This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3184989

Static Placement and Dynamic Assignment of SDN Controllers in LEO Satellite Networks

Jianming Guo, Lei Yang, David Rincón, *Member, IEEE,* Sebastià Sallent, *Life Member, IEEE,* Quan Chen, and Xianfeng Liu

***Abstract*—Software-defined networking (SDN) logically separates the control and data planes, thus opening the way to more flexible configurations and management of low-Earth orbit (LEO) satellite networks. Since one or, more generally, multiple distributed controllers are needed, a significant challenge in SDN is the controller placement problem (CPP). Due to characteristics such as the dynamic network topology, limited bandwidth and traffic variations, the CPP is quite complex in SDN-based satellite networks. In this paper, we propose solving the CPP by means of a static placement with dynamic assignment (SPDA) method for LEO satellite networks. The SPDA method has two parts: the first is to incorporate SDN controllers into some fixed satellites by formulating a mixed integer programming model; the second is to dynamically assign switches to existing controllers according to the switch-controller latency and the traffic load of controllers. The SPDA method takes the topological dynamics into account by effectively dividing time snapshots, and it has a lower bandwidth consumption compared with methods involving controller migrations. Real satellite constellations are used to evaluate the performance of our controller placement solution. The results show that SPDA outperforms existing methods in terms of reducing the switch-controller latency, and it also has good load balancing performance.**

***Index Terms*—software-defined networking, low-Earth orbit, controller placement, satellite network, dynamic topology, load balancing.**

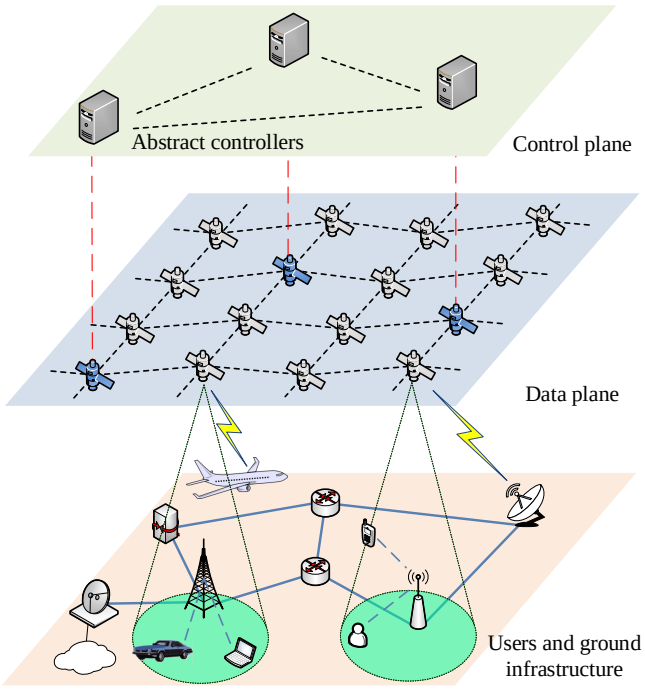


Fig. 1. Architecture of an SDN-based LEO satellite network.

I. Introduction

AIMING to provide global Internet services, low-Earth orbit (LEO) satellite networks have gained more attention in recent years. Some LEO constellation projects with hundreds or thousands of satellites are being invested in and developed, such as Telesat, OneWeb, and Starlink, to name a few. However, due to the rigidity of the legacy architecture, existing satellite networks also face challenges [1]. On the one hand, network management and configuration lack flexibility due to the inherent coupling of software and hardware in traditional satellite networking devices [2]. On the other hand, satellite networks are hindered from directly merging with new terrestrial network technologies (e.g., 5G) because their protocols evolve differently and are developed independently [3]. The introduction of software-defined networking (SDN) into satellite networks may overcome the aforementioned problems. With a logically centralized control plane that is decoupled from the data plane, SDN provides the capability of simplifying network programming and management. The architecture of an SDN-based LEO satellite network is depicted in Fig. 1, where controllers are co-located with switches on LEO satellites in the data plane and compose the control plane. The ground infrastructure comprises gateways as well as telemetry, tracking and control (TT&C) stations, while users consist of fixed or mobile devices using specific modems, such as vehicles, aircraft, sensors and actuators, among others.

Nonetheless, embracing SDN can also incur new challenges. The logically centralized control plane often has multiple controllers that are physically distributed for catering to network scale expansion, which gives rise to the controller placement problem (CPP). CPP generally consists in deciding how many

This work was supported in part by the National Natural Science Foundation of China No. 11725211 and No. 12002383, and by the Agencia Estatal de Investigación of Ministerio de Ciencia e Innovación of Spain under project PID2019-108713RB-C51 MCIN/ AEI/10.13039/501100011033.

Jianming Guo is with the College of Aerospace Science and Engineering, National University of Defense Technology, Changsha, Hunan 410073, China, and also with the Science and Technology on Complex Aircraft System Simulation Laboratory, Beijing 100094, China. E-mail: gjm08110@hotmail.com.

Lei Yang, and Quan Chen are with the College of Aerospace Science and Engineering, National University of Defense Technology, Changsha, Hunan 410073, China. E-mail: {yanglei, chenquan11}@nudt.edu.cn.

David Rincón and Sebastià Sallent are with the Department of Network Engineering, Universitat Politècnica de Catalunya (UPC), Castelldefels, 08860, Barcelona, Spain. E-mail: {drincon, sallent}@entel.upc.edu.

Xianfeng Liu is with the Beijing Institute of Tracking and Telecommunications Technology, Beijing 100094, China. E-mail: liuxianfeng_edu@163.com.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3184989

controllers are needed for network management and where to locate them along with other considerations such as SDN global reliability. CPP also involves the mapping of controllers and switches (assigning the appropriate switches to a specific controller). Controller placement can determine switch-controller latency, which mainly affects network resilience and quality of service (QoS) [4].

Although some studies have been conducted on CPP in both terrestrial and satellite networks, they rarely consider the intrinsic characteristics of satellite networks, such as dynamic topology, bandwidth limitations, and traffic variations. As LEO satellites fly around the Earth, the network topology changes quickly and frequently, which indeed influences the switch-controller latency. Thus, static controller placement (SCP) methods like those described in [5], [6] cannot be directly adopted in satellite scenarios. On the other hand, dynamic controller placement (DCP) methods for satellite networks (like in [7]) adopt a controller migration strategy, which considers the topological dynamics exactly and also adapts to traffic variations. However, the limited bandwidth of on-orbit satellites makes it costly to frequently migrate controllers from one satellite to another, thereby preventing the practical implementation of DCP. Moreover, the switch-controller mapping has not been thoroughly analyzed. SCP methods maintain static assignment between switches and controllers while DCP methods migrate controllers, thus causing a dynamic switch-controller assignment. Nevertheless, neither SCP nor DCP methods can provide explicit guidance on how to determine a new controller for a given switch.

This paper aims to develop a new method that combines both DCP and SCP in order to avoid their aforementioned shortcomings. We solve the CPP in LEO satellite networks by considering their specific characteristics, including the dynamic topology, limited bandwidth and traffic variability. The main contributions of this paper are listed as follows:

- A static placement and dynamic assignment method is proposed to solve CPP in LEO satellite networks.
- A snapshot division method is developed to deal with the dynamic topology of LEO satellite networks.
- An arrival traffic estimation model is presented to estimate the traffic load of controllers and switches in LEO satellite networks.

The rest of the paper is organized as follows. In Section II, we summarize the existing works on CPP in both terrestrial and satellite networks. In Section III, we describe an SDN-based LEO satellite constellation architecture, and provide a set of parameters to represent the constellation. We then describe the switch migration model, illustrate the migration process, and outline the costs. Moreover, we analyze the changeable and periodical topology of satellite constellations and present an improved scheme for dividing time snapshots. We also provide a traffic estimation model for controllers. Accordingly, Section IV proposes a new method for solving CPP, namely static placement with dynamic assignment (SPDA), which involves two parts: statically placing controllers on certain nodes and then dynamically assigning switches to them according to switch-controller latency and controller traffic load. We formulate the static controller placement into a mixed-integer programming (MIP) model, develop the static placement schemes, and then obtain the dynamic assignment results in terms of latency and load balancing. In Section V, we adopt two types of real constellations to test SPDA and compare it with methods developed in previous works. The results indicate that our method outperforms both SCP and DCP, because SPDA not only reduces the bandwidth consumption of controller migrations but also solves the highly dynamic topology problem in satellite networks. Finally, we conclude the paper in Section VI.

II. Related Works

To the best of our knowledge, the first attempt at solving CPP is the research in [8], which proposes a min-cut based graph partitioning algorithm to maximize the resilience between the controller and switches. [8] points out that CPP basically consists of selecting a subset for connecting several controllers that meet some requirements (e.g., maximizing the total network resilience). Since then, researchers have explored CPP mainly for terrestrial networks, which can be divided into roughly two categories: SCP and DCP. In general, SCP aims to deploy controllers on fixed nodes and maintain stationary switch-controller assignment. An example of SCP is given in [9], where controllers are statically placed on the ground together with satellite gateways. By contrast, DCP allows a controller to migrate from one node to another, and thus the switch-controller assignment is changeable [10]. The remainder of this section will briefly summarize SCP and DCP in both terrestrial and satellite networks, respectively.

Most studies on terrestrial CPP focus on SCP methods. Heller *et al.* [5] first use switch-controller latency to evaluate controller placement schemes in terrestrial networks. Hu *et al.* [6] study CPP from the perspective of maximizing the reliability of SDN control networks, where they define a control path as the path either between a switch and its controller or between different controllers. The expected percentage of control path loss is adopted as the evaluation metric. Yao *et al.* [11] take the traffic load of controllers into account in order to obtain a more traffic-balanced controller placement solution. Jiménez *et al.* [12] aim to find the minimum number of controllers and their locations, which involves both network robustness and load balancing. Since there are many objectives for CPP, Lange *et al.* [13] create a Pareto-based framework to find a balanced trade-off between different objectives, or in other words, performance metrics, for optimal controller placement solutions. Apart from the above, other objectives such as QoS-guaranteed and energy consumption are also discussed in [14], [15] and [16], respectively. A survey on controller placement in terrestrial SDN is presented in [4], where a classical CPP formulation is described as an integer linear programming (ILP) model. Different CPP objectives are summarized and discussed, mainly control latency, load balancing, resiliency, reliability, and cost efficiency, among others. Furthermore, the authors sum up both heuristic and optimal methods to solve CPP, among which the greedy algorithm and ILP are the most popular methods. Moreover,

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3184989

Wang *et al.* [17] propose a clustering-based network partition algorithm to address CPP, which partitions a network into subnetworks to shorten the propagation latency between controllers and their associated switches. However, these methods cannot be adopted directly to a satellite scenario due to the dynamic network topology.

There are also a few studies concerned with DCP in terrestrial networks. Bari *et al.* [10] propose a multi-controller placement framework that dynamically adjusts the number of active controllers and assigns switches to each controller according to topological dynamics. Two heuristic (greedy-based and simulated annealing-based) algorithms are developed in order to ensure a minimal flow setup time and communication costs. He *et al.* [18] present a DCP model to simultaneously optimize controller locations and switch-to-controller assignments for minimum average flow setup time with respect to the network's different traffic conditions. However, controller migration costs are neglected in these works. Furthermore, Torkzaban *et al.* [19] aim to minimize the failure probability of SDN control paths to ensure reliability between controllers and switches by deploying controllers close to satellite gateways, which would be a good choice for terrestrial controller placement but not necessarily for the on-board controllers in satellites.

Both SCP and DCP struggle to adapt to the characteristics of LEO satellite networks. Because satellites fly around the Earth, the links between them may change with time, thereby leading to a highly dynamic topology. As a result, SCP cannot satisfy all the topological configurations, which causes long switch-controller latency in some time instances. On the other hand, although applying DCP migrations can meet the requirements of network topology changes, it undoubtedly consumes excessive time, bandwidth and power. When a satellite periodically orbits around the Earth, its traffic load will change over time due to the dynamic topology and the Earth's regular rotation. Thus, in order to balance the traffic load, previous studies tend to dynamically migrate controllers [20], [21], which requires frequent and synchronous migrating operations that consume high bandwidth and a large number of calculations for the real-time controller placement scheme. Therefore, traffic-based DCP might not be entirely appropriate for satellite networks. A combined static-dynamic method for CPP is proposed in [22], where controllers are statically placed on ground control centers and geostationary orbit (GEO) satellites but dynamically placed on two LEO satellites for each orbit closest to 0° latitude. Nevertheless, [22] does not consider the mobility of LEO satellites and provides no details on the controller migration process. In a word, existing SCP and DCP methods are not quite suitable for LEO satellite networks.

Apart from SCP and DCP, some studies propose dynamic assignment methods for changing the mapping between controllers and switches for the sake of latency constraint, load balancing, etc. Dixit *et al.* [23] provide an elastic distributed controller architecture, i.e., *ElastiCon*, in which the controller pool can automatically grow or shrink as traffic conditions change over time. *ElastiCon* enables the switch migration process to better distribute the controller load, and it also implements a minimal disruption migration protocol. Wang *et al.* [24] propose a dynamic controller assignment model and develop an efficient online algorithm to reassign switches to controllers according to network conditions. The proposed method mainly aims to solve the problem of long response times caused by traffic variations in data center networks. Based on the dynamic assignment strategy, the load-balancing problem is considered in controller placement schemes. Hu *et al.* [25] assume that the control traffic demands of switches are the same and thus use the number of assigned switches as a metric to evaluate the traffic load. Furthermore, Ma *et al.* [26] propose a load-balancing mechanism for controllers in terrestrial SDNs by eliminating the bottleneck of centralized control, which is based solely on controller traffic. Hu *et al.* [27] also provide a load-balancing controller deployment scheme while considering the reliability of nodes and links, which can be easily affected by controller deployment results in terrestrial networks. Lan *et al.* [28] bring up a load balancing mechanism for terrestrial networks, which can dynamically migrate switches from heavily-loaded to lightly-loaded controllers. Xu *et al.* [29] propose two switch migration schemes not only to achieve load balance among controllers but also to reduce migration costs and provide low computation overhead. Apart from that, software-defined wide area networks (SDWANs) also study load-balancing methods for controller placement based on network topology, minimum-cost flow [30] and flow traffic distribution [31]. Due to the intrinsic characteristics of LEO satellite networks, these load-balancing methods cannot be adopted directly. To the best of our knowledge, there is no such research on dynamic controller assignment in satellite networks, and thus ours is the first work that combines static controller placement and dynamic assignment together.

III. System Model

Before confronting the CPP in LEO satellite networks, we need to address some issues with the system model concerning the constellation architecture, switch migration model, dynamic topology, and traffic estimation.

A. SDN-based Satellite Constellation Architecture

Our model takes into account both *Walker-δ* and *Walker-Polar* constellations, which cover almost all existing systems. A *Walker-δ* constellation usually has inclined orbits, while a *Walker-Polar* constellation's inclination angle is around 90° [32]. The architecture of a typical LEO satellite network is depicted in Fig. 2. In a LEO constellation, all satellites are organized into several orbit planes. Let us assume there are P planes and S satellites in each plane; thus, the total number of satellites is $N = P \times S$. A LEO constellation can be denoted by a set of parameters, i.e., $h{:}i{:}P/S/F$, which is slightly different from Walker's classical notation $h{:}i{:}N/P/F$. Here, h is the orbit altitude, i is the inclination angle, and F is an integer phase factor, representing the inter-plane phase difference of two neighboring satellites, namely $\Delta\omega_f$ (see Fig. 2). For a *Walker-δ* constellation, $F = 0, 1, ..., P-1$, and $\Delta\omega_f = 2\pi F/N$, while for a *Walker-Polar* constellation, $\Delta\omega_f = \pi/S$, which is half of the intra-plane phase difference of neighboring

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3184989

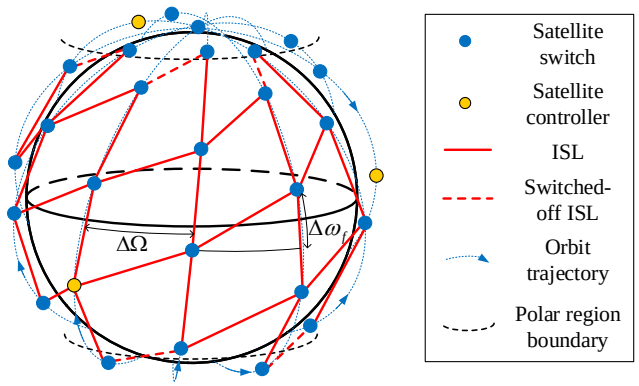


Fig. 2. Architecture of a typical LEO satellite network (*Walker-Polar* constellation). Some ISLs and satellites in the polar region are omitted for simplicity.

satellites. Moreover, $\Delta\Omega$ is the longitude difference between adjacent planes, and it usually equals $2\pi/P$ or π/P for, respectively, *Walker-δ* and *Walker-Polar* constellations.

Inter-satellite links (ISLs) are established between neighboring satellites in either the same orbit or adjacent orbits, which are called, respectively, intra-orbit and inter-orbit ISLs. It should be noticed that, in our model, inter-orbit ISLs exist only between two satellites moving in the same direction; otherwise they become unstable due to high Doppler shift and have a short duration, which incurs a so-called *seam* between two counter-rotating orbits in *Walker-Polar* constellations. We can define a time-varying graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ to describe the network topology of the constellation, where $\mathcal{V}$ and $\mathcal{E}$ are two sets of satellite nodes and ISLs, respectively.

In our model, all satellites are SDN switches and form the potential controller set, but only some of them will take the function of SDN controllers. In other words, controllers are physically co-located with satellite switches but logically separated from them. Switches that are assigned to a certain controller constitute the controller's domain; the controller is in charge of them and manages their flow tables. Switches only need to handle data packets in accordance with the instructions of their controllers. Let us assume that a controller c_i co-locates with the switch v_j, where $c_i, v_j \in \mathcal{V}$ and the total number of satellite nodes is N. We can thereby define an N-dimensional vector of 0-1 elements, i.e., $\mathbf{x} = (x_1, x_2, ..., x_N)$, to represent controller locations,

$$x_i = \begin{cases} 1, & \text{if } c_i \text{ exists} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The switch-controller assignment matrix $\mathbf{Y}$ is defined as

$$y_{ij} = \begin{cases} 1, & \text{if } v_j \text{ is assigned to } c_i \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where index i represents controller c_i and j represents switch v_j. Therefore, both $\mathbf{x}$ and $\mathbf{Y}$ are regarded as the design variables of the CPP.

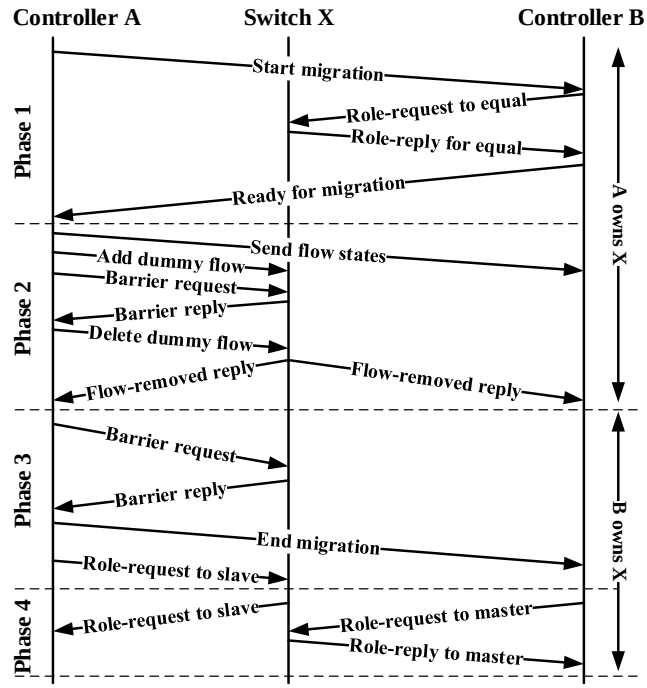


Fig. 3. Switch migration process in four phases.

B. Switch Migration Model

The SPDA method includes a dynamic assignment process requiring switch migrations, which means that a switch is detached from its original controller and reassigned to a new controller. This process occurs when a controller is overloaded and the network topology changes, thus increasing the switch-controller latency. Assume that we migrate switch X from controller A to controller B, and the switch migration process operates in four phases according to [23], [29] as shown in Fig. 3:

- In phase 1, controller A first sends a *start-migration* message to controller B, which then sends a *role-request* message to switch X and changes its role to equal. After B receives a *role-reply* message from X, it then immediately sends a *ready-for-migration* message to inform A that it can now receive X's messages but will just ignore them.
- In phase 2, A first sends the flow states of X to B to make sure B can take over immediately after the migration. A inserts a dummy flow to switch X, and then X deletes it for the purpose of sending a *flow-removed* message as an event signal to both A and B. After that, B will be in charge of processing messages (caching at first) from X while A ignores them, although both A and B are in equal mode.
- In phase 3, B will wait until A finishes its pending requests that arrived before the *flow-removed* message. After processing all the pending requests, A sends a *barrier-request* message to X and waits for the *barrier-reply* message, which will induce an *end-migration* signal sent from A to B.
- In phase 4, B will send a *role-request* message to X, and X will change the role of A to slave and that of B to

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3184989

master. All messages from X cached at B in phase 3 can now be processed.

We denote $C = \{c_i \in \mathcal{V} | c_i = 1\}$ to represent the controller set, and for $\forall c_i \in C$, it has an assignment switch set $\mathcal{S}_i \subset \mathcal{V}$. For $\forall v_j \in \mathcal{S}_i$, we have $y_{ij} = 1$. If v_j should be reassigned to a new controller denoted by c_ξ at some time, the assignment matrix $\mathbf{Y}$ should be updated as

$$\begin{cases} y_{ij} \leftarrow 0 \\ y_{\xi j} \leftarrow 1 \end{cases} \tag{3}$$

Meanwhile, the flow tables concerning the reassigned switch should also be migrated to the new controller. The switch migration cost defined in [29] is the number of switches that need to migrate, which is much smaller than the number needed for migrating a controller because a controller generally contains flow tables of more switches. It can be roughly estimated that an Openflow switch can have 256 tables with 256 flow entries. Each rule record (flow entry) is on average 100 bytes. If all the tables of the switch are full, there will be about 6-10 MB of data. If the link's data rate is 1 Gbps, the switch migration time will be less than 80 ms. By comparison with the controller migration process illustrated in [21] and [33], the total data storage size of a controller is estimated to be 100 MB on average. Although the bandwidth of migration links is set to 1 Gbps (too large for current satellites), the time cost of controller migration is at least 800 ms, which will be larger if we consider the propagation delay between the previous and newly selected controllers.

C. Topological Dynamics and Snapshot Division

Due to the relative motion of satellites, a LEO satellite network has a dynamic topology. We take here a *Walker-Polar* constellation as an example to elaborate on. As shown in Fig. 4, before flying into the polar region at t_1, satellite S_0 can maintain inter-orbit ISLs L_1 and L_2 with S_1 and S_2, respectively. When these satellites enter the polar region at t_2, L_1 and L_2 will be temporarily switched off, because their range, azimuth and elevation angles change too fast and antenna tracking thus becomes difficult. After passing through the polar region, S_0 reestablishes L_1 and L_2 at t_3, but their targeting nodes are exchanged. Similarly, the ISL switch-off will still exist in a *Walker-δ* constellation, because adjacent orbits intersect each other at the latitude near the orbit inclination angle. For the sake of simplicity, we will likewise use the polar region to represent the orbit intersection area of *Walker-δ* constellations in the following paragraphs.

In order to deal with the dynamic topology, existing studies have proposed two kinds of methods, namely virtual node (VN) and virtual topology (VT) [34]. Although the VN methods shield the topological dynamics by dividing the Earth's surface into several cells and binding them with certain satellites, they cannot reflect the intrinsic changes in ISLs and, thus, the dynamic network topology. By contrast, VT methods maintain the topological changes by dividing the satellite motion period into several time snapshots, which is indeed an effective way to deal with CPP. However, dividing the snapshots evenly by a fixed time step Δt (as in [7])

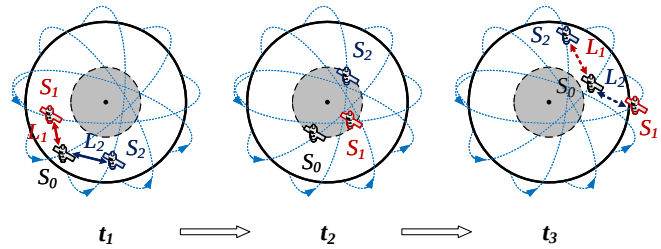


Fig. 4. Inter-orbit ISLs $L1$ and L_2 are switched off in the polar region (in grey) where two orbits intersect each other (from the view of the North/South pole).

might be unable to capture some topological configurations with a duration shorter than Δt. On the other hand, if we divide the time period by topological changes and choose one instance for each time snapshot [20], the network topology in each time period will be stable. Nonetheless, each time instance will contribute the same to the CPP model and, thus, the duration of each time period (which represents a steady network topological configuration) will be useless and cannot affect the CPP results.

To address the snapshot division problem, we modify the VT method in our model. Let us assume that the time period that a satellite cycles around the Earth is T, which is also the period of the constellation, and the network topology changes at a sequence of time instances T_i $(i = 1, 2, ...M)$ (M is the total number of time instances). Then the network snapshots are intervals between two consecutive time instances, and we use the snapshot length δ_i to represent it, which satisfies $\delta_i = T_{i+1} - T_i$. We first calculate the minimum snapshot length, namely $\delta_{min} = \min_{\forall i<M} \delta_i$, based on the method described in [32], where a virtual orbital plane is created, and all satellites in the LEO constellation are mapped onto it. Then we set the fixed time step that satisfies $\Delta t < \delta_{min}$. Thus, we can sample a sequence of time instances, i.e., t_k $(k = 1, 2, ...)$, and $t_{k+1} - t_k = \Delta t$. t_k will be the instances chosen for solving the CPP.

D. Arrival Traffic Estimation

When satellites move along their orbits, their traffic load will change with the coverage area, which may cause overload or idleness on switches and, specially, on controllers. Hence, it is important to estimate the time-varying traffic load of controllers.

According to [35], data packets arriving at an SDN switch can be divided into two types, i.e., local packets coming from the switch's own domain and forwarded packets from other neighboring switches. The local arrival rate of switch v_j within its coverage area at time instance t_k is denoted as λ_j^k, where $j = 1, 2, ..., N$. The probability that a packet belongs to a new flow and needs to be sent to the controller is assumed to be p_j^{nf}; thus the traffic sent from v_j to its controller is

$$\gamma_j^k = p_j^{nf} \lambda_j^k \tag{4}$$

The proportion of data that switch v_j forwards from switch v_m is q_{mj}. Because each switch has an independent coverage

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3184989

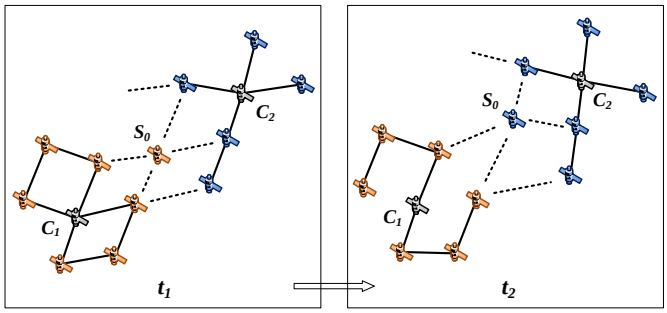


Fig. 5. Switches are reassigned to controllers when network topology changes. Here, solid and dashed lines represent ISLs and indirect connections of satellites, respectively.

area on the Earth and the traffic rates of each coverage area are independent for a certain time, any two switches will have independent arrival rates at the same time. Hence, the net data packet arrival rate of switch v_j is

$$\Lambda_j^k = \sum_{m=1}^{N} \lambda_m^k q_{mj} \tag{5}$$

where we assume that $q_{jj} = 1$.

Therefore, the traffic arrival rate of the controller c_i, denoted by Γ_i^k is given by

$$\Gamma_i^k = \sum_{j=1}^{N} \gamma_j^k y_{ij} \tag{6}$$

Once we obtain the real-time local arrival rate of switches (i.e., λ_j^k) and the switch-controller mapping y_{ij}, we can calculate the controller traffic load Γ_i^k. While y_{ij} needs further investigation, we can directly compute λ_j^k, which is the sum of user traffic within the coverage area of satellite v_j at time instance t_k.

IV. Static Placement and Dynamic Assignment Method

We aim to statically place controllers on fixed nodes while dynamically assigning switches to controllers. Therefore, the SPDA consists mainly of two parts: first, we choose appropriate controller locations within a constellation period; second, we calculate the dynamic switch-controller assignment according to the topological changes and traffic variations. We will illustrate this using the example of the switch reassignment process caused by topological dynamics. As shown in Fig. 5, where solid lines represent ISLs and dashed lines represent indirect connections, some satellites exist between S_0 and others. Satellite S_0 is initially managed by controller C_1 in t_1. When the topology changes between t_1 and t_2, the latency between S_0 and C_1 may be longer than that between S_0 and C_2. Hence, switch S_0 will be detached from the previous controller C_1 and reassigned to its new controller C_2 in t_2. On the other hand, the reassignment caused by traffic happens whether the controller is overloaded or low-load, so an evaluation process for traffic load should take place before calculation. The overall flow diagram of the SPDA algorithm is depicted in Fig. 6.

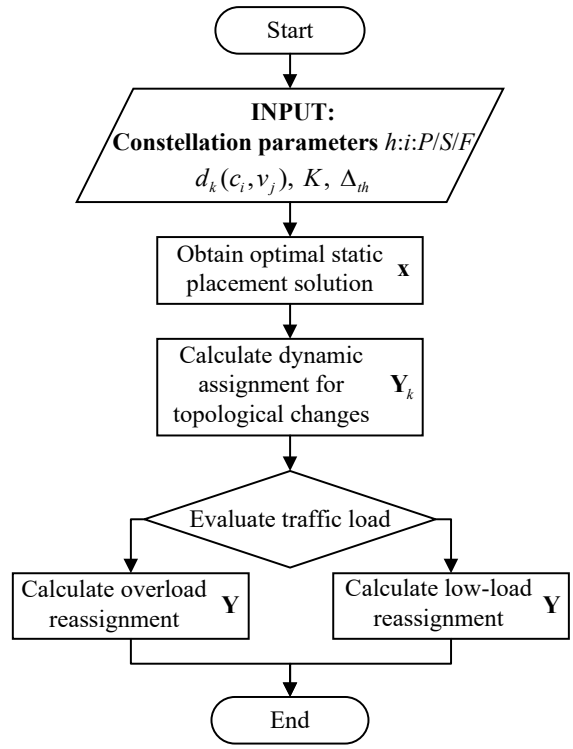


Fig. 6. Flow diagram of the SPDA algorithm.

In terms of the evaluation metrics, the switch-controller latency is most commonly used for CPP [5]. It is also intrinsically equivalent to the flow setup time defined in [7], which is the sum of twice the switch-controller propagation latency plus the end-to-end flow forwarding latency, because the latter actually has no effect on controller placement. We adopt the switch-controller latency as the evaluation metric, and consider the optimal trade-off between the worst-case and average-case of latency. Although heuristic algorithms can find feasible solutions [4], we apply the linear programming framework and formulate the CPP as a MIP model.

A. Static Placement Problem Formulation

The latency between controller c_i and switch v_j will change when satellites are orbiting, and it is denoted as $d_k(c_i, v_j)$ in time instance t_k. We use the satellite orbit elements to generate the orbit files containing satellite positions at each time instance, and the switch-controller latency can thus be calculated. We use f_1 to represent the average latency of switch-controller pairs in all time instances:

$$f_1 = \frac{1}{|\mathcal{V}|} \sum_{v_j \in \mathcal{V}} \bar{d}(c_i, v_j) y_{ij} \tag{7}$$

where $\bar{d}(c_i, v_j) = \frac{1}{T} \sum_k d_k(c_i, v_j)$. Thus, the first objective is to minimize f_1.

Similarly, we denote f_2 as the maximum latency of switch-controller pairs in all time instances:

$$f_2 = \max_{i,j} d^{\max}(c_i, v_j) y_{ij} \tag{8}$$

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3184989

where $d^{\max}(c_i, v_j) = \max_k d_k(c_i, v_j)$. The second objective aims to minimize f_2, i.e., $\min(\max_{i,j} d^{\max}(c_i, v_j) y_{ij})$. Due to the $\min\max(*)$ function, f_2 needs to be linearized by introducing a new continuous variable z, which satisfies $f_2 = z$. Thus, new constraints should be added as follows:

$$z \geq d^{\max}(c_i, v_j) y_{ij}, \ \forall c_i, v_j \in \mathcal{V} \tag{9}$$

The constraints come from several different aspects [4]. First, the total number of controllers K should be:

$$\sum_{i=1}^{N} x_i = K \tag{10}$$

Second, we need to ensure that one switch is assigned to exactly one controller:

$$\forall v_j \in \mathcal{V}, \ \sum_{c_i \in \mathcal{V}} y_{ij} = 1 \tag{11}$$

Third, c_i should exist if switch v_j is assigned to it:

$$\forall c_i, v_j \in \mathcal{V}, \ y_{ij} \leq x_i \tag{12}$$

Fourth, the latency between each switch-controller pair should be smaller than a preset threshold Δ_{th}:

$$\forall c_i, v_j \in \mathcal{V}, t_k \in T, \ d_k(c_i, v_j) y_{ij} \leq \Delta_{th} \tag{13}$$

Here, Δ_{th} is introduced because the switch-controller latency should be as small as possible, and a threshold can reduce the feasible solution space to accelerate the optimization process.

The static placement problem can then be transformed into the following standard linear programming formulation:

$$\begin{aligned}
&\text{find } \mathbf{x}, \mathbf{Y}, z \\
&\min \begin{cases} f_1 = \frac{1}{|\mathcal{V}|} \sum_{v_j \in \mathcal{V}} \bar{d}(c_i, v_j) y_{ij} \\ f_2 = z \end{cases} \\
&\text{s.t.} \begin{cases} \sum_{i=1}^{N} x_i = K \\ \sum_{c_i \in \mathcal{V}} y_{ij} = 1, \ \forall v_j \in \mathcal{V} \\ -x_i + y_{ij} \leq 0, \ \forall c_i, v_j \in \mathcal{V} \\ d_k(c_i, v_j) y_{ij} \leq \Delta_{th}, \ \forall c_i, v_j \in \mathcal{V}, t_k \in T \\ d^{\max}(c_i, v_j) y_{ij} - z \leq 0, \ \forall c_i, v_j \in \mathcal{V} \end{cases}
\end{aligned} \tag{14}$$

Furthermore, we can define a total objective function $\mathbf{f} = w_1 f_1 + w_2 f_2$ to demonstrate the trade-off between f_1 and f_2, where w_1 and w_2 are their weight coefficients. Given the MIP model in Eqn. (14), we will use a linear programming solver called Gurobi to deal with it. This will be elaborated on in Section V.

B. Dynamic Assignment for Topological Changes

When network topology changes drastically, switch-controller latency may become much longer at some point, which requires the implementation of dynamic assignment to reduce latency. In other words, when controllers enter or leave the polar region, switches probably need to be reassigned to new controllers. Because satellites move regularly and periodically, we can pre-calculate the dynamic assignment to conserve on-orbit computing resources. With the static controller placement result $\mathbf{x}$, we calculate the time periods that require dynamic assignment. Based on that, we develop a shortest-path based dynamic assignment (SDA) algorithm to calculate the new assignment y_{ij} in those time instances while keeping x_i constant.

Algorithm 1 Shortest-path based dynamic assignment.

Input: $\mathbf{x}$, $d_k(c_i, v_j)$
Output: $\mathbf{Y}_k$

```
ResNode_m ← 1 ∀m ∈ [1, N], Hops ← 1
for m = 1 to N do
    if x_m = 1 then
        ResNode_m ← 0
    end if
end for
while Σ_m ResNode_m ≠ 0 do
    for all c_i ∈ C do
        Find neighboring set of c_i within Hops, AdjSat
        for all v_j ∈ AdjSat do
            PathLen ← ShortestPath(d_k(c_i, v_j), x, v_j)
            if ResNode_j ≠ 0 then
                ResNode_j ← 0, and assign v_j to controller c_i
            else
                Calculate current path length of v_j, OLen
                if PathLen < OLen then
                    Assign v_j to controller c_i
                end if
            end if
            Update Y_k
        end for
    end for
    Hops ← Hops + 1
end while
```

The SDA algorithm is executed for each time instance t_k that requires dynamic assignment. As illustrated in Algorithm 1, SDA uses $ResNode$ to label the residual switches that are not assigned and $Hops$ to represent the hops between two satellite nodes. According to Eqn. (1), $ResNode_m$ is initially set to 0 if controller c_m exists; otherwise, it is set to 1. In Algorithm 1, K controllers find their assigned switches simultaneously within a given number of hops. While there exist unassigned switches, we first find the neighboring node set, $AdjSat$, within given $Hops$ for a certain controller $c_i \in C$. Then for each $v_j \in AdjSat$, we use the shortest path algorithm to calculate the current $PathLen$, which is recorded as the shortest if node v_j is unassigned; otherwise, we compare $PathLen$ with the current path length $OLen$ and choose the best controller. Once we obtain the best controller for v_j, we update the switch-controller mapping $\mathbf{Y}_k$. The computational complexity of SDA is $O(N)$, which is better than that of traversing all node pairs, i.e., $O(N^2)$.

C. Dynamic Assignment for Load Balancing

The traffic load of controllers varies when satellites fly over different terrestrial regions and may sometimes exceed the capacity of controllers, thereby causing traffic congestion. Moreover, a significant amount of bursty traffic may also incur controller overload, which must be handled by a dynamic

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3184989

assignment process. On the other hand, a controller with low traffic load could be shut down to save power, thus requiring reassignment of its switches. Therefore, we expect to keep controller traffic balanced; maintain acceptable latency between controllers and newly-reassigned switches; and reduce the reassignment times as much as possible.

Unlike the optimal SDA method implemented in dynamic assignment due to topology, we develop heuristic algorithms for load balancing to obtain the dynamic assignment solutions instead of optimal balancing. The reasons are stated as follows:

- Optimal methods generally consume large computational resources. Although the SDA algorithm can be executed before satellite launch, load balancing is still needed for practical operations. Therefore, optimal methods may not be a reasonable choice for load balancing.
- Optimal balancing may require multiple switch migrations, which also increases total migration cost. Instead, a relatively good balancing result with fewer migrations is more suitable for our case.
- Optimal balancing may lead to higher latency results. It is difficult to achieve both low latency and load balancing at the same time, so, to some extent, it is unnecessary to seek optimal balancing.

Therefore, we will consider both latency and load balancing in the dynamic assignment of switches, and discuss the trade-off between them. The problem with dynamically assigning switches for load balancing has two aspects: *overload* and *idleness*. The former requires migrating switches from overloaded controllers to those with low load in order to maintain load balancing. The latter requires shutting down idle or low-load controllers and migrating their switches to neighboring controllers to save energy. Hence, we can assume that the threshold traffic load for a controller is either Γ_{max} or Γ_{min} for, respectively, overload or idleness.

The case of overload can be assumed to happen to a controller c_i for a certain time period $[t_a, t_b]$, and one or more of its switches should be reassigned to a new controller; in other words, given that the switch set of c_i is $\mathcal{S}_i \subset \mathcal{V}$, our aim is to determine which switches (e.g., $v_m \in \mathcal{S}_i$) should be reassigned to which controller (e.g., c_ξ). The time period $[t_a, t_b]$ can be determined by comparing the real traffic load of controllers with the thresholds Γ_{max} and Γ_{min}. It can be inferred that the chosen switch v_j should have a relatively large amount of traffic, and thus it can significantly influence the traffic load of c_i. On the other hand, the reassignment should not increase the switch-controller latency by too much, which means that the latency between v_j and c_ξ should not be much longer than the latency between v_j and c_i. Therefore, the chosen switch v_j should have a relatively long switch-controller latency before the migration. Based on that, we have developed a load-latency product-based overload reassignment (LPOR) algorithm as shown in Algorithm 2. The computational complexity of LPOR is $O(N^2)$ for the worst case.

There are three main steps for the overload case in Algorithm 2. First, we aim to find the most suitable switch that needs to be reassigned. Given that the latency between c_i and v_j in time instance t_k is $d_k(c_i, v_j)$, and the amount of traffic

Algorithm 2 Load-latency product-based overload reassignment.

Input: $\mathbf{x}$, $\mathbf{Y}$, c_i, $[t_a, t_b]$
Output: $\mathbf{Y}$

1: Calculate the average traffic load of c_i, Γ_i
2: **while** $\Gamma_i \geq \Gamma_{max}$ **do**
3: Calculate $\bar{d}_\lambda(c_i, v_j)$ according to Eqn. (16)
4: $\mathcal{C}_{ex} = \{c_i\}$, $\mathcal{V}_{ex} = \emptyset$
5: Find the potential switch $v_m \leftarrow \arg\max \bar{d}_\lambda(c_i, v_j)$, where $v_m \in \mathcal{V} - \mathcal{V}_{ex}$
6: $\forall c_\chi \in \mathcal{C} - \mathcal{C}_{ex}$, calculate $\bar{d}_{ab}(c_\chi, v_m)$ according to Eqn. (18)
7: Find the best controller $c_\xi \leftarrow \arg\min \bar{d}_{ab}(c_\chi, v_m)$
8: **while** $t_k \in [t_a, t_b]$ **do**
9: **if** $d_k(c_\xi, v_m) \leq \Delta_{th}$ **then**
10: $\mathcal{V}_{ex} \leftarrow \mathcal{V}_{ex} + v_m$, **Goto** Line 5
11: **end if**
12: **end while**
13: Calculate the average traffic load of c_ξ, Γ_ξ
14: **if** $\Gamma_\xi \geq \Gamma_{max}$ **then**
15: $\mathcal{C}_{ex} \leftarrow \mathcal{C}_{ex} + c_\xi$, **Goto** Line 7
16: **else**
17: Update $\mathbf{Y}$ according to Eqn. (3)
18: **end if**
19: Calculate the average traffic load of c_i, Γ_i
20: **end while**

sent from v_j to c_i is γ_j^k, we have the traffic-weighted latency of switch-controller pair c_i and v_j,

$$d_\lambda(c_i, v_j, k) = d_k(c_i, v_j)\gamma_j^k \tag{15}$$

Thus, we can define a load-latency product function in $[t_a, t_b]$:

$$\bar{d}_\lambda(c_i, v_j) = \frac{1}{N_t} \sum_{t_k \in [t_a, t_b]} d_\lambda(c_i, v_j, k) \tag{16}$$

where N_t is the number of time instances in $[t_a, t_b]$. We adopt $\bar{d}_\lambda(c_i, v_j)$ to evaluate the suitability of the switch reassignment before implementing the dynamic assignment method, and the best switch v_m is thus determined by

$$v_m = \arg\max_{v_j \in \mathcal{V}} \bar{d}_\lambda(c_i, v_j) \tag{17}$$

After determining the switch v_m that requires reassignment, the second step is to find a new controller for v_m. We calculate the average latency between v_m and other controllers (e.g., c_χ) in $[t_a, t_b]$:

$$\bar{d}_{ab}(c_\chi, v_j) = \frac{1}{N_t} \sum_{t_k \in [t_a, t_b]} d_k(c_\chi, v_m) \tag{18}$$

The potential controller denoted by c_ξ, to which v_m is reassigned, should have relatively small latency and is thus determined by

$$c_\xi = \arg\min_{c_\chi \in \mathcal{C}} \bar{d}_{ab}(c_\chi, v_m) \tag{19}$$

We then check the switch-controller latency between c_ξ and v_m to find whether it is smaller than the threshold Δ_{th}. The

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3184989

algorithm will go back to find another switch if the threshold constraint is not satisfied. Since the LPOR algorithm adopts a load-latency product function, the latency between c_ξ and v_m is expected to be only slightly longer than the latency between c_i and v_m, which generally guarantees the threshold requirement. On the other hand, we need to ensure that the traffic load of new controller c_ξ after reassignment does not exceed the limitation Γ_{max}. We can calculate the average traffic load of c_ξ in $[t_a, t_b]$ according to Eqn. (6),

$$\Gamma_\xi = \frac{1}{N_t} \sum_{t_k \in [t_a, t_b]} \Gamma_\xi^k \tag{20}$$

If Γ_ξ exceeds the traffic limitation, we repeat the second step and choose another controller.

The final step is to migrate switch v_j from c_i to c_ξ according to Eqn. (3). After switch migration, we calculate the average traffic load of c_i in $[t_a, t_b]$, i.e., $\Gamma_i = \frac{1}{N_t} \sum_{t_k \in [t_a, t_b]} \Gamma_i^k$. If Γ_i still exceeds the traffic limitation, we return to the first step and find another switch in $\mathcal{S}_i$ to operate.

Algorithm 3 Minimal-latency based low-load reassignment.

Input: $\mathbf{x}$, $\mathbf{Y}$, c_i, $[t_a, t_b]$
Output: $\mathbf{Y}$

1: **for all** $v_j \in \mathcal{S}_i$ **do**
2: $\quad C_{ex} = \{c_i\}$
3: $\quad \forall c_\chi \in C - C_{ex}$, calculate $\bar{d}_{ab}(c_\chi, v_j)$ according to Eqn. (18)
4: $\quad$ Find the best controller $c_\xi \leftarrow \arg\min \bar{d}_{ab}(c_\chi, v_j)$
5: $\quad$ Calculate the average traffic load of c_ξ, Γ_ξ
6: $\quad$ **if** $\Gamma_\xi \geq \Gamma_{max}$ or $d_k(c_\xi, v_m) \geq \Delta_{th}$ **then**
7: $\quad\quad C_{ex} \leftarrow C_{ex} + c_\xi$
8: $\quad\quad$ **Goto** Line 4
9: $\quad$ **else**
10: $\quad\quad$ Update $\mathbf{Y}$ according to Eqn. (3)
11: $\quad$ **end if**
12: **end for**

Similarly, we develop the minimal-latency based low-load reassignment (MLR) algorithm for the idleness case, as shown in Algorithm 3. The computational complexity of MLR is $O(N^2)$ for the worst case. Given a controller c_i that is idle or has low load and for a switch v_j within its domain, i.e., $v_j \in \mathcal{S}_i$, we first use Eqn. (18) to calculate $\bar{d}_{ab}(c_\chi, v_j)$, where c_χ is a controller that is not c_i. Second, we find the best controller c_ξ that has the minimum value of $\bar{d}_{ab}(c_\chi, v_j)$. Third, we calculate the average traffic load of the best controller, i.e., Γ_ξ. If Γ_ξ is not smaller than Γ_{max} or the latency threshold is exceeded, we continue trying to find another controller until these two conditions are satisfied. Hence, we update the switch-controller mapping $\mathbf{Y}$ according to Eqn. (3) and finish the low-load reassignment.

V. Case Study

In this section, we will provide the simulation settings and will then evaluate the performance of the SPDA algorithm specifically in terms of three aspects: the static placement scheme; the reduction in switch-controller latency; and the load-balancing effects.

TABLE I
Parameters of the constellations

Constellation	N	P	S	h(km)	i(°)
Iridium	66	6	11	780.0	86.4
Celestri	63	7	9	1400.0	48.0

A. Simulation Settings

We adopt Iridium and Celestri (a *Walker-Polar* and *Walker-δ* constellation, respectively) to evaluate our SPDA method. The parameters of the two constellations as defined in Section III-A are listed in Table I, and the phase factor F of Celestri is set to 1 for simplicity. Based on the constellation parameters, we develop a MATLAB program script to automatically generate the satellite orbit elements in STK, an aerospace simulation software, and export the orbit files containing satellite positions at each time instance. As a result, we can calculate the switch-controller latency by adopting the orbit files. We then build the MIP model on a computer with Intel Core i7-6560U CPU @2.2 GHz and 8 GB RAM, and then use Gurobi 9.0.1 to solve the static placement model as illustrated in Eqn. (14). The simulation settings for Iridium and Celestri, respectively, are given as follows:

- The polar region is defined by a latitude boundary, β, which is smaller than i. Here, we set β to 80° and 45°, respectively.
- The simulation time is set to the period of constellations T, i.e., 6030 s and 6840 s. The minimum time interval of snapshots, δ_{min}, is calculated as 38.6 s and 24.8 s.
- The fixed step for sampling time instances, Δt, is set to 30 s and 20 s to obtain a more precise snapshot division. Subsequently, the number of time instances is calculated as 202 and 343.
- Because there is at least one controller in the satellite network, the geometric distance of any switch and controller is at most half of the orbit perimeter, i.e., about 20000 km and 24000 km for Iridium and Celestri, which results in a latency of about 0.06 s and 0.08 s given the lightspeed 3×10^5 km/s. Although multiple hops occur through ISLs between switches and controllers and thus cause longer delays, it is reasonable to set the switch-controller latency threshold, Δ_{th}, at 60 ms and 80 ms for Iridium and Celestri, respectively, without loss of generality.
- The number of controllers, K, is initially set to 6, and this will change in our following simulations.

It is reasonable to use the world population distribution to represent the arrival traffic of users [36]. The world population density data with resolution of 1° are generated from [37], which is the first version in 2020. We use a log scale map to illustrate it as shown in Fig. 7. We denote $\mathbf{P}$ as the world population distribution matrix, and p_s will be an element of $\mathbf{P}$, where the subscript s denotes the element location (see [36]). Assuming that the coverage area of v_j is $\mathcal{B}_j$, whose position and range can be calculated through constellation parameters, the traffic load of satellite v_j in t_k, namely λ_j^k, is therefore

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3184989

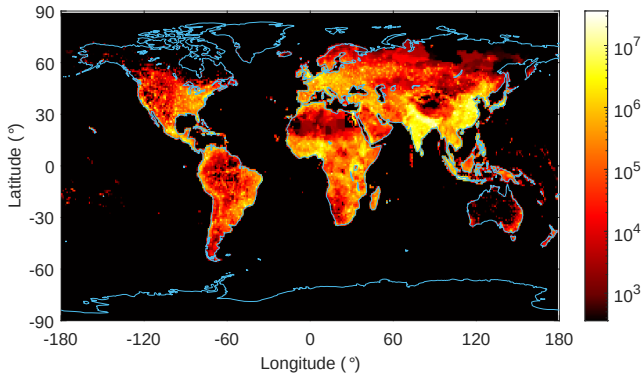


Fig. 7. The world population density distribution in 2020. Colors represent the population in an area of 1° by 1°.

measured by

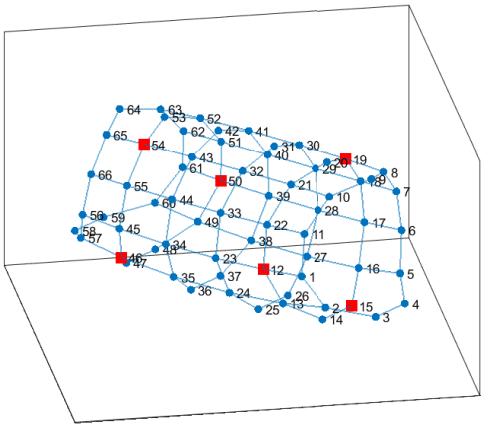
$$\lambda_j^k = \sum_{s \in \mathcal{B}_j} \epsilon p_s \tag{21}$$

where ϵ is the traffic demand of a single user, which can be set at 100 Kbps [36]. In order to test the load balancing performance of both LPOR and MLR, the traffic limitations of a controller, i.e., Γ_{min} and Γ_{max} are set at 1% and 200% of the average value of the overall traffic load, respectively.

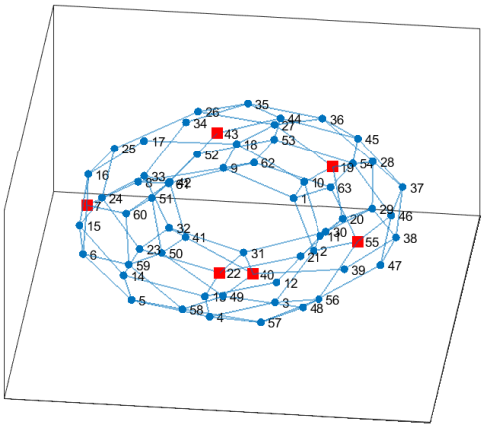
B. Results and Discussion

We first verify the static placement model, and then show the results of both the Iridium and Celestri in Fig. 8, where the abstracted network topology is captured in a certain time instance. The satellites are numbered globally from 1 to N for the sake of simplicity, and the red squares represent controllers while the blue circles are switches. Fig. 8 (a) clearly has 6 orbit planes, with 6 controllers (Nos. 12, 15, 19, 46, 50, 54) deployed in 2 orbit planes, i.e., $P = 2$ and $P = 5$, respectively. In each plane, three satellites are distributed evenly. All the chosen controllers are in different phase for each plane, which ensures that no two of them will come into the polar regions at the same time. In Fig. 8 (b), the network topology seems more complicated with 6 controllers (Nos. 7, 19, 22, 40, 43, 55) distributed in 4 planes: one for each plane of $P = 1$ and $P = 7$, and two for each plane of $P = 3$ and $P = 5$. The locations of these controllers are also symmetric and in different phases.

Fig. 9 shows the plots for the two objective functions f_1 and f_2 in Eqn. (14), using one period for each constellation. For the Iridium case in Fig. 9 (a), f_1 is about 20 ms while f_2 mostly remains at about 44 ms; nonetheless, f_2 can sometimes reach around 52 ms. In terms of Celestri in Fig. 9 (b), f_1 is about 30 ms while f_2 is usually between 55 and 60 ms, but it sometimes exceeds 75 ms. The peak values in both Iridium and Celestri are due to controllers entering polar regions, where inter-orbit ISLs are switched off and thus cause longer switch-controller latencies. The results reveal good periodic variations in f_1 and f_2 for both Iridium and Celestri. It should be noticed that both the average and maximum latencies of Iridium are smaller than those of Celestri, which can be explained by the length of the ISLs: Iridium generally has shorter ISLs than Celestri. To sum up, our SCP model shows good results in spite of the latency peaks caused by the dynamic topology of satellite networks.



(a) Iridium

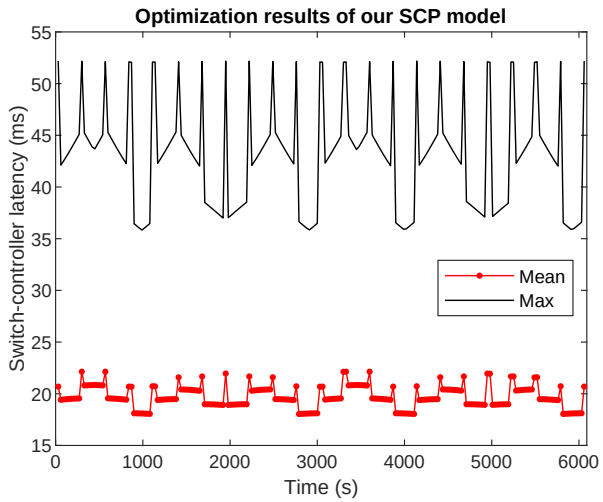


(b) Celestri

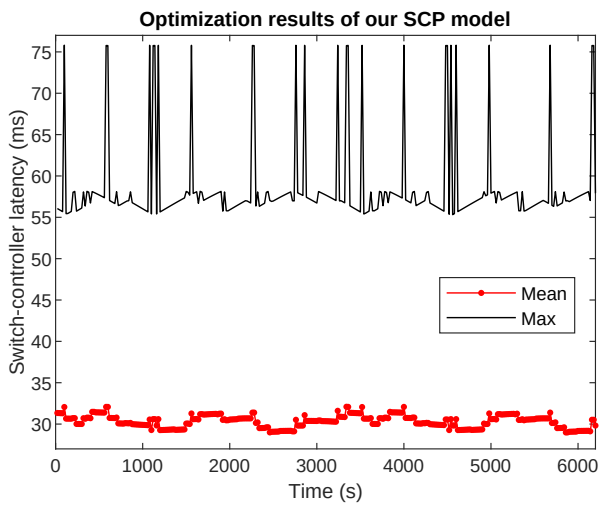
Fig. 8. Static placement results of Iridium and Celestri constellations illustrated in graphic topology. The red squares are regarded as controllers.

In order to evaluate the trade-off between the average and worst cases in our static placement model, we set the number of controllers $K = 8$ and change the weight coefficients w_1 and w_2 for the two objectives. As the values of f_1 and f_2 are on the same order of magnitude, w_1 and w_2 should differentiate the two objectives from each other. We also need to guarantee that the value of the total objective $\mathbf{f}$ is normalized so that we can set three pairs of weights: $(1, 9)$, $(5, 5)$, $(9, 1)$ for (w_1, w_2). Moreover, because Gurobi solver also has an integer priority (default value is 0), that optimizes for the objectives in decreasing priority order, we can set the priority parameter for f_1 and f_2 at either 0 or 1. Three cases for the priority should be considered: $(0, 0)$, $(1, 0)$, $(0, 1)$ for (f_1, f_2), since $(1, 1)$ is the same as $(0, 0)$. Consequently, there are 9 cases for the trade-off evaluation in terms of the weights and priorities.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3184989

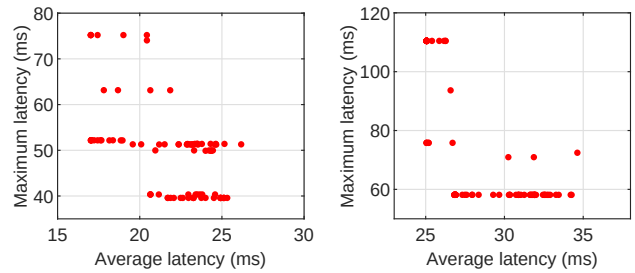


(a) Iridium



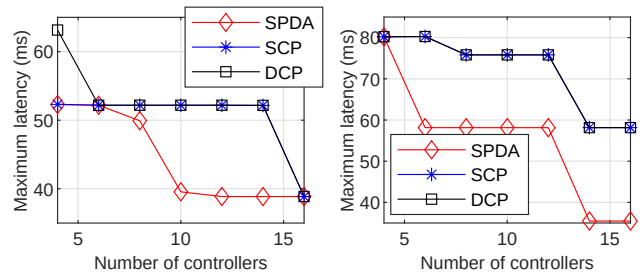
(b) Celestri

Fig. 9. Static placement results for Iridium and Celestri.

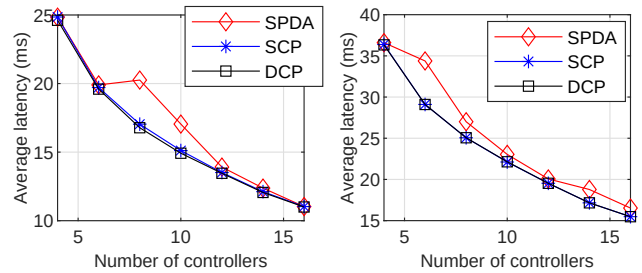


(a) Iridium (b) Celestri

Fig. 10. Trade-off between the worst and average latencies.



(a) Maximum for Iridium (b) Maximum for Celestri



(c) Average for Iridium (d) Average for Celestri

Fig. 11. Relationships between switch-controller latency and the number of controllers.

Accordingly, the MIP optimizer in Gurobi is run independently 9 times, and we record the 10 best solutions for each trial, resulting in a total of 90 solutions, which are shown in Fig. 10. Each red point represents a solution, and the results are approximately distributed across several lines quantized by values on the y-axis. This is because the maximum latency reflects the number of hops between controllers and switches. Moreover, the Iridium case generally shows lower latency over the Celestri case in both maximum and average values. It is reasonable that the maximum latency remains stable while the average latency varies significantly for different controller placement schemes. Therefore, our goal is to reduce the average latency while maintaining a relatively low maximum latency.

Based on the static placement solutions, we now adopt the SDA algorithm to improve our results by considering the dynamic network topology. We calculate the best controller placement schemes for different K (the number of controllers) and then compare our results with those of the SCP and DCP methods reported in [7], [20]. The relationships between the maximum/average switch-controller latencies and numbers of controllers are shown in Fig. 11.

When K increases, the maximum latency of all three methods tends to decrease but also remains stable at some value of K for both the Iridium and Celestri cases in Fig. 11 (a) and (b); and the average latency has a monotonously decreasing trend as shown in Fig. 11 (c) and (d). On the whole, it can be seen that the SPDA method evidently reduces the maximum latency in contrast to SCP and DCP while it slightly increases the average latency for both Iridium and Celestri. The reason for this is that addressing the longer latency caused by the dynamic topology may result in the SDA algorithm increasing the latency of some nodes, which thereby increases the average latency. At the cost of a small increment in the average latency, our SPDA method produces a marked effect in terms of the switch-controller latency. Moreover, with the same number of controllers, both the average and maximum cases of Iridium always have lower latencies than those of Celestri, because the ISL length, determined by each constellation, is generally

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3184989

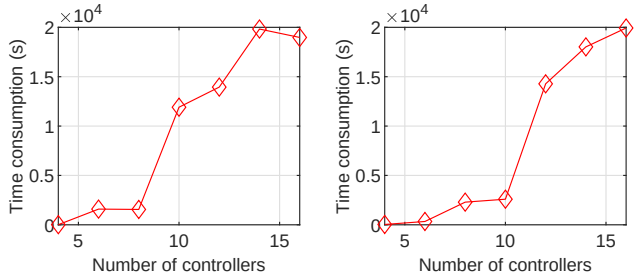


Fig. 12. Relationships between time consumption and number of controllers (on a computer with Intel Core i7-6500U CPU @2.2 GHz and 8 GB RAM).

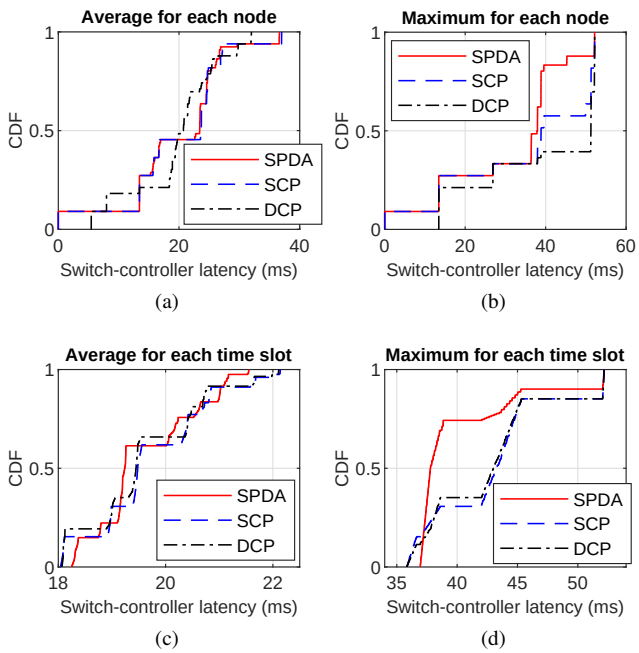


Fig. 13. CDF of switch-controller latency in Iridium.

shorter for Iridium than for Celestri.

We also present the dependence between the computational time consumption of SPDA method and the number of controllers, which is shown in Fig. 12. As the number of controllers increases, the time consumption remains at a relatively low level (below 300 s) when K is relatively small. Because more controllers bring about much more feasible solutions, the time consumption leaps up to more than 1×10^4 s when $K \geq 10$ or $K \geq 12$ for Iridium and Celestri, respectively.

More specifically, we explore the results of $K = 6$ and compare SPDA with existing SCP and DCP methods regarding the cumulative density function (CDF) of both the average and maximum switch-controller latencies. The results are presented in Figs. 13 and 14 for Iridium and Celestri, respectively. Sub-figures (a) and (b) show the average and maximum latencies of each node for all time instances, while sub-figures (c) and (d) show those of each time instance for all nodes.

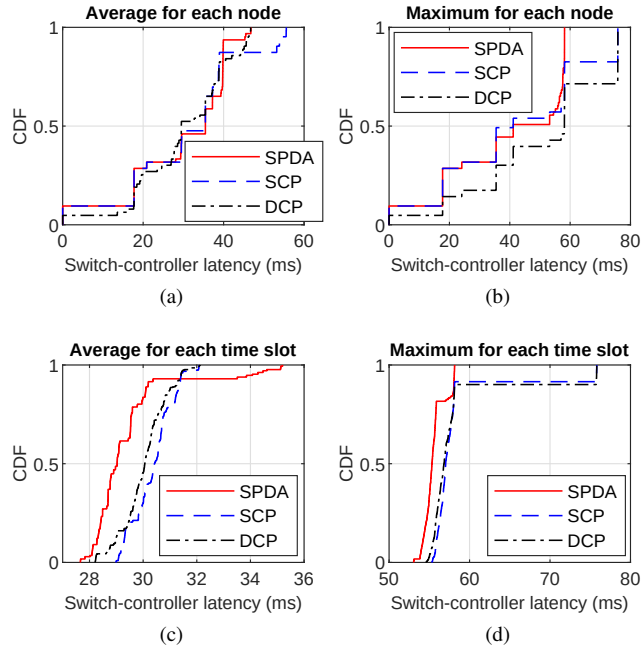


Fig. 14. CDF of switch-controller latency in Celestri.

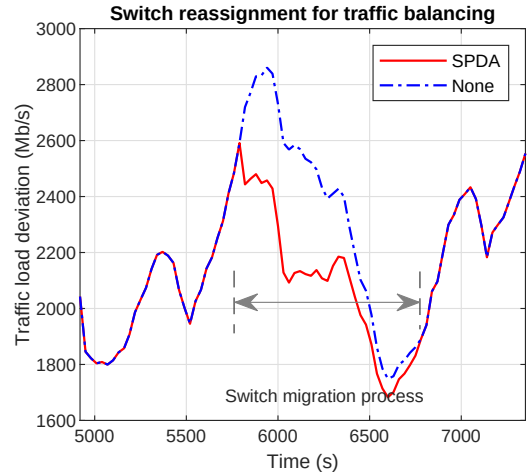
For the average case of Iridium shown in Fig. 13 (a) and (c), the three methods generate almost the same results, which means that SPDA can achieve similar performance to SCP and DCP in terms of the average switch-controller latency. However, for the maximum case, both SCP and DCP are bounded by SPDA as shown in Fig. 13 (b). About 80% nodes have a switch-controller latency lower than 40 ms for SPDA, while this is the case for only, at most, 55% and 40% nodes, respectively, with SCP and DCP. According to Fig. 13 (d), although no time instance has a latency lower than 37 ms for SPDA, 80% of them are between 37 and 39 ms. By contrast, only 40% of time instances are lower than 39 ms for SCP and DCP. The ratio of the maximum to average latency is on the order of about 1.25 to 2 for not only each node but also for each time instance. Therefore, we can conclude that the SPDA performs better than both SCP and DCP.

In Fig. 14 (a) and (c), the largest average latency for each node in SPDA is smaller than that of SCP and DCP; and SPDA generates about 90% of time instances with a much smaller average latency, i.e., 30 ms. In Fig. 14 (b) and (d), SPDA also incurs lower maximum latency for not only each node but also each time instance. When comparing Figs. 13 and 14, we can see that the maximum switch-controller latency for Iridium is smaller than that for Celestri, which possibly results from the topology difference between them. Moreover, SPDA has better performance in the maximum case for Iridium while this is true only for the average case with Celestri, which could be an interesting question for future investigations. Overall, SPDA has the best performance in terms of the dynamic network topology because of the improved time division method, and it also has a relatively low bandwidth utilization rate as well as low computational costs. In a word, SPDA is applicable to both *Walker-δ* and *Walker-Polar* constellations.

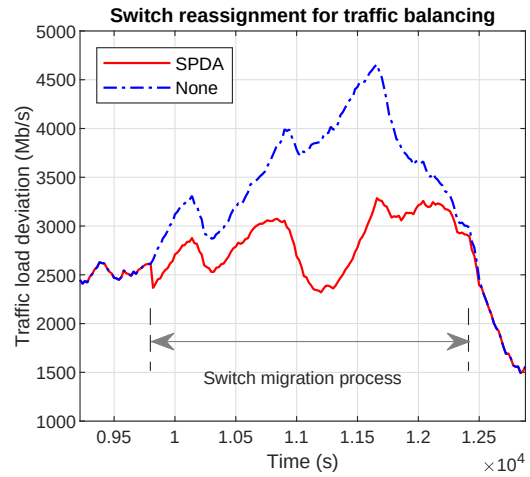
This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3184989

The load balancing results are shown in Fig. 15, which depicts the example of a period of time during which the switch migration occurs. The latency increment due to load-balancing is verified to be only at most 2% for all time instances during the switch migration process, which can still satisfy the threshold constraint. This small latency increment is because the proposed load-balancing reassignment algorithms keep the original assignment and choose the best and nearest alternative controller for migrating switches. Moreover, the load-latency product function defined in Eqn. (16) tends to choose a switch with longer latency as well as heavier traffic load, of which the latency will not change too much after migrations, because it is often far from both the original and the new controller; by contrast, a switch with shorter latency is always near the original controller but far from other controllers, and if the switch is chosen for reassignment the latency will change significantly after migrations.

In Fig. 15 (a), when the traffic of controller No. 12 reaches the maximum limitation Γ_{max} at about 5800 s (the start of time period $[t_a, t_b]$), our SPDA method executes the LPOR algorithm to decrease the traffic load deviation of controllers. The traffic load deviation is reduced to at most 20% at about 6100 s, and it always remains lower than the case of no SPDA until the period $[t_a, t_b]$ ends. In Fig. 15 (b), the switch migration process starts at about 9800 s and ends at 12400 s with a traffic load deviation reduction of at most 35% at 11200 s. It can be concluded that our proposed algorithm has a significant effect on traffic load balancing.



(a) Iridium



(b) Celestri

Fig. 15. Switch reassignment for traffic load balancing.

VI. Conclusion

Considering the limited bandwidth and dynamic topology in LEO satellite networks, we propose using the SPDA method to statically place controllers and dynamically adjust the switch-controller assignments. To the best of our knowledge, this is the first method that combines static placement and dynamic assignment for SDN controllers. To test the method, we have formulated the static placement problem into a MIP model and developed heuristic algorithms to assign switches effectively in terms of dynamic topology and load balancing. The simulation results show that SPDA outperforms both SCP and DCP, and our method also has good performance in terms of load balancing. The trade-off between the average and worst cases of latency has also been evaluated, and the maximum latency remains stable while the average latency changes greatly with different controller placement schemes. Furthermore, the average latency decreases while it is difficult to reduce the maximum latency with more controllers. Future works on the CPP in LEO satellite networks could involve the evaluation of node reliability and other controller placement metrics. Mega-constellations and multi-layer complex constellations are also worth investigating in future works.

References

[1] Y. G. Bi, G. J. Han, S. Xu, X. W. Wang, C. Lin, Z. B. Yu, and P. Y. Sun, "Software defined space-terrestrial integrated networks: Architecture, challenges, and solutions," *IEEE Network*, vol. 33, no. 1, pp. 22–28, 2019.

[2] L. Bertaux, S. Medjiah, P. Berthou, S. Abdellatif, A. Hakiri, P. Gelard, F. Planchou, and M. Bruyere, "Software defined networking and virtualization for broadband satellite networks," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 54–60, 2015.

[3] R. Ferrus, H. Koumaras, O. Sallent, G. Agapiou, T. Rasheed, M. A. Kourtis, C. Boustie, P. Gelard, and T. Ahmed, "SDN/NFV-enabled satellite communications networks: Opportunities, scenarios and challenges," *Physical Communication*, vol. 18, pp. 95–112, 2016.

[4] T. Das, V. Sridharan, and M. Gurusamy, "A survey on controller placement in SDN," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 472–503, 2020.

[5] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12. ACM, 2012, Conference Proceedings, pp. 7–12.

[6] Y. N. Hu, W. D. Wang, X. Y. Gong, X. R. Que, and S. D. Cheng, "On reliability-optimized controller placement for software-defined networks," *China Communications*, vol. 11, no. 2, pp. 38–54, 2014.

[7] A. Papa, T. D. Cola, P. Vizarreta, M. He, C. M. Machuca, and W. Kellerer, "Dynamic SDN controller placement in a LEO constellation satellite network," in *IEEE Global Communications Conference (GLOBECOM)*, 2018, Conference Proceedings, pp. 206–212.

[8] Y. Zhang, N. Beheshti, and M. Tatipamula, "On resilience of split-architecture networks," in *IEEE Global Telecommunications Conference (GLOBECOM)*, 2011, Conference Proceedings, pp. 1–6.

[9] J. Liu, Y. Shi, L. Zhao, Y. Cao, W. Sun, and N. Kato, "Joint placement of controllers and gateways in SDN-enabled 5G-satellite integrated

network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 2, pp. 221–232, 2018.

[10] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dynamic controller provisioning in software defined networks," in *9th International Conference on Network and Service Management*, 2013, Conference Proceedings, pp. 18–25.

[11] G. Yao, J. Bi, Y. Li, and L. Guo, "On the capacitated controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 18, no. 8, pp. 1339–1342, 2014.

[12] Y. Jiménez, C. Cervelló-Pastor, and A. J. García, "On the controller placement for designing a distributed sdn control layer," in *2014 IFIP Networking Conference*, 2014, Conference Proceedings, pp. 1–9.

[13] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic approaches to the controller placement problem in large scale sdn networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4–17, 2015.

[14] T. Y. Cheng, M. Wang, and X. Jia, "Qos-guaranteed controller placement in sdn," in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015, Conference Proceedings, pp. 1–6.

[15] N. Perrot and T. Reynaud, "Optimal placement of controllers in a resilient sdn architecture," in *2016 12th International Conference on the Design of Reliable Communication Networks (DRCN)*, 2016, Conference Proceedings, pp. 145–151.

[16] Y. Hu, T. Luo, N. C. Beaulieu, and C. X. Deng, "The energy-aware controller placement problem in software defined networks," *IEEE Communications Letters*, vol. 21, no. 4, pp. 741–744, 2017.

[17] G. D. Wang, Y. X. Zhao, J. Huang, and W. Wang, "The controller placement problem in software defined networking: A survey," *IEEE Network*, vol. 31, no. 5, pp. 21–27, 2017.

[18] M. He, A. Basta, A. Blenk, and W. Kellerer, "Modeling flow setup time for controller placement in SDN: Evaluation for dynamic flows," in *2017 IEEE International Conference on Communications (ICC)*, 2017, Conference Proceedings, pp. 1–7.

[19] N. Torkzaban and J. S. Baras, "Controller placement in SDN-enabled 5G satellite-terrestrial networks," 2021. [Online]. Available: https://arxiv.org/abs/2108.09176

[20] S. Wu, X. Chen, L. Yang, C. Fan, and Y. Zhao, "Dynamic and static controller placement in software-defined satellite networking," *Acta Astronautica*, vol. 152, pp. 49–58, 2018.

[21] A. Papa, T. d. Cola, P. Vizarreta, M. He, C. Mas-Machuca, and W. Kellerer, "Design and evaluation of reconfigurable SDN LEO constellations," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1432–1445, 2020.

[22] S. Xu, X. Wang, B. Gao, M. Zhang, and M. Huang, "Controller placement in software-defined satellite networks," in *2018 14th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, 2018, Conference Proceedings, pp. 146–151.

[23] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. R. Kompella, "Elasticon: An elastic distributed sdn controller," in *10th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, 2014, Conference Proceedings, pp. 17–27.

[24] T. Wang, F. M. Liu, and H. Xu, "An efficient online algorithm for dynamic SDN controller assignment in data center networks," *IEEE-ACM Transactions on Networking*, vol. 25, no. 5, pp. 2788–2801, 2017.

[25] Y. Hu, T. Luo, W. Wang, and C. Deng, "On the load balanced controller placement problem in software defined networks," in *2nd IEEE International Conference on Computer and Communications (ICCC)*, 2016, Conference Proceedings, pp. 2430–2434.

[26] Y. W. Ma, J. L. Chen, Y. H. Tsai, K. H. Cheng, and W. C. Hung, "Load-balancing multiple controllers mechanism for software-defined networking," *Wireless Personal Communications*, vol. 94, no. 4, pp. 3549–3574, 2017.

[27] T. Hu, P. Yi, J. H. Zhang, and J. L. Lan, "Reliable and load balance-aware multi-controller deployment in SDN," *China Communications*, vol. 15, no. 11, pp. 184–198, 2018.

[28] W. Lan, F. Li, X. Liu, and Y. Qiu, "A dynamic load balancing mechanism for distributed controllers in software-defined networking," in *2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, 2018, Conference Proceedings, pp. 259–262.

[29] Y. Xu, M. Cello, I. C. Wang, A. Walid, G. Wilfong, C. H. P. Wen, M. Marchese, and H. J. Chao, "Dynamic switch migration in distributed software-defined networks to achieve controller load balance," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 515–529, 2019.

[30] N. Cai, Y. Han, Y. Ben, W. An, and Z. Xu, "An effective load balanced controller placement approach in software-defined WANs," in *IEEE Military Communications Conference (MILCOM)*, 2019, Conference Proceedings, pp. 361–366.

[31] K. Yang, D. Guo, B. Zhang, and B. Zhao, "Multi-controller placement for load balancing in SDWAN," *IEEE Access*, vol. 7, pp. 167 278–167 289, 2019.

[32] J. F. Wang, L. Li, and M. T. Zhou, "Topological dynamics characterization for LEO satellite networks," *Computer Networks*, vol. 51, no. 1, pp. 43–53, 2007.

[33] M. He, A. Basta, A. Blenk, and W. Kellerer, "How flexible is dynamic SDN control plane?" in *2017 IEEE Conference on Computer Communications Workshops*, 2017, Conference Proceedings, pp. 689–694.

[34] Q. Chen, J. Guo, L. Yang, X. Liu, and X. Chen, "Topology virtualization and dynamics shielding method for LEO satellite networks," *IEEE Communications Letters*, vol. 24, no. 2, pp. 433–437, Feb 2020.

[35] B. Xiong, K. Yang, J. Y. Zhao, W. Li, and K. Q. Li, "Performance evaluation of openflow-based software-defined networks based on queueing model," *Computer Networks*, vol. 102, pp. 172–185, 2016.

[36] J. Guo, D. Rincón, S. Sallent, L. Yang, X. Chen, and X. Chen, "Gateway placement optimization in leo satellite networks based on traffic estimation," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 4, pp. 3860–3876, 2021.

[37] "Gridded population of the world, version 4 (gpwv4): Population count, revision 11," Feb 2020. [Online]. Available: https://doi.org/10.7927/H4JW8BX5

Jianming Guo received the B.S. degree in Astronomy in 2015 from the University of Science and Technology of China (USTC), Hefei, China, and the Ph.D. in Aeronautical and Astronautical Science and Technology in 2021 from the National University of Defense Technology (NUDT), Changsha, China. He is currently an assistant research fellow with the Science and Technology on Complex Aircraft System Simulation Laboratory, Beijing, China. His current research interests include satellite communication networks, SDN, and spacecraft system simulation.

Lei Yang received the Ph.D. from the College of Aerospace Science and Engineering, National University of Defense Technology, Changsha, China in 2008. He is currently a professor with the College of Aerospace Science and Engineering, National University of Defense Technology. He is also a member of the Chinese Society of Astronautics and the China Instrument and Control Society. His current research interests include satellite communication networks, measurement and control technology for micro satellites, on-board computers, spacecraft system modeling and simulation.

David Rincón received the M.Sc. in telecommunication engineering and the Ph.D. in Computer Networks from Universitat Politècnica de Catalunya Barcelona Tech. (UPC), Barcelona, Spain. In 1998, he joined the Department of Telematics Engineering at UPC, where he is currently an Associate Professor. He was a visiting researcher with the Teletraffic Research Centre, University of Adelaide, Australia in 2007 and with the Institute of Pure and Applied Mathematics (IPAM), UCLA, Los Angeles, CA, USA in 2008. His research interests include traffic modeling, network softwarization, optical access networks, and energy consumption in computer networks.

This article has been accepted for publication in IEEE Transactions on Network and Service Management. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2022.3184989

Sebastià Sallent received the M.Sc and Ph.D. in Telecommunication Engineering from the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain. He was the Director of the i2CAT Internet Research Center and the President of the Spanish Telematic Association. He is currently a Full Professor with the Department of Network Engineering, UPC, where he leads the Broadband Networks research group. His current research interests include access networks, network and service virtualization (SDN/NFV), 5G, and new Internet architectures.

Quan Chen received the B.E. and Ph.D. degrees in 2015 and 2021, respectively, from the National University of Defense Technology (NUDT), Changsha, China. He is currently a lecturer with the College of Aerospace Science and Engineering, NUDT. His research interests include mega-constellation satellite networks, UAV networks, and integrated space-terrestrial networks. He has served as a reviewer for several journals including IEEE TWC, TMC, TVT, TAES, COMML, etc. He has served as a TPC member of IEEE ICC workshop on mega-constellation in 2021 and 2022.

Xianfeng Liu received the B.E. degree in Electrical and Information Engineering in 2014 from the College of electrical Engineering, University of Electronic Science and Technology of China, Chengdu, China, and the Ph.D. in Aeronautical and Astronautical Science and Technology in 2021 from the College of Aerospace Science and Engineering, National University of Defense Technology, Changsha, China. He is currently an assistant research fellow with the Beijing Institute of Tracking and Telecommunications Technology, Beijing, China. His current research interests include tracking and data relay technology, TT&C scheduling, and satellite network management.