Modeling and Simulation of Computation Offloading at LEO Satellite Constellation Network Edge

Junyu Lai(✉), Huidong Tan, Meilin He, Ying Qu, and Lei Zhong

School of Aeronautics and Astronautics, University of Electronic Science and Technologies of China, Chengdu, China
laijy@uestc.edu.cn

Abstract. Similar to terrestrial networks where edge computing facilities have already been introduced to decrease the user request response delay and to reduce the backhaul bandwidth consumption, low earth orbit (LEO) satellite constellation networks can also be benefited by adopting edge computing technologies. The computation tasks generated by ground users can be offloaded to their accessing LEO satellites to enhance network QoS and user QoE. This paper focuses on modeling and simulating computation offloading at LEO constellation network edge. A one-dimensional networking model for edge computing enabled LEO constellation networks is derived, and on that basis, a Monte Carlo simulator is developed from scratch to evaluate system performance. As a case study, three different computation offloading schemes are elaborated and implemented on the simulator. Comparative evaluation experiments have been conducted and the results indicate that, in resource restricted scenarios, allowing computation offloading to the neighbors of the access satellites can considerably reduce the request blocking probability with only slightly increasing the average request response delay.

Keywords: Satellite network · Edge computing · Computation offloading · Simulation · Performance evaluation

1 Introduction

In the past several years, numerous projects on constructing the next-generation LEO satellite constellation networks such as *Starlink* and *OneWeb* were proposed, and their satellites are currently launching to the space at an astonishingly fast speed which had never happened before. The major purpose of constructing these LEO satellite constellation networks are to complement the traditional terrestrial networks, so as to provide wireless broadband networking services directly from space for the ground users no matter where they are in the globe. On the other hand, Edge computing refers to move compute resource conventionally located in the backend cloud towards the users at the frontend. It can be realized in any nodes such as gateways, base stations, or even user terminals themselves aiming at reducing request response latency, enhancing user quality

H. Song and D. Jiang (Eds.): SIMUtools 2020, LNICST 370, pp. 451–463, 2021.
https://doi.org/10.1007/978-3-030-72795-6_37

of experience (QoE) and decreasing the backhaul bandwidth consumption. Integrating edge computing ability in a terrestrial network has already attracted tremendous research efforts. It is obvious that edge computing in LEO satellite constellation networks can gain even larger profits on network performance improvements considering the networking scales. However, to our best knowledge, few research efforts are now existing on modeling and simulating edge computing offloading in the upcoming LEO constellation networks. This paper explores this research topic, and the major contribution includes:

- Designs a simplified one-dimensional networking model for the edge computing ability enabled LEO satellite constellation network;
- Elaborates an innovative *Monte Carlo* simulator from scratch to evaluate the performance of diverse edge computing offload schemes in the LEO satellite constellation networks;
- Proposes three different edge computing offload schemes and conducts comparative simulation experiments to evaluate them as a comprehensive case study.

The remaining part of this paper goes as follows. Section 2 introduces the state-of-the-art on applying edge computing in satellite networks. Section 3 derives a system model on network architecture, user behavior, as well as network applications in LEO satellite constellation networks. An innovative Monte Carlo simulator, developed from scratch to evaluate the performance of computing offloading at the LEO satellite constellation network edge, is presented in Sect. 4, followed by Sect. 5 where three different edge computation offloading schemes are proposed and evaluated in the above simulator. Finally, Sect. 6 concludes the paper and provides the outlook.

2 Related Work

In legacy terrestrial mobile networking domain, real time applications are facing non-neglectful and large request response delays between the frontend mobile users and the backend cloud. In [2], a new paradigm named mobile edge computing (MEC) has been proposed to facilitate its influence. It refers that the edge of introducing computing and storage resources to mobile networks provide users with ultra-low latency and high-bandwidth network services. As one of the key technologies of MEC, computing offloading means that the terminal device will offload part or all of the computing tasks to edge to eliminate the shortcomings of mobile devices in terms of resource storage, computing performance, and energy efficiency. In order to reduce transmission delay, Liu et al. in [3], establish a mathematical model and design the Lyapunov optimized dynamic offload algorithm. Focus on the issue of a long response time of task offloading in multi-cloudlet, in [4], a weighted self-adaptive inertia weight particle swarm optimization algorithm based on multi-cloudlet collaboration is proposed. In [5], Long et al. aimed to minimize the delay and monetary cost on mobile devices under the limited computation resource and communication resource and design a multi-objective computation offloading and resource allocation game algorithm to achieve the optimal computational offload strategy. In [6], an energy optimization model was established and an artificial fish swarm algorithm was used to achieve the goal of reducing energy consumption.

The work in [7] aimed at the trade-off between energy consumption and time delay. It proposes a trade-off model of energy and time delay and an optimization algorithm based on the Lyapunov strategy. Considering delay and energy consumption of mobile devices jointly in a wireless access network, Hao in [8], propose a delay and energy consumption efficient offloading algorithm based on an alternating direction method of multipliers.

Edge computing has also been introduced to satellite networks to reduce propagation delays caused by long distances between satellites and ground terminals. In [9], Bradley *et al.* designed an Orbital Edge Computing System and using Nanosatellite constellations to reduce the transmission delay. Cheng *et al.* in [10], proposed a space-air-ground integrated network architecture, which utilizes the satellite as an access to connect edge servers with the terrestrial network to provide seamless and flexible network coverage and services to large areas. In [11], Cao *et al.* design the space-based cloud-fog satellite network slices, reducing the average delay of services in different slices by cloud-fog satellite which equipped edge computing servers. In [12], it represents a double edge computation offloading algorithm to optimize energy consumption and reduce latency, utilizing an access satellite to combine a satellite terrestrial network and a satellite-terrestrial transport to offer computing offloading services for the region.

The above works rarely consider the upcoming LEO satellite broadband networks, neither the impact caused by the motion of LEO satellites on edge computing. Therefore, it is urgent and meaningful to investigate edge computing in LEO satellite network by mathematical modeling and to build a dedicated simulation platform.

3 System Modeling

3.1 Modeling Network Architecture

The number of satellite orbits in the targeting LEO satellite constellation network is denoted as N_{orbit}. For simplicity, only one orbit is considered in this network model. There are N_s LEO satellites evenly distributed in the orbit. The height of these satellites is l_{gs} kilometers from the Earth surface, and the moving speed can be calculated by the Kepler's law:

$$V_{\mathrm{s}} = 2\pi \frac{R'^{\frac{3}{2}}}{T' \times (R' + l_{gs})^{\frac{1}{2}}} \tag{1}$$

Where, R' is the radius of the Earth and T' is its rotation period.

A ground user can directly communicate with those LEO satellites flying above it, and there is only one satellite serving as the access satellite for each user at a random point of time. Due to different landforms of the Earth, users are unevenly distributed on the ground. This paper divides the Earth surface into M different regions, each with non-identical population distribution. For instance, the population in the mountain regions is much smaller than that in the plain areas, but is larger than that in the ocean regions. Assume that in the ith region, the number of users is denoted as P_i, which conforms to

a uniform distribution with the average value U_i, its probability density function (PDF) is:

$$f(x) = \begin{cases} \frac{1}{b-a} & for\ a \le x \le b \\ 0 & elsewhere \end{cases} \tag{2}$$

Where, a and b are the boundaries of the area.

To summarize, the above network architecture model is shown in Fig. 1. The ith satellite S_i moves at speed V_s. The distance between the S_i and ground terminal is l_{gs} and l_{ss} is the length of inter-satellites link (ISL). Moreover, the earth ground is evenly divided into M areas.

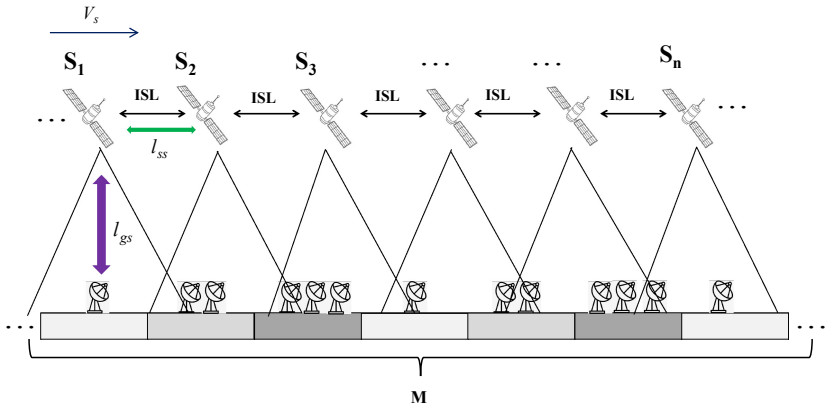

Fig. 1. Networking architecture model

3.2 Modeling User Behavior

Assume all the ground users are independent and are periodically sending computation task requests to their access LEO satellites. The computation task requesting frequencies of all the users are assumed to be the same, and the time interval between a user's two consecutive task requests follows a negative exponential distribution, with the probability density function:

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x}, & x \ge 0 \\ 0, & x < 0 \end{cases} \tag{3}$$

Where, λ is the average number of task request occurrences per unit time. It is worth noting that ground user's moving speed is much smaller than the speed of LEO satellites, so the users' mobility can be ignored.

3.3 Modeling Network Applications

Task Model. Due to the power and space limitations, as well as the critical working environments, computing resources installed on the LEO satellite are limited and cannot be comparable with the edge computing facilities on the ground. The computation tasks offloaded to the LEO satellite are served by micro services running on light-weighted virtual instances assigned by satellite onboard computation resource management system. Therefore, the computation tasks offloaded to the LEO satellite are modeled as atomic operations, and can be presented by $A(S, T_d)$, where S is the task computation amount and T_d is the task completion deadline [13]. Assume all the computation tasks are independent and the task computation volume conforms to normal distributed:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{4}$$

Where, σ^2 is the variance of the task calculation volume and μ is the mean of the task calculation volume. A task $A(S, T_d)$ must be completed within the predefined time deadline T_d, or else the allocated edge computing resource will be released and the task will fail.

Delay Model. The computation task request response delay is denoted as *RRD*, which can be expressed by:

$$RRD = D_T + D_C + D_R \tag{5}$$

RRD includes task uploading delay D_T, task computation delay D_C, and result returning delay D_R. Task uploading delay includes the uplink transmission delay d_{up_t}, uplink propagation delay d_{up_p}, as well as possibly relaying delay $n \times d_r$.

Similarly, result returning delay consists of downlink transmission delay d_{down_t}, down link propagation delay d_{down_p}, and possibly relaying delay $n \times d_r$. The propagation delay is proportional to the distance between the LEO satellite actually running the task and the requesting user. In addition, the data transmission delay is related to the volume of the computation task and the bandwidth of the transmission link [11]. Each relaying satellite forwarding the task or computation result will introduce delay d_r, and if the number of relaying satellites for a transmission is m, the total number of relay times will be n =2 m. The equations to evaluate the task uploading delay D_T and the result returning delay D_R are as follows:

$$D_R = d_{down\_t} + d_{down\_p} + n \times d_r \tag{6}$$

$$D_T = d_{up\_t} + d_{up\_p} + n \times d_r \tag{7}$$

To make the above depiction clearer, the delay model is explained in Fig. 2

4 Designing and Implementing Simulation Platform

4.1 General Assumptions

In order to develop the simulation platform for the LEO satellite constellation network, some general assumptions are made. Firstly, the shape of the orbit is circular, and only

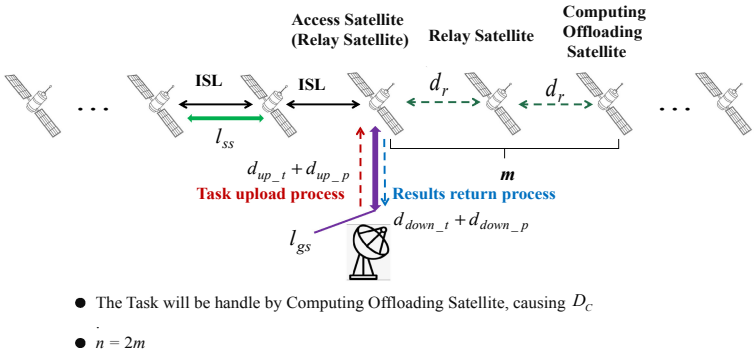

Fig. 2. Application delay model

one orbit is considered. Secondly, the LEO satellites are evenly distributed in the orbit. Thirdly, access satellite handover is seamless, and there is no communication break when during handover. Fourthly, the speed of signal transmission in the satellite network is the speed of light. Fifthly, the user moving speed is zero.

4.2 Designing Principles

The derived *Monte Carlo* simulation platform is based on an event-driven model. There are logically three steps in general:

- **STEP_I.** According to the satellite networking model and the user distribution model, establish the targeted network topology;
- **STEP_II.** Following user requesting model and task model, generate a discrete event trace file (including handover events, user requesting events, etc.) for each single user;
- **STEP_III.** Combine the trace files of all the users into a system-level trace file and dynamically insert new events during the simulation process.

4.3 Implementations

The simulation is divided into three parts: satellite section, user section, user and satellite interacting section. The satellite section mainly initializes the satellite

position and the satellite resource. The user section is to simulate the distribution of users, user parameters, and user event trace files which are the base of the system-level trace file. The satellite-user interaction part is to execute the procedure following the system-level trace file. Firstly, if there are redundant events in the list, it will check the event and policy type, and the communication access situation before occupying computing resources. Then, it will calculate every task end time according to different strategies. Lastly, the end event will be inserted into the system-level trace file followed by returning to step 1. Its flow chart is shown in Fig. 3.

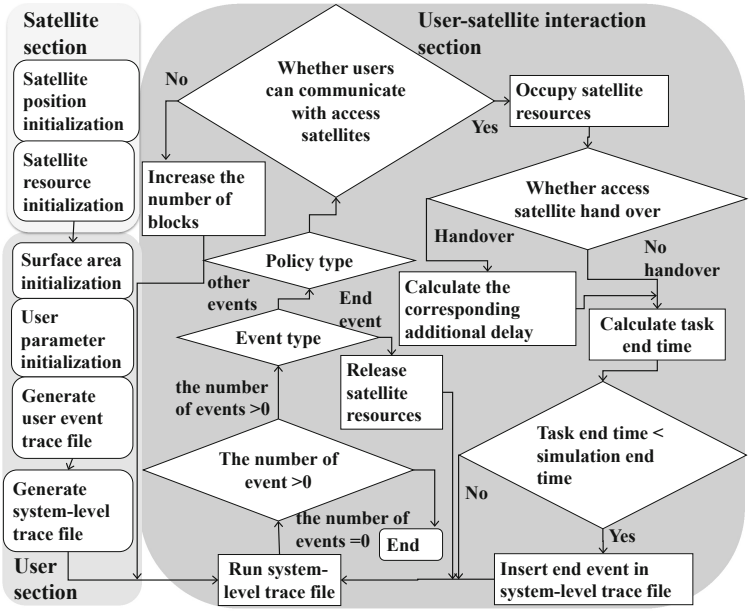

Fig. 3. Simulation platform implementation details

5 Case Study: Comparative Performance Evaluation for Three Different Computation Offloading Schemes

5.1 Three Different Computing Offloading Schemes

Firstly, notations used in this case study are defined in Table 1.

Table 1. Summary of notations.

Notation	Implication	Notation	Implication
d_r	Relay delay due to one satellite pitch	l_{ss}	Inter-satellite link length
l_{gs}	Satellite-to-ground link length	d_{ss}	Propagation delay between two adjacent satellites
S	Task calculation	r_{up}	Task upload rate
r_{down}	Result return rate	r_{cp}	Calculation rate
β	Amount of results/number of tasks	a	Completion of mission uploading when satellite hand over
b	Completeness of result returning during satellite handover	c	Light speed

Secondly, the formula for calculating each type of delay are as listed below:

$$\begin{cases} d_{up\_t} = \dfrac{S}{r_{up}} & (8) \\ d_{down\_t} = \dfrac{\beta \times S}{r_{down}} & (9) \\ d_{up\_p} = d_{down\_p} = \dfrac{l_{gs}}{c} & (10) \\ d_{ss} = \dfrac{l_{ss}}{c} & (11) \\ D_C = \dfrac{S}{r_{cp}} & (12) \end{cases}$$

For edge computing in LEO satellite constellation network, this paper derives three different computing offloading strategies. Illustrated in Fig. 4, where Satellite *C* is the access satellite at the initial time T_0, and Satellites *A, B, D* and *E* are its neighboring satellites.

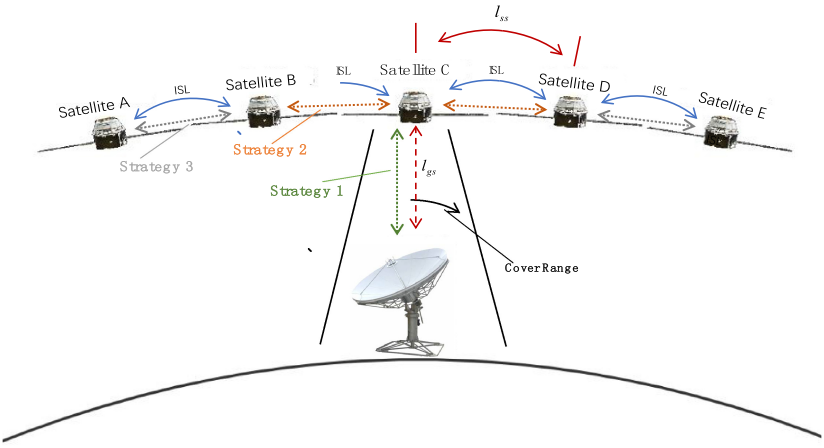

Fig. 4. The Distribution of access satellite and its neighboring Satellites

Strategy 1. Satellite *C* is adopted as the only satellite for conducting the offloaded computing tasks at network edge. If the computing resources of *C* are used up, the newly arrived user task request will be rejected. Since the LEO satellite is always in motion, the coverage time of a certain location on the earth is limited, so that the user's access satellite may have changed during the task processing duration (defined as the time interval starting from the instant when a user generates the computation task to the moment when the user receives the computation result). This process is called handover. In order to accurately evaluate the *RRD*, the time instant when the handover happens are carefully analyzed. Table 2 summarizes all the possible situations.

Table 2. Delay calculation for Strategy 1

Strategy_1		
No. handover		$D_T = d_{up\_t} + d_{up\_p},\ D_R = d_{down\_p} + d_{down\_t}$
Handover happens at	Task upload stage	$D_T = d_{up\_t} + d_{up\_p} + d_r + d_{ss},$ $D_R = d_{down\_t} + d_{down\_p} + d_r + d_{ss}$
	Task calculation stage	$D_T = d_{up\_t} + d_{up\_p},\ D_R = d_{down\_t} + d_{down\_p} + d_r + d_{ss}$
	Task result return stage	$D_T = d_{up\_t} + d_{up\_p},\ D_R = d_{down\_t} + d_{down\_p} + d_r + d_{ss}$

Strategy 2. Based on Strategy 1, if Satellite C do not have sufficient compute resource, it can further estimate whether the computing resources of satellites B and D are available. If one of the above satellites has compute resources, it will offload the computation task to that satellite, and the access satellite becomes a relay satellite to forward the content. Then, if the resource of the two nearby satellite also ran out the compute resources, the user's request is rejected, increasing the blocking counter. Handover occurs in different time periods of task processing duration will have different effects on the *RRD*. Table 3 summarizes all the possible situations.

Table 3. Delay calculation for Strategy 2

Strategy_2	No. handover		
	$D_T = d_{up\_t} + d_{up\_p} + d_r + d_{ss},\ D_R = d_{down\_t} = d_{down\_p} + d_r + d_{ss}$		
	Handover		
		Getting further	Getting closer
	Task upload stage	$D_T = d_{up\_t} + d_{up\_p} + 2(d_r + d_{ss})$ $D_R = d_{down\_t} + d_{down\_p} + 2(d_r + d_{ss})$	$D_T = \max\begin{cases} a \times d_{up\_t} + d_{up\_p} + d_r + d_{ss} \\ d_{up\_t} + d_{up\_p} \end{cases}$ $D_R = d_{down\_t} + d_{up\_p}$
	Task calculation stage	$D_T = d_{up\_t} + d_{up\_p} + d_r + d_{ss}$ $D_R = d_{down\_t} + d_{down\_p} + 2(d_r + d_{ss})$	$D_T = d_{up\_t} + d_{up\_p} + d_r + d_{ss}$ $D_R = d_{down\_t} + d_{down\_p}$
	Task results return stage	$D_T = d_{up\_t} + d_{up\_p} + d_r + d_{ss}$ $D_R = d_{down\_t} + d_{down\_p} + 2(d_r + d_{ss})$	$D_T = d_{up\_t} + d_{up\_p} + d_r + d_{ss}$ $D_R = \max\begin{cases} b \times d_{down\_t} + d_{down\_p} + d_r + d_{ss} \\ d_{down\_t} + d_{down\_p} \end{cases}$

Strategy 3. Based on strategy 2, if the computing resources of satellites B and D are running out, then the strategy will check whether there are available computing resources on satellites A and E. If one of the satellites has sufficient resources, the strategy will offload the computation task to that satellite with satellite B or D as its relay node. If satellites A and E have not enough compute resource, this task request is blocked. It is noted that, strategies 2 and 3 prefer to choose the satellites flying close to the user than the one fly away from the user. The impact of the handover event on *RRD* in Strategy 3 is shown in Table 4.

Table 4. Delay calculation for Strategy 3

Strategy 3	No. handover		
	$D_T = d_{up\_t} + d_{up\_p} + 2(d_r + d_{ss})$, $D_R = d_{down\_t} + d_{down\_p} + 2(d_r + d_{ss})$		
	Handover		
		Getting further	Getting closer
	Task upload stage	$D_T = d_{up\_t} + d_{up\_p} + 3(d_r + d_{ss})$ $D_R = d_{down\_t} + d_{down\_p} + 3(d_r + d_{ss})$	$D_T = \max\begin{cases} a \times d_{up\_t} + d_{up\_p} + 2(d_r + d_{ss}) \\ d_{up\_t} + d_{up\_p} \end{cases}$ $D_R = d_{down\_t} + d_{down\_p} + d_r + d_{ss}$
	Task calculation stage	$D_T = d_{up\_t} + d_{up\_p} + 2(d_r + d_{ss})$ $D_R = d_{down\_t} + d_{down\_p} + 3(d_r + d_{ss})$	$D_T = d_{up\_t} + d_{up\_p} + 2(d_r + d_{ss})$ $D_R = d_{down\_t} + d_{down\_p} + d_r + d_{ss}$
	Task results return stage	$D_T = d_{up\_t} + d_{up\_p} + 2(d_r + d_{down\_p})$ $D_R = d_{down\_t} + d_{down\_p} + 3(d_r + d_{down\_p})$	$D_T = d_{up\_t} + d_{up\_p} + 2(d_r + d_{ss})$ $D_R = \max\begin{cases} b \times d_{down\_t} + d_{down\_p} + 2(d_r + d_{ss}) \\ d_{down\_t} + d_{down\_p} + d_r + d_{ss} \end{cases}$

5.2 Performance Metrics

The performance metrics considered in this case study are request blocking probability (RBP) and average request response delay (ARRD). RBP is the ratio of the number of blocked task requests due to resource restriction to the total number of task requests, while ARRD is the average value of the computation task request response delay.

5.3 Simulation Assumptions

The specific assumptions for the simulation experiments are given in Table 5:

Table 5. Simulation assumptions

Parameters	Values	Parameters	Values
The satellite speed	27000 km/h	The distance between adjacent satellites	4481 km
The height of the satellite	780 km	Relay delay	1 s
The number of satellites	11	The inter-satellite transmission delay of the two adjacent satellites	14.9 ms
Rate parameter λ	1	The number of regions divided on the Earth's surface	9

5.4 Experimental Results

The results of the comparative simulation experiments for the aforementioned three computation offload strategies are as given in Fig. 5 and 6. The vertical lines represent the confidence intervals. According to the above results, it is no surprise that Strategies 2 and 3 have a better quality of RBP than Strategy 1, attributed to fuller use of satellite resources. However, as the user population gets larger, the difference between Strategy 2 and Strategy 3 is getting smaller. Because each strategy has the same amount of computing resources. Even if these two strategies are distinct, the RBP of them will become close with a sufficient utilization of computing resources. From Fig. 6, the initial ADDR of the three strategies are close, indicating that under the initial number of users, there are still surplus resources in the satellite, and users always offload to the access satellite. With the increase in the number of users, the access satellite resources gradually decreases, and users access to nearby satellites through Strategy 2 and Strategy 3. Therefore, the ADDR of Strategy 2 and Strategy 3 begin to increase. Although the trends of these strategies are various, the ARRD of Strategy 3 rise most rapidly, which implicates that the number of relay satellites should be appropriate.

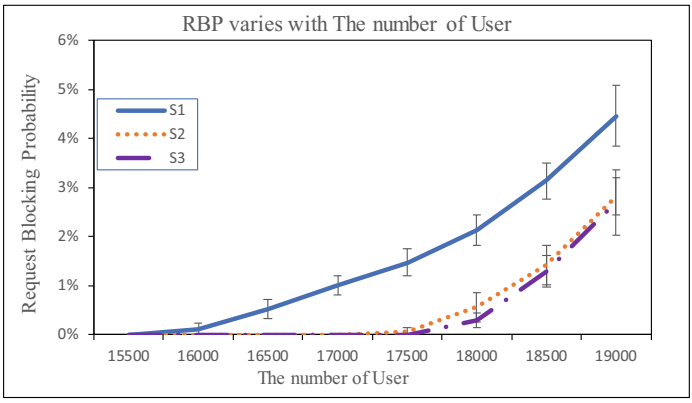

Fig. 5. RBP varies against the number of users

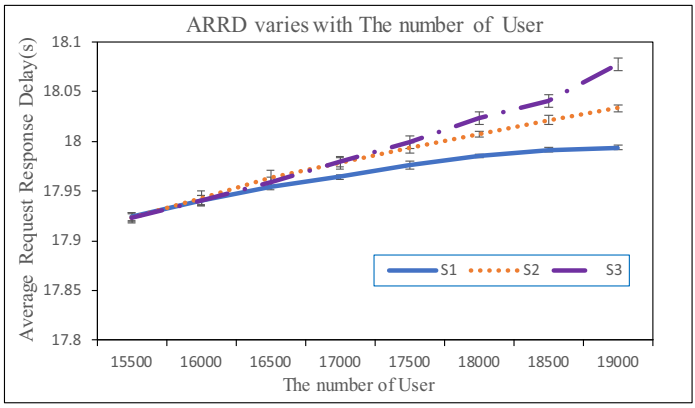

Fig. 6. ARRD varies against the number of users

6 Conclusions

Currently, the edge computing technology is widely deployed in terrestrial networks, and it also has a great potential to be applied in the upcoming broadband LEO satellite networks to effectively reduce user request response delay, as well as to decrease satellite backhaul bandwidth consumption. This paper derives a one-dimensional model for edge computing enabled LEO satellite network, and on that basis, it develops a novel Monte Carlo simulator to evaluate system performance. This paper also elaborates three different computation offloading schemes and conducts comparative evaluation experiments using the simulator. The results indicate that computation offloading to the neighboring LEO satellites can considerably reduce the request blocking probability with only slightly increasing the average request response delay.

The potential research work we have planned for the next step includes the following two points: Firstly, the current one-dimensional simulator will be extended to three dimensional versions so to better depict the target network; second, the comprehensive investigation will be carried on to model user behaviors and edge computing tasks in a more realistic manner.

Acknowledgement. This work is supported by the National Natural Science Foundation of China (Grant No. 61402085), the Science and Technology on Communication Networks Laboratory (Grant No. XX17641X011-03), and the 54th Research Institute of CETC.

References

1. Lovelly, T.M., et al.: A framework to analyze processor architectures for next-generation on-board space computing. In: IEEE Aerospace Conference, pp. 1–10. IEEE, Big Sky (2014)
2. Flores, H., et al.: Mobile code offloading: from concept to practice and beyond. IEEE Commun. Mag. **53**(3), 80–88 (2015)

3. Liu, J., et al.: Delay-optimal computation task scheduling for mobile-edge computing systems. In: 2016 IEEE International Symposium on Information Theory, pp. 1451–1455. IEEE, Barcelona (2016)
4. Wang, Q., et al.: Computation tasks offloading scheme based on multi-cloudlet collaboration for edge computing. In: 2019 Seventh International Conference on Advanced Cloud and Big Data, pp. 339–344. IEEE, Suzhou (2019)
5. Long, L., et al.: Delay optimized computation offloading and resource allocation for mobile edge computing. In: 2019 IEEE 90th Vehicular Technology Conference, pp. 1–5. IEEE, Honolulu (2019)
6. Kamoun, M., et al.: Joint resource allocation and offloading strategies in cloud enabled cellular networks. In: IEEE International Conference on Communications, pp. 5529–5534. IEEE, London (2015)
7. Wang, W., et al.: Computational offloading with delay and capacity constraints in mobile edge. In: IEEE International Conference on Communications, pp. 1–6. IEEE, Paris (2017)
8. Hao, Z., et al.: A delay and energy consumption efficient offloading algorithm in mobile edge computing system. In: 2019 IEEE 11th International Conference on Communication Software and Networks, pp. 251–257, Chongqing (2019)
9. Denby, B., et al.: Orbital edge computing: machine inference in space. IEEE Comput. Archit. Lett. **18**(1), 59–62 (2019)
10. Cheng, N., et al.: Space/aerial-assisted computing offloading for IoT applications: a learning-based approach. IEEE J. Sel. Areas Commun. **37**(5), 1117–1129 (2019)
11. Suzhi, C., et al.: Space edge cloud enabling network slicing for 5G satellite network. In: 15th International Wireless Communications & Mobile Computing Conference, pp. 787–792. IEEE, Tangier (2019)
12. Wang, Y., et al.: A computation offloading strategy in satellite terrestrial networks with double edge computing. In: IEEE International Conference on Communication Systems, pp. 450–455. IEEE, Chengdu (2018)
13. Wang, T.F., et al.: Computing as a service pattern based on edge computing. Appl. Electron. Technol. **45**(5), 80–83 (2019)