IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 30, 2020, accepted November 17, 2020, date of publication November 30, 2020,
date of current version December 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3041346

A Distributed Survivable Routing Algorithm for Mega-Constellations With Inclined Orbits

XIAOXIN QI, BING ZHANG, (Member, IEEE), AND ZHILIANG QIU
State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China
Corresponding author: Bing Zhang (bzhang@mail.xidian.edu.cn)

ABSTRACT Mega-constellations consisting of hundreds to thousands of low-earth-orbit (LEO) satellites are an attractive solution for providing global ubiquitous network access. Due to good coverage properties for populated areas, inclined orbits are gaining popularity among commercial constellations. A scalable routing algorithm with survivability plays a key role in such systems. In this paper, we propose a distributed survivable routing algorithm for mega-constellations with inclined orbits. First, the special topology characteristic of inclined constellations is identified and formalized. Based on the topology characterization, a basic X-Y routing algorithm is presented to determine multiple primary and secondary paths towards each destination utilizing the regularity of the network topology with minimal computation overhead. Then, a failure recovery mechanism which consists of a restricted flooding mechanism and a pre-detour mechanism is proposed to reduce end-to-end delay and signaling overhead in case of link failures. Besides, a partial-record loop avoidance mechanism is proposed to deal with routing loops with minimal overhead. Finally, a vector-based next hop selection mechanism is proposed to facilitate the selection of next hop while incorporating various criteria. The performance of the proposed routing algorithm is evaluated through simulation on the Starlink constellation. Simulation results show that our proposal achieves scalability by reducing signaling overhead and provides better quality of service in terms of end-to-end delay under link failures.

INDEX TERMS Inclined constellation, mega-constellation, routing, survivability.

I. INTRODUCTION

With the reduction of cost for manufacturing and launching satellites, low-earth-orbit (LEO) satellite constellations are regaining popularity from industry and the era of so called Internet of satellites is coming. Different from their earlier version in the late 1990s, LEO constellations today feature a larger number of satellites, hence the name ''mega-constellations''. Larger constellations allow higher frequency reuse efficiency, which brings larger system capacity. Several mega-constellations are under development, with Starlink and OneWeb being the most notable [1].

There are two system architectures for a satellite network [2]. The first is the ''bent-pipe'' architecture in which the satellites only act as transponders between ground stations. The second is the space-based architecture where satellites are equipped with inter-satellite links (ISLs). The space-based systems have several advantages such as lower end-to-end delay, less dependence on the ground infrastructure and improved system capacity [1]. However, one of the difficulties of such a system is how to develop an efficient and robust routing algorithm for the space segment.

Although many routing algorithms have been proposed for LEO satellite networks, they may face scalability problem when applied to mega-constellations. With hundreds to thousands of satellites, the routing algorithm needs to be efficient. Traditional routing algorithms that compute paths based on Dijkstra algorithm may incur large computation overhead. Therefore, a distributed routing algorithm which utilizes the predictability and regularity of the network topology is preferred for a mega-constellation. Besides, the ISLs may suffer from various kinds of failures in the complex space environment [3]. The routing algorithm should be able to alleviate the impact of link or node failures as much as possible. Traditional flooding-based routing protocol would incur large signaling overhead, especially in a mega-constellation. On the contrary, a local detouring mechanism can avoid large signaling overhead, but would introduce extra delay to the packets due to the lack of a global view. Therefore, an efficient routing algorithm with fault-tolerant ability which

The associate editor coordinating the review of this manuscript and approving it for publication was Bijoy Chand Chatterjee.

does not degrade quality of service significantly is urged for recently proposed mega-constellations.

Based on the inclination of the satellite orbits, LEO constellations can be classified into polar constellations and inclined constellations. Polar constellations have an orbit inclination of nearly 90 degrees. In a polar constellation, the ISLs are typically switched off in polar regions due to the strict requirement of antenna tracking and switched on again outside the polar regions. Thus, the topology of polar constellations varies with time. Inclined constellations have an orbit inclination of less than 90 degrees. Through proper selection of ISLs, the topology can remain permanent in an inclined constellation [4]. A permanent topology facilitates the space segment routing. Besides, inclined constellations provide better coverage at the most-populated mid-latitudes at the cost of less coverage in polar areas [5]. Previous work on routing algorithms in LEO networks is mainly focused on polar constellations whereas inclined constellations receive less attention. Although routing in inclined constellations seems simpler because of their permanent topology, inclined constellations show special topology characteristics, which need to be taken into consideration when designing a routing algorithm.

In this paper, we propose a distributed survivable routing algorithm for mega-constellations with inclined orbits. The contributions of this paper are summarized as follows:

(1) The spiral shape formed by inter-plane ISLs in inclined constellations is identified and formalized. Based on this topology characteristic, a basic X-Y routing algorithm is proposed for inclined constellations. Each satellite independently selects multiple primary paths and secondary paths towards each destination utilizing the regularity of the network topology with minimal computation overhead.

(2) A failure recovery mechanism is proposed to deal with link failures which consists of a restricted flooding mechanism and a pre-detour mechanism. In the restricted flooding mechanism, link state changes are flooded only to limited areas to reduce signaling overhead. In the pre-detour mechanism, a satellite detours the packets to alternative paths before actually encountering a link failure to reduce additional end-to-end delay.

(3) A partial-record loop avoidance mechanism is proposed to deal with loops in the routing process. In this mechanism, satellites traversed by a packet are recorded in the packet header only when necessary. This mechanism reduces the overhead of recording traversed satellites, especially when the network is not suffering severe link failures.

(4) A vector-based next hop selection mechanism is proposed to facilitate the selection of next hop while incorporating various criteria including the selection between primary and secondary paths, the pre-detouring, and loop avoidance.

(5) The performance of the proposed routing algorithm is evaluated through simulation based on the Starlink constellation. Simulation results demonstrate the ability of the proposed routing algorithm to reduce signaling overhead and reduce end-to-end delay in case of link failures.

The rest of the paper is organized as follows. Section II reviews related work on routing algorithms for satellite constellations. In section III, the special topology characteristics of inclined constellations are investigated. In section IV, the proposed distributed survivable routing algorithm is presented in detail. The performance of the proposed routing algorithm is evaluated in section V and conclusions are drawn in Section VI.

II. RELATED WORK

In this section, we present related work on routing algorithms in LEO satellite networks and focus on distributed routing and failure recovery mechanisms.

In a distributed routing algorithm, each node in the network determines next hop towards other nodes independently without the help of a central node. The most common distributed routing algorithm is the link state algorithm such as OSPF, which requires global link state to compute shortest paths. Link state algorithms have been applied to LEO satellite networks [6]. However, such an algorithm would incur prohibitively large computation and communication overhead for large-scale networks [7]. Especially, when there is a single link state change, it is unnecessary to flood the information to the whole network and reconstruct the routing table of every node.

Given the regularity and predictability of satellite constellations, more efficient distributed routing algorithms can be implemented. Several distributed algorithms for LEO satellite networks have been proposed. DRA [8] is the earliest work on distributed routing in LEO networks. In DRA, each satellite independently determines the next hop towards the destination based on the logical positions of the satellites without using Dijkstra algorithm. In [9], the authors proposed a longer side priority (LSP) routing strategy. The main idea is to forward packets to the direction with more remaining hops to reduce congestion. The authors in [10] proposed a Semi-Distributed Routing Algorithm (SDRA) to reduce the computation overhead. The source node determines both the next hop and the hop after the next. It attaches the hop after the next in the packet header, so the next hop does not need to conduct path calculation. This process continues until the packet reaches the destination. These algorithms are mainly designed for polar constellations. As we will see in the next section, inclined constellations show different topology characteristics, which should be taken into account when designing a routing algorithm.

The satellite networks are exposed to link or node failures due to several reasons such as electromagnetic interference and military strike. Several mechanisms are proposed to cope with such situations from the perspective of routing algorithm. In DRA, once the optimal next hop is unavailable, the packets are detoured to the suboptimal next hop. However, this failure recovery mechanism is localized and may incur additional end-to-end delay, as shown in section IV. In [11], a routing protocol for hierarchical LEO/MEO satellite networks is proposed. The MEO layer is responsible for

collecting link state and calculating routing tables for the LEO layer. Once there is a link or node failure in the LEO layer, the state is flooded to MEO satellites which re-compute routing tables for the LEO layer. In [12], the authors proposed a deadlock-free fault-tolerant adaptive routing based on minimal-connected-component (MCC) faulty model for LEO satellite networks. In [3], a survivable routing protocol for two-layered LEO/MEO satellite networks is proposed. The protocol deals with three failure cases, i.e., LEO satellite failure, MEO satellite failure and MEO layer failure. Above failure recovery mechanisms require link state flooding in the network, either in the LEO layer or in the MEO layer, which faces scalability problems and limits their applicability in mega-constellations.

There are a few papers that are concerned with mega-constellations. References [13], [14] and [15] are representative work. References [13] and [14] evaluated the end-to-end delay performance of mega-constellations and [15] investigated the topology design problem in mega-constellations. However, routing algorithm is not the focus of these papers and they all used the shortest path algorithm to compute end-to-end paths. As mentioned earlier, such a routing algorithm would incur large computation and signaling overhead in mega-constellations.

In summary, an efficient survivable routing algorithm for large-scale inclined constellations is needed, which is the focus of this work.

III. NETWORK MODEL

In this section, we present the satellite constellation network model and the topology characterization of inclined constellations, which are the foundation of the routing algorithm proposed in the next section.

A. SATELLITE CONSTELLATION

A satellite constellation comprises $N \times M$ satellites distributed in circular orbits, where N is the number of orbits and M is the number of satellites in each orbit. The phase factor, denoted as F, determines the phase offset between neighboring satellites in adjacent orbits and takes values from $\{0, \ldots, N-1\}$. LEO constellations can be classified into polar (Walker star) constellations and inclined (Walker delta) constellations according to their orbit inclination. Polar constellations have an orbit inclination of nearly 90 degrees and inclined constellations have an orbit inclination significantly less than 90 degrees. In an inclined constellation, the N orbits are spaced along a complete circle, hence the name 2π-constellation [5].

Satellites can communicate with each other with ISLs, which can be further divided into intra-plane ISLs and inter-plane ISLs. Intra-plane ISLs connect satellites in the same orbit and their length and direction keep constant. Inter-plane ISLs connect satellites in adjacent orbits and have varying length and direction with the movement of the satellites. Due to constraint of link distance, each satellite is typically equipped with four bidirectional ISLs: two intra-plane ISLs and two inter-plane ISLs. As optical ISLs have lots of advantages over RF ISLs [16], there is a trend to adopt optical ISLs in satellite constellations. However, optical ISLs face challenging problems of pointing, acquisition and tracking (PAT). In an inclined constellation, with proper selection of the inter-plane ISLs, the topology of the constellation can keep constant, which facilitates the use of optical ISLs. In this paper, we focus on inclined constellations.

In [17], the authors proposed using zero phase factor in an inclined constellation to obtain stable inter-plane ISLs, which produces a bi-directional Manhattan street network (MSN). In [18], the authors deduced the parameter combinations that achieve a figure-of-eight ring shape of inter-plane ISLs in inclined constellations, which also produce a bi-directional MSN structure. The use of a bi-directional MSN structure can simplify routing. However, the system parameters depend on many factors including coverage, collision avoidance between satellites, among others. Thus, in an actual constellation, the bi-directional MSN structure may not be obtained. In the next subsection, we present a generalization of the topology of an inclined constellation.

B. TOPOLOGY CHARACTERIZATION

The topology characteristics of a satellite constellation depend largely on the pattern of ISLs. Because the intra-plane ISLs are stable, we focus on inter-plane ISLs. Let (n, m) be the mth satellite in the nth orbit, $n = 0, \ldots, N-1$; $m = 0, \ldots, M-1$. We call m the satellite number and n the orbit number of the satellite. The phase offset between neighboring satellites in adjacent orbits, e.g., between $(0, 0)$ and $(1, 0)$, is $\phi_1 = 2\pi F/(MN)$. The phase offset between neighboring satellites in the same orbit, i.e., between (n, m) and $(n, \mathrm{mod}(m+1, M))$ is $\phi_2 = 2\pi/M$.

Suppose satellite (n, m) is connected to $(n+1, m)$, $n = 0, \ldots, N-2$ via inter-plane ISLs, i.e., the satellites in adjacent orbits with the same satellite number are connected. An exception occurs for the $(N-1)$th orbit because with the accumulation of phase offset, $(N-1, m)$ is not necessarily connected to $(0, m)$. In the following theorem, we will show that the satellite number is offset by F after the inter-plane ISLs traverse N orbits.

Theorem 1: Suppose satellite (n, m) is connected to $(n+1, m)$, $n = 0, \ldots, N-2$ via inter-plane ISLs, then $(N-1, m)$ is connected to $(0, \mathrm{mod}(m+F, M))$.

Proof: Suppose $(N-1, m)$ is connected to $(0, s)$. The phase offset between $(0, s)$ and $(0, m)$ can be calculated as $\phi_3 = \phi_1 \times N = 2\pi F/M = F\phi_2$. Thus, $s = \mathrm{mod}(m+F, M)$, i.e., the satellite number is offset by F after the inter-plane ISLs traverse N orbits. □

Based on the above theorem, the inter-plane ISLs will form a *spiral* shape. In Fig. 1, the first phase of Starlink constellation with 32×50 satellites is shown. The yellow line is part of this spiral formed by 32 contiguous inter-plane ISLs. If we continue interconnecting satellites by such inter-plane ISLs, a closed spiral will be formed. In Fig. 2, the abstracted topology of a smaller constellation with 8×6 satellites is

shown to demonstrate this concept. By simple calculation, the total number of closed spirals in a constellation equals the greatest common divisor of M and F. For example, in Fig. 2, $M = 6, N = 8$ and $F = 3$, so there are 3 closed spirals which are marked by different colors.

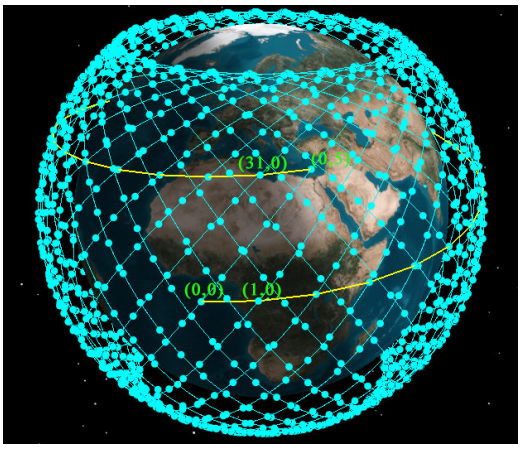

FIGURE 1. The spiral shape in Starlink constellation.

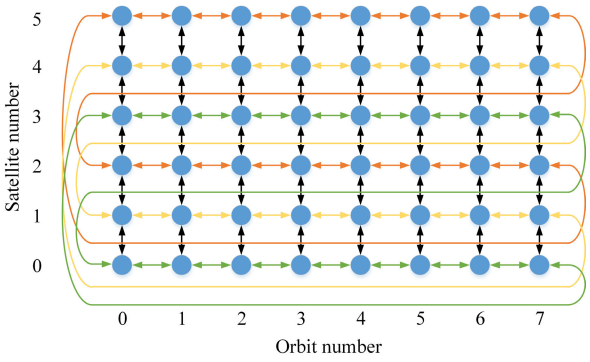

FIGURE 2. Closed spirals in an 8 × 6 constellation.

So far we have discussed the case where (n, m) is connected to $(n + 1, m)$, $n = 0, \ldots, N - 2$. However, in practice, the inter-plane ISLs can be selected such that they provide better connectivity for a certain direction [19]. For example, the inter-plane ISLs shown in Fig. 1 provide good East-West connectivity. One can connect satellites in adjacent orbits whose satellite numbers are offset by -2 (e.g., connect $(0, 0)$ and $(1, 48)$) to achieve better North-South connectivity, as shown in Fig. 3. The spiral shape and the offset of satellite number under such situation can still be formalized by renumbering the satellites, as shown by the following theorem.

Theorem 2: Suppose (n, m) is connected to $(n+1, \text{mod}(m+\delta+M, M))$, $n = 0, \ldots, N-2$ via inter-plane ISLs. Renumber each satellite (n, m) as (n, m') with (1):

$$m' = \begin{cases} m, & n = 0 \\ \text{mod}(m - n\delta + kM, M), & n \neq 0, \end{cases} \tag{1}$$

where k is the minimum nonnegative integer that satisfies $m - n\delta + kM \geq 0$. Then (n, m') is connected to $(n + 1, m')$, $n = 0, \ldots, N-2$. In the renumbered constellation, $(N-1, m')$ is connected to $(0, \text{mod}(m' + F + \delta N + kM, M))$, where k is the minimum nonnegative integer that satisfies $m' + F + \delta N + kM > 0$.

Proof: The phase offset between (n, m') and $(n + 1, m')$ now becomes $\phi_1' = \phi_1 + \delta\phi_2$. Again suppose $(N - 1, m')$ is connected to $(0, s)$. Then the phase offset between $(0, s)$ and $(0, m')$ can be calculated as $\phi_3' = \phi_1' \times N = (F+\delta N)\phi_2$. Thus, $s = \text{mod}(m' + F + \delta N + kM, M)$, where k is the minimum nonnegative integer that satisfies $m' + F + \delta N + kM > 0$. □

For example, in Fig. 3, $\delta = -2$ and $s = \text{mod}(m' + 41, 50)$. For $m' = 0$, we have $s = 41$, i.e., $(31, 0)$ is connected to $(0, 41)$.

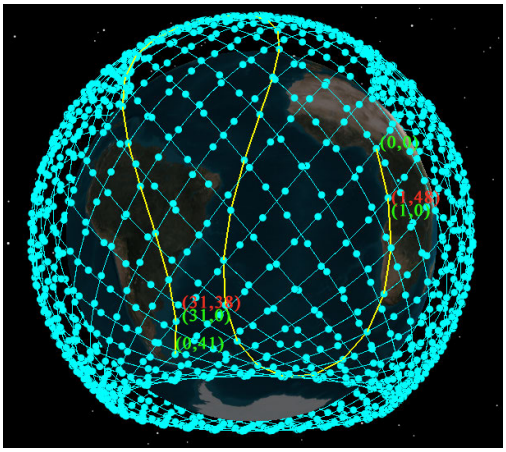

FIGURE 3. Inter-plane ISLs that achieve better North-South connectivity (red: before renumbering; green: after renumbering).

IV. DISTRIBUTED SURVIVABLE ROUTING ALGORITHM

In this section, a distributed survivable routing algorithm for inclined constellations is proposed. First, a basic X-Y routing is proposed in which multiple primary and secondary paths towards each destination are determined. Then, a failure recovery mechanism which consists of a restricted flooding mechanism and a traffic pre-detour mechanism is proposed to deal with link failures. In order to cope with routing loops, a partial-record loop avoidance mechanism is proposed. Finally, a vector-based next hop selection mechanism is proposed to incorporate various criteria efficiently when the next hop is decided.

A. BASIC X-Y ROUTING

In our proposed distributed routing algorithm, each satellite independently selects the next hop for forwarding a packet it generated or received. The dominant part of the end-to-end delay that a packet experiences in a satellite network is the propagation delay. Therefore, the main concern is to forward the packets along a minimum hop count path. The mesh-like topology of a satellite constellation can be utilized to compute routes efficiently without using common routing algorithms such as Dijkstra shortest path (DSP) algorithm. For any source-destination pair, one or two X-Y paths with

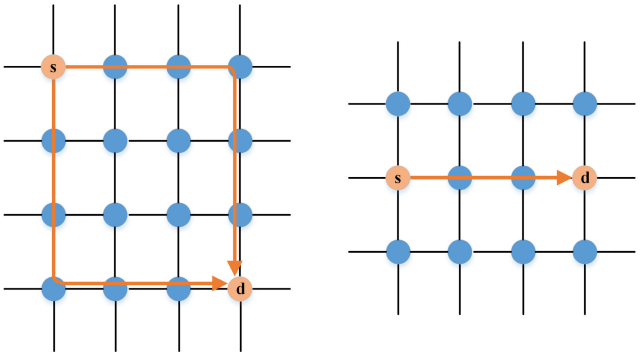

FIGURE 4. Minimum hop count X-Y paths in a satellite constellation.

minimum hop count exist, as shown in Fig. 4. The X component means forwarding a packet along inter-plane ISLs and takes values as *left*, *right* or *zero* (*zero* means no inter-plane ISL is traversed); the Y component means forwarding a packet along intra-plane ISLs and takes values as *up*, *down* or *zero* (*zero* means no intra-plane ISL is traversed). Finding an X-Y path is to determine the X component and Y component. Note that for a pair of nonzero X component and Y component, two X-Y paths exist based on whether to take the X component first or to take the Y component first. These two paths have the same hop count. In a bidirectional MSN network, the minimum hop count X-Y path(s) can be obtained by comparing the orbit numbers and satellite numbers of the two satellites [20]. However, as pointed out in section III, an inclined constellation features the offset of satellite number. For example, in Fig. 5, for source (0, 0) and destination (19, 0), there exist two X-Y paths with different X component (one goes left and the other goes right, as shown by the red line and the yellow line, respectively) although the two satellites have the same satellite number. Thus, the offset of satellite number needs to be taken into account when computing an X-Y path in an inclined constellation.

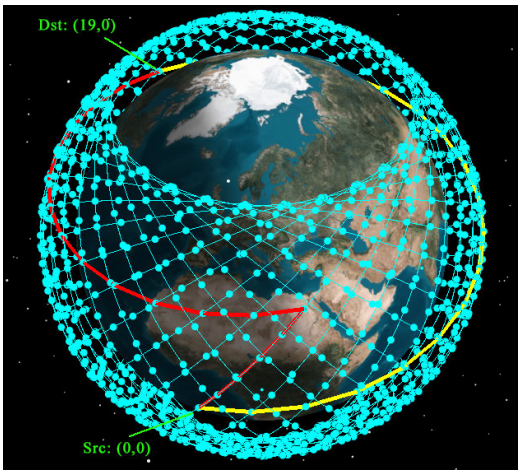

FIGURE 5. Illustration of two X-Y paths with different X component.

To describe the proposed routing algorithm, some definitions are given below first.

(1) Primary direction: Among the four directions that a packet can take (i.e., up, down, left and right), a primary direction is one that takes the packet closer to the destination.

(2) Secondary direction: A secondary direction is one that does not take the packet closer to the destination.

(3) Primary path: A primary path is a path whose next hop from the current satellite is a primary direction.

(4) Secondary path: A secondary path is a path whose next hop from the current satellite is a secondary direction.

In the proposed X-Y routing, each satellite determines one or two primary X-Y paths and secondary paths towards each destination. Suppose the current satellite is s and the destination is d. The primary X-Y paths are determined as follows.

(1) If s and d are on the same orbit, then there is only one primary X-Y path which is along the orbit. The X component is zero and the Y component is determined by the satellite number of s and d. The corresponding Y component is selected as the primary direction.

(2) If s and d are not on the same orbit, then the X component can be either *left* or *right*. Let P_l be the X-Y path with X component of *left* and P_r be the X-Y path with X component of *right*. For each path, the Y component is determined by the satellite number of s and d. Let L_l and L_r be the hop count of P_l and P_r respectively. Then we select the shorter one to be the primary X-Y path. Note that if both the X component and Y component of the primary path are nonzero, then there is another primary path with the same X component and Y component but in the reverse order. The corresponding X component and Y component are selected as primary directions.

With the above procedure, the primary paths and primary directions are obtained, which lead packets closer to the destination. However, to deal with link failures, two secondary directions are determined for each satellite pair which are used as backup directions. The secondary directions and paths are determined as follows.

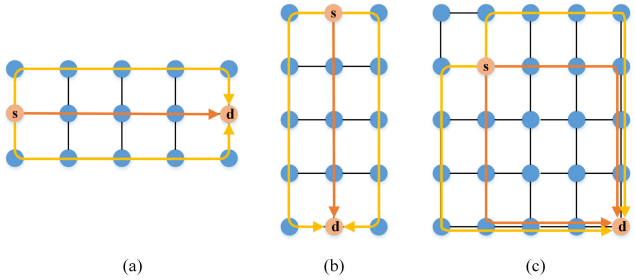

FIGURE 6. Primary and secondary paths in the basic X-Y routing.

(1) If there is one primary path (as shown by the orange line in Fig. 6(a) and (b)), i.e., s and d are on the same line formed by intra-plane ISLs or inter-plane ISLs, then the two directions orthogonal to the primary direction are selected as secondary directions. The secondary paths containing the secondary directions are shown as yellow lines in Fig. 6.

(2) If there are two primary paths (as shown by the orange lines in Fig. 6(c)), then the two directions besides the two primary directions are selected as secondary directions. The secondary paths containing the secondary directions are shown as yellow lines in Fig. 6.

The calculation of primary and secondary paths yields minimal computation overhead because the regularity of the topology can be utilized. Note that a satellite does not need to record every node on each path. It just needs to be "aware" of how the path is formed. In the following subsections, the usage of the primary and secondary paths is further described.

B. FAILURE RECOVERY

The routing protocol can be aware of a link failure or link recovery either by a notification from the lower layers or by a detecting mechanism such as sending and receiving "hello" packets to and from neighboring nodes. In case of link failure, some kind of routing strategy is needed to avoid the impact of the failed link. In a distributed routing algorithm, a typical strategy is to detour the packets to other directions in case that the primary direction is suffering a link break. However, path optimality can be violated with such a strategy, which introduces additional delay. For example, in Fig. 7, if the link between node 1 and node 2 is broken and other nodes do not have knowledge of the broken link, then a packet may take a path like the red one. However, if node s knows the broken link and starts to detour in advance, then the packet would take the green path, which is shorter than the red one.

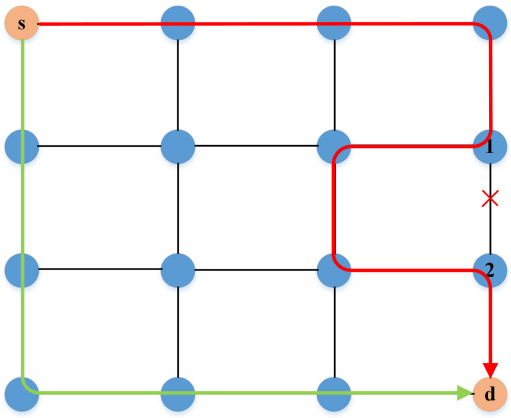

FIGURE 7. Paths with and without pre-detour.

Based on the above observation, we propose a failure recovery mechanism which consists of a restricted flooding mechanism and a pre-detour mechanism to deal with link failures. Our goal is to reduce the signaling overhead of flooding link state and to reduce additional delay for detouring the packets. The main idea is to flood the link state information to neighbors within a limited hop count away and to pre-detour the packets to avoid actually encountering the failed link. Note that a satellite failure can be viewed as a special case of link failure where the four links connected to the satellite are all broken.

1) RESTRICTED FLOODING MECHANISM

The restricted flooding mechanism consists of the following procedure.

(1) Once a link fails, each of the two nodes that are connected to this link generates a link state packet containing the information of the failed link. The link state packet is flooded to neighbors that are within f_{ttl} hops away. This is achieved by adding a Time To Live (TTL) field in the packet and decrementing its value by 1 with each forwarding. Each link state packet has a sequence number associated with the source of the packet to suppress the flooding storm.

(2) For each primary and secondary path towards a destination, a satellite maintains a state variable, N_{block}, indicating the number of broken links on this path. Once a satellite receives a link state packet indicating a link break, it checks for each destination whether the broken link is on any of its primary and secondary paths. If so, the corresponding N_{block} is incremented by one.

(3) When a link recovers, each of the two nodes that are connected to this link generates a link state packet containing the information of the recovered link and floods it to neighbors that are within f_{ttl} hop away. Once a satellite receives a link state packet indicating a link recovery, it checks for each destination whether the recovered link is on any of its primary and secondary paths. If so, the corresponding N_{block} is decremented by one.

Note that when a satellite receives a link state packet, it needs to determine whether or not the indicated link is on a path. It can achieve this by simple calculation based on the index of the satellites due to the regularity of the network topology. Therefore, a satellite does not need to record every node on each path that is determined in the basic X-Y routing.

Let us take Fig. 7 as an example. If the link between node 1 and node 2 fails, then node 1 and node 2 would be aware of that and each generate a link state packet indicating the broken link. Suppose $f_{ttl} = 5$, then node s would receive the link state packet. For destination d, the value of N_{block} for the first primary path (i.e., *right* direction first and *down* direction next) would be incremented by one, whereas the value of N_{block} for the second primary path (i.e., *down* direction first and *right* direction next) would remain unchanged.

2) PRE-DETOUR MECHANISM

When a satellite generates or receives a data packet, it determines the next hop for forwarding the packet based on the state of each available path. Suppose the current satellite is s and the destination is d. Two cases are discussed separately.

(1) If there is one primary path from s to d, then the criterion for selecting the next hop is as follows.

- If the primary path has no broken link, i.e., $N_{block} = 0$, then the primary direction is given priority over other directions.
- If the primary path has any broken link, i.e., $N_{block} > 0$, then the two secondary paths are taken into consideration and s gives priority to the path without link break.

If all of the three paths have any broken link, then s gives priority to the primary direction.

- If the primary direction is unavailable (i.e., the corresponding link is broken), then s gives priority to the secondary directions over the direction that is opposite the primary direction.

(2) If there are two primary paths from s to d, then the criterion for selecting the next hop is as follows.

- Among the four paths (two primary paths and two secondary paths), s gives priority to the two primary paths. Among the two primary paths, s gives priority to the path without broken link.
- If both of the primary directions are unavailable, then s considers the two secondary paths. Among the two secondary paths, s gives priority to the path without broken link.

Note that we say ''give priority to'' here because the direction is not the final next hop and other factors have to be taken into consideration which will be covered in section IV-C. A complete next hop selection mechanism is presented in section IV-D.

For example, in Fig. 7, satellite s prefers *down* direction towards satellite d over *right* direction because N_{block} for *down* direction is 0 and N_{block} for *right* direction is 1.

C. LOOP AVOIDANCE

In a distributed routing algorithm, a common problem is routing loops. Each satellite forwards packets independently without the global view and a packet may thus return back to a satellite that it has traversed. For example, in Fig. 8, a packet from s to d may circulate in a loop shown as the red line. The occurrence of loops causes packet loss and consumes link resources. Therefore, a mechanism to avoid loops during the routing process is needed.

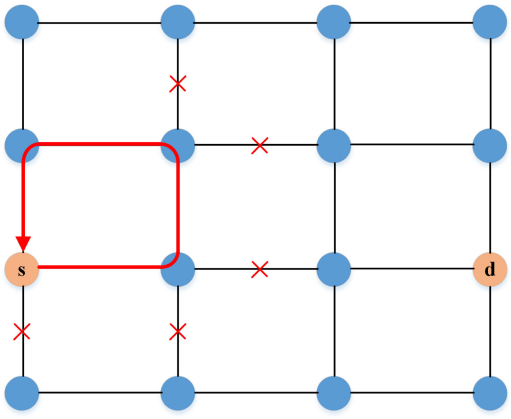

FIGURE 8. Routing loop phenomenon.

Typically, two mechanisms are used to deal with routing loops [21]. The first ensures that a packet is not sent back to the satellite where it just came from. The second sets a TTL field in the packet header so that packets that have traversed too many hops are discarded. However, there are some problems related to these mechanisms. In the first approach, although loops no longer exist between directly connected nodes, they may still happen in a larger ''circle''. On the other hand, the TTL field in the second approach is a mechanism to alleviate the impact of the loop (i.e., to avoid the packet occupying the network resource forever), not a mechanism to ''jump out'' the loop.

In [22], the authors proposed a loop avoidance mechanism along with their routing protocol, which well solved the above problems. Each node traversed by a packet is recorded in the packet header. Once a satellite receives a packet, it checks the packet header to see if the candidate next hop has been traversed. If so, another next hop is selected. If both candidate next hops have appeared in the packet header, the satellite finds the first position where the current satellite's ID appears, and forwards the packet to the node whose ID appears just before the found position. However, with this mechanism, each packet has to record every node it traverses in the packet header. This would incur overhead, especially when the network is in a good state, i.e., when the link failure is sparse and the packets would not suffer loops even without recording the satellites' IDs.

To reduce the overhead of recording each node in the packet header, we propose a partial-record loop avoidance mechanism. The main idea is to record the nodes only when necessary. Accordingly, we call the loop avoidance mechanism proposed in [22] a full-record loop avoidance mechanism. The partial-record loop avoidance mechanism consists of two parts, i.e., setting of the packet header and packet forwarding rule.

1) SETTING OF THE PACKET HEADER

Each packet contains a *nNodeTraversed* field indicating the number of nodes recorded and a list of satellite IDs, *nodeList*, in its header. When a packet originates from the source satellite, *nNodeTraversed* is set to 0 and *nodeList* is set to empty. When forwarding a packet, the satellite records itself in *nodeList* and increments *nNodeTraversed* by 1 when either a) *nNodeTraversed* is not zero, or b) the satellite selects a direction to forward the packet that is not a primary direction.

With this mechanism, the traversed nodes are not necessarily recorded in the packet from the beginning. Only when a satellite selects a direction that is not a primary direction, does it ''trigger'' the recording of the traversed nodes, i.e., the satellite records itself in the packet header and all subsequent satellites along the path record themselves in the packet header. The intuition is that when a satellite selects a direction that is not a primary direction to forward the packet, there is more chance that a loop would happen in the future.

2) PACKET FORWARDING RULE

With the above node-recording mechanism, the packet forwarding rule is enhanced as follows to avoid endless loops. First, define *backtracking node* as follows. When satellite s receives a packet from satellite p, s checks *nodeList* field in the packet header and looks for the position where its own ID first appears.

(a) If the position is 0 (i.e., s is the first node that was recorded in *nodeList*) or s has not been recorded in *nodeList*, then p becomes the backtracking node.

(b) If the position is not 0, then the node whose ID appears before the position becomes the backtracking node.

When deciding the next hop to forward the packet, s prefers next hops that have not been traversed. If all of the available next hops have been traversed, then s forwards the packet to the backtracking node.

With the partial-record loop avoidance mechanism, only a portion of the traversed nodes are recorded in the packet header. Thus the overhead can be reduced, especially when there are only a few link failures in the network. A proof is given in the appendix to show that the proposed loop avoidance mechanism is endless-loop-free.

D. VECTOR-BASED NEXT HOP SELECTION

As discussed above, there are several criteria to be taken into consideration when deciding the next hop for forwarding a packet. These criteria include: a) whether the next hop has been traversed, b) whether the next hop is the backtracking node, c) whether the direction is a primary one, d) whether the path has broken links. These criteria may be intertwined together. However, we have some kind of priority with respect to these criteria when deciding the next hop. For example, a secondary direction that has not been visited is preferred over a primary direction that has been visited. In order to facilitate the selection of the next hop while incorporating all the criteria, we proposed a vector-based next hop selection mechanism. The main idea is to encode the criteria into a numeric vector for each available direction. The next hop can then be selected by comparing the vectors of all the directions.

First, an assessment vector is generated for each available direction when a satellite receives a packet. Suppose the current satellite is s and the destination is d. Two cases are separately dealt with.

(1) If there is one primary path from s to d, then the assessment vector for each direction contains five elements: (*not_traversed*, *backtracking_node*, *no_blockage*, *primary_direction*, *secondary_direction*). Each element is either 0 or 1. The values of these elements are determined as follows:

- *not_traversed*: if the neighboring satellite in the direction has not been traversed (i.e., it is neither the previous hop nor has it appeared in *nodeList*), then this element is set to 1, otherwise 0.
- *backtracking_node*: if the neighboring satellite in the direction is the backtracking node, then this element is set to 1, otherwise 0.
- *no_blockage*: if the direction is either the primary direction or a secondary direction, then this element is set to 1 if $N_{block} = 0$ for the direction and 0 if $N_{block} > 0$; otherwise, this element is set to 0.
- *primary_direction*: if the direction is the primary direction, then this element is set to 1, otherwise 0.
- *secondary_direction*: if the direction is one of the secondary directions, then this element is set to 1, otherwise 0.

(2) If there are two primary paths from s to d, then the assessment vector for each direction contains four elements: (*not_traversed*, *backtracking_node*, *primary_direction*, *no_blockage*). The values of these elements are determined as follows:

- *not_traversed*: if the neighboring satellite in the direction has not been traversed (i.e., it is neither the previous hop nor has it appeared in *nodeList*), then this element is set to 1, otherwise 0.
- *backtracking_node*: if the neighboring satellite in the direction is the backtracking node, then this element is set to 1, otherwise 0.
- *primary_direction*: if the direction is one of the primary directions, then this element is set to 1, otherwise 0.
- *no_blockage*: if $N_{block} = 0$ for the direction, then this element is set to 1, otherwise 0.

After the determination of the assessment vectors, their values are compared with lexicographic ordering. For example, (1, 0, 0, 0) is greater than (0, 1, 1, 1); (0, 1, 0, 0) is greater than (0, 0, 1, 1). The direction with the largest assessment vector is chosen as the final direction to forward the packet.

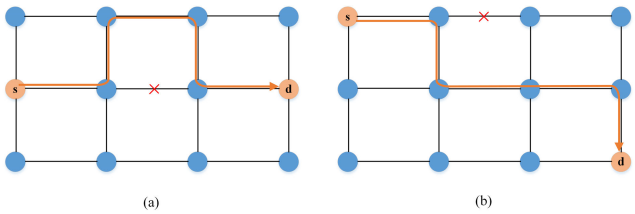

FIGURE 9. Illustration of the priority of the criteria.

Note that in the first case, *no_blockage* is given priority over *primary_direction*. This is because when the primary path has any broken link, the hop count of the final path would increase anyway compared to the hop count of the primary path (as shown in Fig. 9(a)). Therefore, it is reasonable to select a secondary path. In the second case, *primary_direction* is given priority over *no_blockage*. This is because when a primary path has any broken link, the hop count of the final path may not necessarily increase compared to the hop count of the primary path (as shown in Fig. 9(b)). Therefore, we prefer a primary path over a secondary path even when there is any broken link on the primary path.

Let us take Fig. 10 as an example to illustrate the vector-based next hop selection mechanism. Suppose node 1 receives a packet from node s that is destined to d. For each of the available directions (*up*, *down* and *left*) from node 1, the assessment vector is determined. The assessment vector for direction *down* has the largest value. Therefore, node 1 forwards the packet to down to node 2.

E. THE PROTOCOL DESCRIPTION

To put things together, the distributed survivable routing algorithm consists of 3 parts: (1) Path pre-calculation.

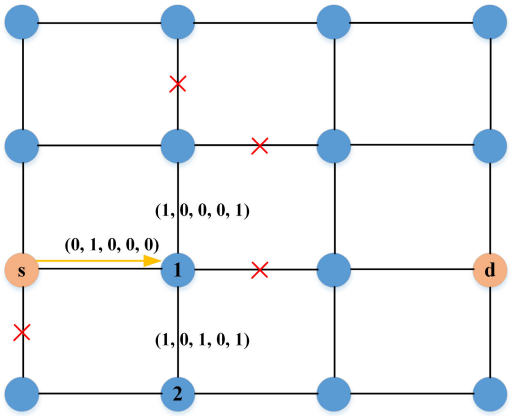

FIGURE 10. Example of vector-based next hop selection.

Each satellite determines multiple primary and secondary directions and paths towards each destination with the basic X-Y routing. (2) Link state update. In case of link failure or recovery, the link state is flooded in a restricted area and the notified satellites update the state of affected paths. (3) Packet forwarding. When a satellite generates or receives a packet, the next hop is selected with the vector-based next hop selection mechanism and the packet header is updated for loop avoidance. These three parts are described in Algorithm 1 to 3 respectively.

F. COMPLEXITY OF THE ALGORITHM

We discuss the complexity of the proposed algorithm from three aspects. First, in the basic X-Y routing, each satellite needs to determine at most four paths towards every destination. For each destination, the path calculation is done based on the orbit number and satellite number of both satellites, which needs $O(1)$ time complexity. Therefore, the time complexity of the basic X-Y routing for each satellite is $O(MN)$. As a comparison, the routing calculation for each satellite in DSP algorithm using a linked list takes time complexity of $O(M^2N^2)$ [23]. The second part is the updating of the path state. When a satellite receives a link state packet, it checks whether the indicated link is on any path for each destination. This process takes $O(MN)$ time complexity, whereas a complete routing table update in DSP algorithm takes $O(M^2N^2)$ time complexity. The third part is the next-hop selection. When a satellite needs to forward a packet, it computes the assessment vector for at most four directions. The time complexity is $O(1)$. To conclude, the proposed routing algorithm achieves low computation overhead, which is desirable for mega-constellations.

V. PERFORMANCE EVALUATION

A. SIMULATION SETUP

In this section, the performance of the proposed distributed survivable routing algorithm is evaluated through simulation. The main performance metrics include end-to-end delay and signaling overhead. Besides, the influence of the algorithms on throughput and packet loss is also investigated. Two routing algorithms are selected as the comparison terms:

(1) Link state algorithm. Each time a link state change happens (i.e., a link fails or recovers), the satellites connected to the link generate a link state packet and flood it to the whole network. When a satellite receives the link state packet, DSP algorithm is used to update the routing table. This algorithm has the global view and is able to obtain optimal routing in case of link failures. However, the signaling and computation overhead is large for flooding the link state packet and conducting DSP algorithm. In our implementation of this algorithm, once a satellite receives a link state packet, it waits for 2 s for more link state packets to arrive so that one update of the routing table takes into account multiple link state changes.

(2) Local distributed routing algorithm. In this algorithm, each satellite only has a local view of its adjacent links. The method for determination of primary and secondary paths is the same as that in the proposed routing algorithm. A node detours a packet only when it actually encounters a failed link. No signaling packet is needed to flood the link state.

Algorithm 1 Path Pre-Calculation

```
Suppose current satellite is s
for each destination d do
    if orb(s) == orb(d) then
        Compute primary direction D_p and primary path P according to sat(s) and sat(d)
    else
        Compute path length L_l of the X-Y path P_l in which X component is left
        Compute path length L_r of the X-Y path P_r in which X component is right
        if L_l < L_r then
            P = P_l
        else
            P = P_r
        end if
        if Y component of P is zero then
            Get one primary path P, and one primary direction D_p
        else
            Get two primary paths P_1 and P_2, and two primary directions D_p1 and D_p2
        end if
    end if
    if there is one primary path from s to d then
        The directions orthogonal to D_p are set as secondary directions D_s1 and D_s2
    else
        The two directions besides D_p1 and D_p2 are set as secondary directions D_s1 and D_s2
    end if
end for
```

Algorithm 2 Link State Update

Suppose the link between s_1 and s_2 is broken or recovered
s_1 and s_2 each construct a link state packet *pkt*. Set the TTL field to f_{ttl}. Send *pkt* to all available neighbors
A satellite s receives *pkt*. Get the TTL field, f, and the broken or recovered link l
for each destination d from s **do**
  for each primary and secondary path P to d from s **do**
    if l is on P **then**
      if $link\_state == failure$ **then**
        $N_{block}++$
      else
        $N_{block}--$
      end if
    end if
  end for
end for
$f--$
if $f > 0$ **then**
  s sets TTL field of *pkt* to f and sends *pkt* to all available neighbors
end if

Algorithm 3 Packet Forwarding

Suppose satellite s receives a packet *pkt* destined to d
if s is not d **then**
  Get value of *nNodeTraversed* field, n, from packet header
  for each available direction D **do**
    Compute $assess\_vec(D)$
  end for
  $D_m = \arg\min_D(access\_vec(D))$
  if n is not zero or D_m is not a primary direction **then**
    Set *nNodeTraversed* field to $n+1$, add id(s) to *nodeList* field
  end if
  Forward *pkt* to D_m
end if

This algorithm is equivalent to setting f_{ttl} to 0 in the proposed routing algorithm.

The simulation is conducted on the first phase constellation of Starlink, which is a representative mega-constellation with inclined orbits. The constellation parameters are listed in Table 1. The phase factor is set in accordance with [13]. Note that although SpaceX has updated the parameters of the constellation in later Federal Communications Committee (FCC) filings, the parameters used in this paper do not loss generality. The ISL rate is set to 100 Mbps, which is a conservative value and achievable with current laser communication terminals. The user data link (UDL) rate is also set to 100 Mbps. The queue size of both ISL and UDL is set to 100 packets. The data packets generated by ground terminals and satellites have a size of 1000 bytes.

TABLE 1. Parameters of the simulated constellation network.

Constellation parameters	
Number of orbit planes	32
Number of satellites per orbit plane	50
Altitude (km)	1150
Inclination	53°
Phase factor	5
Link parameters	
ISL rate (Mbps)	100
UDL rate (Mbps)	100
Queue size (packets)	100

B. SIMULATION RESULTS

1) OVERALL PERFORMANCE

In this experiment, the data packets are generated on satellites and the destination of each packet is selected randomly. The packet inter-arrival time is set according to an exponential distribution with an average of 1 packet per second. This low sending rate is used because our main concern is to investigate the ability of the algorithms to deal with link failure, i.e., to find short end-to-end paths. The simulation time is set to 2 hours to cover as many source-destination pairs as possible.

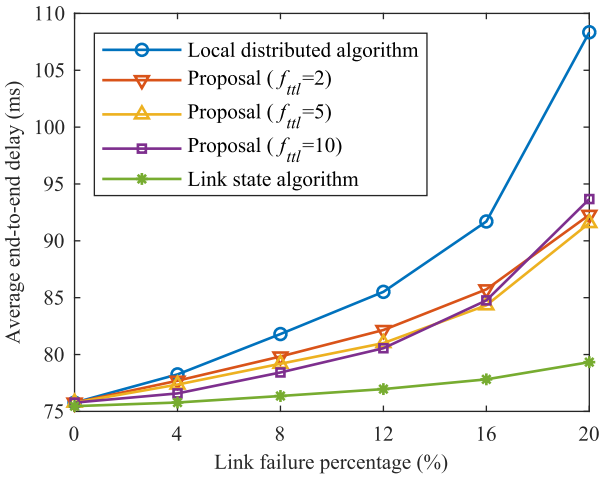

FIGURE 11. Average end-to-end delay at different link failure percentages.

Fig. 11 shows the average end-to-end delay of the packets in different routing algorithms under various link failure percentages. For each scenario with link failure, the graph constructed by the satellite nodes and ISLs is ensured to be connected. For the proposed routing algorithm, the performance under different values of f_{ttl} is evaluated. It can be seen that the proposed algorithm outperforms local distributed algorithm. With $f_{ttl} = 5$, the end-to-end delay is reduced by 15.5% compared with local distributed algorithm under 20% link failure. This is mainly due to the pre-detour mechanism used in our algorithm. When a link breaks, its state is flooded to satellites within several hops away and the notified satellites can detour the packets to alternative paths in advance before encountering the broken link. This reduces the extra delay caused by local detouring. When f_{ttl} increases from 2 to 5, the end-to-end is slightly improved. However, when f_{ttl} increases to 10, the end-to-end delay becomes larger

when the link failure percentage is large. This is because when the network is severely damaged, all of the primary and secondary paths may contain broken links. A larger f_{ttl} may not be able to provide better guidance to the packet than a smaller one and may even cause extra detours. Link state algorithm achieves the smallest end-to-end delay because each satellite has a global view of the network and routes packets along shortest paths.

The signaling overhead is measured by the number of link state packets flooded in the network when there is any link failure. Fig. 12 shows the total number of link state packets flooded in different routing algorithms under various link failure percentages. Local distributed algorithm does not generate link state packets. It can be seen from Fig. 12 that the proposed algorithm floods much fewer link state packets than link state algorithm. For link failure percentage of 20%, the number of link state packets sent in the proposed algorithm is about 1/10 that in link state algorithm with $f_{ttl} = 10$ and about 1/50 with $f_{ttl} = 5$. In the proposed algorithm, the flooding of link state packets is well restricted to a limited area, which greatly reduces signaling overhead in a large-scale constellation.

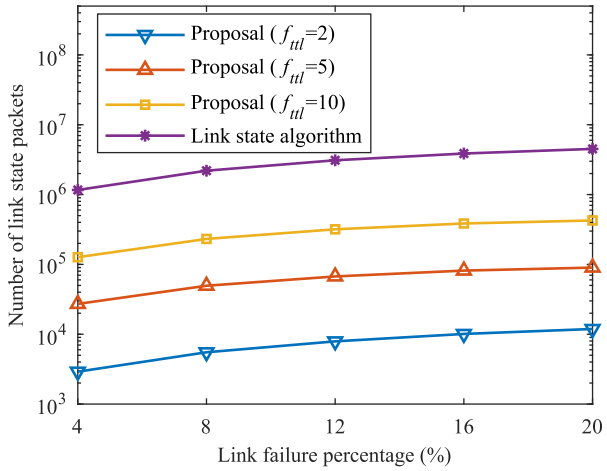

FIGURE 12. Signaling overhead at different link failure percentages.

2) PERFORMANCE OF END-TO-END SESSION

In this experiment, two ground terminals communicate through the satellite constellation network. Terminal A is located at 66.1°W, 10.1°S and terminal B is located at 71.8°E, 14.9°N. The access satellite of each ground terminal remains the same during the simulation time of 2 minutes. Terminal A sends packets to terminal B with a rate of 100 packets per second. Within the rectangular region formed by the access satellites of A and B, random link failure and recovery are generated dynamically. Both the link failure interval and link failure duration follow an exponential distribution with an average of 5 s.

The end-to-end delay versus simulation time is shown in Fig. 13 and the total number of link state packets sent during the simulation is shown in Table 2. The smooth change in end-to-end delay is due to satellite movement and the abrupt change is due to packet detouring in case of link failure and recovery. It can be seen that the proposed algorithm achieves similar end-to-end delay with the link state algorithm most of the time. However, it achieves this with much less signaling overhead. The local distributed algorithm has larger end-to-end delay at the beginning of the simulation because of its inefficient local detouring mechanism.

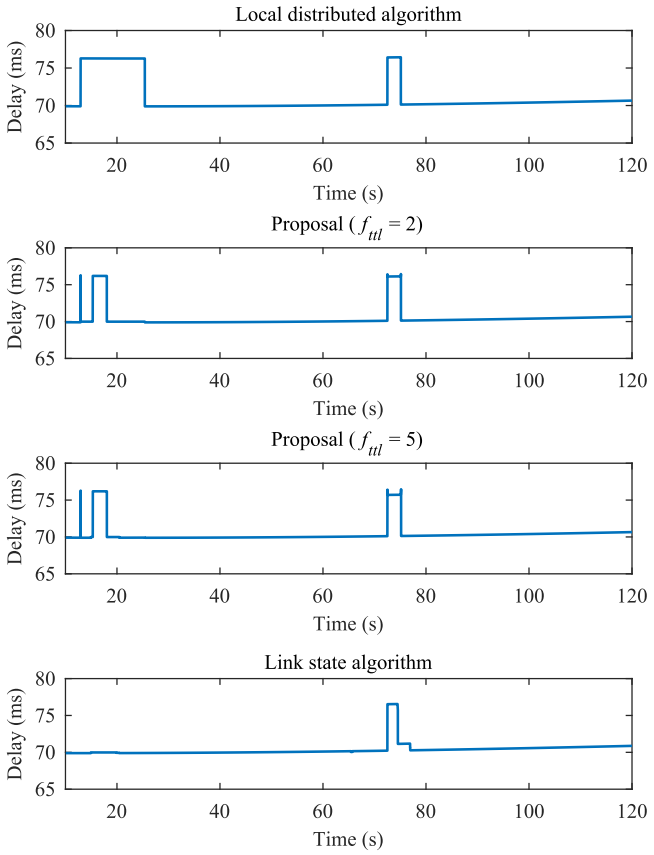

FIGURE 13. End-to-end delay between ground terminals.

TABLE 2. Signaling overhead during the simulation.

Algorithm	Number of link state packets
Local distributed algorithm	0
Proposal ($f_{ttl} = 2$)	4425
Proposal ($f_{ttl} = 5$)	43034
Link state algorithm	1953597

3) EFFECTIVENESS OF SECONDARY PATHS

In the proposed routing algorithm, besides the primary paths, each satellite maintains two secondary paths towards each destination. When a satellite receives a link state packet, the value of N_{block} for each path is updated. When a satellite selects the next hop for a packet, it takes the state of both primary and secondary paths into consideration. In this experiment, we investigate the impact of the secondary paths. The proposed algorithm is modified such that the initial value of N_{block} for each secondary path is set to a value larger than zero and it will not be updated when a link state packet is received.

By such modification, we only maintain the state of primary paths and the secondary directions are selected only then the primary directions are unavailable. We call the modified algorithm proposal-2. In Fig. 14, the average end-to-end delay of the two algorithms is shown. The traffic model is the same as in the first experiment. When the link failure percentage is low, the delay performance of the two algorithms is similar. As link failure increases, the proposal achieves smaller delay because the satellite takes the state of secondary paths into consideration when determining the next hop and therefore has more chance to forward the packet along a path with fewer detours. This experiment demonstrates the effectiveness of the maintenance of the state of secondary paths.

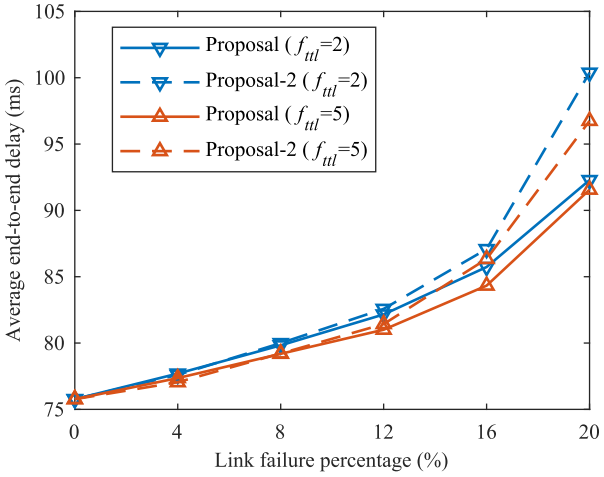

FIGURE 14. Average end-to-end delay with and without maintenance of the state of secondary paths.

4) EFFECTIVENESS OF PARTIAL-RECORD LOOP AVOIDANCE MECHANISM

The partial-record loop avoidance mechanism proposed in our routing algorithm aims to reduce the overhead of recording node IDs in the packet header. To illustrate the effectiveness of this mechanism, we compare the performance of the proposed routing algorithm with full-record loop avoidance mechanism and partial-record loop avoidance mechanism. Fig. 15 shows the end-to-end delay of different mechanisms. The traffic model is the same as in the first experiment. It can be seen that the partial-record loop avoidance mechanism achieves similar performance with the full-record loop avoidance mechanism. To evaluate the overhead of the two mechanisms, we plot the average number of additional bytes per packet for recording traversed satellites in the packet header in Fig. 16. f_{ttl} is set to 5 in this experiment. Each satellite ID occupies 2 bytes. It can be seen from Fig. 16 that the partial-record loop avoidance mechanism has much less overhead when there is few link failure. Even for link failure percentage of 20%, the overhead of the partial-record loop avoidance mechanism is about a half that of the full-record loop avoidance mechanism. To conclude, the partial-record loop avoidance mechanism reduces overhead without degrading the network performance.

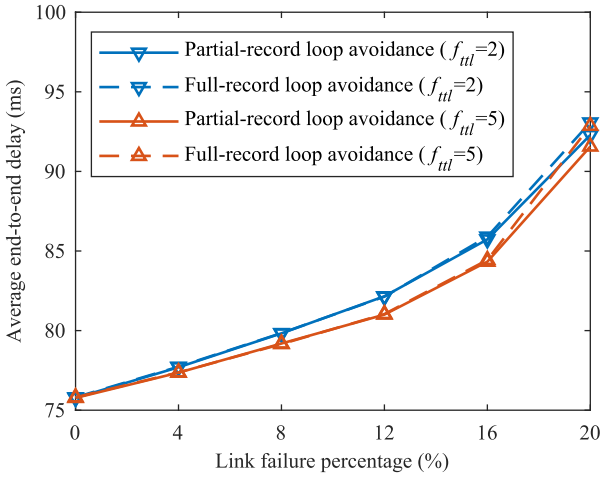

FIGURE 15. Average end-to-end delay of different loop avoidance mechanisms.

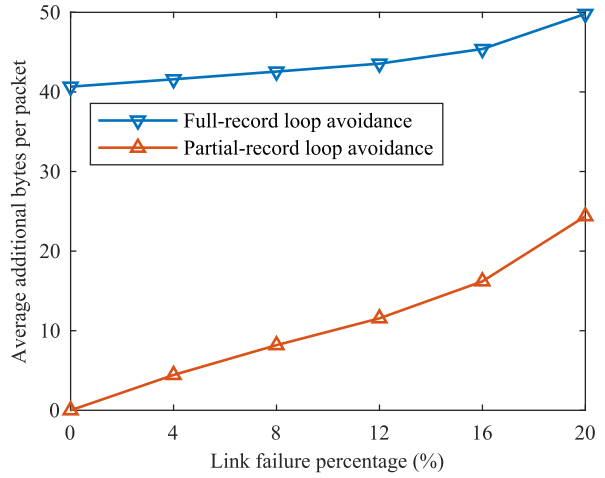

FIGURE 16. Overhead of different loop avoidance mechanisms.

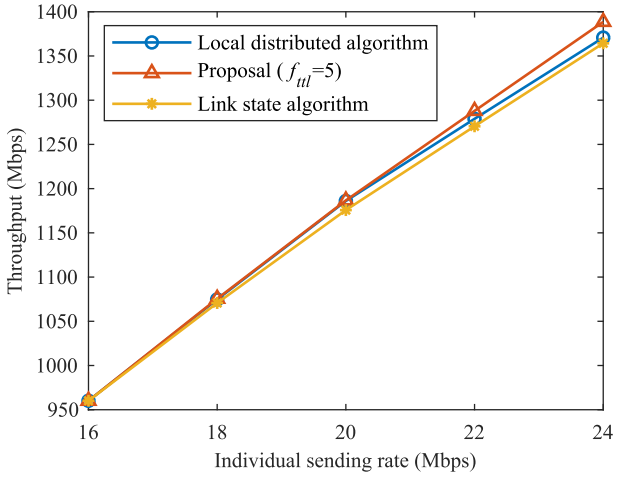

FIGURE 17. Throughput under various sending rates for scenario 1.

5) THROUGHPUT AND PACKET LOSS RATIO

In this experiment, the impact of different routing algorithms on system throughput and packet loss ratio is investigated. The traffic generation model is as follows. Each of

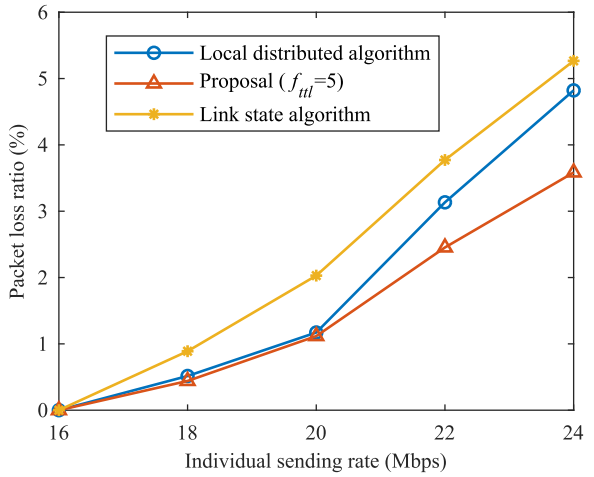

FIGURE 18. Packet loss ratio under various sending rates for scenario 1.

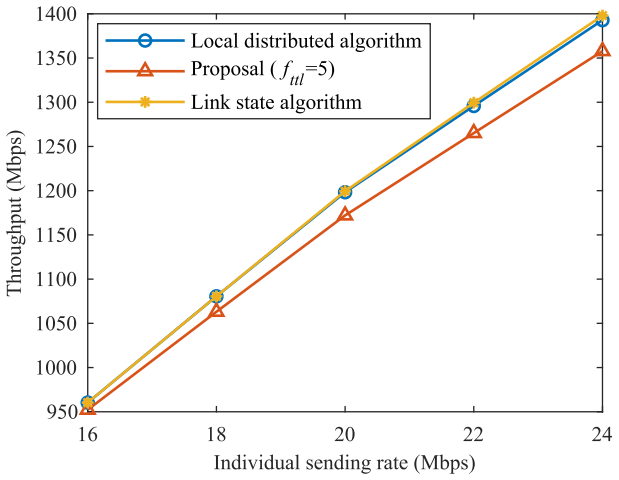

FIGURE 19. Throughput under various sending rates for scenario 2.

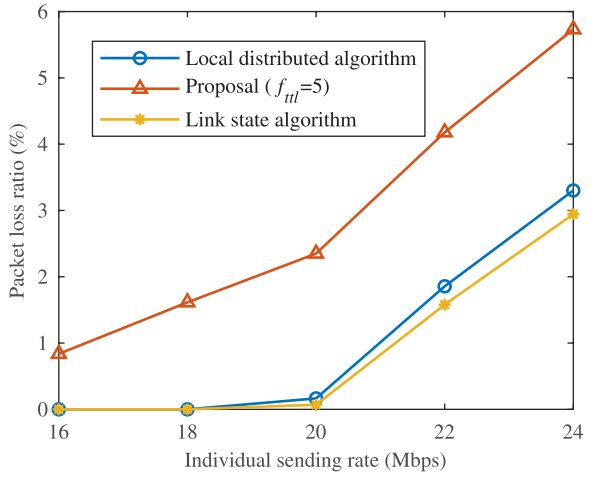

FIGURE 20. Packet loss ratio under various sending rates for scenario 2.

100 ground terminals randomly deployed within 60°S and 60°N on the earth surface sends packets to a randomly selected destination. The packet inter-arrival time is set according to an exponential distribution with varying average values. The simulation duration is set to 30 s. 10% of the ISLs are failed at the beginning of the simulation. Two scenarios with different distribution of the ground terminals and link failures are simulated. The throughput and packet loss ratio under various sending rates for both scenarios are shown in Fig. 17 to Fig. 20. It can be seen that the algorithms perform similarly in terms of throughput. However, there is no algorithm that is superior than other algorithms for both scenarios. Since all of the simulated algorithms here are not designed specifically for load balancing, their performance in terms of throughput depends on the actual traffic condition. The integration of a load balancing mechanism into the algorithms is an interesting topic and deserves further study.

VI. CONCLUSION

In this paper, a distributed survivable routing algorithm is proposed for mega-constellations with inclined orbits. The proposed algorithm includes a basic X-Y routing algorithm, a failure recovery mechanism, a partial-record loop avoidance mechanism and a vector-based next hop selection mechanism. Simulation conducted on Starlink constellation demonstrated that our proposal achieves a good tradeoff between end-to-end delay and signaling overhead in case of link failures. On one hand, with a flooding area of within 5 hops away, the end-to-end delay is reduced by 15.5% compared with local distributed algorithm under 20% link failure. On the other hand, with a flooding area of within 5 hops away, the number of link state packets flooded in the network is only 1/10 that in link state algorithm under 20% link failure. The survivability and scalability of the proposed algorithm are appealing features for mega-constellation networks.

The proposed routing algorithm is designed for inclined constellations. In the future, its applicability to polar constellations and even hybrid constellations which have more intermittent topology will be investigated.

APPENDIX

In this appendix, the endless-loop-free property of the proposed routing algorithm is proved.

Theorem 3: The proposed routing algorithm is endless-loop-free as long as the destination is reachable.

Proof: We proof that two types of endless loop cannot happen as shown in Fig. 21. In the first type, the packet is forwarded back and forth between two adjacent nodes. In the second type, the packet circulates in a multi-hop ring.

Type 1:

Suppose node B receives a packet from node A, as shown in Fig. 21(a). B finds that all of its available directions have been traversed. Thus, B starts to look for the backtracking node. There are 3 cases:

(1) B is the source node and has not been recorded in *nodeList*. In this case, A must have been recorded in *nodeList* because it forwarded the packet to the source, meaning that a detour has happened somewhere along the path. In this case, the backtracking node of B is A. B forwards the packet to A. After receiving the packet from B, A determines the next hop and prefers directions that have not been traversed.

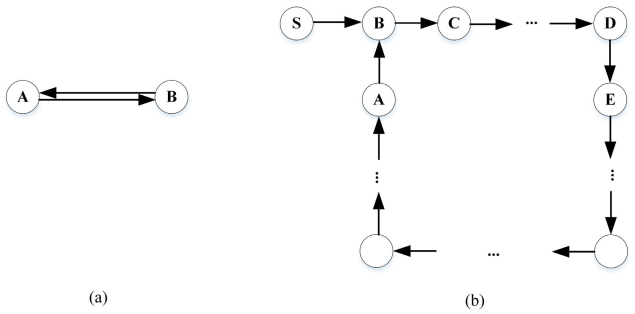

FIGURE 21. Two types of routing loop.

When all of the available directions from A have been traversed, A starts to look for the backtracking node. Note that B cannot appear before A in *nodeList* in this case. We discuss two subcases:

a) The position where A first appears in *nodeList* is not zero. Then A forwards the packet to the backtracking node, which is not B. Thus, an endless loop between A and B cannot be formed. This case is illustrated in Fig. 22(a).

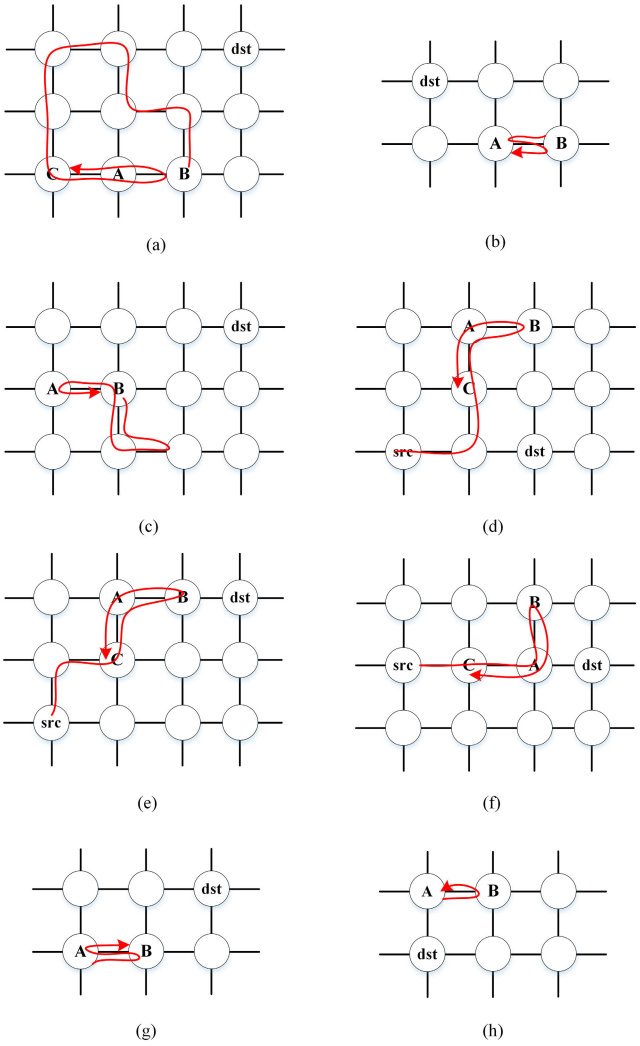

FIGURE 22. Different cases for type 1 loop.

b) The position where A first appears in *nodeList* is zero. This case happens only when the path taken by the packet is B->A->B->A, which implies that the destination is not reachable. This contradicts to the assumption that the destination is reachable. This case is illustrated in Fig. 22(b).

(2) B is the source node and is the first node that is recorded in *nodeList*. This means that the destination is not reachable because we have backtracked to the first node that is recorded and all of the available directions from that node have been tried. This contradicts to the assumption that the destination is reachable. This case is illustrated in Fig. 22(c).

(3) B is not the source node and has not been recorded in *nodeList*. Then, the backtracking node of B is A. B forwards the packet to A. After receiving the packet from B, A determines the next hop and prefers directions that have not been traversed. Note that B cannot appear before A in *nodeList* in this case. We discuss three subcases:

a) A has been recorded in *nodeList* and is not the first node that is recorded. If A finds that all of the available directions have been traversed, then A can forward the packet to the backtracking node, which is not B. Thus, an endless loop between A and B cannot be formed. This case is illustrated in Fig. 22(d).

b) A is the source node and has not been recorded in *nodeList* or is the first node that is recorded in *nodeList*. If A finds that all of the available directions have been traversed, then this means that the path taken by the packet is A->B->A, which implies that the destination is not reachable. This contradicts to the assumption that the destination is reachable. This case is illustrated in Fig. 22(e) and Fig. 22(f).

c) A is not the source node and has not been recorded in *nodeList* or is the first node that is recorded in *nodeList*. In this case, the node from which A first receives the packet is not recorded in *nodeList*. Therefore, A can forward the packet to that node, which is not B. Thus, an endless loop between A and B cannot be formed. This case is illustrated in Fig. 22(g) and Fig. 22(h).

Type 2:

Suppose node B receives a packet from node S, as shown in Fig. 21(b). The packet is then forwarded along the path B->C->. . .->A. We discuss three cases:

(1) Node S has been recorded in *nodeList*. In this case, all the nodes along the path B->C->. . .->A have been recorded. When node A receives the packet and finds that all of its available directions have been traversed, it starts backtracking. Suppose that the packet is backtracked along the reverse path A->. . .->C->B. When node B receives the packet, it sends the packet to the backtracking node S. Thus, an endless loop inside the ring is not formed.

(2) Node B is the first node that is recorded in *nodeList*. When node A receives the packet and finds that all of its available directions have been traversed, it starts backtracking. Suppose that the packet is backtracked along the reverse path A->. . . ->C->B. When node B receives the packet, it can send the packet to node S because S has not been recorded in *nodeList*. Thus, an endless loop inside the ring is not formed.

(3) The first node that is recorded in *nodeList* is some node that appears after B in the path B->C->. . .->A. Suppose it is E. When node A receives the packet, it continues forwarding the packet until the packet again reaches D. From D, the packet starts backtracking. When it again reaches B, B can send the packet to S because S has not been recorded in *nodeList*. Thus, an endless loop inside the ring is not formed.

The theorem is proved. □

REFERENCES

[1] I. del Portillo, B. G. Cameron, and E. F. Crawley, "A technical comparison of three low earth orbit satellite constellation systems to provide global broadband," *Acta Astronautica*, vol. 159, pp. 123–135, Jun. 2019.

[2] Y. Su, Y. Liu, Y. Zhou, J. Yuan, H. Cao, and J. Shi, "Broadband LEO satellite communications: Architectures and key technologies," *IEEE Wireless Commun.*, vol. 26, no. 2, pp. 55–61, Apr. 2019.

[3] Y. Lu, Y. Zhao, F. Sun, and H. Li, "A survivable routing protocol for two-layered LEO/MEO satellite networks," *Wireless Netw.*, vol. 20, no. 5, pp. 871–887, Jul. 2014.

[4] M. Werner, J. Frings, F. Wauquiez, and G. Maral, "Topological design, routing and capacity dimensioning for ISL networks in broadband LEO satellite systems," *Int. J. Satell. Commun.*, vol. 19, no. 6, pp. 499–527, 2001.

[5] L. Wood, "Internetworking with satellite constellations," Ph.D. dissertation, Univ. Surrey, Guildford, U.K., 2001.

[6] T. Pan, T. Huang, X. Li, Y. Chen, W. Xue, and Y. Liu, "OPSPF: Orbit prediction shortest path first routing for resilient LEO satellite networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[7] P. Narvaez, K.-Y. Siu, and H.-Y. Tzeng, "Local restoration algorithm for link-state routing protocols," in *Proc. 8th Int. Conf. Comput. Commun. Netw.*, 1999, pp. 352–357.

[8] E. Ekici, I. F. Akyildiz, and M. D. Bender, "A distributed routing algorithm for datagram traffic in LEO satellite networks," *IEEE/ACM Trans. Netw.*, vol. 9, no. 2, pp. 137–147, Apr. 2001.

[9] Q. Chen, X. Chen, L. Yang, S. Wu, and X. Tao, "A distributed congestion avoidance routing algorithm in mega-constellation network with multi-gateway," *Acta Astronautica*, vol. 162, pp. 376–387, Sep. 2019.

[10] Z. Guo and Z. Yan, "A weighted semi-distributed routing algorithm for LEO satellite networks," *J. Netw. Comput. Appl.*, vol. 58, pp. 1–11, Dec. 2015.

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

[12] Y. Lu, Y. Zhao, F. Sun, H. Li, and D. Wang, "Dynamic fault-tolerant routing based on FSA for LEO satellite networks," *IEEE Trans. Comput.*, vol. 62, no. 10, pp. 1945–1958, Oct. 2013.

[13] M. Handley, "Delay is not an option: Low latency routing in space," in *Proc. 17th ACM Workshop Hot Topics Netw.*, Nov. 2018, pp. 85–91.

[14] D. Bhattacherjee, W. Aqeel, I. N. Bozkurt, A. Aguirre, B. Chandrasekaran, P. B. Godfrey, G. Laughlin, B. Maggs, and A. Singla, "Gearing up for the 21st century space race," in *Proc. 17th ACM Workshop Hot Topics Netw.*, Nov. 2018, pp. 113–119.

[15] D. Bhattacherjee and A. Singla, "Network topology design at 27,000 km/hour," in *Proc. 15th Int. Conf. Emerg. Netw. Exp. Technol.*, Dec. 2019, pp. 341–354.

[16] C. Carrizo, M. Knapek, J. Horwath, D. D. Gonzalez, and P. Cornwell, "Optical inter-satellite link terminals for next generation satellite constellations," *Proc. SPIE*, vol. 11272, Mar. 2020, Art. no. 1127203.

[17] Y. Li, J. Wu, S. Zhao, W. Meng, L. Ma, L. Shi, X. Chu, R. Hou, and T. Li, "A novel two-layered optical satellite network of LEO/MEO with zero phase factor," *Sci. China Inf. Sci.*, vol. 53, no. 6, pp. 1261–1276, Jun. 2010.

[18] R. Suzuki and Y. Yasuda, "Study on ISL network structure in LEO satellite communication systems," *Acta Astronautica*, vol. 61, nos. 7–8, pp. 648–658, Oct. 2007.

[19] B. Gavish and J. Kalvenes, "The impact of intersatellite communication links on LEOS performance," *Telecommun. Syst.*, vol. 8, nos. 2–4, pp. 159–190, 1997.

[20] C.-J. Wang, "Structural properties of a low earth orbit satellite constellation-the walker delta network," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, vol. 3, Oct. 1993, pp. 968–972.

[21] Z. Wu, G. Hu, F. Jin, Y. Fu, J. Luo, and T. Zhang, "Hop-limited adaptive routing in packet-switched non-geostationary satellite networks," *IEICE Trans. Commun.*, vol. E98.B, no. 11, pp. 2359–2368, 2015.

[22] 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.

[23] P. Narváez, K.-Y. Siu, and H.-Y. Tzeng, "New dynamic algorithms for shortest path tree computation," *IEEE/ACM Trans. Netw.*, vol. 8, no. 6, pp. 734–746, Dec. 2000.

XIAOXIN QI was born in 1994. He received the B.S. degree in telecommunication engineering from Xidian University, Xi'an, China, in 2016, where he is currently pursuing the Ph.D. degree in information and telecommunication engineering. His research interests include internetworking and routing in satellite networks.

BING ZHANG (Member, IEEE) was born in 1970. He received the B.S., M.S., and Ph.D. degrees from Xidian University, Xi'an, China, in 1992, 1995, and 2008, respectively. He is currently a Professor with the State Key Laboratory of Integrated Services Networks, Xidian University. His research interests include broadband networks and switching, broadband access networks and home networking, and satellite networks.

ZHILIANG QIU received the B.S. degree in communication engineering and the M.S. and Ph.D. degrees in communication and information systems from Xidian University, Xi'an, China, in 1986, 1989, and 1999, respectively. He is currently a Professor with the State Key Laboratory of Integrated Services Networks (ISN), Xidian University. His research interests include broadband networks and switching technology.