A Dynamic Multicast Tree Selection Algorithm in LEO Satellite Networks

Ke-Jun Zheng* and Jang-Ping Sheu†
*Dept. of Information Systems and Applications, National Tsing Hua University, Hsinchu, Taiwan
†Dept. of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
s108065536@m108.nthu.edu.tw, sheujp@cs.nthu.edu.tw

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10000887

***Abstract*—Satellite networks are a promising way to provide ubiquitous access to network service. However, due to the mobility of satellites, the traditional routing scheme cannot be adapted to the constellation directly. This paper studies the challenge of multicast routing on satellite networks. The mobility of satellites makes the serving satellites of ground users change with time. When performing a multicast on the constellation, the changing of serving satellites leads to many link handover control messages. To minimize the control message overhead is NP-hard. Therefore, a dynamic programming-based algorithm called Dynamic Multicast Tree Selection (DMTS) is proposed to find the sub-optimal result with polynomial time complexity. Besides, we proposed a tree generation algorithm called LMBBSP with DMTS to avoid link congestion in unbalanced network load. The simulation results show that our proposed schemes outperform the baselines with link handover and request rejection rate.**

***Index Terms*—Satellite networking, multicast routing, dynamic programming, link handovers**

I. Introduction

In recent years, network communications have increased dramatically worldwide. One of the critical issues of the traditional terrestrial network is the accessibility in rural and remote areas, such as desert and mountain areas [1]. The satellite constellation networks have been proposed to eliminate this issue. A satellite network contains several Low Earth Orbit (LEO) satellites, and the satellites connect each other to construct a mesh-like topology [2]. To access the constellation, users on the ground must install an antenna to get the service from the sky. Software-Defined Network (SDN) is used to control the operations of the satellite constellation network. The SDN controller is deployed at geostationary satellites (GEO) [3]. Note that in practice, a GEO cannot cover the whole earth. There must be some cooperation between GEOs to reach the full function of the SDN controller. In this paper, we view the GEO cluster as one centralized controller.

The satellite constellation is a promising solution for ubiquitous network accessibility beyond 5G or even 6G mobile networks [4]. Many companies have raised several projects to build their constellations, for example, Starlink from SpaceX [5], Kuiper from Amazon, and OneWeb. One of the promising applications in satellite networks is the scenario of multicast video streaming. A multicast request contains a source host and several destination hosts with their bandwidth requirements and the duration of the video streaming. However, the mobility of satellites makes the traditional multicast tree cannot be directly used in the satellite constellation.

Two main challenges exist for the conventional multicast tree to the satellite networks. 1) *Dynamic changing serving satellites of the source and destinations*. Since the satellites keep moving in a constellation, both source and destinations on the ground keep changing their serving satellites. Therefore, the multicast tree must be recomputed when satellites handover. The multicast tree algorithms for fixed topology cannot be used at the dynamic one. 2) *A large number of control messages*. The satellite mobility forces the controller to recompute the multicast tree in a fixed time interval. When the controller wants to deploy a new multicast tree to deal with the mobility, it will send the link update control messages to the network. The longer a request is, the more tree updates would happen. Thus, reducing the SDN link updates is essential for multicast streaming in a constellation network.

The multicast in satellite networks has been studied in some literature [6]–[9]. [6] proposes a distributed method for robust multicasting based on the unicast in [10]. This method allows each node to make its own decision to reroute the path fast when a link failure happens, making the network more robust. [7] proposes a DLMRA algorithm to exploit the concept of the topology snapshot to simplify the mobility of the LEOs. [8] used the geographical information of neighbor satellites to perform a multicast tree. The proposed method aims to send messages to the geographical center of group members who have not received the packets to reduce the control messages. This multicast strategy has low signaling, computation, and memory usage. [9] proposed a multi-layer rectilinear Steiner tree with a multi-layer LEO constellation. They extend the spanning graph into a 3D scenario to generate a 3D Steiner tree. The multicast tree is generated by minimizing the tree's total length, saving more bandwidth than traditional multicast trees. However, none of them consider the control message overhead when switching between different multicast trees caused by the *dynamic changing serving satellites of the source and destinations*.

In this paper, we formulate a minimization problem to reduce the number of link handovers for multicast in satellite

This work was supported in part by the Ministry of Science and Technology, Taiwan, under grant MOST 109-2221-E-007-079-MY3 and Qualcomm Technologies, Inc. under grant SOW NAT-435533.

networks. Since the minimization problem is NP-hard, a dynamic programming-based tree selection algorithm called DMTS is proposed to find the sub-optimal result in polynomial time. The main contributions of this paper are summarized as follows:

- First, to our best knowledge, this is the first paper to consider the multicast tree with link handovers minimization into the satellites constellation system.
- Second, we propose a dynamic programming algorithm, DMTS, to solve this problem in polynomial time. In addition, we proposed a multicast algorithm LMBBSP with DMTS to avoid link congestion in unbalanced network load.
- Finally, simulation results show that the performance of our algorithm is better than the baselines.

The rest of this paper is organized as follows. Section II presents our multicast network system and the main objective of this paper. Our algorithm is proposed in Section III. The simulation result is shown in Section IV. Finally, Section V concludes this paper.

II. System Model and Problem Formulation

A. System Model

The Satellite Multicast System (SMS) consists of N LEO satellites and one GEO as SDN controller to deal with the incoming multicast requests from ground stations. The GEO is responsible for monitoring the satellites' allocation and serving for multicast tree computation when receiving multicast requests. The satellites are responsible for actual data forwarding for the multicast transmission. And the sources on the ground raise multicast requests to the constellation.

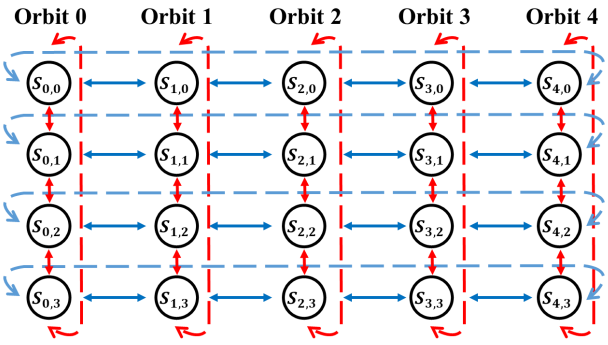

Fig. 1. A 5×4 connected mesh network

In our system model, the +grid model [2], is used to construct the constellation network. We deploy the constellation of N satellites into an $O \times L$ mesh, where O is the number of orbits and L is the number of LEOs within an orbit. An example of a +grid network model with $O = 5$ and $L = 4$ is shown in Fig. 1. In a +grid model, a satellite has four satellite links. Two are connected to the successor and predecessor satellites in the same orbit. Such type of link is called an intra-orbit link (vertical links in Fig. 1). And the other two links connect to the satellites at the left and right orbits, called inter-orbit links (horizontal links in Fig. 1). By adopting this method, the network becomes a mesh topology. The neighbors of each node are fixed. The +grid model makes the regular connections between satellites, which allows us to implement the system quickly without consideration of dynamic neighbor satellites. In practice, the +grid model provides lower latency variance and better path stability [2].

Splitting the whole orbit period into time slices is widely used in satellite network research [11]. Here, we divide the orbit period into several time slices. The orbit period is when a satellite goes around the earth to the same position as the beginning. In each time slice, the positions of all LEOs are viewed as constant. Therefore, the relative positions of the satellites and their serving users are the same within a time slice. Hence, we can use the same multicast tree for a request during each time slice. By splitting the orbit period into time slices, we only need to handle the mobility (i.e., the relation between LEOs and ground users) when the system goes into the next time slice. For example, assume a request spanning over three-time slices, so the request duration is set to 3, which means we need three different multicast trees with two tree transitions to serve the multicast request. The link handover control messages are the cost of the tree transitions. We aim to find the optimal tree transition sequence to reduce the control overhead.

B. Problem Formulation

Let $M = (G_0, \{G_1, G_2, ..., G_d\}, k, bw)$ be a multicast request consisting of a ground node G_0 as the source, d ground nodes $G_1, G_2, ..., G_d$ as destinations, a number of time slices k, and the bandwidth requirement of the multicast streaming bw. After receiving a multicast request, the controller would plan a sequence of multicast trees $\hat{T}$ for the multicast transitions. A sequence of the multicast trees is denoted as $\hat{T} = [T_1, T_2, ..., T_k]$, where k is the number of time slices this request would continue (i.e., how many trees we need to serve this request). After a sequence of multicast trees $\hat{T}$ is established, the SDN controller sends link set-up messages to LEO satellites on the first tree of the $\hat{T}$ to connect all ground source and destination nodes to T_1. Also, the SDN controller reserves the link bandwidth for the future trees in $[T_2, T_3, ..., T_k]$ to ensure enough link bandwidth in the k time slices for the multicast request.

Let S^t and D^t denote the satellite serving the source G_0 and the set of satellites serving the destinations $\{G_1, G_2, ..., G_d\}$ in each time slice t, respectively. After that, the controller computes a multicast tree sequence $\hat{T}$ with length k for serving a request with k time slices. The shape of the t-th tree T_t in the sequence $\hat{T}$ is represented as a link usage matrix X^t:

$$X^t = \begin{bmatrix} x^t_{11} & x^t_{12} & x^t_{13} & \cdots & x^t_{1N} \\ x^t_{21} & x^t_{22} & x^t_{23} & \cdots & x^t_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x^t_{N1} & x^t_{N2} & x^t_{N3} & \cdots & x^t_{NN} \end{bmatrix}, \tag{1}$$

where

$$x^t_{ij} = \begin{cases} 1, & \text{if link } (i,j) \text{ is used at time slice } t; \\ 0, & \text{if link } (i,j) \text{ is not used at time slice } t. \end{cases} \tag{2}$$

Equation (1) is the tree matrix of the t-th tree in $\hat{T}$, where N is the number of LEO satellites in a network. Equation (2) means each item of the tree matrix X^t is 1 if a link from satellite i to satellite j is used in time t and 0 otherwise. Note that the indices i and j here are the global identification of satellites. The controller would compute k trees to ensure there is always a tree to serve the request in k-time slices. When the time goes from slice $t-1$ to t, the controller sends control messages to the satellites not in use and newly joined at slice t to perform the tree transition. Our goal is to minimize the number of satellite (link) handover control messages. The link handover matrix $Diff^t$ is defined as:

$$Diff^t = \begin{cases} X^t \oplus X^{t-1}, & \text{if } 2 \le t \le k; \\ 0, & \text{if } t = 1; \end{cases} \tag{3}$$

Equation (3) shows how to get the matrix of link handovers. The $\oplus$ operator is the element-wise Exclusive OR operation for two matrices. The result after Exclusive OR is the matrix of links that be released or be added from time slice $t-1$ to t, for $2 \le t \le k$. Thus, the objective of this paper is:

$$\min_{(X)} \sum_{t=1}^{k} |Diff^t|. \tag{4}$$

In equation (4), the $||$ operator sums up the number of ones in the matrix $Diff^t$, which indicates the number of link handovers from time slice $t-1$ to t, and the $\sum$ operator adds up the total link handovers from the first time slice to the last time slice. The equation (4) is an allocation problem. The controller allocates the binary variable x_{ij}^t in equation (1) to form a multicast tree. And the objective is a summation of the Exclusive OR operation, which is not a linear function. The allocation problem is a well-known Integer Non-Linear Programming (INLP) problem [12] that is NP-hard.

The multicast tree expression above in equation (1) is a general multicast tree that leads to a large number of possible trees. Fortunately, the traditional multicast tree algorithms have their QoS considerations, such as the shortest path tree (SPT) [13] or the Maximum Bottleneck Bandwidth Shortest Path (MBBSP) [14] tree. So, we can generate multicast trees based on one of the QoS considerations for all time slices and reduce the number of link handovers. Nevertheless, many possible trees can satisfy the same QoS consideration in each time slice. To solve objective equation (4) at a reasonable cost, we restrict the number of multicast trees in each time slice to be a constant c.

For example, we can use the shortest path tree algorithm to generate c different trees for each time slice. And store these trees for time slice t in τ_t, where $1 \le t \le k$. In this way, we can limit the possible combinations of different transition sequences. An example of three-time slices with $c = 3$ is shown in Fig. 2. In Fig. 2, nodes in each time slice are the tree candidates. The number on each node is the accumulative transition cost of the node. The numbers on the arrows are the tree transition cost between two-time slices. To find the minimum transition cost, we can compute the cost of all combinations of the possible tree transitions. However, the time complexity is $O(c^k)$, which is exponential in k. To solve this problem in polynomial time, we propose a dynamic programming-based algorithm called Dynamic Multicast Tree Selection (DMTS).

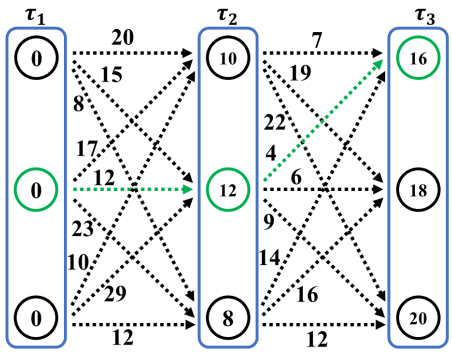

Fig. 2. An example of DMTS with $c = 3$ and $k = 3$

C. Tree Candidates of DMTS

This paper uses two traditional multicast QoS requirements to show the performance of our algorithm. The first one is the Shortest Path Tree (SPT), and the other is Maximum Bottleneck Bandwidth Shortest Path (MBBSP).

1) Shortest Path Tree (SPT): The shortest-path tree means that all paths to serve all the destinations are the shortest. The shortest path of each destination can be easily computed by Breadth-First Search (BFS). The complexity of BFS is $O(N+E)$, where N is the number of nodes and E is the number of links. By the +grid model, the number of E is equal to $4N$. Therefore, the BFS can be solved in $O(N)$. It takes $O(dN)$ to find a multicast tree with the shortest path, where d is the number of destinations.

2) Maximum Bottleneck Bandwidth Shortest Path: The Maximum Bottleneck Bandwidth Shortest Path (MBBSP) [14] aims to prevent network from congestion. The goal of MBBSP is to find out paths with Maximum Bottleneck Bandwidth (MBB) to prevent link congestion. MBBSP avoids routing to the heavy loading satellites when the network load is unbalanced. MBBSP computes MBB for all nodes with a modified Dijkstra algorithm, where we modify the cost value of the shortest distance to the value of MBB. Then, the MBBSP uses BFS to find the shortest path that meet the MBB requirement for each destination to form a multicast tree. It takes $O(NlogN + dN)$ to find a multicast tree candidate.

III. Dynamic Multicast Tree Selection (DMTS)

In this section, we propose a dynamic multicast tree selection (DMTS) algorithm to reduce the link handovers with polynomial time complexity. In addition, a modified version of MBBSP, called LMBBSP, is proposed to improve the reduction rate of link handovers in unbalanced network load. To solve the proposed tree transition selection problem in a polynomial time, we use the technique of dynamic programming to find out the optimal tree transition sequence. The DMTS computes the accumulative transition cost of each tree

in each time slice recursively. After that, DMTS chooses the tree in the last time slice with the minimum accumulative cost to be the last tree transition and then traverses backwardly to figure out the whole transition sequence.

We elaborate the dynamic programming with a top-down explanation. Let A_i^t be the accumulative optimal cost of multicast tree i in time slice t. First, we introduce the recursive recurrence of computing the A_i^t, where $1 \leq i \leq c$ and $1 \leq t \leq k$. The recursion of A_i^t is derived by:

$$A_i^t = \begin{cases} \min_{1 \leq j \leq c}(A_j^{t-1} + C_{j,i}^t), & \text{if } 2 \leq t \leq k \\ 0, & \text{if } t = 1, \end{cases} \tag{5}$$

where $C_{j,i}^t$ is the transition cost from the j-th tree in time slice $t-1$ to the i-th tree in time slice t. The recursion of equation (5) explains the optimal substructure of this dynamic programming. The case when $2 \leq t \leq k$ shows the optimal cost of time slice t is based on the result of optimal costs of time slice $t-1$. The base case of $t = 1$ returns zero because there is no transition in the first time slice. By equation (5), DMTS can get the optimal accumulative cost of each time slice. Second, we can derive the transition cost $C_{j,i}^t$ as follows.

$$C_{j,i}^t = \begin{cases} |X_j^{t-1} \oplus X_i^t|, & \text{if } 2 \leq t \leq k; \\ 0, & \text{if } t = 1. \end{cases} \tag{6}$$

Here, we extend equation (1) to X_i^t by adding index i to represent the i-th candidate's tree matrix in time slice t. Equation (6) uses the Exclusive OR operator $\oplus$ to get the matrix difference between the j-th tree in time slice $t-1$ and the i-th tree in time slice t. The $||$ operator indicates the number of ones in a matrix.

Finally, after all costs and selected tree indices are computed, DMTS selects the tree with the lowest accumulative cost in the last time slice k.

$$R = \begin{cases} argmin_{1 \leq i \leq c}(A_i^k), & \text{if } k > 1; \\ argmin_{1 \leq i \leq c}(|X_i^k|), & \text{if } k = 1. \end{cases} \tag{7}$$

Equation (7) returns the index R in the last time slice to indicate the optimal multicast tree selected in time slice k. In equation (7), if k = 1, it means that the request starts and ends in the first time slice. So, DMTS selects the tree with the least total hop counts. The pseudo code of DMTS is given in Algorithm 1 with a bottom-up implementation. The time complexity of DMTS is $O(kc^2N^2)$.

The SPT algorithm aims to find paths with the shortest hops without considering the remaining bandwidth of links, leading to network congestion. While the MBBSP aims to find paths with MBB to prevent link congestion, it leads to longer paths than SPT. We propose a compromising algorithm called Loose Maximum Bottleneck Bandwidth Shortest Path (LMBBSP), which looses the strict limit of MBB to allow more links can be used for path-finding. In many cases, the required bandwidth bw may be much lower than the mbb when the network load is low. The tight limit of MBBSP removes too many valid links when the network load is still light, which causes a lot of unnecessary detours. To reduce the unnecessary detours,

Algorithm 1 Dynamic **M**ulticast **T**ree **S**election (DMTS)

Input: The request duration k, the number of tree candidates in each time slice c, and the tree candidates $[\tau_1, \tau_2, ..., \tau_k]$
Output: The optimal result index R in the candidate list of the last time slice

```
Compute C^t_{j,i} by eq. (6), where 1 ≤ i, j ≤ c and 1 ≤ t ≤ k;
Initialize A^t_i to ∞, where 1 ≤ i ≤ c and 1 ≤ t ≤ k;
for i = 1 to c do
    A^1_i ← 0;
for t = 2 to k do
    for i = 1 to c do
        for j = 1 to c do
            if A^{t-1}_j + C^t_{j,i} < A^t_i then
                A^t_i ← A^t_j + C^t_{j,i};
if k = 1 then
    R = argmin_{1≤i≤c}(|X^k_i|);
if k > 1 then
    R = argmin_{1≤i≤c}(A^k_i);
return R
```

LMBBSP looses the tight limit of MBB. The main difference of the LMBBSP is that LMBBSP allows the BFS to search links above mbb with a tolerance factor $0 \leq \alpha \leq 1$. The new valid link threshold is defined as:

$$threshold = \max(mbb \times (1 - \alpha), bw), \tag{8}$$

where mbb is the MBB of the searching destination node, and bw is the bandwidth requirement of this multicast request. Link with remaining bandwidth $rb \geq threshold$ is a valid link for path-finding. This expression means that the link whose rb is greater or equal to $mbb \times (1 - \alpha)$ is valid. Note that the valid link's rb must be greater or equal to the requirement bw, so the $\max$ operation is needed. The tolerance factor α makes a trade-off between MBBSP and SPT. When α is close to 0, the LMBBSP performs more like MBBSP with more bandwidth consideration. When α is close to 1, the LMBBSP performs more like SPT, which focuses on minimizing the hop counts and has more diversity of path-finding. By using LMBBSP, the DMTS has impressive improvement in the aspect of the number of link handovers while slightly sacrificing the request rejection rate compared to the original MBBSP.

IV. Performance Evaluation

The simulation results of our DMTS with different multicast tree algorithms, including SPT, MBBSP, and LMBBSP, are shown in this section. The simulation is running on Intel Xeon Silver 4110 CPU with Python 3.6.9. All the simulation results are the average of 20 different cases. The number of orbits and satellites per orbit are set to 25 and 25, respectively. The total number of satellites is 625. The bandwidth of each link is 1000Mbps. The number of requests is 200. We randomly choose the size d of the multicast group in $[10, 40]$. The positions of the ground source and d destinations are randomly generated. The duration k is randomly chosen within $[1, 6]$, and the bandwidth requirement bw is randomly selected in $[10, 30]$ Mbps. The simulations would deploy the constellation in the sky and connect them with the +grid model.

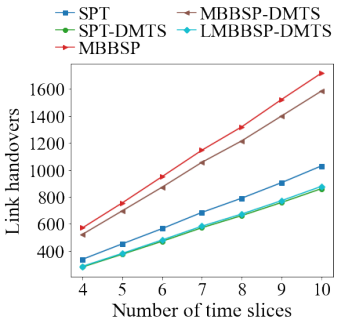

Fig. 3. Number of link handovers vs. request duration

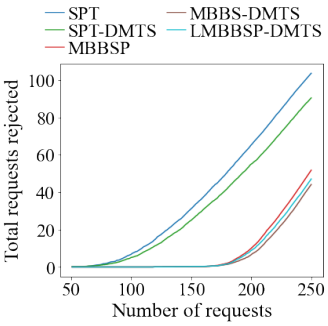

Fig. 4. Number of rejections vs. number of requests

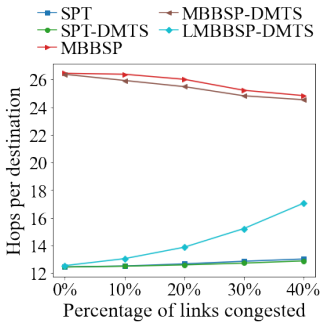

Fig. 5. Average hops vs. percentage of links congested

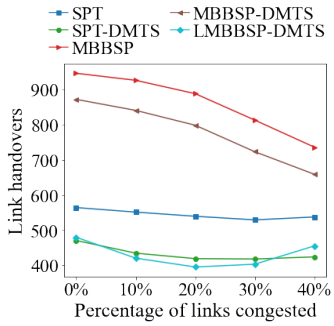

Fig. 6. Number of link handovers vs. percentage of links congested

When the SDN controller receives a multicast request, the geographical information of longitude and latitude of the ground nodes is first converted into 3D coordinates with the formula in [15], and the nearest satellites in each time slice are assigned to be the serving satellites. After converting the nodes' information into satellites, the controller computes the tree sequence according to the algorithm used. The controller allocates the link bandwidth and reserves bandwidth for future time slices when a tree sequence is found. If the controller cannot find a proper tree to serve the request, the request will be rejected. In our simulations, we choose $c = 5$ because it has a better balance between execution time and link handover reduction. Besides, we choose $\alpha = 0.2$ because it provides a sharp reduction in number of link handovers and only sacrifices 1.1% request rejection rate compare to $\alpha = 0.0$ (i.e., MBBSP).

A. Link Handovers and Request Rejections

Here, we show the benefits of DMTS and LMBBSP in aspects of link handovers and request rejection rate. We simulate two different versions of algorithms. The version without DMTS means that the controller only generates one multicast tree for each time slice. Using the DMTS version, the controller generates five multicast trees and chooses the lowest cost transition sequence. Fig. 3 shows the average number of link handovers for a request. We can observe that the SPT and MBBSP with DMTS can reduce link handovers by 16.4% and 7.7% compared to the original SPT and MBBSP, respectively. The LMBBSP-DMTS has a similar performance as the SPT-DMTS. The link handover of LMBBSP-DMTS is 45% lower than the MBBSP-DMTS. The main reason is that the LMBBSP allows the path-finding to relax the original MBB constraint. Therefore, the trees generated by LMBBSP are shorter than MBBSP, which leads to lower transition cost. Thus, DMTS can select a better sequence with LMBBSP than MBBSP.

The request rejection rate in an unbalance network load is shown in Fig. 4. Here, we congest 40% of links with a load of $[75\%, 95\%]$ to simulate the unbalance network load. We can observe that the methods with consideration of the MBB have a significantly lower rejection rate than the SPT ones. The algorithms with DMTS have a lower rejection rate than those without DMTS. The reason is that the DMTS tends to choose trees with lower total hops, so the bandwidth occupation is lower than the original algorithm, leading to a lower rejection rate. In the aspect of rejection rate, the MBBSP-DMTS outperforms all the others since it follows the tightest limit of MBB, and the algorithm with DMTS tends to choose the trees with lower total hops leading to a lower link occupation. MBBSP-DMTS outperforms the simple SPT and SPT-DMTS with 23.7% and 18.4%, respectively. Compared to MBBSP-DMTS, LMBBSP-DMTS just slightly sacrifices 1.2% of the rejection rate, but it can reduce the number of link handovers by 44.6%.

B. Performance with Different Network Load

Fig. 5 shows the average hops of multicast algorithms in congested environments. We randomly select different percentages of links to simulate various network loads. The algorithms with DMTS have slightly lower average hop counts than the algorithms without DMTS such as SPT and MBBSP algorithms. Since dynamic programming minimizes the number of link handovers, the trees with lower total lengths have more selection opportunities. The MBBSP algorithm has larger hops than the SPT one because the MBBSP takes a detour to use links with larger remaining bandwidth. Note that hop counts of MBBSP become shorter when the network load becomes heavy. The reason is that the tight limit of MBB increases the number of invalid links in the light-loaded network even if the links still have enough bandwidth to support multicast routing, so MBBSP takes more unnecessary detours. For the LMBBSP-DMTS, it performs almost the same hop counts as SPT when the network load is light since it avoids the unnecessary detours of the MBBSP.

The number of link handovers with different network loads is shown in Fig. 6. Both SPT and MBBSP have lower links handovers as network load increases. The reason is that when network loading increases, the number of valid links decreases, so there is more opportunity to select the same links. Besides, MBBSP decreases sharper than SPT because it not only avoids the congested links but also avoids the heavy-loaded links. Therefore, MBBSP has more opportunities to choose the same links than SPT. Note that the number of link handovers of the compromising method LMBBSP-DMTS decreases when

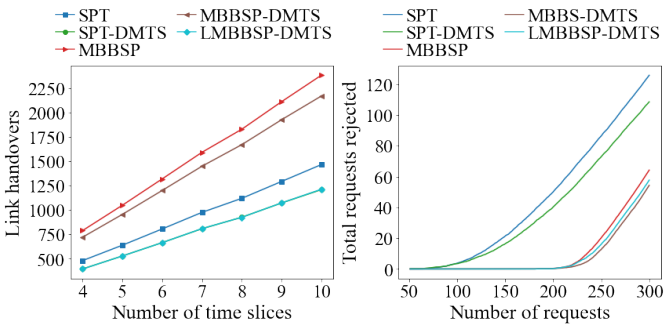

Fig. 7. Number of link handovers vs. request duration with large network

Fig. 8. Number of rejections vs. number of requests with large network

network loading is light. Because when loading is light, LMBBSP-DMTS performs more like SPT to reduce the length of multicast trees. However, when the network load becomes heavy, the property of MBBSP takes the lead on path-finding.

C. Performance with Large Network Size

In this simulation, the number of orbits and satellites per orbit are 32 and 32, respectively. The total number of satellites is 1024. Deploying a denser constellation brings some benefits. For example, the accessibility of ground users increases, and the transmitting power needed to reach the same data rate is reduced [16]. But the number of link handovers also increases. When using a larger topology, the size of trees and the density of satellites increase, so the length of paths increases. Therefore, the total number of link handovers increases. However, the effect of DMTS is still noticeable. Fig. 7 shows the number of link handovers for different schemes. Our DMTS scheme still works well with varying sizes of networks. The DMTS can reduce 17.5% and 8.9% of link handovers for SPT and MBBSP, respectively.

The reduction rate of link handovers slightly increases than the number of satellites is 625. The reason is that the number of nodes used for data forwarding increases for the same source and destination pair, so there are more choices for path-finding. Thus, DMTS can find a better transition sequence. Fig. 8 shows the requests rejection rate. With a larger topology, the number of requests that can be accepted increases, and the LMBBSP-DMTS can still outperform the SPT and SPT-DMTS with 22.7% and 16.9% of rejection rate, respectively. LMBBSP-DMTS can perform close to the MBBSP-DMTS with only 1% gap of rejection rate while reducing the number of link handovers with the impressive ratio of 49% in unbalanced network load. In conclusion, no matter the size of the network topology, DMTS works well to reduce the number of link handovers, and the compromising method LMBBSP-DMTS reduces a large number of link handovers while scarifies a little reduction rate compared to the MBBSP-DMTS.

V. Conclusion

In this paper, we propose a multicast-tree link handover reduction problem. This problem can be solved by a sequence of multicast tree selection in a number of time slices. The proposed DMTS is a general tree selection scheme that can be used with traditional multicast tree generation algorithms. DMTS brings the benefits of shorter hops and a lower number of link handovers. A compromising multicast tree generation algorithm called LMBBSP is proposed to make a trade-off between the shortest path and congestion avoidance in a load unbalanced network. The simulation results show that the algorithms with DMTS have fewer hops and link handovers than the original ones. Besides, the LMBBSP can reduce the number of link handovers compared to the MBBSP while its request rejection rate is close to MBBSP.

References

[1] O. Kodheli *et al.*, "Satellite communications in the new space era: A survey and future challenges," *IEEE Communications Surveys Tutorials*, vol. 23, no. 1, pp. 70–109, 2021.
[2] B. Kempton and A. Riedl, "Network simulator for large low earth orbit satellite networks," in *ICC*, 2021, pp. 1–6.
[3] H. Zhang and C. Wang, "Research on routing control with delay constraint based on contact plan for integrated satellite terrestrial network," in *ICICN*, 2020, pp. 155–159.
[4] X. Zhu and C. Jiang, "Integrated satellite-terrestrial networks toward 6g: Architectures, applications, and challenges," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 437–461, 2022.
[5] T. Bilen *et al.*, "Aeronautical networks for in-flight connectivity: A tutorial of the state-of-the-art and survey of research challenges," *IEEE Access*, vol. 10, pp. 20 053–20 079, 2022.
[6] E. Ekici, I. Akyildiz, and M. Bender, "A multicast routing algorithm for LEO satellite IP networks," *IEEE/ACM Transactions on Networking*, vol. 10, no. 2, pp. 183–192, 2002.
[7] C. Yuan and X. Wang, "A multicast routing algorithm for GEO/LEO satellite IP networks," in *IEEE International Conference on Dependable, Autonomic and Secure Computing*, 2013, pp. 595–599.
[8] Y. Ma *et al.*, "A source-based share-tree like multicast routing in satellite constellation networks," in *FTRA International Conference on Mobile, Ubiquitous, and Intelligent Computing*, 2012, pp. 240–245.
[9] M. Hu *et al.*, "Software defined multicast for large-scale multi-layer LEO satellite networks," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2022.
[10] E. Ekici, I. Akyildiz, and M. Bender, "Datagram routing algorithm for LEO satellite networks," in *Proceedings IEEE INFOCOM 2000*, vol. 2, 2000, pp. 500–508 vol.2.
[11] F. Shen, H. Yu, and X. Zhang, "HATS:a handover optimized routing algorithm for the low earth orbit (LEO) satellite network," in *International Conference on Information, Communications and Signal Processing (ICICS)*, 2009, pp. 1–5.
[12] R. Hemmecke, M. Köppe, J. Lee, and R. Weismantel, "Nonlinear integer programming," *50 Years of Integer Programming 1958-2008*, p. 561–618, Nov 2009.
[13] L. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *Proceedings 22nd International Conference on Distributed Computing Systems Workshops*, 2002, pp. 575–578.
[14] J.-P. Sheu, C.-W. Chang, and Y.-C. Chang, "Efficient multicast algorithms for scalable video coding in software-defined networking," in *PIMRC*, 2015, pp. 2089–2093.
[15] H. Xu *et al.*, "A hybrid routing algorithm in terrestrial-satellite integrated network," in *ICCC*, 2020, pp. 90–95.
[16] O. Popescu, "Power budgets for cubesat radios to support ground communications and inter-satellite links," *IEEE Access*, vol. 5, pp. 12 618–12 625, 2017.