



## Article

# Delay-Sensitive Service Provisioning in Software-Defined Low-Earth-Orbit Satellite Networks

Feihu Dong <sup>1,2,\*</sup> , Yasheng Zhang <sup>2</sup>, Guiyu Liu <sup>3</sup> , Hongzhi Yu <sup>3</sup> and Chenhua Sun <sup>2</sup><sup>1</sup> School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China<sup>2</sup> The 54th Research Institute of CETC, Shijiazhuang 050081, China<sup>3</sup> The School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

\* Correspondence: dongfh@bupt.edu.cn

**Abstract:** With the advancement of space technology and satellite communications, low-Earth-orbit (LEO) satellite networks have experienced rapid development in the past decade. In the vision of 6G, LEO satellite networks play an important role in future 6G networks. On the other hand, a variety of applications, including many delay-sensitive applications, are continuously emerging. Due to the highly dynamic nature of LEO satellite networks, supporting time-deterministic services in such networks is challenging. However, we can provide latency guarantees for most delay-sensitive applications through data plane traffic shaping and control plane routing optimization. This paper addresses the routing optimization problem for time-sensitive (TS) flows in software-defined low-Earth-orbit (LEO) satellite networks. We model the problem as an integer linear programming (ILP) model aiming to minimize path handovers and maximum link utilization while meeting TS flow latency constraints. Since this problem is NP-hard, we design an efficient longest continuous path (LCP) approximation algorithm. LCP selects the longest valid path in each topology snapshot that satisfies delay constraints. An auxiliary graph then determines the routing sequence with minimized handovers. We implement an LEO satellite network testbed with Open vSwitch (OVS) and an open-network operating system (ONOS) controller to evaluate LCP. The results show that LCP reduces the number of path handovers by up to 31.7% and keeps the maximum link utilization lowest for more than 75% of the time compared to benchmark algorithms. In summary, LCP achieves excellent path handover optimization and load balancing performance under TS flow latency constraints.

**Keywords:** low earth orbit; satellite network; time-sensitive; traffic shaping; routing optimization



**Citation:** Dong, F.; Zhang, Y.; Liu, G.; Yu, H.; Sun, C. Delay-Sensitive Service Provisioning in Software-Defined Low-Earth-Orbit Satellite Networks. *Electronics* **2023**, *12*, 3474. <https://doi.org/10.3390/electronics12163474>

Academic Editors: Peiyang Zhang, Haotong Cao and Keping Yu

Received: 17 July 2023

Revised: 10 August 2023

Accepted: 13 August 2023

Published: 16 August 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Entering the second decade of the 21st century, low-Earth-orbit (LEO) constellation networking has entered a second wave of development after more than 10 years of dormancy. The emergence of LEO giant constellations such as Starlink, OneWeb, and Lightspeed has driven the rapid development of space-based networks. LEO satellite networks can provide global low-latency and broadband network access through hundreds to tens of thousands of satellites. Networks characterized by LEO giant constellations and hybrid network architectures are developing rapidly. Massive satellites are usually deployed at multiple orbit heights, using a deployment method mainly based on low-inclination constellations and including a small number of near-polar orbit satellites to achieve global coverage. Additionally, the deployment of inter-satellite links has become very common, especially with significant advancements in laser inter-satellite link technology, greatly improving the data transmission capacity between satellites. This allows the LEO constellation network to be completely independent of the ground network, achieving the “non-landing” transmission of massive data. Consequently, LEO satellite networks are rapidly evolving

into multi-functional, composite space-based networks for global communication, mobile internet, and more.

Compared to high-Earth-orbit satellites, LEO satellites offer advantages such as low latency, low transmission loss, high speed, and lower cost. They are considered a vital solution for improving network coverage and communication efficiency. With the rapid development of LEO satellite networks, LEO constellation networks have emerged as a viable option for numerous application scenarios and service requirements. However, as the demand for delay-sensitive services like global sensing, emergency communications, remote sensing, and remote control continues to increase, the performance requirements for transmission in LEO satellite networks are becoming more stringent [1].

Currently, the mainstream technology for solving real-time transmission problems is time-sensitive networking (TSN) technology proposed by the IEEE 802.1 working group. TSN integrates key technologies such as time synchronization, resource management, gate control scheduling, preemption mechanism, and seamless redundancy. It not only enables the real-time transmission of delay-sensitive services based on traditional Ethernet but also ensures compatibility with non-delay-sensitive service. Currently, TSN technology is mainly applied to wired networks such as industry and vehicular networks and is gradually expanding to fields such as Wi-Fi, 5G fronthaul networks, and space networks. However, compared to ground networks, the propagation delay of inter-satellite links in space networks is usually larger and rapidly changing due to the long distance and constant variation of inter-satellite links. For this reason, it is very difficult to achieve time-deterministic service guarantees based on precise time gate control scheduling strategies in LEO satellite networks. Therefore, based on existing literature, current solutions for real-time transmission in space missions mainly focus on the wired networks within satellites and lack end-to-end transmission solutions for inter-satellite links. However, with the increase in the number of nodes in low-orbit satellite networks and the continuous reduction in inter-satellite link propagation delay, supporting delay-sensitive services on LEO satellite networks is feasible through the support of satellite nodes and effective network management strategies.

In order to effectively support delay-sensitive services in LEO satellite networks, three requirements must be met: first, satellite nodes need to have a bandwidth guarantee strategy to meet service bandwidth requirements and avoid bandwidth contention between services. Second, network traffic distribution should be balanced to avoid network link congestion. Third, the propagation delay of the service path should meet the service's requirements for delay. For the first requirement, we can achieve it through efficient traffic shaping techniques such as the asynchronous traffic shaper (ATS) [2]. For the second and third requirements, we can develop network dynamic traffic routing optimization schemes based on software-defined networking (SDN) technology on the network control plane. Therefore, this paper mainly studies how to combine the above two methods to effectively support delay-sensitive services in LEO satellite networks.

However, achieving efficient dynamic traffic routing optimization in LEO satellite networks still presents significant challenges. The reasons are as follows.

1. Rapid and dynamic changes in network topology: LEO satellites move rapidly and periodically in orbit, which leads to rapid changes in network topology. This may result in frequent reconfigurations of service routes, introducing significant reconfiguration overhead. Therefore, the service routing problem needs to consider how to minimize the number of path handovers while satisfying delay requirements.
2. Large network scale: The plans of various countries indicate that future LEO satellite networks will have thousands or even tens of thousands of satellites. Dynamic service routing in such a large-scale network requires algorithms with low complexity.
3. Unbalance traffic distribution: The distribution of users in LEO networks is uneven, which results in unbalance traffic distribution in LEO satellite networks. For example, densely populated areas have higher traffic load, while sparsely populated areas have

lower traffic load. Therefore, traffic routing optimization algorithms need to have better load balancing performance.

In traditional networks, there has been a lot of research on network traffic optimization problems [3,4], and many traffic optimization algorithms have been proposed. However, due to the unique characteristics of LEO satellite networks, these algorithms cannot be applied to LEO satellite networks. Therefore, based on the characteristics of LEO satellite networks, this paper studies the routing optimization problem of delay-sensitive services in LEO satellite networks. The main contributions of this paper are as follows:

1. This paper first presents the LEO satellite network model and the traffic routing optimization requirements for delay-sensitive service and then models the traffic routing optimization problem of delay-sensitive services in LEO satellite networks as an integer linear programming (ILP) problem based on the network model and service requirements;
2. Since the traffic routing optimization problem for delay-sensitive service in LEO satellite networks is an NP-hard problem, we propose an efficient approximation algorithm to solve this problem. The algorithm can calculate a routing sequence that meets the requirements of each delay-sensitive service. Specifically, the proposed algorithm has the following characteristics: (1) it can satisfy the delay constraints of delay-sensitive service; (2) the number of paths in the computed routing sequence is small, which can reduce the number of path handovers; (3) it can achieve network-wide traffic load balancing; (4) it has low complexity and can be applied to large-scale network scenarios; and (5) it has good optimization performance in terms of routing handovers and network load balancing;
3. We develop an LEO satellite network evaluation testbed based on OpenVSwitch (OVS) and the open-network operating system (ONOS) controller to evaluate our proposed delay-sensitive service provision solution.

The rest of this paper is organized as follows. Section 2 introduces the related work and motivations. Section 3 presents the network model and problem formulation. We introduce the proposed approximation algorithm for the routing optimization problem of delay-sensitive services in Section 4. The experimental testbed and results are described in Sections 5 and 6. We conclude the paper in Section 7.

## 2. Related Work and Motivation

### 2.1. Related Work

In terrestrial networks, one of the technologies for solving the real-time transmission problem of network services is time-sensitive networking (TSN) proposed by the IEEE 802.1 working group [5]. TSN integrates technologies such as time synchronization, traffic shaping, traffic routing and scheduling, and preemption mechanisms. This not only enables real-time transmission of delay-sensitive services based on traditional Ethernet but also ensures compatibility with non-delay-sensitive services. Currently, TSN research mainly focuses on issues like protocol standards formulation [5], performance analysis [6], traffic routing and scheduling [7–10], and time synchronization [11]. Although TSN network technology can provide deterministic delay guarantees in terrestrial networks, existing TSN network technologies cannot be directly applied to LEO satellite networks due to the variability of inter-satellite link distances.

On the other hand, edge computing technology extends the cloud computing platform to network edge nodes, providing users with multi-layer and heterogeneous computing resources. As a promising paradigm, edge computing is considered a key enabler to further increase the transmission rate and reduce processing delay for integrated satellite-terrestrial networks [12–14]. One typical and novel work in [14] addressed the allocation of bandwidth and computation resources in low-Earth-orbit satellite networks for Earth observation applications. In this context, the satellites themselves process observation data to avoid transmitting useless data to ground stations and wasting precious bandwidth. To transfer the data from the data-gathering satellite to the satellite within the station's range,

a routing path was needed. Both routing (determining the data's path) and computation resource allocation (deciding where the data should be processed—either on a satellite or on the ground) are interconnected challenges in this problem. Consequently, algorithms were developed to tackle these interconnected problems. However, the algorithms presented in [14] cannot be directly applied as a benchmark for our work because: (1) Our work focuses on the routing problem for a given set of continuous, infinite-time data transmission flow requests. In contrast, ref. [14] primarily considers routing design for Earth observation data, which typically involves periodic transmission; (2) In [14], the routing request only specifies the source satellite, while the destination is not predetermined and can be any ground station. The destination is determined simultaneously by computation resource allocation. Then, the emphasis of the routing is on arranging computation resources for applications and transmitting data to any ground station. Conversely, our work deals with predefined routing requests, and our concern is solely on fulfilling the routing requests from the source to the designated destination while guaranteeing QoS.

In this paper, we mainly support delay-sensitive services in LEO satellite networks through traffic shaping techniques in the data plane (such as ATS) combined with routing optimization algorithms in the control plane. Currently, there are many related studies on traffic routing optimization in terrestrial networks [3,4]. However, these methods cannot be directly applied to the rapidly changing topology of LEO satellite network scenarios through simple extensions. Although the topology structure of LEO satellite networks is dynamic, it also changes in a predictable manner [15,16]. Therefore, based on the characteristics of the periodic changes in satellite network topology, the motion period  $T$  of a satellite network is divided into  $m$  time slots  $T_1, T_2, \dots, T_m$ , where the time slot  $T_i$  corresponds to a topology snapshot  $G_i$  [17,18], which means that the topology snapshot  $G_i$  does not change during time slot  $T_i$ . Therefore, satellite network routing is actually finding a suitable path for each flow on  $m$  topologies, that is, a flow's routing strategy  $P$  is a set of  $m$  paths  $p_1, p_2, \dots, p_m$ . It is worth noting that paths  $p_i$  and  $p_j$  of flow  $f$  corresponding to two time slots  $T_i$  and  $T_j$  may be the same ( $p_i = p_j$ ). In other words, a flow may choose the same routing path across multiple time slots. The method based on virtual topology was first proposed in [17]. In order to reduce the computational cost on satellites, the routing scheme for each time slot is calculated in the ground control center and the routing scheme corresponding to each time slot is uploaded when the satellite passes through the ground control center. In addition, to improve routing performance, the paper proposes a topology snapshot segmentation strategy, which comprehensively considers system performance and limited satellite buffer space. This optimization strategy sacrifices equal-length time slots and divides them into unequal-length time slots, merging too-short time slots to balance routing delay and storage overhead.

Another model for dealing with dynamic changes in satellite network topology is the virtual node model. In the virtual node model, the ground coverage area is divided into a grid, and virtual nodes are created to represent each grid position. Each virtual node corresponds to a physical satellite that is directly above the corresponding grid position at a certain time and is responsible for communication with the ground. When the satellite moves away from the grid position, another satellite in the same orbit will take over its communication responsibilities. Through the virtual node model, the entire satellite network is transformed into a static topology structure, and the routing forwarding between satellite nodes can be realized by maintaining the mapping relationship between physical satellites and virtual nodes [19]. The work in [19] is the first to establish the virtual node model, establish a mapping table between virtual nodes and satellites, and propose a distributed routing algorithm (DRA). DRA determines the forwarding path of data packets on satellite nodes by finding the shortest path between virtual nodes. DRA is a distributed and independent routing strategy that does not involve data interaction between satellite nodes. However, the performance of this algorithm significantly declines when satellite nodes or links fail.

To improve DRA, the research community has proposed several routing optimization problems in satellite networks, most of which aim for load balancing. To improve the DRA algorithm, the low-complexity routing algorithm (LCRA) was proposed in [20]. The LCRA algorithm allows satellite nodes to write and exchange information about their network congestion state and consider the congestion state of neighboring nodes in the routing decision process. This reduces local congestion in the routing results, thereby enhancing the algorithm's performance. In [21], the authors proposed to allocate the remaining bandwidth of lightly loaded neighbors to congested satellites to avoid link congestion. In [22], each node in the network maintains the state information of the entire network and filters out overloaded nodes when making routing decisions. To reduce information exchange, the authors in [23] limit the exchange of state information between neighboring nodes. Assuming a globally synchronized state, the authors in [24] proposed an agent-based load balancing routing scheme. In [25], the authors proposed an on-demand load balancing routing scheme, which limits the range of state synchronization.

It is true that the existing routing optimization solutions have a coarse control granularity. Although they can achieve a certain degree of load balancing based on network status, these methods cannot provide fine-grained routing schemes for each flow and thus cannot guarantee the latency requirements of each latency-sensitive application. Therefore, this article proposes a fine-grained traffic routing optimization method for latency-sensitive applications based on the flexible control capability of SDN.

## 2.2. Motivation

With the development of the 6G integrated space-terrestrial network, some important delay-sensitive services are bound to be carried on the LEO satellite network. As mentioned earlier, ensuring service transmission delay in a satellite network requires the cooperation of both the network data plane and control plane. In this paper, we assume that the data plane mainly uses traffic shaping methods to ensure the transmission bandwidth of flows, avoiding long queuing delays caused by node overload. There are already some mature solutions for traffic shaping methods, so our work will mainly focus on establishing the best way to ensure the transmission delay requirements of delay-sensitive services through control plane routing optimization. On the other hand, with the maturity of SDN technology, achieving per-flow routing control in a satellite network based on SDN technology becomes possible. However, the routing optimization of delay-sensitive services in a low-Earth-orbit satellite network based on the SDN architecture still faces the following challenges.

(1) Knowing how to reduce the overhead of routing handovers. During the entire cycle of satellite topology changes, a flow's path may experience multiple handovers. Although path handover can be achieved by installing flow tables in the SDN architecture, frequent flow table installations will generate a large amount of control overhead. In addition, path handovers may also cause packet reordering and increase delay jitter. Therefore, when calculating routing strategies for flows, it is necessary to minimize the number of path handovers as much as possible. For example, in Figure 1 we will prioritize routing strategy 1 among the three consecutive time slots (slot  $k - 1$ , slot  $k$ , and slot  $k + 1$ ) because it only requires two handovers compared to three for routing strategy 2.

(2) Knowing how to achieve load balancing while ensuring the delay constraints of delay-sensitive services. Due to the uneven distribution of population, network traffic in satellite networks often fluctuates and is unbalanced. To avoid the impact of congestion on the transmission delay of delay-sensitive services, it is crucial to achieve maximum network traffic distribution. Therefore, in addition to the cost of path handovers, the routing optimization algorithm also needs to consider achieving load balancing as much as possible while meeting the constraints of flow transmission delay. For example, in the example given in Figure 2, we will prioritize selecting routing strategy 2 because both strategies have the same number of path handovers, but strategy 2 has better load balancing.



(3) Knowing how to reduce the complexity of the algorithm. In the future, the number of nodes in the LEO satellite network may be large, and the constrained routing optimization problem under dynamic topology is more complex than that under static topology. In this case, low-complexity algorithms need to be designed to provide delay-sensitive services in a short period.

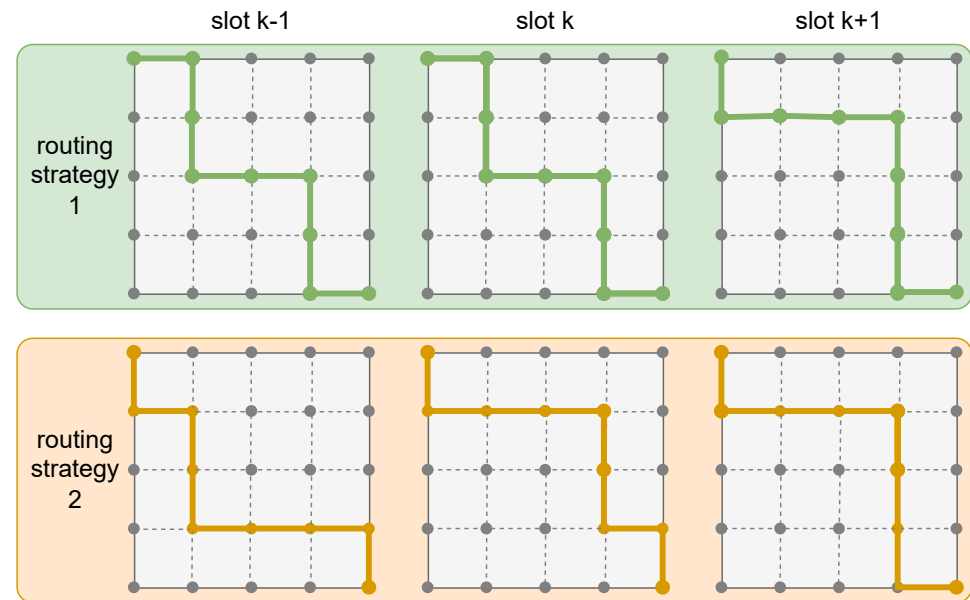


Figure 1. Routing strategies with different number of path handovers.

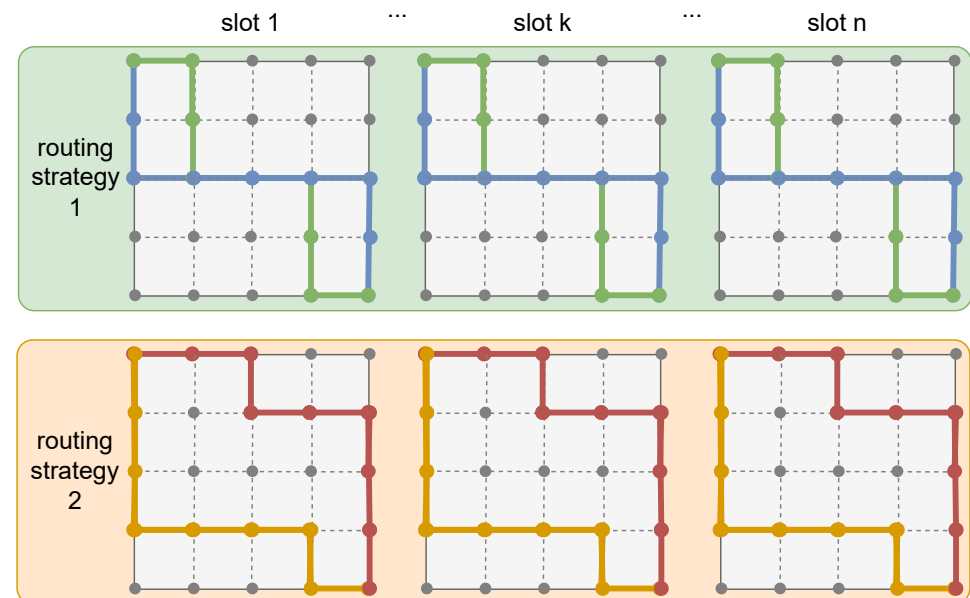


Figure 2. Routing strategies with different load balancing performances.

### 3. Network Model and Problem Formulation

#### 3.1. Network Topology Model

We model an LEO satellite network as a directed graph  $G(V, E)$ , where  $V$  is the set of satellites and  $E$  is the set of all possible links between satellites. In order to cope with the dynamic changes of topology, this paper adopts the virtual topology approach. The virtual topology method divides the motion period  $T$  of the satellite network into  $|T|$  time slots. Each time slot  $i$  corresponds to a topology snapshot [17,18], indicating that there are no changes in the topology snapshot during time slot  $i$ . For simplicity, we assume that all time

slots have the same length. However, the algorithm designed in this paper can also be used directly in the cases of non-equal time slots.

Since satellites move continuously over time, a link  $e \in E$  may be disconnected during time slot  $i$ . Therefore, we use  $y_e^i \in \{0, 1\}$  to indicate whether link  $e$  exists in the  $i$ -th time slot. Additionally, the propagation delay of links varies over time, so we use  $d_e^i$  to represent the maximum propagation delay of link  $e$  in time slot  $i$ .

In this paper, we mainly focus on scheduling time-sensitive (TS) flows. The set of TS flows to be scheduled is denoted as  $F$ , where the  $j$ -th TS flow is represented by  $f_j$ . Let  $b_j$  and  $r_j$  represent the bandwidth requirement and maximum transmission delay constraint of flow  $f_j$ .

### 3.2. Problem Formulation

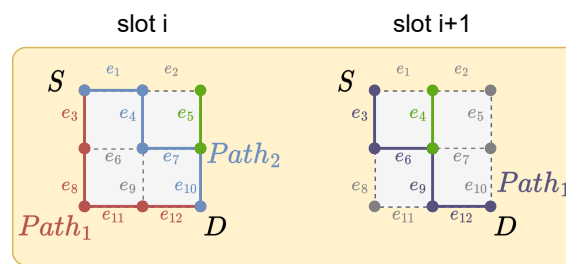
In this work, our primary goal is to meet the propagation delay constraints and minimize queuing delay for TS services in LEO satellite networks through optimized routing. Furthermore, due to the highly dynamic topology of LEO satellite networks, we need to consider minimizing the control overhead caused by frequent routing changes. Therefore, in modeling the routing problem, we have incorporated service propagation delay requirements and network bandwidth constraints, while load balancing and routing handover overhead are set as optimization objectives. Moreover, our proposed routing algorithm can also easily incorporate other optimization goals, such as minimizing energy consumption, through simple extensions. Since the routing optimization problem for TS flows in LEO satellite networks needs to consider multiple topology snapshots simultaneously, it is much more complex than the routing optimization problem in static topologies. There are usually two modeling approaches for network routing optimization problems: node-link and link-path [26]. In the link-path model, the candidate paths for each flow are given in advance, and thus we can control the complexity of the model by adjusting the number of candidate paths. However, the node-link model requires the direct computing of all paths for the TS flows, thus introducing more variables and constraints, which makes the node-link model difficult to solve in large-scale networks. To reduce the complexity of the model, we choose the link-path approach to model the problem in this paper.

For the sake of clarity, we provide a comprehensive list of the notations used in Table 1. In the following text, we will first introduce the routing constraints and then give the optimization objective.

**Table 1.** Notations.

Notations	Meaning
$\mathcal{T}$	The set of time slots within a motion period
$i$	The $i$ -th time slot
$G(V, E)$	The directed graph of the network topology
$V$	The set of satellite nodes in the network
$E$	The set of links between satellites
$F$	The set of TS flows
$f_j$	The $j$ -th TS flow
$c_e$	The capacity of link $e$
$b_j$	The bandwidth requirement of the $j$ -th flow
$d_e^i$	The maximum propagation delay of link $e$ in time slot $i$
$r_j$	The maximum delay constraint of the $j$ -th flow
$y_e^i \in \{0, 1\}$	A given binary constant indicating whether link $e$ exists in time slot $i$
$P_{j,i}$	A given set of eligible candidate paths of flow $j$ in time slot $i$
$l_{k,e}^{i,j} \in \{0, 1\}$	A given binary constant indicating whether the $k$ -th candidate path flow $f_j$ in time slot $i$ goes through link $e$
$x_k^{i,j} \in \{0, 1\}$	A decision binary variable indicating whether path $k$ is chosen for flow $f_j$ in time slot $i$
$h_j^{i-1,i} \in \{0, 1\}$	A dummy binary variable indicating whether flow $f_j$ changes its route during time slot $i - 1$ to $i$

Using Figure 3, we can illustrate the decision variables more clearly. There are two consecutive time slots, slot  $i$  and slot  $i + 1$ ; each node represents a satellite, and solid lines denote available links between two satellites. We observe that there is a change in the state of  $e_1, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}$ , while  $e_3, e_4, e_{12}$  remain in a connected state, and  $e_2$  remains unconnected. Given that  $y_e^i$  indicates whether link  $e$  exists in time slot  $i$ , it can be easily observed that  $y_{e_1}^i = 1, y_{e_2}^i = 0$ . Then, we assume there is a TS flow  $f_j$  from source node  $S$  to destination node  $D$ , and in time slot  $i$ , there are two paths, Path 1 and Path 2, from the candidate paths set  $P_{j,i}$ . We can observe that Path 2 includes link  $e_1$  and excludes link  $e_2$ , so  $l_{2,e_1}^{i,j} = 1, l_{2,e_2}^{i,j} = 0$ . Then, we assume that in slot  $i$  and  $i + 1$ , we have both chosen Path 1, that is,  $x_1^{i,j} = 1, x_2^{i,j} = 0$  and  $x_1^{i+1,j} = 1$ . However, a path handover occurs due to the fact that the links of the paths selected in the two slots are not entirely coincident, so  $h_j^{i,i+1}$  will be set to 1.



**Figure 3.** Example of decision variables.

Firstly, the maximum delay of the path chosen for each TS flow in each topology snapshot should meet the flow's delay constraint.

$$\sum_{k \in P_{j,i}} \sum_{e \in E} x_k^{i,j} l_{k,e}^{i,j} d_e^i \leq r_j, \quad \forall j \in F, \forall i \in \mathcal{T} \quad (1)$$

If flow  $f_j$  uses path  $k$  in time slot  $i$ , then every link traversed by path  $k$  should exist in time slot  $i$ , i.e., we have the following constraint:

$$x_k^{i,j} l_{k,e}^{i,j} \leq y_e^i, \quad \forall j \in F, \forall i \in \mathcal{T}, \forall e \in E, \forall k \in P_{j,i} \quad (2)$$

To avoid packet reordering, we assume that each TS flow can only use one path within each time slot. Therefore, the decision variable  $x_k^{i,j}$  needs to satisfy the following constraint:

$$\sum_{k \in P_{j,i}} x_k^{i,j} = 1, \quad \forall j \in F, \forall i \in \mathcal{T} \quad (3)$$

Furthermore, to avoid introducing significant unpredictable delays due to traffic overload, we require that the total size of TS flows carried on each link should be smaller than its capacity.

$$\sum_{j \in F} \sum_{k \in P_{j,i}} x_k^{i,j} l_{k,e}^{i,j} b_j \leq c_e, \quad \forall e \in E, \forall i \in \mathcal{T} \quad (4)$$

After introducing the constraints that TS flows need to satisfy, we now turn our attention to the optimization objective of the problem. As introduced in Section 2, we mainly consider two optimization objectives in this problem. First of all, we believe that the future LEO satellite networks will use SDN technology to control flow routing in a fine-grained manner. Therefore, in order to reduce the control overhead of SDN, we need to minimize the total number of path handovers. Secondly, network congestion can lead to a significant increase in packet queuing delay. Therefore, to better guarantee the transmission delay requirements of TS flows, we need to achieve network load balancing.



Firstly, we address the optimization of the number of path handovers. For any given flow, within two consecutive time slots  $i - 1$  and  $i$ , a path handover occurs if the routing path changes between these two time slots. Specifically, a path handover is counted if any link used by the flow's path in time slot  $i - 1$  is different from the links used in time slot  $i$ . Furthermore, even if multiple links change between the two time slots, only one path handover is counted. To model the path handovers, we introduce a binary variable  $h_j^{i-1,i}$  which indicates whether flow  $j$  changes its path during time slot  $i - 1$  to  $i$ . The following constraint are added to relate  $h_j^{i-1,i}$  with the path selection variables  $x_{j,i}^k$ :

$$\left| x_k^{i-1,j} l_{k,e}^{i-1,j} - x_k^{i,j} l_{k,e}^{i,j} \right| \leq h_j^{i-1,i} \quad (5)$$

To satisfy the constraints of the ILP formulation, we expand the above constraint into the following two constraints:

$$x_k^{i,j} l_{k,e}^{i,j} - x_k^{i-1,j} l_{k,e}^{i-1,j} \leq h_j^{i-1,i}, \quad \forall e \in E, \forall j \in F, \forall i \in [2, |\mathcal{T}|] \quad (6)$$

$$x_k^{i-1,j} l_{k,e}^{i-1,j} - x_k^{i,j} l_{k,e}^{i,j} \leq h_j^{i-1,i}, \quad \forall e \in E, \forall j \in F, \forall i \in [2, |\mathcal{T}|] \quad (7)$$

To quantitatively analyze the number of path switches, we let  $\beta$  denote the average number of path switches per time slot for each flow, which is defined as:

$$\beta = \frac{\sum_{i=2}^{|\mathcal{T}|} \sum_{j \in F} h_j^{i-1,i}}{(|\mathcal{T}| - 1)|F|} \quad (8)$$

Then, to indicate the degree of the load balancing of the satellite network, we define  $\alpha$  as the maximum link utilization in the network. From the right-hand side of Equation (9), we can calculate the link utilization for each link within all time slots, where  $\alpha$  represents the maximum value. Therefore, by utilizing Equation (9) we ensure that  $\alpha$  is greater than the link utilization of each individual link within all time slots.

$$\alpha \geq \frac{\sum_{j \in F} \sum_{k \in P_{j,i}} x_k^{i,j} l_{k,e}^{i,j} \cdot b_j}{c_e}, \quad \forall e \in E, \forall i \in \mathcal{T} \quad (9)$$

Finally, taking into account both the number of path handovers and network load balancing, the ILP formulation of our problem is:

$$\begin{aligned} & \text{minimize} && \alpha + \beta \\ & \text{s.t.} && \text{Equations (1)–(9)} \end{aligned} \quad (10)$$

So far, we have successfully modeled the routing optimization problem for TS flows in LEO satellite networks as an integer linear programming (ILP) model. While solving this model can provide the optimal routing solution for TS flows, the complexity of solving ILP models is exponential, which is not feasible for large-scale LEO satellite networks. Furthermore, it can be easily proven that this problem is an NP-complete problem since even when simplifying the problem to a single time slot scenario, it remains NP-hard [27]. To effectively solve this problem, we propose an approximate algorithm called the longest continuous path (LCP), which strikes a good balance between solution speed and performance. The details of the LCP algorithm will be discussed in the next section.

#### 4. Approximation Algorithm Design

The heuristic algorithm LCP sorts traffic flows in descending order of the priority of the traffic flows; then, these flows are sequentially routed one by one at each time slot. We consider that there are six priorities of traffic flows, which can be justified by the fact that there are usually eight priorities in a switch and that two priorities are reserved for other traffic flows. We note that with this sorting, flows with higher priority are routed

first, and if flows have the same priority, the flows with larger bandwidth requirements are routed first.

After the sorting, the routing paths of a traffic flow have to be derived at all time slots, where each of the paths is the routing detail of the traffic flow at the time slot. Then, the traffic flow can be carried throughout the sequence of all the time slots. Algorithm 1 shows the details of the LCP algorithm, where the majority of the commandlines are to derive routing path for a traffic flow.

In Algorithm 1, from line 2 to the end, the routing paths of each traffic flow are derived at all time slots of the satellite network. At line 5 and line 6, a routing path satisfying the latency requirement and considering load balance is selected at each time slot. Specifically, in a time slot  $\mathcal{T}_i$ , the K Dijkstra shortest paths (K-DSP) considering latency are obtained first. Then, the one with the minimal of the maximum link utilization is chosen for load balance consideration in the network. This selected path is stored in the set  $P$ . We repeat this process in subsequent time slots, resulting in a set of routing paths of the traffic flow  $f_j$  in each of the time slots.

---

**Algorithm 1:** The LCP algorithm
 

---

**Data:** Set of traffic flows  $F$ , satellite network  $G$

**Result:** Routing results  $R$

- 1 Obtain the set of time slots  $\mathcal{T}$  based on  $G$ ;
- 2 Sort flows in  $F$  in descending order of priority and bandwidth requirements if flows have the same priority;
- 3 **for**  $j$  in  $1 \dots |F|$  **do**
- 4     **for**  $i$  in  $1 \dots |\mathcal{T}|$  **do**
- 5         Use the K-DSP algorithm to find at most K paths on  $\mathcal{T}_i$  that satisfy the delay requirement for  $f_j$ ;
- 6         Choose  $path_j^i$  that minimizes the maximum link utilization; record  $path_j^i$  in set  $P$ ;
- 7     Run Algorithm 2 to construct the auxiliary graph  $G'_j$  for flow  $f_j$  based on  $P$ ;
- 8     Use the DSP algorithm on  $G'_j$  to determine the paths chosen by flow  $f_j$  in each time slot, and record them in the actual routing result set  $R$ ;
- 9     Update the link capacity at each time slot according to the final paths;

**Output:**  $R$

---

From line 8 to the end, the routing paths of the traffic flow are improved considering the path handovers. Although the routing paths obtained above can effectively meet the latency requirements of traffic flows and consider network load balancing, this ignores a factor that may affect the performance of the satellite network: the number of path handovers. A path handover is the online handover of the routing path that happens between two adjacent time slots, which may cause the packet reordering problem and increase the network latency jitter. Therefore, in the path selections for traffic flows, it is important to consider the number of path handovers in all time slots. To address this, based on the set  $P$ , we will construct an auxiliary graph  $G'$  for the final path selection.

The details of the construction of the auxiliary graph are shown in Algorithm 2. For a better understanding, we give an example to illustrate the auxiliary graph in Figure 4. In the graph, for each time slot, one path that satisfies the delay requirement while minimizing the maximum link utilization is selected (storing in  $P$ ). The selected path is then checked for availability in future time slots, which are shown from line 3 to line 7 in Algorithm 2. For example, in time slot  $\mathcal{T}_1$ , a flow selects  $path_1$  and checks if  $path_1$  is also available in  $\mathcal{T}_2$ . Therefore, a line is added between  $\mathcal{T}_1$  and  $\mathcal{T}_2$  in the diagram to represent the availability of  $path_1$  in both time slots. The process continues to check the availability of  $path_1$  in  $\mathcal{T}_3$ ,  $\mathcal{T}_4$ , and  $\mathcal{T}_5$ . It is found that  $path_1$  is available in  $\mathcal{T}_1$ ,  $\mathcal{T}_2$ ,  $\mathcal{T}_3$ , and  $\mathcal{T}_4$  but not in  $\mathcal{T}_5$ . As a result, in the auxiliary diagram  $G'$ , there are three lines from  $\mathcal{T}_1$  to  $\mathcal{T}_2$ ,  $\mathcal{T}_3$ , and  $\mathcal{T}_4$ . In  $\mathcal{T}_2$ ,  $path_2$  is the

routing path, and a similar process is followed to determine the availability of  $path_2$  in  $\mathcal{T}_3$  and  $\mathcal{T}_4$ . All of the routing paths in  $P$  for all time slots are evaluated at line 2 in Algorithm 2, and the auxiliary graph  $G'$  for the traffic flow is obtained. In line 8, we set each of the link costs to 1 in the auxiliary graph to signify that a path handover between two time slots incurs a cost of 1. We hope that the shortest number of hops is the number of path handovers of the target flow throughout the entire satellite topology cycle, meaning that every edge passed except for the first hop represents a path handover. Therefore, finding the shortest path in the auxiliary graph enables the selection of a routing solution that minimizes the number of path handovers across time slots.

---

**Algorithm 2:** Construct auxiliary graph
 

---

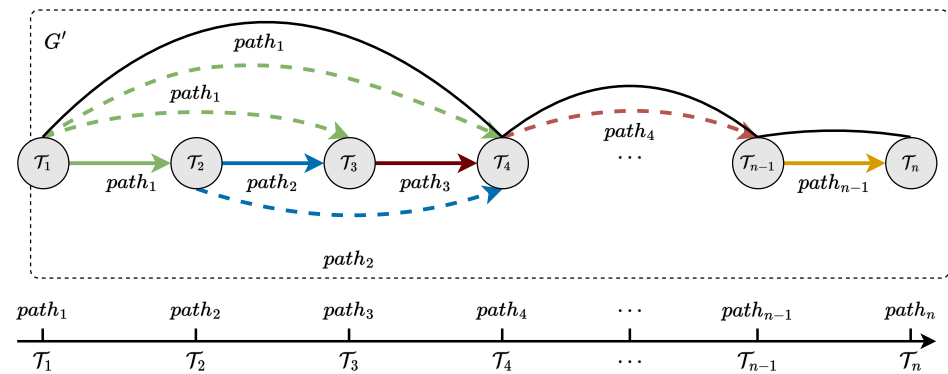
**Data:** Set of routing path  $P = \{path_1, path_2, \dots, path_{|\mathcal{T}|}\}$

**Result:** Auxiliary graph

```

1 Add  $|\mathcal{T}|$  nodes in an empty graph;
2 for  $j$  in  $1, \dots, |\mathcal{T}|$  do
3   for  $t$  in  $j + 1, \dots, |\mathcal{T}|$  do
4     if  $path_j$  is valid in  $\mathcal{T}_t$  then
5       Add a link between  $\mathcal{T}_j$  and  $\mathcal{T}_t$ ;
6     else
7       Break;
8 Set each of link costs to 1.
```

---



**Figure 4.** Routing auxiliary graph.

At line 8 of Algorithm 1, the auxiliary graph is built for the traffic flow by running Algorithm 2, and at line 9, the shortest path that aims to reduce the number of path handovers is derived as the final routing path of the traffic flow. It is noted that the final routing path has already met the delay requirement and considered the load balance; moreover, it reduces the number of path handovers.

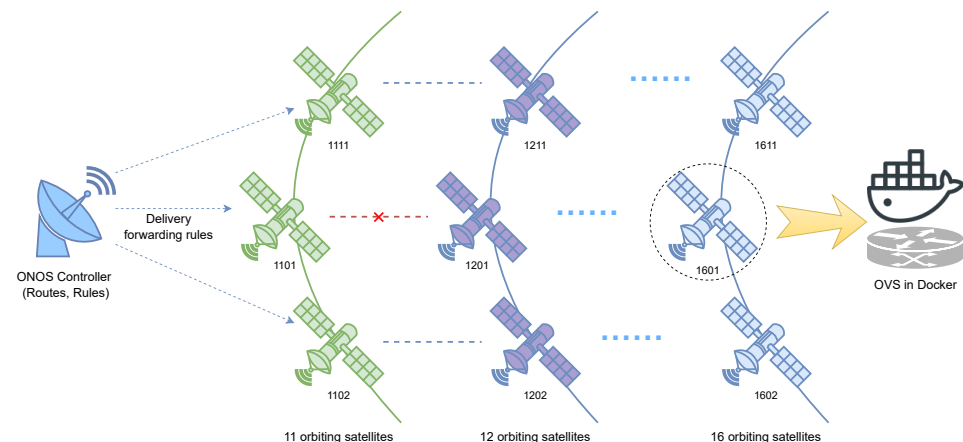
Finally, I will discuss the computational complexity of the proposed LCP algorithm. The LCP algorithm has two main steps: (1) Finding candidate paths in each time slot (lines 5–6 in Algorithm 1); this step uses the K-DSP algorithm to find at most K candidate paths in each time slot satisfying the delay constraint. The computational complexity of the K-DSP algorithm is  $O(KV(E + V \log V))$  for a graph with V nodes and E edges. (2) Constructing the auxiliary graph and finding the final path sequence (lines 7–9 in Algorithm 1); this step constructs an auxiliary graph with V nodes (number of time slots). Then, it runs the Dijkstra's shortest path algorithm on the auxiliary graph, which has a computational complexity of  $O(V^2)$ . Therefore, the overall computational complexity of the LCP algorithm is  $O(T[KV(E + V \log V) + V^2])$ , where T is the number of time slots. Since T, K, V, and E are all bounded in a given network, the LCP algorithm has a polynomial computational complexity.

In contrast, directly solving the ILP formulation of the problem has exponential complexity. Therefore, the proposed LCP approximation algorithm is much more efficient at finding solutions than directly solving the ILP model for large problem sizes. In summary, the LCP algorithm has a polynomial time complexity, making it suitable for real-time path computation in large dynamic LEO satellite networks. However, the lower complexity comes at the cost of approximate solutions instead of optimal ones. But LCP achieves excellent performance in terms of path handovers and load balancing compared to the other algorithms considered in this paper.

## 5. Satellite Experiment Testbed

In this section, we will introduce a satellite experiment testbed that will be used for the performance evaluation of the algorithms in this paper. To build a satellite experiment testbed, the satellite tool kit (STK) [28] software will be used to export information such as link connectivity and propagation delay among satellites during a orbital cycle period.

In the testbed, each satellite has packet forwarding capability, and this forwarding capability is provided by OVS running within a Docker container shown in Figure 5. Then, we can install as many Dockers as satellites. For the links between satellites, we design that a link between two satellites is provided by a pair of virtual network interfaces (veth pair). Then, by adding ends of the veth pair to two OVSs that are in two separate Dockers, an inter-satellite link is established. Removing the end from OVS means the link is disconnected. The delay and bandwidth of the inter-satellite links are configured using the traffic control tool provided by the kernel. We note that to realize the variability of the satellite topology and the delay and bandwidth of the links, a testbed manager is proposed and works as the background process of the testbed (not shown in the figure), which dynamically configures the veth pairs in the testbed according to the link connectivity and propagation delay information among satellites input from STK.



**Figure 5.** Satellite experiment testbed.

In this testbed, the experiment system is the Iridium satellite system, which is partially shown in Figure 5, with six orbital planes numbered from 11 to 16. Each orbital plane has 11 satellite nodes numbered from 01 to 11. For example, satellite 1101 represents satellite 01 on orbital plane 11. The distance between satellites with the same number but on adjacent orbital planes varies as the satellites move, resulting in changes in the propagation delay and link disconnections or reconnections. Satellites with adjacent numbers on the same orbital plane can always communicate, and the propagation delay remains stable.

Meanwhile, in the testbed, there is an ONOS controller that is an openflow controller. This controller calculates routing paths for traffic flows and gives routing information (flow tables) to OVSs that tells the OVSs how to forward the packets. Specifically, the ONOS receives routing requests for TS flows and runs the traffic routing algorithm proposed to allocate a routing path for the TS flows and distribute the flow tables to each satellite node (i.e., OVS). If the routing path of the traffic flow changes (path handover), the flow

tables must be resent from ONOS to the OVSs along the new path. Table 2 shows the basic configurations used in this experiment testbed.

**Table 2.** Basic configuration.

Software and Hardware	Configuration Information
OS	Ubuntu 20.04.4 LTS
CPU	Intel(R) Xeon(R) Gold 6133 CPU @ 2.50GHz
OpenVSwitch	2.16.0+
DPDK	20.11.3
ONOS	2.8.0
Redis	6.2.6
Docker	23.0.3

## 6. Experimental Result

In this section, we first introduce an approximation algorithm based on differential evolution (DE) as a comparison algorithm to the LCP algorithm and then evaluate the performances of the algorithms on the satellite testbed described above. We solve the ILP in the IBM ILOG CPLEX optimization studio solver, version 12.6.3.0, to solve the traffic routing optimization model.

### 6.1. Comparison Algorithm

The benchmark algorithm used in this paper is designed based on the DE algorithm, which is an efficient and general problem solving framework. In this paper, we optimize the design of parameters and strategy selection based on the original DE algorithm to better adapt to the problem in this paper. The DE-based algorithm searches the solution space extensively according to certain strategies, thereby obtaining a good solution. Therefore, we use the DE based algorithm to solve the routing optimization problem and use it as a benchmark algorithm. Based on the original DE algorithm framework, there have been many improved algorithms, which adopt different parameter control and strategy selection methods. In this paper, we adopt the linear population size reduction—success history-based adaptive DE (LSHADE) algorithm [29] to solve the problem. To adapt to our problem, we build a population model and design a fitness function.

In the path-based ILP formulation, the routing of a flow  $f_j$  in a time slot  $i$  can be represented as  $z^{j,i} = \{0, 1, 2, \dots, |P_{j,i}|\}$ .  $z^{j,i}$  represents that flow  $f_j$  chooses the  $z^{j,i}$ -th path for transmission in time slot  $\mathcal{T}_i$ . During the topology cycle of the network, the routing of flow  $f_j$  can be represented by a vector  $Z^j = (z^{j,1}, z^{j,2}, \dots, z^{j,|\mathcal{T}|})$ , which contains the routing paths for flow  $f_j$  in each time slot. The dimension of the vector is equal to the total number of time slots of the satellite network, denoted as  $|\mathcal{T}|$ . Therefore, a population individual can be represented by the vector  $POP = (Z^1, Z^2, \dots, Z^{|F|})$ , which contains the routing decisions for each flow on the given path in each time slot. Hence, the problem dimension is  $D = |\mathcal{T}| |F|$ . The fitness function used in the LSHADE algorithm can be nonlinear, so we can directly use the optimization objective without linearization. Meanwhile, in the LSHADE algorithm, the mutation process can generate decimal numbers. To ensure that the variables  $z^{j,i}$  are integers, we round off the decimal results generated during the mutation operation.

The LCP algorithm and the LSHADE algorithm proposed in this paper are two approaches to solving the ILP formulation. These algorithms aim to efficiently obtain approximate optimal solutions. We convert the solutions obtained by these three algorithms for the same problem into routing schemes for each individual TS flow. These routing schemes are then tested on the satellite experiment testbed to compare the performance of different algorithms.

For the other two comparison algorithm, TGM [30] focuses on finding the shortest path in each time slot as the routing result, while discrete-time dynamic virtual topology routing (DT-DVTR) [17] aims to find the path with the fewest path handovers in each time slot.

### 6.2. Number of Path Handovers COMPARISON

This simulation scenario contains 50 TS flows with bandwidth requirements varying around 10 Mbps. The proposed LCP algorithm, and the comparison algorithms DT-DVTR, TGM, LSHADE, and ILP, are used to allocate routes for these 50 TS flows. The routing results are analyzed to find the total number of path handovers during the entire network cycle period.

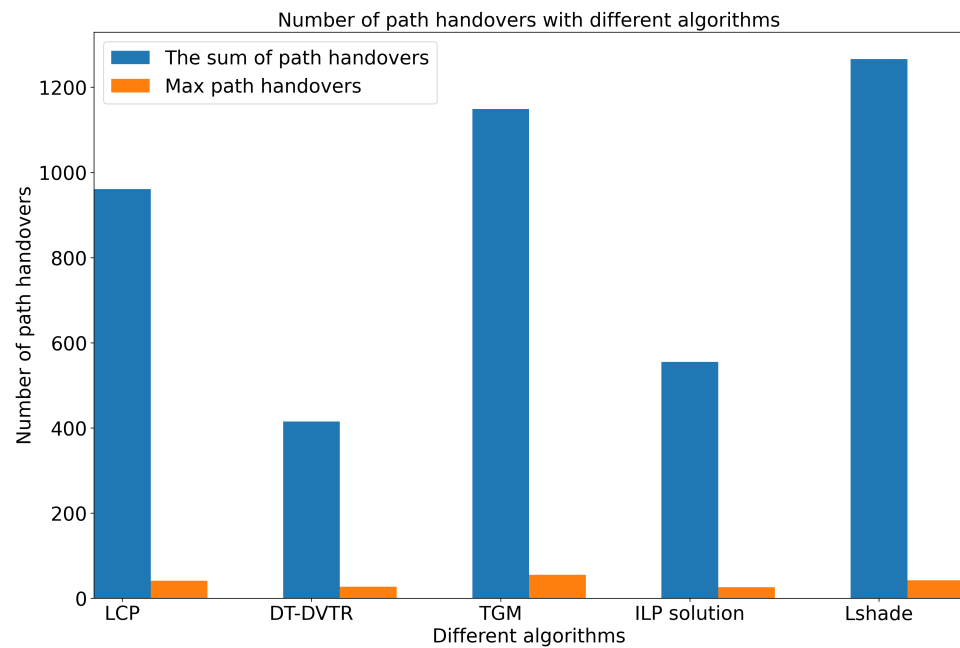
Table 3 shows the number of path handovers of ten flows and the average path handovers for all flows. We only list ten flows here since the remaining forty flows witness the similar results. LSHADE has the highest number of path handovers, followed by TGM and LCP, while DT-DVTR performs the best. This is because the LCP algorithm finds the longest-lasting path and uses the auxiliary graph to find the routing paths that have a smaller number of path handovers than TGM and LSHADE. However, this value is larger than DT-DVTR because DT-DVTR aims to find the path with the fewest path handovers in each time slot.

**Table 3.** Path handovers in each TS flow routing result.

Flow	Path Handovers				
	LCP	DT-DVTR	TGM	ILP Solution	Lshade
1	27	13	46	16	42
2	23	27	51	18	41
3	20	4	28	22	38
4	28	13	38	6	27
5	16	6	8	10	26
6	18	6	14	8	32
7	25	3	37	15	38
8	22	17	32	26	41
9	35	13	41	12	38
10	22	15	44	21	39
Average	19.22	8.3	22.98	11.1	25.32

The summation of the number of path handovers and the maximum number of path handovers in Table 3 are shown in Figure 6; DT-DVTR has the lowest summation value, followed by LCP and TGM.

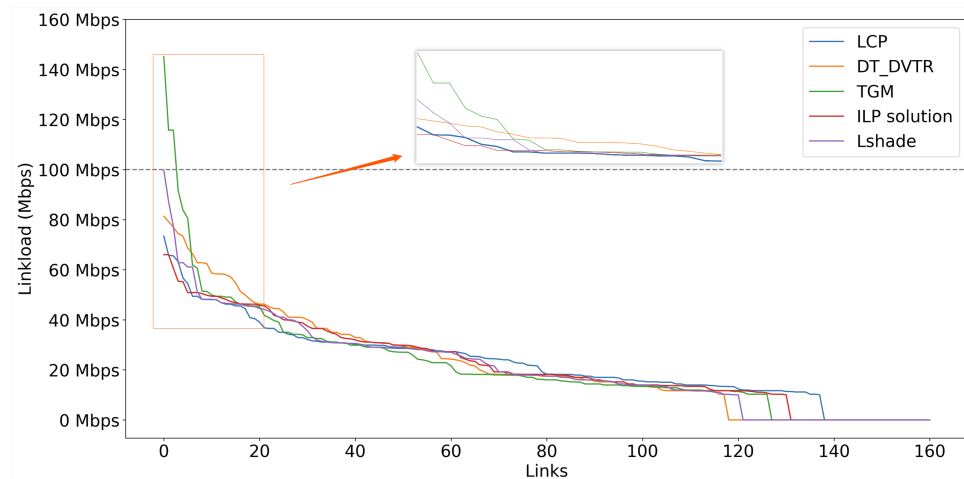




**Figure 6.** Path handovers for all TS flows.

### 6.3. Link Load Comparison

We calculate the traffic load on each link according to the routing results provided by the algorithms. Figure 7 shows the link loads of all network links. The link loads are sorted in descending order shown in the figure. Here, we only consider the link load in one time slot.

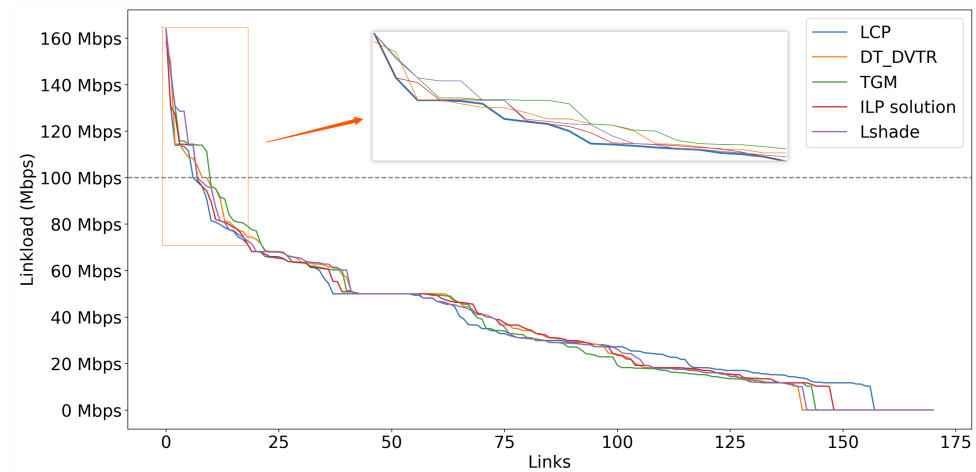


**Figure 7.** Link load with only TS flow.

In this experiment, the bandwidth capacity of link is 100 Mbps. Only the routing solution provided by the TGM algorithm lead's several linksexceeds the bandwidth capacity, which causes flow congestion. Compared to all the comparison algorithms, the LCP algorithm achieves the fewest maximum link loads due to load balancing considered in the algorithm. About 83% of the links (196 out of 236) have resource utilization rates below 40% during the network operation. This allows the link to accommodate more best effort (BE) traffic flows. The load balancing in LCP is slightly higher than that of the ILP model but still has high link-load-balancing performance.

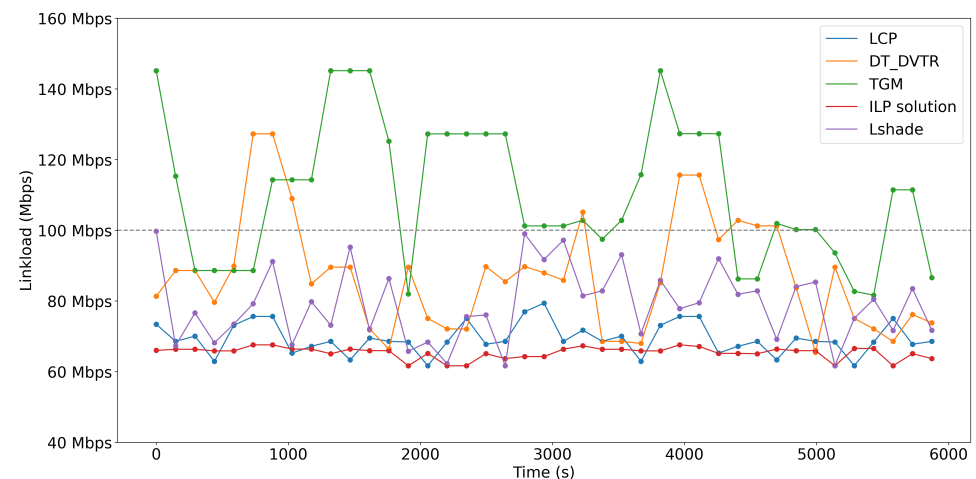
Figure 8 shows the link load after adding the 10 BE traffic flows with a bandwidth requirement of 50 Mbps. All BE flows are forwarded using the shortest path. It can be observed that after adding the BE flows, the LCP algorithm has the fewest congested links;

70% of the links (166 out of 236) have resource utilization rates below 40% during network operation; and most links having utilization rates between 20% and 40%, representing the best performance in load balancing.



**Figure 8.** Link load with TS and BE flow.

Then, we test the link load in one network topology cycle. Figure 9 shows the maximum link loads for each algorithm at every time slot. It can be seen that, in most time slots, ILP achieves a good load balancing, with the maximum link load stabilizing at around 70%. The proposed LCP algorithm and LSHADE also achieve load balancing and perform better than the two comparison algorithms.

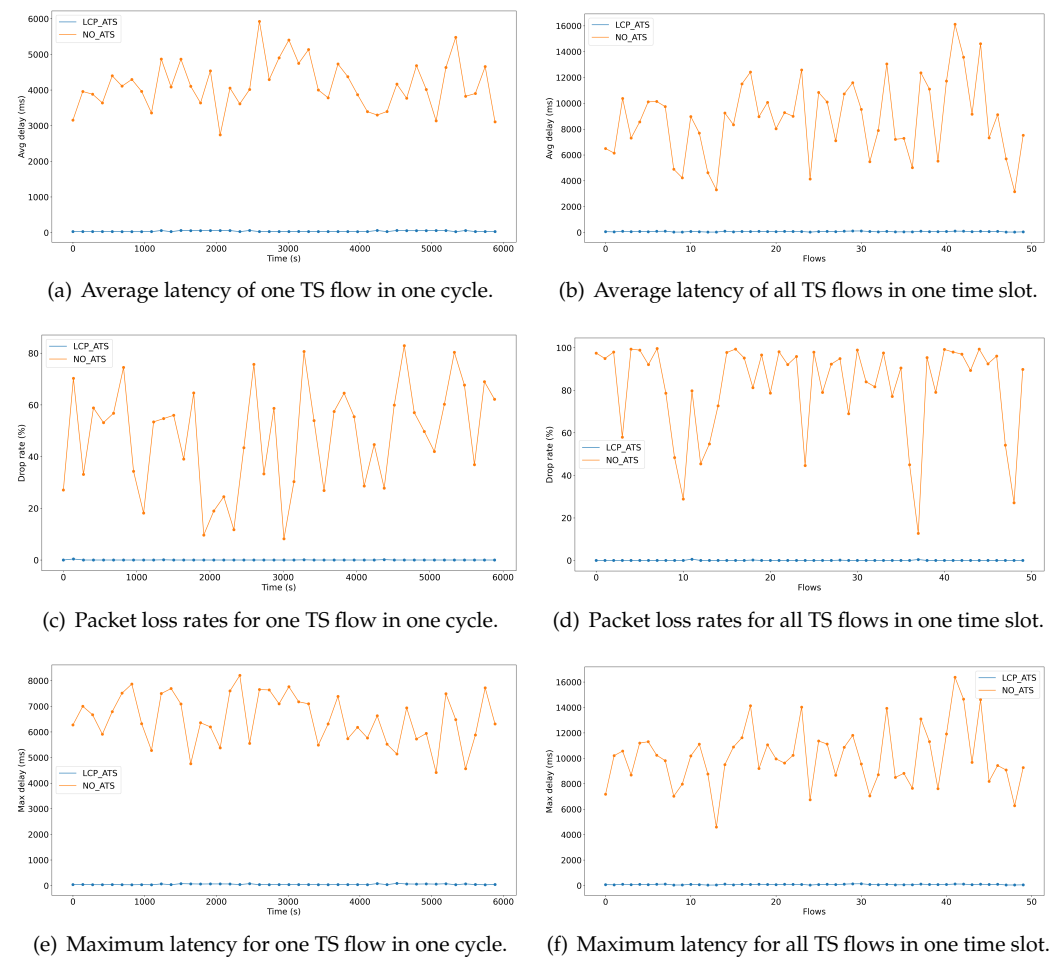


**Figure 9.** Link load with TS flow in one cycle.

In this experiment, we observe the similar result in the other time slot; for simplicity, we only show the result of one slot here.

#### 6.4. ATS Function Comparison

By enabling and disabling the ATS function, we can demonstrate the impact of ATS on network performance. Figure 10 shows the results of multiple experiments, clearly indicating that the function of ATS significantly reduces the delay and packet loss probability of TS traffic, which leads to a more stable network operation.



**Figure 10.** ATS function evaluation.

### 6.5. Delay and Packet Loss Comparison

The delay and packet loss probability of the three algorithms are compared. Tables 4–6 show the maximum delay, average delay, and packet loss probability for each flow in a single time slot. In Tables 4 and 5, the delay requirement (upper delay) of each TS flow is also shown. It can be observed that all the algorithms have cases where the maximum delay exceeds the delay threshold, while both the maximum and average delay of TGM for all the flows are higher than the upper delay. This is because the maximum delay is influenced by system jitter, which is uncontrollable. For the average delay shown in Table 5, DT-DVTR demonstrates a significantly lower delay compared to LCP. The reason is that TGM always selects the shortest path, which may result in congestion on certain links and cause high delay for some flows. On the other hand, LCP considers selecting the path with the low link utilization when the delay requirement is satisfied, resulting in a relatively high delay but still meeting the delay requirements of TS flows and exhibiting a better load balancing performance. Table 6 displays the number of flows with packet loss in each algorithm, where only TGM, with its preference for the shortest path, experiences congestion and packet loss.

**Table 4.** Maximum delay in one time slot.

Flow	Maximum Delay				
	LCP	DT-DVTR	TGM	Lshade	Upper Delay
1	68.068	68.595	65.451	63.800	60.040
2	52.394	45.882	47.577	45.998	46.790
3	99.825	94.964	100.391	90.018	101.470
4	67.611	58.752	66.937	63.055	59.870
5	87.470	85.249	87.692	84.193	87.860
6	66.774	59.156	63.079	61.711	60.300
7	98.847	90.580	99.687	90.105	102.190
8	110.993	103.022	112.567	96.113	115.530
9	37.606	32.767	35.958	36.397	59.020
10	40.423	33.275	43.799	31.900	32.129
Average	77.402	70.756	516.260	70.601	76.685

**Table 5.** Average delay in one time slot.

Flow	Average Delay				
	LCP	DT-DVTR	TGM	Lshade	Upper Delay
1	57.140	56.860	56.655	53.800	60.040
2	40.652	40.195	40.100	40.998	46.790
3	86.420	84.906	87.026	81.018	101.470
4	57.159	53.203	54.221	54.055	59.870
5	77.440	76.941	77.376	77.193	87.860
6	54.162	52.943	53.513	53.711	60.300
7	86.633	83.476	88.672	83.105	102.190
8	95.168	93.406	100.206	90.113	115.530
9	28.101	24.790	25.166	26.398	59.020
10	28.823	28.308	28.445	26.900	32.129
Average	64.640	62.018	493.689	63.261	76.685

**Table 6.** Packet loss in one time slot.

	LCP	DT-DVTR	TGM	Lshade
The number of flows that have experienced packet loss	0	0	3	0

Figures 11–16 shows the maximum delay, average delay, and packet loss probability for a high-priority and a low-priority TS flow. It can be observed that TGM fails to meet the delay requirement and has high packet loss of the low-priority TS flow. On the other hand, both LCP and DT-DVTR guarantee the delay requirement and low packet loss for the low-priority TS flow. Figure 11 indicates that due to jitters in the experiment testbed, all three algorithms still have maximum delays exceeding the delay thresholds. However, Figure 12 demonstrates that all three algorithms ensure that the high-priority TS flow experiences delays below the delay threshold, and Figure 13 shows a very low packet loss probability. Overall, all three algorithms guarantee the delay and packet loss of the

high-priority TS flow, but LCP and DT-DVTR perform better than TGM in terms of the low-priority TS flow.

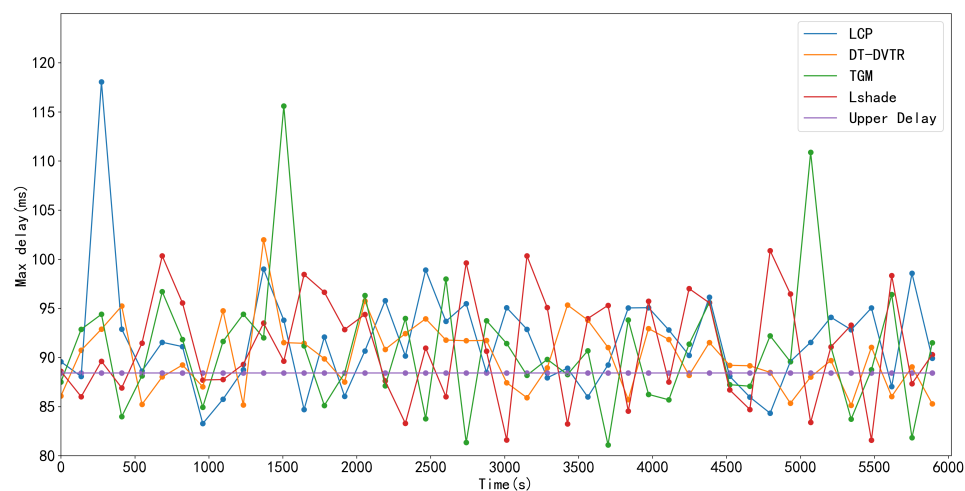


Figure 11. Maximum delay for high-priority TS flows.

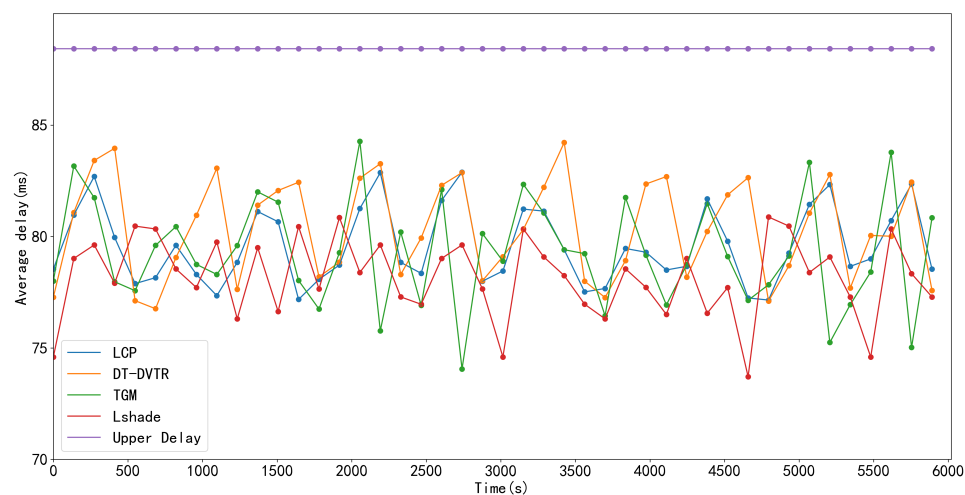


Figure 12. Average delay for high-priority TS flows.

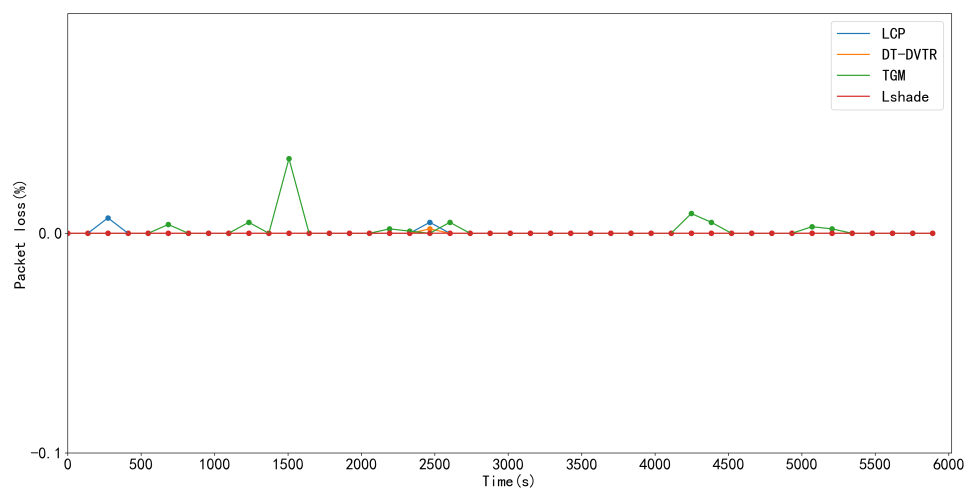


Figure 13. Packet loss rates for high-priority TS flows.

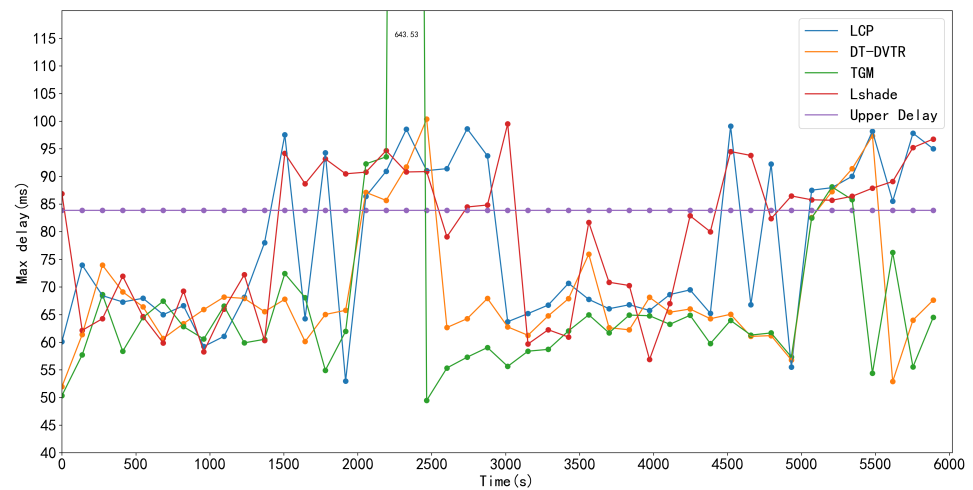


Figure 14. Maximum delay for low-priority TS flows.

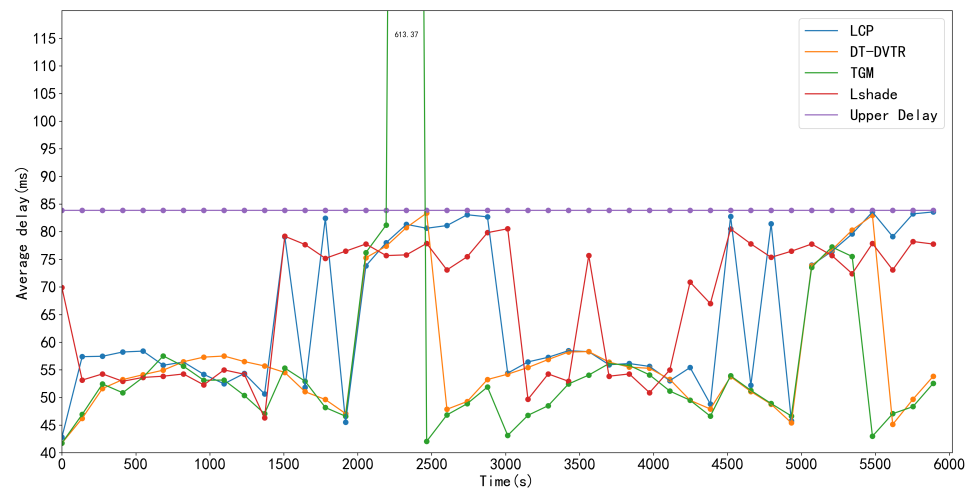


Figure 15. Average delay for low-priority TS flows.

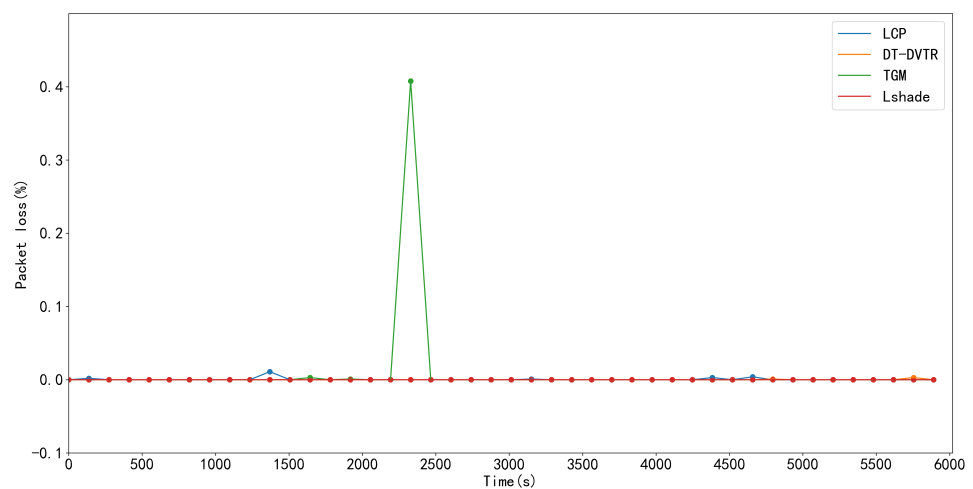


Figure 16. Packet loss rates for low-priority TS flows.

## 7. Conclusions

This paper has addressed the routing problem of time-sensitive traffic flow in satellite networks, which is challenging due to the variability of satellite network topology and the specific delay requirements of TS flows, such as delay, and load balancing. We have provided a link-path traffic-scheduling model that aims to optimize several objectives,



including minimizing the number of path handovers, minimizing link utilization, and ensuring deterministic delay by introducing delay constraints.

We have proposed an approximate algorithm called longest continuous path (LCP). This algorithm selects the longest continuous path within each time slot and then chooses the path with the minimum link utilization that satisfies the latency requirement. By constructing an auxiliary graph, LCP determines a routing solution that also reduces the number of path handovers. LCP takes into account the specific requirements of TS flows and the complex topology of satellite networks, which outperforms in terms of a smaller number of path handovers, a commendable load balancing performance, and fulfilling the latency requirements.

**Author Contributions:** Conceptualization, F.D. and Y.Z.; methodology, F.D. and Y.Z.; software, G.L.; validation, F.D., G.L., and H.Y.; formal analysis, Y.Z.; investigation, G.L.; resources, F.D. and Y.Z.; data curation, G.L.; writing—original draft preparation, G.L. and H.Y.; writing—review and editing, G.L. and F.D.; supervision, F.D.; and project administration, C.S. and Y.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research work was supported by National Natural Science Foundation of China (Grant No. 92067206).

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

LEO	Low Earth orbit
6G	Sixth generation
TSN	Time-sensitive networking
SDN	Software-defined networking
ILP	Integer linear programming
DRA	Distributed routing algorithm
LCP	Longest continuous path
STK	Satellite tool kit
OVS	OpenVSwitch
DE	Differential evolution
ATS	Asynchronous traffic shaper
ONOS	Open-network operating system

## References

1. Vitturi, S.; Zunino, C.; Sauter, T. Industrial communication systems and their future challenges: Next-generation Ethernet, IIoT, and 5G. *Proc. IEEE* **2019**, *107*, 944–961. [\[CrossRef\]](#)
2. *IEEE Std 802.1Qcr-2020*; IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 34: Asynchronous Traffic Shaping. IEEE: Piscataway, NJ, USA, 2020; pp. 1–151.
3. Mendiola, A.; Astorga, J.; Jacob, E.; Higuero, M. A Survey on the Contributions of Software-Defined Networking to Traffic Engineering. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 918–953. [\[CrossRef\]](#)
4. Akyildiz, I.F.; Lee, A.; Wang, P.; Luo, M.; Chou, W. A road map for traffic engineering in SDN-OpenFlow networks. *Comput. Netw.* **2014**, *71*, 1–30. [\[CrossRef\]](#)
5. Finn, N. Introduction to time-sensitive networking. *IEEE Commun. Stand. Mag.* **2018**, *2*, 22–28. [\[CrossRef\]](#)
6. Zhang, J.; Chen, L.; Wang, T.; Wang, X. Analysis of TSN for industrial automation based on network calculus. In Proceedings of the 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, 10–13 September 2019.
7. Wang, Z.C.; Dai, J.L.; Zhong, L. Delay optimization strategy based on a periodic traffic in time-sensitive networking. *J. Phys. Conf. Ser.* **2021**, *1920*, 012093. [\[CrossRef\]](#)
8. Durr, F.; Nayak, N.G. No-wait packet scheduling for IEEE time-sensitive networks. In Proceedings of the 24th International Conference on Real-Time Networks and Systems, Brest, France, 19–21 October 2016; pp. 203–212.

9. Hellmanns, D.; Glavackij, A.; Falk, J.; Hummen, R.; Kehrer, S.; Dürr, F. Scaling TSN Scheduling for Factory Automation Networks. In Proceedings of the 16th IEEE International Conference on Factory Communication Systems (WFCS), Porto, Portugal, 27–29 April 2020; pp. 1–8.
10. Craciunas, S.S.; Oliver, R.S.; Chmelić, M.; Steiner, W. Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks. In Proceedings of the 24th International Conference on Real-Time Networks and Systems, Brest, France, 19–21 October 2016; pp. 183–192.
11. Romanov, A.M.; Gringoli, F.; Sikora, A. A precise synchronization method for future wireless TSN networks. *IEEE Trans. Ind. Inform.* **2020**, *17*, 3682–3692. [\[CrossRef\]](#)
12. Cheng, N.; Lyu, F.; Quan, W.; Zhou, C.; He, H.; Shi, W.; Shen, X. Space/Aerial-assisted computing offloading for IoT applications: A learning-based approach. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1117–29. [\[CrossRef\]](#)
13. Xie, R.; Tang, Q.; Wang, Q.; Liu, X.; Yu, F.R.; Huang, T. Satellite-terrestrial integrated edge computing networks: Architecture, challenges, and open issues. *IEEE Netw.* **2020**, *34*, 224–231. [\[CrossRef\]](#)
14. Valente, F.; Eramo, V.; Lavacca, F.G. Optimal bandwidth and computing resource allocation in low Earth orbit satellite constellation for Earth observation applications. *Comput. Netw.* **2023**, *232*, 109849–109866. [\[CrossRef\]](#)
15. Alagoz, F.; Korcak, O.; Jamalipour, A. Exploring the routing strategies in next-generation satellite networks. *IEEE Wirel. Commun.* **2007**, *14*, 79–88. [\[CrossRef\]](#)
16. Korçak, Ö.; Alagöz, F. Virtual topology dynamics and handover mechanisms in Earth-fixed leo satellite systems. In Proceedings of the International Conference on Computer Communications and Networks, San Francisco, CA, USA, 3–6 August 2009; Volume 53, pp. 1497–1511.
17. Werner, M. A dynamic routing concept for ATM-based satellite personal communication networks. *IEEE J. Sel. Areas Commun.* **1997**, *15*, 1636–1648. [\[CrossRef\]](#)
18. Hong, S.C. FSA-based link assignment and routing in low-Earth orbit satellite networks. *IEEE Trans. Veh. Technol.* **1998**, *47*, 1037–1048. [\[CrossRef\]](#)
19. Ekici, E.; Akyildiz, I.F.; Bender, M.D. A distributed routing algorithm for datagram traffic in LEO satellite networks. *IEEE/ACM Trans. Netw.* **2001**, *9*, 137–147. [\[CrossRef\]](#)
20. Liu, X.; Yan, X.; Jiang, Z.; Li, C.; Yang, Y. A Low-Complexity Routing Algorithm Based on Load Balancing for LEO Satellite Networks. In Proceedings of the 2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall), Boston, MA, USA, 6–9 September 2015; pp. 1–5.
21. Kim, Y.S.; Bae, Y.; Kim, Y.; Park, C.H. Traffic load balancing in low Earth orbit satellite networks. In Proceedings of the 7th International Conference on Computer Communications and Networks, Lafayette, LA, USA, 12–15 October 1998; pp. 191–195.
22. Franck, L.; Maral, G. Static and adaptive routing in ISL networks from a constellation perspective. *Int. J. Satell.* **2002**, *20*, 455–475. [\[CrossRef\]](#)
23. Taleb, T.; Mashimo, D.; Jamalipour, A.; Hashimoto, K.; Nemoto, Y.; Kato, N. SAT04-3: ELB: An Explicit Load Balancing Routing Protocol for Multi-Hop NGEOSatellite Constellations. In Proceedings of the IEEE Globecom 2006, San Francisco, CA, USA, 27 November–1 December 2006; pp. 1–5.
24. Rao, Y.; Wang, R.-C. Agent-based load balancing routing for LEO satellite networks. *Comput. Netw.* **2010**, *54*, 3187–3195. [\[CrossRef\]](#)
25. Yuan, J.; Chen, P.; Liu, Q.; Li, H. A load-balanced on-demand routing for LEO satellite networks. *J. Netw. Comput. Appl.* **2014**, *9*, 3305. [\[CrossRef\]](#)
26. Pioro, M.; Medhi, D. *Routing, Flow, and Capacity Design in Communication and Computer Networks*; Morgan Kaufmann: Burlington, MA, USA, 2004.
27. Wang, X.; Zhang, Q.; Ren, J.; Xu, S.; Wang, S.; Yu, S. Toward efficient parallel routing optimization for large-scale SDN networks using GPGPU. *J. Netw. Comput. Appl.* **2018**, *113*, 1–13. [\[CrossRef\]](#)
28. Ansys STK. Software for Digital Mission Engineering and Systems Analysis. Available online: <https://www.ansys.com/products/missions/ansys-stk> (accessed on 1 April 2022).
29. Tanabe, R.; Fukunaga, A.S. Improving the search performance of SHADE using linear population size reduction. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China, 6–11 July 2014; pp. 1658–1665.
30. Gounder, V.V.; Prakash, R.; Abu-Amara, H. Routing in LEO-based satellite networks. In Proceedings of the 1999 IEEE Emerging Technologies Symposium. Wireless Communications and Systems (IEEE Cat. No. 99EX297), Richardson, TX, USA, 12–13 April 1999; pp. 22.1–22.6.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.