aerospace

MDPI

Article

Application and Performance Evaluation of Resource Pool Architecture in Satellite Edge Computing

Junxiang Qin, Xiye Guo, Xiaotian Ma, Xuan Li and Jun Yang *

College of Intelligence Science and Technology, The National University of Defense Technology, Changsha 410000, China
* Correspondence: junyang0628@163.com

Abstract: Satellites will play a vital role in the future of the global Internet of Things (IoT); however, the resource shortage is the biggest limiting factor in the regional task of massiveequipment in the IoT for satellite service. Compared with the traditional isolated mode of satellite resources, the current research aims to realize resource sharing through satellite cooperation in satellite edge computing, to solve the problems of limited resources and low service quality of a single satellite. We propose a satellite resource pool architecture-oriented regional task in satellite edge computing. Different from fixed servers in ground systems, the satellite orbital motion brings challenges to the construction of the satellite resource pool. After the capacity planning of the satellite resource pool for regional tasks is given, an algorithm based on search matching is proposed to solve the dynamic satellite selection problem. A ground semi-physical simulation system is built to perform experiments and evaluate the performance of three modes of satellite resource sharing: isolated mode, cooperative mode, and pooled mode. The results show that the pooled mode, compared with the isolated mode, improves the task success rate by 19.52%, and at the same time increases network resources and energy consumption in the same scenario. Compared with the cooperation mode, the performance of task success rate and resource utilization rate is close to that of the pooled mode, but it has more advantages in response time and load balancing of satellite resources. This shows that in the IoT, the resource pool is of great benefit as it improves the task response time and improves the load balance of satellite resources without degrading the performance, which makes sense in task-demanding scenarios.

Keywords: satellite edge computing; satellite resource pool; resources sharing; capacity planning; performance evaluation; load balance

Citation: Qin, J.; Guo, X.; Ma, X.; Li, X.; Yang, J. Application and Performance Evaluation of Resource Pool Architecture in Satellite Edge Computing. *Aerospace* **2022**, *9*, 451. https://doi.org/10.3390/aerospace9080451

Academic Editor: Danil Ivanov

Received: 25 June 2022
Accepted: 15 August 2022
Published: 17 August 2022

1. Introduction

The space-based information system (SBIS) is a satellite application system that uses remote sensing, navigation, positioning, and communication to realize real-time observation of natural landscapes and human activities on the earth's surface [1]. The SBIS was originally built for independent functions, such as navigation constellations, communication constellations, etc. Satellites also have some functions in early development, such as navigation satellites and communication satellites [2]. The development of technologies, such as software-defined satellites, has made it possible for satellites to provide more services [3]. As a hotspot of current construction, a real-time, space-based information service system is a multifunctional space-based system composed of hundreds of low earth orbit (LEO) satellites with remote sensing, navigation, and communication functions [4]. With the rapid development of traditional and commercial space, many kinds of space-based system plans, such as Star-Link, OneWeb, XingYun, and other space-based systems are in the construction stage, promoting the development of the SBIS to involve integration, networking, and intelligence [5]. With the coming of the Internet of Things era, satellite is becoming increasingly important and is providing services to increasing numbers of people and terminals [6].

However, traditional satellite functions cannot be changed, nor can the resource of satellites be shared. This severely limits the type and quantity of tasks that satellites can provide. An increasing number of ground terminals need satellites to provide resource services which aggravates this problem in the IoT, especially in task-demanding scenarios, such as combat. Recent research has shown that satellite functions can be defined by software and demonstrates how these satellites are interconnected through the network. In this case, satellites have the opportunity to cooperate with others to provide flexible services. Researchers have previously attempted to apply cloud computing or edge computing technology to satellites [7–15]. The authors of Refs. [7–9] propose the application of edge computing technology on satellites to realize satellite resource sharing, whereas Refs. [10,11] propose the use of LEO satellites as edge nodes to realize network resource sharing. The authors of Refs. [12–15], respectively, study application deployment, server placement, network slicing, and migration in satellite cooperation and resource sharing. These studies discuss resource sharing from the perspective of users and tasks, which may lead to an excessive workload for some satellites. Resource pool architecture based on edge computing is a good method to solve the above problems. Liu [16] and Zhu [17] discussed the resource pool in the satellite-ground cooperation task, but both the resource pool and cloud are on the ground, with servers fixed in the computer room. The difficulty of establishing a resource pool in satellites is that the satellites move in orbit, and the satellite network topology will change dynamically, which will make the satellite resource pool unstable. The key point is to solve the problem of resource pool capacity in satellite resource sharing and dynamic satellite selection in order to establish the resource pool.

Traditional satellite resources are used in isolated mode. A further model of resource sharing is to cooperate to complete tasks after networking. The resource pool is a kind of resource organization paradigm based on resource sharing, which can effectively improve resource utilization. At the same time, the service life of the system is improved to some extent by realizing the load balance of system resources [18]. This paper evaluates these three types of resource organization paradigms.

The contributions of this paper are as follows:

- We firstly propose a satellite resource pool architecture oriented for regional tasks in satellite edge computing, which aims to integrate multiple resources of satellites in the SBIS into the resource pool for task service in the IoT.
- We then solve the capacity planning of the resource pool using queuing theory and propose a search matching algorithm to solve the instability problem of satellite resource pools.
- Through the queuing model and a hardware-in-the-loop simulation, the performances of isolated mode, cooperative mode, and pooled mode in the same scenario are theoretically and experimentally evaluated.

This paper is organized as follows. Section 2 introduces current research on satellite edge computing and satellite resource pools. Section 3 introduces the system model and the architecture proposed in this paper, including resource sharing mode, capacity planning, and satellite node selection. Section 4 first introduces the experimental platform and the experimental scenario, and then analyzes and evaluates the experiments that have been carried out. Finally, Section 5 gives the conclusion of this paper, describes the limitations of this study, and offers future research directions.

2. Related Works

2.1. Satellite Edge Computing

In recent years, cloud and edge computing have been gradually introduced into space-based systems. At present, the research mainly focuses on the system architecture or the key technology in the application of edge computing to satellites.

Regarding the architecture, Zhang et al. propose satellite mobile edge computing (SMEC), which integrates network resources through dynamic network virtualization technology, realizes satellite resource sharing, and improves QoS based on collaborative

computing and collaboration technology [7]. Denby et al. define and characterize orbital edge computing, which is similar to satellite edge computing (SEC) [8]. Han et al. combine mobile edge computing with satellite edge computing. They propose a satellite edge computing framework which realizes the integration of scattered satellite computing power and provides sufficient computing power for auxiliary processing of satellite big data and time-sensitive tasks [9]. Wang et al. use LEO satellites as edge computing nodes to establish an edge computing satellite cooperation network in an emergency and combine the software-defined network model to guide inter-satellite link connection and resource sharing [10]. Li et al. propose a system architecture for edge computing in LEO networks and implement an LEC prototype system to verify their design [11].

In addition to the system architecture, other research focuses on solving the problem of the application of edge computing on satellites. Li et al. are the first to propose a novel online service placement algorithm by leveraging the Lyapunov optimization and Gibbs sampling [12]. Yan et al. focus on the edge server placement problem, that is, where the edge servers should be placed in LEO satellite networks. However, the key problem is that each satellite itself is an edge server. According to their placement algorithm, satellites that are not placed on edge servers will not work [13]. Taeyeoun et al. analyze the architecture of satellite edge computing and propose a network slice scheduling for IoT support [14]. Zhang et al. discuss the offloading problem when artificial intelligence applications are learned and trained in orbit, but they skip the fact that artificial intelligence applications require a lot of computing resources, which means that such applications may not be able to run on satellites [15].

These researchers are studying how to apply edge computing to satellites, or how to realize resource sharing through the satellite cooperation. However, they do not consider that the difference between satellites and ground servers lies in the strict limitation and mobility of satellite resources, nor do they optimize resource sharing from the perspective of the whole system.

2.2. Satellite Resource Pool

At present, resource pool technology is mainly used in the fields of the internet, cloud computing, fog computing, edge computing, etc. As a new application scenario, the space-based network has not been involved and has no on-orbit application [4]. Alessandro is the first to define the paradigm of federated satellite systems (FSS) and create a resource pool for missions. However, he only discusses the FSS paradigm and its economic assessments and does not give the implementation method [18].

After the popularity of the concept of software-defined satellites, and increasing research on satellite cloud/edge computing as described in Section 2.1, some research has began to involve resource pools. A space-based cloud computing architecture, combining cloud computing technology with the space-based information network, is proposed in [19], and it is verified that satellite resource sharing can save the transmission bandwidth of satellite links and improve real-time performance of time-sensitive service processing. Its cloud is assumed to be a large satellite in high orbit, but so far there is not a satellite with such abundant resources, so its application is limited.

Wang et al. introduce task migration technology into the space-based system, and effectively prove that task migration can reduce task delay compared with the traditional space-based system through scene simulation [4]. Only the benefits brought by task migration are discussed, not the selection of satellites for migration, and task migration is not the construction of the resource pool.

Liu et al. use user-oriented virtualization technology in satellite ground stations and implement resource sharing based on processor virtualization and software-defined networking. It shows that the resource pool is built based on virtualization technology, but resource sharing is implemented on the ground station rather than on the satellite [16].

Zhu et al. investigate the cooperative transmission and resource allocation in cloud-based integrated terrestrial-satellite networks. This paper verifies that through satellite

ground cooperation, satellites can use the resources from the resource pool on the ground. Without further consideration, satellites, like computers, can also cooperate to establish resource pools in space [17].

To provide more resources for on-orbit applications, it is still necessary to study the technology of establishing resource pools on satellites. As there is no resource pool mode on satellites, this paper aims to implement this mode and compares it with the isolated mode and the cooperative mode [19].

3. Proposed Architecture

3.1. Satellite Resource Pool Architecture

The resource pool is generally a collection of the various hardware and software involved in the cloud computing data center. In [4], the resources on the satellite onboard processing board are virtualized into a resource pool. In space-based systems, satellite resources are scattered and heterogeneous. This not only increases the difficulty of satellite resource management, but is also not conducive to improving the success rate of the task. This is especially true when there are massive devices that need a large number of resources and there are many kinds of tasks in the IoT, such as artificial intelligence applications and target recognition, for instance. To solve these problems, this paper proposes a satellite resource pool model based on edge computing for target area tasks, as shown in Figure 1.

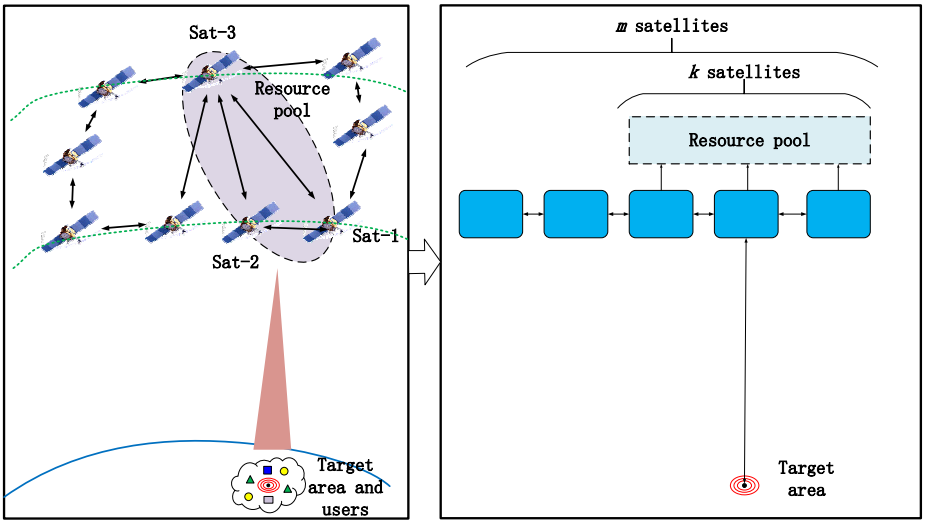

Figure 1. Model of satellite resource pool architecture oriented for regional task.

The satellite resource pool is a virtual resource pool based on satellites in space-based systems. As shown in Figure 1, *m* satellite networks need to be interconnected in a distributed satellite cluster through an inter-satellite link. Following this, for the task service in the target area, *k* satellites share the virtualized resources of satellites through the distributed cloud network operating system and build a resource pool. The management node in the resource pool uniformly schedules the satellite resources in the resource pool according to the resource requirements of the tasks in the region. There will be several resource pools in the whole space-based system to serve different regions. Resource pools will cooperate through management nodes to realize resource sharing among resource pools and even the whole space-based system.

Each satellite of the resource pool first virtualizes its resources into general computing, storage, and sensing resources through resource virtualization technology. Similar to ground virtualization technology, it adopts the resource virtualization scheme based on kernel-based virtual machine to virtualize the computing, storage, and sensing resources. Each type of resource establishes a sub-pool, such as a computing pool and a sensor pool, and then establishes its resource directory and index. This provides a unified resource

directory and general index method, so that the logical resource can be mapped to the satellite actually sharing the resource through the resource directory and index.

The resource pool is established for task requirements. The satellites in the space-based system are in the whole network, but not all satellites will join a resource pool, which is determined according to the requirements of the resource pool and the dynamic state of the satellite. The dynamic states of the satellite include the orbital motion of the satellite, the change of the satellite network topology, the satellite's task execution, and the consumption of satellite resources, and so on. There is an optimal problem regarding the capacity of the resource pool, which will be described in Section 3.2, and the satellite selection will be discussed in Section 3.3.

Before the emergence of satellite networks, satellites were used in isolated utilization modes, as shown in Figure 2a. In this mode, satellites work independently and do not cooperate with each other. Moreover, due to the satellite orbit movement, the coverage area is limited, and only some users can be served in a certain time window. With the development of the inter-satellite link, satellites are gradually networked, and space-based networks are established. At this stage, satellites interact with each other to complete tasks together, as shown in Figure 2b. This model, similar to the model in [19], promotes cooperation among satellites and improves the utilization efficiency of satellite resources. However, due to the limited resources of a single satellite, some tasks with a large demand for resources cannot be effectively completed on the satellite. This is why we proposed the pooled model, as shown in Figure 2c. For users in the target region, several satellites can be equivalent to a large satellite with a common resource pool, and all satellite resources of the resource pool can be used.

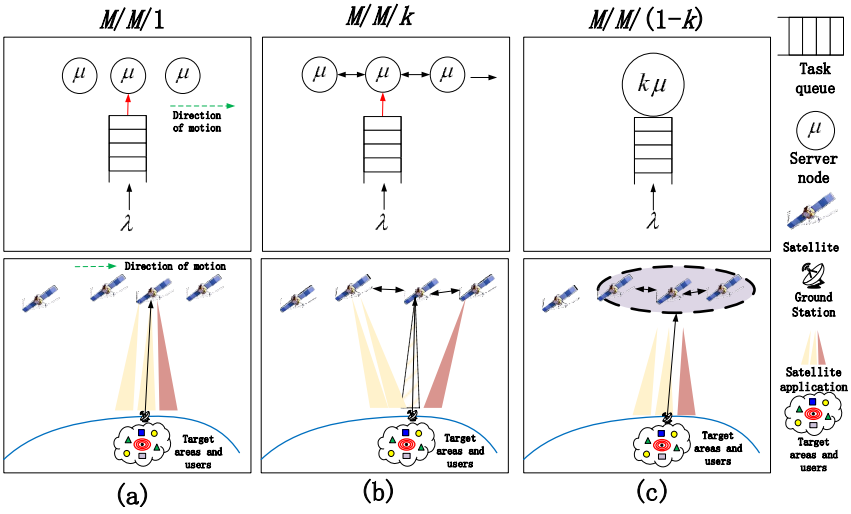

Figure 2. Modes of satellite resource sharing showing (**a**) isolated, (**b**) cooperative, and (**c**) pooled.

It is worth noting that the tasks of the satellite are not necessarily sent by the ground station, but also generated by the satellite itself. However, traditional satellites, especially corresponding to the isolated mode, are generally transmitted by the ground station. Therefore, other models make the same assumptions. Based on this, the satellite in independent utilization mode is equivalent to the classical $M/M/1$ queuing system. In cooperative mode, tasks are released to a satellite through a channel, and then the satellite services the tasks. In this mode, when a task comes, if there are idle satellites, the task will be distributed to them for execution. In the pooled mode, k satellites build a unified computing cluster, which is equivalent to a server with larger resources and faster service.

The notations frequently used in this paper are summarized in Table 1.

Table 1. The notations are used in this paper.

Notation	Definition
N	the number of satellite nodes
M	the number of satellite resource types
K	the number of satellite tasks
k_{α}^{*}	the minimum number of satellites in the pool
T_{sat_i}	the task set of satelite i
R_i^k	the i resource requirement of the task k
rc_i^k	the computing requirement of the task k in satellite i
rs_i^k	the storage requirement of task k in satellite i
d_i^{send}, $d_i^{receive}$	the input and output file sizes describing task i
$T_{i,j}^{w\_off}$	task sending waiting time from satellite i to j
$T_{i,j}^{trans}$	task transmission time from satellite i to j
$T_{i,j}^{w\_que}$	task waiting time in the execution queue from satellite i to j
$T_{i,j}^{sat}$	task execution time from satellite i to j
$T_{i,j}^{w\_back}$	result return time from satellite i to j
$E_{i,j}^{trans}$	the average transmission energy consumption of the task from satellite to j
$E_{i,j}^{sat}$	the average execution energy consumption of the task from satellite i to j
$C_i^{(s)}$	the number of CPU cycles of satellite i
v_m	The resource utilization of satellite resource m
v	The average utilization rate of m resources
d	The measurement method of resource load balance
$R_{S_i}^{*}$	* = {cpu,mem,sen}, the * resources of the satellite i, respectively,
$Link_{n_k}^{S_{ij}}$	the connection link relationship between the satellite i and the satellite j during the time period n_k
$Sync_{n_k}^{S_{ij}}$	the relationship between the two satellites i and j in the time period n_k

3.2. Pool Capacity Planning

Too many satellites participating in the resource pool will result in a waste of resources, while too few will affect the service quality of the task. The participation of satellites over long distances in building resource pools will also bring greater network delay. Thus, in order to meet certain task objectives, it is necessary to have an appropriate number of satellites, which brings the problem of capacity planning of the satellite resource pool.

Assume that there are k satellites participating in the target area service. Each satellite is a server, and the satellite task execution queue is represented by the M/M/1 queuing system, that is, the time interval of task generation obeys the exponential distribution, and the task execution time obeys the arbitrary probability distribution. It is assumed that the rate at which the ground station generates tasks is λ, that is, the time interval of task generation is an independent, identically distributed random variable, subject to exponential distribution. The service rate of each satellite is μ, which represents the number of tasks completed by the satellite in unit time.

The task requirements of the target area determine the number of satellites to be involved. Given the target area and task type, we can estimate the number of satellites needed for service. We can then solve the number of satellites based on the square root configuration rule [20]. Considering that the goal is to keep the average response time of the system to less than α, and the minimum number of servers required to achieve such quality of service is k_{α}^{*}, it is obtained from the square root configuration rule:

$$k_{\alpha}^{*} \approx \rho + \varepsilon\sqrt{\rho} \tag{1}$$

The ρ is defined by $\rho = \frac{\lambda}{\mu}$. The ε is given by (2), and $\Phi(\varepsilon)$ represents the standard normal cumulative distribution function and its probability density function.

$$\frac{\varepsilon\Phi(\varepsilon)}{\phi(\varepsilon)} = \frac{1-\alpha}{\alpha} \tag{2}$$

The author of Ref. [21] gives the empirical value tables of ε and α. When α is 0.2 and $\varepsilon \approx 1$. This means that when the average response time is less than 20%, the minimum number of servers that can be solved is $k_\alpha^* \approx R + \sqrt{R}$.

After determining the number of satellite nodes, the problem remains that the satellite is different from the ground computer room. As the satellite is in space orbit, as shown in Figure 3, the motion of the satellite is not considered in the queuing theory model. Therefore, one of the difficulties of creating a satellite resource pool is to select the suitable satellites from all satellites. Satellite resource sharing in the resource pool will be further described below, and a satellite node selection method will be proposed.

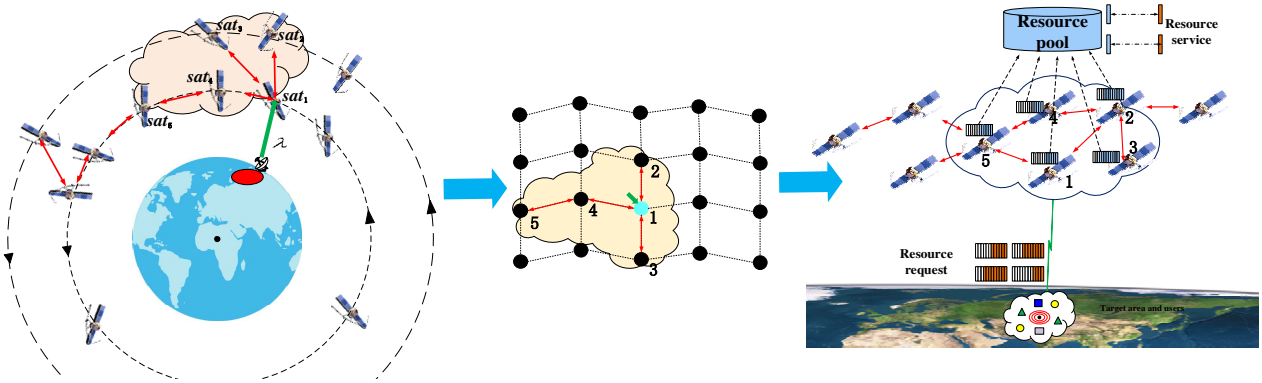

Figure 3. Process of satellite resource pool service regional tasks.

3.3. Satellite Node Selection

Assume that there are N satellites in the space-based system, and M resources in each satellite. The dynamic task set of a satellite sat_i is defined as $T_{sat_i} = \{task_k | 1 \le k \le K\}$, and expression of the task is $task_k = \left\{T_i^k, R_i^k\right\}$. T_i^k is the execution time window of the task k, which corresponds to the task target area and $T_i^k = \langle t_{start}, t_{end} \rangle$. R_i^k refers to the resource requirements of the task. The tasks generated by satellite i can be expressed by the required resources and the data size of the task itself, i.e., $R_i^k = \left\{rc_i^k, rs_i^k, d_i^{send}, d_i^{receive}\right\}$, where rc_i^k indicates the number of computing resources required to perform the task, which can be quantified by the number of CPU cycles $C_i^{(s)}$. rs_i^k is the storage requirement. $d_i^{send}, d_i^{receive}$ represent the input and output file sizes, respectively, describing task information such as program code and response data.

An advantage of using the pooled mode over the cooperative mode is that it can pool more resources for tasks, especially for tasks with large resource requirements [22]. It can dynamically split the tasks and distribute them to the satellites with idle resources in the resource pool for execution, realizing satellite resource sharing [23]. The problem of satellite selection can be described as selecting k satellites from the whole space-based system to share resources. In Figure 3, satellites one to five are used to build the resource pool.

3.3.1. Average Task Response Time

When satellite i migrates some tasks to satellite j for execution, the following three steps need to be completed in the whole process: sending the task to satellite j, satellite j executes the task, and then returns the task result to satellite i. Under the resource pool, the response time of task $T_{i,j}$ is composed of five parts: task sending waiting time $T_{i,j}^{w\_off}$, task transmission time $T_{i,j}^{trans}$, task waiting time in the execution queue $T_{i,j}^{w\_que}$, task execution time $T_{i,j}^{sat}$, and result return time $T_{i,j}^{w\_back}$. Some tasks may not return results. In this case, the time and energy consumption of returning results are zero. Local processing does not require data transmission. Among them, the satellite transmits data through the inter-satellite link, and the task transmission of the inter-satellite link includes link establishment delay and transmission delay. The chain establishment delay can be divided into tracking

delay, synchronization delay, and handshake delay. Taking the common microwave inter-satellite link as an example, the two sides of satellite communication need to communicate. Firstly, the link must be built under conditions which ensure the visibility of the satellite. As shown in Figure 4 below, the visible condition of the satellite means that the connection between the two satellites does not pass through the earth, that is $\Delta h \succ 0$.

$$\varphi = \arccos\left(\frac{(r_1 \bullet r_2)}{|r_1| \bullet |r_2|}\right) \tag{3}$$

$$\Delta h = h - R_e = \frac{|r_1| \bullet |r_2| \bullet sin\varphi}{|r_2 - r_1|} \tag{4}$$

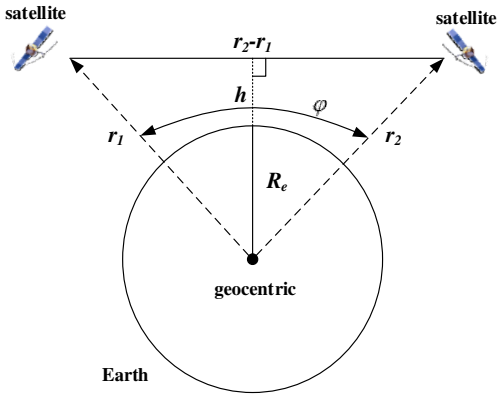

Figure 4. Satellite visibility relationship.

In the process of establishing the link between two satellites, the antenna must be aligned first. However, since the satellite ephemeris is known, future satellite links can be predicted, so the tracking delay can generally be regarded as a stable value. The synchronization delay is related to the coding mode of the inter-satellite link, generally QPSK, which can also be regarded as a stable value. The handshake delay is mainly caused by propagation. Assuming that the distance between two satellites is $|r_2 - r_1|$ and the handshake is m times during chain establishment, c is the speed of light, and the handshake delay is:

$$t_{\text{hand}} = m\frac{|r_2 - r_1|}{c} \tag{5}$$

Assuming that the inter-satellite link transmission rate is $R_{i,j}^{send}$, the expression of $T_{i,j}^{trans}$ is obtained:

$$T_{i,j}^{trans} = \frac{d_j^{send}}{R_{i,j}} + t_{hand} \tag{6}$$

Expression for $T_{i,j}^{sat}$ is:

$$T_{i,j}^{sat} = \frac{rc_j}{C_j^{(s)}} \tag{7}$$

During the visible window time between the satellite and the ground, the satellite cluster network topology constituting the resource pool can be regarded as unchanged [24]. At this time, the task sending waiting time and result return waiting time are only related to the message sending queue, which will be counted by the task queue in subsequent experiments. To sum up, the response time $T_{i,j}$ of the task migrating to satellite j is:

$$\begin{aligned} T_{i,j} &= T_{i,j}^{w\_off} + T_{i,j}^{trans} + T_{i,j}^{w\_que} + T_{i,j}^{sat} + T_{i,j}^{w\_back} \\ &= T_{i,j}^{w\_off} + T_{i,j}^{w\_que} + \frac{d_j^{send}}{R_{i,j}^{send}} + n^{hop} m\frac{|r_2 - r_1|}{c} + \frac{rc_j}{C_j^{(s)}} + \frac{d_j^{receive}}{R_{i,j}^{receive}} \end{aligned} \tag{8}$$

In particular, when the task is processed locally, data transmission is not required, so the response time of the task $T_{i,i}$ only consists of two parts: $T_{i,i}^{w\_que}$, the waiting time of the task in the execution queue, and $T_{i,i}^{local}$, the execution time of the task. Furthermore,$T_{i,i}^{wait}$ equals to the $T_{i,i}^{w\_que}$ for the local task processing. To sum up, the average response time of tasks performed locally on the satellite is:

$$T_{i,i} = T_{i,i}^{w\_que} + T_{i,i}^{local} = T_{i,i}^{wait} + \frac{rc_i}{C_i^{(s)}} \tag{9}$$

3.3.2. Energy Consumption

Following this, the average energy consumption $E_{i,j}$ of task execution in the resource pool mode will be calculated, which is composed of two parts: the average transmission energy consumption of the task $E_{i,j}^{trans}$, and the is the average execution energy consumption of the task $E_{i,j}^{sat}$. According to the parameters of the above communication model, the average transmission energy consumption of the task can be expressed as:

$$E_{i,j} = E_{i,j}^{trans} + E_{i,j}^{sat} = \frac{p_i d^{send}}{R_{i,j}} + \kappa \left(C_j^{(s)}\right)^2 rc_j \tag{10}$$

$E_{i,j}^{sat}$ means that the energy consumption of the processor is proportional to the square of the CPU frequency. p_i is the energy consumed by transmitting each unit file, and κ in the $E_{i,j}^{sat}$ represents the equivalent switching capacitance, which is related to the architecture of the chip [25]. When the task is processed locally, there is no transmission energy consumption, i.e., $E_{i,j}^{trans} = 0$.

3.3.3. Load Balancing Rate of Resource Nodes

One benefit of the resource pool is that it can avoid overusing resources by some domestic satellites while other overseas satellites are idle, that is, to achieve resource load balance. Load balance is evaluated by resource utilization. The resource utilization of a certain kind of satellite refers to the time ratio of resource m on satellite j during the completion time of a task, which is defined as follows:

$$\nu_m = \frac{\sum\limits_j tend_j - tstart_j}{T_j} \tag{11}$$

In (11), $tend_j$ is the time when task j finishes executing on resource m, $tstart_j$ is the time when task j starts executing on resource m, and T_j is the time difference between the time when task j finally finishes the task and the time when task j first starts executing. The calculation is as follows:

$$T_j = \max_{0 \le j \le n} \{tend_j\} - \min_{0 \le j \le n} \{tstart_j\} \tag{12}$$

There are various resources on satellites, including computing, storage, network, sensor, and other resources. The average utilization rate of m resources is:

$$\nu = \frac{\sum\limits_i^m \nu_i}{m} \tag{13}$$

In this paper, the load balance degree is used to evaluate the load between satellites. It means that the same resources should be allocated to each satellite as much as possible, rather than being used too intensively on one satellite [26]. Resource load balance can ensure the efficient execution of tasks, improve the utilization of resources, and avoid

excessive consumption of a single node, thus prolonging the service life of satellites. The measurement method of resource load balance is:

$$d = \sqrt{\frac{\sum_{i}^{m} (\nu - \nu_i)^2}{m}} \tag{14}$$

The closer d is to zero, the smaller the difference in the operation of each resource, and the better the resource load balance.

3.3.4. Planning Objectives

The response time of tasks is critical for users, especially in situations where real-time requirements are high. Waiting for the response for a long time will lose users, and there will be great losses in major cases such as natural disasters. According to the above description, for the task-oriented resource pool node selection, this paper focuses on the goal of minimizing the average response time of the task and maximizing the resource load balance of the node (that is, minimizing the d), which are modelled by (15) and (16), respectively. K is the number of tasks.

$$U_1 = \min \int_{t_0}^{T} \sum_{1}^{K} T_i \Big/ K \tag{15}$$

$$U_2 = \min(\sum_{1}^{M} d_i \Big/ M) \tag{16}$$

There are many research constraints for satellites to complete tasks submitted by ground users. First, the resources required by the task shall not exceed those owned by the satellite. Expressed as follows:

$$\begin{aligned} \sum_{1}^{K_i} R_{S_{i,k}}^{cpu} &\leq R_{S_i}^{cpu}; \\ \sum_{1}^{K_i} R_{S_{i,k}}^{mem} &\leq R_{S_i}^{mem}; \end{aligned} \tag{17}$$

where $R_{S_i}^{cpu}$, $R_{S_i}^{mem}$ represent computing and storage resources of the satellite i, respectively. This means that the total computing and storage resources occupied by the K_i tasks executed on each satellite cannot exceed the resources owned by that satellite.

It is assumed that n_k is a time slot. For the completion of a specific task, a responsive sensor of the corresponding satellite is required. Although satellites can reconfigure their software, they cannot perform tasks without corresponding hardware support. Therefore, the following constraints shall be met:

$$R_{s_i,k}^{sen}(n_k) \cap R_{S_i}^{sen} \neq \varnothing \tag{18}$$

The $R_{s_{i,k}}^{sen}(n_k)$ indicates that in time slot n_k, the sensor occupied by task k on satellite i is consistent with the sensor of satellite i. Service performances are constrained by the resolution, field angle, and orbital space position of the sensor. In this paper, only orbital factor is considered. The field angle of the sensor is taken as a fixed parameter, and the initial angle of the sensor is temporarily ignored. It is uniformly set to be aligned with the geocentric. Visibility is constrained by (3) and (4).

When considering communication constraints, communication must first satisfy the visibility between satellite and ground and between the satellites. Communication resources then need to consider the number of satellite links and communication bandwidth. It is expressed as follows:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} (Link_{n_k}^{S_{ij}}) \leq ck_{com} \tag{19}$$

$$\sum_{i=1}^{N} (Link_{n_k}^{S_{ij}}) \leq \min\left\{ \mathrm{B}_{S_i}^{band}, \mathrm{B}_{S_j}^{band} \right\} \tag{20}$$

The $Link_{n_k}^{S_{ij}}$ is the connection link relationship between satellite i and satellite j during the time period n_k. A value of one means that the two satellites establish an inter- satellite link. If it is zero, there is no communication. ck_{com} is the number of communication channels of the satellite, which limits the number of links that the satellite can establish. $\mathrm{B}_{S_i}^{band}$ is the communication bandwidth of the satellite i. The total bandwidth used in the time slot of task execution cannot exceed the total bandwidth of the satellite.

Synchronization constraints are largely limited to a resource conflict with observation and other attributes in the task to avoid resource waste or shortage. For example, the navigation function needs at least four satellites, and some remote sensing tasks need multiple coverage to meet the accuracy requirements of the service.

$$SYNC_{sig}^{\min} \leq \sum_{i=1}^{N} \sum_{j=1}^{N} (Sync_{n_k}^{S_{ij}}) \leq SYNC_{sig}^{\max} \tag{21}$$

The $Sync_{n_k}^{S_{ij}}$ represents the relationship between the two satellites i and j in the time period n_k, and $SYNC_{sig}^{\min}$ and $SYNC_{sig}^{\max}$, respectively, represent the minimum and maximum time relationship required for signal processing [27]. Generally, only the minimum relationship represented by $SYNC_{sig}^{\min}$ needs to be considered, but the load balancing rate of resources will be considered at the same time to maintain the resource balance between space-based systems, so it is constrained by $SYNC_{sig}^{\max}$ to avoid taking up too many satellites.

3.3.5. Selection Algorithm

The response time is mainly considered in the task-oriented resource pool node selection problem. The constraints include resource constraints, communication constraints, and synchronization constraints. To solve the above problem, the paper adopts a search matching algorithm, which can comprehensively consider three constraints and planning objectives, and finally select the target node group according to the priority of planning objectives. The purpose of the resource pool is to serve the task. As satellite tasks have the highest time requirements, the average response time of the task is given priority in the selection of nodes [27]. At the same time, the resources of the satellite are strictly limited. Due to the characteristics of satellite orbit movement, it is easy to cause the centralized use of resources at some nodes. Therefore, it is necessary to consider the load balance of resource nodes to prolong the service life of the satellite. Based on the above considerations, the task-oriented resource pool node selection algorithm in this paper is shown in Algorithm 1.

A satellite orbit parameter file is used to obtain the position of satellites. The sort operation in row 11 is sorted by decreasing order. R_{flag} is the flag of resource demand satisfaction. In this algorithm it is four, which represents four resource marks: CPU, storage, communication, and sensor. Following this, the operation in the selection algorithm aims to select k satellites according to the target ranking of response time and load balance. The process is: checkout (15) to satisfy the basis of meeting the response time, then calculate (16) using the (14) to get the resource balance. Finally, output the k satellites matrix. It can be seen from the algorithm that the goals are to complete the task, improve the response speed, and ensure the load balance of the SBIS, to promise service quality for users and improve the service life of the satellite. However, it can also be found that not all tasks can respond successfully, and the failure rate of tasks will be further analyzed in the experiment.

Algorithm 1 Satellite node selection of resource pool

Input: λ, N, M, K, Task file, Satellite orbit parameter file
Output: k_α^* satellites matrix $S_{k_\alpha^*}$
Initialization: task resource requirements, $t_e = \max\{tend_i | 1 \le i \le N_K\}$
Read the satellite resource status matrix; read satellite orbit position
Obtained from (1) k_α^*, $L = 0$
For $i = 1$ to t_e
    For $j = 1$ to M_k
        calculate $R_{S_{i,m}}^{sen}(N_k), R_{S_{i,m}}^{cpu}(N_k), R_{S_{i,m}}^{mem}(N_k)$
        $R_{S_{i,m}} = R_{S_{i,m}}^{sen}(N_k) + R_{S_{i,m}}^{sen}(N_k) + R_{S_{i,m}}^{sen}(N_k)$
            If $\left|R_{S_{i,m}}\right| \ge 1$
                checkout (19), (20), update $R'_{S_{i,m}}$
                If $\left|R'_{S_{i,m}}\right| \ge 1$
                    Sort $R'_{S_{i,m}}$
                End if
            End if
        End for
        Count $num = \left\{R'_{S_{i,m}} \middle| R'_{S_{i,m}}(t_i) = R_{flag}\right\}$
        If $num \succ k_\alpha^*$
            $S_{k_\alpha^*} = \left\{RT_S^m | 1 \le i \le k_\alpha^*\right\}$
        Else
            $S_{k_\alpha^*} = \left\{R'_{S_{i,m}} | 1 \le i \le num\right\}$
    End if
End for

After the satellite node selection algorithm gives the satellites that construct the resource pool, the first one will be selected as the management node. The reason for this is that the ranking is based on the matching degree of satellite resources and tasks. The first one is therefore the best in terms of capability, and choosing the first one can provide the best service in the selected satellite. During the task period, the node selection algorithm will be executed once for each time slot, but the node selection will be executed only after the selection node changes to avoid the node being in constant change.

4. Experiment and Analysis

4.1. Experimental Platform

In order to complete the verification and performance evaluation of the isolated, cooperative, and pooled modes, the hardware in the loop simulation experimental platform shown in Figure 5 was designed. The cooperative mode is based on [19], and the pooled mode is based on the algorithm proposed in Algorithm 1. The system is mainly composed of a space-based system simulation driving platform and three demonstration sample satellites. The space-based system simulation driving platform is used to visualize the operation of aerospace scenes [28] and simulate cloud and edge computing [29]. Three satellites are configured to match with the satellites of the space-based system and to verify the software function and performance of the satellites. The space-based system simulation driving platform constructs a constellation composed of 442 satellites [30], which can simulate the space-time and network relations of the constellation in orbit, as well as task initiation, analysis, and function execution. Satellites initiate the task in those with an orbit of 500–1150 km, corresponding to the 'device count' of the later experiment. The 'device count' is equal to the number of devices to send tasks, and the more the devices launch tasks, the more load the space-based system will receive. All 442 satellites are used in the experiment in Table 2 and the statistical data in the experiment are also the satellites in the whole constellation.

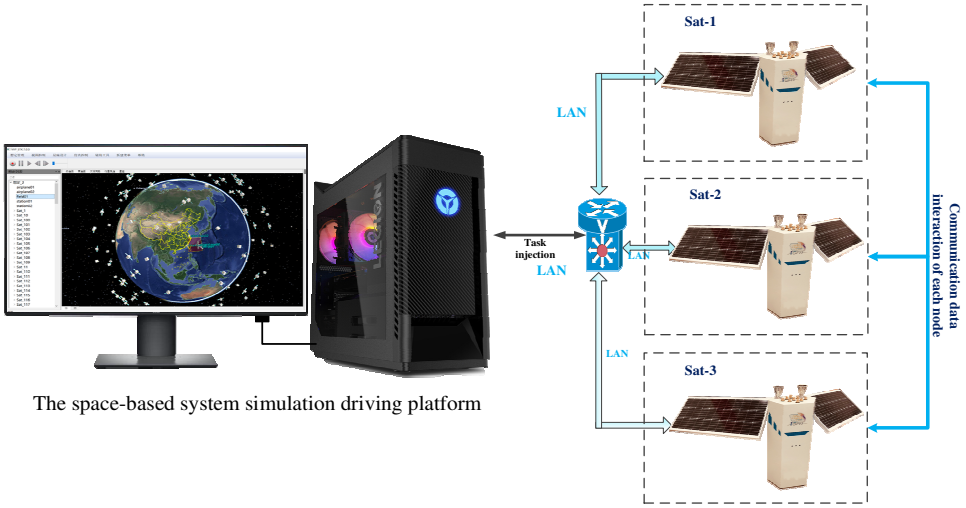

Figure 5. Hardware in the loop simulation experimental platform.

Table 2. The parameter of satellite constellation.

Satellite Height (km)	Orbit Inclination (°)	Constellation Parameter (Walker)
3500	0	6/1/0
3600	60	12/4/2
2150	56	24/6/2
1150	53	160/32/2
1110	53.8	40/5/2
1130	74	40/8/2
550	45	160/32/2

The shape and internal hardware topology of the sample demonstration satellite (Sat_x) are shown in Figure 6. The main part is the satellite body, including the main storage module, interface module, power module, general processor, microwave link module, camera array, and the SDR of the satellite. The camera array can be used for detection and the SDR can be used for reconfigurable RF signal function. The general processor adopts the raspberry pi which has been verified in orbit, and the microwave module is used to simulate the data transmission of satellite telemetry and remote control. The sample demonstration satellite is a platform to complete the task demonstration. It is mapped to one of the satellites in the constellation and used to verify the feasibility of the software. The applications developed in the experiment include *App1*, *App2*, and *App3*. The sample demonstration satellites can be configured as required according to the task requirements to execute the corresponding applications. The configuration of each application is shown in Table 3. The task resource requirements file mainly includes resource types, and the size of each type of resource requirement is shown in Table 3. The resource type of each task and application is fixed, but the value changes randomly within its specified range for each task. The mapping of tasks and application types is judged according to the sensing resources. After reading the task resource requirements file, the platform will establish a one-to-one mapping between the task and the software to complete the mapping from task to resource.

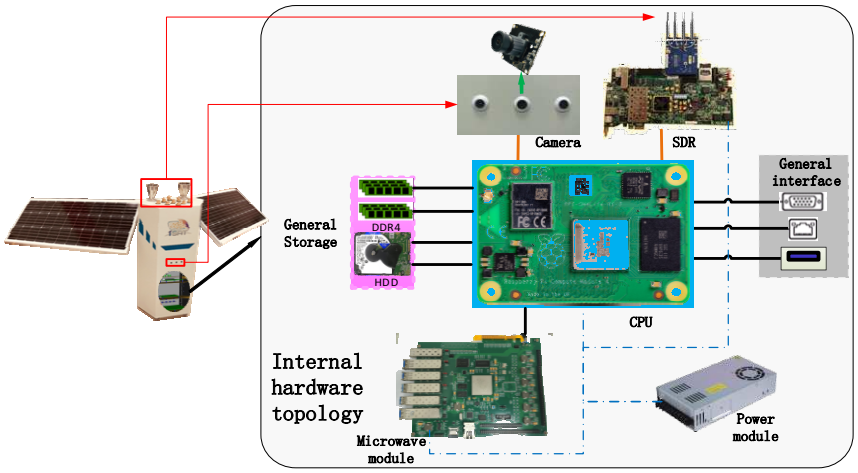

Figure 6. Structure diagram of sample demonstration satellite system.

Table 3. Application resource configuration table.

	App1	*App2*	*App3*
Max. delay (s)	5	5	200
Container size (KB)	2500	9000	13,000
Task length (MI)	60,000	15,000	200,000
Required core (PE)	3	2	2
Required memory (MB)	100	500	300
Required sensor	SDR	camera	Microwave
Poission interarrival (s)	40	10	30
remarks	No target area	Target area	Visible to another satellite

4.2. Results Comparison

4.2.1. Task Response Time

Firstly, the hardware in the loop simulation experiments of the three modes is carried out in Figure 2. In the isolated mode, the task is only submitted to the satellite which is visible to the ground, and the task is not migrated. When the satellite leaves the visible window, the task of *App2* fails because it leaves the target area. The task of *App1* is unaffected and ends when the execution time is finished. In the cooperative mode, the ground station submits a task to a satellite visible to the ground. When the resources of one satellite are exhausted, that satellite sends a task request to other satellites and selects the nearest satellite to execute it. In the pooled mode, k_α^* satellites are calculated according to (1), and the resources of k_α^* satellites are used together. At the same time, with the satellite movement, satellites will join or exit to maintain the balance of the resource pool.

The results of the average response time of tasks in the three modes on our platform are shown in Figure 7. It can be seen that the response time increases with the number of tasks under the three modes. In the isolated mode, the average response time of tasks is approximately 9.28 s (device count: 60–500). Even when the number of tasks is small, the response time is too long because tasks can only be processed locally, which is approximately 11.49 s (device count: 1–60). In the pooled mode and cooperative mode, when the number of tasks is small, they can be quickly executed without waiting too long, and the time is approximately 6.63 s (device count: 1–60). It is worth noting that the average waiting time drops after the device count increases beyond 180. This is because the task success rate has increased, and due to delay the task failure rate decreases, as can be seen in Figures 9 and 11. In contrast, there are fewer waiting tasks in the queue, and therefore the waiting time is decreased. The average response time of the last three modes tends to be the same, because when the workload reaches the cluster load, it enters a balanced state.

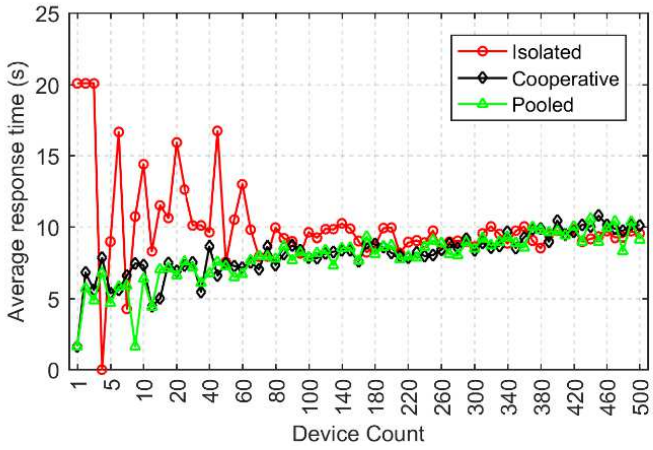

Figure 7. Average response time of tasks.

The results of the average waiting time and task success rate are shown in Figures 8 and 9. It can be seen that in the isolated mode, the average waiting time of tasks is 17.84 s (device count: 60–500). Even when the number of tasks is small, the waiting time is long because the required resources must wait for local processing. In pooled mode and cooperative mode, when the number of tasks is small, they can be executed quickly without waiting for too long. However, when the number of tasks becomes larger, the waiting time becomes approximately twice the average waiting time of the isolated mode due to the time spent waiting for the task to be sent and received. We first make it clear that the task success rate refers to the ratio of the number of successfully executed tasks to the total number of initiated tasks in the simulation time. In terms of task success rate, it can be seen from Figure 9 that through resource sharing, both the pooled mode and the cooperative mode have greatly improved the task success rate, from 26.21% of the isolated mode to 45.73%, increased by 19.52%. This is because resource sharing results in more tasks being assigned to resources, while tasks have to wait for resources in the isolated mode, which improves resource utilization and task success rate. When there are few tasks, the task success rate changes greatly. This is because there are few tasks and the resource demand is met easily. However, due to visibility and satellite movement, some tasks will fail, which makes the proportion of task failure change greatly when there are few tasks.

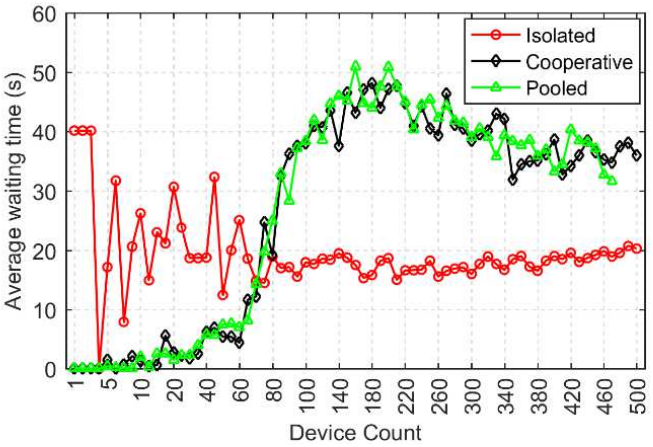

Figure 8. Average waiting time of tasks.

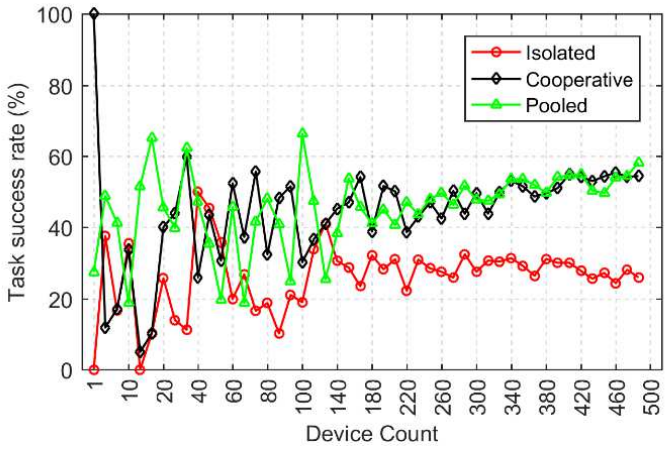

Figure 9. Task success rate.

4.2.2. Task Failure Rate

Compared with the task success rate, the task failure rate is the ratio of the number of failed tasks to the total number of initiated tasks in the simulation time. Before analyzing the task failure rate, it should be emphasized that the process of mapping the relationship between the three demonstration satellites and the satellites in the simulation platform is: select one in the orbit of 3600 km, one in the orbit of 1100 km, and one in the orbit of 550 km. At the same time, because the visible window of 550 km orbit to the earth is very small, the sample demonstration satellite mapped with 550 km orbit changes periodically, while the other two are fixed in mapping. This is set to ensure the authenticity and randomness of the data. This paper counts the task failure rate in instances of failure caused by mobility, insufficient resources, and unsatisfied delay.

The results of the task success rate can be seen in Figure 9, where it is shown that pooled and cooperative modes greatly improve the task success rate. We analyze the causes of task failure through Figures 10–12. As can be seen from Figure 10, pooled mode and cooperative mode can share resources, which solves the problem that arises when tasks fail without resources. The failure rate is reduced from 51.45% (peak at 75.00%) in isolated mode to 1.64% (peak at 8.57%) in both pooled and cooperative mode. However, resource sharing brings other problems. The first is in terms of delay. Both pooled mode and cooperative mode need to transfer information with other satellites, migrate tasks, and send information and executable software, which will cause delays. Tasks with high real-time requirements may lead to task failure. Therefore, in the pooled mode and cooperative mode, the task failure caused by delay is higher than that in the isolated mode. The task failure caused by delay in isolated mode is stable at approximately 25.19% in Figure 11, because it is only processed locally, and the task queue brings delay. The reason for the decrease in task failure rate in pooled mode and cooperative mode is that, when there are few tasks, the task planning algorithm pursues the resource load balance of the system and allocates tasks in the resource pool as much as possible, which will make some tasks fail with a probability of 57.40%.

In addition to the two issues mentioned above that may cause task failure, mobility is also a key factor in satellites. Task failure due to mobility refers to the satellite leaving the target area due to the orbital motion of the satellite, or that it is invisible to other satellites. Figure 12 shows that the proportion of this part is small, up to only 3%. Interestingly, cooperative and pooled modes have higher failure rates caused by mobility than the isolated mode. The reason for analyzing the data is that the probability of delay and mutual invisibility caused by data transmission in resource sharing has increased, but the value is still very small. In the isolated mode, the task failure rate is relatively low because the task is only sent to the satellite visible to the target, and the task is only executed locally.

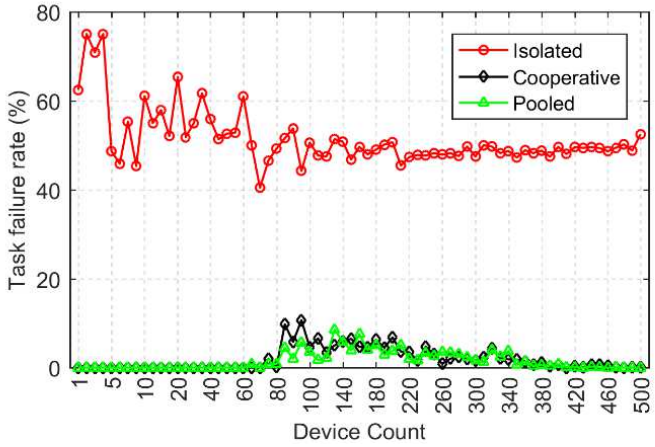

Figure 10. Task failure rate due to lack of resources.

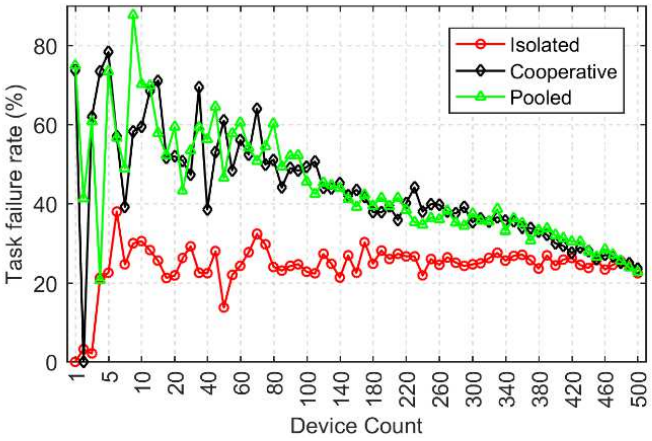

Figure 11. Task failure rate due to delay.

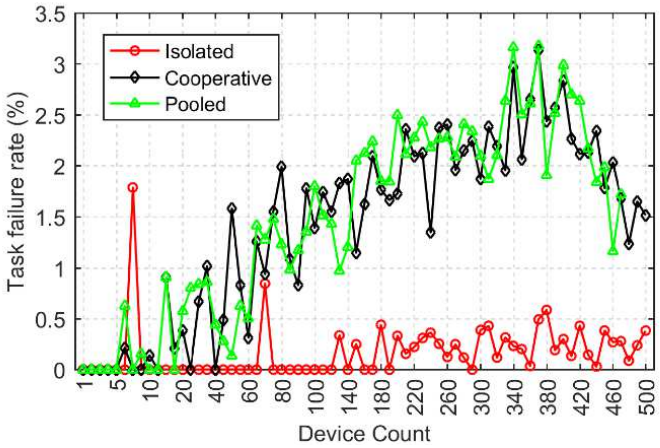

Figure 12. Task failure rate due to mobility.

4.2.3. Resource Utilization

The paper records the CPU utilization rate, energy utilization rate, and network utilization rate under each mode, as shown in Figures 13–15, respectively. As can be seen in Figure 13, whether in the cooperative mode, pooled mode, or isolated mode, the utilization rate of the CPU resource will increase with the increase in devices and tasks. The CPU usage in pooled mode is significantly higher than that of isolated mode; this is because pooled mode performs more tasks. To a certain extent, the CPU resource utilization in the two modes tends to be balanced, indicating that the resource utilization boundary of the

system has been reached. In the isolated mode, the utilization rate of the CPU resource is 37.41%, while it can reach 70.68% in the pooled mode and cooperative mode, which is nearly 33.27% higher.

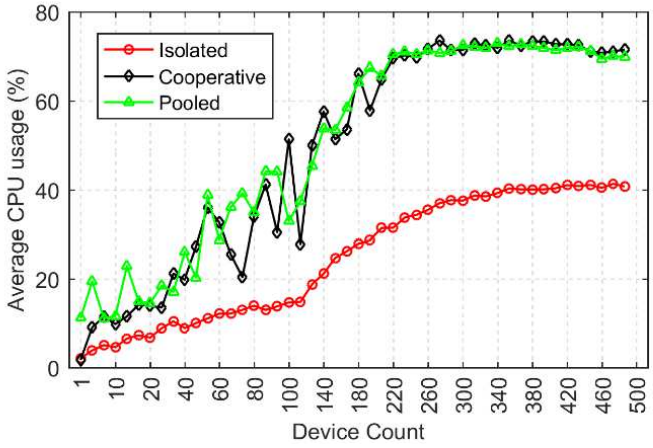

Figure 13. Average CPU usage of satellite.

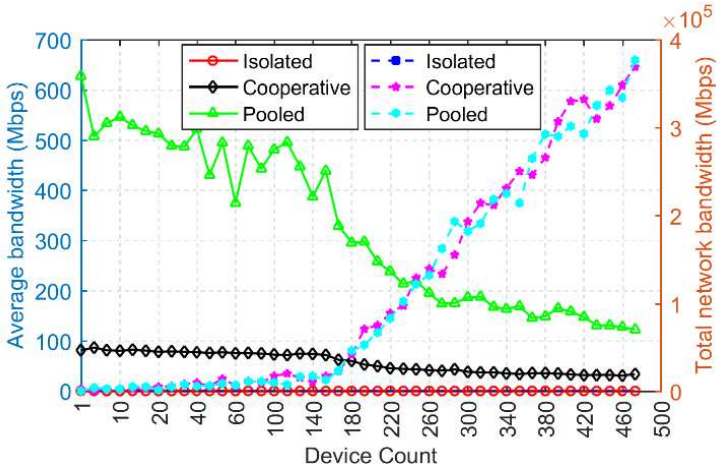

Figure 14. Average bandwidth per task and total network usage.

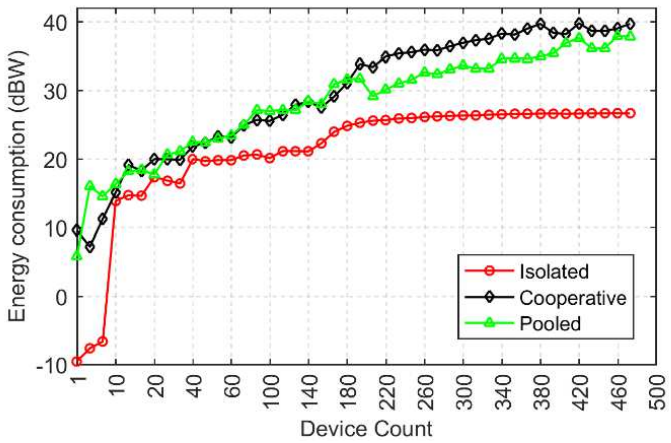

Figure 15. Average energy consumption per satellite.

The usage of network resources is shown in Figure 14. The graph counts the average bandwidth allocated to the task and the network resource usage in each experiment. The average allocated bandwidth of tasks decreases with the increase in tasks, but the average

bandwidth of tasks is relatively balanced in pooled mode. This is undertaken by the pooled mode to maintain the load balance of resources in the pool at all times. The load balance is more significant than the cooperative mode, because the cooperative mode is equal to a cold load balancing mode. Resource sharing starts after more than half of a satellite's resources are used, which is also the most obvious difference between the cooperative and pooled modes. The use of network resources in pooled and cooperative modes increases with the increase in tasks, and the two modes are relatively close. The reason why the isolated mode consumes fewer network resources is that there is no resource sharing between satellites, and information is not transmitted between them. In the pooled and cooperative modes, when the number of tasks is small, the network resources are less used, because the resources are still abundant and there is no need for a large amount of information transmission and task migration. With the increase in the number of tasks, the network resources are rapidly increasing due to the need to continuously transmit information, share resources, and complete tasks.

The energy usage is shown in Figure 15. It can be observed that when the task volume is low, the network resource usage of the pooled mode is higher than that of the cooperative mode, because of the system overhead required to maintain the resource pool. In the three modes, the more tasks, the larger the average energy consumption. However, when it is increased to 60 devices in isolated mode, it will not increase. At this point, the peak of task execution in this mode has been reached. As can be seen in (10), energy usage includes data transmission and processing tasks. In isolated mode, there is no data transmission, so the average energy consumption is lower than that in the other two modes. The pooled mode and cooperative mode continue to slow down due to the increased energy consumption caused by data transmission, from 26.12% to 39.88% (cooperative) and 37.82% (pooled). The reason why the energy consumption in pooled mode is higher than in cooperative mode is due to the energy consumed when maintaining the resource pool. For satellites, it is worthwhile to improve the task success rate brought by energy consumption and network usage. As the satellite can be charged with solar energy, it is very cost-effective to consume energy to obtain better service without affecting the health of the satellite.

From the above results, it can be seen that the pooled mode improves the success rate of tasks while maintaining the basic service performance compared with the cooperative mode. It is critical in scenarios that require high task success rates. If the resource is short and the task is not urgent, switching the working mode is possible. The working mode should depend on the needs of the task, rather than remain unchanged, to get the best balance.

5. Conclusions

This paper solves the problems of resource pool establishment and performance evaluation in satellite edge computing. When a resource pool is introduced into satellites, the resource pool is unstable due to the orbital motion of satellites and the dynamic change of network topology. Using the predictability of satellite orbit and a search matching algorithm, this paper solves the dynamic satellite selection problem. The queuing theory is used to evaluate the performance of the traditional isolation mode, cooperation mode, and resource pool mode. Compared with the isolated mode, the task success rate of the pooled mode can be increased from 26.21% to 45.73%, and the response time can be accelerated by 19.52% in the case of fewer tasks. The response time is consistent when the number of tasks is high. In terms of resource utilization, the CPU resource utilization of satellites can be improved by 33.27%, and the network usage is significantly increased. However, it also increases the energy consumption, from nearly 26.12% to approximately 37.82%. Compared with the cooperative mode, the average task waiting time, task failure rate, and resource utilization performance are close to those of the pooled mode, but there are two improvements. Firstly, the response of the pooled mode is faster and more stable in the case of a small workload. Secondly, in terms of network and CPU resource utilization, the resource utilization among satellites is more balanced in the pooled mode.

Therefore, compared with the isolated mode of the traditional satellite, both cooperative mode and pooled mode not only improve the task success rate, but also improve resource utilization at the cost of consuming more energy. However, compared with the cooperative mode, the pooled mode can improve the response time of tasks and improve the load balance of system resources. The pooled mode has a wide application prospect in the battlefield and other scenarios with high task requirements. In future work, the effect of the space-based system network and other factors on the resource pool will be studied to optimize the architecture, such as autonomous decision-making methods for maintaining service performance and communication resource consumption. Further consideration will also be given to issues such as data security in resource sharing in the resource pool.

Author Contributions: Conceptialization, J.Y. and J.Q.; Methodology, X.G.; Software, X.L.; Validation, J.Q. and X.M.; Formal Analysis, J.Q.; Writing, J.Q.; Supervision, J.Y. and X.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, S.; Zhang, Y.; Li, D.; Wang, L.; Wang, K.; Xu, Y. Current status and future prospects of space-based information systems in China. *China Eng. Sci.* **2020**, *22*, 6.
2. Min, S.; Liu, G.; Chen, B.; Liu, J. *Space-Ground Integrated Information Network*; Publishing House of Electronics Industry: Beijing, China, 2020; pp. 23–25, 147–152, 174.
3. Wu, Q. Research status and technology development prospects of software definition satellite. *J. China Acad. Electron. Sci.* **2021**, *16*, 333–337.
4. Wang, Y.; Yang, J.; Guo, X.; Qu, Z. Satellite Edge Computing for the Internet of Things in Aerospace. *Sensors* **2019**, *19*, 4375. [CrossRef] [PubMed]
5. Boley, A.C.; Byers, M. Satellite mega-constellations create risks in Low Earth Orbit, the atomospere and on Earth. *Sci. Rep.* **2021**, *11*, 10642. [CrossRef]
6. He, Y.; Chen, Y.; Lu, J.; Chen, C.; Wu, G. Scheduling multiple agile earth observation satellites with and edge computing framework and a constructive heuristic algorithm. *J. Syst. Archit.* **2019**, *95*, 55–66. [CrossRef]
7. Zhang, Z.; Zhang, W.; Tseng, F.-H. Satellite Mobile Edge Computing: Improving QoS of High-Speed Satellite-Terrestrial Networks Using Edge Computing Techniques. *IEEE Netw.* **2019**, *33*, 70–76. [CrossRef]
8. Denby, B.; Lucia, B. Orbital Edge Computing: Machine Inference in Space. *IEEE Comput. Achitecture Lett.* **2019**, *18*, 59–62. [CrossRef]
9. Han, J.; Wang, H.; Wu, S.; Wei, J.; Yan, L. Task Scheduling of high Dynamic Edge Cluster in Satellite Edge Computing. In Proceedings of the 2020 IEEE World Congress on Service, Beijing, China, 18–23 October 2020; pp. 287–293.
10. Wang, F.; Jiang, D.; Qi, S.; Qiao, C.; Shi, L. A Dynamic Resource Scheduling Scheme in Edge Computing Satellite Networks. *Mob. Netw. Appl.* **2020**, *26*, 597–608. [CrossRef]
11. Li, C.; Zhang, Y.; Xie, R.; Hao, X.; Huang, T. Integrating Edge Computing into Low Earth Orbit Satellite Networks: Architecture and Prototype. *IEEE Access* **2021**, *9*, 39126–39137. [CrossRef]
12. Li, Q.; Wang, S.; Ma, X.; Sun, Q.; Wang, H.; Cao, S.; Yang, F. Service Coverage for Satellite Edge Computing. *IEEE Internet Things J.* **2021**, *9*, 695–705. [CrossRef]
13. Yan, Z.; de Cola, T.; Zhao, K.; Li, W.; Du, S.; Yang, H. Exploiting Edge Computing in Internet of Space Things Networks: Dynamic and Static Server Placement. In Proceedings of the 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), Virtual, 27–28 September 2021; pp. 1–6.
14. Kim, T.; Kwak, J.; Choi, J.P. Satellite edge computing architecture and network slice scheduling for IoT support. *IEEE Internet Things J.* **2021**, *9*, 14938–14951. [CrossRef]
15. Zhang, F.; Chen, X.; Yu, S.; Ji, M.; Liu, Y.; Cao, L. On orbit intelligent technology based on space-based edge computing. *Shanghai Aerosp.* **2021**, *38*, 19–23.
16. Liu, Y.; Chen, Y.; Jiao, Y.; Ma, H.; Wu, T. A Shared Satellite Ground Station Using User-Oriented Virtualization Technology. *IEEE Access* **2020**, *8*, 63923–63934. [CrossRef]

17. Zhu, X.; Jiang, C.; Kuang, L.; Zhao, Z.; Guo, S. Two-Layer Game Based Resource Allocation in Cloud Based Integrated Terrestrial-Satellite Networks. *IEEE Trans. Cogn. Commun. Netw.* **2020**, *6*, 509–522. [CrossRef]
18. Golkar, A.; Ignasi, L. The Federated satellite Systems paradigm: Concept and business case evaluation. *Acta Astronaut.* **2015**, *111*, 230–248. [CrossRef]
19. Cao, S.; Zhao, Y.; Wei, J.; Yang, S.; Han, H.; Sun, X.; Yan, L. Space-Based Cloud-Fog Computing Architecture and Its Applications. In Proceedings of the 2019 IEEE World Congress on Services (SERVICES), Milan, Italy, 8–13 July 2019.
20. Harchol-Balter, M. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*; Cambridge University Press: Cambridge, UK, 2020; pp. 178–183.
21. Tijms, H.C. *A First Course in Stochastic Models*; John Wiley and Sons: Hoboken, NJ, USA, 2003.
22. Zhang, S.; Zhu, Z.; Hu, H.; Li, Y. Research on Task Satellite Selection Method for Space Object Detection LEO Constellation Based on Observation Window Projection Analysis. *Aerospace* **2021**, *8*, 156. [CrossRef]
23. Qin, M.; Cheng, N.; Jing, Z.; Yang, T.; Xu, W.; Yang, Q.; Rao, R.R. Service-Oriented Energy-Latency Tradeoff for IoT Task Partial Offloading in MEC-Enhanced Multi-RAT Networks. *IEEE Internet Things J.* **2021**, *8*, 1896–1907. [CrossRef]
24. Li, X.; Li, C.; Guo, X.; Qu, Z. A Modeling Method for Inter-Satellite Transmission Tasks Planning in Collaborative Network based on PDDL. In Proceedings of the 2019 14th IEEE International Conference on Electronic Measurement & Instruments (ICEMI), Changsha, China, 1–3 November 2019.
25. Wang, Y. Research on Key Technology of Space-based Network Intelligent Satellite Task Collaboration. Ph.D. Thesis, University of National Defense Science and Technology, Changsha, China, 2020.
26. Chen, H.; Li, J.; Du, C.; Peng, S. *Task Pkanning and Scheduling Technology for Earth Observation Satellite*; National Defense Industry Press: Beijing, China, 2021.
27. Rigo, C.A.; Seman, L.O.; Camponogara, E.; Filho, E.M.; Bezerra, E.A. Task scheduling for optimal power management and quality-of-assurance in CubeSats. *Acta Astronaut.* **2021**, *179*, 550–560. [CrossRef]
28. Zeng, A.; Jin, Y.; Ma, Z.; Han, L.; Gao, Y. Research on early warning analysis and display technology of space debris collision. *Space Debris Res.* **2019**, *19*, 7.
29. Mechalikh, C.; Taktak, H.; Moussa, F. PureEdgeSim: A Simulation Toolkit for Performance Evaluation of Cloud, Fog, and Pure Edge Computing Environments. *Comput. Sci. Inf. Syst.* **2021**, *18*, 43–66. [CrossRef]
30. Yan, L.; Cao, S.; Gong, Y.; Han, H.; Wei, J.; Zhao, Y.; Yang, S. SatEC: A 5G Satellite Edge Computing Framework Based on Microservice Architecture. *Sensors* **2019**, *19*, 831. [CrossRef] [PubMed]