Real-Time Multisensor Data Retrieval for Cloud Robotic Systems

Lujia Wang, *Student Member, IEEE*, Ming Liu, *Member, IEEE*, and Max Q.-H. Meng, *Fellow, IEEE*

***Abstract*—Cloud technology elevates the potential of robotics with which robots possessing various capabilities and resources may share data and combine new skills through cooperation. With multiple robots, a cloud robotic system enables intensive and complicated tasks to be carried out in an optimal and cooperative manner. Multisensor data retrieval (MSDR) is one of the key fundamental tasks to share the resources. Having attracted wide attention, MSDR is facing severe technical challenges. For example, MSDR is particularly difficult when cloud cluster hosts accommodate unpredictable data requests triggered by multiple robots operating in parallel. In these cases, near real-time responses are essential while addressing the problem of the synchronization of multisensor data simultaneously. In this paper, we present a framework targeting near real-time MSDR, which grants asynchronous access to the cloud from the robots. We propose a market-based management strategy for efficient data retrieval. It is validated by assessing several quality-of-service (QoS) criteria, with emphasis on facilitating data retrieval in near real-time. Experimental results indicate that the MSDR framework is able to achieve excellent performance under the proposed management strategy in typical cloud robotic scenarios.**

***Note to Practitioners*—This paper was motivated by the problem of sharing resources in cloud robotic systems efficiently for accomplishing real-time tasks. Existing approaches to cloud robotics bear very strict assumptions that the resources are unconstrained and ubiquitous. However, there are technical challenges for multirobot systems to access the cloud and retrieve resources in near real-time. This paper presents a general framework for setting up cloud robotic system with a novel resource management strategy. We mathematically formulate the problem of multisensor data retrieval through the cloud as a Stackelberg game, and propose an optimal solution with proof. We then define the QoS criteria for evaluation considering the constraints of robotic tasks. In the experimental scenarios, our management mechanism significantly improves the performance for multisensor data retrieval in the evaluation of QoS, CPU load, and bandwidth usage.**

***Index Terms*—Cloud robotic system, multisensor fusion, real-time data retrieval.**

Manuscript received October 27, 2014; revised February 24, 2015; accepted February 25, 2015. Date of publication March 13, 2015; date of current version April 03, 2015 This paper was recommended for publication by Associate Editor W. Shen and Editor S. Sarma upon evaluation of the reviewers' comments. The work of M. Q.-H. Meng was supported by RGC GRF #CUHK 14205914 and 415611. This work was supported in part by the HKUST Project IGN13EG03 and RGC #HKUST 16206014, and in part by the National Natural Science Foundation of China under Grant 6140021318.

L. Wang and M. Q.-H Meng are with the Department of Electronic Engineering, Chinese University of Hong Kong, Kowloon, Hong Kong (e-mail: ljwang@ee.cuhk.edu.hk; qhmeng@ee.cuhk.edu.hk).

M. Liu is with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong (e-mail: eelium@ust.hk).

Digital Object Identifier 10.1109/TASE.2015.2408634

I. Introduction

SERVICE robots have become an integral part of our life, and the provided services are getting more and 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 [1], range-finder-based navigation [2], [3], path planning [4], recognition [5], and scene analysis [6], [7]. However, it is infeasible to develop a universal robot that could cover all possible services due to the limitation of cost, reliability, power consumption, payload, sensory and kinematic constraints, among many others. Instead, robots can be relieved from hardware limitations while benefiting from vastly available resources and centralized high computing capability provided by the cloud platform [8]. Therefore, it is reasonable to combine multiple robots of limited capabilities to generate, access and process vast amount of data in a distributed infrastructure facilitated by the cloud infrastructure. The cooperation of multiple robots with various capabilities would provide augmented capabilities and services that are impossible for any single robot. The aforementioned multirobot systems are thus termed as "cloud robotics" [9]. Considering the two-tier architecture proposed in [10], we present a novel framework of a cloud robotic system, as illustrated in Fig. 1. It consists of robots with ubiquitous networks and a cloud-computing infrastructure that connects the robots, sensors, portable devices and most importantly a data-center. By adopting a proxy model, all data can be retrieved from the cloud and managed by the proxy so that the requirements on hardware for each robot can be minimized.

The major contributions of our work are as follows.

- A Stackelberg game-based [11] retrieval management mechanism is proposed with consideration of the interaction among robot clients. We theoretically analyze its optimization and implement its 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 quality-of-service (QoS) criteria are proposed as the primary assessment. The QoS's are defined regarding the fact that sophisticated collaborative robotic tasks are usually time sensitive. *CPU load* and *bandwidth usage* are compared in different scenarios.

In this study, we carried out real-time experiments in typical indoor environments, where several physical clients perform data retrieval. The retrieved data includes multitype data, e.g., on board sensor data, regional maps and images.

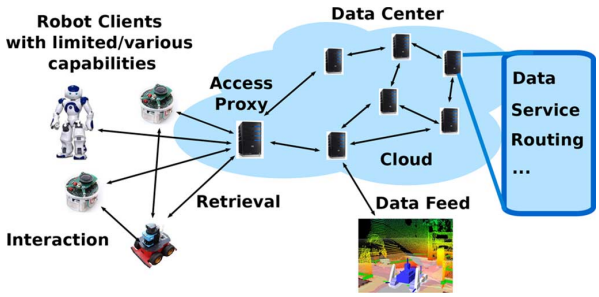


Fig. 1. A novel architecture of cloud robotic system.

The rest of this paper is organized as follows. In Section II, we discuss the related work of cloud robotics. Section III presents our design of a cloud robotic system. In order to solve the inherent conflicts of MSDR, we introduce the theoretical modeling and solution in Section IV. To validate the proposed mechanism, we define two criteria of QoS in Section V. The experimental setup and result analysis are given in Section VI. Finally, Section VII presents the conclusion.

II. Related Work

Multisensor data retrieval (MSDR) is an essential element for cloud robotic systems. Typically, resource retrieval via robot addressing becomes quite low-efficient, if multiple sensor data need to be distributed simultaneously [12]. This is because there exist inherent conflicts: each robot client tries to complete its retrieval in the least possible time with the least possible cost. Conversely, the proxy tries to maximize the resource utilization of the cloud. Therefore, the MSDR is an important issue affecting the performance of cloud robotic systems. Because the "cloud robotics" is a relatively new field, we first briefly review the state-of-the-art works with respect to the architectures of the cloud robotics, current approaches and resource management of multiagents, among others.

A. Architectures of Cloud Robotics

The architecture of cloud robotics as shown in Fig. 1 is comprised of two levels: a network structure among robots called robot-to-robot (R2R) and a cloud infrastructure including connection interface from robot to cloud called robot-to-cloud (R2C). On the R2R level, it is a wireless network of a group of robots, such as Wireless Local Area Networks (WLANs), Mobile Ad-hoc NETworks (MANETs), among others. On the R2C level, the infrastructure of cloud, which is characterized as "Software as a Service" (SaaS), "Platform as a Service" (PaaS), "Infrastructure as a Service" (IaaS), "Hardware as a Service" (HaaS) [13], and "Robot as a Service (RaaS)" [14], provides a pool of shared sensor data, computation and storage resources that could be allocated by the proxy.

As discussed in [15], because of the heterogeneous services and data, cloud is usually addressed by a common middleware to achieve interoperability. Current works have been limited to e-commerce and enterprise computing systems so far, such as Eucalypus of Amazon EC2 [16], OpenNebula [17], and Nimbus [18]. Applying the middle-ware in physical robotic systems is one of the most vital research topics.

B. Current Cloud Robotic Approaches

Although the concept of networked robots or robots as web services can be dated back to the 1990s [19], cloud robotics is now in a better condition of both the network and the robot to approach an innovated outtake:

- "DAvinCi" [20] was a cloud computing infrastructure to generate models of environments, which allowed robots to perform simultaneous localization and mapping (SLAM) by cloud.
- The Gostainet [21] was an infrastructure of cloud robotics for speech recognition on humanoid robot NAO [22].
- A world wide web for robots called RoboEarth [23] was built for robots to autonomously share descriptions of environments and object models [24]. It was based on PaaS [25] and a cloud engine.
- The "cloud-based robot grasping" [26] used the Google Object Recognition Engine to recognize and grasp common household objects.
- Carlos *et al.* presented a software framework to facilitate cloud-hosted robot simulations that addressed the challenge of real-time task-oriented robot competition [27].
- Gouveia *et al.* proposed two distributed architecture for the SLAM problem, and analyzed their efficiency, precision, and accuracy [28].
- A robot cloud center [29] was designed to follow the general cloud computing paradigm, while robots were provided as a service addressing the limitations in capacity and versatility of robotic applications.

Besides, other applications were proposed as well. For instance, knowledge change among small batch assembly robots [30], robot navigation assistance [31], and so on. The aforementioned research took advantage of a wide range of online resources. However, there are still drawbacks and challenges to be further addressed for cloud robotic systems. Among potential benefits of cloud robotics, to provide seamless and low-cost service robots is one of the most meaningful topics at the current stage. In order to simplify the problem, most cloud-based robotic systems set a very strict assumption, i.e., *the resource in cloud is unconstrained* [32]. As a matter of fact, most of the resources in the cloud robotics system are indeed constrained [33]. For instance, network bandwidth for transmitting image data, CPU occupancy for parallel computation, as well as the number of available hosts (proxy) in the cloud are all limited. Therefore, how do we design a module to maximize the utility of available resources on demand is a difficult problem, especially when many robots request the same resource or service in an asynchronous manner.

C. Resource Management Approaches

Resource management problems are NP-hard in general, which exist in computation systems, network communications, transportation system, etc. For traditional resource allocation and task scheduling, researchers proposed different optimization techniques such as colony optimization, genetic algorithm, fuzzy logic, and market-based approaches. These optimizations minimize the execution time of tasks and cost, or maximize the system utilization and throughput.

- Ant colony optimization (ACO) algorithm is used to make efficient resource assignments for computational jobs

being processed. Thiruvady *et al.* proposed a parallel ACO algorithm to efficiently solve the resource constrained scheduling problem for mining supply chains [34].

- Genetic algorithm is used to solve the optimization problem based on a natural selection process that mimics biological evolution. Rodriguez *et al.* proposed a particle swarm optimization algorithm for resource providing and scheduling on Infrastructure as a Service (IaaS) cloud to minimize the overall workflow execution cost [35].
- Fuzzy logic is a many-valued logic that lends itself to make decisions in various systems. Cheng *et al.* proposed a optimization algorithm Fuzzy Clustering Chaotic-based Differential Evolution (FCDE) in order to solve resource constrained project scheduling problem [36].
- Market-based approaches for resource management [37]–[39] and power control and scheduling [40] are characterized by capturing complex interactions among autonomous agents and system, which suits the resource allocation problem of cloud robotics most.

However, most of them have assumptions that are not suitable for practical robotic tasks. For instance, the boundless communication and computation resources are inappropriate. The limited bandwidth resource should be considered in the real-life scenario as presented in [41]–[43].

Autonomous negotiation among multiple robots has become a crucial problem in a cloud robotics system when the robot clients query resources simultaneously. The reasons are twofold: multiagent systems are typically complex and distributed; agents are combined together as an overarching framework for integrated tasks [44]. For robotic systems, there are several approaches introduced as follows.

- Centralized approaches [45], [46]: this kind of methods have the advantage of using global knowledge to manage all the available resources optimally, while the disadvantage is that time and complexity cost are usually high.
- Distributed approaches [47], [48]: these methods are generally low cost, since they only use local information, but they cannot always achieve the global optimum.
- Combinatorial approaches [49], [50]: this kind of approaches allocate resources that are combinations of different tasks, rather than a single task in complex systems. Their computational results indicate that combinatorial auctions generally lead to superior team-level performance than single-task auctions.

In general, the above works are based on theoretical analysis and simulation. Very limited number of real-time robotic scenarios have been reported in physical systems. Our goal is to bridge this gap, such that we introduce the proposed system in the next section.

III. System Design

A cloud robotic system distributes workload of sensing, computation, and communications among a group of robot clients. For design of the system, we introduce the functionalities in the system, and data flow in the software platform.

A. Structure Design

The proposed framework of data retrieval is shown in Fig. 2. It is a host-based network framework, which has three main

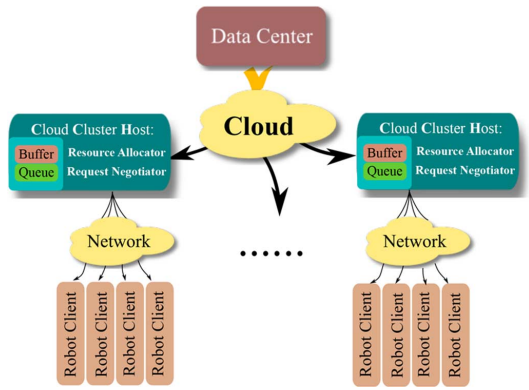
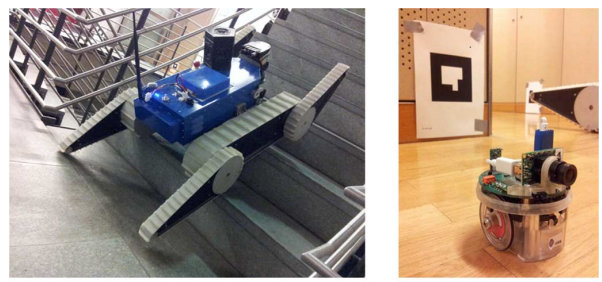


Fig. 2. The data retrieval framework of a cloud robotic system.



(a) (b)

Fig. 3. Robot instances in a typical cloud robotic system. (a) Leading robot: NIFTi. (b) Follower robot: Epuck.

entities involved for supporting the MSDR in a cloud robotic system, namely, the data center (DC), the cloud cluster host (CCH), and the robot clients (RC).

- Data center (DC): It is a relational database built on PostgreSQL that stores various data. All data are maintained and shared by any robot client in the network [33]. At the same time, DC confronts unpredictable simultaneous requests from the robot clients. Therefore, we introduce the next entity.
- Cloud cluster host (CCH): It is a server that manages a large amount of data retrievals. CCH consists of two major modules: requesting negotiator (RN) and resource allocator (RA). RN provides RC with different prices of resources and controls the admission of requests. RA ranks clients in the buffer queue and distributes resources to them in terms of priority derived from the RN.
- Robot client (RC): It is a unit of heterogeneous robots with various sensors in the lowest level of the framework. They can be assigned to certain tasks. Details are introduced in the next sections.

B. Robot Client Setup

The functionality of RC is composed of two major categories of robots: the leading robot mainly to work as the database feeder, while others act as consumers of the fed data.

- Well-equipped leading robot: the leading robot is shown in Fig. 3(a), which is equipped with multiple sensors such as a rotating laser scanner (for 3D point-cloud), an omni-camera (Ladybug™), and an inertial measurement unit (IMU) with a GPS module. It can feed an online database with sufficient mapping and localization data.

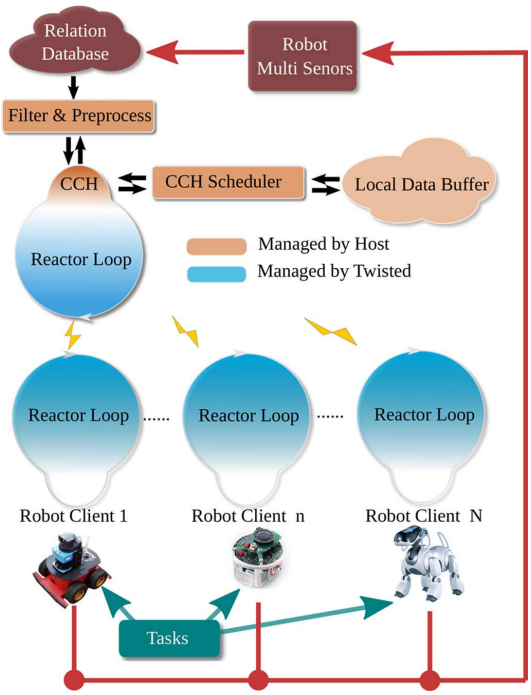


Fig. 4. Dataflow of multidata retrieval and communications in cloud system.

- Relatively poorly equipped follower robot: the follower robot "Epuck," as shown in Fig. 3(b), is equipped only with a Firefly™ camera and a Wireless Fidelit y(WiFi) module. It can request various types of sensor data via WiFi. For example, the camera captures 2D bar-codes on the wall in the target environment, then the WiFi module sends it to CCH to request the location or regional map around the target environment.

C. Dataflow in the Software Platform

The dataflow of multidata retrievals and communications in proposed cloud robotic system (including DC, CCH, and RC) is illustrated in Fig. 4. This system automatically launches a new thread for each client that attempts to connect the network with an approved address and port, with the following functions.

- Database Query.
  This function is launched and managed only by CCH which retrieves data from DC for RC. It utilizes standard SQL [51] syntax to retrieve target data from a dynamically updated DC which is also a relational database. Therefore, DC access would be a bottle-neck in the system. Through the management of CCH, the bottle-neck is alleviated. To this end, we use the following subfunctions to assist the retrieval, namely, Filter and Preprocess, Buffer Management, and Scheduler.
- Filter and Preprocess.
  In the proposed data flow structure, the filter and preprocess blocks stand for general data preprocess. For example, data fusion, feature fusion and decision fusion [52], are the major means to decrease the frequency of database access and to reduce data noise. We do not focus on this problem in this paper.
- Buffer Management.
  This function is launched and managed by CCH where a local buffer is deployed for storage of frequently requested data as depicted in Fig. 4. Because activities of robot clients are usually regular, the same resource would be queried repetitively. Therefore, we build the buffer strategy to help optimize the database access.
- Scheduler.
  Last but not least, the proposed scheduling scheme is launched by CCH that allocates resources for all robot clients' requests on top of asynchronous communication threads. Regarding the software platform, we compare `Twisted`-based socket, `actionlib` package in ROS, and Hadoop MapReduce as follows.
  - `Twisted`-based socket is a framework for deploying asynchronous, event-driven and multithread supported network system which can effectively facilitates the management of asynchronous threads in cloud systems.
  - `actionlib` package only provides tools to create servers that execute long-running goals, but it does not support the message queue management, especially the asynchronous access for multiple tasks in the waiting list. However, cloud robotic systems have the requirements of request queue management.
  - MapReduce includes a large number of disk seeks, by which the bottleneck of disk access significantly slows down the process. However, cloud robotic tasks have a near Hard Real-Time (HRT) requirement when multi-robots simultaneously retrieve data from the CCH.

  Therefore, we preferably choose `Twisted`-based socket [53] as the platform, because it is the user-defined structure that can be flexibly applied to various applications. The asynchronous communication management based on `Twisted` framework is implemented in the CCH to manage all the connections among CCH and robot clients through the `reactor` loop in parallel, as shown in Fig. 4. Please note that `reactor` loop is a fundamental infrastructure of `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 data retrieval is modeled as a Stackelberg game. More details are introduced in the next section.

IV. A Scheduling Mechanism for MSDR

In this section, a MSDR problem is modeled and analyzed to reach fast and reliable responses of the resource retrieval. Regarding the modeled MSDR problem, we propose a Stackelberg game-based mechanism that manage the interaction between robot clients and CCH. Then, we present the process of the resource allocation.

A. The MSDR Problem Formulation

As the number of services and data increases, efficiency of multidata retrieval becomes more challenging. The MSDR optimization problem is a scheduling of resource retrieval and the resources required by those retrievals while taking into consideration both the resource availability and the response time. Regarding game-theoretic studies on the resource allocation problem, we formulate the MSDR problem as a Stackelberg game within our system. CCH and RCs act as the leader and the followers [11] in the game, respectively. The leader maximizes

TABLE I
OVERVIEW OF NOTATIONS IN SECTION IV

$\mathbf{R} = \{R_i\}_{i=1}^{N}$	A set of robot clients
$\mathbf{p} = [p_1 \cdots p_N]^T$	A set of resource price to be retrieved
$\mathbf{t} = [t_1 \cdots t_N]^T$	A set of completion time of resource retrieval
ω_i	Willingness to pay of robot client i
N	Total number of robot clients
n	Admitted number of robot clients
λ	Lagrange multiplier
K_{th}	Threshold of admitted number of robot clients
T_0	Deadline of execution time

its revenue that is the sum of charges from the clients for the use of data retrievals. The followers maximize their utility of data retrievals for each task. We list notations defined in the section in Table I.

Suppose that the resources are allocated from the CCH to a set of N robot clients $\mathbf{R}$. The price set of the resources $\mathbf{p}$ is charged differently among the robots. We use bold symbols to denote vectors in the rest of this paper. We formulate the system model with two problems $\mathcal{P}_1, \mathcal{P}_2$ for robot clients and CCH, respectively.

- For each robot client $R_i \in \mathbf{R}$, the utility surplus function is defined as

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

where ω_i denotes the willingness to pay of robot client i, t_i is the completion time of robot i for resource retrieval. The logarithmic function is a widely used utility function for proportionally fair resource allocation in communication networks (see [54]). This kind of function is selected because it is a concave completion function that can express the quantities of problem interest in closed forms. Specifically, the cost function in this paper is only related to the completion time, which is the main concern of the MSDR problem. Robot clients solve the following maximization problem:

$$\mathcal{P}_1: \quad t_i^* = \underset{t_i \geq 0}{\operatorname{argmax}}\, u_i(t_i, p_i) \tag{2}$$

where $(\cdot)^*$ denotes the optimal value, and (t_i, p_i) is a pair of *strategy profiles* of each robot.

- For CCH, the revenue function is defined as

$$L(\mathbf{t}, \mathbf{p}) = \sum_{i=1}^{n} t_i p_i \tag{3}$$

where n is the number of robot clients that are allocated resources. CCH maximizes its revenue by choosing the optimal price for the constrained resource as

$$\mathcal{P}_2: \quad \mathbf{p}^* = \underset{\substack{\mathbf{t} \geq 0 \\ \mathbf{p} \geq 0}}{\operatorname{argmax}}\, L(\mathbf{t}, \mathbf{p}) \tag{4}$$

where $(\mathbf{t}, \mathbf{p})$ is pair of *strategy profiles* vector of the CCH corresponding to the action of n robot clients, and $\mathbf{t} \geq 0$ and $\mathbf{p} \geq 0$ are in elementwise sense.

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

$$\sum_{i=1}^{n} t_i \leq T_0, \quad n = 0, \ldots, N. \tag{5}$$

Please note that the bandwidth cost in communication is not taken into consideration.

B. The Optimization Solution of MSDR Problem

The optimization problem $\mathcal{P}_2$ of maximization revenue function defined in (4) is not straightforward to solve, because it is a nonconvex optimization problem with a nonconvex objective function, a coupled constraint (5). However, it can be converted into an equivalent convex formulation through the following transformations and thus solved efficiently.

First, for each robot client R_i, the utility surplus function u_i defined in (1) is increasing, strictly concave, and twice continuously differentiable with respect to t_i. Considering the unconstrained optimization problem $\mathcal{P}_1$ of maximization utility surplus function $u_i : \mathbb{R}^n \to \mathbb{R}+$, defined in (2), the *first-order necessary condition* that t_i^* is a local optimum is

$$\frac{\partial u_i(t_i, p_i)}{\partial t_i}|_{t_i = t_i^*} = 0 \tag{6}$$

Therefore, we differentiate the utility surplus function as

$$\begin{aligned} \frac{\partial u_i(t_i, p_i)}{\partial t_i} &= \frac{\partial(\omega_i \cdot \log(1 + t_i) - t_i p_i)}{\partial t_i} \\ &= \frac{\omega_i}{1 + t_i} - p_i \\ &= 0. \end{aligned} \tag{7}$$

The optimal completion time of resources retrieval from robot client i is derived as

$$t_i^* = \frac{\omega_i}{p_i} - 1. \tag{8}$$

Additionally, $t_i \geq 0$, there is no need to set p_i higher than ω_i; the CCH demands zero revenue when $p_i = \omega_i$. This means (8) can be rewritten as

$$p_i = \frac{\omega_i}{1 + t_i^*}. \tag{9}$$

Second, assuming that the optimal admitted number of robot clients is known as $n^* = K_{\text{th}}$, we can convert the problem by plugging (9) into (4), resulting in

$$\mathbf{t}^* = \underset{\substack{\omega_i \geq 0 \\ t_i \geq 0}}{\operatorname{argmax}} \sum_{i=1}^{n} \frac{\omega_i t_i}{1 + t_i}. \tag{10}$$

Remark: The optimum is changed from $\mathbf{p}^*$ to $\mathbf{t}^*$ resulting from transformation of (9), because the previous optimization problem cannot be straightfowardly solved. With the transformation, it can be easily proved that the Jacobian matrix of function $\sum_{i=1}^{n}(\omega_i t_i)/(1 + t_i)$ in (10) is positive-definite. Therefore, the nonconvex optimization problem (4) is converted to a convex problem with a strictly concave function, and its constraint set is convex and compact.

Considering the problem in (10) and the constraints in (5), we define the Lagrange function as

$$\Lambda(t_i, p_i, \lambda) = L(t_i, p_i) + \lambda \left(\sum_{i=1}^{n} t_i - T_0 \right) \tag{11}$$

where λ is the Lagrange multiplier. Let $\nabla_{t_i}\Lambda(t_i, p_i, \lambda) = 0$, we get the optimal completion time

$$t_i^* = \sqrt{\frac{\omega_i}{\lambda}} - 1. \tag{12}$$

Note that the time constraint defined in (5) must hold with equality, because the objective is a strictly increasing function with respect to t_i. Thus, by plugging (12) into (5), we have a boundary condition as

$$\sum_{i=1}^{n}\left(\sqrt{\frac{\omega_i}{\lambda}} - 1\right) = T_0. \tag{13}$$

As derived in (9), the willingness to pay ω_i is proportional to the optimal price p_i^*, which is used to schedule response priority of requests. We assume $\omega_1 \geq \omega_2 \geq \cdots \geq \omega_N$, then the admitted robot clients have higher willingness to pay than those are not admitted, and λ^* must satisfy the condition of (13). A threshold K_{th} of the admitted number of robot clients should satisfy

$$\frac{\omega_{K_{\text{th}}}}{\lambda^*} > 1 \quad \text{and} \quad \frac{\omega_{K_{\text{th}}+1}}{\lambda^*} \leq 1 \tag{14}$$

where K_{th} is used for the admission control, so only K_{th} or less robot clients can retrieve data. Moreover, we have $\lambda^* = ((\sum_{i=1}^{K_{\text{th}}}\sqrt{\omega_i})/(T_0 + K_{\text{th}}))^2$ derived from (13).

The property of above solutions lead to the following **Algorithm 1** to compute λ^* and K_{th}: we start by assuming $K_{\text{th}} = N$ and calculate λ. If the condition of (14) is not satisfied, K_{th} is decreased by one and λ is recalculated until it is satisfied. Because $\omega_1 \leq \lambda_1$ and $\lambda_1 = (1)/(T_0 + 1)$, Algorithm 1 always converges and returns the unique value of K_{th} and λ^*. The complexity is $\mathcal{O}(\mathcal{N})$, which has a linear relationship with the number of robot clients. In addition, the optimal price p_i^* and completion time t_i^* are calculated.

Algorithm 1: The Revenue Maximization Algorithm

Inputs: ω_i, T_0 and N

Outputs: $K_{\text{th}}, \lambda^*, t_i^*$, and p_i^*

1. BEGIN
2. **function** Revenue$(i, \omega_i, T_0, i \in N)$
3. $k \leftarrow N, \lambda(k) \leftarrow (\frac{\sum_{i=1}^{k}\sqrt{\omega_i}}{T_0+k})^2$
4. **while** $\omega_k \leq \lambda(k)$ **do**
5. $k \leftarrow k-1, \lambda(k) \leftarrow (\frac{\sum_{i=1}^{k}\sqrt{\omega_i}}{T_0+k})^2$
6. **end while**
7. $K_{\text{th}} \leftarrow k, \lambda^* \leftarrow \lambda(k)$
8. $t_i^* = \sqrt{\frac{\omega_i}{\lambda^*}} - 1$
9. $p_i^* = \frac{\omega_i}{1+t_i^*}$
10. **return** $K_{\text{th}}, \lambda^*, t_i^*, p_i^*$
11. END

Definition 3.1: Nash Equilibrium (NE): Given the above Stackelberg game, a pair of strategies profile (p_i^*, t_i^*) is an NE for the Stackelberg game if for any player i:

$$\begin{aligned} u_i(t_i^*, p_i) &\geq u_i(t_i, p_i) \\ L(t_i, p_i^*) &\geq L(t_i, p_i). \end{aligned} \tag{15}$$

Then, we have the following Theorem.

Theorem 3.1: Optimal Time Response and Prices for NE Points: With limited bandwidth and K_{th} robot clients are admitted into the network, aforementioned Stackelberg game admits NE strategy profiles that satisfy the conditions in (14). There exists a λ^* when the optimal admitted number of clients K_{th} is achieved, such that each robot client i receives an optimal response time

$$t_i^* = \begin{cases} \sqrt{\frac{\omega_i}{\lambda^*}} - 1, & i = 1, \ldots, K_{\text{th}} \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

with the optimal price

$$p_i^* = \begin{cases} \sqrt{\omega_i \lambda^*}, & i = 1, \ldots, K_{\text{th}} \\ \omega_i, & \text{otherwise.} \end{cases} \tag{17}$$

The value of λ^* and K_{th} can be computed using **Algorithm 1**, for all $i \in N$.

C. Resource Allocation Process

Previous theoretical analysis indicates the proposed a Stackelberg game-based mechanism can optimize the MSDR problem. The basic operation of the mechanism is implemented in the CCH and comprises the following processes.

- **Admission control**: When a resource request is submitted, request negotiator of CCH utilizes the proposed admission control strategy (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 data, and sufficient robot client requests can be fulfilled successfully.
- **Request ranking**: The request negotiator of CCH is also in charge of ranking the admitted requests considering their willingness to pay ω_i and time deadline T_0 as presented in **Algorithm 2**. Having access to the allocation requests of all robot clients, the CCH can keep tracking of current clients, and update the ranking list when a new request is registered.
- **Resource distributing**: Requests with admission and a priority are responded in accordance with the current order in the rank list. In this situation, it optimizes both the utility of each robot client and the revenue of CCH. When new requests from robot clients arrive, the resource allocator would respond the requests with updated rank list.

Algorithm 2: Priority Ranking 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** $i \leq n_{threshold}$
7. update_priority_list(p_i^*)
8. is_lowest_priority(p_i^*)
9. **return** current_priority_list
10. END

Given the above presented scheduling scheme, we define the QoS in the following section to evaluate of the proposed mechanism with applications.

V. Quality-of-Service (QoS) Criteria

QoS, which is generally used to assess performance of a SOA, plays a crucial role in impacting both users' utilization and resource providers' revenue. It advertises performance quality levels of service which are provided by resource providers. At the same time, clients use it to optimally select a 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.

In common cases, bandwidth usage is one of the most important factors to define QoS, because 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. We selectively define the following QoS's as primary criteria to assess the proposed framework.

Definition 4.1: Time of Response (ToR): *ToR* defines the period between sending a request and receiving the corresponding response. It is formulated as follows:

$$ToR = T_{\text{Data\_received}} - T_{\text{Request\_sent}}. \tag{18}$$

ToR in near real-time situation has been considered, because sophisticated collaborating robotic tasks are usually time sensitive. For instance, cooperative semantic mapping or 3D mapping using several robots needs to be completed in real-time, although there exist bottlenecks in data transmission.

Definition 4.2: Reliability of Response (RoR): *RoR* is defined as a success rate of the issued data retrievals. Its value is given in a percentile and calculated as follows:

$$RoR = \frac{\#_{\text{Succeeded\_Requests}}}{\#_{\text{Total\_Requests}}}. \tag{19}$$

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.

In addition, the *CPU load* can indicate the computation complexity, and *bandwidth usage* can vividly demonstrate the effects of resource retrieval on limited bandwidth, which directly affect the value of *ToR*. Based on the above criteria, we implement the experiments and evaluate the proposed strategy in the next section.

VI. Experiment and Evaluation

In this section, we first describe the experiment setup, then we implement a simulation of parameter investigation to instruct the following experiment scenarios. Afterwards, we test the proposed strategies in two scenarios of data retrieval, one is for homogeneous resources of large size, the other is for heterogeneous resources.

A. Experiment Design

When exploring an environment, the map is not known as a prior. A raw database should be built before other clients can query the data when they need. In this work, the proposed system enables several poorly equipped robots without 3D sensors to work in parallel to retrieve data of 3D map, which is built by a well-equipped robot with an elaborated 3D laser scanner. Detailed experimental phases are described as follows.

- Build a relational database including 3D maps and image data of typical indoor environment, as shown in Fig. 5, using a well-equipped robot.
- Each poorly equipped robot sends several requests to the CCH by providing with its pose. Then, the CCH accesses to the database, and matches the target data by sending SQL requests using either the ID or other properties such as the time-stamp and the type of data in the relational database.
- CCH manages all requests with predefined scheduling strategy as introduced in Section IV.
- *ToR*, *RoR*, *CPU load*, and *bandwidth usage* are logged on each robot and CCH to evaluate the experimental results.

The test is carried out in a multithread loop of communications which we introduced in Section III. Robot clients perform as network nodes to request multiple data from the CCH. As shown in Table II, different configurations of CCH and clients are selected to retrieve data. The request data in our case are demonstrated in Table III. The network throughput of each client is limited to 2 Mb/s.

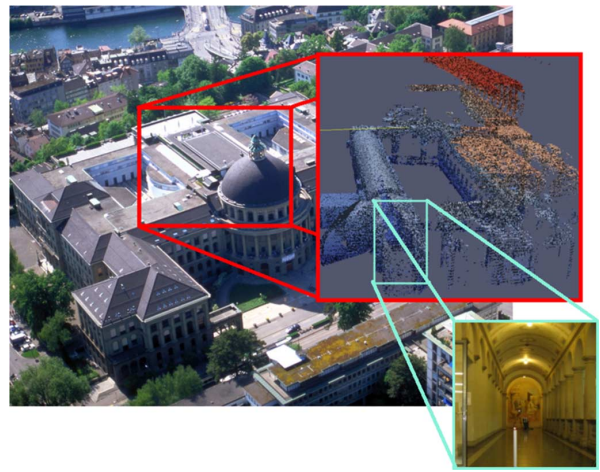


Fig. 5. 3D point-clouds and image instance of a building.

TABLE II
Configuration of CCH and Clients in Experiment

Node	CPU	Memory	Hard Disk
Host	Intel(R) Core(TM) i3 550 processor 3.20GHz× 4	8GB	1TB
Well-equipped Robot Clients	Intel(R) Core(TM) i5-2540 processor 2.6GHz ×2	4GB	300GB
Poorly-equipped Robot Clients	Intel(R) Celeron CPU 877 processor 1.4GHz ×2	1.8GB	120GB

B. Parameter Investigation

In the proposed admission control, the K_{th} is determined by the distribution of willingness to pay from 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 would reduce the K_{th} to fulfill the resource retrieval

TABLE III
CONFIGURATION OF REQUEST DATA

Data type (Message)	ROS topic	Data size (Bytes)
navigation occupancy grid	/map	15745107
wheel odometry	/odom	372
navigation odometry	/pose	2911
transformation	/tf	479

before deadline, namely, admission control. However, there is no restriction on how to choose willingness to pay as a robot client. We choose Weibull distribution because it is a versatile distribution that can represent different kinds of statistical distribution and therefore can take on various characteristics based on the following function:

$$f(x;\alpha,\beta) = \begin{cases} \frac{\beta}{\alpha}\left(\frac{x}{\alpha}\right)^{\beta-1} e^{-(\frac{x}{\alpha})^{\beta}}, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (20)$$

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 that are willing to pay, and the Weibull distribution demonstrates the proportion of the high willingness to pay robot 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 $\sigma = (\alpha)/(\sqrt{2})$ when $\beta = 2$.

In order to indicate the relationship between the willingness to pay and the threshold of admitted number of clients in the proposed admission control, we compare the optimal K_{th} considering different distributions of willingness to pay of 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 was a required time for a certain task.

In the simulation, we tested the admission control proposed in Section IV 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 $k = \{12, 24, 48, 96, 192\}$, respectively. One hundred runs were carried out on each configuration. In Fig. 6, we can see 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 $0 < \beta < 1$. In addition, the variation trends of K_{th} are quite similar when the size of clients is 24, 48, 96, and 192, so we only show $k = 12$ in Fig. 6(a), and $k = 192$ in Fig. 6(b). Moreover, the results also show that the willingness to pay is a key factor for designation of the scheduler since it can affect the QoS. Moreover, the above results are references for the evaluation in the next section.

C. Data Retrieval Results

By differentiating the queried data into homogeneous and heterogeneous, we implemented the following two scenarios to evaluate the proposed strategies.

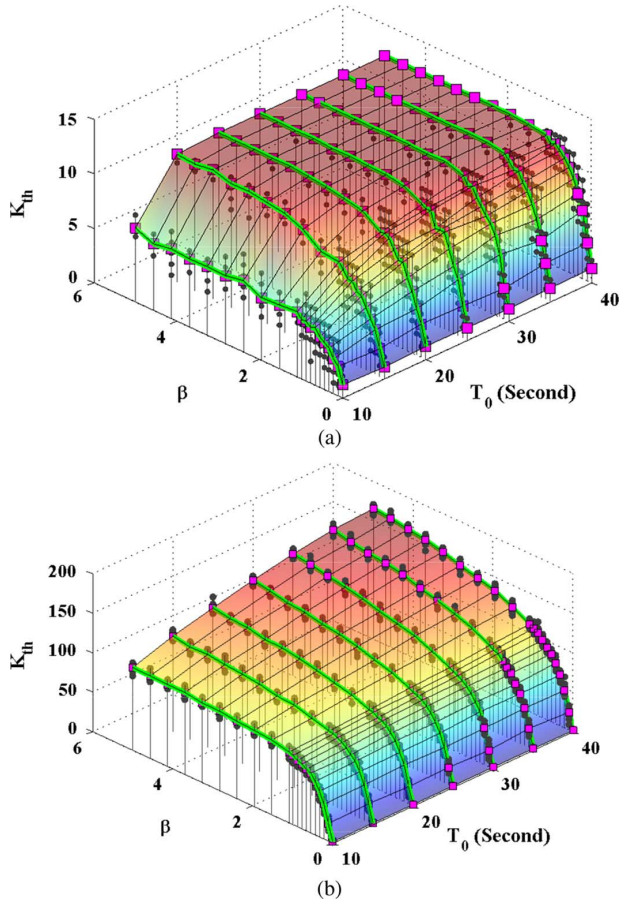


Fig. 6. Comparison of threshold of admitted user number. The black points are the calculated K_{th} of 100 runs on each configuration, magenta squares are average values on each 100 runs, green curves are average values of K_{th} under different T_0, and the colored surface is a regression over all the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

1) Scenario 1—Homogeneous Data Retrieval: In this scenario, 12 robot clients attempted to request the same type of messages, namely, `map`. The transmission of large binary objects `map` can easily overload the network. The aim of the scenario is to justify the efficiency and reliability of data retrieval, therefore the *RoR* in a series of Timeout Period, *CPU load* and *Bandwidth usage* were used to evaluate the proposed scheduler in the CCH.

To help the understanding of the process of the MSDR in the proposed system, we describe the case that multiclients are querying data simultaneously from the CCH in Fig. 7. A time chart of processing 12-parallel requests on CCH is shown in Fig. 7(a), which includes clients connection, database initialization, client querying, request scheduling, and request response. In this case, the peak value of the *CPU load* is almost 50%. The considerably dense load demonstrates that many clients were building connections and querying to CCH. Moreover, we divided the above 12-parallel requests into 4 successive periods of 3-parallel requests, where the maximal *CPU load* is 33%, as shown in Fig. 7(b). These results indicate that scheduling of requests can benefit the *CPU load* on the CCH. In addition, the second and third 4-parallel-requests save around 13% of *CPU*

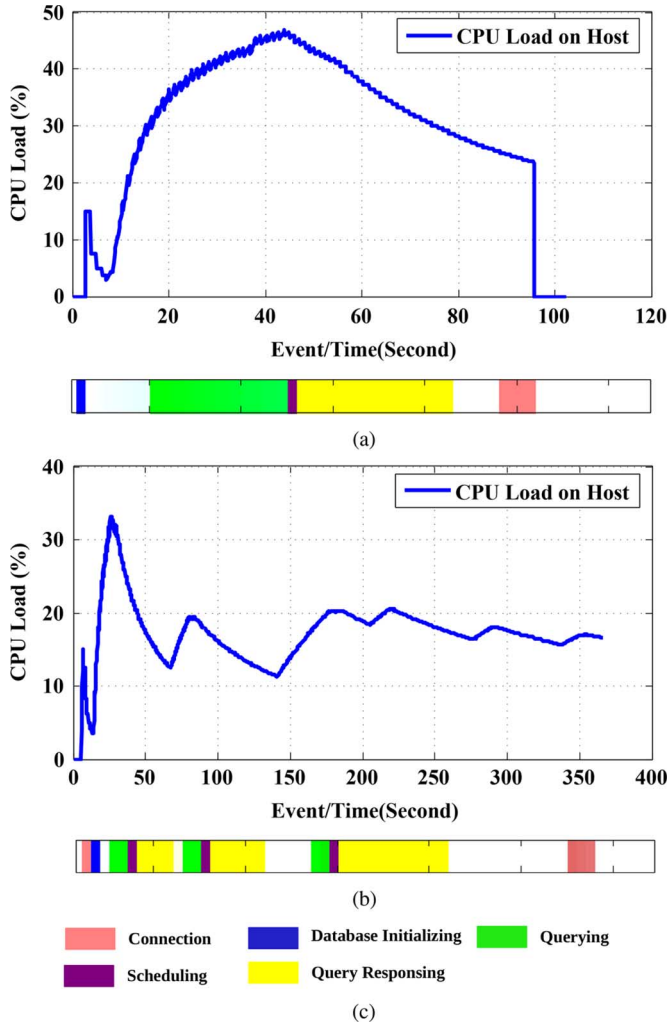


Fig. 7. Time chart of MSDR processing. Note that even with minor management, subfigure (b) demonstrates an optimized performance in terms of CPU load. (a) Process of one period of 12-parallel requests. (b) MSDR process with three successive periods of 4-parallel requests. (c) Color denotation.

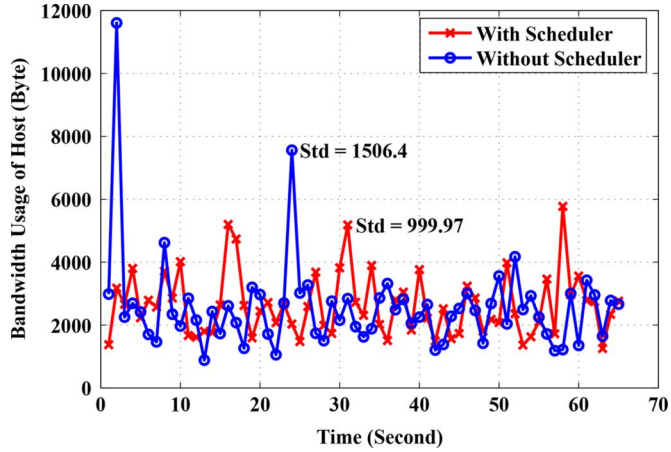


Fig. 8. Bandwidth usage comparison between with and without scheduler.

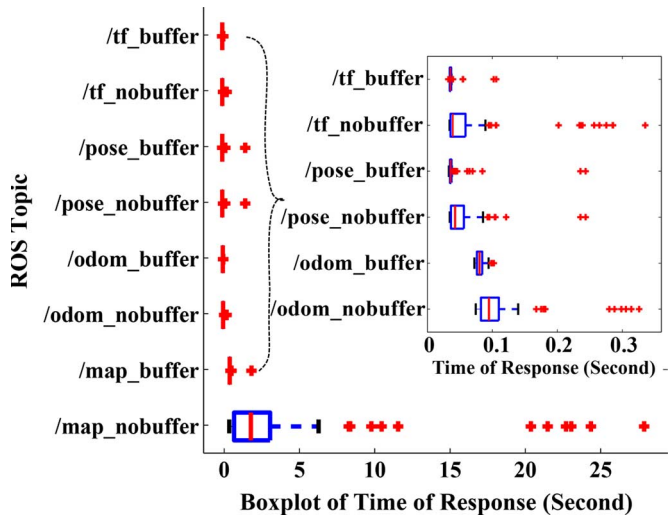


Fig. 9. ToR comparison between with and without buffer. (Red lines mark the mean of ToR. The edges of the blue box are the 25th and 75th percentiles, Black lines mark some extreme data points.)

load than the first one [see Fig. 7(b)]. This shows that the *local data buffer* introduced in Section IV stores the queried data and alleviates data retrieval even if multiclients are requesting the data simultaneously.

We compared the *bandwidth usage* under the following two situations. One is using the proposed scheduling strategy in the CCH, as shown in Fig. 2, and the other one is not. Fig. 8 depicts the *bandwidth usage* of MSDR between clients and the CCH. The standard variance of *bandwidth usage* is 1506.4 without the scheduler, and is 999.97 with the scheduler. As depicted in the red curve decorated by triangle, the *bandwidth usage* confronts two peaks when no scheduler is available. This would result in packet dropping, network congestion, and unstable response.

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, which means the data retrieval is from a data center located in outside networks. For each request task, it includes six independent requests from one client, the package size is $s = 15.625\ \text{Mb} \times 6$, then the ideal transmission time should be $s/(2\ \text{Mb/s}) = 46.875$ 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 IV, 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 scheduler 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.

2) Scenario 2—Heterogeneous Data Retrieval: In this scenario, clients queried one type of data among `map`, `tf`, `pose`,l and `odom` each request. The aim of the scenario is to justify the following:

- effects of constrained bandwidth resource;
- improvements of data retrieval achieved by the proposed buffer in the CCH;
- complexity of the scheduler proposed in the CCH.

Therefore, we compared the *ToR* and *CPU load* under the two cases when there was a buffer or not, and *CPU load* under the two cases whether there was a scheduler or not, respectively.

TABLE IV
RoR Comparison Between With and Without Scheduler in the Buffer Under Different Timeout Period

Threshold K_{th}	Timeout Period T_0 (Second)							
	10		20		30		40	
	No Scheduler	Scheduler	No Scheduler	Scheduler	No Scheduler	Scheduler	No Scheduler	Scheduler
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%

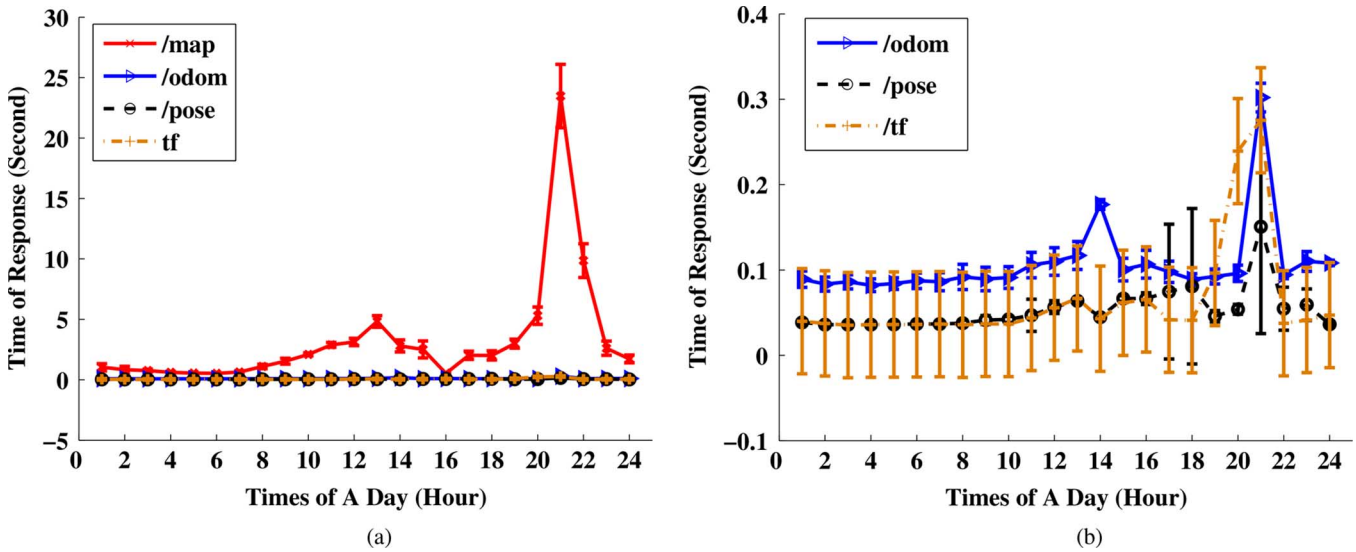


Fig. 10. ToR comparison among ROS topics during a day. (a) ToR comparison among `/map`, `/odom`, `/pose`, `/tf`. (b) ToR comparison among `/odom`, `/pose`, `/tf`.

At first, we compared the statistical *ToR* of requests through the Internet during a day when there was applying the aforementioned buffer or not. As shown in Fig. 9, the average of *ToR* have reduced significantly when there is a buffer that stores the frequently queried data. Especially for messages with large data size such as `map`, the median value of *ToR* reduces from 1.945 to 0.5424, which is much more than other three types of messages. The other three types of messages `odom`, `pose`, and `tf` reduces from 0.1675, 0.04236, and 0.03917 to 0.0928, 0.039, and 0.0323, respectively. The reason is that the large size messages are easily affected by the Internet status. The network bandwidth can be considered as an unconstrained resource, because the other three types of messages have very small data size. This indicates that CCH is not necessary to process the requests by retrieving them again in the database, since such data have persisted in the buffer once the same request has been responded. Time spends only on request admission and data matching. Thus, CCH is relieved from redundant query, which reduces the response time. We also show the *ToR* performances of different types of messages through the Internet during a day in Fig. 10. It can be observed from Fig. 10(a) that the network traffic is very busy from 20 o'clock to 22 o'clock, especially for large data such as `map`, which leads to a long time delay. Due to the *ToR* values of message *odom, pose*, and `tf` are too small to see, we separately plot them in Fig. 10(b), which also shows the longer time delay between 20 o'clock and 22 o'clock.

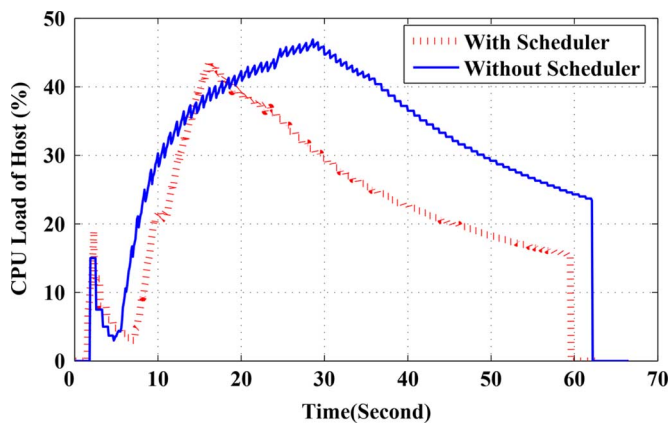


Fig. 11. CPU load comparison between with and without scheduler.

Second, we compared the *CPU load* between with and without the scheduler. In order to get a stable result, 12 robot clients simultaneously request 12 heterogeneous data, which are composed of `map`, `odom`, `pose`, and `tf`, and are demonstrated in Fig. 11. We can see the red curve shows the CCH has higher *CPU load* when scheduling multi requests around the 15th second, but it saves more computation power for data retrievals in the database afterwards. Without the scheduler, the blue curve shows higher *CPU load* around the 28th second for the congestion of multirequests. Therefore,

the *CPU load* performs better as well when the scheduler is applied.

VII. Conclusion

In this paper, we have introduced the design, implementation and evaluation of multi sensor data retrieval strategies for cloud robotic systems. We proposed an architecture consists of a data center, cloud cluster hosts and robot clients. In addition, we tackled the problem of MSDR among the host-based framework by defining the problem into a Stackelberg game and offered theoretical optimization analysis. Our proposed scheduling scheme with a data buffer are implemented in the cloud cluster host module. In order to evaluate the proposed strategies, we define the QoS criteria that is used in the experiments. Our experimental results demonstrate significant improvement of the proposed approach in terms of *ToR*, *RoR*, *bandwidth usage*, and *CPU load*, by adopting the proposed strategies for resource retrieval. For future study, aiming at the optimization of data requirement for dynamic robotic tasks, the scheduler will be explored concerning a prediction model for the completion time of the required data.

References

[1] M. Liu, C. Pradalier, F. Pomerleau, and R. Siegwart, "Scale-only visual homing from an omnidirectional camera," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2012, pp. 3944–3949.

[2] M. Liu and R. Siegwart, "Navigation on point-cloud—A Riemannian metric approach," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 4088–4093.

[3] M. Liu, C. Pradalier, and R. Siegwart, "Visual homing from scale with an uncalibrated omnidirectional camera," *IEEE Trans. Robot.*, vol. 29, no. 6, pp. 1353–1365, Dec. 2013.

[4] F. Colas, S. Mahesh, F. Pomerleau, M. Liu, and R. Siegwart, "3d path planning and execution for search and rescue ground robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2013, pp. 722–727.

[5] M. Liu and R. Siegwart, "DP-FACT: Towards topological mapping and scene recognition with color for omnidirectional camera," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2012, pp. 3503–3508.

[6] M. Liu, F. Colas, L. Oth, and R. Siegwart, "Incremental topological segmentation for semi-structured environments using discretized GVG," *Auton. Robot.*, vol. 38, no. 2, pp. 143–160, 2014.

[7] M. Liu and R. Siegwart, "Information theory based validation for point-cloud segmentation aided by tensor voting," in *Proc. IEEE Int. Conf. Inform. Autom. (ICIA)*, 2013, p. 168173.

[8] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 398–409, Apr. 2015.

[9] E. Guizzo, "Robots with their heads in the clouds," *IEEE Spectrum*, vol. 48, no. 3, pp. 16–18, Mar. 2011.

[10] G. Hu, W.-P. Tay, and Y. Wen, "Cloud robotics: Architecture, challenges and applications," *IEEE Network*, vol. 26, no. 3, pp. 21–28, May 2012.

[11] P. Matti, "Nash and stackelberg solutions in a differential game model of capitalism," *J. Econ. Dyn. Control*, vol. 6, no. 0, pp. 173–186, 1983, 0165-1889 doi: 10.1016/0165-1889(83)90048-9.

[12] L. Wang, M. Liu, M. Q.-H. Meng, and R. Siegwart, "Towards real-time multi-sensor information retrieval in cloud robotic system," in *Proc. IEEE Conf. Multisens. Fusion Integr. Intell Syst. (MFI)*, Sep. 2012, pp. 21–26.

[13] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Proc. 5th IEEE Int. Joint Conf. INC, IMS and IDC, NCM'09*, 2009, pp. 44–51.

[14] Y. Chen, Z. Du, and M. García-Acosta, "Robot as a service in cloud computing," in *Proc. IEEE Int. Symp. Service Oriented Syst. Eng.*, Jun. 2010, pp. 151–158.

[15] A. Sheth and A. Ranabahu, "Semantic modeling for cloud computing, Part 1," *IEEE Internet Comput.*, vol. 14, no. 3, pp. 81–83, May–Jun. 2010.

[16] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, Eucalyptus: A technical report on an elastic utility computing architecture linking your programs to useful systems Tech. Rep., 2008.

[17] OpenNebula Project, "Opennebula.org—The open source toolkit for cloud computing," 2008. [Online]. Available: http://www.opennubula.org/

[18] P. Sempolinski and D. Thain, "A comparison and critique of eucalyptus, opennebula and nimbus," in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2010, pp. 417–426.

[19] M. Piaggio and R. Zaccaria, "Distributing a robotic system on a network: The ethnos approach," *Adv. Robot.*, vol. 11, no. 8, pp. 743–758, 1998.

[20] R. Arumugam, V. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, A. Kumar, K. D. Meng, and G. W. Kit, "Davinci: A cloud computing framework for service robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2010, pp. 3084–3089.

[21] Gostai Coop, "Gostai," 2010. [Online]. Available: http://www.gostai.com/

[22] Aldebaran Robotics, "Nao robot," 2013. [Online]. Available: http://aldebaran-robotics.com/

[23] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft, "Roboearth," *IEEE Robot. Autom. Mag.*, vol. 18, no. 2, pp. 69–82, 2011.

[24] M. Stenmark, J. Malec, and K. Nilsson, "On distributed knowledge bases for small-batch assembly," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 519–528, Apr. 2015.

[25] G. M. Hunziker Dominique, "The RoboEarth Cloud Engine," 2013. [Online]. Available: http://doc.roboearth.org/rce

[26] B. Kehoe, D. Warrier, S. Patil, and K. Goldberg, "Cloud-based grasp analysis and planning for toleranced parts using parallelized monte carlo sampling," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 455–470, Apr. 2015.

[27] C. E. Aguero, N. Koenig, I. Chen, H. Boyer, S. Perters, J. HSU, B. Gerkey, E. Krotkov, and G. Pratt, "Inside the virtual robotics challenge: Simulating real-time robotic disaster response," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 494–506, Apr. 2015.

[28] B. D. Gouveia, D. Portugal, D. C. Silva, and L. Marques, "Computation sharing in distributed robotic systems: A case study on slam," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 410–422, Apr. 2015.

[29] Z. Du, W. Yang, Y. Chen, X. Sun, X. Wang, and C. Xu, "Design of a robot cloud center," in *Proc. 10th Int. Symp. Auton. Decentralized Syst. (ISADS)*, Mar. 2011, pp. 269–275.

[30] M. Stenmark, J. Malec, and K. Nilsson, "On distributed knowledge bases for small-batch assembly," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 519–528, Apr. 2015.

[31] J. Salmeron-Garcia, J. Malec, and K. Nilsson, "A tradeoff analysis of a cloud-based robot navigation assitant using stereo image processing," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 444–454, Apr. 2015.

[32] B. Liu, Y. Chen, E. Blasch, K. Pham, D. Shen, and G. Chen, "A holistic cloud-enabled robotics system for real-time video tracking application," in *Future Information Technology*. New York, NY, USA: Springer, 2014, pp. 455–468.

[33] L. Wang, M. Liu, and M. Q.-H. Meng, "Towards cloud robotic system: A case study of online co-localization for fair resource competence," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO'12)*, Dec. 2012, pp. 2132–2137.

[34] D. Thiruvady, A. Ernst, and G. Singh, "Parallel ant colony optimization for resource constrained job scheduling," *Ann. Oper. Res.* pp. 1–18, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10479-014-1577-7

[35] M. Rodriguez and R. Buyya, "Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 222–235, Apr. 2014.

[36] M.-Y. Cheng, D.-H. Tran, and Y.-W. Wu, "Using a fuzzy clustering chaotic-based differential evolution with serial method to solve resource-constrained project scheduling problems," *Autom. Construction* vol. 37, no. 0, pp. 88–97, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0926580513001593

[37] B. An and V. Lesser, "Characterizing contract-based multiagent resource allocation in networks," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 40, no. 3, pp. 575–586, Jun. 2010.

[38] V. Jalaparti, G. Nguyen, I. Gupta, and M. Caesar, Cloud Resource Allocation Games Dept of Computer Science, Tech. Rep, 2010.

[39] L. Wang and M.-H. Meng, "A game theoretical bandwidth allocation mechanism for cloud robotics," in *Proc. 10th IEEE World Congr. Intell. Control Autom. (WCICA)*, 2012, pp. 3828–3833.
[40] H. Ren and M.-H. Meng, "Game-theoretic modeling of joint topology control and power scheduling for wireless heterogeneous sensor networks," *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 4, pp. 610–625, Oct. 2009.
[41] L. Wang, M. Liu, and M.-H. Meng, "Hierarchical auction-based mechanism for real-time resource retrieval in cloud mobile robotic system," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Jun. 2014.
[42] M. Tan, L. Wang, D. Tardioli, and M. Liu, "A resource allocation strategy in a robotic ad-hoc network," in *Proc. IEEE Int. Conf. Auton. Robot Syst. Competitions (ICARSC)*, May 2014, pp. 122–127.
[43] L. Wang, M. Liu, and M.-H. Meng, "An auction-based resource allocation strategy for joint-surveillance using networked multi-robot systems," in *Proc. IEEE Int. Conf. Inform. Autom. (ICIA)*, Aug. 2013, pp. 424–429.
[44] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proc. IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.
[45] M. G. Lagoudakis, E. Markakis, D. Kempe, and P. Keskinocak, "Auction-based multi-robot routing," in *Proc. Int. Conf. Robot.: Sci. Syst. (ROBOTICS)*, 2005, pp. 343–350.
[46] J. Higuera, C. Gamboa, and G. Dudek, "Fair subdivision of multi-robot tasks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA'13)*, 2013, pp. 2999–3004.
[47] M. Zavlanos, L. Spesivtsev, and G. Pappas, "A distributed auction algorithm for the assignment problem," in *Proc. 47th IEEE Conf. Decision Control (CDC'08)*, Dec. 2008, pp. 1212–1217.
[48] J. Capitan, M. T. Spaan, L. Merino, and A. Ollero, "Decentralized multi-robot cooperation with auctioned pomdps," *Int. J. Robot. Res.*, vol. 32, no. 6, pp. 650–671, 2013.
[49] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt, "Robot exploration with combinatorial auctions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS'03)*, Oct. 2003, vol. 2, pp. 1957–1962.
[50] L. Lin and Z. Zheng, "Combinatorial bids based multi-robot task allocation method," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA'05)*, 2005, pp. 1145–1150.
[51] C. Date and H. Darwen, *A Guide to the SQL Standard*. Reading, MA, USA: Addison-Wesley, 1987, vol. 3, et al..
[52] D. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proc. IEEE*, vol. 85, no. 1, pp. 6–23, Jan. 1997.
[53] G. L. Moshe Zadka, "The twisted network framework," 2010. [Online]. Available: http://twistedmatrix.com/user/glyph/ipc10/paper.html
[54] T. Basar and R. Srikant, "Revenue-maximizing pricing and capacity expansion in a many-users regime," in *Proc. 21st IEEE Annu. Joint Conf. Comput. Commun. Soc. (INFOCOM'02)*, 2002, vol. 1, pp. 294–301.

Lujia Wang (S'11) received the M.S. degree in information engineering from the Northeastern University, Shenyang, China, in 2008, and the Ph.D. degree from the Department of Electronic Engineering, Chinese University of Hong Kong (CUHK), Hong Kong, in 2015.

She was a Research Assistant with the Shenzhen Institutes of Advanced Technology Chinese Academy of Sciences from 2008 to 2010. Her current research interests market-based resource allocation strategies for cloud robotics, collaboration and localization of multirobot systems, network robots, sensor fusion and intelligent perception of robots, etc.

Ming Liu (M'12) received the B.A. degree in automation from Tongji University, Shanghai, China, in 2005. During his master study at Tongji University, he stayed for one year at Erlangen-Nurnberg University and Fraunhofer Institute IISB, Germany, as a Visiting Scholar. He received the Ph.D. degree from the Department of Mechanical Engineering and Process Engineering, ETH Zurich, Zurich, Switzerland, in 2013.

He is now a Visiting Assistant Professor with the Department of Electronics and Computer Engineering, Hong Kong University of Science and Technology. His current research interests include autonomous mapping, visual navigation, topological mapping and environment modeling, etc.

Prof. Liu is the recipient of the Best Student Paper Award at IEEE MFI 2012, the Best Paper in Information Award at IEEE ICIA 2013, the Best RoboCup Paper at IEEE IROS 2013, and twice the Winning Prize of the Chunhui-Cup Innoviation Contest.

Max Q.-H. Meng (S'88–M'91–SM'06–F'08) received the Ph.D. degree in electrical and computer engineering from the University of Victoria, Victoria, BC, Canada, in 1992.

He has been a Professor of Electronic Engineering at the Chinese University of Hong Kong since 2002, after working for ten years in the Department of Electrical and Computer Engineering, University of Alberta, Canada, as the Director of the ART and Professor. He is currently holding honorary positions as a Distinguished Professor at the State Key Laboratory of Robotics and Systems, Harbin Institute of Technology, a Distinguished Provincial Professor of the Henan University of Science and Technology, and the Honorary Dean of the School of Control Science and Engineering, Shandong University, China. His research interests include robotics, perception and sensing, human-robot interaction, active medical devices, biosensors and sensor networks, and adaptive and intelligent systems. He has published more than 500 journal and conference papers and served on many editorial boards.

Proc. Meng is an Elected Member of the Administrative Committee (AdCom) of the IEEE Robotics and Automation Society. He is a recipient of the IEEE Third Millennium Medal award and he is a Fellow of IEEE.