A Pricing Mechanism for Task Oriented Resource Allocation in Cloud Robotics

Lujia Wang, Ming Liu and Max Q.-H. Meng

Abstract Cloud robotics is currently driving interests in both academia and industry, especially for systems with limited computation capability. Resource allocation is the fundamental and dominant problem for resource sharing among agents in the cloud robotics system. This chapter introduces a novel resource allocation framework for cloud robotics and proposes a Stackelberg game model and the corresponding task oriented pricing mechanism for resource allocation. Simulation investigates the parameter selection and time cost of the proposed mechanism. Experimental results of co-localization task demonstrate that the proposed mechanism achieve an optimal performance in resource allocation.

Keywords Pricing algorithm · Resource allocation · Cloud robotics

1 Introduction

Nowadays, there is a growing need for service robots in human daily life, and the involved services are more complicated than ever before. For traditional robotic systems, robots have to carry adequate physical processing power and various sensors among other resources to facilitate the completion of various tasks such as visual navigation [38], range-finder-based navigation [41, 43], path planning [14], recog-

L. Wang (✉)
School of Electrical and Electronic Engineering, Nanyang Techonological University, Singapore, Singapore
e-mail: wanglj@ntu.edu.sg

M. Liu
Department of Mechanical and Biomedical Engineering, City University of Hong Kong, Hong Kong, Hong Kong
e-mail: mingliu@cityu.edu.hk

M.Q.-H. Meng
Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, Hong Kong
e-mail: qhmeng@ee.cuhk.edu.hkk

A. Koubaa and E. Shakshuki (eds.), *Robots and Sensor Clouds*,
Studies in Systems, Decision and Control 36,
DOI 10.1007/978-3-319-22168-7_1

nition [40] and scene analysis [39, 42]. However, developing a practical robot that can cover many services would be extremely expensive and require a long time. It is thus reasonable to combine multiple robots that have limited capabilities, and access variety of information or services. This leads to the so-called paradigm "Cloud Robotics", which combines robot technology with ubiquitous network and cloud-computing infrastructures that link a lot of robots, sensors, portable devices and data centers. Therefore, robots can be remitted from hardware limitations while benefit from plenty of resources and computing capabilities in the cloud. However, resource competition is pervasive in practical applications for networked robotics today. It necessitates the allocation of limited bandwidth as an essential problem to be taken into account for the system design.

The authors of [25] first described a dual-level system architecture for cloud robotics, consisting of a machine-to-machine (M2M) level and a machine-to-cloud (M2C) level. On the M2M level, a team of robots communicates via wireless links such as Local Area Network (LAN) or Mobile Ad-hoc Networks (MANETs). On the M2C level, the infrastructure cloud provides a pool of shared sensor data, computation and storage resources, to be allocated among robotic agents. Considering the aforementioned dual-level architecture, this chapter presents a novel framework of a cloud robotic system. It consists of networked robots and a cloud-computing infrastructure. The latter connects the robots, sensors, portable devices and most importantly a centralized data-center. By adopting such a proxy-based model, all primary data can be retrieved from the cloud and managed by the proxy so that the requirements on hardware for each robot can be minimized. In addition, the proposed pricing resource allocation mechanism is task-oriented, which focuses on completing the necessary task or series of tasks in order to achieve optimized resource allocation.

Briefly speaking, this chapter deals with the resource allocation problem for cloud robotics by using a market-based mechanism. The following major contributions are addressed.

- A novel cloud robotic architecture is proposed based on an asynchronous data flow framework [70] for resource allocation managements among multiple robots. Especially, the architecture of cloud robotics is classified as an inter-cloud formed by robot-to-robot (R2R) and an infrastructure cloud enabled by robot-to-cloud (R2C).
- A Stackelberg game-based [45] resource management mechanism is proposed with consideration of the interaction among robot clients. The mechanism optimization is theoretically proved and implemented as functionalities of admission control, request ranking and resource distributing. Besides, a data buffer is set up on the access proxy for frequently requested data.
- A set of task-oriented Quality-of-service (QoS) criteria are proposed as the primary assessment metric of a co-localization scenario. The QoS's are defined regarding the fact that sophisticated collaborative robotic tasks are usually time sensitive.

The rest of the chapter is organized as follows. In Sect. 2, we discuss the related work in the area of resource allocation and cloud robotics. Section 3 presents our design of a typical cloud robotic system with a resource management middleware.

Afterwards, we define criteria of QoS at the end of the section. In order to solve the inherent conflicts of MSDR, we introduce the theoretical modeling and solution in Sect. 4. In Sect. 5, the parameter investigation and time cost of the proposed mechanism are presented. The experimental setup and results analysis are demonstrated in Sect. 6. At last, Sect. 7 presents the conclusion and future work.

2 Related Work

In this section, current works in the aspect of service-oriented architecture, cloud robotic systems, robotic task allocation and resource allocation mechanisms are reviewed and discussed.

2.1 Service-Oriented Architecture in Cloud Computing

The Service-oriented architecture (SOA) [6] is a widely used framework for cloud computing, where the cloud hosts and clients are synthesized under an elastic architecture. It represents computing in three parallel processes: service development, service publication and application composition using services that have been published. Cloud computing extends the scope of SOA by including the development of platform and infrastructure. So far, it is usually characterized by four paradigms: Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS) and Hardware as a Service (HaaS) [55]. Cloud computing can speed up many computationally intensive robotic and automation systems or applications, such as Simultaneous Localization and Mapping (SLAM) [54], robotic big data analysis, sample-based uncertainty sensing model analysis [32]. Thus, Robot as a Service (RaaS) [10] can also be added to such a scope. Most existing works using this framework are web service-based or database dependent. All major information technology companies and providers, including Google [9], IBM [13], Intel [27], Oracle [20], SAP [46] have adopted and supported this computing paradigm.

Because of the heterogeneous service and data that are discussed in [59], the cloud is usually addressed by a common middleware to get interoperability. Many researchers have worked on the resource management that is constrained in the field of e-commerce and enterprise computing systems, such as Eucalypus of Amazon Elastic Computing Cloud (EC2) [50], OpenNebula [51] and Nimbus [58]. Specifically, for RaaS, SOA can also be introduced. For example, Microsoft Robotics Developer Studio (MRDS) [15] was a vital product in applying SOA to embedded systems [63, 64]. Microsoft also released Visual Programming Language (VPL) in 2006 [57], which marked a milestone in SOA and in robotics. Many robot manufacturers have used VPL as their programming platform, including Coroware, iRobot, Kuka, LEGO NXT Mindstorm, Parallax, Robosoft, Robotics Connection, Whitebox Robotics, MOLMC IntoRobots, etc [15]. In addition, many of them use a web-based platform to configure the infrastructure that processing power, memory capacity, and communication bandwidth.

2.2 Current Cloud Robotic Systems

Although the concept of networked robots can date back to the 1990s [52], cloud robotics is now in better condition for both network and robot to provide an innovated outtake.

A large number of works took advantage of the big data in the cloud to simplify algorithms in robotics. ASORO laboratory in Singapore built a cloud computing infrastructure "DAvinCi" [3] to generate 3D models of environments, which allowed robots to perform SLAM. The Google autonomous driving project indexed maps and images that were collected and updated by satellite, street view, and crowdsourcing to facilitate accurate localization. The "cloud-based robot grasping" [31] used the Google Object Recognition Engine to recognize and grasp common household objects.

Some works provide designs of programming, middleware management framework enable robots sharing of data in the cloud. Google and Willow Garage initialized a software enables an Android phone to control robots based on platforms such as Lego Mindstroms, iRobot create and Vex Pro. MOLMC provides a sophisticated solution to an internet of things using MQTT protocol. The Gostainet [22] is an infrastructure of cloud robotic for executing the speech recognition on humanoid robot Nao [1]. This system is used to improve the interactions with children as part of a research project at a hospital in Italy. Authors of [26] presented a PaaS based cloud engine Rapyuta, which can allocate secure computing environments for robots.

Some works focus on accessing to databases of robotic sensing data such as maps and images. RoboEarth [67] is built for robots to autonomously share descriptions of environments and object models. Microsoft implemented a RaaS [10] platform, which included services for performing functionality, service broker for discovery and publishing, and applications for clients' direct access.

The aforementioned research took advantage of a wide range of online data resources, which is one of the most meaningful fields at the current stage. However, many robotic systems have a very strict assumption, such as *the resource in the cloud is unlimited*. In the matter of fact, most resources in cloud robotics systems are indeed limited [68]. For instance, network bandwidth for transmitting image data, CPU occupancy for parallel computation, as well as available number of hosts (proxy) are limited. Therefore, how to design a module that maximizes the utility of available resources on demand is a quite challenging problem, especially when multiple robots request the same kind of resource or service in an asynchronous manner.

2.3 Robotic Task Allocation Mechanisms

In robotics, tasks are usually regarded as resources to be allocated to robots in a collaborative system. Therefore, multi-robot task allocation (MRTA) problems are widely studied and can be characterized as the following types [21].

- *ST-SR-IA* is a simple optimal assignment problem and can be solved by both the centralized algorithm [8] and the distributed algorithm [37]. Centralized approaches usually can find the optimal faster than distributed approaches, but lead to a higher communication overhead.
- *ST-SR-TA* is an instance of problems when the task information and utility of robots can be predicted with some accuracy. This problem is building a time-extended schedule of tasks for each robot, with the goal of minimizing total weighted cost [65].
- *ST-MR-IA* is a kind of problems that involve tasks that require the combination of multiple robots. It is referred as coalition formation in the multi-agent community and is more difficult than the previously mentioned MRTA problems. The authors of [66] proposed a service-based approach with the principal idea that a robot can ask for services from other robots if the robot cannot execute a task by itself.
- *ST-MR-TA* is a class of problems includes both coalition formation and scheduling. It can be considered as an extension of the ST-MR-IA model with additional scheduling for future allocations. For example, the learning-based probabilistic algorithms [44] and the incremental task allocation algorithms [35] have been proposed.

In [48], the problem is formulated as a set of optimization problems with various objectives:

- *MinMax* (Minimize the maximal cost of nodes): it aims at timely critical missions by finding the shortest mission execution duration, which only concerns the worst node [35].
- *MinSum* (minimize the sum costs of all nodes): it is aimed at optimization of efficiency by minimizing of energy cost. However, it cannot guarantee an optimization on each node, and generally cause some nodes are optimized while some others are not [12].
- *MinAve* (minimize the average cost of all nodes): this objective measures the average time since a task appears in the system until it is completed. It is relevant to the problems where the completion is more important than aggregated global cost [60].

Other objectives are also proposed recently. For instance, minimize the processing time [74], maximize throughput, maximize the utility of the worst node, maximize the sum of individual costs and so on. However, one of the critical challenges in cloud robotic systems is how to optimally manage available resources such as physical robots, and sensor data while considering task constraints. The reasons are two-fold: multi-agent systems are typical complex and distributed, and agents are combined together as an overarching framework for integrated tasks [19]. Therefore, the combined resource allocation should be considered besides robotic task allocation, and current resource allocation mechanisms are reviewed in the next subsection.

2.4 Resource Allocation Mechanisms

In general, resource allocation problems are NP-hard [16, 53], which exist in computation systems, network communications, transportation systems and etc. Resource allocation solutions can be mainly classified into two approaches: one is the optimization-oriented approach which is usually a central planning, and the other is the economic approach which is usually a distributed scheduling.

For the centralized planning, researchers proposed different optimization techniques to minimize the rate of failure and execution time of tasks, or maximize the system utilization and throughput. Colony optimization proposed an algorithm to make an efficient resource assignment for computational jobs being processed, such as Ant colony algorithm [62] which can efficiently solve the resource constrained scheduling problem for mining supply chains. The genetic algorithm is used to solve the optimization problem based on a natural selection process that mimics biological evolution. For example, Rodriguez et al. proposed a particle swarm optimization algorithm for resource providing and scheduling on IaaS cloud to minimize the overall workflow execution cost [56]. Fuzzy logic is a problem-solving control system methodology that lends itself to implementation in various size of systems, such as Fuzzy Clustering Chaotic-based Differential Evolution (FCDE) solved the resource constrained project scheduling problem [11]. Market-based approaches to resource management [2, 28] are characterized by capturing complex interactions among autonomous agents and the system, which suit our problem most. However, most of them have assumptions that are not suitable for practical robotic tasks, such as the boundless communication and computation resources. The limited bandwidth resource should be considered in the real life scenario as presented in [61, 71, 73].

Autonomous negotiation among multiple robots has become a crucial problem in cloud robotic systems when clients query resources in parallel. The key issues of resource allocation for cloud robotics are the uncertain demands for resources such as that in big data mining and computing of robotics, and the large number of unreliable hosts which are physically distributed. Game theory has its advantages in solving this problem since it considers every agent and service provider's profits. For robotic systems, there are also several related works with different structures.

- Centralized approaches: the advantage is that the global knowledge can be used to manage all the available resources optimally while the disadvantage is that time and complexity cost are usually high. For example, the authors of [34] introduced a centralized iterated auction which included three objective functions and six bidding rules for a single task. It firstly demonstrated theoretical guarantees of auction-based methods for such a variety of bidding rules and team objectives. Higuera et al. [24] formulated the task distribution problem as a fair subdivision problem and provided a centralized algorithm to provoke the allocation mechanism for each robot.
- Distributed approaches: these methods are generally low cost since they only use local information, but they cannot achieve the theoretical global optimum. For example, the authors of [65] presented two algorithms for task distribution

problems where multiple tasks can be allocated to a single robot during the negotiation. Later, the authors of [66] presented a distributed market-based algorithm named S + T, which solved a task allocation problem for robot cooperation.

- Combinatorial approaches: allocated resources are a combination of different tasks, rather than a single task in complex systems. In [5, 36], the combinatorial auction was utilized to allocate multiple tasks in a multi-robot system, where robots bid on bundles of targets. They proposed different combinatorial bidding strategies and compared their performances, as well as with single-task auctions. Their computational results indicated that combinatorial auctions generally led to superior team-level performance than single-task auctions.

In general, the above works are based on theoretical analysis and simulation, few real-time robotic scenarios are reported in the real robotic cooperative system. This chapter aims at development of a practical mechanism for real applications.

3 System Architecture of Cloud Robotics

The proposed cloud robotic system is shown in Fig. 1, which includes an R2C network and an R2R network. In the R2C network, an Internet-based cloud infrastructure provides a data center sharing various kinds of sensor data [69]. In the R2R network, a team of robots communicates via wireless links such as LAN or MANETs [72].

3.1 Structure Design

The proposed framework of data retrieval is shown as Fig. 2. It is a host-based network framework which has three main entities involved for supporting Multi-sensor Data

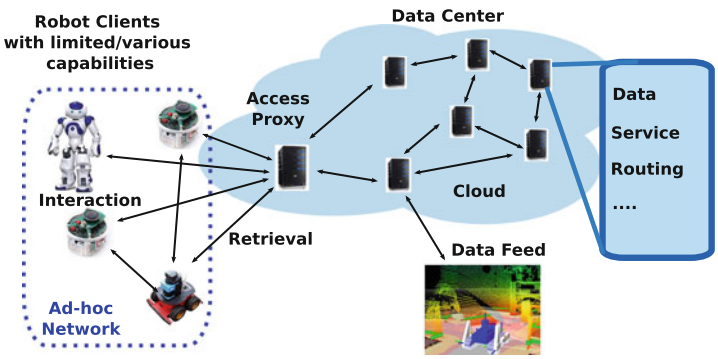


Fig. 1 A architecture of typical cloud robotic systems

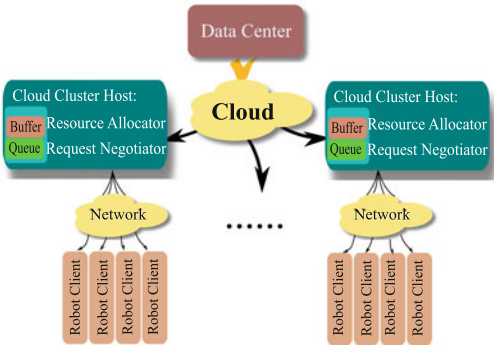


Fig. 2 The data retrieval framework of cloud robotic systems

Retrieval (MSDR) in cloud robotic system, namely, Data Center, Cloud Cluster Host, and Robot Client.

- Data Center (DC): it is a relative database based on PostgreSQL that stores various information, such as point-cloud, images that are established. All data are maintained and shared by any robot client on the network [68]. At the same time, the DC confronts unpredictable parallel requests from the robot clients.
- Cloud Cluster Host (CCH): it is a server that manages a large amount of data retrieval. The CCH consists of two major functionalities: Request Negotiator (RN) and Request Allocator (RA). The RN deals with the interfaces among the robot clients and hosts. It classifies the requests and provides clients with different prices in a pricing scheme. The RA is a function that accesses the cloud for admission control and buffer queue management. The RA distributes resources to robots in term of priority which is derived from the RN.
- Robot Client (RC): at the lowest level of the framework, it is a unit of different kinds of robots with various sensors. RC can be assigned to either an integrated task or separate tasks. Details are introduced in the next subsections.

3.2 Resource Management Framework

In Robotic Operating System (ROS) system, `rosservice` provides task requests and responses among nodes. Although the `actionlib` package provides tools to create servers that execute preemptive long-running goals, it does not support the queue management, especially asynchronous accesses for multiple tasks in the waiting list. Therefore, this is not sufficient for real-time tasks in multi-robot systems. In order to implement the SOA of cloud robotic system, we compared two parallel managements and communication software platforms, Hadoop MapReduce and `twisted` [47]. It is preferable to choose `twisted` as the software platform considering its multi-thread mechanism and compilation of the current database.

Benefits of `twisted` in Parallel Communication `Twisted` [47] is a framework for deploying asynchronous, event-driven and multi-thread supported network system using Python. The obvious advantage is the user-defined structure that can be flexibly applied for various managements. It is composed of the following three primary elements:

- `reactor`
  The `reactor` is the core of the event-driven programming construct in `twisted`. As shown in Fig. 3, it provides a basic interface to a number of services, including network communication, threading, and event dispatching. The application functions can be simply divided into modular and compact parts. It is also easily added specific data query and data retrieval modules for clients and hosts respectively.
- `protocol`
  The `protocol` defines specifications for transmitting and receiving behaviors. Functions of received data and sent data can be constructed following the predefined virtual function names. A `protocol` begins and ends its life with two predefined virtual functions: *connectionMade* and *connectionLost*, which are called whenever a connection is established or dropped, respectively.
- `factory`
  The `factory` is responsible for two tasks: creating new protocols and keeping global configurations and states. The tasks are completed by functions of `buildProtocol` and `management` as shown in Fig. 3. In addition to abstractions of low-level system calls, it also includes a large number of utility functions and classes, which facilitate the establishment of new types of servers. `Twisted` includes the support for popular network protocols, e.g. `SOCKETS`, `HTTP` and `SMTP` etc. It is flexible to define globally visible variables in the `factory`, such as the local data buffer. The host `factory` manages the connections to all the client `reactor` loops. At the same time, it is also in charge of updating the existing relation database, registering with new multiple sensor readings. Details of the framework structure and its assessment are outlined in the following.

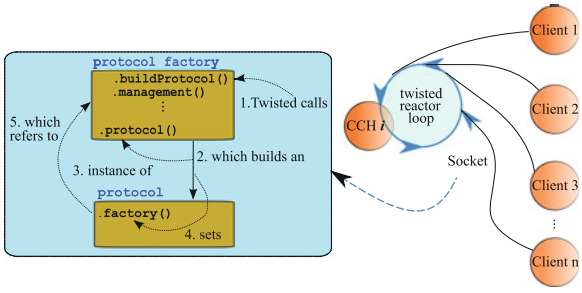


Fig. 3 Protocol creation process in `Twisted` reactor loop between CCH and clients

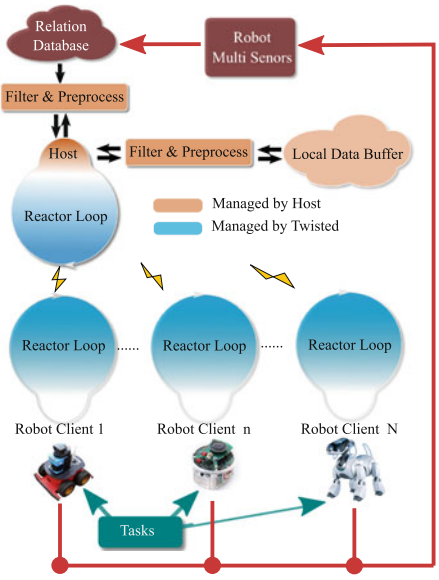


Fig. 4 Data flow of multi-data retrieval and communication in cloud robotic systems

3.3 Data Flow Management

The data flow of multi-data retrieval and communication in our cloud robotic system is shown in Fig. 4. The host and clients are built using the `twisted` framework. The designed program includes a main loop `reactor` and a callback system. This system automatically launches a new thread for each client that attempts to connect the network with a approved address and port. The specific functionalities are defined in the host and clients separately. The major functions in the process are introduced as follows:

- Database Query
  This function is launched and managed only by CCH which retrieves data from DC for RC. It utilizes a standard SQL [18] syntax to retrieve target information from a dynamically updated relation database. The database access may be one bottle-neck in the system, which can be alleviated by the management of the host. To this end, the following two sub-functions is designed to assist the retrieval, namely Filter and Pre-process and Buffer Management and Scheduler.
- Filter and Pre-process
  In the proposed data flow structure, the filter and pre-process blocks stand for general data pre-process. For example, data fusion, feature fusion and decision fusion [17, 23], are the major means to decrease the frequency of database access and reduce data noise.

- Buffer Management
  This function is launched and managed by CCH where a local data buffer is deployed for storage of frequently requested data as depicted in Fig. 4. Because activities of robots are usually regular, the same resource may be queried repetitively. Therefore, the buffered mechanism is built to help alleviate the database access bottle-neck to an extent.
- Scheduler
  Last but not least, the proposed scheduling scheme is launched by CCH that allocates resources for all clients' requests on top of asynchronous communication threads. The asynchronous management based on the `twisted` framework is implemented in CCH to manage all the connections among CCH and robot clients through `reactor` loop in parallel as shown in Fig. 4. Please note that `reactor` loop is a fundamental infrastructure of the `twisted`-based socket, which is used to automate asynchronous data transmission. In addition, the `reactor` loops are running on both CCH and heterogeneous robot clients. The optimization mechanism of resource allocation is modeled as a Stackelberg game. More mechanism details of resource allocation are introduced in the next section.

3.4 Quality of Service Criteria (QoS)

In common sense, bandwidth usage is one of the most important factors to define QoS since the response of most network-based applications is sensitive to it. In cloud robotic systems, instead of taking bandwidth usage as the only criterion, QoS definition can be extended to other aspects regarding the processing or storage capabilities of nodes. The following QoS's are selectively defined as primary criteria to assess the proposed framework.

Definition 3.1: Firm Real-Time Some infrequent deadlines can be missed, which may degrade the system's quality of service. The usefulness of a result is zero after its deadline. In this chapter, FRT is used to measure the performance of such a system. Because FRT is not directly about the delay but about effectivity. It relaxes the certain planned delay and settle with accomplishing the time delay restrictions most of the time.

Definition 3.2: Time of Response (ToR) ToR defines the period that from sending a request to receiving the corresponding response. It is formulated as

$$ToR = T_{\text{Data\_received}} - T_{\text{Request\_sent}}. \tag{1}$$

ToR, including request data transmission period, data matching period and response data transmission period, has been considered because sophisticated collaborating robotic tasks are usually timely sensitive and the long delay of robot's response can

make the task completion meaningless. For instance, cooperative semantic mapping or 3D mapping using several robots is time sensitive. Especially, the data transmission periods are the key factor impacting on the QoS performance.

Definition 3.3: Reliability of Response (RoR) RoR is defined as a success rate of issued data retrievals. Its value is given in percentage and calculated as

$$RoR = \frac{\#_{\text{Succeeded\_Requests}}}{\#_{\text{Total\_Requests}}}. \tag{2}$$

RoR is a key criterion for all services. Typically, in large scale systems, the perception results need to be shared and retrieved with acceptable reliability. The on-board computational capability of the robot clients is usually weak, which implies two inherent requirements as follows.

- The computation and analysis are generally to be off-board from the clients. Therefore, data retrieval from an existing database is inevitable.
- The accuracy of the retrieved data and reliability of the transmission are crucial. Especially, the reliability determines potentials to expand an existing system to a large scale.

QoS criteria advertise performance quality levels of service which are provided by service providers; on the other hand, clients use it to select an optimal candidate data/service, which could in part fulfills the request. Therefore, a well-defined set of QoS's could greatly help the assessment of the quality of a service framework.

4 A Pricing Resource Allocation Mechanism for MSDR

In order to share data and service, the paradigm of cloud robotics enables large numbers of robot clients working in parallel to retrieve multiple data in the cloud. In this section, a system model, MSDR problems, and solutions are proposed as follows.

4.1 System Model

We model the interaction between the cloud service provider and robot client (RC) as a Stackelberg game. The proxy CCH is regarded as the cloud service provider, who sets the price menu for different resource per bandwidth, and RCs respond to the price by presenting a certain amount of requests to the cloud. Suppose that a monopolistic CCH charges RC for usage of a network to maximize profit. Let $N := \{1, \cdots, n\}$ denote the set of RCs, and link of capacity nc accessed by n RCs. For RC i of willingness to pay type ω_i, x_i is usage of bandwidth resource and p_i be the price per unit bandwidth charged by the CCH, then its utility is

$$u_i(t_i, p_i) = \omega_i \cdot \log(1 + t_i) - t_i p_i, \tag{3}$$

where ω_i denotes the willingness to pay of RC i, t_i is the completion time of robot i for resource retrieval. The logarithmic function $\omega_i \log(1 + x_i)$ is verified to ensure a non-trivial and meaningful solution to the Stackelberg game. At the same time, the revenue of CCH is calculated as

$$L(\mathbf{t}, \mathbf{p}) = \sum_{i=1}^{n} t_i p_i, \tag{4}$$

where n is the number of robot clients that are allocated resources.

4.2 Problems and Solutions

The CCH maximizes its revenue by choosing the price p_i and the admitted RC number K for the limited bandwidth,

$$\mathbf{p}^* = \underset{\substack{\mathbf{t}\geq 0 \\ \mathbf{p}\geq 0}}{\operatorname{argmax}} L(\mathbf{t}, \mathbf{p}), \tag{5}$$

RC i determines x_i to maximize its utility and the problem is given by

$$t_i^* = \underset{t_i \geq 0}{\operatorname{argmax}} u_i(t_i, p_i), \tag{6}$$

Constraints are mainly focusing on the deadline of execution time T_0 and the admitted number n as follows

$$\sum_{i=1}^{n} t_i \leq T_0, \quad n = 0, \ldots, N. \tag{7}$$

Remark 1: For each robot client i, the utility function u_i is increasing, strictly concave, and twice continuously differentiable with respect to t_i.

Consider the optimization problem of maximization utility function $u_i : \mathbb{R}^n \rightarrow \mathbb{R}+$, defined in (6), where u_i is twice continuously differentiable in point t_i^*. The *first-order necessary condition* that t_i^* is a local optimum is

$$\frac{\partial u_i(t_i, p_i)}{\partial t_i}|_{t_i = t_i^*} = 0 \tag{8}$$

Therefore, we differentiate the utility function as

$$\begin{aligned}\frac{\partial u_i(t_i, p_i)}{\partial t_i} &= \frac{\partial(\omega_i \cdot \log(1+t_i) - t_i p_i)}{\partial t_i} \\ &= \frac{\omega_i}{1+t_i} - p_i \\ &= 0.\end{aligned} \tag{9}$$

The optimal price of resources queried by robot client i is derived as

$$p_i^* = \frac{\omega_i}{1+t_i^*}. \tag{10}$$

The revenue maximization problem (5) is a non-convex optimization problem with a non-convex objective function. Therefore, we have to convert it into an equivalent convex formulation to solve it. Then the solutions are proposed in the following two steps:

Step 1—Resource allocation: Assuming a fixed admitted RC number K, then plug (10) into (5), the above non-convex optimization problem be easily converted to convex as follows

$$t_i^* = \underset{\substack{\omega_i \geq 0 \\ t_i \geq 0}}{\operatorname{argmax}} \sum_{i=1}^{n} \frac{\omega_i t_i}{1+t_i}. \tag{11}$$

Because the revenue is strictly concave, and the constraint set is convex and compact, this optimization problem admits an optimal solution for the transmission time of robot clients and leads to a unique allocation. Considering the problem in (6) and the constraints in (7), we define the Lagrange function as

$$\Lambda(t_i, p_i, \lambda) = L(t_i, p_i) + \lambda\left(\sum_{i=1}^{n} t_i - T_0\right), \tag{12}$$

where λ is the Lagrange multiplier. By using the Lagrange multiplier technique, we can get the optimal transmission rate that denoted as

$$t_i^* = \sqrt{\frac{\omega_i}{\lambda}} - 1. \tag{13}$$

Step 2—Admission control: Please note the bandwidth constraints (7) must hold equality since the objective is strictly increasing function in t_i. Thus, by plugging the t_i^* into (7), we have

$$\sum_{i=1}^{n}\left(\sqrt{\frac{\omega_i}{\lambda}} - 1\right) = T_0. \tag{14}$$

Assuming that $\omega_1 \geq \omega_2 \geq \cdots \geq \omega_N$, then λ^* must satisfy the above condition (14). For a admitted RC number threshold value K_{th} satisfying

$$\frac{\omega_{K_{th}}}{\lambda^*} > 1 \quad and \quad \frac{\omega_{K_{th}+1}}{\lambda^*} \leq 1, \tag{15}$$

where K_{th} is used for the admission control, so only K_{th} or fewer robot clients can retrieve data. Moreover, we have $\lambda^* = \left(\frac{\sum_{i=1}^{K_{th}} \sqrt{\omega_i}}{T_0 + K_{th}}\right)^2$ derived from (14).

Remark 2: The complexity of **Algorithm 1** is $\mathcal{O}(\mathcal{N})$, which has a linear relationship with the number of robot clients.

Algorithm 1 start to computing λ^* and K_{th} by assuming $K_{th} = N$ and calculate λ. If the condition of (15) is not satisfied, K_{th} is decreased by one and λ is recalculated until it is satisfied. Because $\omega_1 \leq \lambda_1$ and $\lambda_1 = \frac{1}{T_0+1}$, Algorithm 1 always converges and returns the unique value of K_{th} and λ^*.

Algorithm 1: The Revenue Maximization Algorithm

Inputs: ω_i, T_0 and N
Outputs: K_{th}, λ^*, t_i^*, and p_i^*

```
1 BEGIN
2   function Revenue(i, ω_i, T_0, N)
3       k ← N, λ(k) ← (Σ_{i=1}^{k} √ω_i / (T_0+k))^2
4       while ω_k ≤ λ(k) do
5           k ← k − 1, λ(k) ← (Σ_{i=1}^{k} √ω_i / (T_0+k))^2
6       end while
7       K_th ← k, λ* ← λ(k)
8       t_i* = √(ω_i/λ*) − 1
9       p_i* = ω_i/(1+t_i*)
10  return K_th, λ*, t_i*, p_i*
11 END
```

4.3 Scheduling Scheme

All the previous theoretical analysis indicates the proposed scheduling scheme can optimize the MSDR problem. The basic operation of the scheduling scheme is implemented in CCH and comprises the following processes:

- **Admission control**: When a service request is submitted, request negotiator utilizes the proposed admission control mechanism (see **Algorithm 1**) to interpret the request before determining whether to accept or reject it according to the optimal threshold K_{th}. Thus, it ensures that there is no overloading of information, and sufficient requests can be fulfilled successfully. In the `factory`, a threshold is

set for the admitted number of robot clients considering the factors such as ToR and willingness to pay.

- **Request ranking**: The request negotiation is responsible for ranking the admitted requests considering their willingness to pay and time deadline as presented in **Algorithm 2**. Having access to the allocation requests of all robot clients, the CCH can keep tracking current robot clients, and update the ranking list when a new request is registered.
- **Resource distributing**: The admitted requests are responded in accordance with the order in the rank list. In this situation, it optimizes both the utility of each RC and the revenue of CCH. When new requests from robot clients arrive, the resource allocator would response the requests in accordance with the order of the updated rank list.

Algorithm 2: Scheduling Algorithm

Inputs: optimal price p_i^* of request i
Outputs: current_priority_list

```
1 BEGIN
2   function update_priority_list(p_i)
3       current_priority_list.append(p_i)
4   function is_lowest_priority(p_i)
5       current_priority_list.sort(p_i)
6   while i ≤ n_threshold
7       update_priority_list(p_i)
8       is_lowest_priority(p_i)
9   return current_priority_list
10 END
```

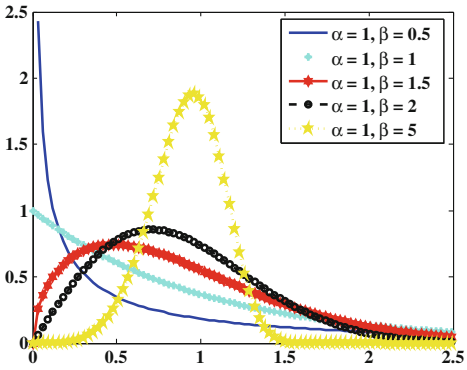
5 Simulation

In this section, the evaluation of mechanism parameter investigation and time cost are presented.

5.1 Parameter Investigation

In the proposed admission control, the K_{th} is determined by the distribution of willingness to pay from robot clients. If there are too many clients with high willingness to pay, the ones with relatively low willingness to pay will not be allocated data. Then the CCH can reduce the K_{th} to fulfill the resource retrieval before the deadline, namely admission control. However, there is no restriction on how to choose willingness to pay as a robot clients. Weibull distribution [49] is chosen because it is a versatile distribution that can represent different kinds of statistical distribution by changing its parameters as shown in Fig. 5. Therefore, Weibull distribution can take on various characteristics based on function as follows:

Fig. 5 Weibull distribution



$$f(x;\alpha,\beta)\begin{cases}\frac{\beta}{\alpha}(\frac{x}{\alpha})^{\beta-1}e^{-(\frac{x}{\alpha})^{\beta}} & x\geq 0,\\ 0 & x<0,\end{cases} \tag{16}$$

where $\alpha \geq 0$ is the scale parameter, and $\beta \geq 0$ is the shape parameter. If the quantity x is the number of clients with a willingness to pay, and the Weibull distribution demonstrates the proportion of the high willingness to pay clients. Then β can be interpreted directly as follows:

- $0 < \beta \leq 1$: $f(x)$ decreases monotonously and is convex as x increases to ∞. Especially, it is an exponential distribution when $\beta = 1$.
- $\beta > 1$: $f(x)$ has a bell-shape, which increases as x increases to the maximum and decreases thereafter. Especially, it is a Rayleigh distribution of mode $\sigma = \frac{\alpha}{\sqrt{2}}$ when $\beta = 2$.

In order to indicate the relationship between willingness to pay and the threshold of admitted number of clients in the proposed admission control, we compare the optimal K_{th} under different distributions of willingness to pay from all clients by tuning three factors: the shape parameter of Weibull distribution β that indicates different distribution of willingness to pay; the *number of clients requested resource* "N"; and the *timeout period* "T_0", which is a required time for a task.

In the simulation, we tested the admission control proposed in Sect. 4 by selecting $\alpha = 1$ and $\beta = \{0.1, 0.5, 1.0, 1.5, 5\}$, the time deadline $T_0 = \{10, 20, 30, 40\}$ and client number $k = \{12, 48, 96, 192\}$ respectively. One hundred runs were carried out on each configuration. In Fig. 6, we can be seen that K_{th} increases as β increases when T_0 is fixed. Especially, the increasing rate of K_{th} when $0 < \beta < 1$ is much larger than the increasing rate when $\beta > 1$. This is because the ratio of clients with high willingness to pay is smaller in the range of $0 < \beta < 1$. Moreover, it also shows that the willingness to pay is a key factor for the designation of the scheduler since it can affect the QoS. Moreover, the above results are references for the evaluation in the next subsection.

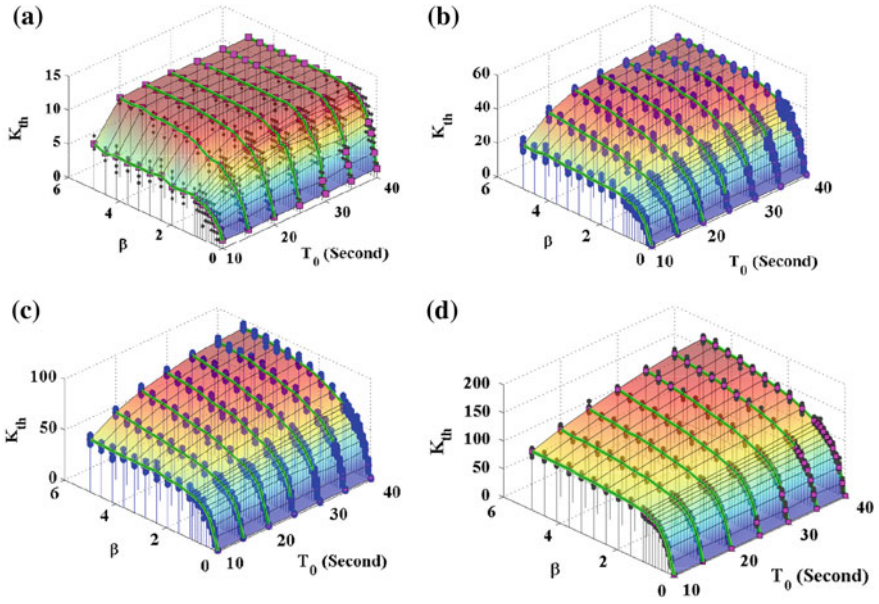
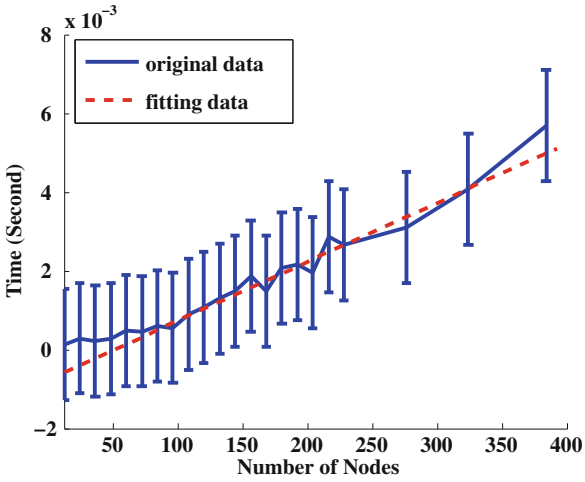


Fig. 6 Comparison of the threshold of admitted user number. The black points are the calculated K_{th} of 100 runs on each configuration, magenta squares are average values from each 100 runs, *green curves* are average values of K_{th} under different T_0, and the colored surface is a regression over all the average values. **a** Comparison of the threshold of admitted number of clients when there are 12 clients in total, **b** Comparison of the threshold of admitted number of clients when there are 48 clients in total, **c** Comparison of the threshold of admitted number of clients when there are 96 clients in total, **d** Comparison of the threshold of admitted number of clients when there are 192 clients in total

5.2 Time Cost of the Scheduling Scheme

The time cost of the scheduling scheme to get the optimal rank, including admission control and request ranking, is directly proportional to the number of nodes as shown in Fig. 7. The time cost is justified when the number of nodes is in the range of [12, 230] with an increment of 12 nodes for each test. In addition, it keeps in tens of milliseconds when the numbers of nodes are 276, 324, and 384. A fitted line is showed in the red dash line. Figure 7 in this document shows the fitted line has a gradient 1.4951×10^5, which means the cost time increase with the number of robot clients.

Fig. 7 Time cost



6 Experiment

This section introduces the experiment design, experimental results and performance discussion.

6.1 Robot Setup

The hardware system is composed of two major categories of robots. The leading robot is set up mainly to work as the database feeder while the others act as consumers of the feeded data.

- Well-equipped leading robot: the leading robot is shown in Fig. 8a. It equips with several sensors like a rotating laser scanner (for 3D point-cloud), an omni-camera (Ladybug™) and an Inertial Measure Unit (IMU) with a GPS module. It can provide an online database with sufficient mapping and localization hints.
- Relatively poorly-equipped follower robot: the follower robot "Epuck" is shown in Fig. 8b. It equips with a Firefly™ camera and a WiFi module. It can request various types of sensor data, for example, the camera can capture 2D bar-codes on the wall in the target environment, then the WiFi module can send it to the host to request the location or the regional map around it.

6.2 Experiment Design

A typical co-localization scenario is shown as Fig. 9. We put up several markers in the environment, which pose can be estimated. The well-equipped robot built a full

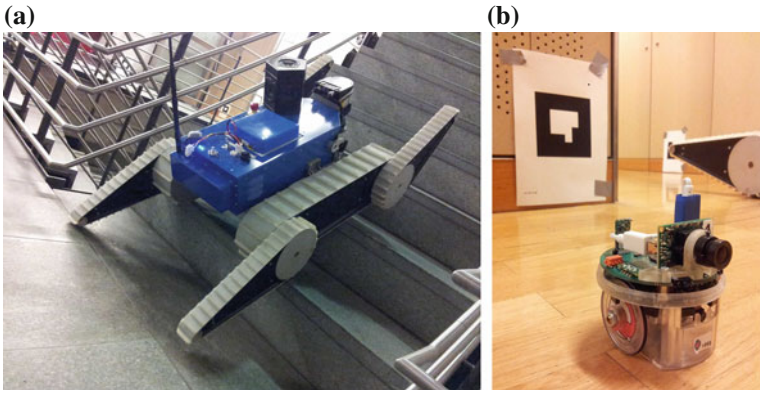


Fig. 8 Robots instances in a typical cloud robotic system, **a** Leading robot: NIFTi, **b** Follower robot: Epuck

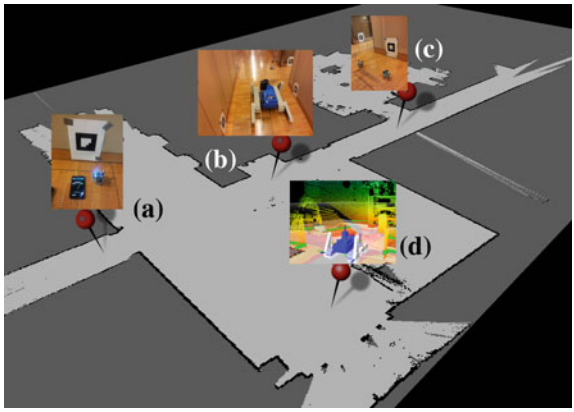


Fig. 9 A map with 3D point cloud of a typical indoor environment for multi-robot co-localization

3D map with marker location registered on it as shown in Fig. 9d. All information on the map were stored in a data center and can be subscribed by follower robots according to the response rank. In the aspect of poorly-equipped robots, they can use their camera to take pictures of AR markers as depicted in Fig. 9a, b. For the specific task, a structure for data flow and resource management is described in Fig. 10. Many services are provided in the cloud such as object detection, node initialization, besides localization as extensions. Robot client can retrieve these services though the structure we introduced in Sect. 3.3.

2D Barcodes Augmented Reality (AR), as shown in Fig. 9, is utilized as 2D barcodes for localization. It provides a package named ARToolKit, which uses computer vision algorithms to calculate the real camera position and orientation relative to physical markers in real time.

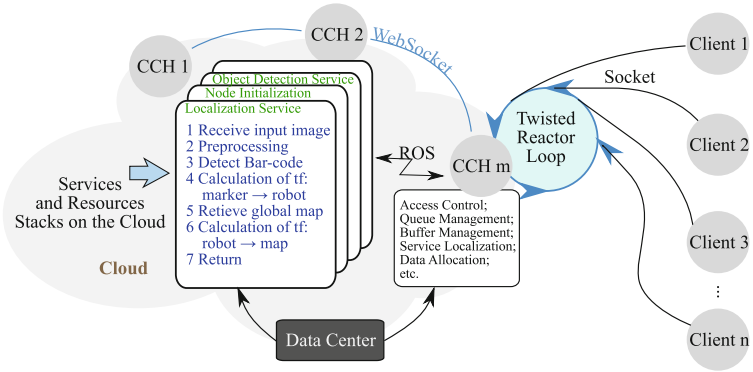


Fig. 10 Data flow and resource management of the proposed system

Marker Pose Registration and Retrieval All this location registration is implemented by ROS since ARToolKit is a package belonged to it. Not all AR markers can be accurately recognized and stable registered, we set a threshold to classify them. Only when the confidence is higher than the preset threshold, the location can be recorded in the database.

Pose Estimation At the beginning, the leading robot builds a 3D map with both images and registered local maps. Then the position of each AR marker is registered on the map by subscribe the related ROS topic, such as transformations. The relative poses are calculated by ARToolKit module.[1] At last, the pose of marker i in the map can be obtained by

$$T^{/map}_{/marker_i} = \underbrace{T^{/map}_{/leading\_robot}}_{\text{SLAM}} \cdot \underbrace{T^{/leading\_robot}_{/camera\_leading\_robot}}_{\text{Sensor Calibration}} \cdot \underbrace{T^{/camera\_leading\_robot}_{/marker_i}}_{\text{ARToolKit}}, \quad (17)$$

where T^a_b is the transformation matrix from frame b to a, namely target frame b in base frame a. Similarly, the pose retrieval for each follower robot j regarding marker i is formulated as

$$T^{/map}_{/robot_j} = \underbrace{T^{/map}_{/marker_i}}_{\text{Data Retreival}} \cdot \underbrace{(T^{/camera\_robot_j}_{/marker_i})^{-1}}_{\text{ARToolKit}} \cdot \underbrace{T^{/camera\_robot_j}_{/robot_j}}_{\text{Sensor Calibration}}. \quad (18)$$

We could see that the data retrieval efficiency will determine the efficiency of the whole co-localization system since it provides a required link. This problem is non-trivial when the system scale is large. Due to the limited bandwidth constraints and constraints of computational ability, the response of such information needs to be negotiated within robot clients and to be managed by the CCH.

[1] www.rog.org/wiki/artoolkit.

Table 1 Matched marker number vs confidence threshold under different WiFi strength

Number of markers	Threshold of confidence	Averaged ratio of matched markers	
		−40 ∼ −50 dB (%)	−50 ∼ −60 dB (%)
30	0.5	84.45	71.24
	0.9	73.28	60.15
100	0.5	80.97	69.22
	0.9	62.64	55.34

Table 2 Measured accuracy results of X-Y-Z offset error

Error marker	Z-offset	X-offset	Y-offset
Standard derivation	32.25884	10.97756	5.197521
Average (mm)	22.87656	–2.36570	0.480358

6.3 Qualitative Results on Localization Behavior

In the online co-localization scenario, all robot clients can be localized in firm real-time by request from the CCH. In this work, localization accuracy is mainly affected by matched marker ratio and marker location registered in the data center. Therefore, as shown in Table 1 we compare the matched marker number with confidence threshold which is pre-set in the `ar_multi.cpp`. For deriving better results, the WiFi strength is recorded when follower robots are requiring location. Moreover, the localization accuracy of ARToolKit is tested in [4], the localization accuracy is evaluated and shown in Table 2. The robot positions are well localized.

6.4 Quantitative Results on Resource Allocation

At first, we compare the *ToR* of continuous requests from 3 robot clients. Figure 11 demonstrates that the *ToR* of requests from all robots in a period. Comparing with the case without mechanism depicted in Fig. 11a, the proposed mechanism managed to reduce the *ToR* according to the priority setup. With the proposed mechanism depicted in Fig. 11b, robot clients received more retrieval data from the cloud. The priority setup for robot 1, 2 and 3 are 1, 3 and 2 respectively. With higher priority, the corresponding requests got a faster response at most of the time duration.

The typical characteristic of cloud robotic is the large number of robot requests in parallel. We compared the *RoR* performance considering the K_{th} and T_0 in the request tasks of 12 clients, which were data retrievals through the Internet. *RoR* of co-localization task depends on the retrieved transformation data registered in the database which can be easily calculated according to (17) and (18). For each request

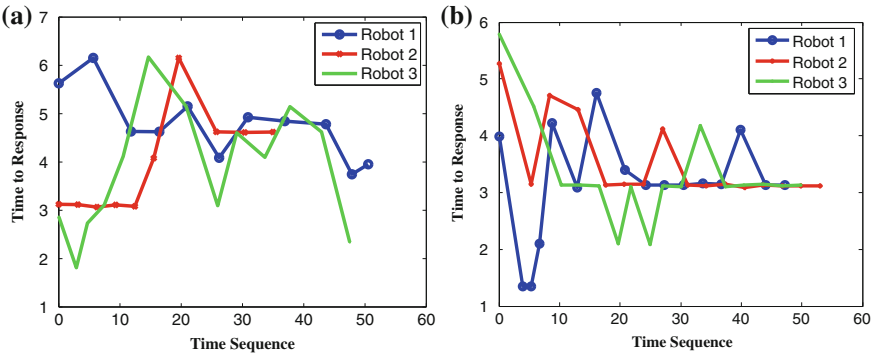


Fig. 11 ToR comparison, **a** Without mechanism, in unit of millisecond, **b** With mechanism, in unit of millisecond

task, it includes 6 independent requests from one client, the package size is $s = 15.625\,\text{Mb} \times 6$, then the ideal transmission time should be $s/(2\,\text{Mb/s}) = 46.875\,\text{s}$. Note that, only partial requests in the buffer queue would get responses from the CCH. It is because the transmission requires time, where the transmission period may be longer than the T_0. In Table 3, we demonstrate the *RoR* among different T_0 and K_{th}. Clients submitted their optimal price of requests, which were determined by their willingness to pay and the desired completing time of the target data retrieval. The results validate that the *RoR* with mechanism performs better, when K_{th} is optimized for each task to respond to their timeout period. In addition, willingness to pay of all clients are uniformly distributed because they have the same requests.

6.5 Discussions

In the online experiment, the proposed system distributed the workload of sensing, localization, computation and communication among a group of robot agents. The regarding characteristics are discussed as follows.

Optimized Constraints of Resource Allocation It greatly reduced the response time for the localization task by deployment of access control, scheduling and protocol management in the proposed cloud robotic system. The optimization is derived from solving the following constraints.

- Data retrieval constraint: in cloud robotic system, a robot can retrieve information from a dynamically updated data center which is built by various types of robots and sensors, therefore it is a heterogeneous structure that needs standard design and regulation to fit it in applications.

Table 3 RoR comparison between with and without mechanism under different timeout period

Threshold K_{th}	Timeout period T_0 (s)							
	10		20		30		40	
	No Mechanism (%)	Mechanism (%)	No Mechanism (%)	Mechanism (%)	No Mechanism (%)	Mechanism (%)	No Mechanism (%)	Mechanism (%)
6	0	0	0	70.61	0.39	100	54.17	100
8	0	0	0	73.83	1.67	100	68.06	100
10	0	0	0	76.39	2.78	100	69.44	100

- Communication constraint: synchronization of data is hard for distributed system, especially when asynchronous tasks are performed [29]. If multi-sensor data are distributed in the network on each client, information sharing among robots is low-efficient without resource allocation.
- Autonomous negotiation constraint: different from research on target tracking [7], automated identification of targets [33], and automated behavior reasoning [30]. Practical autonomous negotiation for resource allocation in firm real-time systems such as cloud robotics is a multi-dimensional problem, therefore both revenues of resource provider and utility of user are considered in this chapter.

Generalization of the Proposed Framework By using such framework, the major generalization can be derived as follows:

- Extension to cloud robotic system with many poorly-equipped robots for information retrieval and communication. It considers of letting robots access a large amount of computational power on demand. The framework sidesteps drawbacks include high computational cost, high configuration, maintenance and update overheads.
- Extension of a more complex hierarchical topology of the system including various types of robots. Especially the negotiation mechanism can be extended according to complex task and environment that it applies.

7 Conclusion

In this chapter, a novel cloud robotic system architecture is developed based on an asynchronous data flow framework. Then the resource allocation problem is formulated as a Stackelberg game and the corresponding solution is proposed. Moreover, the task-oriented QoS criteria are proposed. Afterward, simulations on parameter investigation and time cost are presented. Experiments of co-localization robotic scenarios are implemented for evaluation. Results are discussed and proved the proposed pricing mechanism optimized the resource allocation problem for cloud robotic systems considering physical robots and tasks.

Even if experiments and simulations in the thesis showed promising results, there are still some further work to be developed and extended. Specifically, both the large amount of resource in the cloud and robot clients are heterogeneous. Therefore, a vital issue in this domain is uncertainty about the resource prices, which greatly affect the cost of request retrieval and allocation. As the proposed Stackelberg game mechanism, the optimal price depends on a willingness payment and the corresponding response time. However, the response time is not a prior, which is determined by the willingness payment and a Lagrange multiplier. Instead of depending on the willingness payment, a schedule concerning the price prediction model could be explored to enhance the performance of resource retrieval. Future development of the auction-based mechanisms includes developing and analyzing of optimization

algorithms that are able to tune the variables in the strategy according to the different information conditions. This will then increase the robustness of this resource allocation strategy to make it suitable for all environments and applications.

Acknowledgments This work is supported by RGC GRF Grant CUHK14205914 awarded to Prof. Max Q.-H. Meng; partially supported the Research Grant Council of Hong Kong SAR Government, China, under project No. 16206014 and No. 16212815; National Natural Science Foundation of China No. 6140021318, awarded to Prof. Ming Liu.

References

1. Aldebaran Robotics: Nao robot, http://aldebaran-robotics.com/
2. An, B., Lesser, V.: Characterizing contract-based multiagent resource allocation in networks. IEEE Trans. Syst. Man Cybern. Part B: Cybern. **40**(3), 575–586 (2010)
3. Arumugam, R., Enti, V., Bingbing, L., Xiaojun, W., Baskaran, K., Kong, F.F., Kumar, A., Meng, K.D., Kit, G.W.: DAvinCi: a cloud computing framework for service robots. In: 2010 IEEE International Conference on Robotics and Automation (ICRA), pp. 3084–3089, May 2010
4. Berard, B., Petrie, C., Smith, N.: Quadrotor UAV interface and localization design. In: A Major Qualify Project of Missile Defense Agency under Air Force Contract F19628-00-C-0002 (1Apr00–31Mar05) (2010)
5. Berhault, M., Huang, H., Keskinocak, P., Koenig, S., Elmaghraby, W., Griffin, P., Kleywegt, A.: Robot exploration with combinatorial auctions. In: Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), vol. 2, pp. 1957–1962, October 2003
6. Bichier, M., Lin, K.J.: Service-oriented computing. Computer **39**(3), 99–101 (2006)
7. Brooks, A., Williams, S.: Tracking people with networks of heterogeneous sensors. In: Proceedings of the Australasian Conference on Robotics and Automation, pp. 1–7. Citeseer (2003)
8. Burkard, R.E., Dell'Amico, M., Martello, S., et al.: Assignment Problems. Revised Reprint, SIAM (2009)
9. Chang, M., He, J., Castro-Leon, E.: Service-orientation in the computing infrastructure. In: 2nd IEEE International Symposium on Service-Oriented System Engineering, Shanghai, China, pp. 27–33, October 2006
10. Chen, Y., Du, Z., García-Acosta, M.: Robot as a service in cloud computing. In: 2010 5th IEEE International Symposium on Service Oriented System Engineering, pp. 151–158, June 2010
11. Cheng, M.Y., Tran, D.H., Wu, Y.W.: Using a fuzzy clustering chaotic-based differential evolution with serial method to solve resource-constrained project scheduling problems. Autom. Constr. **37**(0), 88–97 (2014), http://www.sciencedirect.com/science/article/pii/S0926580513001593
12. Choi, H.L., Brunet, L., How, J.P.: Consensus-based decentralized auctions for robust task allocation. IEEE Trans. Robot. **25**(4), 912–926 (2009)
13. Clark, J.: Inside "indigo"—infrastructure for web services and connected applications. Microsoft Press (Apr 2005)
14. Colas, F., Mahesh, S., Pomerleau, F., Liu, M., Siegwart, R.: 3d path planning and execution for search and rescue ground robots. In: 2013 IEEE/RSJ InternationalConference on Intelligent Robots and Systems (IROS), pp. 722–727. IEEE (2013)
15. Corporation, M.: Microsoft robotics developer studio, http://www.microsoft.com/Robotics
16. Darmann, A., Pferschy, U., Schauer, J.: Resource allocation with time intervals. Theor. Comput. Sci. **411**(49), 4217–4234 (2010). http://www.sciencedirect.com/science/article/pii/S0304397510004639
17. Dasarathy, B.: Decision Fusion, vol. 1994. IEEE Computer Society Press (1994)

18. Date, C., Darwen, H.: A Guide to the SQL Standard, vol. 3. Addison-Wesley Reading (1987)
19. Dias, M.B., Zlot, R., Kalra, N., Stentz, A.: Market-based multirobot coordination: a survey and analysis. Proc. IEEE **94**(7), 1257–1270 (2006)
20. Heinemann, F.C.: Web Programming with the Sap Web Applications Server. SAP Press (2003)
21. Gerkey, B.P., Mataric, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. Int. J. Robot. Res. **23**(9), 939–954 (2004). http://dx.doi.org/10.1177/0278364904045564
22. Gostai Coop.: Gostai, http://www.gostai.com/
23. Hall, D., Llinas, J.: An introduction to multisensor data fusion. Proc. IEEE **85**(1), 6–23 (1997)
24. Higuera, J., Gamboa, C., Dudek, G.: Fair subdivision of multi-robot tasks. In: In: Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA 2013), pp. 2999-3004 (2013)
25. Hu, G., Tay, W.P., Wen, Y.: Cloud robotics: architecture, challenges and applications. Netw. IEEE **26**(3), 21–28 (2012)
26. Hunziker, D., Gajamohan, M., Waibel, M., D'Andrea, R.: Rapyuta: the RoboEarth cloud engine. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 438–444, May 2013
27. Intel Coop.: Service-oriented enterprise, the technology path to business transformation. (2 Oct 2005). http://www.intel.com/business/bss/technologies/soe/
28. Jalaparti, V., Nguyen, G., Gupta, I., Caesar, M.: Cloud resource allocation games. Technical report, Department of Computer Science (2010)
29. Kaempchen, N., Dietmayer, K.: Data synchronization strategies for multi-sensor fusion. In: Proceedings of the IEEE Conference on Intelligent Transportation Systems. Citeseer (2003)
30. Kam, M., Zhu, X., Kalata, P.: Sensor fusion for mobile robot navigation. Proc. IEEE **85**(1), 108–119 (1997)
31. Kehoe, B., Matsukawa, A., Candido, S., Kuffner, J., Goldberg, K.: Cloud-based robot grasping with the Google object recognition engine. In: IEEE International Conference on Robotics and Automation (ICRA) (2013)
32. Kehoe, B., Warrier, D., Patil, S., Goldberg, K.: Cloud-based grasp analysis and planning for toleranced parts using parallelized Monte Carlo sampling. IEEE Trans. Autom. Sci. Eng. **12**(2) (2015)
33. Klimentjew, D., Hendrich, N., Zhang, J.: Multi sensor fusion of camera and 3d laser range finder for object recognition. In: 2010 IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pp. 236–241. IEEE (2010)
34. Lagoudakis, M.G., Markakis, E., Kempe, D., Keskinocak, P.: Auction-based multi-robot routing. In: Proceedings of the International Conference on Robotics: Science and Systems (ROBOTICS), pp. 343–350 (2005)
35. Lemaire, T., Alami, R., Lacroix, S.: A distributed tasks allocation scheme in multi-UAV context. In: Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA'04, vol. 4, pp. 3622–3627. IEEE (2004)
36. Lin, L., Zheng, Z.: Combinatorial bids based multi-robot task allocation method. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005), pp. 1145–1150. IEEE (2005)
37. Lingzhi, L., Nilanjan, C., Sycara, K.: Distributed algorithm design for multi-robot task assignment with deadlines for tasks. In: IEEE International Conference on Robotics and Automation, (ICRA 2013), pp. 2992–2998 (2013)
38. Liu, M., Pradalier, C., Pomerleau, F., Siegwart, R.: Scale-only visual homing from an omnidirectional camera. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 3944–3949. IEEE (2012)
39. Liu, M., Colas, F., Oth, L., Siegwart, R.: Incremental topological segmentation for semi-structured environments using discretized GVG. Auton. Robot. **38**(2), 143–160 (2014)
40. Liu, M., Siegwart, R.: DP-FACT: towards topological mapping and scene recognition with color for omnidirectional camera. In: 2012 IEEE International Conferenceon Robotics and Automation (ICRA), pp. 3503–3508. IEEE (2012)

41. Liu, M., Pradalier, C., Siegwart, R.: Visual homing from scale with an uncalibrated omnidirectional camera. IEEE Trans. Robot. **29**(6), 1353–1365 (2013)
42. Liu, M., Siegwart, R.: Information theory based validation for point-cloud segmentation aided by tensor voting. In: International Conference on Information and Automation (ICIA). IEEE (2013)
43. Liu, M., Siegwart, R.: Navigation on point-clouda—Riemannian metric approach. In: 2014 IEEE International Conference on Robotics and Automation(ICRA), pp. 4088–4093. IEEE (2014)
44. de Lope, J., Maravall, D., Quiñonez, Y.: Response threshold models and stochastic learning automata for self-coordination of heterogeneous multi-task distribution in multi-robot systems. Robot. Auton. Syst. **61**(7), 714–720 (2013)
45. Matti, P.: Nash and Stackelberg solutions in a differential game model of capitalism. J. Econ. Dyn. Control **6**(0), 173–186 (1983), 0165-1889. doi:10.1016/0165-1889(83)90048-9
46. McMurtry, C., Mercuri, M., Watling, N.: Microsoft Windows Communication Foundation: Hands-on. Sams Press (May25 2006)
47. Moshe Zadka, G.L.: The twisted network framework. http://twistedmatrix.com/user/glyph/ipc10/paper.html (2010)
48. Mosteo, A.R., Montano, L.: A survey of multi-robot task allocation. Technical report, Instituto de Investigacin en Ingenierła de Aragn (I3A) (2010)
49. Mudholkar, G.S., Srivastava, D.K., Kollia, G.D.: A generalization of the weibull distribution with application to the analysis of survival data. J. Am. Stat. Assoc. **91**(436), 1575–1583 (1996)
50. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: Eucalyptus: a technical report on an elastic utility computing architecture linking your programs to useful systems (2008)
51. OpenNebula Project: Opennebula.org—the open source toolkit for cloud computing, http://www.opennubula.org/
52. Piaggio, M., Zaccaria, R.: Distributing a robotic system on a network: the ETHNOS approach. Adv. Robot. **11**(8), 743–758 (1998)
53. Rai, A., Bhagwan, R., Guha, S.: Generalized resource allocation for the cloud. In: Proceedings of the 3rd Symposium on Cloud Computing (SOCC). San Jose, CA, October 2012
54. Riazuelo, L., Civera, J., Montiel, J.: C^2 TAM: a cloud framework for cooperative tracking and mapping. Robot. Auton. Syst. **62**(4), 401–413 (2014)
55. Rimal, B.P., Choi, E., Lumb, I.: A taxonomy and survey of cloud computing systems. In: Fifth International Joint Conference on INC, IMS and IDC, 2009. NCM'09. pp. 44–51. IEEE (2009)
56. Rodriguez, M., Buyya, R.: Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. IEEE Trans. Cloud Comput. **2**(2), 222–235 (2014)
57. Steven, C.: Robots, incorporated. IEEE Spectrum (August 2007). http://www.spectrum.ieee.org/aug07/5391
58. Sempolinski, P., Thain, D.: A comparison and critique of eucalyptus, opennebula and nimbus. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), 30 2010-December 3 2010, pp. 417 –426
59. Sheth, A., Ranabahu, A.: Semantic modeling for cloud computing, part 1. Internet Comput. IEEE **14**(3), 81–83 (2010)
60. Sung, C., Ayanian, N., Rus, D.: Improving the performance of multi-robot systems by task switching. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 2999–3006. IEEE (2013)
61. Tan, M., Wang, L., Tardioli, D., Liu, M.: A resource allocation strategy in a robotic ad-hoc network. In: 2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 122–127, May 2014
62. Thiruvady, D., Ernst, A., Singh, G.: Parallel ant colony optimization for resource constrained job scheduling. Ann. Oper. Res. 1–18 (2014). http://dx.doi.org/10.1007/s10479-014-1577-7
63. Tsai, W., Huang, Q., Sun, X.: A collaborative service-oriented simulation framework with microsoft robotic studio. In: 41st Annual. Simulation Symposium, ANSS 2008, pp. 263–270, April 2008

64. Tsai, W., Sun, X., Huang, Q., Karatza, H.: An ontology-based collaborative service-oriented simulation framework with microsoft robotics studio. Simul. Model. Pract. Theory **16**(9), 1392–1414 (2008)
65. Viguria, A., Maza, I., Ollero, A.: Set: an algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets. In: 2007 IEEE International Conference on Robotics and Automation, pp. 3339–3344. IEEE (2007)
66. Viguria, A., Maza, I., Ollero, A.: S+T: an algorithm for distributed multirobot task allocation based on services for improving robot cooperation. In: IEEE International Conference on Robotics and Automation, ICRA 2008, pp. 3163–3168 (2008)
67. Waibel, M., Beetz, M., Civera, J., D'Andrea, R., Elfring, J., Galvez-Lopez, D., Haussermann, K., Janssen, R., Montiel, J., Perzylo, A., Schiessle, B., Tenorth, M., Zweigle, O., van de Molengraft, R.: RoboEarth. IEEE Robot. Autom. Mag. **18**(2), 69–82 (2011)
68. Wang, L., Liu, M., Meng, M.Q.H.: Towards cloud robotic system: a case study of online co-localization for fair resource competence. In: IEEE International Conference on Robotics and Biomimetics (ROBIO 2012), pp. 2132–2137, December 2012
69. Wang, L., Liu, M., Meng, M.Q.H.: Real-time multi-sensor data retrieval for cloud robotic systems. IEEE Trans. Autom. Sci. Eng. **12**(2) (2015)
70. Wang, L., Liu, M., Meng, M.Q.H., Siegwart, R.: Towards real-time multi-sensor information retrieval in cloud robotic system. In: 2012 IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pp. 21–26 (September 2012)
71. Wang, L., Liu, M., Meng, M.H.: An auction-based resource allocation strategy for joint-surveillance using networked multi-robot systems. In: 2013 IEEE International Conference on Information and Automation (ICIA), pp. 424–429, (August 2013)
72. Wang, L., Liu, M., Meng, M.H.: Hierarchical auction-based mechanism for real-time resource retrieval in cloud mobile robotic system. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), June 2014
73. Wang, L., Meng, M.H.: A game theoretical bandwidth allocation mechanism for cloud robotics. In: 2012 10th World Congress on Intelligent Control and Automation (WCICA), pp. 3828–3833, July 2012
74. Zhang, Y., Parker, L.E.: Multi-robot task scheduling. In: 2013 IEEE International Conference on Robotics and Automation (ICRA), pp. 2992–2998. IEEE (2013)