An intern-sufficient cloud for large-scale multi-robot systems and its application in multitarget navigation

Jingtao Zhang¹, Jun Zheng², Xiaodong Zhang³, Kun Zhang¹, Pengjie Xu¹ and Qirong Tang¹

Abstract
Due to generally limited computing capability of an individual robot, cloud-based robotic systems are increasingly used. However, applications in large-scale multi-robot systems will be hindered by communication congestion and consequent lack of computing resources. In this study, an intern-sufficient cloud is investigated to alleviate the burden of communication and thus support more robots. At the same time, it enables heterogeneously idle computing resources of robots inside the system to be shared on demand, instead of relying on cloud servers and communication infrastructures, to make the scope of application wider. To this end, a hierarchical communication mechanism and a resource schedule algorithm are proposed. In the mechanism, the transmission power, signal-to-noise ratio, available bandwidth, and other relevant features are taken into account to estimate link quality for data transmission. Then, the constrained communication conditions and heterogeneous computing resources are balanced by the resource scheduling algorithm, so that the most appropriate computing resources of the robots are contributed to the mobile cloud. Furthermore, a multitarget navigation task is applied on the cloud to validate the work. Thereby, simulations and experiments are performed. The results show that the proposed intern-sufficient cloud can provide stable resources of communication and computation for a multi-robot system with 20 physical robots while achieving more effective multitarget navigation.

Keywords
Multi-robot system, intern-sufficient cloud, multitarget navigation

Introduction
In recent years, the multi-robot system gains a lot of attention due to its high efficiency and robustness.¹ Although the computing capability of processors installed on robots has been getting stronger, many algorithms for better robotic performance are more complicated with the development of robotic technology, such as the application of deep learning.² It means that robots have always been under constrained computing resources. Furthermore, advanced hardware represents high financial costs, which cannot be ignored in multi-robot systems. A possible solution is cloud computing.³,⁴
Nowadays, some researches on cloud-based multi-robot systems have increasingly been carried out. Generally, the robots can access the developed cloud servers by wireless communication systems to obtain computing resources.\textsuperscript{5–7} However, the developed cloud servers, for example, DAvinCi\textsuperscript{9} and RoboEarth,\textsuperscript{9} maybe unavailable in many scenes, such as disaster relief after an earthquake, in which the developed communication infrastructures, for example, 5G base station, are destroyed. Meanwhile, not all robots perform computing expensive algorithms all the time, which results in a waste of computing resources. Therefore, an intern-sufficient cloud is strongly demanded to effectively schedule the resources inside the multi-robot system.

The ad hoc cloud is an emerging paradigm that enabling devices share computing resources on demand inside the system.\textsuperscript{10,11} It aims at forming a self-organized resource-sharing cloud system through resource virtualization and system management.\textsuperscript{12} Some researches have focused on scheduling the heterogeneous computing resources in this cloud paradigm.\textsuperscript{13,14} Specifically, the idle computing resources can be effectively shared by a dictionary-based algorithm\textsuperscript{15} and a double-sided bidding mechanism.\textsuperscript{16} However, the execution time is not considered, which means that the contributed resources may not be the most appropriate. A probabilistic scheduling algorithm was proposed for time-constrained tasks.\textsuperscript{17} Also, efficient task allocation methods were proposed for minimizing execution time.\textsuperscript{18} These two studies made great contributions to scheduling the idle computing resources and using the contributed resources in the mobile ad hoc cloud to minimize computing time. However, the communication conditions were not scheduled in abovementioned methods, which are absolutely indispensable because whether the required computing tasks can be executed in the contributed computing resources in time are also determined by how long the task data are transmitted. Furthermore, communication conditions are usually worse under irrational mechanism in multi-robot systems as the number of robots increases, not to mention the process of obtaining computing resources from the cloud.

The original communication mechanism of the ad hoc network is used in some robotic researches,\textsuperscript{19} but more hops will be caused by the increasing number of robots, which results in a lower success rate of data transmission. Therefore, communication mechanism is a key point of the ad hoc cloud. A communication mechanism using robot operation system (ROS) for multi-robot systems is presented, in which the good performance of the cloud-based robotic system was validated by three robots.\textsuperscript{20} Obviously, the communication system will be heavily congestion if the number of robots is large in a limited area due to the inherent broadcasting mechanism of ROS. There is a Foundation for Intelligent Physical Agents (FIPA)-based communication infrastructure to deal with dynamically changing reconfigurable multi-robot systems, but it was not tested with a large number of robots.\textsuperscript{21} In simulation, a cloud-robot system was constructed to investigate communication mechanism with a large number of robots and the algorithm complexity is discussed with robots increasing.\textsuperscript{22} So it is hard to say its effectiveness in physical large-scale multi-robot systems.

In this study, an intern-sufficient cloud is investigated to improve the ability to support more robots in large-scale multi-robot systems without relying on cloud servers nor communication infrastructures. Furthermore, a multitarget navigation task is applied on the cloud to test the availability of the guaranteed communication links and the contributed computing resources. The major contributions can be summarized as follows:

1. A hierarchical network is proposed, in which communication mechanisms for heavyweight data and lightweight messages are constructed respectively to ensure available communication links.
2. A resource scheduling algorithm is proposed to ensure that only the most appropriate computing resources will be shared, which is achieved by real-locating communication resources and then matching.
3. A multi-robot multitarget navigation task is applied on the intern-sufficient cloud, and experiments with a physical multi-robot system, which has 20 robots, are performed to show the effectiveness.

The rest of this article is organized as follows. The second section is the problem description. The third section details the intern-sufficient cloud. The fourth section shows the implementation of the multitarget navigation task in the cloud. In the fifth section, simulations are performed. In the sixth section, experiments are carried out and results are analyzed. The seventh section gives conclusions.

**Problem description**

In this study, the large-scale multi-robot system is composed of \( N \) \( (N \in \mathbb{N}, N \geq 20) \) robots. An intern-sufficient cloud is constructed by the robots inside the system. Available communication links and computing resources are provided by the cloud, supporting the large-scale multi-robot system to perform tasks in outdoor scenes without relying on the developed communication infrastructures and cloud servers.

During the process of tasks, it is assumed that not all of the robots perform complex computing algorithms all the time, which is reasonable. So there are sufficient idle computing resources inside the system to be scheduled. For individual robot \( i (i \in \{1, \cdots, N\}) \), the idle computing resources at time \( t \) can be represented by \( C_i(t) \), which is related to CPU-take-up rate. However, for robot \( j \) who wants to use the computing resources of robot \( i \), the transmission rate \( T_{ij}(t) \) between robot \( i \) and robot \( j \) is supposed to be considered. For example, the \( C_i(t) \) can process an
image of 100 Kb within 0.5 s, while $T_{0}(t)$ is merely 10 Kb/s. To meet the requirement to complete processing within 1 s, only those images smaller than 10 Kb can be transmitted, which results in a waste of computing resources. In contrast, if there is a large transmission rate but few idle computing resources, a waste of communication resources has appeared. Thereby, the communication resources should be reallocated to keep stable communication links and obtain more available computing resources. The intern-sufficient cloud is composed of these available computing resources.

Further, to better show the effectiveness, not only the performance of the cloud has been validated but a multi-target navigation task combined with the cloud has also been tested. During the task, because the required resources and available resources are constantly changing according to their actual situation, the successful application can definitely indicate the robustness of the cloud.

The proposed intern-sufficient cloud

The intern-sufficient cloud is constructed by a hierarchical network and a resource scheduling algorithm. The multi-robot system is abstractly divided into three layers by the hierarchical network, as shown in Figure 1.

The first (top) layer is a virtualization of the contributed computing resources. The second and third layers are composed of the physical robots, in which the multi-robot system is divided into many subgroups. In one subgroup, there is one leader robot. The second layer is composed of all leader robots, which are selected according to spatial positions and communication conditions to better cover all the robots, which is clearly shown in Figure 2 from a top–down perspective. The resource scheduling algorithm is run in the leader robot to schedule the heterogeneously idle computing resources inside the subgroup, then the scheduled resources are contributed to the cloud. Thereby one subgroup is regarded as one minimal unit of the cloud. The situation of only one subgroup has been researched in our previous publication. The complete cloud and implementation of a multitarget navigation task in the large-scale multi-robot system are emphasized in this study.

The hierarchical network

The hierarchical network is generated by a topology control (TC) algorithm and a subgroup generation (SG) algorithm, which are run sequentially in pairs. Generally, the data that need to be transmitted can be divided into two categories, heavyweight data, for example, images, and lightweight messages, for example, coordinates. So, each robot is assumed to be equipped with two communication devices of different frequency bands, for example, 5 GHz and 2.4 GHz, for those two categories, respectively. The TC algorithm uses 2.4 GHz for fast long-distance transmission of lightweight data among all robots, whereas the SG algorithm uses 5 GHz for heavyweight data in each subgroup.

![Figure 1. Abstract representation of the intern-sufficient cloud.](image1)

![Figure 2. Spatial positions, network topology, and procedures of resource scheduling. Here, $j \in [1, J]$ represents one of $J$ robots in the subgroup of leader robot $i$.](image2)
In the TC process, the robot that firstly received start signals sends messages to all other robots with as few hops (relays) as possible. Fewer hops mean higher success rate of transmission of lightweight messages, which can be represented by

$$S_{Tr} = (1 - P)^m$$ (1)

where $P$ is the failure probability of a hop, and $m$ is the number of hops. Assumed $P = 0.1$, the transmission success rate is 59% after five hops. So, the purpose of the TC is to minimize the number of hops when lightweight messages arrive in the last robot, and the process of which is shown in Algorithm 1. Here, $e_{ij}$ is the signal strength of robot $j$ received by robot $i$, which can be calculated by

$$e_{ij} = \xi \cdot P_j \cdot (d_{ij})^{-\delta}$$ (2)

where $\xi$ is a system coefficient related to the wavelength of signal, transmitter, and receiver antenna gains, etc. $P_j$ is the transmission power, $d_{ij}$ is the distance, and $\delta$ is an attenuation factor of the wireless channel, which is equal to 2 in a common environment in this study. Then, the TC will be run again after a fixed-time interval to update network topology, and lightweight messages can be transmitted via links within the topology.

In the SG process, the robot that firstly received the start signal sends messages to all other robots with as few hops (relays) as possible. Fewer hops mean higher success rate of transmission of lightweight messages, which can be represented by

$$S_{Tr} = (1 - P)^m$$ (1)

where $P$ is the failure probability of a hop, and $m$ is the number of hops. Assumed $P = 0.1$, the transmission success rate is 59% after five hops. So, the purpose of the TC is to minimize the number of hops when lightweight messages arrive in the last robot, and the process of which is shown in Algorithm 1. Here, $e_{ij}$ is the signal strength of robot $j$ received by robot $i$, which can be calculated by

$$e_{ij} = \xi \cdot P_j \cdot (d_{ij})^{-\delta}$$ (2)

where $\xi$ is a system coefficient related to the wavelength of signal, transmitter, and receiver antenna gains, etc. $P_j$ is the transmission power, $d_{ij}$ is the distance, and $\delta$ is an attenuation factor of the wireless channel, which is equal to 2 in a common environment in this study. Then, the TC will be run again after a fixed-time interval to update network topology, and lightweight messages can be transmitted via links within the topology.

In the SG process, the robot that firstly received the start signal becomes the first leader robot. To be noted, the leader robot is responsible for scheduling resources of the robots inside its subgroup, unlike traditional leader–follower strategy, for example, formation control.24 Other leader robots are selected according to their positions and communication conditions and then cover as many robots as possible while ensuring available communication links, as shown in Algorithm 2. The SG is run after the TC, which means that $R_{k}^{2.4}$ is known. The $E_5$ is much larger than $E_4$ because a larger transmission rate is required. For example, the execution time of object recognition in an image of 100 Kb is required to be less than 1 s. It can be completed within 0.5 s when computing resources in the cloud are used, so the time for transmission of the 100-Kb image is less than 0.5 s, which implies that a
transmission rate larger than 200 Kb/s is required. In addition, the distance corresponding to signal strength $E_{2.4}$ should be twice that of signal strength $E_{5}$ to achieve better coverage. According to Eq. (2), the relationship between $E_{2.4}$ and $E_{5}$ can be represented by

$$E_5 = \frac{E_{2.4}^{\lambda J P_j^{(s)}}}{E_{2.4}^{\lambda J P_j^{(s)}}}$$

(3)

where superscripts “(2.4)” and “(5)” indicate parameters from communication devices with 2.4 GHz and 5 GHz, respectively. The SG will be run again only when it is detected that changes of communication conditions and idle computing resources are greater than certain thresholds.

To be noted, all robots use the 2.4 GHz devices to communicate before subgroups are divided. After the hierarchical network is generated, the 5 GHz devices are used for communication between the second layer and the third layer (inside subgroups), and the 2.4 GHz devices are used for communication within the second layer (between leader robots), respectively, shown as Figure 1.

The resource scheduling algorithm

As soon as the hierarchical network is generated, the resource scheduling process is carried out. The resource scheduling algorithm is run on every leader robot to contribute available computing resources inside the subgroups to the intern-sufficient cloud, as shown in Figure 2.

Inside a subgroup, firstly, idle computing resources of the robots in $R_{ij}$ are reported to leader robot $i$. Due to the idle computing resources are not equal to the available computing resources, the resource scheduling algorithm is used to schedule the communication resources, already obtained in the process of generating hierarchical network, and the idle computing resources to enable idle computing resources to match appropriate transmission rate, which is achieved by reallocating bandwidth because the transmission rate is mostly determined by bandwidth. It has

$$T_j = B_j \cdot \log_2(1 + \frac{\mathcal{R}_{ij}}{\sigma^2 d_{ij}})$$

(4)

$$\text{s.t. } \mathcal{R}_{ij} = \frac{\mathcal{R}_{ij}}{\sigma^2 d_{ij}}$$

(5)

where $\mathcal{R}_{ij}$ is signal-to-noise ratio (SNR), and $B_j$ is the allocated bandwidth to robot $j$. After executing the resource scheduling algorithm, the reallocated bandwidth of robot $j$ can be represented by

$$B_j = \frac{E_{2.4}}{E_{2.4}} \frac{\mathcal{R}_{ij}}{\sigma^2 d_{ij}}$$

(6)

s.t. $B_j = \min\{T_j^{\text{required}}, T_j^{\text{only}}\}$

(7)

where $T_j^{\text{required}}$ is the transmission rate required to use all idle computing resources $C_{ij}$, and $T_j^{\text{only}}$ is the estimated transmission rate in the process of generating hierarchical network. The detailed process of reallocating bandwidth can be seen in our previous publication.23

At the system level, messages of available computing resources of all subgroups are shared among leader robots, which means the information of available computing resources of any robot is known by each robot, that is, available computing resources have been contributed to the cloud. At this time, if there is a computing requirement in a robot, available computing resources in the intern-sufficient cloud can be requested.

When robot $i$ prepares to use the intern-sufficient cloud, firstly, the computation task is supposed to be divided into $m$ sub-computations. Secondly, the required data of the sub-computations will be transmitted to appropriate robots and calculated. Finally, the sub-results will be reassembled on robot $i$. In the process, the computation division time in robot $i$ is $t_{dc}$, the data transmission time of a sub-computation for robot $j$ is $t_{ij}^{dc} = S_{ij}^{\text{data}} / T r_{ij}^{c}$, the data processing time in robot $i$ is $t_{pi}^{dc} = H(S_{ij}^{\text{data}})$ in which $Ht$ represents the time function of processing $S_{ij}^{\text{data}}$ with available computing resource of robot $j$, the data transmission time back to robot $i$ of a result is $t_{dr}^{ij}$, the recombination time in robot $i$ is $t_{cr}^{ij}$, and the time constraint of the task is assumed to be $t$. Finally, the execution time of robot $i$ needs to satisfy

$$H(S_{ij}^{\text{data}}) + \max_{j=1}^{n} \{t_{ij}^{dc}\} + \max_{j=1}^{n} \{t_{ij}^{dr}\} < t - t_{dc} - t_{cr} - t_0$$

(8)

The robots who share available computing resources need to satisfy e.g. robot $j$

$$H(S_{ij}^{\text{data}}) + t_{ij}^{dc} + t_{ij}^{dr} < t - t_{dc} - t_{cr} - t_0$$

(9)

where $t_0$ is the time for message queuing, retransmission, and other mechanisms.

Up to now, the whole intern-sufficient cloud is presented, by which the robots are allowed to share computing resources on demand while keeping stable connections.

Implementation of the multitarget navigation task

A multi-robot multitarget navigation task is applied on the intern-sufficient cloud. All of the robots are equipped with normal performance processors, electromagnetic compasses, infrared obstacle avoidance modules, and other cheap but necessary hardware. Some of them are equipped with cameras. During the navigation process, algorithms with high computational complexity, such as image processing, are executed on those robots equipped with cameras while relatively simple algorithms are run in other robots. It is assumed that extra computing resources are
needed by the robots equipped with cameras, while surplus computing resources are wasted in other robots.

In the task, a physical target and a virtual target are set, the positions of which are not known by the robots. At the beginning, the robots move randomly to detect and localize positions of the targets. Once the target positions are localized, the robots will be navigated to their target positions.

**Navigated to physical target**

The robots that are equipped with cameras are supposed to be navigated to the physical target. The navigation rule to the physical target is simple. After the camera perceives the target, the robot moves towards the target. An example is shown in Figure 3. Robot 6 is equipped with differential global positioning system (DGPS), which has high localization accuracy, but other robots can only use the camera for relative localization. Once the connected perception graph as shown in Figure 3 is formed, all robots can obtain absolute position information. In Figure 3, robots 2 and 6 have found the target, so all the robots will move towards the target from now on. In the process, the visual measurement algorithm should be run all the time. A single-camera can intuitively only measure angle information, and navigation using only bearing measurements is challenging.\(^\text{25}\)

Therefore, the height of the robot is used to calculate depth information

$$Z_c = \frac{h_{\text{cam}} K^{-1} \Theta}{n^2 K^{-1} \Theta}$$  \hspace{1cm} (10)

where \(K\) is camera internal parameters, and \(\Theta\) is bounding box information.

Obviously, an error estimation algorithm is indispensable to improve the accuracy of localization, which is essentially a data fusion algorithm.

The data fusion algorithm is based on extended Kalman filter, which includes two steps, propagation and update. The propagation process can be represented by

$$\ddot{x}^t = f(\dot{x}^{t-1}, u^{t-1})$$  \hspace{1cm} (11)

$$P_i^{t+1} = F_i^{t} P_i^{t-1} F_i^{t-1\top} + Q_i$$  \hspace{1cm} (12)

$$y^t = z^t - h(\dot{x}^t)$$  \hspace{1cm} (13)

$$S^t = H_i^t P_i^{t-1} (H_i^t)^\top + R_i$$  \hspace{1cm} (14)

The update process can be represented by

$$G^t = P_i^{t-1} (H_i^t)^\top (S_i^t)^{-1}$$  \hspace{1cm} (15)

$$\hat{x}^t = \ddot{x}^t + G^t \cdot y^t$$  \hspace{1cm} (16)

$$P_i^{t} = (I - G_i^t H_i^t) P_i^{t-1}$$  \hspace{1cm} (17)

The \(-, +\), and \(\hat{\cdot}\) mean an estimation, not optimal, and optimal, respectively. Here \(f\) represents the nonlinear function of state transition, \(h\) represents the nonlinear function of observation, \(t\) represents time step, \(x\) is the state, \(u\) is the control value, and \(z\) is the observation vector. The \(y\) is different between measurement and estimated output, and \(S\) is the covariance. Here, \(P\) is the covariance matrix of state-error, \(F\) is the Jacobian matrix of function \(f\) called system propagation matrix, and \(Q\) is the covariance matrix of process noise. And \(G\) is Kalman gain, \(H\) is the Jacobian matrix of function \(h\), that is, observation matrix, \(R\) is the covariance matrix of observation noise, and \(I\) is an identify matrix.

Obviously, extra computing resources are needed by these robots due to the object perception algorithm and the data fusion algorithm, while the communication resources are not in short supply because of that these robots communicate only when perception occurs.

**Navigated to virtual target**

The robots without cameras are supposed to be navigated to the virtual target, which can be assumed to be the position of the gas leakage source. These robots can sense the concentration of the smell, which is assumed to be normalized reciprocal of distance from the robot to the position of the virtual target. The navigation algorithm for these robots is based on partial swarm optimization, which is a simple version of our previous publication.\(^\text{26,27}\) The core equations are

$$\dot{p}_i^{t+1} = \omega \dot{p}_i^t + c_1 r_1 (x_{b_i} - x_i^t) + c_2 r_2 (x_g - x_i^t)$$  \hspace{1cm} (18)

$$p_i^{t+1} = p_i^t + \dot{p}_i^{t+1} \Delta T$$  \hspace{1cm} (19)
multi-robot system. Parameter \( x_i^g \) represents individual best position, which is obtained from moving experience of robot \( i \). Parameter \( x_g^g \) is the global best position, which is the position of the robot with the highest signal concentration in the multi-robot system. Parameter \( \omega \) is a self-learning factor, \( c_2 \) is a social factor, and \( r_1 \) and \( r_2 \) are two independent random parameters.

Obviously, idle computing resources are excessive in robots without cameras.

### Implementation on the intern-sufficient cloud

When the multitarget navigation task is carried out, the robots equipped with cameras will encounter the issue of communication congestion and lack of computing resources. It can be solved by the proposed intern-sufficient cloud. The process of using computing resources from the cloud is shown in Figure 4. Simultaneously, in step 1, the signal concentration is also transmitted to calculate the global best position, which is then informed to each robot in step 4. An example of the message format is shown in Figure 5.

Therefore, the key to implement the navigation task in the cloud is to make the task decomposable. In the process of being navigated to the physical target, the task of robot \( i \) is divided into six subtasks, which are object detection \( D_i^1 \) and components for data fusion \( x_i^1, G_i^1, y_i^1, H_i^1, P_i^{1i-1} \). Firstly, the robots report to their leader robots the requirements of computing resources. Secondly, a task allocation algorithm is run on the leader robots to match the required computing resources and the available computing resources. Thirdly, the leader robots inform the robots of their subgroup that calculations can be offloaded to the corresponding robots. Fourth and finally, the results \( x_i^1, P_i^0 \) will be reassembled on robot \( i \). In this process, a Hungarian algorithm-based task allocation algorithm is proposed. Under the constraints of Eqs. (16) and (17), the required computing resources of robot \( i \) is \( C_i^\text{required} \). Inside the subgroup, the available computing resources of robot \( j \) is \( C_j \). The matching process is shown in Algorithm 3. Here, \( \sum_{iKm} C_i^\text{required} + \cdots + C_i^\text{required} \) is a sum of required computing resources of \( q \) robots, where \( K_m, m \in \{1,2,\ldots,l\} \), is one of the \( l \) permutations, calculated by \( C_k \). This \( C \) is an equation of permutation and combination.

### Simulation

Two steps, that is, generating intern-sufficient cloud and using resources from it, are simulated sequentially to validate the effectiveness.

#### Parameters setting

In this case, the large-scale multi-robot system is composed of a hundred robots. These robots and two targets are randomly initialized in a two-dimensional square area (100 m \( \times \) 100 m). The bandwidth of the leader robot is assumed to be 40 MHz (20 MHz of 2.4 GHz device and 20 MHz of 5 GHz device). The assumed range of signal power (–90 dBm to –40 dBm) and noise power (–140 dBm to –110 dBm) is closed to the actual condition extremely, which ensures the SNR ranges from 20 dB to 100 dB.

Twenty-five robots inside the system are equipped with cameras. It is assumed that extra computing resources are needed by the robots equipped with cameras, whereas idle computing resources are excessive in the robots without cameras. Specifically, 30% of the processor has been taken up to guarantee the basic functions of the robot. Then, the idle computing resources of the robots without cameras are assumed to be 50% ± 10%. The computing resources needed by the robots equipped with cameras are assumed to be 120% ± 10%, which, together with the 30% of basic function, is a total of 150%. The ±10% is measurement error that follows Gaussian distribution. Overall, in the large-scale multi-robot system, the idle computing resources are 50% \( \times \) 75 = 3750%, whereas the extra computing resources needed by the robots equipped with cameras are 50% \( \times \) 25 = 1250%.
Effectiveness of the intern-sufficient cloud

The effectiveness of hierarchical network is firstly validated. Then, together with the resource scheduling algorithm, the effectiveness of the intern-sufficient cloud is validated.

Validation of the hierarchical network. The hierarchical network is constructed by TC and SG algorithms. Firstly, as long as an signal to construct the intern-sufficient cloud is received by a robot, TC is run on this robot. After the communication topology is constructed, a subgroup of the intern-sufficient cloud is generated. Then, via SG, the leader robot of this subgroup searches other candidate leader robots according to signal strength, and the selected leader robots generate their subgroups. Go on this loop. Finally, the hierarchical network is constructed. The process is shown in Figure 6.

Furthermore, simulations of the TC process are performed. The results are shown in Figure 7, in which six hops are the maximal in the system with a hundred robots, which can effectively ensure the transmission success rate of lightweight messages.

Algorithm 3. Task allocation

Input: \( R_i = \{R_{i1}, \ldots, R_{iK}\} \), set of robots that need computing resources. 
\( R_J = \{R_{J1}, \ldots, R_{JK}\} \), set of robots that share computing resources. 
\( K = 6 \), number of subtasks. 
\( R^s \), set of robots that have been matched computing tasks. 
\( C^a_i \), set of tasks that have been matched robots. 
\( C^T_i \), set of tasks of robot \( i \) to be offloaded. 

Output: \( G \), bipartite graph representing the mapping of task allocation.

1. Function Match \((C^\text{required}, R_J)\):
   2. \( R^a, C_i^a \leftarrow \text{Hungarian}\{C^\text{required}, R_J\} \)
   3. return \( R^a, C_i^a \)

4. while \( C^T_i \) is not empty do
   5. \( q = \text{len}(C^T_i), K = q \)
   6. // get the number of tasks to be allocated
   7. \( C_i^{\text{required}} = \{C_k^{\text{required}}_i \mid k \in [1, K_i]\} \)
   8. \( R^a, C_i^a \leftarrow \text{Match}\{\sum_{k=1}^{K_i} C_k^{\text{required}}_i \mid R_J\} \)
   9. \( R_J \leftarrow R_J - R^a \)
   10. \( C_i^T \leftarrow C_i^T - C_i^a \)
   11. while \( R^a \) is \( \emptyset \) do
      12. \( q = q - 1 \)
      13. for \( m \in C^a_i \) do
         14. \( R^a, C_i^a \leftarrow \text{Match}\{\sum_{k=1}^{K_i} C_k^{\text{required}}_i \mid R_J\} \)
         15. \( R_J \leftarrow R_J - R^a \)
         16. \( C_i^T \leftarrow C_i^T - C_i^a \)
      17. end
   18. end
   19. end

Figure 6. The process of constructing the hierarchical network.

Figure 7. The results of topology construction.
It can be easily concluded from Figures 6 and 7 that the hierarchical network can effectively connect the robots, in which SG is able to cover all robots and TC can minimize the number of hops.

**Validation of resource scheduling.** The size of the data to be processed on the intern-sufficient cloud is decided by available computing resources according to \( H_i(S_{\text{data}}^i) \), and the data \( S_{\text{data}}^i \) to be transmitted is constrained by communication conditions, that is, the use of the shared computing resources \( C_{\text{shared}}^i \) requires specific communication resources, called shared transmission rate \( T_{\text{shared}}^i \). Therefore, it is vital to reasonably schedule the communication resources and the computing resources. In this study, ratio \( T_{\text{shared}}^i / T_{\text{actual}}^i \) of the shared computing resources to the actual transmission rates is discussed, three kinds of which are proposed, such that

\[
U_1 = \max \left( \frac{\sum_{i=1}^{n} T_{\text{shared}}^i}{\sum_{i=1}^{n} T_{\text{actual}}^i} \right) 
\]

\[
U_2 = \frac{\sum_{i=1}^{m} T_{\text{shared}}^i}{\sum_{i=1}^{m} T_{\text{actual}}^i} 
\]

\[
U_3 = \min \left\{ \frac{T_{\text{shared}}^i}{T_{\text{actual}}^i} \right\}, i \in [1, \ldots, N] 
\]

\[
s.t. \ N = n + m 
\]

where \( N \), \( n \), and \( m \) are numbers of all robots, the robots that share all computing resources, and the robots that share

---

**Figure 8.** The ratio of shared transmission rate and actual transmission rate.

**Figure 9.** The distribution of the ratio.

**Figure 10.** The maximal available computing resources that contributed to the intern-sufficient cloud.

**Figure 11.** Simulation setup.
parts of computing resources, respectively. Then, simulation with 10,000 runs is performed, and the results are compared with round robin algorithm and max-SNR algorithm in the mobile cloud,\textsuperscript{28,29} as shown in Figures 8 and 9.

The $U_1$, $U_2$, and $U_3$ in Figure 8 are overall parameters, which show better performance of the proposed method in matching computing resources and communication resources. The distribution of the ratio in Figure 9 is detailed parameters of $U_1$, which further proves the effectiveness of the proposed algorithm.

These results are obtained under the condition that the idle computing resources are set to 3750\% (parameters setting). Next, how much available computing resources can be contributed is discussed. For more versatile, the idle computing resources within [2300\%, 4900\%] are simulated, as shown in Figure 10. It can be concluded that computing resources contributed to the mobile cloud can reach 1872\% in the specific case of idle computing resources of 3750\% and different resource scheduling algorithms lead to different available computing resources, and the proposed algorithm is the best.

**Effectiveness of the multitarget navigation**

From the data of the intern-sufficient cloud alone, computing resources of 1250\% can be easily supported when there are idle computing resources of 3750\%. However, the cloud can be proven effective only if it can adapt to the changes of spatial positions (corresponding communication conditions) caused by tasks. Thereby, a multitarget navigation task that applied on the cloud is simulated in this subsection. The scene of 100 m $\times$ 100 m square area with a hundred robots, a physical target, and a virtual target is shown in Figure 11.

The time step is assumed to be 1 s, the task is assumed to be divided into six parts (20\% is needed by each part), and the data (0.08 Mb) of each part should be transmitted to the appropriate robot within 0.1 s to meet Eq. (8), so the required transmission rate of the 25 robots can be calculated by $\frac{0.08 \text{ Mb}}{0.1 \text{ s}} \times 4 \times 25 = 80 \text{ Mb/s}$, in which “4” represents four parts should be transmitted and calculated on the intern-sufficient cloud. It also presents the relationship that sharing computing resources of 10\% needs 0.4 Mb/s. Therefore, the application of the navigation task is validated by whether the total transmission rates of 80 Mb/s can be guaranteed and whether the task requirements of 1250\% can be successfully calculated by the intern-sufficient cloud.

However, during the process of the task, there may not be enough available computing resources for each leader robot inside its subgroup. When the failure happens, the robots keep their former states and keep requesting

---

**Table 1. Part of data from the 1000 runs.**

| Cases | Completion time (s) | $F$ (%) |
|-------|---------------------|---------|
| 1     | 18                  | 4.6     |
| 2     | 17                  | 4.8     |
| 3     | 19                  | 4.6     |
| 4     | 31                  | 3.3     |
| 5     | 29                  | 2.6     |
| 6     | 26                  | 2.8     |
| 7     | 30                  | 5.4     |
| 8     | 20                  | 4.4     |
| 1000  | 23                  | 2.6     |
| Min.  | 16                  | 1.5     |
| Max.  | 34                  | 7.9     |
| Mean  | 25                  | 4.7     |

**Figure 12.** The multi-robot multitarget navigation process. See supplemental material “video_simulation.mp4”.

**Figure 13.** Distribution of the failure rates in the 1000 runs.
computing resources. At each time step, the CPU-take-up rate above 90% is regraded as a failure. Specifically, the failure rate can be represented by

$$F = \frac{\sum_{i=1}^{T} \sum_{j=1}^{N_i} f_n}{\sum_{i=1}^{T} \sum_{j=1}^{N_i} 1}$$  

(24)

where $T$ is total time steps, $N_i$ is the number of leader robots at time step $i$, and $f_n$ equals 1 if the CPU-take-up rate exceeds 90%, otherwise it is 0. For example, the dotted line in Figure 12 maybe a failure due to insufficient robots inside the subgroup.

In simulation, totally 1000 runs are executed under exactly the same initial setup of navigation and differently varying conditions of communication. A navigation process is shown in Figure 12. The time of completing this process is 18 s. The CPU-take-up rate above 90% occurred 85 times, that is, the failure rate is 4.6%. The results of total 1000 runs are presented in Table 1 and Figure 13.

It can be seen that the multitarget navigation task can be completed even if the cloud maybe ineffective in some time steps, that is, the failure cases only make the completion time longer. We can conclude that the multitarget navigation task can be successfully supported by the intern-sufficient cloud.

All in all, according to Figures 6 to 10, the proposed hierarchical network is constructed and then combined with the proposed resource scheduling algorithm to construct an
effective intern-sufficient cloud, in which the available links and the available computing resources are both guaranteed by the scheduled $T^\text{shared}_i$ and corresponding $C^\text{shared}_i$. Secondly, when executing a multitarget navigation task in a large-scale multi-robot system, the common problems of communication congestion and corresponding lack of computing resources are solved by deploying it on the intern-sufficient cloud, as presented in Table 1 and Figures 11 to 13. All the results prove that the computing resources can be simultaneously contributed to and used from the intern-sufficient cloud efficiently.

Experiment

The physical large-scale multi-robot system is shown in Figure 14, which is composed of 20 mobile robots. An individual robot is shown in Figure 15. The robot uses a Raspberry Pi 3 Model B running Linux OS fixed underneath of the plastic disc with a DGPS, a camera, and a compass on the top, to process the position and signal information and an STM32F407 microcontroller for motor control. The Raspberry Pi communicates with the STM32 controller via a serial port.

**Experimental parameters**

The operation system, RAM, and frequency of the Raspberry Pi 3 Model B are Raspbian 4.14, 1 GB, and 1.2 GHz, which shows a low-computing capability. The object detection algorithm running on the robots equipped with cameras is NanoNets after transfer learning. During the navigation process, the time step is set to 1 s as in simulation, the intern-sufficient cloud is updated per 5 s, and the information of idle computing resources and estimated transmission rate is updated every time step.

Next, the computing resources, that is, CPU and RAM, needed by the object detection algorithm, the data fusion algorithm, and particle swarm optimization (PSO)-based navigation algorithm are tested. The results after 100 runs are shown in Figure 16.

---

**Figure 17.** Initial state of the experiment.

**Figure 18.** The navigation process in the experiment. (a) Initial positions. (b) Main stages: robots are searching and moving to the respective targets. (c) Task completion. See supplemental material “video_experiment_using_the_proposed_method.mp4”.

---
Here, the PSO-based navigation algorithm, data fusion algorithm, and object detection algorithm are represented by A1, A2, and A3, respectively. The frames per second is 5 when the object detection algorithm runs. It can be seen that the required RAM is relatively stable and sufficient, while the required CPU is unstable and insufficient. Therefore, the required computing resource is represented by the CPU-take-up rate, which are medians, that is, A1 is 5%, A2 is 14%, and A3 is 63%. In addition, a safety factor of 10% is set, that is, if the estimated CPU-take-up rate is higher than 90%, the data fusion algorithm is required to be decomposed and then calculated on the intern-sufficient cloud.

The message to be transmitted shown in Figure 5 is about 0.5 Kb. So, the required transmission rate is defined to be 10 Kb/s, which also has a safety factor 2.

**Experimental settings**

The field experimental area is a square with a size of 10 m × 10 m, in which 20 robots, a physical target, and a virtual target are initialized in the area, see Figure 17.

Before the experiment, the robots are randomly initialized in the area, and target positions are the same as in the simulation. Then, the five robots use cameras to detect other robots, to localize themselves, and to discover the physical target, while the 15 robots will be navigated to the virtual target using distance as the signal strength. In this process, extra computing resources are needed by the robots equipped with cameras, but the computing resources of other robots are in surplus. So, the multi-robot navigation task cannot be completed without the intern-sufficient cloud.

**Effectiveness**

The effectiveness of the intern-sufficient cloud has been theoretically validated in the previous section. However, during the process of the task, there may not be enough available computing resources for each leader robot inside its subgroup. The robots will keep their former states if the
computing requirements cannot be offloaded successfully. The navigation process is shown in Figure 18, and the navigation results are shown in Figure 19.

It can be qualitatively concluded that the intern-sufficient cloud is effective since all robots have been navigated to their targets. The navigation task lasts 19 s, during which there are robots failed to offload computing requirements. The results are shown in Figure 20. The failure rate is 18.9%, higher than simulation 4.7%. There are two reasons for such a large gap. The first is the update process of the intern-sufficient cloud. At 5 s and 10 s, none of the five robots can offload their computing requirements successfully since the intern-sufficient cloud is updated per 5 s. At 15 s, four of the five robots have arrived in their target positions, that is, only one robot needs computing resources, and this robot still cannot offload its computing resources. Subtracting these failures, the failure rate is reduced to 7.4%. The second is because the CPU-take-up rate of the physical robot is unstable.

For comparison, the proposed communication mechanism is replaced by least common multiple (LCM). The results are shown in Figure 21. During this process, there are two robots equipped with cameras that have not even moved. Neither communication nor computing resources can be guaranteed under LCM.

Finally, it can be concluded that the proposed intern-sufficient cloud can provide stable resources of communication and computation for a multi-robot system with 20 robots while effectively achieving a multitarget navigation.

Conclusion
An intern-sufficient cloud for large-scale multi-robot systems is investigated in this study. On the one hand, it does not rely on developed cloud servers and communication infrastructures compared to traditional cloud methods. On the other hand, it maximizes available resources inside the robotic system from the perspective of performing tasks, which is more practical compared to traditional ad hoc cloud methods. To some extent, the intern-sufficient cloud has breakthrough the limitation of the number of robots in multi-robot applications in the real-world, as the simulation and the experiment proved. Specifically, the communication congestion can be alleviated by the hierarchical network, and a robot that needs computing resources is allowed to use available computing resources of the nearby robots. Nevertheless, the small experimental area (10 m × 10 m) results in generating only one subgroup. In the future, larger experimental areas and more complex tasks are the mean work in the next step.

Acknowledgement
All funding supports are highly appreciated.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work is supported by the projects of the National Natural Science Foundation of China (No.61873192; No.61603277; and No.61733001), the Quick Support Project (No.61403110321), and Innovative Project (No.20-163-00-TS-009-125-01). Meanwhile, this work is also partially supported by the Fundamental Research Funds for the Central Universities and the Youth 1000 program project. It is also partially sponsored by the International Joint Project Between Shanghai of China and the Baden-Württemberg of Germany (No. 19510711100) within the Shanghai Science and Technology Innovation Plan as well as the projects supported by the China Academy of Space Technology and Launch Vehicle Technology.

ORCID iD
Qirong Tang https://orcid.org/0000-0002-5541-4660

Supplemental material
Supplemental material for this article is available online.

References
1. Rizk Y, Awad M, and Tunstel EW. Cooperative heterogeneous multi-robot systems: a survey. ACM Comput Surv (CSUR) 2019; 52(2): 1–31.
2. Caley JA, Lawrance NR, and Hollinger GA. Deep learning of structured environments for robot search. Autonom Robot 2019; 43(7): 1695–1714.
3. Foster I, Zhao Y, Raicu I, et al. Cloud computing and grid computing 360-degree compared. In: 2008 Grid computing environments workshop, Austin, TX, USA, 12–16 November 2008.
4. Aguiar RL, Gomes D, Barraca JP, et al. Cloudthinking as an intelligent infrastructure for mobile robotics. Wirel Pers Commun 2014; 76(2): 231–244.
5. Turnbull L and Samanta B. Cloud robotics: formation control of a multi robot system utilizing cloud infrastructure. In: 2013 Proceedings of IEEE southeastcon, Jacksonville, FL, USA, 4–7 April 2013.
6. Du Z, He L, Chen Y, et al. Robot cloud: bridging the power of robotics and cloud computing. Future Gener Comput Syst 2016; 21(4): 301–312.
7. Zhu X, Qiu C, Deng F, et al. Cloud-based real-time outsourcing localization for a ground mobile robot in large-scale outdoor environments. J Field Robot 2017; 34(7): 1313–1331.
8. Arumugam R, Enti VR, Liu B, et al. DaVinci: a cloud computing framework for service robots. In: 2010 IEEE international conference on robotics and automation, Anchorage, AK, USA, 3–7 May 2010.
9. Waibel M, Beetz M, Civera J, et al. Roboearth. IEEE Robot Autom Mag 2011; 18(2): 69–82.
10. Chi F, Wang X, Wei C, et al. Ad hoc cloudlet based cooperative cloud gaming. IEEE Trans Cloud Comput 2018; 6(3): 625–639.
11. Gong Y, Chi Z, Fang Y, et al. Protecting location privacy for task allocation in ad hoc mobile cloud computing. IEEE Trans Emerg Top Comput 2019; 6(1): 110–121.
12. McGilvary GA, Barker A, and Atkinson M. Ad hoc cloud computing: from concept to realization. In: 2015 IEEE 8th international conference on cloud computing. New York, NY, USA, 27 June–2 July 2015.
13. Yaqoob I, Ahmed E, Gani A, et al. Mobile ad hoc cloud: a survey. Wirel Commun Mob Comput 2016; 16(16): 2572–2589.
14. Shila DM, Shen W, Cheng Y, et al. Amcloud: toward a secure autonomic mobile ad hoc cloud computing system. IEEE Wirel Commun 2017; 24(2): 74–81.
15. Yousafzai A, Chang V, Gani A, et al. Directory-based incentive management services for ad-hoc mobile clouds. Int J Inform Manage 2016; 36(6): 900–906.
16. Tang L, He S, and Li Q. Double-sided bidding mechanism for resource sharing in mobile cloud. IEEE Trans Veh Technol 2017; 66(2): 1798–1809.
17. Shi T, Yang M, Li X, et al. An energy-efficient scheduling scheme for time-constrained tasks in local mobile clouds. Pervasive Mob Comput 2016; 27: 90–105.
18. Yaqoob I, Ahmed E, Gani A, et al. Heterogeneity-aware task allocation in mobile ad hoc cloud. IEEE Access 2017; 5: 1779–1795.
19. Khalifa A, Azab M, and Eltoweissy M. Towards a mobile ad-hoc cloud management platform. In: 2014 IEEE/ACM 7th international conference on utility and cloud computing, London, UK, 8–11 December 2014.
20. Hartanto R and Eich M. Reliable, cloud-based communication for multi-robot systems. In: 2014 IEEE international conference on technologies for practical robot applications (TePRA), Woburn, MA, USA, 14–15 April 2014.
21. Roehr TM and Herfert S. A FIPA-based communication infrastructure for a reconfigurable multi-robot system, advances in intelligent systems and computing, volume 417. Cham: Springer, 2016.
22. Yong D and Yu X. Multi-robot system based on cloud platform. In: 2016 IEEE Chinese guidance, navigation and control conference (CGNCC), Nanjing, China, 12–14 August 2016.
23. Tang Q, Zhang J, Yu F, et al. A resource management algorithm for real-time response of mobile ad hoc cloud in swarm robotic system. In: 2018 IEEE international conference on robotics and biomimetics, Kuala Lumpur, Malaysia, 12–15 December 2018.
24. Chen L, Mei J, Li C, et al. Distributed leader-follower affine formation maneuver control for high-order multiagent systems. IEEE Trans Autom Control 2020; 65(11): 4941–4948.
25. Chen L, Cao M, and Li C. Angle rigidity and its usage to stabilize multi-agent formations in 2d. IEEE Trans Autom Control 2020: 1–15. DOI:10.1109/TAC.2020.3025539.
26. Tang Q, Yu F, Zhang Y, et al. A stigmergetic method based on vector pheromone for target search with swarm robots. J Exp Theor Artif Intell 2019; 32(3): 1–23.
27. Tang Q and Eberhard P. Cooperative search by combining simulated and real robots in a swarm under the view of multibody system dynamics. Adv Mech Eng 2013; 5: 284782.
28. Hahne EL. Round-robin scheduling for max-min fairness in data networks. IEEE J Sel Areas Commun 1991; 9(7): 1024–1039.
29. Letaief KB and Zhang YJ. Dynamic multiuser resource allocation and adaptation for wireless systems. IEEE Wirel Commun 2006; 13(4): 38–47.
30. Huang AS, Olson E, and Moore DC. Lcm: lightweight communications and marshalling. In: IEEE/RSJ international conference on intelligent robots and systems, Taipei, Taiwan, 18–22 October 2010, pp. 4057–4062.