task i from flexible tasks if it reaches its deadline
    end
  end
end

satisfies the condition 2 in Definition 3.3 because it decides the allocation λ_i that maximise $\hat{v}_i(\tilde{t}_i) - p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}^a_i \rangle})$ so the $\tilde{t}_i$ incurred by λ_i also maximises the utility of user i. Taken together, the online greedy mechanism is DSIC and IR according to Theorem 3.4. In addition, because the payment of task i equals to the operational cost of that task ($p_i = o_i$), the total sum of payments equal the total operational cost of the fog ($\sum_{i \in I} p_i = o$). Thus, OG satisfies WBB. □

3.3.4 SWMOA2 Mechanism

Although the SWMOA mechanism from Shi *et al.* (2017) cannot be directly applied to our model, we develop a variant of it called SWMOA2 as a suitable benchmark. The main difference between this mechanism (given in Algorithm 3) and OG is that it keeps a virtual cost instead of an operational cost for every resource. For convenience, we use

Algorithm 2: The online greedy mechanism

$\theta_{arrived} \leftarrow \emptyset$ ▷ The set of arrived tasks
$\Lambda \leftarrow \emptyset$ ▷ The set of committed allocation decisions
$P \leftarrow \emptyset$ ▷ The set of payment decisions
for t *in* T **do**
  while *new tasks arrive within* t **do**
    When a new task i arrives ▷ Tasks arrive over time
    $\theta_{arrived} \leftarrow \theta_{arrived} \cup \{i\}$ ▷ Update the set of arrived tasks
    Solve the optimal utility allocation for task i (i.e., $\arg\max_{\lambda_i} \sum_{t \in T} (\hat{v}_i - o_i(\lambda_i))$, given Λ & $\tilde{\theta}_i$) ▷ Find the allocation for task i that maximise its utility
    $\Lambda \leftarrow \Lambda \cup \{\lambda_i\}$ ▷ Commit this allocation
    $p_i \leftarrow o_i(\lambda_i)$ ▷ Compute the payment for task i
    $P \leftarrow P \cup \{p_i\}$ ▷ Update the payment decisions
  end
  Allocate resources according to Λ
end

M to denote the set of every computational resource at each FN and the bandwidth resource on each link, and m is one type of them. To compute the virtual costs, we define the load factor $\kappa_{m,t}$ to be the proportion of occupied resource m at time step t. Then, the virtual cost accordingly is: $c_{m,t} = \mu^{\kappa_{m,t}} - 1, \forall t \in T, m \in M$, where $\mu = 2|M|F + 2$, and F is the upper limit of the ratio between the highest and the lowest task valuation per time step.

Then, the virtual cost of task i is $c_i = \sum_{r \in R, w \in W, t \in T} (a_{i,r} z^i_{w,t} c_{w,r,t}) + \sum_{l \in L, w \in W, (j,k) \in \mathbb{E}, t \in T} (2 c_{j,k,t} f^i_{l,w,j,k,t})$. Thus, the user can use resources cheaply when resources are abundant and is restrained when resources are in shortage. In the original paper by Shi *et al.* (2017), the virtual cost also prevents allocations that violate the resource constraints, which no longer works in our model, because, unlike them, we do not assume an upper bound of each task's resource requirements. Therefore, resource constraints are added to this mechanism. SWMOA2 is also DSIC and IR and the detail of it is shown in Algorithm 3 below.

Theorem 3.8. *The SWMOA2 mechanism is DSIC and IR.*

Proof. Following a similar argument, we can prove that SWMOA2 satisfies conditions one and three in Definition 3.3. Furthermore, the payment p_i is also independent of $\hat{v}_i$ because the virtual cost c_i does not depend on $\hat{v}_i$. In addition, $p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}^a_i \rangle})$ is also monotonic because increasing $\hat{T}^a_i$ not only results in resource being allocated to other users but also increases the virtual costs of resources, increasing $\hat{T}^s_i$ or decreasing $\hat{T}^d_i$ still reduces the available time steps to allocate, and increasing $\hat{t}_i$, $\{\hat{a}_{i,r}\}_{r \in R}$, $\{\hat{\Gamma}^i_l\}_{l \in L}$ increases the resource demands. Therefore, the SWMOA2 mechanism satisfies all the conditions in Definition 3.3 and is also DSIC and IR by Theorem 3.4.

Algorithm 3: The SWMOA2 mechanism

$\theta_{arrived} \leftarrow \emptyset$ ▷ The set of arrived tasks
$\Lambda \leftarrow \emptyset$ ▷ The set of committed allocation decisions
$\kappa_{m,t} \leftarrow 0, \forall m, t$ ▷ The load factors of resources
$c_{m,t} \leftarrow 0, \forall m, t$ ▷ The virtual costs of resources
for t *in* T **do**
  while *new tasks arrive within* t **do**
    When a new task i arrives ▷ Tasks arrive over time
    $\theta_{arrived} \leftarrow \theta_{arrived} \cup \{i\}$ ▷ Update the set of arrived tasks
    Solve the maximum virtual utility allocation for task i (i.e., $\arg\max_{\lambda_i}(\hat{v}_i(\lambda_i) - c_i(\lambda_i))$) ▷ Find the allocation that maximises task i's virtual utility
    $\Lambda \leftarrow \Lambda \cup \{\lambda_i\}$ ▷ Commit this allocation
    $p_i \leftarrow c_i(\lambda_i)$ ▷ Compute the payment for task i
    $\kappa_{m,t} \leftarrow \kappa_{m,t} + z^i_{w,t} a_{i,r}/A_{w,r}, \forall m \in P \times R, t \in T$ ▷ Update load factors of computational resources
    $\kappa_{m,t} \leftarrow \kappa_{m,t} + \sum_{l \in L, w \in W} f^i_{l,w,j,k,t}/b_{j,k}, \forall m \in \mathbb{E}, t \in T$ ▷ Update load factors of bandwidth resources
    $c_{m,t} = \mu^{\kappa_{m,t}} - 1, \forall t \in T, m \in M$ ▷ Update the virtual costs of resources
  end
  Allocate resources for next time step $(t+1)$ according to Λ
end

□

3.4 Flexible Online Greedy (FlexOG) Mechanism

Although OG is truthful, it is not efficient enough. This is because it commits the allocation schemes for arrived tasks, which may affects the allocation of future high-value tasks negatively. Our mechanism, FlexOG, builds upon OG by allocating newly arrived tasks greedily but keeps their specific allocation schemes flexible. In more detials, FlexOG uses a technique called pre-commitment, i.e., FlexOG only commits the usage time to a task. This gives it the DSIC property of OG but adds more flexibility. This also results in higher social welfare because there is more space for optimisation when high-value tasks arrive in the future. FlexOG is summarised in Algorithm 4. After receiving a report of task i, FlexOG finds the allocation that maximises the social welfare of all flexible tasks given the constraints of their committed usage time. Then, FlexOG computes the usage time $\tilde{t}_i$ for task i from its corresponding allocation scheme, and commits it to task i, which means that task i is guaranteed to get $\tilde{t}_i$ usage time before its reported deadline $\hat{T}^d_i$. Afterwards, FlexOG requires payment for task i as the marginal total operational cost, and task i is added to the set of flexible tasks. In addition, at the end of each time step, FlexOG allocates resources for the next time step

according to the latest allocation schemes. Finally, if a task will get all of its committed usage time in the next time step, it will be removed from the set of flexible tasks. In summary, the key idea of our mechanism is that we only commit the usage time $\tilde{t}_i$ to task i but keep its allocation scheme flexible.

Algorithm 4: The FlexOG mechanism

$\theta_{arrived} \leftarrow \emptyset$ ▷ The set of arrived tasks
$\theta_{flex} \leftarrow \emptyset$ ▷ The set of flexible tasks
$o \leftarrow 0$ ▷ The total operational costs
$\tilde{T} \leftarrow \emptyset$ ▷ The set of committed processing times
for t *in* T **do**
  while *new tasks arrive within* t **do**
    When a new task i arrives ▷ Tasks arrive over time
    $\theta_{arrived} \leftarrow \theta_{arrived} \cup \{i\}$ ▷ Update the set of arrived tasks
    $\theta_{flex} \leftarrow \theta_{flex} \cup \{i\}$ ▷ Update the set of flexible tasks
    Solve the maximum utility allocation for tasks in θ_{flex} (i.e., $\arg\max_{\lambda_j} \sum_{j \in \theta_{flex}} (\hat{v}_j(\lambda_j) - o_j(\lambda_j))$) ▷ Find the allocation for tasks in θ_{flex} that maximise their social welfare, given their committed usage time
    $\tilde{T} \leftarrow \tilde{T} \cup \{\tilde{t}_i(\lambda_i)\}$ ▷ Commit the processing time to i
    $p_i \leftarrow \sum_{j \in \theta_{arrived}} o_j(\lambda_j) - o$ ▷ Compute the payment for i
    $o \leftarrow \sum_{j \in \theta_{arrived}} o_j(\lambda_j)$ ▷ Update the total operational costs
  end
  for i *in* θ_{flex} **do**
    Allocate resources for the next time step $(t+1)$ according to λ_i
    $\tilde{t}_i \leftarrow \tilde{t}_i - \sum_{w \in W} z^i_{w,t+1}$ ▷ Update the remaining processing time of task i
    for $j = 1;\ j \leq \tilde{t}_i;\ j{+}{+}$ **do**
      $\hat{v}_i(j) \leftarrow \hat{v}_i(j + \sum_{w \in W} z^i_{w,t+1})$ ▷ Update the valuation function of task i
    end
    if $\tilde{t}_i = 0$ **then**
      $\theta_{flex} \leftarrow \theta_{flex} \setminus \{i\}$ ▷ Delete task i from the set of flexible tasks if it gets its required usage time
    end
    if $t = \hat{T}^d_i$ **then**
      $\theta_{flex} \leftarrow \theta_{flex} \setminus \{i\}$ ▷ Delete task i from the set of flexible tasks if it reaches its deadline
    end
  end
end

Theorem 3.9. *The FlexOG mechanism is DSIC, IR and WBB.*

Proof. Following a similar argument, the FlexOG mechanism satisfies condition three in Definition 3.3. The payment $p_i(\lambda_i, \hat{\theta}^{\langle \hat{T}^a_i \rangle})$, which equals the marginal operational cost under this mechanism, is monotonic and independent of $\hat{v}_i$ for the following reasons.

Since the value of flexible tasks $\sum_{t\in T, j\in\theta_{flex}} \hat{v}_j$ is independent of the specific allocation of resources, by maximising $\sum_{t\in T, j\in\theta_{flex}} (\hat{v}_j - o_i)$, the mechanism actually minimises the total operational cost for every possible allocation $\tilde{t}_i(\lambda_i)$. So the payment is independent of $\hat{v}_i$ because the total operational cost is independent of $\hat{v}_i$. Moreover, increasing $\hat{T}_i^a, \hat{T}_i^s$, $\hat{t}_i$, $\{\hat{a}_{i,r}\}_{r\in R}$, $\{\hat{\Gamma}_l^i\}_{l\in L}$ or decreasing $\hat{T}_i^d$ can only increase the total operational cost $\sum_{t\in T, j\in\theta_{flex}} o_j$ following a similar argument in the proof of Theorem 2. Hence, this mechanism satisfies condition 1 in Definition 3.3 too. The mechanism also satisfies condition two because it maximises $(\sum_{t\in T, j\in\theta_{flex}} \hat{v}_j - o^i_{total})$, which is equivalent to maximising $\hat{v}_i - p_i = \hat{v}_i - (o^i_{total} - o^{i\prime}_{total})$ (o^i_{total} and $o^{i\prime}_{total}$ are the total operational costs of flexible tasks after and before i arrives). From the above, the FlexOG mechanism is DSIC and IR by Theorem 3.4. Finally, because the payment of task i equals the marginal operational cost ($p_i = o^i_{total} - o^{i\prime}_{total}$), the total payment equals the total operational cost of the fog ($\sum_{i\in I} p_i = o$). Thus, FlexOG satisfies WBB. □

3.4.1 No Limited Misreport of Deadlines

As we discussed before, users can report later deadlines for some types of tasks, when withholding the results until the reported deadline is not feasible. In that case, OG and SWMOA2 mechanisms are still DSIC, while FlexOG is not DSIC any more. We first give an example showing why FlexOG is not DSIC. For example, suppose user i reports a later deadline $\hat{T}_i^d$ ($\hat{T}_i^d > T_i^d$) and reports other information of its task truthfully. It may gets a lower payment because the payment function p_i is monotonic according to Definition 3.3. Then, it is possible that its committed time steps get rescheduled in its time window ($[T_i^s, T_i^d]$) because of tasks that arrive in the future. In that case, user i gets more utility by misreporting a later deadline, because it gets the same value with a lower payment. However, the following theorem shows that OG and SWMOA2 are still DSIC and IR.

Theorem 3.10. *The OG and SWMOA2 mechanisms are DSIC and IR when users are able to report their deadlines arbitrarily.*

Proof. First, a user cannot increase its utility by reporting an earlier deadline or misreporting other information about its task (under the limited misreport assumptions in Section 3.1.2) according to Theorem 3.7 and 3.8. Then, we analyse the case where user i reports a later deadline. Suppose the utility of user i is u_i when it reports truthfully, and the utility of user i is u_i' when it reports a later deadline. If the usage time is still allocated in its time window $[T_i^s, T_i^d]$, it gets the same utility ($u_i' = u_i$). If some time steps ($\tilde{t}_1$) are allocated in its time window, while some time steps ($\tilde{t}_2$) are not, then, $u_i' = v_i(\tilde{t}_1) - p_i(\tilde{t}_1) - p_i'(\tilde{t}_2)$, and $u_i = v_i(\tilde{t}_1 + \tilde{t}_2) - p_i(\tilde{t}_1) - p_i(\tilde{t}_2)$. Now, $v_i(\tilde{t}_1 + \tilde{t}_2) - p_i(\tilde{t}_1) - p_i(\tilde{t}_2) \geq v_i(\tilde{t}_1) - p_i(\tilde{t}_1)$ because OG and SWMOA2 always choose

the allocation that maximises user i's utility. Thus, the utility of user i when it reports a later deadline is also no more than its utility when it reports truthfully ($u'_i \leq u_i$). Therefore, the OG and SWMOA2 mechanisms are still DSIC when users can report later deadlines. Users will get a utility of at least zero because the payment for no allocation is zero and the machanisms maximise the users' utilities. Therefore, these mechanisms also satisfy IR. □

3.5 Summary

In this chapter, we described the details of the RAFC model studied in this thesis and formalised it as a constraint optimisation problem. Then we introduced a class of resource allocation mechanisms called price-based mechanisms, which is DSIC and IR, and our proposed mechanism FlexOG belongs to this class of mechanisms. After that, we gave the algorithms of the benchmarks we used to evaluate the performance of FlexOG and presented some properties of these benchmarks. Finally, we described FlexOG in detail and proved that it is DSIC and IR.

Chapter 4

Simulations and Results

In this section, we evaluate FlexOG by extensive simulations. Firstly, we describe how the synthetic data is generated. We use synthetic data because there currectly exists no comprehensive data set of real-world fog computing tasks. Following that, we evaluate FlexOG's performance in terms of social welfare. We show that FlexOG is more efficient than all truthful benchmark mechanisms and achieves a social welfare close to the theoretical upper bound.

4.1 Experimental Setup

Although some work used the Google cluster data (Shi *et al.*, 2015, 2017; Zhang *et al.*, 2015b), which are traces (including numbers of CPU cores, CPU time and memory size) of workloads running on Google compute cells. However, this real dataset is more related to cloud computing rather than fog computing, so it does not include the information that is critical to fog computing such as the locations of IoT devices, the time constraints and the bandwidth demand of each task, and the valuation function of each task.

In order to simulate a small fog computing network, we choose the following parameters. The time span of our discrete time period is $|T| = 12$. The fog provider has 6 FNs ($|P|$=6) and 6 locations ($|L|$=6). Three classic topologies of the network are used in our simulations: an almost fully connected topology, a ring topology and a line topology (see Figures 4.1,4.2 and 4.3). Additionally, there are $|R| = 3$ types of computational resources (CPU, RAM, and storage) at each FN. Another reason why we choose this small setting is that we can run more trials in a reasonable time for all mechanisms.

The number of tasks in this time period is $|I| = 40$. The arrival time T_i^a follows a continuous uniform distribution $U(0, 10)$, so that no tasks arrive at exactly the same time, which is an assumption we made earlier in our RAFC model. Moreover, the number of endpoints for each task E_i is generated uniformly from $\{1, 2, \ldots, 6\}$, and the

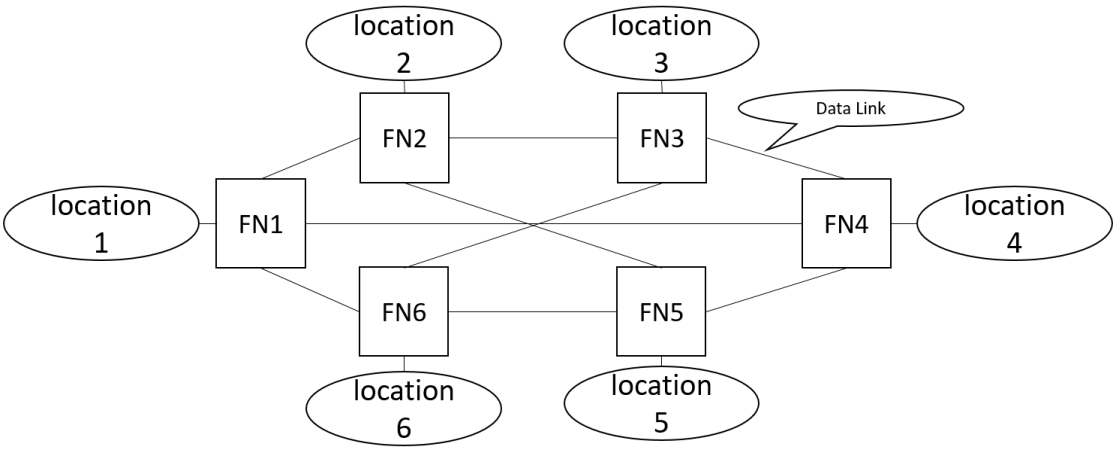

FIGURE 4.1: The (almost fully connected) topology of the fog computing system.

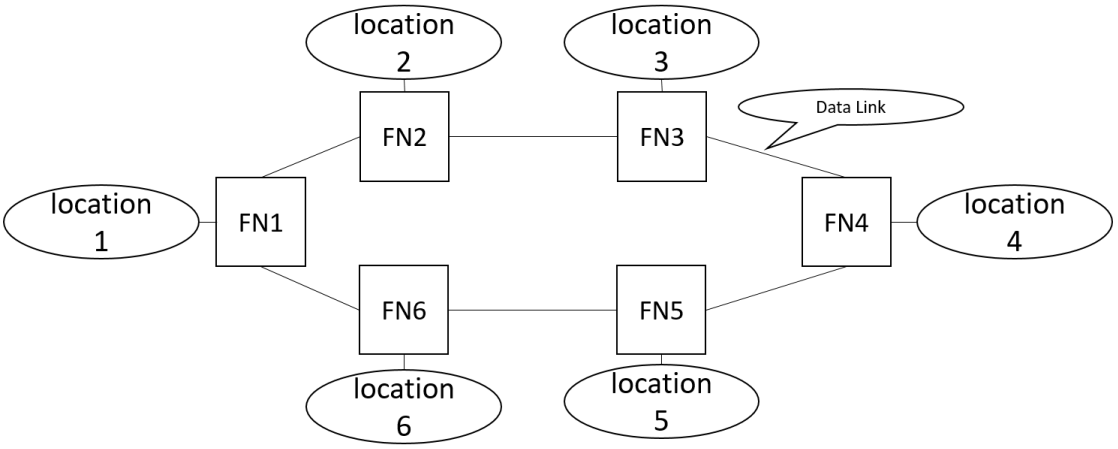

FIGURE 4.2: The (ring) topology of the fog computing system.

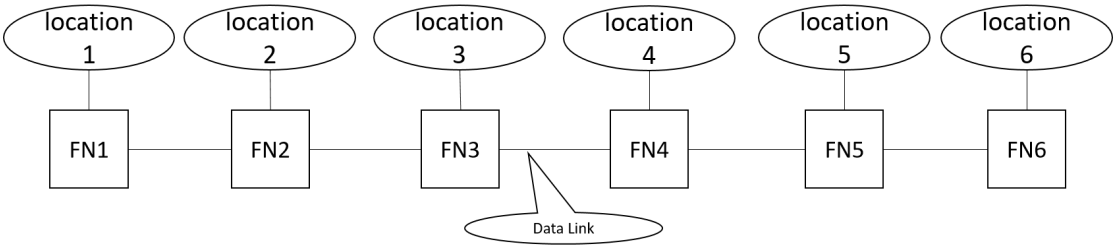

FIGURE 4.3: The (line) topology of the fog computing system.

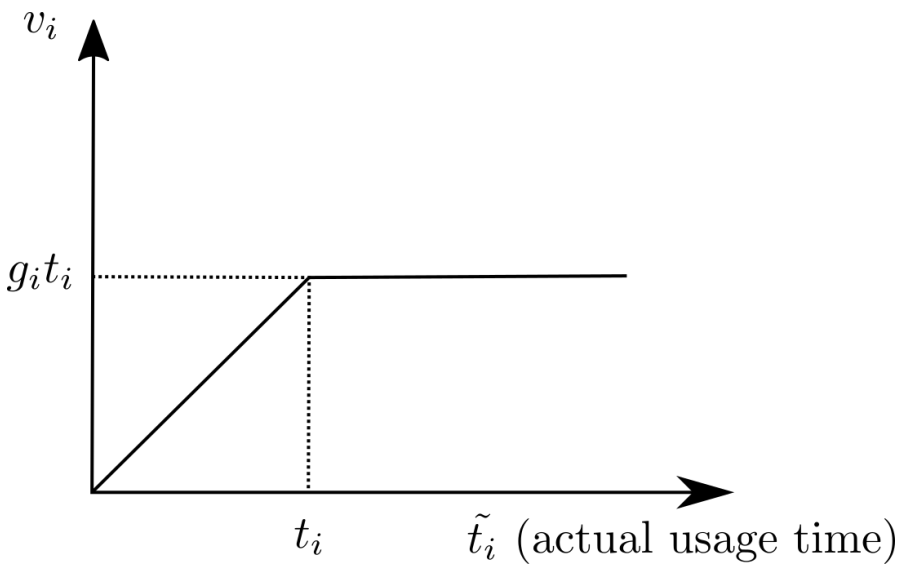

FIGURE 4.4: The valuation function of task i.

location of each endpoint $u^i_{e,l}$ is chosen uniformly at random from all locations L with replacement.

Furthermore, we choose a secial valuation function v_i in our simulations for simplicity, which is a non-decreasing linear function of the actual usage time $\tilde{t_i}$:

$$v_i(\tilde{t_i}) = \begin{cases} g_i \times \tilde{t_i} & \text{if } \tilde{t_i} \leq t_i \\ g_i \times t_i & \text{if } \tilde{t_i} > t_i \end{cases}$$

where the valuation coefficient g_i represents task i's obtained value per usage time. An example of such a valuation function is shown in Figure 4.4.

Since the value densities (i.e., the average valuation of each time step) of fog tasks vary considerably in real life. To make the resource allocation more realistic, there are two types of tasks in this synthetic data: low-value tasks and high-value tasks. The proportion of high-value tasks is denoted as $q \in [0, 1]$. For task i of either type: $a_{i,r}$ $\forall r \in R$ and Γ^i_l $\forall l \in L$ are all generated from a Gaussian distribution $\mathcal{N}(1, 1)$ with negative results discarded. The usage duration t_i is a positive integer uniformly chosen from $\{1, 2, 3, 4\}$, and the start time T^s_i is an integer uniformly chosen within 2 time steps after the arrival time: $\{\lceil T^a_i \rceil, \lceil T^a_i \rceil + 1, \lceil T^a_i \rceil + 2\}$. Furthermore, the deadline T^f_i is an integer uniformly chosen between a and b time steps after the earliest finish time (not exceeding the last time step): $\{T^s_i + t_i - 1 + a, T^s_i + t_i + a, \ldots, min(T^s_i + t_i - 1 + b, |T|)\}$. So (a, b) defines the deadline slackness of the task, which is an important parameter because it reflects the task's flexibility. For a low-value task i, g_i is uniformly chosen from a continuous interval: $[8, 30]$. However, for a high-value task i, g_i is uniformly chosen from a continuous interval: $[180, 200]$. Thus, the upper bound of the ratio between the highest and lowest valuation coefficient $F = \frac{200}{8} = 25$ in this case.

Finally, the overall resource capacity of each computational resource r: $\sum_{w \in W} A_{w,r}$ is set to be a k fraction of the corresponding total resource demand: $\sum_{i \in I} a_{i,r}$, and the overall bandwidth capacity: $\sum_{(j,k) \in \mathbb{E}} b_{j,k}$ is set to be a $2k$ fraction of the total bandwidth demands: $\sum_{i \in I, l \in L} \Gamma_l^i$ because data traffic usually flows though multiple data links. Then, each FN receives the same fraction of resource r: $\frac{\sum_{w \in W} A_{w,r}}{|W|}$, and each data link receives the same fraction of the available total bandwidth: $\frac{\sum_{(j,k) \in \mathbb{E}} b_{j,k}}{|\mathbb{E}|}$. Finally, the unit operational costs at different FNs and links: $o_{w,r}\, w \in W, r \in R$ and $o_{j,k}, (j,k) \in \mathbb{E}$ are all generated uniformly from [0.03, 0.1].

4.2 Results and Analysis

We have tested the robustness of our mechanism by running simulations with different parameters, such as topologies of the network, resource scarcity in FNs and data links and deadline slackness of tasks. Across all of these settings, trends are similar. In particular, the FlexOG's performance in social welfare is typically around 90% of the theoretical upper bound, and between 5-10% better than OG's. In the following, we will show the results of these simulations and analyse them.

4.2.1 Social Welfare for Different Levels of Resource Scarcity

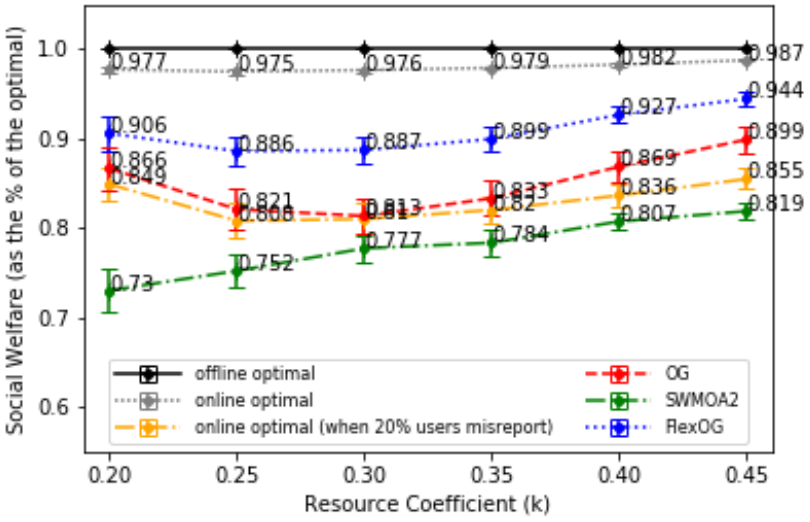

FIGURE 4.5: The social welfare achieved by four mechanisms with parameters $((a, b) = (5, 10), F = 25, q = 0.1)$ and almost fully connected network.

First, we compare the total social welfare achieved by FlexOG with other benchmarks under different resource coefficients k indicating the scarcity of the resources in Figure

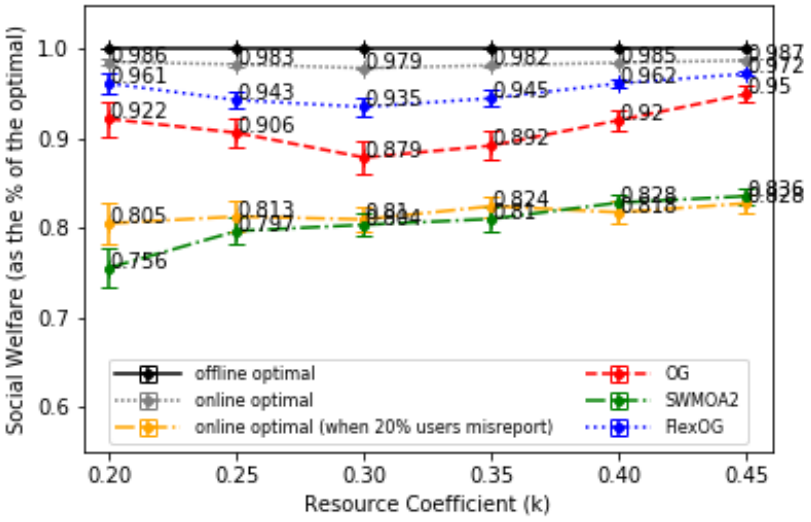

FIGURE 4.6: The social welfare achieved by four mechanisms with parameters $((a, b) = (5, 10), F = 25, q = 0.1)$ and ring network.

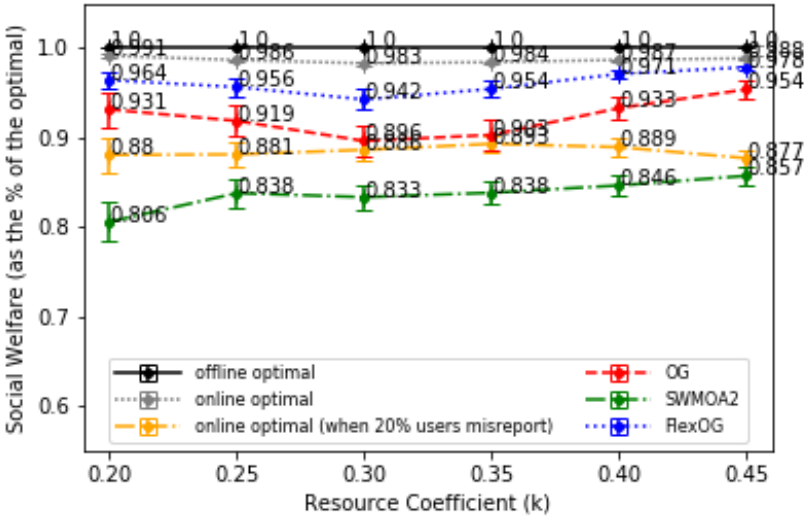

FIGURE 4.7: The social welfare achieved by four mechanisms with parameters $((a, b) = (5, 10), F = 25, q = 0.1)$ and line network.

4.5, Figure 4.6 and Figure 4.7.[1] Note that we normalise the results to the performance of offline optimal so that it is easier to compare the performance of different mechanisms. Since the trend of all three figures is similar, we analyse Figure 4.5 as a representative in the following.

[1]All figures are with 95% confidence intervals based on 200 trials, and the relative tolerance of the CPLEX optimizer is set to 1% for offline optimal, and 5 % for other mechanisms. (A 1% tolerance means that the optimiser stops when a solution is within 1% of optimality) The reason we set the relative tolerance for offline optimal lower is to make the upper bound of social welfare more accurate.

Figure 4.5 shows that FlexOG consistently achieves better social welfare than other truthful benchmark mechanisms (i.e., OG and SWMOA2). In particular, SWMOA2 always has the worst performance mainly because its virtual prices are exponential to the load factors of the resource, and this hinders tasks from getting allocated even when there is enough resource for them. This phenomenon gets even more significant when the resource is more scarce, i.e., when k is lowerc. For example, the average social welfare achieved by SWMOA2 is only 78.15% of that achieved by FlexOG when the resource coefficient $k = 0.2$, while for other higher k this number is around 87%. This is because, with a scarcer resource, the load factors of resources increase faster, and so do the virtual prices.

Furthermore, the performance of FlexOG is about 5%-10% better than that of OG in terms of social welfare. The reason for this is that under FlexOG when and how committed time steps are allocated to tasks is flexible. Therefore, FlexOG can reschedule unfinished tasks to allocate more time steps for the newly arrived high-value task, while OG cannot do this. The figure also shows that the performance difference between FlexOG and OG shrinks when the resource coefficient k is either very small or very big. Intuitively, this is because when there are few resources or there are abundant resources the performance of OG will be closer to the optimal, and there is less space for FlexOG to improve social welfare by rescheduling tasks. This indicates that the superiority of FlexOG is more significant when resources are neither too scarce nor too abundant, which is in correspondence with reality in most cases.

In addition, our mechanism also performs close to offline optimal, achieving around 90% in almost fully connected network and around 50% in other topologies, which indicates that our mechanism is efficient even though it is online. Interestingly, the performance of online optimal is very close to that of offline optimal. The main reason is that all tasks are preemptible, so myopic decisions will not have a big effect in the future. If a high-value task arrives, online optimal can always pause some low-value tasks to process the newly arrived high-value one. However, online optimal is not truthful and vulnerable to manipulations. So, although online optimal performs about 10% better than FlexOG, its performance drops below that of FlexOG when just 20% of users misreport. Here, users who misreport only misreport their valuation coefficient much higher (as 1 million), and report other information truthfully. So the performance of online optimal is mainly used as an indicator of the upper bound of online resource allocation here. In addition, we have also tested whether users have an incentive to misreport by comparing utilities of truthful and non-truthful users (see Section 4.2.4), and the result shows on average non-truthful users get a higher utility. This means that, in a strategic setting where users can misreport, FlexOG can actually achieve significantly more social welfare than online optimal.

Finally, Figure 4.8 shows the number of tasks that get allocated (i.e., get at least one

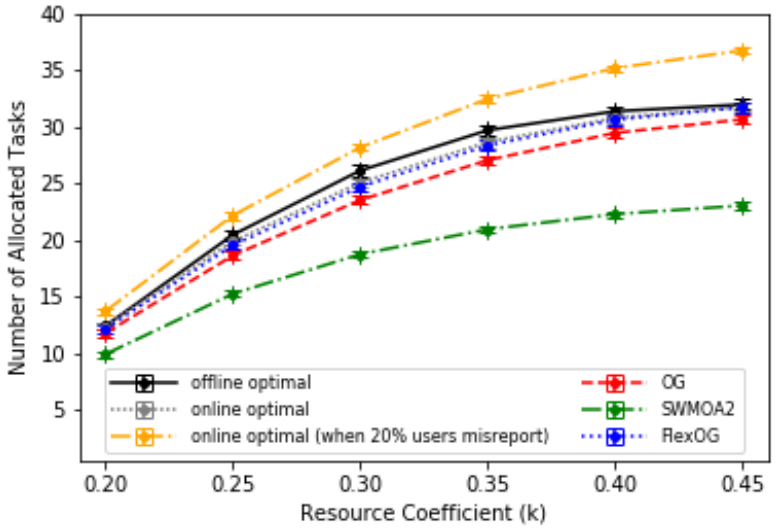

FIGURE 4.8: The number of tasks got allocated by five mechanisms with parameters $((a, b) = (5, 10), F = 25, q = 0.1)$ and the almost fully connected network.

time step usage time) under different mechanisms with the almost fully connected network. The result is similar for other network topologies, so these results are omitted in this thesis. We can see from the figure that resources are truly scarce when $k = 0.2$ because only around 30% of tasks get allocated under offline optimal. While the resources are relatively abundant when $k = 0.45$, where around 75% of tasks get allocated under offline optimal. In addition, FlexOG only allocates about one or two tasks more than OG on average, which means that FlexOG achieves more social welfare mainly by giving more usage time to high-value tasks rather than getting more tasks allocated. Interestingly, more tasks get allocated under online optimal when 20% of users misreport than offline optimal. Intuitively, this is because some low-value tasks which would not get allocated under other mechanisms can get some usage time by misreporting high valuations. Besides, since the virtual cost of SWMOA2 is often much higher than the operational cost, thus preventing tasks from getting allocated even when there are enough resources to process them, it has the lowest number of allocated tasks.

4.2.2 Social Welfare for Different Levels of Deadline Slackness

Next, we compare the performance in terms of social welfare under different levels of deadline slackness in various network topologies (Figures 4.9, 4.10 and 4.11). We do not include the online optimal mechanism here because it is not truthful and we have already shown that its efficiency is lower than that of the OG and FlexOG mechanisms in a strategic setting. A task with a bigger deadline slackness has more time steps between its earliest possible finish time and its deadlines and is more flexible to allocate. As can be seen from the figures, the gap between FlexOG and OG increases as the

deadline slackness of tasks increases. This is because when tasks are more flexible, FlexOG is more likely to reschedule low-value tasks to allocate more high-value tasks, while OG cannot benefit from this since its resource allocation is fixed once it has been made. In addition, SWMOA2's performance is quite stable in terms of different levels of deadline slackness. This is because, instead of the flexibility of tasks, the virtual prices of resources play a major role in affecting the performance of SWMOA2.

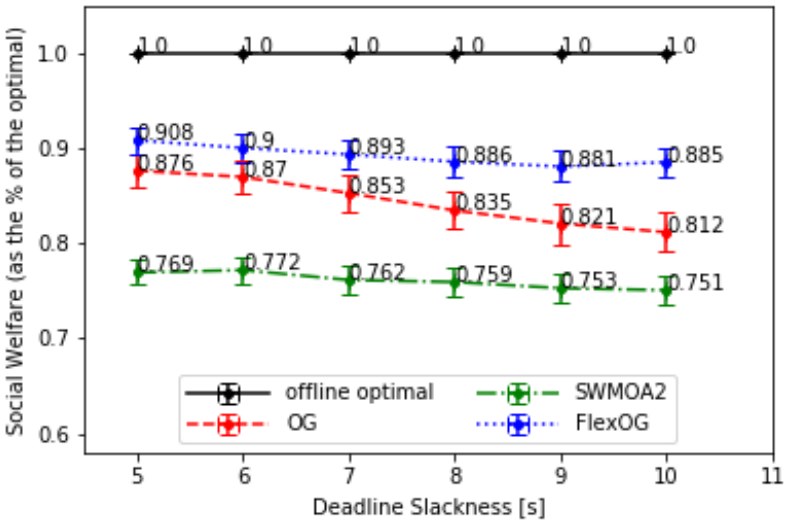

FIGURE 4.9: The social welfare achieved by four mechanisms with parameters $((a, b) = \{(0, 5), (1, 6), (2, 7)(3, 8), (4, 9), (5, 10)\}, F = 25, q = 0.1, k = 0.3)$ and almost fully connected network.

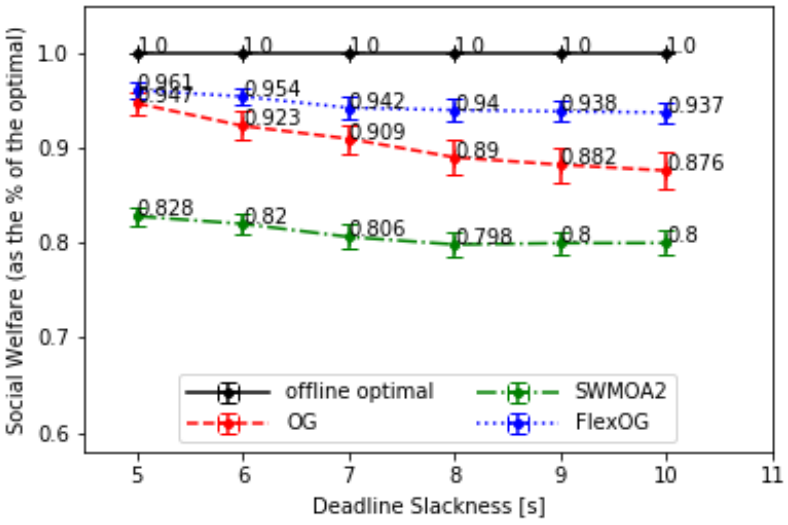

FIGURE 4.10: The social welfare achieved by four mechanisms with parameters $((a, b) = \{(0, 5), (1, 6), (2, 7)(3, 8), (4, 9), (5, 10)\}, F = 25, q = 0.1, k = 0.3)$ and ring network.

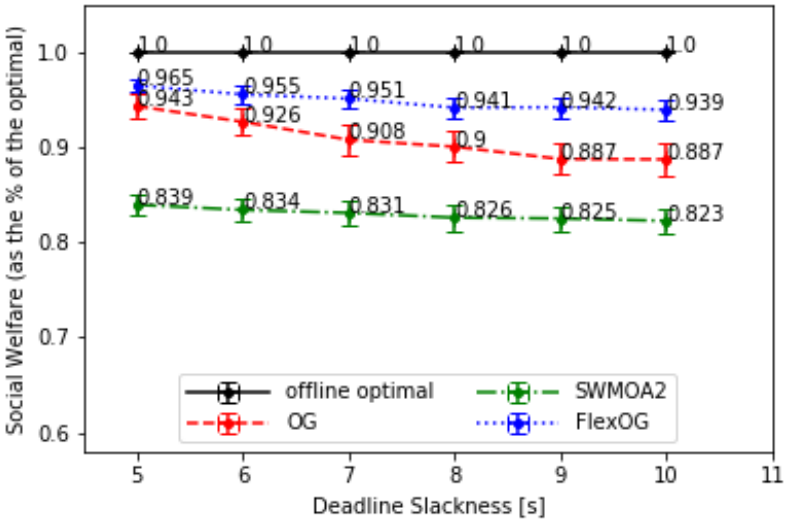

FIGURE 4.11: The social welfare achieved by four mechanisms with parameters $((a, b) = \{(0, 5), (1, 6), (2, 7)(3, 8), (4, 9), (5, 10)\}, F = 25, q = 0.1, k = 0.3)$ and line network.

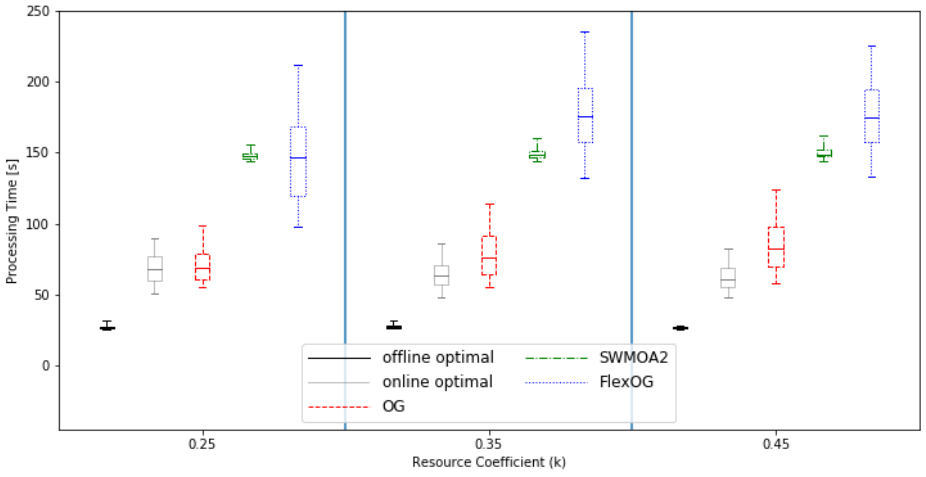

FIGURE 4.12: The processing time of five mechanisms with parameters $((a, b) = (5, 10), F = 25, q = 0.1)$ and almost fully connected network.

4.2.3 Processing Time of Different Mechanisms

In this section, we compare the processing time of FlexOG with the benchmark mechanisms. The processing time is also important because many users cannot wait for a long time for an allocation decision, and the fog provider does not want to use too much computing resource on making allocation decisions. Thus, in this thesis, the shorter the processing time, the better (Challenge 8). The simulation was conducted on the Iridis 4

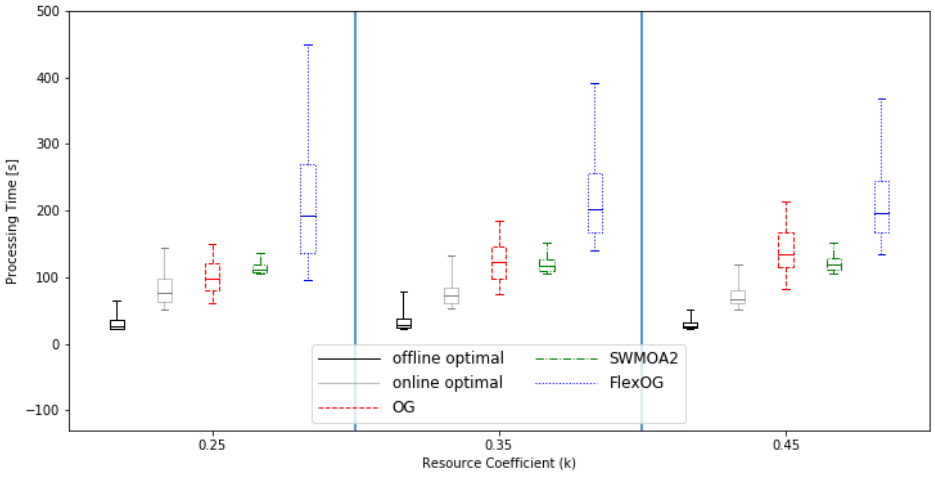

FIGURE 4.13: The processing time of four mechanisms with parameters $((a, b) = (5, 10), F = 25, q = 0.1)$ and ring network.

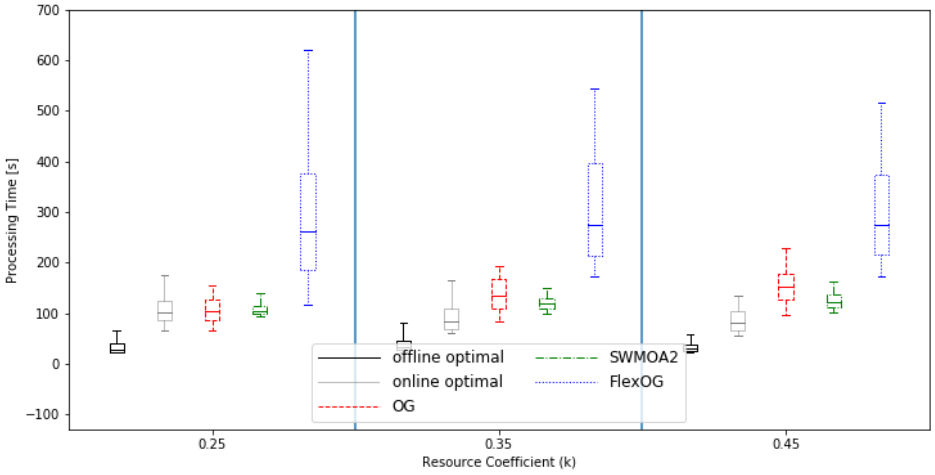

FIGURE 4.14: The processing time of four mechanisms with parameters $((a, b) = (5, 10), F = 25, q = 0.1)$ and line network.

Compute Cluster [2], using four cores of an Intel Xeon E5-2670 ('Sandybridge') processor with 16GB RAM.

The results are shown in Figure 4.12, Figure 4.13 and Figure 4.14. We plot the processing time of all mechanisms under resource coefficients $k = 0.25$, $k = 0.35$ and $k = 0.45$. The processing time under other resource coefficients has a similar trend and is not plotted to make the figures easier to read. Note that the boxes show the lower to upper 25% values of the data with whiskers showing 5 to 95 percentile of the data, and the outliers are discarded.

[2] https://hpc.soton.ac.uk/redmine/projects/iridis-4-support/wiki

The figure shows that on average, offline optimal takes the least processing time, online optimal and OG take more processing time, and SWMOA2 and FlexOG uses the most processing time. This is mainly because offline optimal only needs to solve the optimisation problem once, while all other mechanisms need to solve the optimisation problem multiple times. Online optimal needs to solve the optimisation problem at the start of each time step except for the first time step, so it needs to solve it $|T| - 1 = 11$ times. The remaining mechanisms all need to solve the optimisation problem when a new task arrives, so they need to solve it $|I| = 40$ times. However, the processing time of offline optimal is much more than $\frac{1}{11}$ of that of online optimal. This is because offline optimal needs to make a resource allocation decision for all tasks at once, while online optimal only needs to make a decision for tasks that arrived in the previous time step every time. So offline optimal's optimisation problems have more decision variables and take more time to solve on average. Another reason is that the relative tolerance of offline optimal is set to be 0.01 while this parameter is set 0.05 for other mechanisms, which also causes offline optimal to takes more time to find a solution to its optimisation problem.

Similarly, although OG needs to solve the optimisation problem four times as many time as online optimal, it has a similar processing time on average. This is because OG only needs to make a resource allocation decision for just one task at a time. So, its optimisation problem is solved faster than online optimal's on average.

Finally, FlexOG takes the most time because it not only needs to solve the optimisation problem $|I| = 40$ times, but its optimisation problems also have more decision variables. This is because it needs to make allocation decisions for all flexible tasks each time a new task arrives. The reason why SWMOA2 takes the second-highest time on average is that SWMOA2 needs to compute the load factor and virtual price of each resource after every allocation decision, while other mechanisms just use the fixed operational cost of each resource. Interestingly, the processing time of FlexOG also has a wider dispersion than other mechanisms. This is because with more optimisation problems to solve and with more decision variables, FlexOG has a higher probability of coming across optimisation problems that are hard to solve. For each task, FlexOG takes around four seconds to make the allocation decision on average. For tasks whose arrival time is much earlier than their start time, it is feasible to use FlexOG because there is sufficient time for FlexOG to make allocation decisions. However, if the arrival time and the start time of a task are the same, or tasks arrive frequently, then the processing time of FlexOG will become an issue. We will discuss how we plan to solve this issue in the future work section (Section 5.4).

4.2.4 Utility of Truthful and Non-truthful Users

In this section, we discuss whether fog users are incentivised to report their tasks truthfully by comparing average utilities of truthful and non-truthful users. The result of what

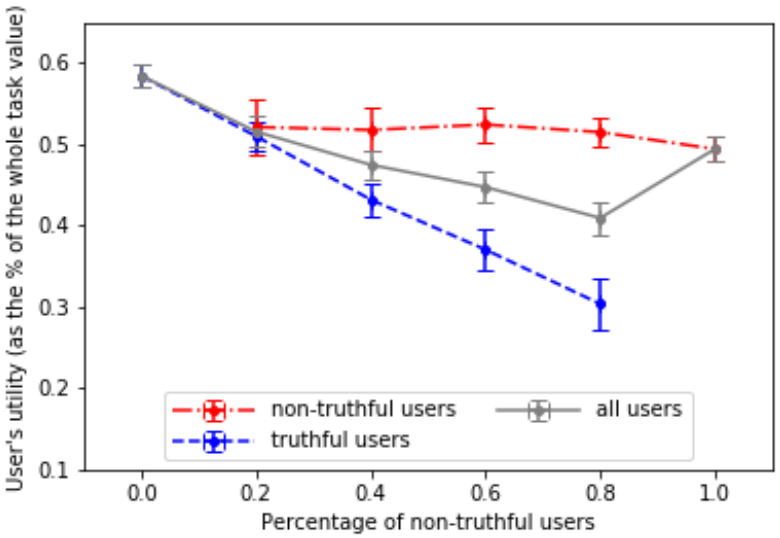

FIGURE 4.15: Utility comparison between truthful and non-truthful users ($(a, b) = (5, 10), F = 25, q = 0.1$) in all connected network.

percentage of the total value of tasks (i.e., the sum of value for fog users if their tasks are all fully accomplished) are achieved by all users, truthful users and non-truthful users respectively under the online optimal mechanism in the almost fully connected topology is shown in Figure 4.15. Recall that non-truthful users only misreport their valuation coefficient as one million. This is because if they can achieve a higher utility compare to truthful users by misreporting their valuation coefficient, they will have an even bigger advantage when they misreport other information as well. The figure shows that non-truthful users have similar utility as truthful users when 20% of fog users are non-truthful. However, the utilities of truthful users decrease rapidly as more users become non-truthful, and non-truthful users have about a 67% higher utility than truthful users on average when 80% of users are non-truthful. This is because the reported valuation coefficients of non-truthful users are much higher than that of truthful users. Online optimal will always prioritise non-truthful users when it makes allocation decisions. This means that under online optimal, fog users have no incentives to be truthful. The social welfare achieved by online optimal when all users report their types truthfully cannot be achieved in a strategic setting.

Figure 4.15 also indicates that non-truthful users have a negative impact on the overall utility (social welfare). This is due to the fact that online optimal makes allocation decisions based on false valuation coefficients, so it may priorities a low-value task over a high-value task. However, when all users are non-truthful, online optimal achieves a slightly higher social welfare than that when 80% users are non-truthful. The reason for this is that when all users are non-truthful, their valuation coefficients are all one million, the case where a low-value task is prioritied over a high-value task because of its misreported valuation coefficient disappears. The reason why social welfare is lower

when all users are non-truthful than that when all users are truthful is mainly that low-value tasks are still prioritised because they need fewer resources on average.

4.3 Summary

In this chapter, we described the synthetic data used in our simulations and analysed the results from a number of empirical simulations. These results highlighted the advantages of the FlexOG mechanism, which consistently outperformed all truthful benchmarks and achieved a social welfare near the theoretical upper bound. Furthermore, we also showed that fog users are incentivised to misreport their types under the non-truthful benchmark online optimal, which degrades social welfare significantly. Specifically, online optimal is outperformed by FlexOG when only 20% of fog users misreport. Besides, the results showed that the advantage of FlexOG is bigger when resources are neither scarce nor sufficient and when tasks have a higher deadline slackness, which is common in practice. Therefore, our proposed mechanism FlexOG addresses the high social welfare resource allocation challenge (Challenge 4). Nevertheless, simulations showed that it usually takes around 2-6 seconds for FlexOG to make an allocation decision for a newly arrived task. Although this is fine for tasks with a long time between their arrival times and start times, it is infeasible for tasks whose arrival times equal or are close to their start times or tasks that arrive frequently. So the timely resource allocation challenge (Challenge 8) has only been partly addressed. We will discuss this in detail in the future work section (Section 5.4).

Chapter 5

Conclusions and Future Work

fhe final chapter concludes this thesis by reviewing its contributions to the field of RAFC and by outlining possible directions for future work. To this end, in Section 5.1, we cast our mind back at the motivations of this thesis and provide a general overview of the techniques we have proposed to deal with it. Then, in Section 5.2, we describe our research contributions in detail and relate them back to our research challenges from Section 1.4, and in Section 5.3, we summarise the evaluation results of our proposed mechanism by comparing it with benchmarks. Finally, in Section 5.4, we propose several possible ways in which our mechanism can be extended in the future.

5.1 Research Summary

Nowadays, the IoT is developing very fast and is predicted to revolutionise the way we work and live by making all kinds of physical devices into IoT devices and connect them to the Internet. In this context, fog computing is proposed to complement cloud computing in providing computing resource to IoT devices. Fog computing has many important benefits, such as low latency and reducing data traffic, which are critical to the popularisation of the IoT.

Hitherto there is no truthful resource allocation mechanism that achieves high social welfare in our RAFC model. As we have argued in this thesis, it is necessary to treat fog users as intelligent agents who are rational and have their own goals (i.e., utility maximisation). Therefore, it is vital to make sure the resource allocation mechanism is truthful and thus can be used in strategic settings.

Reviewing the existing resource allocation mechanisms, we found that they cannot apply to our RAFC model or only address part of our research challenges. A lot of existing approaches only deal with homogeneous resource allocation, whereas there are several heterogeneous resources in RAFC. Some other approaches can only be used in offline

settings, while in RAFC tasks arrive over time and allocation decisions must be made upon task arrival without future information. Finally, most existing mechanisms are not truthful and therefore infeasible to work in strategic settings.

In the following section, we provide a summary of our mechanism, including the contributions we have made to the state of the art and how it addresses the research challenges listed in Section 1.4.

5.2 Research Contributions

In this thesis, we aimed to design a truthful resource allocation mechanism for our RAFC problem that is efficient in terms of social welfare. We achieved this by extending a class of mechanisms called price-based mechanisms and a technique called pre-commitment. In the following, we outline each of the contributions of this thesis along with the research challenges they address (Section 5.2.1-5.2.2).

5.2.1 Formulation of the RAFC Problem

In this thesis, we described the RAFC problem we study in detail and formulated it as a constraint optimisation problem (Optimisation Problem 3.1a) considering all related research challenges (Challenges 1, 2, 3, 4 and 7). Firstly, the optimisation problem including the bandwidth and traffic routing constraints (Challenge 1). Secondly, it allows dynamic generation of VMs in any FNs (Challenge 2) and uses time-oriented valuation functions (Challenge 3). Finally, its objective function includes the operational cost of the fog (Challenge 7) and is to maximise the social welfare (Challenge 4).

5.2.2 Design of FlexOG

In order to allocate resources efficiently in strategic settings (Challenges 4, 6 and 5), we designed a price-based online mechanism called FlexOG. This mechanism belongs to a class of mechanisms called price-based mechanisms, which ensures that it is truthful and works in strategic settings. Furthermore, FlexOG uses a technique called pre-commitment, which means that FlexOG only commits the usage time to a task when it arrives but leaves how the usage time is scheduled flexible. This technique gives FlexOG an advantage in achieving higher social welfare than benchmark mechanisms.

5.3 Empirical Evaluation of FlexOG

To make sure that FlexOG can truly achieve higher social welfare than other state-of-the-art mechanisms (Challenge 4), we carried out extensive simulations to evaluate the performance of FlexOG. More specifically, we evaluated FlexOG using synthetic data of different network topologies, resource scarcities or deadline slackness of tasks. Across all of these settings, FlexOG achieves social welfare higher than the two truthful benchmarks (around 5-10% better than OG, and 10-20% better than SWMOA2) and close to the theoretical upper bound (around 90%). In particular, we showed through simulations that users are incentivised to misreport under the non-truthful benchmark online optimal, and the social welfare achieved by online optimal falls below FlexOG when 20% of users misreport their valuation. In short, these simulations indicated that FlexOG outperforms all benchmarks in strategic settings.

5.4 Future Work

The work in this thesis can be extended in various ways. First, we plan to complement the empirical evaluation of resource allocation mechanisms with real data evaluation or using fog computing simulation tools, and we detail these in Section 5.4.1. Second, we aim to design mechanisms which are more scalable or more efficient in social welfare. We discuss how this might be done in Section 5.4.2. Finally, we show a Gantt chart of the future work plan.

5.4.1 Future Empirical Evaluation

Although our evaluation is performed over a variety of configurations using synthetic data, it is still of value to evaluate FlexOG using real data. Although, there is currently a lack of comprehensive real-world data from fog computing systems, one way to circumvent this problem is to combine real-world cloud computing data with synthetic data. For example, we can set the amount of resources required by a VM according to the resource demand of each task in Google cluster data and generate other data such as the number of IoT devices, bandwidth demands and required usage time of each tasks synthetically. Some work takes this approach to evaluate their mechanisms (Zhang *et al.*, 2015b; Shi *et al.*, 2017).

Another way to make the evaluation more convincing is to use widely accepted simulation tools for fog computing such as *FogNetSim++* (Qayyum *et al.*, 2018) and *iFogSim* (Gupta *et al.*, 2017a). They enable simulations of fog computing environments for evaluating resource allocation mechanisms. There is much work that used such tools to evaluate their approaches, especially *iFogSim* (Rahbari and Nickray, 2017; Taneja and

Davy, 2017). Thus, to extend our empirical evaluation, we can either adapt available cloud computing data to our RAFC model or use a widely accepted simulation tool for fog computing in future work.

5.4.2 Future Resource Allocation Mechanisms

First, we plan to improve the scalability of our mechanism, which addresses Challenge 8. Our simulations showed that the mechanism we proposed in this report-FlexOG needs $2-6$ seconds on average to make an allocation decision for a newly arrived task. Furthermore, our simulations are only on a small setting with 6 FNs, 6 locations, 12 time steps and 40 tasks. In reality, fog computing systems can have thousands of FNs and tasks. Therefore, the scalability of FlexOG needs to be improved to make it more practical. One approach to improve the scalability is to simplify the associated data routing problem. For example, we could restrict the number of hops from IoT devices of a task to the VM of that task, which is also in accordance with the low latency requirements of IoT applications. Another option is to design an efficient algorithm to solve the MILP problem of social welfare maximisation instead of solving it using CPLEX. Finally, we are also considering to use sub-optimal heuristics, for example by greedily allocating tasks based on their value densities. Since finding the optimal allocation for newly-arrived tasks is critical to the truthfulness of FlexOG, we would need to ensure that the DSIC and IR properties still hold. Thus, using heuristics instead of solving the optimisation problem is challenging.

Secondly, to further address Challenge 4, i.e., to design mechanisms that are more efficient in social welfare. We are considering using online auctions or reinforcement learning techniques instead of price-based mechanisms to allocate resources. Under online auction mechanisms, resource allocation decisions are made at the start of each time step instead of at the arrival of new tasks. Therefore, decisions are based on more information and can achieve higher social welfare. However, intuitively, users may get more utility by reporting their tasks within a time step that are "less competitive", which means that during that time step only few high-value tasks are reported. Thus, the online auction mechanisms need to be carefully designed to be truthful. Another approach to design a truthful mechanism that is more efficient in terms of social welfare is to combine mechanism design and reinforcement learning techniques. In reality, the external environment can be dynamic, and the pattern of IoT tasks can gradually change over time. For example, the percentage of high-value tasks may be much higher this week than last week. This approach is promising since mechanisms designed using this approach can achieve higher social welfare by automatically adjusting to dynamic environments. For example, there is research showing that reinforcement learning can achieve high efficiency in a dynamic environment (Aydin and Öztemel, 2000; Cui *et al.*, 2017). However, users may misreport tasks to increase their utility immediately or to influence what the mechanism

learns and gain advantage in the future. So the mechanism needs to ensure that any users cannot be better off at present or in the future by misreporting their tasks. For this reason, we also need to carefully design the reinforcement learning mechanism to make it truthful and efficient at the same time.

The following figure is a rough work plan for the next 15 months until the PhD thesis. It contains the plan for the above future work, and the plan for writing them up into a journal paper and a conference paper respectively. Finally, I will write all our work into a PhD thesis.

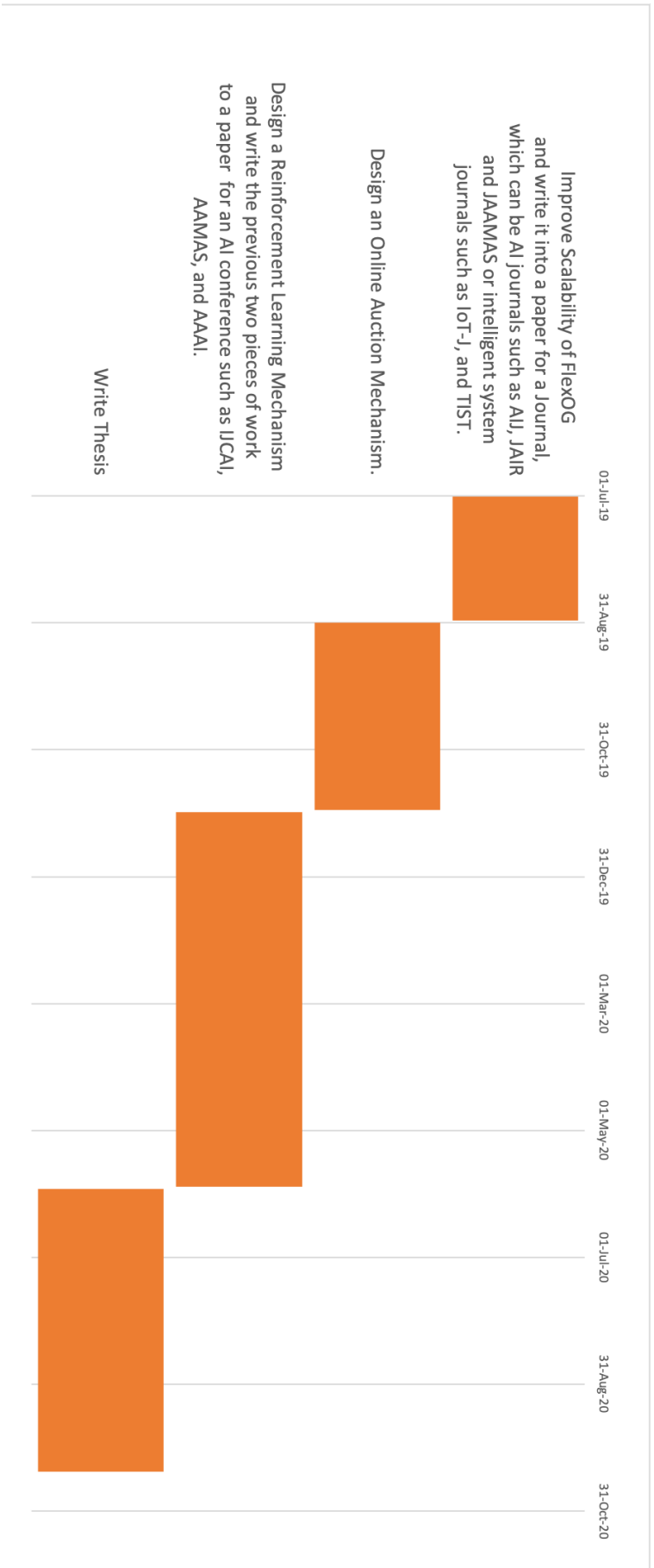

FIGURE 5.1: Gantt chart up to PhD thesis

References

Mohammad Aazam and Eui-Nam Huh. Fog computing micro datacenter based dynamic resource estimation and pricing model for iot. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, pages 687–694. IEEE, 2015.

Swati Agarwal, Shashank Yadav, and Arun Kumar Yadav. An efficient architecture and algorithm for resource provisioning in fog computing. *International Journal of Information Engineering and Electronic Business*, 8(1):48–61, 2016.

May Al-Roomi, Shaikha Al-Ebrahim, Sabika Buqrais, and Imtiaz Ahmad. Cloud computing pricing models: a survey. *International Journal of Grid and Distributed Computing*, 6(5):93–106, 2013.

Muhammad Ali, Ashiq Anjum, M Usman Yaseen, A Reza Zamani, Daniel Balouek-Thomert, Omer Rana, and Manish Parashar. Edge enhanced deep learning system for large-scale video stream analytics. In *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*, pages 1–10. IEEE, 2018.

Arwa Alrawais, Abdulrahman Alhothaily, Chunqiang Hu, and Xiuzhen Cheng. Fog computing for the internet of things: Security and privacy issues. *IEEE Internet Computing*, 21(2):34–42, 2017.

Aymen Abdullah Alsaffar, Hung Phuoc Pham, Choong-Seon Hong, Eui-Nam Huh, and Mohammad Aazam. An architecture of iot service delegation and resource allocation based on collaboration between fog and cloud computing. *Mobile Information Systems*, 2016, 2016.

Palo Alto. Saguna and gridraster partner to bring bigh-quality vr/ar experiences to mobile devices by leveraging multi-access edge computing. `https://www.prnewswire.com/news-releases/saguna-and-gridraster-partner-to-bring-high-quality-vrar-experiences-to-mobile-devices-by-leveraging-multi-access-edge-computing-682157481.html`, May 2018.

Ganesh Ananthanarayanan, Paramvir Bahl, Peter Bodík, Krishna Chintalapudi, Matthai Philipose, Lenin Ravindranath, and Sudipta Sinha. Real-time video analytics: The killer app for edge computing. *Computer*, 50(10):58–67, 2017.

Hany Atlam, Robert Walters, and Gary Wills. Fog computing and the internet of things: a review. *Big Data and Cognitive Computing*, 2(2), 2018.

M Emin Aydin and Ercan Öztemel. Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems*, 33(2-3):169–178, 2000.

Yossi Azar, Umang Bhaskar, Lisa Fleischer, and Debmalya Panigrahi. Online mixed packing and covering. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 85–100. Society for Industrial and Applied Mathematics, 2013.

Yossi Azar, Inna Kalp-Shaltiel, Brendan Lucier, Ishai Menache, Joseph Seffi Naor, and Jonathan Yaniv. Truthful online scheduling with commitments. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 715–732. ACM, 2015.

Yair Bartal, Rica Gonen, and Noam Nisan. Incentive compatible multi unit combinatorial auctions. In *Proceedings of the 9th conference on Theoretical aspects of rationality and knowledge*, pages 72–87. ACM, 2003.

Russell Bent and Pascal Van Hentenryck. The value of consensus in online stochastic scheduling. In *ICAPS*, volume 4, pages 219–226, 2004.

Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

Luiz Fernando Bittencourt, Marcio Moraes Lopes, Ioan Petri, and Omer F Rana. Towards virtual machine migration in fog computing. In *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pages 1–8. IEEE, 2015.

Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

Piotr Borylo, Artur Lason, Jacek Rzasa, Andrzej Szymanski, and Andrzej Jajszczyk. Energy-aware fog and cloud interplay supported by wide area software defined networking. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2016.

Alessio Botta, Walter De Donato, Valerio Persico, and Antonio Pescapé. Integration of cloud computing and internet of things: a survey. *Future generation computer systems*, 56:684–700, 2016.

Sylvain Bouveret and Jérôme Lang. A general elicitation-free protocol for allocating indivisible goods. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

Frederick P Brooks. What's real about virtual reality? *IEEE Computer graphics and applications*, 19(6):16–27, 1999.

Kuan-Ta Chen, Yu-Chun Chang, Po-Han Tseng, Chun-Ying Huang, and Chin-Laung Lei. Measuring the latency of cloud gaming systems. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1269–1272. ACM, 2011.

Shanzhi Chen, Hui Xu, Dake Liu, Bo Hu, and Hucheng Wang. A vision of iot: Applications, challenges, and opportunities with china perspective. *IEEE Internet of Things journal*, 1(4):349–359, 2014.

Ning Chen, Yu Chen, Yang You, Haibin Ling, Pengpeng Liang, and Roger Zimmermann. Dynamic urban surveillance video stream processing using fog computing. In *Multimedia Big Data (BigMM), 2016 IEEE Second International Conference on*, pages 105–112. IEEE, 2016.

Yann Chevaleyre, Paul E Dunne, Ulle Endriss, Jérôme Lang, Michel Lemaitre, Nicolas Maudet, Julian Padget, Steve Phelps, Juan A Rodriguez-Aguilar, and Paulo Sousa. Issues in multiagent resource allocation. *Informatica*, 30(1), 2006.

Annalisa Cocchia. Smart and digital city: A systematic literature review. In *Smart city*, pages 13–43. Springer, 2014.

Delong Cui, Zhiping Peng, Weiwei Lin, et al. A reinforcement learning-based mixed job scheduler scheme for grid or iaas cloud. *IEEE Transactions on Cloud Computing*, 2017.

Dominique Demougin and Claude Fluet. Monitoring versus incentives. *European Economic Review*, 45(9):1741–1764, 2001.

Paul Dempsey. The teardown: HTC Vive VR headset. *Engineering & Technology*, 11(7-8):80–81, 2016.

Ruilong Deng, Rongxing Lu, Chengzhe Lai, and Tom H Luan. Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing. In *2015 IEEE International Conference on Communications (ICC)*, pages 3909–3914. IEEE, 2015.

Parth Rajesh Desai, Pooja Nikhil Desai, Komal Deepak Ajmera, and Khushbu Mehta. A review paper on oculus rift-a virtual reality headset. *arXiv preprint arXiv:1408.1173*, 2014.

Jianbo Du, Liqiang Zhao, Jie Feng, and Xiaoli Chu. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Transactions on Communications*, 66(4):1594–1608, 2018.

Lee Alan Dugatkin and Hudson Kern Reeve. *Game theory and animal behavior*. Oxford University Press on Demand, 2000.

Mica R Endsley. Autonomous driving systems: A preliminary naturalistic study of the tesla model s. *Journal of Cognitive Engineering and Decision Making*, 11(3):225–238, 2017.

Muhammad Junaid Farooq and Quanyan Zhu. Adaptive and resilient revenue maximizing dynamic resource allocation and pricing for cloud-enabled iot systems. In *2018 Annual American Control Conference (ACC)*, pages 5292–5297. IEEE, 2018.

Rainer Feldmann, Martin Gairing, Thomas Lücking, Burkhard Monien, and Manuel Rode. Selfish routing in non-cooperative networks: A survey. In *International Symposium on Mathematical Foundations of Computer Science*, pages 21–45. Springer, 2003.

Niroshinie Fernando, Seng W Loke, and Wenny Rahayu. Mobile cloud computing: A survey. *Future generation computer systems*, 29(1):84–106, 2013.

Tiago M. Fernández-Caramés, Paula Fraga-Lamas, Manuel Suárez-Albela, and Miguel Vilar-Montesinos. A fog computing and cloudlet based augmented reality system for the industry 4.0 shipyard. *Sensors*, 18(6), 2018.

B Friedlander. A decentralized strategy for resource allocation. *IEEE Transactions on Automatic Control*, 27(1):260–265, 1982.

Jun-Song Fu, Yun Liu, Han-Chieh Chao, Bharat K Bhargava, and Zhen-Jiang Zhang. Secure data storage and searching for industrial iot by integrating fog computing and cloud computing. *IEEE Transactions on Industrial Informatics*, 14(10):4519–4528, 2018.

Mariangely Iglesias Pena Anastacia MacAllister Eliot Winer Gabriel Evans, Jack Miller. Evaluating the microsoft hololens through an augmented reality assembly application, 2017.

Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, 45(5):37–42, 2015.

Enrico H Gerding, Valentin Robu, Sebastian Stein, David C Parkes, Alex Rogers, and Nicholas R Jennings. Online mechanism design for electric vehicle charging. In

The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2, pages 811–818. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

Enrico H. Gerding, Sebastian Stein, Valentin Robu, Dengji Zhao, and Nicholas R. Jennings. Two-sided online markets for electric vehicle charging. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 989–996, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

Michel Goemans, Vahab Mirrokni, and Adrian Vetta. Sink equilibria and convergence. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 142–151. IEEE, 2005.

Peter Gradwell and Julian Padget. Distributed combinatorial resource scheduling. In *Proc. AAMAS Workshop on Smart Grid Technologies (SGT-2005)*, 2005.

Albert Greenberg, James Hamilton, David A Maltz, and Parveen Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM computer communication review*, 39(1):68–73, 2008.

QL Gu, TG Gao, and LS Shi. Price decision analysis for reverse supply chain based on game theory. *Systems Engineering-theory & Practice*, 3:21–25, 2005.

Yunan Gu, Zheng Chang, Miao Pan, Lingyang Song, and Zhu Han. Joint radio and computational resource allocation in iot fog computing. *IEEE Transactions on Vehicular Technology*, 67(8):7475–7484, 2018.

Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296, 2017.

Varun Gupta, Benjamin Moseley, Marc Uetz, and Qiaomin Xie. Stochastic online scheduling on unrelated machines. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 228–240. Springer, 2017.

Peter Hammerstein and Reinhard Selten. Game theory and evolutionary biology. *Handbook of game theory with economic applications*, 2:929–993, 1994.

Keiichiro Hayakawa, Enrico H. Gerding, Sebastian Stein, and Takahiro Shiga. Online mechanisms for charging electric vehicles in settings with varying marginal electricity costs. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 2610–2616. AAAI Press, 2015.

Keiichiro Hayakawa, Enrico H Gerding, Sebastian Stein, and Takahiro Shiga. Price-based online mechanisms for settings with uncertain future procurement costs and multi-unit

demand. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 309–317. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Linhai He and Jean Walrand. Pricing and revenue sharing strategies for internet service providers. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 1, pages 205–216. IEEE, 2005.

Dorothea Herreiner and Clemens Puppe. A simple procedure for finding equitable allocations of indivisible goods. *Social Choice and Welfare*, 19(2):415–430, 2002.

Leonid Hurwicz. The design of mechanisms for resource allocation. *The American Economic Review*, 63(2):1–30, 1973.

Anthony Kelly. *Decision making using game theory: an introduction for managers.* Cambridge University Press, 2003.

Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413. Springer, 1999.

Diego Kreutz, Fernando MV Ramos, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

Wei-Yu Lin, Guan-Yu Lin, and Hung-Yu Wei. Dynamic auction mechanism for cloud resource allocation. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 591–592. IEEE Computer Society, 2010.

Dapeng Liu, JC Zuniga, Pierrick Seite, H Chan, and CJ Bernardos. Distributed mobility management: Current practices and gap analysis. Technical report, 2015.

Gaocheng Liu, Shuai Liu, Khan Muhammad, Arun Kumar Sangaiah, and Faiyaz Doctor. Object tracking in vary lighting conditions for fog based intelligent surveillance of public spaces. *IEEE Access*, 6:29283–29296, 2018.

Brendan Lucier, Ishai Menache, Joseph Seffi Naor, and Jonathan Yaniv. Efficient online scheduling for deadline-sensitive jobs. In *Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures*, pages 305–314. ACM, 2013.

Knud Lasse Lueth. State of the iot 2018: Number of iot devices now at 7b – market accelerating. `https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/`, 2018.

Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. Fog computing: A taxonomy, survey and future directions. In *Internet of everything*, pages 103–130. Springer, 2018.

Bernard Marr. Machine learning in practice: How does amazon's alexa really work? `https://www.forbes.com/sites/bernardmarr/2018/10/05/how-does-amazons-alexa-really-work/#740e537b1937`, 2018. Accessed: 201-03-30.

Christian Matt. Fog computing. *Business & Information Systems Engineering*, pages 1–5, 2018.

Mahammad Shareef Mekala and P Viswanathan. A survey: Smart agriculture iot with cloud computing. In *2017 International conference on Microelectronic Devices, Circuits and Systems (ICMDCS)*, pages 1–7. IEEE, 2017.

Oliver J Muensterer, Martin Lacher, Christoph Zoeller, Matthew Bronstein, and Joachim Kübler. Google glass in pediatric surgery: an exploratory study. *International journal of surgery*, 12(4):281–289, 2014.

Iyswarya Narayanan, Aman Kansal, Anand Sivasubramaniam, Bhuvan Urgaonkar, and Sriram Govindan. Towards a leaner geo-distributed cloud infrastructure. In *6th USENIX Workshop on Hot Topics in Cloud Computing*, 2014.

Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 1–12, New York, NY, USA. ACM.

Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*. Cambridge university press, 2007.

Feyza Yildirim Okay and Suat Ozdemir. A fog computing based smart grid model. In *2016 international symposium on networks, computers and communications (ISNCC)*, pages 1–6. IEEE, 2016.

Jessica Oueis, Emilio Calvanese Strinati, and Sergio Barbarossa. The fog balancing: Load distribution for small cell cloud computing. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–6. IEEE, 2015.

Simon Parsons and Michael Wooldridge. Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 5(3):243–254, 2002.

Nisha Peter. Fog computing and its real time applications. *International Journal of Emerging Technology and Advanced Engineering*, 5(6):266–269, 2015.

Tariq Qayyum, Asad Waqar Malik, Muazzam A Khan Khattak, Osman Khalid, and Samee U Khan. Fognetsim++: A toolkit for modeling and simulation of distributed fog environment. *IEEE Access*, 6:63570–63583, 2018.

Tao Qin, Wei Chen, and Tie-Yan Liu. Sponsored search auctions: Recent advances and future directions. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):60, 2015.

Dadmehr Rahbari and Mohsen Nickray. Scheduling of fog networks with optimized knapsack by symbiotic organisms search. In *2017 21st Conference of Open Innovations Association (FRUCT)*, pages 278–283. IEEE, 2017.

Charles Reiss, John Wilkes, and Joseph L Hellerstein. Google cluster-usage traces: format+ schema. *Google Inc., White Paper*, pages 1–14, 2011.

Vincent Ricquebourg, David Menga, David Durand, Bruno Marhic, Laurent Delahoche, and Christophe Loge. The smart home concept: our immediate future. In *2006 1st IEEE international conference on e-learning in industrial electronics*, pages 23–28. IEEE, 2006.

Vasja Roblek, Maja Meško, and Alojz Krapež. A complex view of industry 4.0. *Sage Open*, 6(2):1–11, 2016.

Valentin Robu, Sebastian Stein, Enrico H Gerding, David C Parkes, Alex Rogers, and Nicholas R Jennings. An online mechanism for multi-speed electric vehicle charging. In *International Conference on Auctions, Market Mechanisms and Their Applications*, pages 100–112. Springer, 2011.

B Myerson Roger. Game theory: analysis of conflict. *The President and Fellows of Harvard College, USA*, 1991.

Jörg Rothe. *Economics and computation*, volume 4. Springer, 2015.

Subhadeep Sarkar, Subarna Chatterjee, and Sudip Misra. Assessment of the suitability of fog computing in the context of internet of things. *IEEE Transactions on Cloud Computing*, 6(1):46–59, 2018.

Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, (4):14–23, 2009.

Elham Semsar-Kazerooni and Khashayar Khorasani. Multi-agent team cooperation: A game theory approach. *Automatica*, 45(10):2205–2213, 2009.

Jaydip Sen. Security and privacy issues in cloud computing. In *Cloud Technology: Concepts, Methodologies, Tools, and Applications*, pages 1585–1630. IGI Global, 2015.

Weijie Shi, Chuan Wu, and Zongpeng Li. Rsmoa: A revenue and social welfare maximizing online auction for dynamic cloud resource provisioning. In *Quality of Service (IWQoS), 2014 IEEE 22nd International Symposium of*, pages 41–50. IEEE, 2014.

Weijie Shi, Linquan Zhang, Chuan Wu, Zongpeng Li, and Francis Lau. An online auction framework for dynamic resource provisioning in cloud computing. *ACM SIGMETRICS Performance Evaluation Review*, 42(1):71–83, 2014.

Weijie Shi, Chuan Wu, and Zongpeng Li. A shapley-value mechanism for bandwidth on demand between datacenters. *IEEE Transactions on Cloud Computing*, 6(1):19–32, 2015.

Weijie Shi, Linquan Zhang, Chuan Wu, Zongpeng Li, Francis Lau, Weijie Shi, Linquan Zhang, Chuan Wu, Zongpeng Li, and Francis Lau. An online auction framework for dynamic resource provisioning in cloud computing. *IEEE/ACM Transactions on Networking (TON)*, 24(4):2060–2073, 2016.

Weijie Shi, Chuan Wu, and Zongpeng Li. An online auction mechanism for dynamic virtual cluster provisioning in geo-distributed clouds. *IEEE Transactions on Parallel and Distributed Systems*, 28(3):677–688, 2017.

Shailendra Singh and Abdulsalam Yassine. Iot big data analytics with fog computing for household energy management in smart grids. In *International Conference on Smart Grid and Internet of Things*, pages 13–22. Springer, 2018.

Sebastian Stein, Enrico Gerding, Valentin Robu, and Nicholas R Jennings. A model-based online mechanism with pre-commitment and its application to electric vehicle charging. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 669–676. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

Philipp Ströhle, Enrico H Gerding, Mathijs M de Weerdt, Sebastian Stein, and Valentin Robu. Online mechanism design for scheduling non-preemptive jobs under uncertain supply and demand. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 437–444. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

Kehua Su, Jie Li, and Hongbo Fu. Smart city and the applications. In *Electronics, Communications and Control (ICECC), 2011 International Conference on*, pages 1028–1031. IEEE, 2011.

Hassan Takabi, James BD Joshi, and Gail-Joon Ahn. Security and privacy challenges in cloud computing environments. *IEEE Security & Privacy*, 8(6):24–31, 2010.

Mohit Taneja and Alan Davy. Resource aware placement of iot application modules in fog-cloud computing paradigm. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 1222–1228. IEEE, 2017.

Fan TongKe. Smart agriculture based on cloud computing and IOT. *Journal of Convergence Information Technology*, 8(2), 2013.

Eva Marín Tordera, Xavi Masip-Bruin, Jordi Garcia-Alminana, Admela Jukan, Guang-Jie Ren, Jiafeng Zhu, and Josep Farré. What is a fog node a tutorial on current concepts towards a common definition. *arXiv preprint arXiv:1611.09193*, 2016.

Maria Lorena Tuballa and Michael Lochinvar Abundo. A review of the development of smart grid technologies. *Renewable and Sustainable Energy Reviews*, 59:710–725, 2016.

Jianzong Wang, Yanjun Chen, Daniel Gmach, Changsheng Xie, Jiguang Wan, and Rui Hua. pcloud: an adaptive i/o resource allocation algorithm with revenue consideration over public clouds. In *International Conference on Grid and Pervasive Computing*, pages 16–30. Springer, 2012.

Qian Wang, Kui Ren, and Xiaoqiao Meng. When cloud meets ebay: Towards effective pricing for cloud computing. In *INFOCOM, 2012 Proceedings IEEE*, pages 936–944. IEEE, 2012.

Wei Wang, Ben Liang, and Baochun Li. Revenue maximization with dynamic auctions in iaas cloud markets. In *2013 IEEE/ACM 21st International Symposium on Quality of Service (IWQoS)*, pages 1–6. IEEE, 2013.

Wanyuan Wang, Yichuan Jiang, and Weiwei Wu. Multiagent-based resource allocation for energy minimization in cloud computing systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(2):205–220, 2017.

Chao Wei, Zubair Md Fadlullah, Nei Kato, and Ivan Stojmenovic. On optimally reducing power loss in micro-grids with power storage devices. *IEEE Journal on Selected Areas in Communications*, 32(7):1361–1370, 2014.

Chuan Wu, Zongpeng Li, Xuanjia Qiu, and Francis Lau. Auction-based p2p vod streaming: Incentives and optimal scheduling. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 8(1S):14, 2012.

Yu Xiao and Chao Zhu. Vehicular fog computing: Vision and challenges. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 6–9. IEEE, 2017.

Xiao Yang, Zhiyong Chen, Kuikui Li, Yaping Sun, Ning Liu, Weiliang Xie, and Yong Zhao. Communication-constrained mobile edge computing systems for wireless virtual reality: Scheduling and tradeoff. *IEEE Access*, 6:16665–16677, 2018.

Shanhe Yi, Cheng Li, and Qun Li. A survey of fog computing: concepts, applications and issues. In *Proceedings of the 2015 workshop on mobile big data*, pages 37–42. ACM, 2015.

Quan Yuan, Haibo Zhou, Jinglin Li, Zhihan Liu, Fangchun Yang, and Xuemin Sherman Shen. Toward efficient content delivery for automated driving services: An edge computing solution. *IEEE Network*, 32(1):80–86, 2018.

Deze Zeng, Lin Gu, Song Guo, Zixue Cheng, and Shui Yu. Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. *IEEE Transactions on Computers*, 65(12):3702–3712, 2016.

Xiaoxi Zhang, Zhiyi Huang, Chuan Wu, Zongpeng Li, and Francis Lau. Online auctions in iaas clouds: Welfare and profit maximization with server costs. In *ACM SIGMETRICS Performance Evaluation Review*, volume 43, pages 3–15. ACM, 2015.

Xiaoxi Zhang, Chuan Wu, Zongpeng Li, and Francis CM Lau. A truthful (1-ε)-optimal mechanism for on-demand cloud resource provisioning. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 1053–1061. IEEE, 2015.

Hong Zhang, Hongbo Jiang, Bo Li, Fangming Liu, Athanasios V Vasilakos, and Jiangchuan Liu. A framework for truthful online auctions in cloud computing with heterogeneous user demands. *IEEE Transactions on Computers*, 65(3):805–818, 2016.

Zijun Zhang, Zongpeng Li, and Chuan Wu. Optimal posted prices for online cloud resource allocation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(1):23, 2017.

Ruiting Zhou, Zongpeng Li, Chuan Wu, and Zhiyi Huang. An efficient cloud market mechanism for computing jobs with soft deadlines. *IEEE/ACM Transactions on networking*, 25(2):793–805, 2017.

Yuefei Zhu, Baochun Li, and Zongpeng Li. Truthful spectrum auction design for secondary networks. In *INFOCOM, 2012 Proceedings IEEE*, pages 873–881. IEEE, 2012.