Temporal Netgrid Model-Based Dynamic Routing in Large-Scale Small Satellite Networks

Jian Li, Hancheng Lu, *Member, IEEE*, Kaiping Xue, *Senior Member, IEEE*, and Yongdong Zhang, *Senior Member, IEEE*

***Abstract*—In recent years, Small Satellite Networks (SSNs) are attracting increasing attention due to its economical prospects and advantages in high bandwidth and low latency. More and more companies and organizations are planning to construct large-scale SSNs to provide global services. In this way, the traffic pattern is more complicated because of random transmission requirements and stochastic packet generations/arrivals, and routing faces more challenges due to network scale and limited resource in small satellites. Traditional satellite routing algorithms, which attempt to exploit the predictable satellite's trajectory with time-discrete graph models, cannot handle these challenges. In this paper, we propose a novel Temporal Netgrid Model (TNM) to portray the time-varying topology of large-scale SSNs. In TNM, the whole space is divided into small cubes (i.e., netgrids) and then, satellites can be located by netgrids instead of coordinates. By doing so, we can construct a network topology for random traffic routing. Furthermore, an efficient Netgrid-based Shortest Path Routing (NSR) algorithm is proposed based on TNM; NSR attempts to find the optimal path from source netgrid to any other reachable netgrids. In this way, the routing complexity is significantly reduced. We also develop a Large-scale Satellite Network Simulator (LSNS) to validate our study. The results show that NSR achieves a significant reduction in computational complexity as well as near-optimal routing performance in terms of end-to-end delay and packet drop rate in scenarios of large-scale SSNs compared with existing satellite routing algorithms.**

***Index Terms*—Large-scale small satellite networks, temporal netgrid model, dynamic routing, computation complexity.**

I. Introduction

In RECENT years, the research on Low Earth Orbit (LEO) Small Satellite Networks (SSNs) is attracting increasing attention due to rapid growth of ubiquitous and seamless communication requirements [2]–[4]. Compared to Medium Earth Orbit (MEO) and Geostationary Earth Orbit (GEO) satellites, LEO small satellites not only have convenience in launching and maintenance, but also have ability to achieve higher throughput and shorter propagation latency [5]–[7]. Thanks to these outstanding features, the construction of SSNs are more preferred to provide broadband service for public users. Many companies such as OneWeb/Airbus, SpaceX/Google and Samsung, are planning to develop large-scale SSNs in order to establish a massive global broadband constellation that will provide worldwide Internet access. According to their plan, the OneWeb system requires about 648 small satellites to be built, the SpaceX system and Samsung system require even more than 1000 satellites and they are foreseen to be reached by 2020 [8]–[11].

Due to the dynamic environment including satellite motion and time-varying Inter-Satellite Links (ISLs), satellite routing becomes more complicated than terrestrial Internet routing. Lots of efforts on routing design have been done to achieve efficient data delivery in satellite networks. In scheduled routing schemes, by making full use of the periodic satellite's trajectory, the topology can be predicted in advance to mitigate the on-board state collection overhead. Since scheduled routing schemes usually adopt pre-calculated topology description or pre-designed contacts such as temporal graph [12] and contact plan [13], they can hardly reflect real-time situations in satellite networks. To overcome this shortcoming, opportunistic routing schemes have been proposed. This kind of routing scheme utilizes dynamic link detection and collects network state information on-board [14]. However, the overhead of the network state collection is also related to the scale of the satellite network. In order to mitigate the overhead of collection and routing process, some studies [15], [16] adopt multi-layered satellite structure and divide satellites into small-scale groups for large-scale satellite network. The basic idea is to take higher layer satellites like MEO or GEO satellites as managers who is in charge of network state collection and routing table calculation for LEO satellites. Because of the expensive cost and longer propagation delay of higher layer satellites, the multi-layered satellite structure is rarely considered in practical application.

Considering the highly dynamic topology variations and stochastic packet arrivals, routing in large-scale SSNs becomes more challenging than that in traditional satellite networks. Therefore, aforementioned routing schemes can not be applied in large-scale SSNs directly. First of all, due to the random traffic transmission pattern, no matter whether it is pre-calculated topology description or pre-designed contacts in scheduled routing schemes, there exists the mismatching between scheduled topology information and stochastic packet arrivals. The accuracy of topology description can be improved by decreasing the

Manuscript received October 23, 2018; revised January 31, 2019; accepted March 31, 2019. Date of publication April 11, 2019; date of current version June 18, 2019. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB0800301, in part by the National Science Foundation of China under Grants 91538203 and 61771445, and in part by the Fundamental Research Funds for the Central Universities. The review of this paper was coordinated by Dr. I. Haque. This paper was presented in part in the Proceedings of the IEEE International Conference on Communications 2017 [1]. *(Corresponding author: Hancheng Lu.)*

The authors are with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China (e-mail: lijian9@mail.ustc.edu.cn; hclu@ustc.edu.cn; kpxue@ustc.edu.cn; zhyd73@ustc.edu.cn).

Digital Object Identifier 10.1109/TVT.2019.2910570

length of discrete time slot in description models like temporal graph. But doing this also leads to a significant increase in storage, which is a bad bargain for resource-limited small satellites. Secondly, due to high-speed motion of LEO satellites whose orbital period is often less than 2 hours, there are frequent and fast topology variations especially when the network consists of hundreds or even thousands of small satellites. This highly dynamic characteristic makes the overhead of state collection in opportunistic schemes very enormous. As a consequence, the massive overhead will reduce the routing performance significantly.

Based on the discussion above, a new efficient routing scheme for large-scale SSNs is urgently in need. To design such routing scheme, the first and the most important step is to develop a new topology description model for large-scale SSNs so that the benefits from the predictable and periodic satellite's trajectory can be exploited. Considering application scenarios of large-scale SSNs with random traffic such as Internet access, we prefer time-continuous description model instead of time-discrete description model. Then, an efficient and low-complexity routing algorithm is proposed for resource-limited small satellites to calculate the optimal data delivery path. The main contributions of this paper are summarized as follows.

1) We propose a time-continuous Temporal Netgrid Model (TNM) to describe the time-varying topology of large-scale SSNs. In TNM, we use static cubes (i.e., netgrids) to replace coordinates of satellites. In this way, TNM can record time-continuous topology information and only a fixed number of netgrids need to be stored at each satellite. On the one hand, TNM has the ability to satisfy random traffic transmission requirements thanks to its time-continuous characteristic. On the other hand, TNM can realize the benefits of scalability and high-efficiency since the topology variations are only related to a fixed number of netgrids.
2) In order to provide different accuracy of topology description in TNM, we propose a multi-layer netgrid partition method and theoretically analyze the relationship between the number of partition layers and the description accuracy.
3) Based on TNM, we propose an efficient and low-complexity routing algorithm for resource-limited small satellites. Since the complexity of the proposed routing algorithm only related to the number of netgrids other than the number of satellites, its computation complexity can be significantly reduced compared with traditional satellite routing algorithms.
4) We evaluate the proposed routing algorithm and compare it with several existing algorithms. Simulation results show that the proposed routing scheme can achieve a significant reduction in computation complexity as well as near-optimal routing performance.

The rest of this paper is organized as follows. In Section II, the related works about traditional routing schemes in satellite networks are discussed. In Section III, we firstly introduce our motivation for proposing TNM, then the definition and data structure of TNM are described. After that, a multi-layer netgrid partition method applied in large-scale SSNs is proposed in Section IV and a netgrid-based routing algorithm is also proposed in Section V. To evaluate the performance of our routing scheme, simulation results are shown in Section VI. At last, conclusions are drawn in Section VII.

Notations: In this paper, the word about "neighbor node" means that the node can be reached directly without any intermediate node.

II. Related Work

In satellite networks, routing problem has attracted great attention and related research has been studied for many years [2], [8]. Two kinds of routing schemes, i.e., scheduled routing schemes and opportunistic routing schemes, are concerned in existing studies. The former basically utilizes scheduled topology description information such as temporal graph or contact graph. The latter adopts dynamic state collection process to acquire the topology information periodically.

Scheduled routing schemes attempt to make full use of periodic satellite trajectories and predictable topology variations. Markus *et al.* [12] proposed a dynamic virtual topology routing, which adopted off-board (i.e., offline) temporal graph calculation and on-board (i.e., online) shortest path search algorithm to find the optimal routing path. Since traditional literatures mainly divided the system period into equal-length snapshots [17], Huang *et al.* [18] focused on the storage cost saving of such time-discrete snapshots method and proposed an optimized division method of the system period, which made a tradeoff between the storage cost and the system performance. The modified temporal graph proposed by Alaoui *et al.* [19] also tried to reflect actual topology variations rather than equal-length partitions of time slots. Another representative scheduled routing scheme is Contact Graph Routing (CGR), which is proposed for delay tolerant networks as a promising routing scheme and has been applied in satellite communications successfully [20]–[22]. The basic idea of CGR is to utilize a scheduled contact graph to achieve the optimal routing. The performance of CGR is highly related to the pre-scheduled contacts, and some studies on optimization of contact plan design have been done for different situation such as mission planning [13] and fairness guarantee [23]. It should be noted that the contact plan in CGR can hardly be optimized without perceiving transmission requirements in advance.

In opportunistic routing schemes, routing process based on on-board state collection is considered. In this way, routing can better respond to dynamic topology variations and unpredictable link interruptions. Song *et al.* proposed [14] a Traffic-Light-based Routing (TLR) for non-GEO satellite IP networks. In TLR, each orbit had a speaker to collect un-deterministic state information such as queuing delay and so on. The main contribution in TLR was that it would adjust the route dynamically according to the real-time state at intermediate nodes to avoid congestion. It has been proved that this kind of single layered opportunistic routing schemes can achieve good performance in load balancing and routing performance via state collection [24], [25], but near-real-time collection process also means lots of

TABLE I
NOTATION

Symbol	Definition/Explanation
G	Topology graph
V	Satellite nodes set
E	Edges set
N	Total number of satellite nodes
N^*	The number of all non-empty netgrids in TNM
$\mathcal{T}_v$	Orbit period of satellite node v
x, y, z	Cartesian coordinates that indicates the location of the netgrid
$\mathcal{C}_i$	i-th netgrid in the netgrid table, coordinate x, y, z and index i share a one-to-one mapping
$\mathcal{P}_{v,i}$	Triple which includes the trajectory information of satellite v in netgrid $\mathcal{C}_i$
$t^e_{v,i}$	The time that satellite v enters the netgrid $\mathcal{C}_i$
$t^l_{v,i}$	The time that satellite v leaves the netgrid $\mathcal{C}_i$
V_i	All satellites that will pass through $\mathcal{C}_i$ during time interval $[t^e_{v,i}, t^l_{v,i})$.
R_a	Communication radius of all small satellites in the same layer
L_l	The side length of a netgrid in partition layer l
R^b	Transmission rate collected by beacon packet
R	Estimated transmission rate between two satellites
T_p	Estimated propagation time between two netgrids
$\|D\|$	Size of data packet D
$\mathcal{R}$	Router Table

overhead and high complexity. In order to make full use of multi-layered satellite resources and mitigate the overhead, Akyildiz *et al.* [15] proposed a Multi-layered Satellite Network (MLSN) architecture. They introduced the concept of satellite groups to reduce the complexity involving topology control and route decision. After that, Chen *et al.* [16] adopted a two-layered MLSN and proposed Satellite Grouping and Routing Protocol (SGRP). In SGRP, LEO satellites were divided into different groups managed by MEO satellites. Since group manager took charge in routing table calculation and distribution, each manager would collect state information from LEO satellites periodically. There also exist some multi-layered routing schemes concerning on the traffic congestion and QoS problem, etc [2], [26], [27].

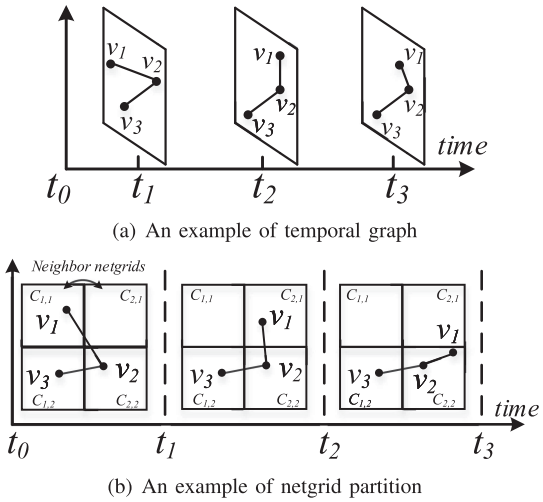
Most of the existing satellite schemes use the time-discrete graph model to describe the network topology, with the goal to exploit benefits from the predictable and periodic satellite's trajectory. However, they suffer from poor performance when on-demand routing for randomly arriving traffic is considered. Although on-board state collection makes routing schemes adapt to dynamic satellite network conditions, it also significantly increases routing signal overhead and computation complexity when applied to large-scale satellite networks, especially that consisting of resource-limited small satellites.

III. TEMPORAL NETGRID MODEL

In this section, a novel topology description model, TNM is proposed. By using cubes (i.e., netgrids) to locate satellites instead of coordinates, TNM can provide time-continuous topology information for on-demand routing under random traffic transmission requirements.

A. Motivation

In order to improve satellite routing performance, the predictable and periodic satellite's trajectory is considered in TNM.



(a) An example of temporal graph

(b) An example of netgrid partition

Fig. 1. Comparison between TGM and TNM.

There are many studies on the topology description model using periodic characteristic in satellite networks. For instance, a widely used topology description model, Temporal Graph based Model (TGM) has been successfully applied in traditional satellite routing [12], [17], [28]. The basic idea of TGM is to divide the system period into discrete time denoted by $\{t_1, t_2, \ldots, t_M\}$. In this way, the whole topology variations in the system period can be represented as a set of static topology graph, i.e., $\{G_1, G_2, ...G_M\}$ and sub-graph at t_m is denoted by $G_m = \{V_m, E_m\}$. An example is shown in Fig. 1(a), where topology evolving is composed of three static topology graphs. However, TGM can only precisely describe the topology at a specific time. If there is a request at $t \notin \{t_1, t_2, \ldots, t_M\}$, only static graph G_n at nearest time point $t_n \in \{t_1, t_2, \ldots, t_M\}$ can be provided. Thus, time-discrete models such as TGM result in inaccurate network topology when time-continuous random traffic transmission is considered. Motivated by this fact, a novel TNM is proposed to better utilize the periodic trajectory and provide time-continuous topology information for routing calculation.

B. Definition

In TNM, the first step is to divide the whole space where satellite's orbit exists into a number of cubes. After the space partition, the location of each satellite in the network can be indicated by the discrete cube (i.e., netgrid). Meanwhile, the specific time slot that a satellite stays in each netgrid should also be recorded. By doing so, the satellite's trajectory can be described as a continuous time ordered set of discrete netgrids. A two-dimensional example is shown in Fig. 1(b) where the space is divided into 4 static netgrids denoted by $\mathcal{C}_{x,y}$. Along with the time evolving, the relationship between satellite nodes and netgrids are varying. In Fig. 1(b), t_1 and t_2 represent the moment that v_1 enters $\mathcal{C}_{2,1}$ (i.e., leaves $\mathcal{C}_{1,1}$) and enters $\mathcal{C}_{2,2}$ (i.e., leaves $\mathcal{C}_{2,1}$), respectively. Thus, for any given time point $t \in (t_0, t_3)$, v_1's location could be obtained. For example, if we request v_1's location at time $t' \in (t_1, t_2)$, then $\mathcal{C}_{2,1}$ could be obtained. Note

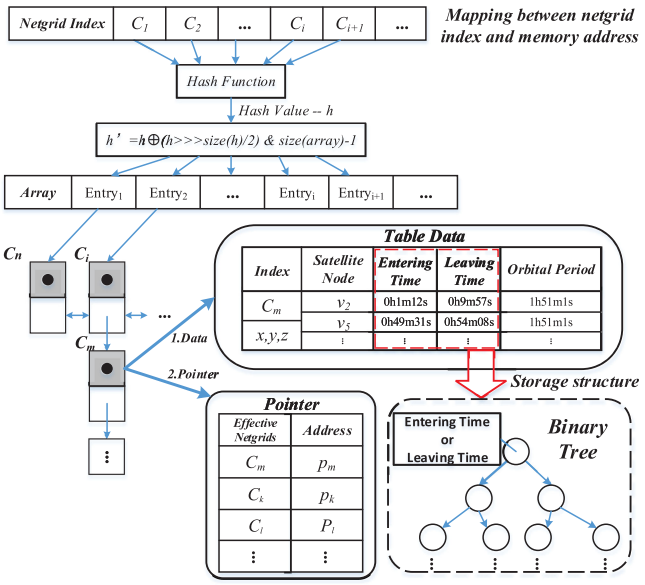


Fig. 2. Specific data structure of the netgrid table stored in the satellite.

that Cartesian coordinates x, y, z of the netgrid center is adopted to identify each netgrid in three-dimensional space.

Next, we construct a netgrid table to gather all netgrids and give a mathematic description of each netgrid. For ease of notation, the symbol of netgrid $\mathcal{C}_{x,y,z}$ is replaced by $\mathcal{C}_i$, and the coordinate of the netgrid center x, y, z and index i share a one-to-one mapping to guarantee unique index. Let K denote the total number of static cubes (i.e., netgrids), and V_i denote all satellite nodes that would pass through netgrid $\mathcal{C}_i$ according to their trajectory. We define a triple which includes necessary information for the netgrid table at first:

$$\mathcal{P}_{v,i} = \{v, \mathcal{T}_v, (t^e_{v,i}, t^l_{v,i})\}, v \in V, t^e_{v,i}, t^l_{v,i} \in [0, \mathcal{T}_v]. \quad (1)$$

Then each netgrid $\mathcal{C}_i$ in the netgrid table can be described as

$$\mathcal{C}_i = \{\mathcal{P}_{v,i} | v \in V_i\}, i \in [1, K], \quad (2)$$

where $\mathcal{T}_v$ represents the orbital period of satellite node v, and $v \in V_i$ indicates that v's trajectory overlaps with the space of netgrid $\mathcal{C}_i$, i.e., satellite v will pass through $\mathcal{C}_i$ during time interval $[t^e_{v,i}, t^l_{v,i})$. Symbols $t^e_{v,i}$ and $t^l_{v,i}$ denote the moment that satellite v enters and leaves netgrid $\mathcal{C}_i$, respectively. Due to the periodic trajectory of satellites, each satellite will enter and leave the same netgrid periodically. When time t expires orbital period $\mathcal{T}_v$, t can be transformed into $[0, \mathcal{T}_v]$ easily by using remainder operation (i.e., $t \ mod \ \mathcal{T}_v$).

C. Data Structure

Considering resource-limited satellite environments, we design an efficient data structure for the netgrid table. Our data structure is introduced from two aspects. One is how we map each netgrid to each array entry, and the other is what data would be stored inside each netgrid element.

In order to accelerate searching process in the netgrid table, we adopt a hashmap method to allocate the memory address. Fig. 2 shows the designed data structure for the netgrid table, where $\mathcal{C}_m = \{\mathcal{P}_{v_{2,i}}, \mathcal{P}_{v_{5,i}}, ...\}$ and the memory address in $Entry_1$ and $Entry_2$ indicates $\mathcal{C}_n$ and $\mathcal{C}_i$, respectively. It needs to be emphasized that the Cartesian coordinates in netgrid $\mathcal{C}_{x,y,z}$ has been replaced by index i in $\mathcal{C}_i$ for notation convenience.

Our method generates an array to store the memory address of each netgrid and utilizes a hash operation to achieve $\mathcal{O}(1)$ searching complexity. At first, netgrid index will be input into a hash function. In order to limit the output hash value h in the range of array address, h will execute AND operation with the size of array as shown in Fig. 2. Note that symbol $\oplus$ represents XOR operation. In most cases, it would achieve one-to-one mapping between each netgrid index and array entry. However, there might exist hash value collision situation, where netgrids with the same h' would be mapped into the same array entry. In this situation, netgrids $\mathcal{C}_i$ and $\mathcal{C}_m$ with the same h' are stored as a linked list as shown in Fig. 2. Then the complexity of searching $\mathcal{C}_i$ and $\mathcal{C}_m$ becomes $\mathcal{O}(logN_{e_2})$, where N_{e_2} denotes all netgrids stored in $Entry_2$.

Before introducing the specific data in each netgrid, we give a definition about *effective netgrids* at first.

Definition 1: If satellite $v \in V_i$ at a given time t, neighbor netgrids of $\mathcal{C}_i$, which are totally located within the communication range of v, are defined as *effective netgrids* of v.

According to the definition, all satellites contained in *effective netgrids* of v means that satellite v has direct ISLs with these satellites. An intuitional example will be shown in Section IV.

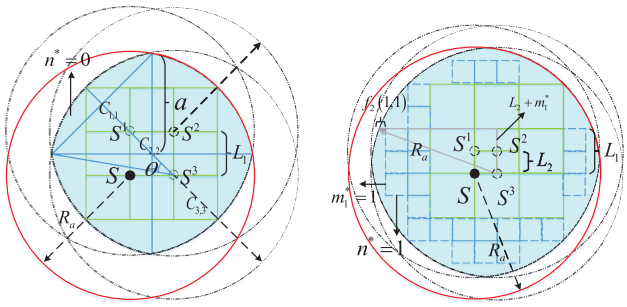
For each netgrid element in the array, it contains $Data$ (i.e., table data) and $Pointer$ (i.e., *effective netgrids* pointer) as shown in Fig. 2. In $Data$, to improve searching efficiency, a binary tree storage structure is adopted. Then the searching complexity for a particular time is $\mathcal{O}(log\overline{N})$, where $\overline{N}$ denotes the average number of satellites contained in one netgrid. In $Pointer$, according to the definition of *effective netgrids*, we can find all neighbor satellite nodes in *effective netgrids* by looking up the netgrid table for routing decision. If we assume F as the average number of *effective netgrids*, then the complexity of neighbor nodes searching operation will be $\mathcal{O}(Flog(\overline{N}))$.

IV. Multi-Layer Netgrid Partition Method in Large-Scale SSNs

Since the satellite's trajectory can be described with discrete netgrids rather than the coordinate of satellite nodes, the topology description accuracy is determined by netgrid partition. In this section, we propose a multi-layer netgrid partition method and theoretically analyze the relationship between the accuracy of topology description and the number of partition layers.

A. Netgrid Partition

The specific multi-layer partition approach is introduced at first. Then an example is given to explain why we partition the netgrid in this way. It should be emphasized that the multi-layer partition approach is to partition the netgrid with different size to acquire different topology description accuracy, while the multi-layered satellite structure in previous works represents that the network is composed of different types of satellites. Thus,



(a) Multi-layer netgrid partition method in layer 1 (b) Multi-layer netgrid partition method in layer 2

Fig. 3. A simplified two-dimensional example about netgrid partition method in different layer.

the multi-layer partition approach and the multi-layered satellite structure are not the same at all.

Let L_l represent the side length of a cube (i.e., netgrid) in layer l, and R_a represent the communication radius of each satellite node. Then the side length of each netgrid in different partition layer l should be

$$L_l = 2L_{l+1} = \frac{\sqrt{3}}{3 \cdot 2^l} R_a, l \in \mathbb{Z}, \tag{3}$$

where $L_1 = \frac{\sqrt{3}}{6} R_a$ in partition layer 1. Here, dichotomy method based on partition layer 1 is adopted to achieve multi-layer netgrid partition. In order to explain the reason why we choose $\frac{\sqrt{3}}{6} R_a$ as partition size in layer 1, a specific netgrid partition example in layer 1 is shown in Fig. 3(a).

For expression convenience, the example in Fig. 3 is shown as a two-dimensional form which can be seen as cross-section drawing of three-dimensional netgrid, and two-dimensional Cartesian coordinates $\mathcal{C}_{x,y}$ is adopted as the index of netgrids. Squares in Fig. 3 represent cubes (i.e. netgrids) in TNM.

As we can see in Fig. 3(a), the cube in the center of netgrids is represented by $\mathcal{C}_{2,2}$ and a satellite node S is located at one vertex of cube $\mathcal{C}_{2,2}$. Since the exact location of S inside the cube is unknown according to the static netgrid, we take all vertices of $\mathcal{C}_{2,2}$ which are S, S^1, S^2 and S^3 into consideration, and separately draw circle (i.e., communication coverage) in the centre of them. Then we can get an overlapping area which is shown as colorful area in Fig. 3(a). This area represents the effective communication coverage of S due to the fact that the exact location of S inside $\mathcal{C}_{2,2}$ is unknown. In this way, no matter where exact S is inside $\mathcal{C}_{2,2}$, we can assure that all satellite nodes located in neighbor cubes, which are inside the overlapping area, are in the communication coverage of S. To achieve this effect, all neighbor cubes of satellite S must be located inside the overlapping area as shown in Fig. 3(a), then the side length of three-dimensional netgrid should be less than or equal to $\frac{\sqrt{3}}{6} R_a$ in three-dimensional space. That is why we choose $L_1 = \frac{\sqrt{3}}{6} R_a$ as partition layer 1.

Based on the example in Fig. 3, we define *effective coverage* of satellite S inside netgrid $\mathcal{C}_{2,2}$ as the overlapping area of the communication coverage in each extreme location of the netgrid. According to our definition in Section III, *effective netgrids* are neighbor netgrids contained by *effective coverage* of S. In Fig. 3(a), the *effective coverage* of S is the colorful area, and *effective netgrids* of S are 9 cubes totally contained by the *effective coverage*. Since all nodes contained by a netgrid has the same overlapping area, if both satellite v_1 and satellite v_2 locate inside netgrid $\mathcal{C}_{x,y}$, they will have the same *effective netgrids* and *effective coverage*.

B. Accuracy Analysis

According to the above description, our method can not find satellite nodes outside the overlapping area, which brings the loss of the description accuracy. To measure this sacrifice, a definition about the netgrid *description accuracy* is given in the follows. Considering the three-dimensional space in reality, the volume of *effective netgrids* and the volume of the *effective coverage* are assumed to be V_l^{cubes} and V_l^{ol}, respectively.

Definition 2: Netgrid *description accuracy* η_l in layer l is defined as the ratio between the volume of *effective netgrids* and the volume of communication coverage, i.e., $\eta_l = \frac{V_l^{cubes}}{\frac{4}{3}\pi R_a^3}$.

Next, we discuss about the *description accuracy* in different partition layer. It is worth emphasizing that three-dimensional netgrid partition is considered in our discussion. Compared to the actual communication coverage of S, which is drawn by red circle, the partition method in Fig. 3(a) sacrifices the *description accuracy* too much as *effective netgrids* of S is much smaller than the actual communication coverage of S. If we reduce the size of each netgrid in Fig. 3(a), the size of *effective netgrids* will increase and become closer to the actual communication coverage of S. Fig. 3(b) shows the netgrid partition method in layer 2, where $L_2 = \frac{L_1}{2}$, and more *effective netgrids* located inside the *effective coverage*. It is worth noting that the topology information in TNM is obtained by looking up the netgrid table. Along with the increase of *effective netgrids* in higher partition layer, the cost of lookup table method is also rising. Considering resource-limited satellite environments, a tradeoff should be considered between the *description accuracy* and the cost of looking up the netgrid table. Our simulation results in Section VI show that netgrid partition in layer 2 can achieve near-optimal routing performance.

Given the side length L_l of each netgrid in Equation (3), $\lim_{l\to\infty} L_l \to 0$ should be satisfied. Thus, we have the following proposition.

Proposition 1: In multi-layer netgrid partition method, when $l \to \infty$, the volume of each netgrid in layer l approaches zero, and both the volume of *effective netgrids* V_l^{cubes} and the volume of the *effective coverage* V_l^{ol} approach the actual communication coverage.

Proof: As shown in Fig. 3(a), the angle of θ can be calculated by

$$\theta = \arccos\left[3 \cdot 2^{l-1} + \frac{1}{2^{l+2}} - \frac{1 + \sqrt{1 + \frac{2^{l+1}-1}{2^l} \cdot R_a}}{2^{1-l} \cdot R_a} - \frac{1}{4}\right],$$

and the side length of a can be calculated by

$$a = \frac{\sqrt{1 + \frac{2^{l+1}-1}{2^l} R_a^2} - 1}{2}.$$

Depending on the knowledge of solid geometry, we can calculate the volume of *effective coverage* in layer l by

$$V_l^{ol} = \frac{\theta}{180} 2\pi \cdot \frac{4}{3}\pi R_a^3 + \frac{1}{3}\pi R_b^2 \left(\sqrt{R_a^2 - R_b^2} - \sqrt{a^2 - R_b^2} \right),$$

where $R_b = R_a \cdot \sin\theta$. The volume of *effective netgrids* can be calculated by

$$\begin{aligned} V_l^{cubes} &= \sum_{n=1}^{n=n^*} V_l^n + (3L_1 + L_2 + \cdots + L_l)^3 \\ &= \sum_{n=1}^{n=n^*} V_l^n + \left(\frac{2^{l+1}-1}{2^l} \cdot R_a \right)^3, l \in \mathbf{Z}, \end{aligned}$$

and the volume of cubes V_l^n in level n is

$$V_l^n = [6 \cdot (2m_n + 1)^2] \cdot (L_l)^3.$$

For a given l, there exists a equation

$$f_l(n, m_n) = \sqrt{R_a^2 - \sqrt{(m_n + 1) \cdot L_l^2}} - \left(2^l + n - \frac{1}{2} \right) \cdot L_l,$$

and it must satisfy $f_l(n, m_n) > 0$. Then we assume the maximum level n^* and maximum number of cubes m_n^* in level n, they can be calculated by

$$m_n^* = argmin[f_l(n, m_n)], f_l(n, m_n) > 0, \quad (4)$$

$$n^* = argmin[f_l(n, 0)], f_l(n, m_n) > 0, \quad (5)$$

where $n, m_n \in \mathbb{N}$. An example about the maximum level n^* and m_n^* in layer 2 is shown in Fig. 3(b). When $l \to \infty$, then $n^* \to \infty$. In this case, we have

$$\lim_{l\to\infty} V_l^{ol} = \lim_{l\to\infty, n^*\to\infty} V_l^{cubes} = \frac{4}{3}\pi \cdot R_a^3,$$

which implies $\lim_{l\to\infty} \eta_l = 1$. ■

According to the proof of Proposition 1, total number of *effective netgrids* in layer l can be calculated by $\frac{V_l^{cubes}}{(L_l)^3}$, and the boundary of *effective netgrids* in each direction can also be calculated by Equation (4) and (5).

V. Netgrid-Based Shortest Path Routing Algorithm

In this section, we focus on a status-aware netgrid-based routing algorithm. A beacon protocol is proposed at first. By collecting status of neighbor satellites periodically, satellites can adjust routing decisions dynamically. Then, in order to achieve efficient data delivery in large-scale SSNs, a low-complexity routing algorithm is proposed. Due to the spatial characteristic of netgrids, each hop in the routing path actually contains multi-choice. Our further analysis confirms the optimality and complexity reduction of the proposed algorithm.

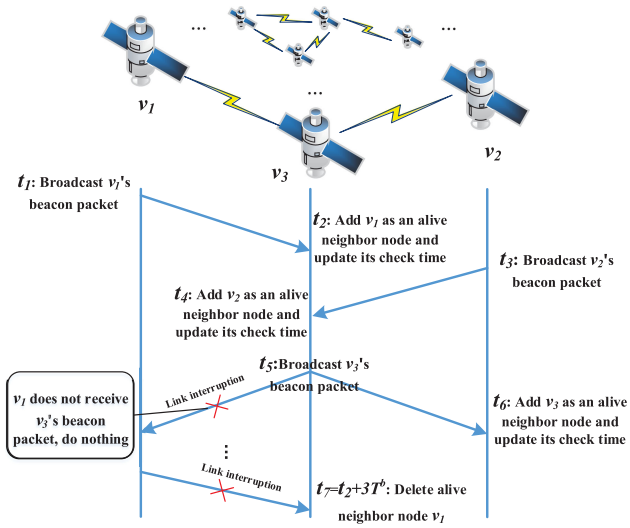


Fig. 4. Workflow of the beacon protocol.

A. Beacon Protocol

The beacon protocol is proposed by taking into account two factors. One is to adapt the dynamic environments in satellite networks, such as unpredictable link interruption and node failure. The other is to diminish the influence of the accuracy sacrifice in TNM. In short, the basic idea of the beacon protocol is to collect status (e.g., queuing delay, transmission rate and so on) of neighbor satellites periodically for routing decision.

In the beacon protocol, each satellite node will broadcast beacon packet periodically. When the other satellite nodes receive the beacon packet for the first time, they will mark the sender as an alive neighbor and save the corresponding information contained in beacon packet (e.g., broadcast period T^b, transmission rate R^b and so on). By recording the receiving time as the latest check time, status of all alive neighbors can be updated periodically. In order to detect link interruption and node failure, alive neighbor v will be deleted after three broadcast periods (i.e, $3T_v^b$) unless a new beacon packet broadcasted by v is received. For notation convenience, all alive neighbor nodes of satellite s will be described as alive neighbor set $\mathcal{N}_s$.

The specific workflow of the beacon protocol is shown in Fig. 4. In the figure, v_1, v_2 and v_3 are neighbors with each other. Since v_3 can receive beacon packet from v_1 and v_2, v_1 and v_2 will be marked as an alive neighbor of v_3 at t_2 and t_4, respectively. However, due to the fact that link interruption occurs between v_1 and v_3 during $[t_5, t_7]$, v_3 will not be added as an alive neighbor of v_1, and v_1 will be deleted from alive neighbor list of v_3 after time t_7.

B. Routing Strategy

Based on the topology information from the netgrid table and collected status from the beacon protocol, a Netgrid-based Shortest-path Routing (NSR) algorithm is proposed to solve the optimal path-finding problem. In the follows, we introduce the details of NSR algorithm.

1) Neighbor Nodes Check: Algorithm 2 is in charge of neighbor nodes check in initialization. At the beginning, NSR checks that if the destination node d is an alive neighbor node of source node s (i.e., $d \in \mathcal{N}_s$) in initialization phase. If $d \in \mathcal{N}_s$, NSR algorithm will be ended and data packet D can be transmitted to the destination directly; If $d \notin \mathcal{N}_s$, the corresponding netgrids of all alive neighbor nodes in $\mathcal{N}_s$ will be added into source set S^*. And the transmission rate R^b from s to alive neighbor node, collected by the beacon protocol, will also be used to calculate the transmission delay. Note that "$A \setminus B$" in Algorithm 2 represents difference set whose elements contained in set A but not contained in set B.

2) Netgrid-Based Shortest Path Routing Algorithm: Algorithm 1 describes the main procedure of NSR algorithm, and Algorithm 3 is in charge of maintaining priority queue Q.

After checking alive neighbor nodes in $\mathcal{N}_s$, NSR will start to search the optimal routing path. Let time t_s denote route start time, which is decided by the packet arrival time. NSR focuses on non-empty netgrids where satellite nodes locate and NSR only considers the network topology at time t_s when the algorithm executes. Since the beacon protocol can only collect neighbor nodes information, the transmission rate R in line 8 of Algorithm 3 is estimated. That is to say, the optimal routing path output by NSR only outputs the estimated shortest path, thus each intermediate node could adjust routing path according to the real-time status of neighbor satellites.

Here, we use greedy approach to find the shortest path from source netgrid $\mathcal{C}_s$ (source node s is contained by $\mathcal{C}_s$) to destination netgrid $\mathcal{C}_d$ (destination node d is contained by $\mathcal{C}_d$). Note that the shortest path found by NSR is composed by netgrids. The following demonstration process shows how our routing algorithm can find the shortest path in a static topology.

Proposition 2: NSR routing algorithm can find the shortest path $p*$ from a given source netgrid to a given destination netgrid.

Proof: Before starting to prove, we assume $\delta(\mathcal{C}_u, \mathcal{C}_v)$ represents the shortest path between netgrid $\mathcal{C}_u$ and netgrid $\mathcal{C}_v$, and $p(\mathcal{C}_v)$ represents the estimated shortest path from source netgrid $\mathcal{C}_s$ to netgrid $\mathcal{C}_v$ obtained by NSR algorithm. Let m denote the number of neighbor netgrids in $\mathcal{N}_{\mathcal{C}_s}$. Then, the mathematical induction is used in the proof:

1) At the beginning, source set S^* contains source netgrid $\mathcal{C}_s$ and neighbor netgrids $\mathcal{N}_{\mathcal{C}_s}$. In Algorithm 2, all neighbor satellite nodes collected from beacon packets are checked and line 9 ensures the minimum transmission time to all corresponding neighbor netgrids of these neighbor nodes. Thus, all netgrids in S^* have found the shortest path started from $\mathcal{C}_s$.

2) In searching process of Algorithm 1, all available neighbor netgrids can be obtained from the netgrid table in line 12, which contains all reachable neighbor netgrids from netgrid $\mathcal{C}' \in S^*$ to next-hop at time t_s. Assuming node $\mathcal{C}^*$ is one neighbor netgrid of $\mathcal{C}' \in S^*$ and its time cost is minimum in priority queue Q. Then, $\mathcal{C}^*$ must be extracted from Q and added into source set S^* at the end of first iteration in line 18 to line 20, and path $p(\mathcal{C}^*) = \delta(\mathcal{C}_s, \mathcal{C}^*)$ is the shortest path from $\mathcal{C}_s$ to $\mathcal{C}^*$.

3) After $k-1$ iterations in line 8, source set already has $k+m$ netgrids in S^* with $k+m < N^*$. We assume that the algorithm finds k-th netgrid $\mathcal{C}_k$ and path $p(\mathcal{C}_k)$ has minimum time cost in k-th iteration. As all netgrids in S^* have found the shortest path started from $\mathcal{C}_s$, i.e., $p(\mathcal{C}') = \delta(\mathcal{C}_s, \mathcal{C}')$ for $\mathcal{C}' \in S^*$. If $p(\mathcal{C}_k) \neq \delta(\mathcal{C}_s, \mathcal{C}_k)$, there must exist at least one netgrid that is not included by S^* in the shortest path $\delta(\mathcal{C}_s, \mathcal{C}_k)$. Let $\mathcal{C}_f$ represent the first netgrid that is not included by S^* in the shortest path $\delta(\mathcal{C}_s, \mathcal{C}_k)$, then the time cost of $p_s(\mathcal{C}_f)$ and the time cost of $p_s(\mathcal{C}_k)$ must have relationship:

$$time[p(\mathcal{C}_f)] \leq time[p(\mathcal{C}_k)] < time[\delta(\mathcal{C}_s, \mathcal{C}_k)].$$

Thus we obtain a contradiction from the assumption that node $\mathcal{C}_k$ has minimum time cost among all nodes from source node $\mathcal{C}_s$. In this case, the path $p(\mathcal{C}_k)$ that NSR found must be the shortest path from $\mathcal{C}_s$ to $\mathcal{C}_k$, i.e., $p(\mathcal{C}_k) = \delta(\mathcal{C}_s, \mathcal{C}_k)$. ■

It should be emphasized that the shortest path $p*$ found by NSR from $\mathcal{C}_s$ to $\mathcal{C}_d$, is composed by netgrids. Therefore, if $\mathcal{C}_m$ is an intermediate netgrid of $p*$, then all satellite nodes contained in $\mathcal{C}_m$ can be selected as an intermediate node during the transmission, which makes each hop in the routing path contain multi-choice.

Algorithm 1: Netgrid-based Shortest-path Routing.

Input: Netgrid table, satellite nodes set V, source node $s \in V$, TTL of data packet, start time t_s, data packet size $|D|$;
Output: Router Table $\mathcal{R}$;

1 **Initialize:**
- Update router table and searched set S_d by executing Algorithm 2, let source set $S^* = \{\mathcal{C}_s, \mathcal{N}_{\mathcal{C}_s}\}$;
- Set termination time $t_t = t_s + TTL$, $time(\mathcal{C}_s) = t_s$, $p(s) = \oslash$, $\mu = true$;
- Set priority queue $Q = \{time(\mathcal{C}), p(\mathcal{C}) | \mathcal{C} \in \mathbb{C}\} = \oslash$;

2 **if** $\mathcal{N}_{\mathcal{C}^s} == \oslash$ **then**
3 | **Return** $\mathcal{R}$;
4 **end**
5 **while** $!(Q == \oslash \ and \ \mu == false)$ **do**
6 | $\mu = false$;
7 | **foreach** $\mathcal{C}' \in S^*$ **do**
8 | | **if** $\mathcal{C}' \in S_d$ **then**
9 | | | **Continue** loop in line 7;
10 | | **end**
11 | | $S_d = S_d \cup \{\mathcal{C}'\}$;
12 | | $Neighbor(\mathcal{C}') \leftarrow$ Find all non-empty neighbor netgrids of $\mathcal{C}'$ within *effective netgrids* at t_s;
13 | | Update Q and μ by executing Algorithm 3;
14 | **end**
15 | Extract netgrid $\mathcal{C}_{min}$ which has minimum $time(\mathcal{C}_{min})$ from Q;
16 | Add $\mathcal{C}_{min}$ into S^*;
17 | Add $\{time(\mathcal{C}_{min}), p(\mathcal{C}_{min})\}$ into $\mathcal{R}$;
18 **end**
19 **Return** $\mathcal{R}$

Algorithm 2: Neighbor Nodes Check.

Input: Source node s, destination node d, data packet size $|D|$
Output: Router table $\mathcal{R}$, searched set S_d, neighbor netgrid set $\mathcal{N}_{\mathcal{C}_s}$;

1 **Initialize:**
- Find corresponding netgrid $\mathcal{N}_{\mathcal{C}_s}$ of all alive neighbor nodes in $\mathcal{N}_s$ at t_s;
- Router Table $\mathcal{R} = \{time(\mathcal{C}), p(\mathcal{C}) | \mathcal{C} \in \mathbb{C}\} = \oslash$;

2 **if** $d \in \mathcal{N}_s$ **then**
3 Find corresponding netgrid $\mathcal{C}'$ of d from the netgrid table;
4 Put $time(\mathcal{C}_i) = \frac{|D|}{R^b(\mathcal{C}_s, \mathcal{C}')}$ and $p(\mathcal{C}') = e(\mathcal{C}_s, \mathcal{C}')$ into $\mathcal{R}$;
5 **Return** $\mathcal{R}$, $\oslash$, $\oslash$
6 **end**
7 **foreach** $v \in \mathcal{N}_s$ **do**
8 Find corresponding netgrid $\mathcal{C}'$ of v from the netgrid table and put $\mathcal{C}'$ into $\mathcal{N}_{\mathcal{C}_s}$;
9 **if** $\mathcal{R}(\mathcal{C}') != \oslash$ *and* $time(\mathcal{C}') > \mathcal{R}(\mathcal{C}')$ **then**
10 **Continue** loop in line 7;
11 **end**
12 Put $time(\mathcal{C}') = \frac{|D|}{R^b(\mathcal{C}_s, \mathcal{C}')}$ and $p(\mathcal{C}') = e(\mathcal{C}_s, \mathcal{C}')$ into $\mathcal{R}$;
13 **end**
14 $Neighbor(\mathcal{C}_s) \leftarrow$ Find all non-empty neighbor netgrids of $\mathcal{C}_s$ within *effective netgrids* at t_s;
15 Set $S_d = Neighbor(\mathcal{C}_s) \setminus \mathcal{N}_s$;
16 **Return** $\mathcal{R}$, S_d, $\mathcal{N}_{\mathcal{C}_s}$

Algorithm 3: Transmission Delay Maintenance.

Input: Source set S^*, start and termination time t_s, t_t, priority Queue Q, data packet size $|D|$, $Neighbor(\mathcal{C}')$, μ;
Output: μ, Q;

1 **foreach** $\mathcal{C}_n \in Neighbor(\mathcal{C}')$ **do**
2 **if** $\mathcal{C}_n \in S^*$ **then**
3 **Continue** loop in line 1;
4 **end**
5 **if** $time(\mathcal{C}') \geq t_t$ **then**
6 **Continue** loop in line 1;
7 **end**
8 $time(e(\mathcal{C}', \mathcal{C}_n)) = \frac{|D|}{R(\mathcal{C}', \mathcal{C}_n)} + T_p(e(\mathcal{C}', \mathcal{C}_n))$;
9 $time(\mathcal{C}_n) = time(\mathcal{C}') + time(e(\mathcal{C}', \mathcal{C}_n))$;
10 **if** $time(\mathcal{C}_n) < Q(\mathcal{C}_n)$ **then**
11 Let $p(\mathcal{C}_n) = p(\mathcal{C}')$, then add $e(\mathcal{C}', \mathcal{C}_n)$ into $p(\mathcal{C}_n)$;
12 Replace $Q(\mathcal{C}_n)$ with $\{time(\mathcal{C}_n), p(\mathcal{C}_n)\}$;
13 **end**
14 $\mu = true$;
15 **end**
16 **Return** μ, Q

C. Complexity Analysis

The searching process in Algorithm 1 shows that the complexity of NSR is only related to the number of non-empty netgrids rather than satellites nodes. In this case, the computation complexity of NSR is quite different from traditional routing algorithm such as Dijkstra's Shortest Path First (SPF) algorithm.

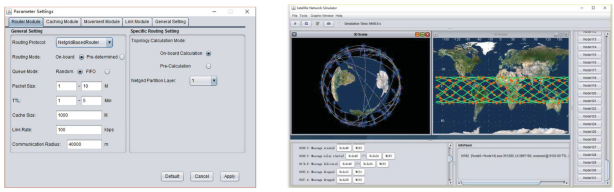
Let $\overline{N}_n$ denote the average number of all non-empty netgrids in TNM. Then the total number of executions about the loop in line 5 of Algorithm 1 equals to $\overline{N}_n$ at most. In order to obtain the minimum value from priority queue Q in NSR, the maintenance cost about priority queue Q for each loop in line 8 of Algorithm 3 is $\mathcal{O}(\overline{N}_n)$, and the total number of calculations about cumulative transmission time $time(\mathcal{C}_n)$ in line 9 of Algorithm 3 is $\overline{E}_n$. Consequently, the computation complexity of NSR is $\mathcal{O}({\overline{N}_n}^2 + \overline{E}_n)$, where $\overline{N}_n \leq N$ and $\overline{E}_n \leq E$ are satisfied. It is no doubt that the value of non-empty netgrids $\overline{N}_n$ is related to the netgrid partition layer and the distribution of satellites. We assume that $\overline{N}_n = \frac{N}{W_l}$, $\overline{E}_n = \frac{E}{W'_l}$, where W_l and W'_l represent the overlap factor and edge overlap factor in partition layer l, respectively. Then $W_l = \frac{N}{\overline{N}_n} \geq 1$ and $W'_l = \frac{E}{\overline{E}_n} \geq 1$ are also satisfied. In this way, the computation complexity of NSR in partition layer l becomes $\mathcal{O}((\frac{N}{W_l})^2 + \frac{E}{W'_l})$.

As a comparison, Dijkstra's SPF algorithm based on TGM is analyzed as follows. After the calculation of topology graph at a given time, Dijkstra's SPF algorithm can search the shortest path with greedy approach. There are two main operations in the algorithm, "Extract-min" and "Relaxation", and the typical computation complexity is $\mathcal{O}(N^2 + E)$ [29].

D. Discussion

In the proposed NSR algorithm, the optimal path is only considered in static topology. There is no doubt that this approach can achieve significant complexity reduction. However, when route time becomes long enough, the shortest path obtained from NSR might be non-optimal. There are two reasons why we do not consider time-varying topology in NSR. One reason is that frequent topology variations especially in large-scale SSNs will cause extremely high computation cost, which is a challenging problem to resource-limited small satellites. The other reason is that only estimated shortest path can be obtained since each satellite only collects status of neighbor satellites, which only provides limited performance improvement.

To complete our work, we provide a simple idea to consider all topology variations during $[t_s, t_t]$ in NSR, where $t_t = t_s + TTL$. In specific, line 12 in Algorithm 1 is changed. That is to say, the algorithm will search all non-empty neighbor netgrids of $\mathcal{C}'$ within *effective netgrids* during $[time(\mathcal{C}'), t_t]$, where $time(\mathcal{C}')$ is the transmission time obtained from current router table. As neighbor netgrids of $\mathcal{C}'$ are static, another operation in order to ensure there exist satellites in neighbor netgrids should be considered. The specific searching condition is $time(\mathcal{C}') < t_t$, then the transmission time calculation in line 8 of Algorithm 3 will be changed to $time(e(\mathcal{C}', \mathcal{C}_n)) = T_w(C_n) + \frac{|D|}{R(\mathcal{C}', \mathcal{C}_n)} + T_p(e(\mathcal{C}', \mathcal{C}_n))$, where $T_w(C_n)$ represents the waiting time to netgrid $\mathcal{C}_n$ and $T_w(C_n) = EnteringTime(\mathcal{C}_n) - time(\mathcal{C}')$ if $EnteringTime(\mathcal{C}_n) >= time(\mathcal{C}')$; otherwise, $T_w(C_n) = 0$.



(a) setting interface (b) operation interface

Fig. 5. GUI of LSNS simulator.

TABLE II
SIMULATION PARAMETERS

Parameters	*Values*
Type of Satellites	LEO
Altitude	780km
Number of Planes	6
Eccentricity	0
Inclination	86.4 degrees
TTL of Packet	10min
Transmission Rate	100kbps
Total Generated Packets	1000
Communication Range	5000km
Simulator Update Interval	0.1s

VI. SIMULATION RESULTS

In this section, we perform a series of simulations to evaluate the performance of the proposed routing algorithm and the efficiency of TNM compared with traditional methods. To better support large-scale satellite network simulations, an extended simulator is also developed. Simulation results show that our routing scheme can approach the optimal routing performance in partition layer 2 and achieve significant computation cost reduction.

A. Configuration

To the best of our knowledge, most of the existing simulators can hardly support large-scale SSNs for execution efficiency problems and lack of satellite orbit module. Thus, an extended simulator based on Opportunistic Network Environment (ONE), called LSNS, is developed as an open-source project [30]. Based on ONE, we develop a new graphical user interface, satellite orbit module, satellite link module and event trigger to achieve scalable satellite network simulation. The specific GUI of the simulator is shown in Fig. 5.

In this simulation, Iridium-like constellation is adopted to construct satellite networks and the software and hardware configurations of simulation platform are: Core i3-4150 CPU, 3.50 GHz, 12G RAM, O.S. Windows 10 Professional 64 bits. The orbit calculation model is two-body model. Since we consider random traffic transmission situation, users' requests in the simulation are generated randomly. We also assume that all satellites have one omni-directional antenna, and each satellite can only connect to one satellite at the same time. The other specific simulation parameters are shown in Table II. Comparison routing schemes are given as follows.

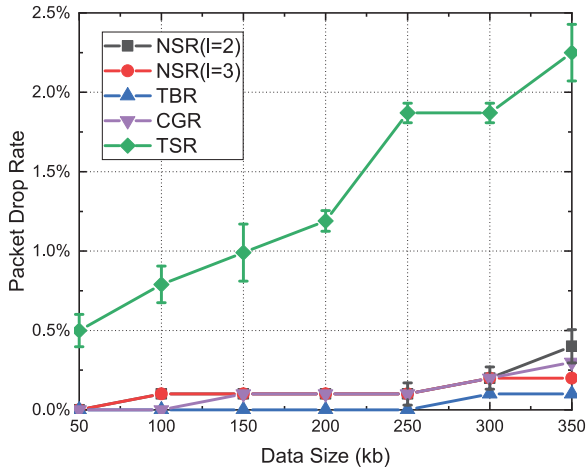
- *Proposed routing (NSR):* NSR algorithm based on TNM in partition layer 2 and layer 3 is performed, respectively. NSR will be firstly executed at the source node to find the optimal routing path. If multi-choice satellites in the path are unreachable, NSR will be executed at intermediate node to find another routing path.
- *TGM based Source Routing (TSR):* TSR only executes routing algorithm once when user's data packet arrives at the source node, and the intermediate nodes only forward the packet by extracting the predetermined routing path information from the packet header.
- *Contact Graph Routing (CGR):* CGR utilizes Dijkstra's SPF algorithm based on the scheduled contact plan, and it executes routing algorithm at each intermediate node during the transmission from the source node to the destination node.
- *TGM based Brute-force Routing (TBR):* TBR adopts brute-force method based on the topology information of TGM to find the optimal routing path, it executes brute-force algorithm at each intermediate node during the transmission.
- *Earliest Arrival Space Routing (EASR):* As our proposed routing algorithm in the previous work [1], EASR also adopts topology information provided by TNM. To justify the improvement of NSR, the EASR algorithm based on TNM (in partition layer 3) is also implemented in the simulation.

Note that we repeat the same simulation 10 times with different 1000 generated packets for each point in routing performance comparisons, and we plot the average value with 95% confidence interval in Fig. 6 and Fig. 7.

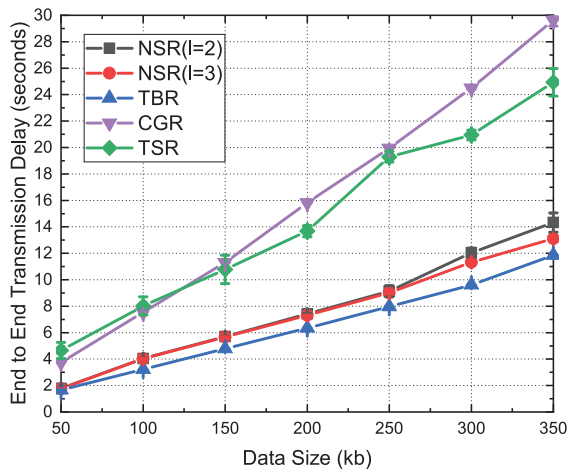
B. Routing Performance

The performance comparison versus different packet size is shown in Fig. 6. Since TBR utilizes brute-force calculation to find the routing path, it always has the optimal performance both in end-to-end delay and packet drop rate. We can see that our routing schemes, which are NSR in partition layer 2 and layer 3, can approach the optimal performance of TBR. In most cases, NSR in partition layer 2 can offer similar performance with NSR in partition layer 3, which proves our routing scheme can achieve good performance in Iridium-like scale of satellite network. When the packet size is exceeded 300 KB, the performance of NSR in partition layer 2 degrades compared with NSR in partition layer 3, this is due to the fact that TNM sacrifices more topology information with the lower number of partition layers. In general, the performance comparison in Fig. 6 validates the effectiveness of TNM and NSR.

Since the average end-to-end delay increases as the packet size increases, the probability of topology's changing would be higher while packets transmitting. For TSR scheme, it only executes the algorithm once at the source node to obtain a routing path. Thus, if there is ISL disconnection in the routing path due to topology variations during the transmission, TSR would wait until ISL reconnects or TTL expires. In this case, there is no doubt that TSR will result in bad routing performance when the packet size becomes larger. For CGR scheme, it executes on-board routing algorithm at each intermediate node, which can adapt to the topology variations while packets transmitting. Thus, the packet drop rate of CGR is steady with the increase

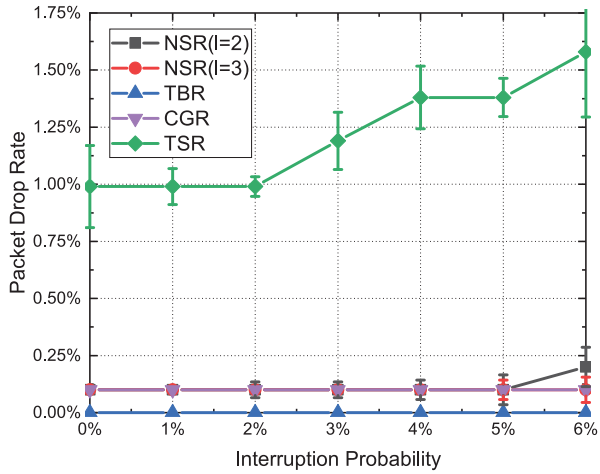


(a) performance in terms of packet drop rate

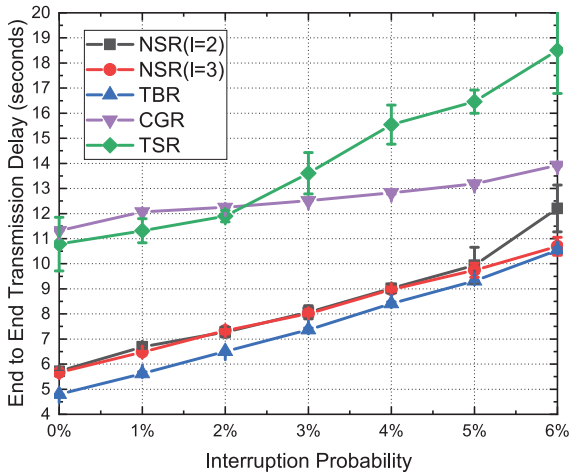


(b) performance in terms of average delay

Fig. 6. Performance comparison versus different packet size among five routing schemes (No Interruption).



(a) performance in terms of packet drop rate



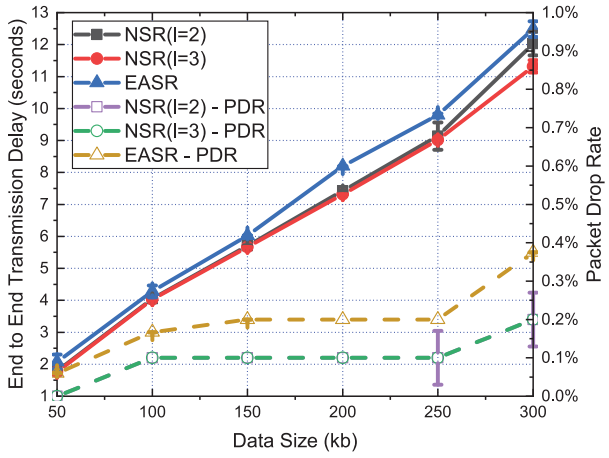
(b) performance in terms of average delay

Fig. 7. Performance comparison versus different interruption probability among five routing schemes (Packet Size = 150 KB).

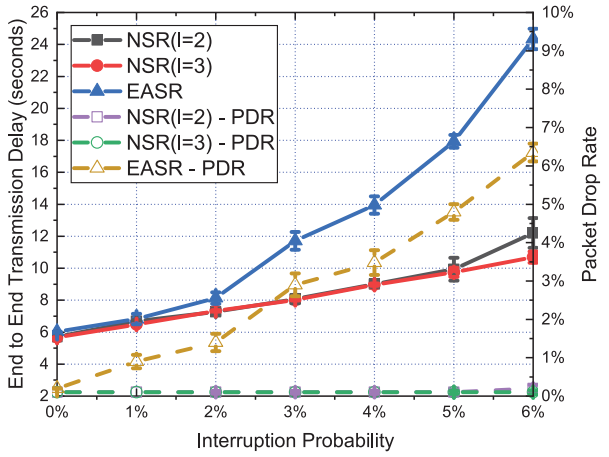
of the packet size. However, those contacts in contact plan still lack flexibility. During the transmission of user's data packets, each intermediate node need to wait for these contacts in contact plan to set up, then the transmission can start. In this way, the delay performance of CGR is still worse than NSR.

Considering the complex environment of the satellite network, intermittent interruptions of ISLs could be existing. Thus, it is necessary to evaluate the ability of the proposed routing scheme to handle this link interruption situation. We depict the performance of all routing schemes under random ISL interruption situation as shown in Fig. 7. The result shows that, with the increase of the interruption probability, the packet drop rate of TBR, NSR and CGR, is more steady compared with TSR. The reason is that these routing schemes can execute re-routing process at intermediate nodes to obtain an alternative routing path when interruption occurs. However, with the increase of the interruption probability, the number of re-routing process is rising. As a consequence, the end-to-end delay of TBR, NSR and CGR also rises. Especially, the performance of our routing schemes, whether packet drop rate or end-to-end delay, can still approach the optimal performance of TBR with different interruption probability, which proves the robustness of NSR.

Since we have proposed EASR based on TNM in our previous work [1], it is necessary to make a performance comparison between NSR and EASR. The simulation results of NSR and EASR are shown in Fig. 8. To make the performance curves easier to follow, solid lines in the figure represent the performance of end-to-end delay, and dotted lines with PDR label in the figure represent the performance of packet drop rate. We can see that NSR outperforms EASR as packet size varies in terms of end-to-end transmission delay and packet drop rate. The performance gap becomes more obvious with the increase of interruption probability. The reason is described as follows. Although EASR utilizes the topology information provided by TNM, it searches future available neighbor nodes at each hop (as described in line 5 of EASR algorithm [1]). Thus, EASR actually predicts the shortest path for each packet. However, considering time-varying network topology and unpredictable queuing time at each intermediate node, this prediction becomes inaccurate

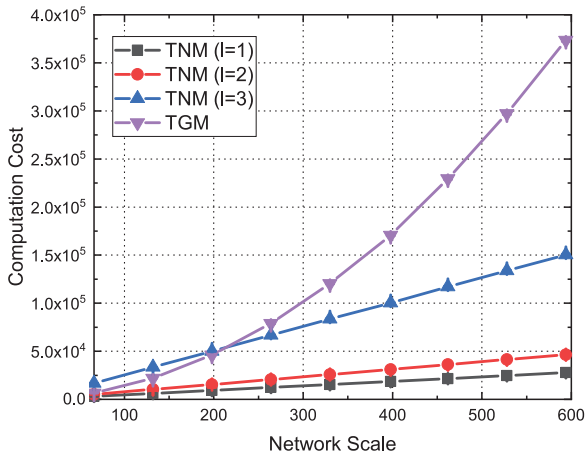


(a) performance versus different packet size

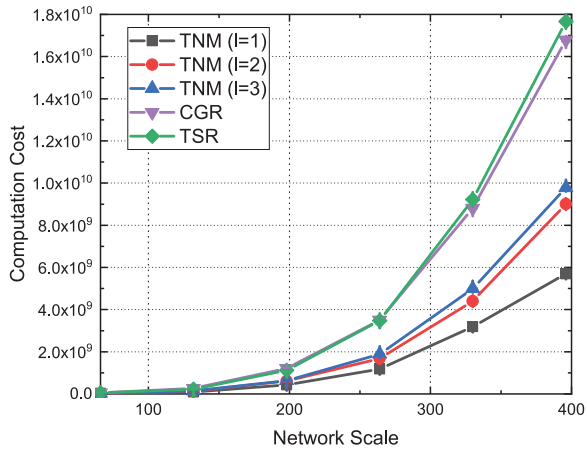


(b) performance versus different interruption probability

Fig. 8. Performance comparison between NSR and EASR.



(a) topology computation cost in terms of c_o



(b) routing algorithm computation cost in terms of c_r

Fig. 9. Computation cost comparison versus different network scale.

and the routing path might be sub-optimal or even invalid. Particularly, interruption will further reduce the accuracy of prediction. Therefore, the increase of interruption probability leads to rapid performance degradation in EASR, as shown in Fig. 8(b). Different from EASR, NSR calculate the routing path at each hop. Besides the topology information provided by TNM, NSR also introduce the beacon protocol to collect status of neighbor nodes for routing calculation.

Although the specific computation complexity of TNM and TGM has been analyzed in our previous work [1], in order to demonstrate the efficiency of NSR and TNM, the comparison of computation cost in different network scales is made as shown in Fig. 9. We repeat the same operation 10000 times for each kind of computation cost in the simulator and plot the average value with 95% confidence interval as the result for each point in Fig. 8. At first, the topology computation cost about TGM and TNM are depicted in Fig. 9(a). Here, c_o, which represents the overhead of single distance judgement in TGM, is set as the unit. As we expected, the topology computation cost of TGM will grow as $\mathcal{O}(N^2)$ with the increase of network scale. For TNM, although the computation cost will rise in higher partition layer, all of them are growing linearly with the increase network scale. Then, in Fig. 9(b), the computation cost for routing algorithm is given. The overhead c_r for single transmission time calculation in line 8 of Algorithm 3 is set as the unit. As we can see, under the same partition layer, the computation cost of NSR grows slower than other algorithms with the increase of network scale. This is due to the fact that the network density is rising with the increase of network scale, then the probability of different satellite nodes locating at the same netgrid will rise as well, i.e., W_l and W_l' become larger. Since the computation complexity of TGM based Dijkstra's SPF and NSR algorithm is $\mathcal{O}(N^2+E)$ and $\mathcal{O}((\frac{N}{W_l})^2+\frac{E}{W_l'})$, respectively, there is no doubt that the difference between their computation cost will rise with the increase of network scale. In conclusion, this result verifies the low complexity of TNM and NSR compared with traditional methods.

VII. Conclusion

In this paper, we developed an efficient and low-complexity routing scheme for large-scale SSNs. Firstly, we proposed a novel topology description model named TNM to describe the time-varying network topology of large-scale SSNs. By using

discrete netgrids to locate satellite and multi-layer netgrid partition method, TNM can provide time-continuous topology information and alternative description accuracy to support random traffic transmission requirements. Based on the topology information provided by TNM, we also proposed a netgrid-based routing algorithm to find the optimal path in large-scale SSNs. Thanks to the status collected by the beacon protocol, NSR has the ability to adjust routing path according to the status of neighbor satellites. Compared with traditional routing algorithm, our analysis showed that the computation complexity can be reduced from $\mathcal{O}(N^2+E)$ to $\mathcal{O}((\frac{N}{W_l})^2+\frac{E}{W_l})$. We developed an extended simulator called LSNS to evaluate the performance of NSR in large-scale SSNs. Simulation results showed that the proposed routing algorithm can approach the optimal routing performance with lower computation cost.

REFERENCES

[1] J. Li, H. Lu, and Y. Wang, "Temporal netgrid model based routing optimization in satellite networks," in *Proc. IEEE Int. Conf. Commun.*, 2017, pp. 1–6.

[2] H. Nishiyama, Y. Tada, and N. Kato, "Toward optimized traffic distribution for efficient network capacity utilization in two-layered satellite networks," *IEEE Trans. Veh. Technol.*, vol. 62, no. 3, pp. 1303–1313, Mar. 2013.

[3] E. K. Jaff, P. Pillai, and Y. F. Hu, "IP multicast receiver mobility support using PMIPv6 in a global satellite network," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 30–37, Mar. 2015.

[4] G. Araniti, I. Bisio, M. de Sanctis, A. Orsino, and J. Cosmas, "Multimedia content delivery for emerging 5G-satellite networks," *IEEE Trans. Broadcast.*, vol. 62, no. 1, pp. 10–23, Mar. 2016.

[5] X. Jia, T. Lv, F. He, and H. Huang, "Collaborative data downloading by using inter-satellite links in LEO satellite networks," *IEEE Trans. Wireless. Commun.*, vol. 16, no. 3, pp. 1523–1532, Mar. 2017.

[6] F. He, Q. Liu, T. Lv, C. Liu, H. Huang, and X. Jia, "Delay-bounded and minimal transmission broadcast in LEO satellite networks," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–7.

[7] Y. Xie and Y. Fang, "A general statistical channel model for mobile satellite systems," *IEEE Trans. Veh. Technol.*, vol. 49, no. 3, pp. 744–752, Mar. 2000.

[8] N. P. Joseph and J. Bernard, "Distributed internet-optimized services via satellite constellations," in *Handbook of Satellite Applications*, J. Pelton, S. Madry, S. Camacho-Lara, Eds., Cham, Switzerland: Springer, 2017, pp. 249–269.

[9] T. Lee, "Building technical communities for the benefit of humanity," *IEEE Microw. Mag.*, vol. 17, no. 1, pp. 8–11, Jan. 2016.

[10] J. Radtke, C. Kebschull, and E. Stoll, "Interactions of the space debris environment with mega constellations-using the example of the OneWeb constellation," *Acta Astronautica*, vol. 131, pp. 55–68, 2017.

[11] A. Pérez-Neria, M. A. Lagunas, and M. A. Vázquez, "High throughput satellites in 5G and MIMO interference limited communications," in *Proc. 20th Int. Conf. Circuits, Syst., Commun. Comput.*, 2016, Paper 03008.

[12] M. Werner, "A dynamic routing concept for ATM-based satellite personal communication networks," *IEEE J. Sel. Areas Commun.*, vol. 15, no. 8, pp. 1636–1648, Oct. 1997.

[13] D. Zhou, M. Sheng, X. Wang, C. Xu, R. Liu, and J. Li, "Mission aware contact plan design in resource-limited small satellite networks," *IEEE Trans. Commun.*, vol. 65, no. 6, pp. 2451–2466, Jun. 2017.

[14] G. Song, M. Chao, B. Yang, and Y. Zheng, "TLR: A traffic-light-based intelligent routing strategy for NGEO satellite IP networks," *IEEE Trans. Wireless Commun.*, vol. 13, no. 6, pp. 3380–3393, Jun. 2014.

[15] I. F. Akyildiz, E. Eylem, and M. D. Bender, "MLSR: A novel routing algorithm for multilayered satellite IP networks," *IEEE/ACM Trans. Netw.*, vol. 10, no. 3, pp. 411–424, Jun. 2002.

[16] C. Chen and E. Ekici, "A routing protocol for hierarchical LEO/MEO satellite IP networks," *Wireless Netw.*, vol. 11, no. 4, pp. 507–521, 2005.

[17] H. S. Chang *et al.*, "FSA-based link assignment and routing in low-earth orbit satellite networks," *IEEE Trans. Veh. Technol.*, vol. 47, no. 3, pp. 1037–1048, Aug. 1998.

[18] J. Huang, Y. Su, L. Huang, W. Liu, and F. Wang, "An optimized snapshot division strategy for satellite network in GNSS," *IEEE Commun. Lett.*, vol. 20, no. 12, pp. 2406–2409, Dec. 2016.

[19] S. E. Alaoui and B. Ramamurthy, "Routing optimization for DTN-based space networks using a temporal graph model," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–6.

[20] W. Shi, Q. Xu, B. Feng, and H. Zhou, *A Mechanism Coping With Unexpected Disruption in Space Delay Tolerant Networks*, *IETF Draft*, 2018. [Online]. Available: https://datatracker.ietf.org/doc/draft-shi-dtn-amcud/

[21] G. Araniti *et al.*, "Contact graph routing in DTN space networks: Overview, enhancements and performance," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 38–46, Mar. 2015.

[22] C. Caini and R. Firrincieli, "Application of contact graph routing to LEO satellite DTN communications," in *Proc. IEEE Int. Conf. Commun.*, 2012, pp. 3301–3305.

[23] J. A. Fraire, P. G. Madoery, and J. M. Finochietto, "On the design and analysis of fair contact plans in predictable delay-tolerant networks," *IEEE Sensors J.*, vol. 14, no. 11, pp. 3874–3882, Nov. 2014.

[24] T. Taleb, D. Mashimo, A. Jamalipour, N. Kato, and Y. Nemoto, "Explicit load balancing technique for NGEO satellite IP networks with on-board processing capabilities," *IEEE/ACM Trans. Netw.*, vol. 17, no. 1, pp. 281–293, Feb. 2009.

[25] Y. Rao and R. Wang, "Agent-based load balancing routing for LEO satellite networks," *Comput. Netw.*, vol. 54, no. 17, pp. 3187–3195, 2010.

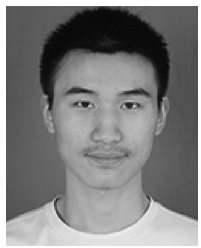
[26] Y. Zhou, F. Sun, and B. Zhang,"A novel QoS routing protocol for LEO and MEO satellite networks," *Int. J. Satell. Commun. Netw.*, vol. 25, no. 6, pp. 603–617, 2007.

[27] Y. Kawamoto, H. Nishiyama, N. Kato, and N. Kadowaki, "A traffic distribution technique to minimize packet delivery delay in multilayered satellite networks," *IEEE Trans. Veh. Technol.*, vol. 62, no. 7, pp. 3315–3324, Sep. 2013.

[28] P. Holme and J. Saramaki, "Temporal networks," *Phys. Rep.*, vol. 519, no. 3, pp. 97–125, 2012.

[29] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2001.

[30] *Large-Scale Satellite Networks Simulator*, *Github*, 2019. [Online]. Available: https://github.com/infonetlijian/ONE-Extended-Simulator



Jian Li received the B.S. degree from Anhui University, Hefei, China, in 2015. He is currently working toward the Ph.D degree in communication and information systems with the Department of Electronic Engineering and Information Science, University of Science and Technology of China. His research interests include satellite networks, cooperative transmission in integrated satellite/terrestrial networks, and cache-enabled wireless heterogeneous networks.

Hancheng Lu (M'07) received the Ph.D. degree in communication and information systems from the University of Science and Technology of China (USTC), Hefei, China, in 2005. He is currently an Associate Professor with the Department of Electronic Engineering and Information Science, USTC. His research interests include resource management in wireless heterogeneous networks and routing and scheduling in delay/disruption-tolerant networks.

Kaiping Xue (M'09–SM'15) received the B.S. degree from the University of Science and Technology of China (USTC), Hefei, China, and the Ph.D. degree from USTC, in 2003 and 2007, respectively. From May 2012 to May 2013, he was a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA. He is currently an Associate Professor with the Department of Information Security and Department of EEIS, USTC. His research interests include next-generation Internet, distributed networks, and network security.

Yongdong Zhang (A'09–SM'13) received the Ph.D. degree in electronic engineering from Tianjin University, Tianjin, China, in 2002. He is currently a Professor with the Department of Electronic Engineering and Information Science, University of Science and Technology of China. He has authored over 100 refereed journal and conference papers. His current research interests include multimedia content analysis and understanding, multimedia content security, video encoding, and streaming media technology.