Link Handover-aware Multicast Algorithms for LEO Satellite Networks

Hsuan-Wei Yeh, Jang-Ping Sheu, Nguyen Van Cuong, and Yong Cheng Lin
Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
Emails: aaa9471@gmail.com, {sheujp@cs, cuongnv@gapp, timjackduncan@gapp}.nthu.edu.tw

***Abstract*—Low Earth orbit (LEO) satellite networks have emerged as an efficient solution to offer communications services worldwide, especially to remote areas uncovered by current terrestrial networks. A network with sufficient satellites deployed can connect any two points on the ground. However, routing for multicast requests is more challenging due to the inherent mobility of LEO satellites. This paper aims to design the shortest routes for multicast requests while reducing the number of link handovers. The formulated problem appears to be an NP-hard problem. We first propose a low-complexity heuristic algorithm, the Minimum Handover Tree (MHT) algorithm, to find a suboptimal solution efficiently. We then further design a Multicast Tree Size Aware Handover (MTSAH) algorithm to improve the MHT algorithm in the tree size. The simulation results demonstrate that our proposed solutions outperform several baselines in the existing work.**

***Index Terms*—Satellite networks, multicast routing, LEO handover minimization**

I. Introduction

According to the Cisco white paper [1], there were about 5.3 billion Internet users worldwide in 2023. It means that approximately 2.7 billion people still lack access to the Internet. This is mainly due to the limitations of traditional terrestrial networks in reaching remote and isolated areas, such as deserts and mountainous regions. In response to this challenge, 5G standardization efforts within the 3GPP have proposed using non-terrestrial networks to complement terrestrial networks, with satellite communication being one of the key components [2]. Satellites with the inherent capability of operating without geographical constraints are emerging as an ideal choice for providing broadband connectivity to any global location. Among various satellite technologies, LEO satellites typically have altitudes ranging from 500 to 900 km and have the crucial advantages of lower launch costs and reduced path loss [2]. Motivated by these advantages, various applications of LEO satellites have been proposed in recent works in the literature [3]–[6]. The rise of LEO satellite constellations, such as Starlink, OneWeb, and Iridium-NEXT [7], [8] further strengthens the potential of LEO satellites in providing worldwide satellite-based broadband services.

Constructing an optimal multicast tree is critical to yielding good network performance while providing multicast transmissions. Continuous updates of the multicast tree are required to sustain multicast services but generate a significant number of control messages. Therefore, designing an efficient multicast tree should aim to minimize the number of control messages sent to the nodes. In contrast to traditional networks, where the topology remains static, the mobility of satellites within a satellite constellation introduces a dynamic topology. This makes conventional multicast tree structures unsuitable for satellite constellation-enabled networks.

The multicast-related issues in satellite networks have been studied in several existing works in the literature, e.g., [7]–[12]. For example, [7] proposed a QoS-aware software-defined multicast approach using a weighted rectilinear Steiner minimal trees algorithm to optimize video distribution in LEO satellite networks. In [8], a joint user grouping and resource allocation scheme was presented to maximize the capacity of a LEO satellite multicast system. In [9], a multicast source routing scheme was proposed to determine the routes to multiple LEO satellites covering the same ground station to ensure seamless satellite transfer. In [10], a software-defined multicasting framework with an efficient and scalable multicast routing protocol was proposed for LEO satellite networks. In [11], multicast routing algorithms were proposed for large-scale LEO networks, considering satellite failures and link congestion. It is worth noting that most of the above works focused on a single time frame during which the LEO satellites' coverage remained unchanged. The handover overhead associated with the transitioning between different multicast trees caused by the movement of LEO satellites was not adequately investigated. To address this issue, [12] proposed a dynamic programming-based algorithm to minimize the number of link handovers and alleviate link congestion, particularly in scenarios involving unbalanced network loads.

This paper studies the problem of designing algorithms for routing multicast requests while reducing the number of link handovers in LEO networks, which was also examined in [12]. The main contributions of the paper are summarized as follows. We propose an efficient multicast routing algorithm called the Minimum Handover Tree (MHT). We then design a Multicast Tree Size Aware Handover (MTSAH) algorithm based on the MHT to minimize the multicast tree's size and reduce the rejection rate. We also provide extensive simulation results to demonstrate that our proposed algorithms outperform the solutions in [12] with the number of link handovers.

The rest of this paper is organized as follows. In Section II, we introduce our multicast network system and problem

This work was supported in part by the National Science and Technology Council, Taiwan, under grants NSTC 112-2221-E-007-046-MY3 and NSTC 113-2221-E-007-137.

ICC 2025 - IEEE International Conference on Communications | 979-8-3315-0521-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICC52391.2025.11161779

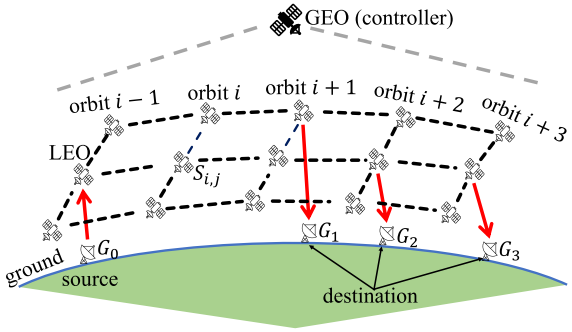


Fig. 1. System model

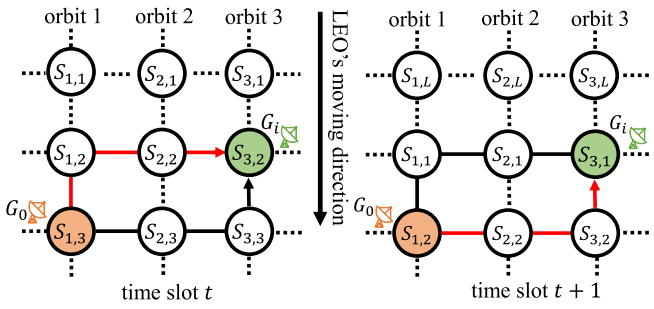


Fig. 2. Illustration of handovers due to LEO satellites' movement

formulation. Section III presents our proposed algorithms. Section IV evaluates the performance of our algorithms with simulation results. We conclude this paper in Section V.

II. System Model and Problem Formulation

We consider an LEO network consisting of N satellites, organized into O orbits, each orbit comprising L satellites, as illustrated in Fig. 1. The network topology is constructed following the widely used +grid model [13], in which each LEO satellite has direct links to four neighbors, including one satellite ahead and one satellite behind in the same orbit, along with the two nearest satellites on the left and right orbits. We assume all LEO satellites are controlled by geostationary earth orbit (GEO) satellites that serve as software-defined networking (SDN) controllers. A multicast request is initiated by a ground station as a source and sent to multiple other ground stations as destinations via the LEO network. Note that each ground node can connect to only one satellite at a time. We assume that the orbit period is divided into L equal time slots, and that the network topology remains static during each time slot. Due to the movement of the LEO constellation, each ground station is covered by different LEO satellites in different time slots. At the beginning of each time slot, the SDN controller (i.e., GEO satellite) sends control messages to nodes (LEO satellites) when the multicast tree changes from the previous time slot. Let us denote satellite j in orbit i as $S_{i,j}$. The shortest distance in terms of the number of nodes between satellite $S_{i,j}$ and satellite $S_{i',j'}$ can be computed as

$$\begin{aligned}\delta(S_{i,j}, S_{i',j'}) = &\min(|i-i'|, O-|i-i'|) \\ &+ \min(|j-j'|, L-|j-j'|),\end{aligned} \tag{1}$$

for all $1 \le i \le O$, $1 \le j \le L$. Since the satellites form a spherical constellation, $S_{i,1}$ links directly to $S_{i,L}$ and $S_{1,j}$ links directly to $S_{O,j}$.

In this work, each multicast request is defined as $M = (G_0, \{G_1, G_2, ..., G_K\}, d)$, where G_0 is the source ground node, $\{G_1, G_2, ..., G_K\}$ is the set of destination ground nodes, and d denotes the duration in units of time slots during which the multicast request will remain active. Upon receiving the multicast request, the SDN controller strategizes a series of multicast trees as $\mathcal{T}^* = [\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_d]$, where $\mathcal{T}_t$ denotes the multicast tree designed for transmission in time slot t, for all $1 \le t \le d$. During each time slot t, the satellite covering the source node G_0 is denoted as $\mathcal{S}^t$, and the satellite covering the destination node G_i is denoted as $\mathcal{D}_i^t$, for all $1 \le i \le K$.

A. Problem Formulation

A simple example of the handover problem in LEO networks is illustrated in Fig. 2. Suppose that G_0 is connected to satellite $S_{1,3}$ (i.e., 3-th satellite on orbit 1), and G_i is connected to the satellite $S_{3,2}$, in time slot t. The solid black path in the left-hand side graph is one of the shortest paths for transmission from the source to the destination. In time slot $t+1$, due to the movement of LEO satellites, G_0 and G_i are covered by satellites $S_{1,2}$ and $S_{3,1}$, respectively. In this case, the shortest transmission path can be the solid black path on the right-hand side of the graph. Since this path differs entirely from the previous one, the controller must send routing information to all nodes. However, if we select the transmission paths in time slots t and $t+1$ as the solid red paths, then only satellites $S_{1,3}$ and $S_{3,1}$ need to be updated routing information. Determining an optimal transmission path is challenging when the number of nodes is large and the duration of the multicast request spreads over more time slots.

Recall that $\mathcal{T}_t$ is the multicast tree in time slots t. If it is different from that of the previous time slot, i.e., $\mathcal{T}_t \neq \mathcal{T}_{t-1}$, the controller must send control messages to update the routing information to all nodes that made up the change in the multicast tree. This work aims to reduce the total number of link handovers (or the number of control messages sent) under the condition of shortening the transmission path during the duration of the multicast request. Thus, the problem can be formulated as

$$\min_{\mathcal{T}_t, \forall t} \sum_{t=2}^{d} |\mathcal{T}_t \oplus \mathcal{T}_{t-1}|. \tag{2}$$

Here, the element-wise XOR operator determines the handovers between two shortest-path multicast trees. The problem in (2) is an NP-hard problem [12]. In the next section, we propose an algorithm with polynomial-time complexity to efficiently find a suboptimal solution.

III. Proposed Multicast Algorithms

In this section, we propose an MHT algorithm to find the multicast trees to route the multicast request, with the aim of reducing the number of link handovers induced by the

movement of LEO satellites. We then introduce an MTSAH algorithm to reduce the size of multicast trees yielded by the MHT algorithm.

A. Minimum Handover Tree (MHT) Algorithm

The primary objective of our proposed MHT algorithm is to construct a sequence of multicast trees with minimum link handovers over the time slots. Choosing optimal multicast trees to route data to multiple destinations in time slots is particularly challenging. Our approach is to find paths with minimum link handovers to each destination at a time. In the following, we refer to these paths as Minimum Handover Paths (MHPs). Once MHPs are found for all destinations, we can merge them to form the multicast trees of the multicast request. Suppose $\mathcal{T}_t^i$ is the MHP expected for transmission from the source node (denoted S^t) to the destination i (denoted D_i^t) in time slot t. If a node x is in $\mathcal{T}_t^i$, the path distance is a summation of S^t to node x and x to D_i^t, which is the shortest path from S^t to D_i^t. To select a node x to reduce link handover within d time slots, we need to consider the path length through node x at $\mathcal{T}_1^i, \mathcal{T}_2^i, \ldots, \mathcal{T}_d^i$.

To do this, we propose the following function $f(x,i,t)$ to evaluate the cost of the transmission path from the source node to the destination i passing through a node x in time slot t

$$f(x,i,t) \triangleq \gamma(x,i,t-1) + \sum_{\tau=t}^{d} 2^{t-\tau}\gamma(x,i,\tau), \qquad (3)$$

for each satellite x, $1 \le i \le K$, and $1 \le t \le d$, where $\gamma(x,i,t) \triangleq \delta(\mathcal{S}^t,x) + \delta(x,\mathcal{D}_i^t), \forall t \ge 1$, and $\gamma(x,i,t) = 0$ for $t = 0$, represents the total distance from S^t to node x and x to D_i^t in the time slot t. The first term in (3) indicates the shortest distance passing through node x in the preceding time slot. This incorporation makes the MHP in the current path not deviate much from the path found in the previous time slot. The second term in (3) computes the total shortest distances via node x over the current and the following time slots. This term evaluates the path length of the future time slots through node x. In particular, our proposed cost function imposes a weighting factor of $2^{t-\tau}$ to ensure that MHP in the time slot t always passes through nodes on the shortest path in the time slot t. Without such a weighting factor, the expected MHP might deviate from the shortest path connecting the source and the destination. This is because the shortest paths in different time slots would have similar contributions to the cost function, but our expectation is the dominance of the current time slot's shortest path in the current time slot's MHP.

Note that, we denote $\mathcal{T}_t$ as the multicast tree in time slot t and it is the union of MHPs, i.e., $\mathcal{T}_t = \bigcup_{i=1}^{K} \mathcal{T}_t^i$. $\mathcal{T}_t^i$ can be found using the following procedure, which includes two steps: *discovering* and *backtracking*. In the *discovering* step, we aim to explore potential nodes to construct the MHP. Note that a node can be in one of three states: *undiscovered*, *discovered*, and *visited*. Initially, the source node is marked as *discovered*, and all the others are marked as *undiscovered*. Starting from the source node, we find all its *undiscovered* neighbors and calculate their costs using equation (3). After that, we mark those neighbors as *discovered* and save their parent information (i.e., source node at this moment). In addition, the parent node is marked as *visited* to avoid it being revisited in the future. We now choose to visit a node with the minimum cost among *discovered* nodes and repeat the above process until the destination node is discovered. In the *backtracking* step, starting from the destination, we search for its parent node, which has been saved before. Repeating the above process will eventually return to the source node. The path created in this step is the MHP $\mathcal{T}_t^i$ of the destination i in the time slot t. Once the MHPs of all destinations are acquired, they are merged to obtain the multicast tree $\mathcal{T}_t$ for the time slot t.

Algorithm 1: Minimum Handover Tree (MHT) Algorithm

Input: a multicast request $M = (G_0, \{G_1, \ldots, G_K\}, d)$
Output: a series of multicast trees $\mathcal{T}^* = [\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_d]$

1 **for** $t = 1$ *to* d **do**
2 $\mathcal{T}_t \leftarrow \emptyset$;
3 **for** $i = 1$ *to* K **do**
4 Mark S^t as *discovered*; $x \leftarrow S^t$; $\mathcal{T}_t^i \leftarrow \emptyset$;
5 **while** $x \neq D_i^t$ **do**
6 $\mathcal{N} \leftarrow \{$*undiscovered* neighbors of $x\}$;
7 Compute cost for all $n \in \mathcal{N}$ using (3);
8 Mark all $n \in \mathcal{N}$ as *discovered*;
9 Save x as parent for all $n \in \mathcal{N}$;
10 Mark x as *visited*;
11 $x \leftarrow$ a random node has minimum cost among *discovered* nodes;
12 **end**
13 $x \leftarrow D_i^t$;
14 **while** $x \neq S^t$ **do**
15 $\mathcal{T}_t^i \leftarrow \mathcal{T}_t^i \cup x$;
16 $x \leftarrow$ parent of x ;
17 **end**
18 $\mathcal{T}_t \leftarrow \mathcal{T}_t \cup \mathcal{T}_t^i$;
19 **end**
20 $\mathcal{T}^*[t] \leftarrow \mathcal{T}_t$;
21 **end**

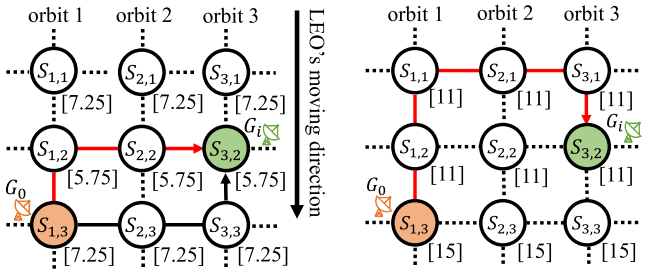
The pseudocode of the MHT algorithm is summarized in Algorithm 1. Given the multicast request, the algorithm returns a series of multicast trees, each for a time slot of the request. For each destination i in time slot t, the MHP is found from lines 4 to 17, in which the *discovering* and *backtracking* steps are performed, respectively, in the first and second **while** loops. In the first step, we define a set $\mathcal{N}$ containing *undiscovered* neighbors of node x (line 6) and compute their costs using equation (3) (line 7). Then, the *undiscovered* neighbors are marked as *discovered* (line 8), and node x is saved as their parent (line 9). We now mark node x as *visited* (line 10) before visiting the next node. The next visiting node is selected as a random node that has the minimum cost among *discovered* nodes (line 11). The chosen node is then assigned as node x, and the process is repeated until it reaches the destination. In the second step, we perform a backtracking process starting from the destination and going through the

parent of the *discovered* node that has been saved before arriving at the source. The MHP $\mathcal{T}_t^i$ is the path connecting the parent nodes considered in the above backtracking process. Regarding computation complexity, the **while** loops perform N iterations in the worst case. Hence, the overall complexity of Algorithm 1 is $\mathcal{O}(dKN)$, where d, K, and N indicate the duration of multicast, the number of destinations for the multicast request, and the number of nodes, respectively.

An example of determining the MHP using the Algorithm 1 is illustrated in Fig. 3(a), in which we consider a multicast request with a duration $d = 3$. The cost of the nodes is calculated using the equation (3) and is noted at the lower right corner of each respective node. For example, considering node $S_{1,3}$, its shortest paths from the source to the destination in time slots $t = 1$, $t = 2$, and $t = 3$ can be $S_{1,3} \to S_{1,2} \to S_{2,2} \to S_{3,2}$, $S_{1,2} \to S_{1,3} \to S_{2,3} \to S_{3,3} \to S_{3,2} \to S_{3,1}$, and $S_{1,1} \to S_{1,2} \to S_{1,3} \to S_{2,3} \to S_{3,3} \to S_{3,2} \to S_{3,1} \to S_{3,L}$, respectively. Thus, its cost can be computed as $f(S_{1,3}, i, t) = (0+3) + (1+4)/2 + (2+5)/4 = 7.25$. Starting from the source node, we select the node with the lowest cost among neighbors while storing the *parent* of the neighbors (e.g., $S_{1,2}$ is the parent of nodes $S_{1,1}$, $S_{2,2}$, and $S_{1,3}$). Repetition of the above procedure can obtain a discovery path as $S_{1,3} \to S_{1,2} \to S_{2,2} \to S_{3,2}$ (i.e., red path). Next, we perform a backtracking process to the source, in which we go through the current node's parent starting from the destination. The path resulting in this step is the discovery path itself and is the MHP of the destination i in the time slot t, that is, $\mathcal{T}_t^i$. Finally, the MHPs of the destinations are merged to establish the multicast tree. It should be noted that without the weighting factors proposed in the evaluation function in (3), the MHP can deviate from the shortest path of the current time slot, as neighbors can have the same cost. An example of this situation is illustrated in Fig. 3(b). In this case, the cost of $S_{1,3}$ is calculated as $f(S_{1,3}, i, t) = (0+3) + (1+4) + (2+5) = 15$ and that of other nodes without imposing weighting factors are indicated in their lower right corners. We can see that multiple nodes share the same lowest cost, and randomly selecting one of them for the next movement may result in a path marked by red color in Fig. 3(b) that deviates from the shortest path (i.e., red path in Fig. 3(a)).

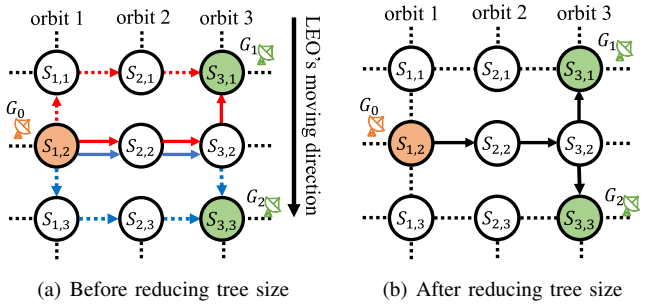
B. Multicast Tree Size Aware Handover (MTSAH) Algorithm

In this subsection, we propose the MTSAH algorithm to improve the performance of Algorithm MHT. The MTSAH algorithm also aims to minimize handovers but also considers the optimality of the multicast tree size. We can observe that by randomly choosing a node among the discovered nodes having the same minimum cost in Algorithm 1 may produce different MHPs in different attempts. In addition, merging MHPs of different destinations to create the multicast tree may result in a large tree since the possibly redundant links are not eliminated. To address this issue, we propose an improved solution as follows. First, we create multiple MHPs for each destination in each time slot by repeating the MHP finding process (i.e., lines 4 to 18 in Algorithm 1) for a given small constant I iterations (e.g., we set $I = 5$ in our simulations). Note that each iteration may produce a different MHP due to the random selection of the next node in the case of multiple discovered nodes having the same minimum cost. An illustration is shown in Fig. 4(a) with two different MHPs (dotted and solid paths) for each destination. We can observe that the multicast tree can be reduced to a smaller tree illustrated by the solid black path in Fig. 4(b). Reducing the size of the multicast tree can be referred to as the directed Steiner tree problem of the graph formed by MHPs, and the objective is to find the shortest possible tree connecting a set of specified nodes (source and destinations). Various algorithms have been proposed in the literature to solve this NP-hard problem. There is always a trade-off between the optimality of the solution and the complexity of time when solving such an NP-hard problem. Here, we adapt the heuristic algorithm proposed in [14]. Specifically, after obtaining feasible MHPs for all destinations, we form a complete graph with varying weights. Subsequently, we iteratively select the required edges until all destination nodes are interconnected. In each iteration, the algorithm aims to build a Steiner tree of the source and a subset of the destinations only. Once the Steiner tree of such a subset of destinations is determined, we remove them from the list of given destinations and repeat the process until all destinations are completed. The algorithm for solving this Steiner tree problem has a time complexity of $O(N^3K^2)$ [14]. Thus, the overall time complexity of the MTSAH Algorithm in the worst case is $\mathcal{O}(dN^3K^2)$.



(a) With weighting factor (b) Without weighting factor

Fig. 3. An illustration of finding minimum handover path (MHP)



(a) Before reducing tree size (b) After reducing tree size

Fig. 4. An illustration of reducing multicast tree size

IV. Performance Evaluation

In this section, we provide the numerical results to verify the efficacy of our proposed algorithms. We set up the LEO

network following the configuration of SpaceX's Starlink with $O = 72$ orbits, and each orbit is deployed with $L = 22$ LEO satellites [15]. The link capacity of inter-satellite connections is set to 1 Gbps [12]. The simulation results average over 20 realizations, each executing 1000 multicast requests in sequence. Unless mentioned otherwise, the number of destinations K for each request is randomly selected in a range of $[10, 40]$, and the duration d is randomly chosen in a range of $[2, 6]$ in units of time slots. The source and destinations of each request are randomly selected from LEO satellites. We further assume that each request demands a bandwidth of w randomly selected between 10 and 30 Mbps. In this case, the request will be rejected if the total bandwidth of active requests exceeds the inter-satellite link capacity. For performance comparison, we examine the Dynamic Multicast Tree Selection (DMTS) algorithm in [12]. The DMTS generates multiple multicast trees for each time slot and employs the dynamic programming technique to find the optimal sequence of tree transitions. It first calculates the cumulative transition cost for each tree in every time slot through a recursive process. Afterward, it selects the tree in the last time slot with the minimum accumulative cost as the final tree transition and then backtracks to determine the complete transition sequence. We implement two baselines based on two multicast tree types in [12], including the Shortest Path Tree (SPT-DMTS) and the Loose Maximum Bottleneck Bandwidth Shortest Path (LMBBSP-DMTS). The SPT implies that the routes to the destinations are the shortest. The LMBBSP is based on its previous version called MBBSP in [16], with the aim of finding a maximum bottleneck bandwidth (MBB) path to avoid link congestion. Specifically, the LMBBSP relaxes the stringent constraints of MBB to allow the utilization of more links for the path-finding process, thereby reducing unnecessary detours.

Fig. 5 shows the average number of handovers per request concerning the duration of the request, which is randomly selected in a range of $[2, d_{\max}]$, where $d_{\max} = 2, 4, \ldots, 10$. We can see that the number of link handovers increases as the duration of requests increases in all schemes. This is expected since different satellites will continually cover the request if it is active over more time slots. The performance of our proposed algorithms outperforms the baselines, verifying the efficiency of the proposed evaluation function and the MHP generation procedure. Notably, our MHT and MTSAH schemes achieve similar performance in the number of handovers since they are designed with the same principle, but the MTSAH further aims to reduce the multicast tree size.

Fig. 6 shows the total number of links (hops) per request on average, corresponding to the test shown in Fig. 5. As the request spans over a longer duration, the number of handovers increases, resulting in a larger number of total links used during the duration of the requests. Although the performances of MHT and MTSAH are similar, as shown in Fig. 5, MTSAH obtains better performance regarding the number of links. This proves that solving the resulting Steiner tree problem in Algorithm MTSAH efficiently reduces tree size.

Fig. 7 examines the number of link handovers for network

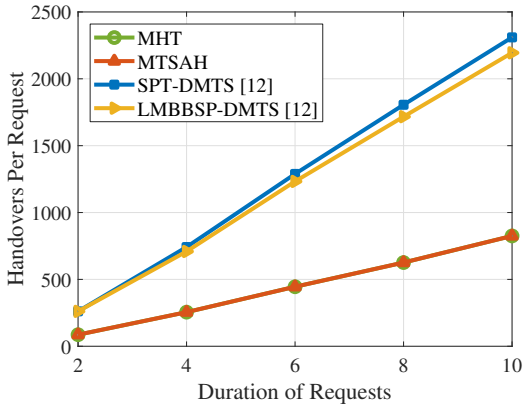


Fig. 5. Link handovers versus request duration d

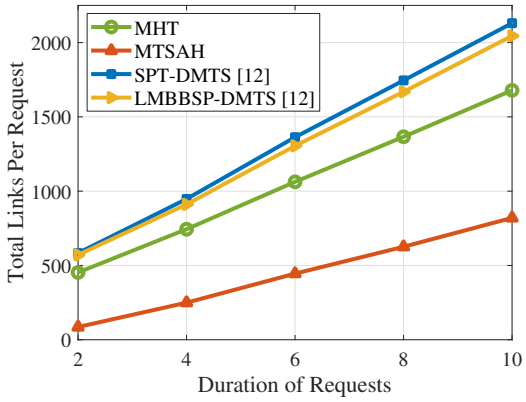


Fig. 6. Total links (hops) versus request duration d

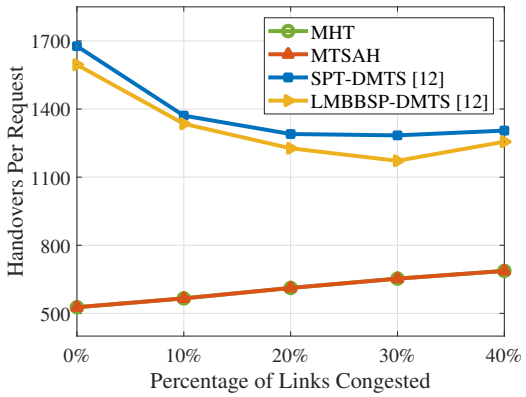


Fig. 7. Link handovers versus link condition

loads. Here, we assume that a percentage of the number of inter-satellite links is congested due to, for example, overloading. Specifically, we assume that the capacity of the congested link will remain from 5% to 25% of the maximum capacity. Both the MHT and MTSAH schemes demonstrate an increase in link handovers as the number of congested links increases. This is because the decline of available paths results in longer MHPs with more handovers. The DMTS schemes can select the same link with their link selection policy if the number of available links decreases. Thus, their average number of link handovers decreases as the percentage of congested links

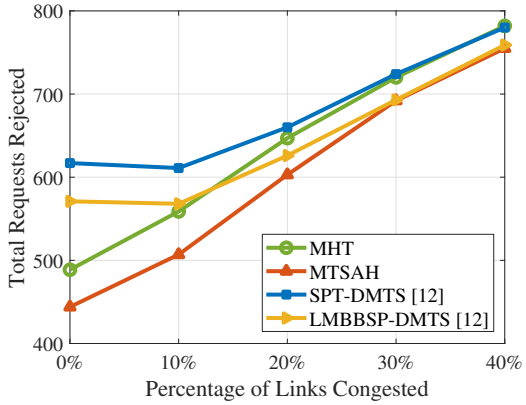


Fig. 8. Rejection rate versus link condition

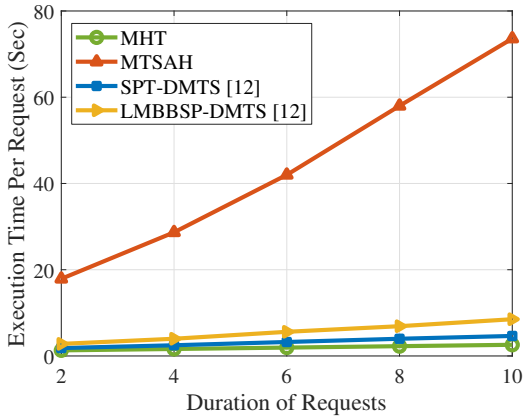


Fig. 9. Execution time versus request duration

increases.

Fig. 8 shows the number of rejected requests under different network loads described in Fig. 7. We can see that as the network load becomes more serious, more requests will be rejected due to insufficiency of the link capacity. In particular, MTSAH surpasses all other schemes in this metric, showing that minimizing the size of the multicast tree in Algorithm MTSAH efficiently reduces the resources required for each request and then allows more requests to be served. The performance difference is moderate if the network load is extremely heavy since the number of feasible paths is significantly reduced in this situation.

Fig. 9 examines the average execution time (in seconds) with respect to the duration of the request. We can see that MHT is faster than the baselines since it only needs to compute a single tree, while the baselines with the DMTS algorithm require several multicast trees to calculate each request. Note that our MTSAH algorithm requires a longer execution time than other schemes. However, as shown in the above figures, it offers a superior benefit in reducing the multicast tree links and mitigating the requests' rejection rate. The MTSAH algorithm is more suitable for small-scale networks, e.g., a network formed by a subset of LEOs covering a specific area.

V. Conclusion

This paper studied the multicast link handover problem in LEO networks. Our proposed algorithms decoupled the multicast tree problem into multiple unicast transmission issues. An efficient evaluation function was designed to assess the optimality of the nodes constructing transmission paths. The unicast transmission paths were then merged to obtain the multicast tree. We also incorporated the Steiner tree problem algorithm to optimize the multicast trees further. The simulation results showed that our MHT significantly mitigated the link handovers compared to several algorithms in the previous work while acquiring a low time complexity. In particular, the MTSAH algorithm efficiently reduced the size of the multicast trees to reduce the rejection rate in unbalanced-load networks.

References

[1] Cisco, "Cisco annual internet report (2018–2023) white paper," Mar 2020. [Online]. Available: https://www.cisco.com/c/en/us/index.html

[2] O. Kodheli *et al.*, "Satellite communications in the new space era: A survey and future challenges," *IEEE Commun. Survey Tuts.*, vol. 23, no. 1, pp. 70–109, 2021.

[3] Y. Lyu, Z. Liu, R. Fan, C. Zhan, H. Hu, and J. An, "Optimal computation offloading in collaborative LEO-IoT Enabled MEC: A multiagent deep reinforcement learning approach," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 2, pp. 996–1011, 2023.

[4] D. Jiang *et al.*, "QoE-aware efficient content distribution scheme for satellite-terrestrial networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 443–458, 2023.

[5] X. Cao *et al.*, "Edge-assisted multi-layer offloading optimization of LEO satellite-terrestrial integrated networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 2, pp. 381–398, 2023.

[6] J.-H. Lee, J. Park, M. Bennis, and Y.-C. Ko, "Integrating LEO satellites and multi-UAV reinforcement learning for hybrid FSO/RF non-terrestrial networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 3, pp. 3647–3662, 2023.

[7] M. Hu *et al.*, "QoS-aware software-defined multicast in LEO satellite networks," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 58, no. 6, pp. 5307–5317, 2022.

[8] Y. Li, S. Zhu, and J. Dai, "Joint user grouping and resource allocation for LEO satellite multicast," *IEEE Syst. J.*, vol. 17, no. 3, pp. 4695–4702, 2023.

[9] P. Lian, F. Yan, H. Luo, Z. Wang, and S. Zhang, "Multicast source routing based on bloomed link identifiers for LEO satellite network," in *Proc. IEEE Int. Conf. Satell. Comput. (Satellite)*, 2022, pp. 13–18.

[10] M. Hu, M. Xiao, Y. Hu, C. Cai, T. Deng, and K. Peng, "Software defined multicast using segment routing in LEO satellite networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 835–849, 2024.

[11] L. Wang *et al.*, "Obstacle-aware multicast routing algorithm for large-scale LEO constellations," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 5, pp. 4551–4563, 2024.

[12] K.-J. Zheng and J.-P. Sheu, "A dynamic multicast tree selection algorithm in LEO satellite networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2022, pp. 1546–1551.

[13] B. Kempton and A. Riedl, "Network simulator for large low earth orbit satellite networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2021, pp. 1–6.

[14] D. Watel and M.-A. Weisser, "A practical greedy approximation for the directed Steiner tree problem," *J. Comb. Optim.*, vol. 32, pp. 1327–1370, 2016.

[15] N. Pachler, I. del Portillo, E. F. Crawley, and B. G. Cameron, "An updated comparison of four low earth orbit satellite constellation systems to provide global broadband," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC WKSHPS)*, 2021, pp. 1–7.

[16] J.-P. Sheu, C.-W. Chang, and Y.-C. Chang, "Efficient multicast algorithms for scalable video coding in software-defined networking," in *Proc. IEEE 29th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun (PIMRC)*, 2015, pp. 2089–2093.