Blockchain-based Volunteer Edge Cloud for IoT Applications

Ming-Tuo Zhou
Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences
Shanghai, China
mingtuo.zhou@mail.sim.ac.cn

Feng-Guo Shen
Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences
Shanghai, China
University of Chinese Academy of Sciences
Beijing, China
guofeng.shen@mail.sim.ac.cn

Tian-Feng Ren, Xin-Yu Feng
Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences
Shanghai, China
University of Chinese Academy of Sciences
Beijing, China
tfren@mail.sim.ac.cn,
fengxinyu0630@163.com

2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring) | 978-1-7281-8964-2/20/$31.00 ©2021 IEEE | DOI: 10.1109/VTC2021-Spring51267.2021.9449041

***Abstract*—Cloud computing has been widely used in the field of information services. However, large-scale Internet of things (IoT) applications are raising new challenges to cloud computing architecture. Edge computing, which complements cloud computing, is considered to be the way to address these challenges. Volunteer computing, which harvests idle resources in the network can improve the hardware utilization rate and support tens of billions of IoT devices. In view of the limitations of traditional volunteer computing that cannot provide realtime services and has no mechanism to reward services in existing volunteer clouds, this paper presents blockchain-based volunteer edge cloud. A common runtime environment is provided by container technology, and blockchain smart contract is used for critical business steps and computing service payment. Volunteer edge cloud systems based on blockchain is introduced from a top-level perspective, and a prototype system build on Ethereum and KubeEdge is described in detail. On top of the prototype system, we deployed an example IoT application of robot formation control. It demonstrates the benefits of volunteer edge cloud in reducing the complexity of IoT devices, improving the flexibility of software development, and paying the computing service.**

***Keywords*—blockchain, edge cloud, volunteer computing**

I. Introduction

The past decade or two has witnessed the success of cloud computing. Supported by virtualization, large-scale storage, platform management and other technologies, cloud computing is now applied in 90% of companies, carrying 83% of the enterprise workload [1][2]. With cloud computing, resource is delivered as services over the Internet by implementing hardware and system software in the data center, and it has made the dream of computing as a utility come true [3].

However, fast-growing IoT applications are posing new challenges to cloud computing architecture[4]. These challenges include 1) strict delay requirements: for most industrial control systems, the delay between sensors and control nodes should be in the order of milliseconds. Many other IoT applications, like vehicle-to-vehicle communication, virtual reality applications, etc., require a latency of less than tens of milliseconds. Such delay requirements are difficult for existing mainstream cloud services to provide. 2) Network bandwidth limitation. Internet of Things devices are constantly producing a lot of data, for example, an autonomous vehicle will generate 1GB of raw data every second [5]. For billions of devices connected to the Internet of Things, transferring data to remote data centers for processing can place enormous pressure on network facilities and data center bandwidth. 3) Data security concerns. The particularity of IoT applications inevitably requires the collection of user privacy data, but many people have expressed concerns about data security. Therefore, they hope that IoT devices can work behind a controlled firewall, rather than upload all data to cloud servers. 4) Terminal performance limitations. In order to extend the battery life of IoT devices, designers often choose to use low-power processors and set software programs to sleep state most of the time. In addition, considering hardware costs, consumer products will also be equipped with low-end processors. This means that the computing power of IoT terminal devices is limited, and external resources are needed to complete their tasks. However, it is difficult to encrypt the data multiple times before sending it to the data center.

Emerging edge computing, complementary to cloud computing, is considered a new computing paradigm to meet the challenges of the Internet of Things. Edge computing extends resources (computing, network, storage) to the edge of the network and responds to requests near where data is generated. However, how to build a vast and ubiquitous edge computing facility is another problem that needs to be solved. Inspired by voluntary computing, organizing devices distributed in the network and using their resources to build a volunteer edge cloud can naturally support many IoT applications and improve the hardware utilization efficiency in the community. However, as presented in the literature review part in this article, the conventional volunteer computing schemes have no real-time response, mechanism to encourage sharing resource and pay for services, or method to maintain the credibility of data that may be processed in different edge devices belonging to different owners.

Blockchain technology is essentially a kind of distributed ledger, with the characteristics of decentralization, openness, independence and security. These advantages make blockchain a technology to make up for the lack of volunteer edge cloud computing. First of all, based on the blockchain technology to build a resource trading platform, to provide incentives for users who provide computing resources, so as to achieve the role of motivating users. Secondly, only transaction records are recorded on the blockchain. Users are anonymous to each other, which ensures the privacy of users.

This work was supported by the STCSM Science and Technology Innovation Program on Hightech under Grant No. 18511106500.

Transaction records on the blockchain can not be tampered with, which ensures the security of transactions. Therefore, for volunteer cloud computing, it is necessary to supplement blockchain technology, which can solve many existing problems.

In view of the limitations of the applicable scenarios of the existing volunteer computing model, this paper proposes a decentralized volunteer edge cloud architecture that supports real-time applications for the Internet of Things. The proposed architecture consists of consumers, resource providers, resource pool, and blockchain infrastructure. Consumers are bodies that use the edge cloud services; computing resource providers are nodes distributed in the network that belong to different owners and provide resource for the edge cloud; resource pool is the hub of the volunteer edge cloud that coordinates the providers and the consumers; blockchain is used to record contributions of source providers, make immediate payment, and secure data, etc. We also developed a testbed robot formation control based on the proposed architecture and demonstrated the benefits of the proposed blockchain-based volunteer edge cloud.

The rest of paper is organized as follows. Section 2 reviews literature. Section 3 presents an overview of the proposed edge cloud architecture. Section 4 is the details of system deployment. Section 4 describes an IoT use case, which is driven by our volunteer edge cloud, and Section 5 concludes the article.

II. Literature Review

There has been some research in voluntary computing. In the 1990s, there was an upsurge in the research of volunteer computing and grid computing. At that time, the main motivation was to overcome the problem of insufficient single-machine computing power and run complex computing tasks through idle computers on a federated network. During this period, a number of successful engineering projects were generated. BONIC is a high-performance distributed volunteer computing platform created by the University of California, Berkeley, in 2002 [6]. It was originally designed to implement the SETI @ home computing project (it was suspended indefinitely at the end of March 2020). This project enables personal computer users to run radio telescope data analysis tasks when the CPU is idle and participate in the search for alien civilizations. BONIC is an open computing platform and currently runs more than 30 science and technology projects. XtremWeb-HEP from University Paris Sud is another volunteer computing platform [7], it was originally designed to meet large-scale computing needs of physicists at Pierre Auger Observatory. XtremWeb is developed in Java and is divided into two parts: worker and server. It has the characteristics of multi-application support, high scalability and error tolerance. It has designed a unique communication protocol and local code security operation mechanism according to actual needs. At that time, volunteer computing was aimed at offline computing scenarios, but it was not suitable for real-time services such as Internet of Things applications.

Some other work has studied the edge cloud platform based on the volunteer model. A. M. Khan and others have established a shared edge cloud based on the home server [8]. Based on container technology, they allow hardware platforms of different providers to provide unified services for IoT applications. In the face of low efficiency of distributed data intensive applications, M. Ryden et al. proposed a distributed cloud infrastructure named Nebula, which uses voluntary edge resources for computing and data storage [9]. They also verified the robustness of Nebula to various failures by deploying MapReduce framework. T. M. Mengistu and others believe that volunteer cloud is an economic, safe and green way to replace the existing cloud computing model [10], and they propose the concept of Volunteer Computing as a Service (VCaaS). The authors introduced the implementation details of integration with the open source IaaS cloud management system CloudStack, and then deployed a Hadoop-based use case on Amazon EC2 to evaluate system performance.

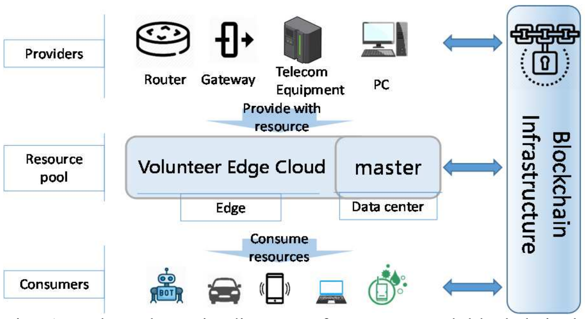


Fig. 1. The schematic diagram of our proposed blockchain based volunteer edge cloud

The above projects have pushed the applicable scenarios of volunteer computing to a more versatile program. However, the nodes participating in the construction of the voluntary edge cloud have different ownerships, forming a peer-to-peer situation. In order to encourage different resource owners to share resource, it is necessary to design a mechanism to inspiring computing resource sharing and rewarding, meanwhile, carefully maintain the credibility of data in such a decentralized system. There are few considerations and solutions to this problem in previous studies.

III. The Proposed Architecture

The so-called voluntary edge cloud is such that it builds an edge cloud by organizing resources (computing, storage, and network) contributed by volunteer nodes in the network to provide real-time interactive services. Volunteer edge cloud mainly has the following three characteristics: a resource pool is formed by distributed volunteer work nodes (volunteer feature), resources are provided by edge nodes, and IoT terminal devices serve as users (edge feature), supporting realtime interactive services (Cloud computing features).

Blockchain technology is used to address issues of recording service provision and payment, data reliability and credibility in the system. Blockchain is basically a technology of distributed ledger and has the nature of automatic payment for contributions, then it is very convenient to use blockchain to record contributions and monetizing the services. Meanwhile, since the edge nodes participating in construction of the volunteer edge cloud belong to multiple owners, the users who utilize the resources are also independent individuals, which forms a typical decentralized system. Any individual may act maliciously for various purposes. For example, an edge node claims that it has more resources than it actually does, thereby tricking the scheduler to obtain more tasks in return for rewards; malicious users keep submitting tasks to exhaust the resources of the edge cloud, and so on.

With the help of tamper-proof and data-transparent features of blockchain technology, malicious behavior can be countered.

The schematic diagram of our proposed blockchain-based volunteer edge cloud is shown in Fig. 1. There are mainly four components in the proposed architecture, namely providers, consumers, resource pool, and blockchain infrastructure. Providers are node devices in the network that contribute resources to the edge cloud. They are at the edge of the network and belong to different owners. They can be network server facilities such as soft routers, gateways, telecommunications equipment, or personal computers. In order to improve resource utilization and increase return on investment, administrators set these facilities as edge nodes and volunteer to join the edge cloud resource pool. Consumers are the main body using volunteer edge cloud resources, especially low-performance terminal devices that need edge cloud-assisted computing, such as robots, connected vehicles, mobile phones, and sensor nodes. The resource pool is the hub of the volunteer edge cloud, which includes a collection of working nodes at the edge and a master controller at data center. The master is responsible for managing edge nodes, receiving task requests, and orchestrating task instances. Therefore, the resource pool coordinates providers and consumers. Blockchain infrastructure is responsible for the communication between above three types of components and stores key data. Including multiple edge nodes will inevitably cause data reliability problems. To this end, the blockchain (distributed ledger) technology is used to perform service transactions and deliver messages, and the key business logic is executed by the program on the chain, so that the data can be publicly traceable and finally reach an agreement. The decentralized feature of blockchain technology is naturally suitable to solve the data security problems in such distributed applications.

The working process of volunteer edge clou d is as follows. First, those devices that are willing to contribute their own resources become edge nodes. Under the management of master controller, they join the resource pool of volunteer edge cloud and may sell their services using the blockchain system. Then, when computing task requirements arise in IoT devices, a task request is sent to the edge cloud controller through the blockchain network. Finally, the edge cloud controller schedules task to appropriate edge node and starts it, and when task completed the quality of the service is checked and paid if the results are reliable. The running service program provides support for the Internet of Things applications.

IV. Implementation Details

To implement the blockchain-based volunteer edge cloud, there are two aspects of work. One is the design of a blockchain smart contract that supports automatic transactions and data credibility, and the other is to manage edge nodes and orchestrate request tasks. In addition, to lower the usage barrier, a web-based client is designed for end users to submit tasks.

A. *On Chain Contract*

The main responsibility of the smart contract on chain is to store and certificate key work-flow and data, and make payment to the services. For work-flow, including the joining of volunteer nodes, task request, etc., for the data, covering the specifications of nodes, the score of a user owns etc.

The smart contract is based on Ethereum, which is a public blockchain infrastructure that supports Turing completed program. Such a decision is based on the fact that the Ethereum network is large scare, with complete development documentation and an active community.

Three data structures, Node, ClaimedNodeSpecs, TaskRequest are defined in the smart contract, and a mapping of the contract address to each structure instances is defined respectively. These maps are used to store resource pool node collections, nodes that send join requests but wait to be processed by the master, and tasks that have not been started after submission.

The three methods requestToJoin, orderTask, taskComplate are used for the interaction between the contract and the client. They are used for new node join request, task submission, and task operation completion, respectively. The method of each contract corresponds to an event, and the event log is emitted under corresponding conditions, asynchronously communicates with the client, and the event is permanently recorded in the block.

As shown in Fig 2, tasks are deployed in the form of containers, kubeedge is responsible for managing and scheduling tasks, and edge tasks interact with the blockchain platform through edge client.

Users send requests to the blockchain platform, which records transaction information and seeks transaction tasks through edge client. Edge hub and cloud search and execute user request tasks in their respective architectures. When the task is finished, the transaction is completed and recorded on the block.

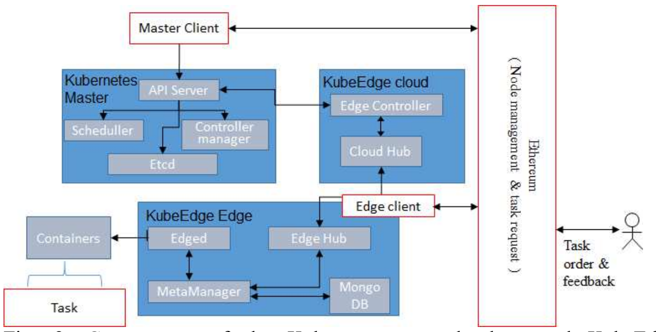
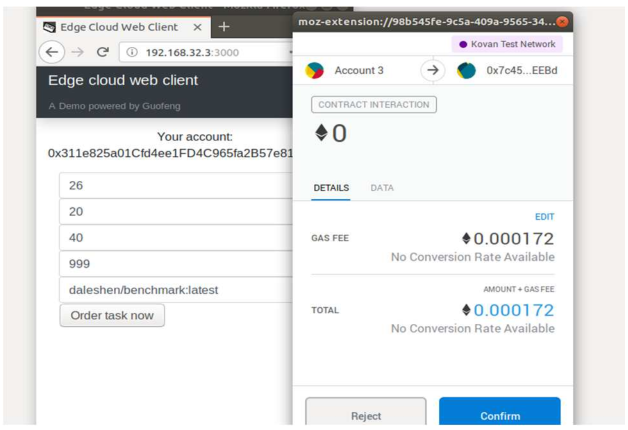


Fig. 2. Components of the Kubernetes control plane and KubeEdge components, as well as the integration of smart contracts and off-chain software.



Edge Cloud Web Client

192.168.32.3:3000

Edge cloud web client

A Demo powered by Guofeng

Your account:
0x311e825a01Cfd4ee1FD4C965fa2B57e81

26

20

40

999

daleshen/benchmark:latest

Order task now

moz-extension://98b545fe-9c5a-409a-9565-34...

Kovan Test Network

Account 3 → 0x7c45...EEBd

CONTRACT INTERACTION

0

DETAILS DATA

EDIT

GAS FEE 0.000172
No Conversion Rate Available

AMOUNT + GAS FEE
TOTAL 0.000172
No Conversion Rate Available

Reject Confirm

Fig. 3. Order task via web-based contract client.

B. *Edge Node Management*

Edge node management is based on KubeEdge. This is an open source edge cloud project that extends containerized application orchestration and device management to host nodes at the edge. KubeEdge is built on Kubernetes, retains the control plane of Kubernetes, and a node agent is redeveloped to replace the original kubelet. KubeEdge is divided into two parts: Edge and Cloud. Together with the Kubernetes native Master, the components of each part of KubeEdge are shown in Fig. 2. In order to integrate smart contracts into the off-chain edge cloud, a Master Client and an Edge Client are developed on the cloud and node respectively to interact with smart contracts on the blockchain.

C. *Web Client*

The only way for users to submit task requests to the volunteer edge cloud is through smart contracts. In order to reduce the difficulty for ordinary users, we developed a web-based smart contract client using Web3.js, as shown in the Fig. 3. Users could send a task request by simply filling in the required node specifications and pressing a button to invoke the orderTask method. The resource needed for the task is automatically matched, and the price of the service is given for choose different providers from the resource pool.

V. Testbed Demonstration

Based on the aforementioned blockchain-based volunteer edge cloud platform, we deployed an IoT application-robot formation to verify the proposed architecture and further illustrate the workflow of the volunteer edge cloud.

A. *The Task of Robot formation following*

At present, mobile robots have been widely used in various fields. In many scenarios, multiple robots are required to perform tasks in cooperation [11][12]. The increasingly complex robot applications are placing higher demands on the performance of local processors and data interaction, which double the complexity and cost of robot systems. For this reason, the industry has proposed the concept of "cloud robots"[13]. However, because the network performance from the terminal to the data center is not controllable, the traditional cloud computing architecture is only suitable for limited types of services, but not support delay-sensitive real-time control applications. With edge cloud, nearest computing power is provided for IoT devices, dispersing centralized cloud computing facilities to the edge of the network.

Software part of the robot formation task can be run on the edge node, thereby decoupling hardware and software module of robots. In this way, robot hardware design, especially the selection of on-board computers, no longer needs to consider requirements for specific applications, and only need to care about sensor data reading and actuator control. The hardware cost of robot system can therefore be greatly reduced, and the approaches of abstract and modular also increase the versatility of the robot.

Figure 4 shows the composition of the robot formation system based on volunteer edge cloud. Among them, the mobile robot is a 4WD platform, its local computing power comes from STM32 and Raspberry Pi 3b, running ROS (robot operation system). Edge nodes are software defined LTE base stations [14]. Because these stations are actually x86 computers, they can join the volunteer edge cloud to contribute idle computing power.

The robot formation control application (Control App) running on the edge node is responsible for keeping the formation. During formation movement, each robot sends speed message to the Control App, and the follower sends its video stream data of the camera to the Control App. After calculation, the Control App conveys the speed command that should be executed to the follower, and thus achieve the following control.

B. *Main Calculation Load*

In the edge computing mode, hardware and software are decoupled and the robot is only responsible for collecting sensor data and manipulating the actuator according to the instructions it receives. The main computing load will be in the Control App which is physically running on edge node.

The program in the Control App is computationally intensive. Firstly, the leader's track should be calculated according to robot moving model, and then the desired following trajectory needs to be obtained. Secondly, the image data from the follower needs to be processed in real time, and the relative error between the leader and the follower is calculated with computing vision method. Finally, an algorithm based on iterative learning control is used to calculate the follower's speed command [15].

C. *Task Deployment and Submission*

As mentioned earlier, the robot's software system is based on the ROS framework, which is a distributed meta-operating system for robot domain, or a middleware dedicated to robot software development. Control App is made as an independent ROS Package, which is then built into a Docker image. ROS provides the basic image kinetic-ros-core-xenial, based on which we can build an image containing the ROS Package we developed.

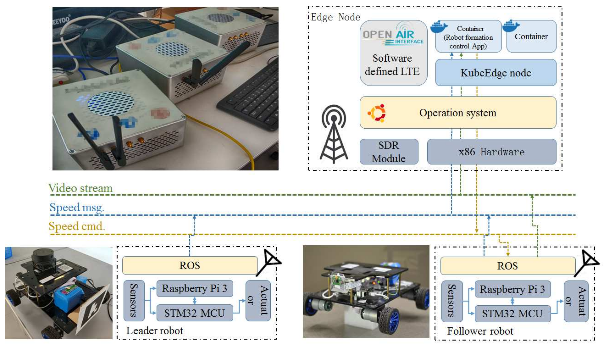


Fig. 4. The composition of the robot formation system.

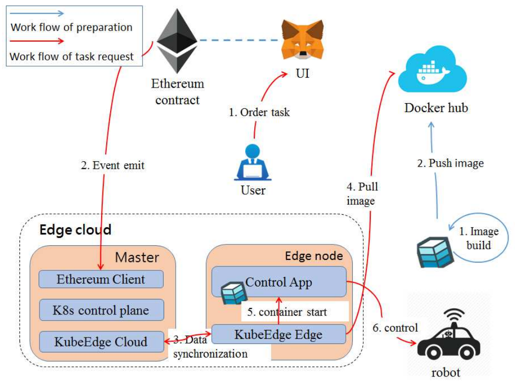


Fig. 5. The work flow of volunteer edge cloud happened in the robot formation control application

To start the program of robot formation control on edge node, we can order the task Control App from edge cloud through the web client we designed. Figure 5 shows the work flow of volunteer edge cloud, in the scenario of robot formation control. It should be pointed out that it shows two kinds of workflow, which are the process of pre-preparing and the process of task submission and running. In the pre-preparation stage, users build application images and push them to public image repositories such as DockerHub. In the task request stage, 1) The user submits a task running request to the Ethereum smart contract, and order the service; 2) The Ethereum client running on the master of the edge cloud is driven by contract events, and read the task running parameters, passing it to Kubernetes control plane; 3) The edge cloud control plane determines the edge node where the task actually runs, and synchronizes the commons to corresponding edge node through the KubeEdge cloud part; 4) The KubeEdge Edge component on edge node downloads the image of Control App from image repository; 5) Kubedge starts the Control App container, at this time, the control program is formally started; 6) The control app running on the edge node connects with the robot, and begins to process the request from robots, sending real-time motion instructions.

D. Experiment Results

The experiment first tested the execution time of the following control program on the edge node and the robot onboard computer, respectively. Secondly, ignore the running performance, but pay attention to the power of the robot. We obtain the power consumption of the onboard computer by measuring the supply voltage and current in two cases, that is, the Control App is offloaded to edge node and is run locally on robot computer.

The two experiment results are shown in Fig 6. The sub-figure on the left shows the time costs for the control app to process a frame of video and iteratively calculate a speed command. On the edge node, the single step calculation time is about 30ms; however, when calculating on the mobile robot's onboard computer, the calculation time is more than 400ms, which is not feasible for a control frequency of 5Hz.

The power consumption test result on the right shows that after the control app is offload from local computer to edge cloud, the power consumption of the robot's onboard computer is reduced from 3.4W to about 2.5W, a decrease of about 25%. Considering that if computing locally, a more powerful processor is actually needed to meet the computing requirements, which has higher power consumption, so the power consumption saved is more significant than that shows in the experiment.

Then we test the following effect of robot formation. Three robots were used in the test, which was carried out in an indoor venue of about 8m by 6m. The test scenarios include 2 types, 1) formation tracking with initial error, and 2) long-distance polyline tracking. The experimental results are shown in Fig. 7, where the trajectory of each robot and the tracking errors of two stages are marked. Regarding the marking symbols, □ is the starting point of the trajectory, ▷ is the end of the trajectory. Result shows that the solution has achieved expected performance. It has the ability to correct initial error. The maximum tracking error is within 0.15m and the average tracking error is about 0.03m when operating long-distance moving.

VI. Conclusion

In this paper, we analyzed the challenges of IoT applications for traditional cloud computing, proposed architecture of volunteer edge cloud, and introduced blockchain technology to deal with the problem of service payment and data credibility in a decentralized system. Then a proof-of-concept system based on Ethereum and KubeEdge was designed. On top of this, we deployed as demonstration of robot formation application, and shown that with the help of edge cloud, the hardware of the IoT terminal can be simplified, a more flexible application design can be achieved, and the service provision can be paid so that more resource sharing can be encourage. This could be a business operation model for, particularly, the conventional network operators to provide edge computing services using equipment nodes distributed in their network and device nodes of their users.

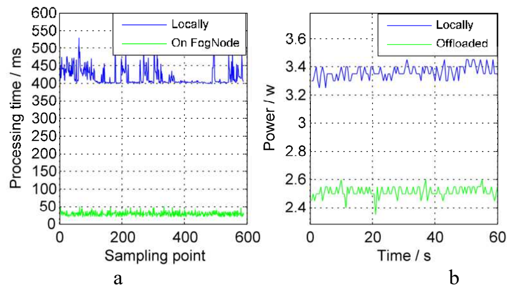


Fig. 6. a) The test result of the program execution time. b) Power consumption of the onboard computer in two conditions.

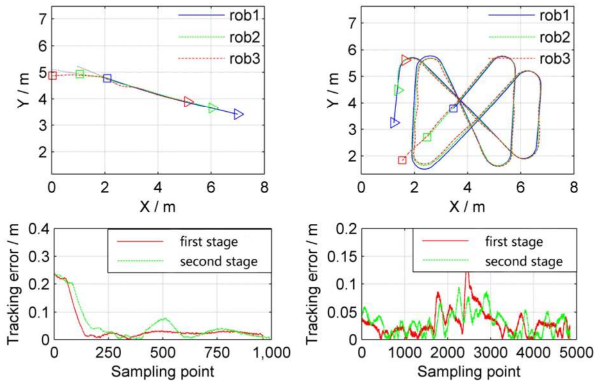


Fig. 7. The trajectory of each robot and the tracking errors of two stages

References

[1] W. Fellows , "A Sea of Change - Migrating Workloads and Applications to the Cloud," BrightTALK, 12-Sep-2018. [Online]. Available:https://www.brighttalk.com/webcast/10363/318591?utm_source=451social. [Accessed: 06-May-2020].

[2] L. Columbus, "83% Of Enterprise Workloads Will Be In The Cloud By 2020," Forbes, 25-Jan-2018. [Online]. Available: https://www.forbes.com/sites/louiscolumbus/2018/01/07/83-of-enterprise-workloads-will-be-in-the-cloud-by-2020. [Accessed: 29-Apr-2020].

[3] Armbrust, M. , Fox, A. , Griffith, R. , Joseph, A. D. , & Zaharia, M. . (2010). A view of cloud computing. Communications of the ACM, 53(4), 50-58.

[4] M. Chiang and T. Zhang, "Fog and IoT: An Overview of Research Opportunities," IEEE Internet of Things Journal, vol. 3, no. 6, pp. 854–864, 2016.

[5] S. Liu, Engineering autonomous vehicles and robots: the DragonFly modular-based approach. Hoboken: Wiley-IEEE Press, 2020, pp. 3.

[6] “BONIC Technicaldocumentation,” https://github.com/BOINC/boinc-devdoc/wiki.[Online].Available:https://boinc.berkeley.edu/trac/wiki/ProjectMain. [Accessed: 30-Apr-2020].

[7] G. Fedak, C. Germain, V. Neri, and F. Cappello, “XtremWeb: a generic global computing system,” Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid.

[8] A. M. Khan and F. Freitag, “On Edge Cloud Service Provision with Distributed Home Servers,” 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2017.

[9] M. Ryden, K. Oh, A. Chandra, and J. Weissman, “Nebula: Distributed Edge Cloud for Data Intensive Computing,” 2014 IEEE International Conference on Cloud Engineering, 2014.

[10] T. M. Mengistu, A. M. Alahmadi, Y. Alsenani, A. Albuali, and D. Che, “cuCloud: Volunteer Computing as a Service (VCaaS) System,” Lecture Notes in Computer Science Cloud Computing – CLOUD 2018, pp. 251–264, 2018.

[11] M. Lutz, C. Verbeek, and C. Schlegel, “Towards a robot fleet for intra-logistic tasks: Combining free robot navigation with multi-robot coordination at bottlenecks,” 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), 2016.

[12] D. Zhaohui, W. Min, and C. Xin, “Multi-robot cooperative transportation using formation control,” 2008 27th Chinese Control Conference, 2008.

[13] S. A. Miratabzadeh, N. Gallardo, N. Gamez, K. Haradi, A. R. Puthussery, P. Rad, and M. Jamshidi, “Cloud robotics: A software architecture: For heterogeneous large-scale autonomous robots,” 2016 World Automation Congress (WAC), 2016.

[14] Inoue Yoshio, “OpenAirUsage,” GitLab. [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/OpenAirUsage.

[15] M.-M. Lv, X.-D. Li, and T.-F. Xiao, “Iterative learning control for linear time-variant continuous systems with iteration-varying initial conditions and iteration-varying reference trajectories,” IEEE ICCA 2010, 2010.