Achieving Resilient and Performance-Guaranteed Routing in Space-Terrestrial Integrated Networks

Zeqi Lai[‡*], Hewu Li[‡*¶], Yikun Wang[‡], Qian Wu[‡*], Yangtao Deng[‡], Jun Liu[‡*], Yuanjie Li[‡*], Jianping Wu[‡*]
[‡]*Institute for Network Sciences and Cyberspace*, Tsinghua University, [*]*Zhongguancun Laboratory*, Beijing, China

IEEE INFOCOM 2023 - IEEE Conference on Computer Communications | 979-8-3503-3414-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/INFOCOM53939.2023.10229104

***Abstract*—Satellite routers in emerging space-terrestrial integrated networks (STINs) are operated in a failure-prone, intermittent and resource-constrained space environment, making it very critical but challenging to cope with various network failures effectively. Existing resilient routing approaches either suffer from continuous re-convergences with low network reachability, or involve prohibitive pre-computation and storage overhead due to the huge amount of possible failure scenarios in STINs.**

This paper presents StarCure, a novel resilient routing mechanism for futuristic STINs. StarCure aims at achieving fast and efficient routing restoration, while maintaining the low-latency, high-bandwidth service capabilities in failure-prone space environments. First, StarCure incorporates a new network model, called the *topology-stabilizing model (TSM)* to eliminate topological uncertainty by converting the topology variations caused by various failures to traffic variations. Second, StarCure adopts an adaptive hybrid routing scheme, collaboratively combining a constraint optimizer to efficiently handle predictable failures, together with a location-guided protection routing strategy to quickly deal with unexpected failures. Extensive evaluations driven by realistic constellation information show that, StarCure can protect routing against various failures, achieving close-to-100% reachability and better performance restoration with acceptable system overhead, as compared to other existing resilience solutions.

I. Introduction

Thanks to emerging innovations in the aerospace industry, in the past few years we have witnessed the rapid evolution and deployment of satellite Internet constellations (SIC) in low earth orbits (LEO), such as SpaceX's Starlink [1] and Amazon Project Kuiper [2]. Such broadband constellations facilitate the construction of space-terrestrial integrated networks (STINs), regarded as an important direction of the next generation of Internet, promising to realize pervasive, high-throughput, low-latency network services for terrestrial customers [3], [4], [5].

Towards the goals above, *network routing* plays a critical role in the service quality of STINs, since it not only determines the reachability between any two communication ends in the network, but also affects the achievable network performance perceived by customers. Ideally, a STIN routing mechanism is expected to simultaneously: (i) maintain high network reachability for geo-distributed customers during any period of operation; and (ii) provide low latency and high throughput paths for delivering various Internet traffic over the STIN.

However, due to a series of unique characteristics of LEO satellites, achieving highly available and performant routing is still challenging in STINs. First, the backbone network of a STIN is exposed in outer space, suffering from risks such as debris collisions [6], [7] and radiation hazards [8] *etc.* Second, LEO satellites are *constantly* moving at a high velocity in their orbits, and such continuous dynamics can lead to frequent inter-visibility changes and link disruptions. Finally, emerging mega-constellations are constructed by shorter-lifespan small satellites, which significantly reduce the production cost, but are inherently more brittle and prone to failures [9]. All factors above can result in node or link failures in a STIN. *How should STIN service providers cope with various network failures effectively in such a failure-prone, intermittent environment?*

Today's widely deployed Internet routing protocols, such as OSPF and ISIS, deal with failures in a *reactive manner*, relying on global link state advertisements to discover network topology changes and compute correct routing tables. However, reactive solutions have to experience a convergence period jeopardizing routing stability. Since network failures occur frequently and constantly in STINs, directly applying such reactive solutions could lead to incessant routing convergence, resulting in very poor network reachability in STINs.

As alternative solutions, many existing works propose to tackle network failures in a *proactive manner* and accomplish convergence-free routing [10], [11], [12], [13], [14], [15], [16]. The underlying idea of these efforts is to *pre-compute* the correct routing tables for possible failure scenarios in advance, and then perform fast re-routing once a real failure happens. However, it is also difficult to directly apply such proactive methods to a STIN environment for two main reasons. First, the combination of huge constellation scale, constant LEO dynamics and the complex error-prone environment *jointly* create a significantly large amount of possible failure scenarios and topology variations. Pre-computing decisions for all these possible scenarios can involve prohibitive computation overhead. Second, satellites are resource-constrained, and storing too many backup routing tables for all failures at each node can easily overwhelm the storage system of satellites.

In this paper, we present StarCure, a novel resilient routing mechanism for emerging STINs. StarCure targets at achieving fast and efficient routing restoration while maintaining the low-latency, high-bandwidth service capabilities for STINs in error-prone, constantly-dynamic, resource-constrained space environments. Specifically, StarCure incorporates two key techniques to cope with various network failures effectively.

First, StarCure adopts a new network model, called topology-stabilizing model (TSM) (§IV-B) to convert the *topology variations* under various failure scenarios to *traffic variations*, and formulate the resilient space routing problem

¶ Hewu Li is the corresponding author.

upon a stable network topology. TSM exploits two important insights obtained from STINs: (i) a long-duration traffic demand affected by a predictable failure can be modeled as a series of consecutive demands issued by different source-destination pairs; and (ii) an unexpected link failure can be viewed as a burst traffic fully exhausting that link. Therefore, with TSM, StarCure converts the original resilient routing problem which requires pre-calculating routing decisions for *(nearly)* an infinite number of topology variations, to a dynamic routing scheduling problem upon a stable logical network topology.

Second, to solve the above dynamic routing problem in an efficient and practical manner, StarCure incorporates an adaptive hybrid routing scheme (§IV-C). While TSM enables us to solve the resilient routing problem upon a stable topology, there still remains two practical issues that have to be addressed. On one hand, the conversion by TSM significantly increases the traffic variations and makes it challenging to use standard linear programming to solve the problem efficiently. On the other hand, iterating all possible traffic demands generated by unexpected failures in advance for pre-calculation can still involve significant computation overhead. To overcome these practical problems, our hybrid routing scheme combines a dynamic-tolerant *basic routing* to efficiently adapt and handle predictable failure with guaranteed network performance, together with a *location-guided protection routing* to quickly deal with unexpected failures and maintain routing continuity.

To validate the feasibility and effectiveness of StarCure, we implement a StarCure prototype, and build a hardware-in-the-loop testbed which can create a large-scale simulated STIN environment to load real routing software and network traffic for experimentation (§V). Extensive evaluations based on realistic constellation information, traffic pattern and various failure events demonstrate that, as compared to other resilience solutions, StarCure can protect routing against both common, predictable failures and rare, unexpected failures for different constellation topologies, achieving close-to-100% network reachability and better performance after the restoration.

Summarily, the contributions of this paper can be concluded as follows: (i) we formulate the resilient and performant space routing problem, and quantitatively highlight the technical challenges caused by the combination of LEO dynamics and failure uncertainty (§III); (ii) we present StarCure, a novel routing mechanism that incorporates TSM and an adaptive hybrid routing scheme to achieve resilient and performance-guaranteed routing in failure-prone STINs (§IV); (iii) we implement a StarCure prototype (§V) and conduct extensive evaluations driven by realistic constellation information to demonstrate the feasibility and effectiveness of our solutions (§VI).

II. Preliminaries for Futuristic Space-Terrestrial Integrated Networks (STINs)

A. STINs Quick Primer

Figure 1 illustrates a typical architecture of STINs in brief, which integrates two major components to provide Internet service from space for terrestrial users: (i) a *space backbone network*, which consists of hundreds to thousands of LEO

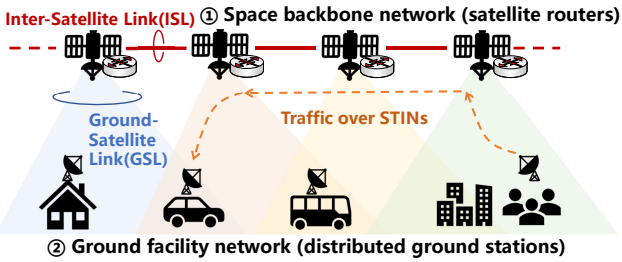


Fig. 1: Space-terrestrial integrated networks (STINs) in brief.

broadband satellites (*i.e.,* satellite routers). These satellites can be equipped with high-speed ground-satellite links (GSL, *e.g.,* Ka/Ku/V-band radio links) and inter-satellite links (ISL, *e.g.,* laser communication links [17]) to construct a high-capacity backbone network to forward data in space; (ii) a *ground facility network*, including a large number of geo-distributed ground stations (and satellite terminals) to enable terrestrial users and content providers to access the space backbone. The entire STIN runs certain space routing mechanisms (*e.g.,* [3], [14], [18]) to forward network traffic and establish end-to-end communications. Collectively, integrating wide-coverage satellite constellation and high-speed communication links, emerging STINs promise to provide pervasive, high-throughput and low-latency Internet services globally [4], [19].

B. Failure-Prone STIN Environments

Unlike conventional terrestrial networks where the backbone is deployed in a sealed, protected circumstance, the space backbone of a STIN is exposed in a public, uncontrollable environment. Network failures, including node and link failures, are prone to happen due to a series of unique characteristics.

LEO dynamics. LEO satellites fly in low orbital altitude (*e.g.,* 500-1200km [1], [2]) to enable low propagation latency for space-ground communication. These satellites move at a high orbital velocity relative to the earth surface. Due to the LEO dynamics, ground-satellite links in a STIN can experience frequent disruptions and re-associations, resulting in frequent network-wide topology fluctuations. A recent investigation [5] has quantitatively shown that the average space-ground link churn interval could be as low as tens of seconds in Starlink.

Environmental risks in complex outer space. Satellites working in the outer space suffer from a number of environmental risks such as debris collision and radiation hazard *etc*. For example, Kessler Syndrome [6] is a phenomenon in which the amount of junk in orbit around earth reaches a threshold where it creates more and more space debris, causing serious failures for satellites. On February 10, 2009, an inactive Russian communications satellite, collided with an active commercial communication satellite operated by Iridium [7]. More recently, a geomagnetic storm doomed 40 Starlink Internet satellites [8].

Small satellite vulnerability. Emerging STINs are built upon small satellites. They involve much lower cost compared to traditional monolithic satellites [20], require shorter manufacturing period, and can use available commercial-off-the-shelf (COTS) technologies to quickly build their on-board systems. On one hand, such a *"build it as cheap as possible"* principle indeed

accelerates the construction and deployment of STINs. But on the other hand, due to the reduced cost, small satellite systems are more vulnerable to failures [9], and typically have a shorter lifespan and weaker radiation resistance. For example, after its first launch in 2019, SpaceX's Starlink has already launched over 3,300 small satellites to space as of the date of Decemeber, 2022, but 353 ($\sim$11%) of the deployed satellites have become decaying or deorbited right now [21].

In a nutshell, STINs are operated an error-prone, constantly-dynamic environments. All factors above can lead to network failures. Note that failures caused by predictable LEO dynamics are common, frequent and transient. Other failures due to unexpected factors such as collisions are rare but can lead to permanent outright errors. We denote the above two classes of failures as predictable and unexpected failures respectively.

III. Understanding The Problem

A. Problem Formulation

Network topology. A STIN contains a space backbone and a number of distributed ground stations. Let $\mathcal{S} = \{s_1, s_2, ..., s_P\}$ denote the set of all satellites in the STIN, with $|\mathcal{S}| = P$ satellites in total. Assume that there are Q ground stations which connect to the space backbone for ground-space communication, and $\mathcal{U} = \{u_1, u_2, ..., u_Q\}$ denotes the set of all ground stations. $\mathcal{V} = \mathcal{S} \cup \mathcal{U}$ denotes the set of all STIN nodes.

In addition, LEO satellites inter-connect to each other via inter-satellite links (ISL), and connect to ground stations via ground-satellite links (GSL). A laser ISL enables point-to-point communication for two broadband satellites at the same time, while a radio GSL allows a satellite to simultaneously connect to many ground stations sharing the entire GSL capacity. Let (i, j) denote an available link in the STIN. In practice, both ISLs and GSLs are bidirectional, and let $\mathcal{L}$ denote the set of all available links. Therefore, in a failure-free scenario, a STIN can be formulated as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$.

Network failures. Essentially, a network failure (*e.g.,* a node or a link failure) causes a topology variation, and consequently affects the network reachability and performance of STINs. Since a node failure can be presented as all its related links becoming unavailable, we focus on the link failures in our formulation. Assume time is slotted, and denote the operation period as $\mathcal{T} = \{t_1, t_2, t_3, ...\}$. A failure event k that occurs in slot t (*e.g.,* a certain set of ISLs/GSLs become unavailable) triggers a network topology variation, and can be described by a function $\phi_t^k : \mathcal{L}_t \rightarrow \mathcal{L}_{t+1}$. In other words, the failure event k changes the available link set from $\mathcal{L}_t$ in slot t to $\mathcal{L}_{t+1}$ in slot $t+1$. We use $\Phi = \{\phi_t^k | t \in \mathcal{T}\}$ to represent the set of all failure events. Considering that those topology variations are caused by continuous network failures, we extend the above network graph in temporal dimension, and use $\mathcal{G}_{all} = \{\mathcal{G}_t | t \in \mathcal{T}\}$ to present all time-varying topology variations generated by Φ. $\mathcal{G}_t = (\mathcal{V}, \mathcal{L}_t)$ describes the network graph in slot t.

Traffic demands. STINs can carry Internet traffic for terrestrial users. Let $f_{ab} = \{st_{ab}^f, et_{ab}^f, d_{ab}^f\}$ denote a traffic demand f from node a to b. Specifically, f_{ab} starts at time slot st_{ab}^f, ends at et_{ab}^f, and its bandwidth requirement is d_{ab}^f. We define $\mathcal{F} = \{f_{ab} | a, b \in \mathcal{V}\}$ as the set of all traffic demands. Let b_{ij} denote the capacity of link (i, j) in the $i \rightarrow j$ direction. There are many existing methods for network operators to estimate traffic matrices $\mathcal{F}$ in their operational networks [22], [23].

Routing scheme. Let binary variables $\tau_{ab}(i, j, t)$ denote whether the traffic demand f_{ab} from a to b is carried by link (i, j). In particular, $\tau_{ab}(i, j, t) = 1$ indicates that traffic from a to b goes through the link (i, j) in slot t. Therefore, an available routing for traffic demand f_{ab}, denoted as r_{ab}^t in slot t, can be described as a set of $\tau_{ab}(i, j, t)$, *i.e.,* $r_{ab}^t = \{\tau_{ab}(i, j, t)\}$, satisfying the following properties, where $\forall i, j, \gamma \in \mathcal{V}, \forall t \in \mathcal{T}$:

$$\sum_{w \in \mathcal{V}} \tau_{\gamma w}(i, j, t) - \sum_{v \in \mathcal{V}} \tau_{v\gamma}(i, j, t) = \begin{cases} 1 & \gamma = a \\ -1 & \gamma = b \\ 0 & otherwise \end{cases}, \quad (1)$$

$$\sum_{a,b} \tau_{ab}(i, j, t) \cdot d_{ab}^f \leq b_{ij}, \forall a, b \in \mathcal{V}. \quad (2)$$

Eq.(1) indicates flow conservation at any intermediate node, and guarantees that there is an available path from source a to destination b. Eq.(2) ensures that all traffic carried by a certain link (i, j) will not exceed its capacity b_{ij}. We further denote $\mathcal{R}_t = \{r_{ab}^t | a, b \in \mathcal{V}, t \in \mathcal{T}\}$ as an available routing scheme in slot t, which includes all available routes for traffic demand $\mathcal{F}$ under the network graph $\mathcal{G}_t$.

Performance guarantees. Note that in an operational STIN, in addition to the basic constraints described above, STIN operators may also require other performance guarantees. We consider two performance metrics critical for many routing and traffic engineering systems: (i) maximum link utilization (MLU) among all links during the operation period, which can be formulated as:

$$MLU = \textbf{max} \frac{\sum_{a,b \in \mathcal{V}} d_{ab}^f \cdot \tau_{ab}(i, j, t)}{b_{ij}}, \forall t \in \mathcal{T}, \forall (i, j) \in \mathcal{L}_t, \quad (3)$$

and (ii) latency requirement $\mathcal{D}_{ab}$, which indicates that the one-way delay of traffic demand f_{ab} is expected to be lower than $\mathcal{D}_{ab}$. In particular, $\mathcal{D}_{ab}$ can be set to $\alpha * \mathcal{D}_{ab}^{sp}$, where $\mathcal{D}_{ab}^{sp}$ is the latency of the shortest path between a and b, and coefficient $\alpha \geq 1$. Ideally, a resilient routing mechanism is expected to quickly react to failures and find an available congestion-free routing for all traffic demands $\mathcal{F}$ (*i.e.,* satisfying Eq.(1) and Eq.(2)), with minimum MLU (*i.e.,* creating more headroom to avoid congestion in the presence of unexpected traffic shifts and outright failure) and bounded latency.

[P1]: resilient space routing problem (RSRP). With the definitions above, we thus formulate RSRP as follows:

$$\textbf{[P1] Objective: min}_{\forall (i,j) \in \mathcal{V}, \forall t \in \mathcal{T}} MLU, \quad (4)$$

$$\textbf{subject to:} \sum_{\forall (i,j) \in \mathcal{L}_t} \tau_{ab}(i, j, t) \cdot l_{ij} \leq \mathcal{D}_{ab}, \forall f_{ab} \in \mathcal{F}, \quad (5)$$

$$\sum_{a,b} \tau_{ab}(i, j, t) \cdot d_{ab}^f \leq MLU, \forall a, b \in \mathcal{V}, \textbf{ and } \text{Eq.}(1), \text{Eq.}(2), \quad (6)$$

where l_{ij} is the latency of link (i,j), $\sum_{\forall(i,j)\in\mathcal{L}_t}\tau_{ab}(i,j,t)\cdot l_{ij}$ is the path one-way latency for traffic f_{ab} in slot $t \in [s^f_{ab}, e^f_{ab})$. Assume there is a sequence of network failures Φ in the STIN, for each failure event $\phi^k_{t-1} \in \Phi$ that changes the network topology to $\mathcal{G}_t$ in slot t, the routing system needs to deal with the failure by *quickly* solving the above integer linear programming problem (ILP) upon $\mathcal{G}_t$ and computing an available routing $\mathcal{R}_t$ presented by variables $\tau_{ab}(i,j,t)$.

B. Limitations of Prior Resilience Solutions

The problem formulation above reveals two fundamental stages for routing restoration: (i) failure discovery, through which each node obtains the correct network topology $\mathcal{G}_t$ when a failure event happens; and (ii) routing re-calculation, by which each node correctly generates and applies new routing tables based on the current topology. The network community has a long history studying the resilient routing techniques, and existing efforts can be classified into two main categories, depending on how they perform these two stages.

Reactive routing restoration. Existing widely deployed Internet routing protocols, such as OSPF [24] and ISIS [25], discover failures and restore routing in a *reactive manner*. In particular, each distributed node periodically exchanges link state to its neighbors to get the global network topology. If a failure happens, they perform global message exchanges and routing re-calculation. However, reactive restoration solutions inevitably experience a *routing re-convergence* period, after the failure has been detected, and before all routers learn about such change. During this period, the routing state might be inconsistent. Note that failures such as GSL disruptions caused by LEO dynamics are common and frequent in a STIN environment. Thus, existing reactive methods may suffer from frequent re-convergence (*e.g.,* in every tens of seconds) and result in routing instability and poor network reachability (*e.g.,* only $\sim$60% in some situations as we will show in §VI).

Proactive routing restoration. To alleviate the impact of routing re-convergence, many existing approaches propose *proactive routing restoration* strategies [26], [11], [12], [13], [14], [15], [16]. These proactive solutions estimate the possible failure scenarios, and pre-compute the backup routing offline. Once a failure occurs, the affected router searches the corresponding backup routing for current scenario, and quickly applies it in the online stage to accomplish fast restoration.

However, the effectiveness of existing pre-computation-based proactive solutions is inherently limited in large-scale, high-dynamic STINs. Although the failures caused by periodical satellite movements are predictable, all possible topology variations are *jointly* generated by the combination of both predictable and unexpected failures. To deal with the failure uncertainty, a proactive resilience solution has to pre-compute backup routes for all failure scenarios. In addition, due to the large constellation size and failure-prone environment, there may be too many possible failure scenarios that need to pre-compute. For example, assume that the predictable LEO dynamics generate $\mathcal{M}$ possible failure-free network snapshots of a STIN, and each snapshot contains $|\mathcal{L}_m|$ links. Assume the STIN needs to cope with up to $\mathcal{N}$ unexpected link failures. Combining predictable and unexpected failures, there will be $\sum_{m=1}^{M}\sum_{p=1}^{\mathcal{N}}\binom{|\mathcal{L}_m|}{p}$ failure scenarios in total. Taking SpaceX's Starlink as an example. Its $|\mathcal{L}_m|$ is more than 3000 and the topology regularly changes in tens of seconds. Resisting against an unexpected solar storm [8] which doomed 40 satellite at once, requires to pre-compute routing decisions for a significantly large amount of failure scenarios which can easily overwhelm the CPU/memory of the routing system. Moreover, even if we can obtain pre-computed routes for all failures, it is also difficult to cache all backup routing tables in the storage-constrained satellites.

Takeaways. Collectively, due to the combination of LEO dynamics and failure-prone space environment, it is challenging for existing resilient solutions to effectively handle failures in STINs. They either involve frequent re-convergence periods resulting low network reachability, or require significant pre-computation overhead which could be unacceptable in practice. This important fact thus motivates us to design StarCure, a new space routing mechanism that achieves resilience and guaranteed performance in emerging STIN environments.

IV. The StarCure Design

A. Mechanism Overview

Core ideas. The design of StarCure incorporates two core ideas. First, we design a new network model, called topology-stabilizing model (TSM) (§IV-B) to convert the *topology variations* caused by various failures in a STIN, to *traffic variations* upon a stable logic topology. TSM is designed based on a key insight that a long traffic demand disrupted by *predictable failures* can be presented by a sequence of short demands, and *unexpected failures* can be modeled as a set of burst traffic demands fully filling corresponding failed links. Second, to efficiently cope with a large number of highly dynamic traffic demands converted by TSM, we propose an adaptive hybrid routing scheme (§IV-C), integrating a constraint optimizer to route predictable traffic with near-optimal performance, and a location-guided protection routing algorithm to route unexpected traffic with bounded performance.

Work phase. Figure 2 plots an example illustrating how StarCure copes with various failures. In this example, there are two ground stations and seven satellites in the network. The entire topology changes due to two predictable failures in $t1$, $t3$, and one unexpected failure in $t2$. There is a traffic demand from $GS1$ to $GS2$. Specifically, StarCure achieves resilient and performance-guaranteed space routing as follows. First, by invoking TSM, StarCure converts the time-varying STIN physical view to a *stable* logical view, together with a highly-dynamic traffic matrix upon it. Second, to tackle predictable failures which are described by a large number of predictable traffic demands, StarCure leverages a basic routing scheme to pre-compute routing decisions *offline* with guaranteed performance, and configure routing decisions *online* in case of predictable failures (*e.g.,* space-ground handovers). Note that because TSM has eliminated the topological uncertainty, the basic routing here only needs to solve an LP problem on

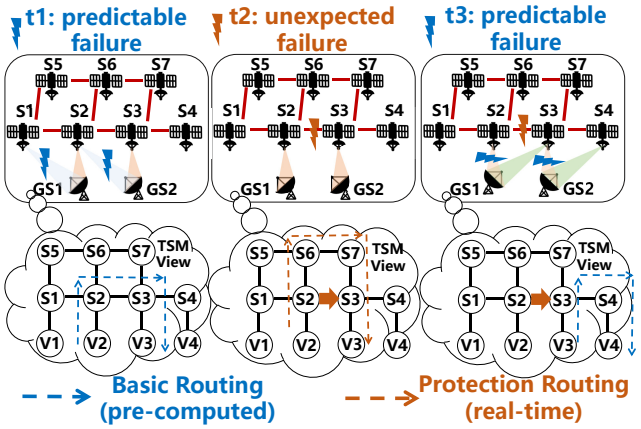


Fig. 2: An example illustrating the work phase of StarCure.

a single topology in the offline stage. Third, if unexpected failures happen (*e.g.,* $t2$ in Figure 2), on one hand, the affected node immediately switches to its protection routing strategy, and fast reroutes affected traffic based on its local information to maintain routing availability. On the other hand, StarCure notifies the unexpected failure to its basic routing strategy. StarCure's basic routing then updates the traffic matrix via adding the burst traffic generated by the unexpected failure, and re-computes routing decisions with performance constraints. Finally, when new routing decisions have been generated by the basic routing, StarCure switches back to the basic routing strategy again (*e.g.,* $t3$ in Figure 2).

B. Topology-Stabilizing Network Model

As the first step to solve the resilient space routing problem, we design a new network model, called *"topology-stabilizing model" (TSM)* to describe a dynamic topology through a stable logical view. TSM incorporates two surjection functions. First, a graph surjection $\psi : \mathcal{G}_T \rightarrow \hat{\mathcal{G}}$ which converts the time-varying $\mathcal{G}_T$ to a stable logical view $\hat{\mathcal{G}}$. Second, a traffic surjection $\omega : \mathcal{F} \rightarrow \hat{\mathcal{F}}$ that describes the traffic demand on the stable $\hat{\mathcal{G}}$.

Stabilizing the network topology. Let $\mathcal{G}_T$ denote a set of sequential network snapshots generated by failure events Φ. As illustrated in Algorithm 1, $\psi(\mathcal{G}_T) = \hat{\mathcal{G}}$ is calculated as follows. First, to create the vertex set $\hat{\mathcal{V}}$ in $\hat{\mathcal{G}}$, for each satellite node $s_i \in \mathcal{S}$ in $\mathcal{G}_T$, TSM creates a mirror in $\hat{\mathcal{V}}$. In addition, TSM creates a *virtual terrestrial node*, denoted as vs_i, for each satellite s_i in $\hat{\mathcal{V}}$ (line 3-4). Each vs_i presents the set of all ground stations that connect to s_i in certain time slots. Thus, assume that there are P satellites in the original $\mathcal{G}_T$, and then $\hat{\mathcal{G}}$ contains $2P$ nodes in total. Second, to build the link set $\hat{\mathcal{L}}$ in $\hat{\mathcal{G}}$, for each s_i and its corresponding vs_i, establish a link (s_i, vs_i) presenting the ground-satellite communication for satellite s_i (line 6-7). The link capacity of (s_i, vs_i) is set equal to the GSL capacity of satellite s. In addition, mirror satellites in $\hat{\mathcal{V}}$ inter-connect to each other following their physical connectivity pattern (line 9). For each ISL in $\mathcal{G}_T$, make a copy of it in $\hat{\mathcal{G}}$, *i.e.,* each mirror of s_i in $\hat{\mathcal{G}}$ connects to two front and back neighbors in the same orbit, and connects to other two left and right neighbors in adjacent orbits. Note that vs_i may map to $\varnothing$ if s_i does not connect to any ground stations in some slots. In these cases where $vs_i = \varnothing$, (s_i, vs_i) still exists, but no traffic will go through it.

Algorithm 1: Topology Stabilization (ψ).

Input : Time-varying physical view $\mathcal{G}_T = (\mathcal{V}, \mathcal{L}_T)$.
Output : Stabilized logic view $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{L}})$.

1 $\hat{\mathcal{V}} \leftarrow \varnothing$, $\hat{\mathcal{L}} \leftarrow \varnothing$ */* (1) initialization. */*;
2 **for** *satellite* $s \in \mathcal{S}$ *in* $\mathcal{V}$ **do**
3 $vs \leftarrow$ `createVN`(s) */* create a virtual node. */*;
4 $\hat{\mathcal{V}}$.`addVertex`(s), $\hat{\mathcal{V}}$.`addVertex`(vs);
5 */* (2) create a link in* $\hat{\mathcal{L}}$ *and set its capacity as the GSL capacity of satellite* s. **/*;
6 $(s, vs) \leftarrow$ `createLink`$(GSL\_capacity_s)$;
7 $\hat{\mathcal{L}}$.`addLink`$((s, vs))$;
8 */* (3) add a copy of each ISL in* $\mathcal{G}_T$ *to* $\hat{\mathcal{G}}$. **/*;
9 $\hat{\mathcal{L}}$.`addLink`$(\forall (i,j) \in \mathcal{L}_T, i, j \in \mathcal{S})$;
10 **Return** $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{L}})$.

After this conversion, $\psi(\mathcal{G}_T) = \hat{\mathcal{G}}$ is a stable graph for two reasons. First, recent mega-constellations like Starlink and Kuiper follow Walker Delta constellation [27] design, in which inclined orbits and satllites are evenly spaced. Satellites in the same orbital shell (*e.g.,* the first-shell 1584 satellites of Starlink) fly at the same altitude with the same velocity and thus the relative positions between adjacent satellites are stable. Second, because vs_i refers to a virtual presentation of the set of ground stations connected to s_i in certain slots, there is always a stable logical link (s_i, vs_i) in $\hat{\mathcal{G}}$, although vs_i maps to different ground stations physically in different time slots.

Converting fluctuating topologies to fluctuating traffic. First, failures caused by predictable, periodical satellite movements can be estimated based on orbital information. For example, space-ground disconnections and reconnections can be typically estimated in advance with the knowledge of the satellite's velocity and orbital position. Thus, TSM splits a long-duration traffic demand affected by such predictable failures into a bunch of sequential short-duration demands. Second, failures due to random, unexpected incidents in the complex space environment (*e.g.,* solar storm or cosmic radiation) can be modeled as burst virtual flows that fully exhaust the link.

Specifically, the traffic surjection $\omega(\mathcal{F}) = \hat{\mathcal{F}}$ is illustrated in Algorithm 2. First, for predictable failures, TSM converts all affected long flows in $\mathcal{F}$ to a bunch of sequential short flows in corresponding $\hat{\mathcal{F}}$. Assume that $f_{ab} \in \mathcal{F}$ is a traffic demand from a to b during the period $[s^f_{ab}, e^f_{ab})$, and the flow of f_{ab} is affected by predictable GSL disruptions. Due to link layer handovers, the logical position of a and b change in $\hat{\mathcal{G}}$ during $[s^f_{ab}, e^f_{ab})$. TSM thus converts f_{ab} to a sequence of short flows between these two pairs of virtual terrestrial nodes in corresponding time slots (line 2-7). Second, for unexpected failures, TSM describes them as a burst traffic demand in the failed link. Assume a link (i, j) suffers from a failure during the period $[t_s, t_e)$. TSM describes a link failure by creating a burst flow $f_{burst} = \{t_s, t_e, b_{ij}\}$ with the traffic demand equal to the link capacity b_{ij} (line 9-11). To ensure that the burst

Algorithm 2: Dynamic Traffic Conversion (ω).

Input : Original traffic matrix $\mathcal{F}$, stabilized graph $\hat{\mathcal{G}}$.
Output : Converted traffic matrix $\hat{\mathcal{F}}$ upon $\hat{\mathcal{G}}$.

$\hat{\mathcal{F}} \leftarrow \varnothing$ /* *(1) initialization.* */;
for *each traffic demand* $f_{ab} \in \mathcal{F}$ **do**
  for *each slot* t *from* s^f_{ab} *to* e^f_{ab} **do**
    /* *(2) find the converted vertex of the original src/dst in* $\hat{\mathcal{G}}$*, and create a new demand.* */;
    $s \leftarrow \texttt{find}(\hat{\mathcal{G}}, a, t)$, $d \leftarrow \texttt{find}(\hat{\mathcal{G}}, b, t)$;
    $f_{sd} \leftarrow \texttt{createDemand}(t, (t+1), d^f_{ab})$;
    $\hat{\mathcal{F}}.\texttt{addDemand}(f_{sd})$;
/* *(3)create virtual demands for unexpected failures.* */;
for *each unexpected failure in* (i, j) *during* $[t_s, t_e)$ **do**
  $f_{burst} \leftarrow \texttt{createDemand}(t_s, t_e, b_{ij})$;
  $\hat{\mathcal{F}}.\texttt{addDemand}(f_{burst})$;
Return $\hat{\mathcal{F}}$. /* *return converted traffic matrix.* */

traffic is routed over link (i, j), we set the latency requirement of f_{burst} equal to that link latency l_{ij}.

Figure 3 shows a concrete example of TSM conversion. Assume that a simplified STIN contains three LEO satellites in the same orbital altitude (S_1, S_2, S_3) and two ground stations (GS_1, GS_2). The space-ground connectivity changes from $(GS_1, S_1), (GS_2, S_2)$ in slot 1 to $(GS_1, S_2), (GS_2, S_3)$ in slot 2 due to the LEO dynamics and link layer handovers. In addition, an unexpected ISL failure occurs in $(S1, S2)$ in slot 2. The entire physical topology accordingly changes from slot 1 (*i.e.,* $[t1, t2)$), to slot 2 (*i.e.,* $[t2, t3)$) due to such failures. Assume there is a traffic demand $f_{GS_1,GS_2} = \{t1, t3, D\}$ from GS_1 to GS_2 during $[t1, t3)$. TSM creates a virtual terrestrial node for each satellite (*i.e.,* V_1, V_2, V_3), and accordingly splits the traffic demand affected by the predictable failure into two sequential traffic $f_{v_1,v_2} = \{t1, t2, D\}$ and $f_{v_2,v_3} = \{t2, t3, D\}$. For the unexpected failure in t_2, TSM generates a burst traffic demand $f_{S_1,S_2} = \{t2, t3, Cap(S1, S2)\}$, where $Cap(S1, S2) = b_{S_1,S_2}$ is the link capacity of $(S1, S2)$.

With TSM, the original resilient space routing problem (**P1**), which requires pre-computing routing decisions for a significantly large number of possible topologies in $\mathcal{G}_T$, is thus converted to a *dynamic routing scheduling problem* (**P2**): given a stable network $\hat{\mathcal{G}}$, a time-varying traffic matrix $\hat{\mathcal{F}}$, finding the available dynamic routing schemes $\mathcal{R}_T$ satisfying the performance constraints formulated in Eq.(1-2), and Eq.(5-6).

TSM eliminates the *topological uncertainty* caused by the combination of LEO dynamics and uncertain failures. However, completely solving **P2** still requires to cope with two additional issues. First, since TSM converts topology dynamics to traffic dynamics, the size of traffic matrix $\hat{\mathcal{F}}$ significantly increases, making it still challenging to directly apply conventional linear programming (LP) solvers to solve the problem. Second, the traffic matrix $\hat{\mathcal{F}}$ still contains unpredictable demands, and thus in practice it is challenging to get the entire $\hat{\mathcal{F}}$ in advance to calculate the optimal solution. We design a new adaptive hybrid routing scheme to solve **P2** effectively and efficiently.

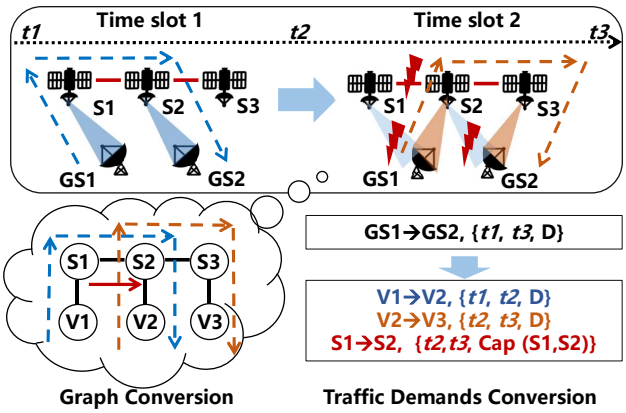


Fig. 3: Converting various failures to dynamic traffic .

C. Adaptive Hybrid Resilient Routing

Our adaptive hybrid resilient routing integrates two strategies to collaboratively handle highly-dynamic traffic due to failures. We split $\hat{\mathcal{F}} = \hat{\mathcal{F}}_{pr} + \hat{\mathcal{F}}_{up}$, where $\hat{\mathcal{F}}_{pr}$ and $\hat{\mathcal{F}}_{up}$ are predictable/unexpected traffic demands respectively. First, StarCure adopts a basic routing strategy which leverages a constraint optimizer to pre-compute (offline) and configure (online) routing decisions for variable traffic $\hat{\mathcal{F}}_{pr}$ caused by common, transit and predictable failures with guaranteed performance. Second, StarCure incorporates a protection routing strategy that exploits location-guided fast rerouting to quickly deal with burst traffic $\hat{\mathcal{F}}_{un}$ caused by unexpected failures locally. Once a rare, unanticipated failure happens, StarCure immediately invokes the protection routing to reroute affected traffic, and simultaneously triggers re-calculation in the basic routing. Finally, once new routing decisions have been computed by the basic routing based on the new traffic matrix, StarCure switches back to the basic routing strategy.

1) StarCure's basic routing strategy: Directly solving P2 with $\hat{\mathcal{F}_{pr}}$ by conventional ILP solvers still has difficulties, since TSM splits long-duration traffic demands into multiple short demands in different time slots. We adopt a constraint optimizer with two optimizations described as follows.

Path filtering. Due to the large size of $\hat{\mathcal{G}}$ and $\hat{\mathcal{F}}_T$, there are too many variables $\tau_{ab}(i, j, t)$ constructing a large solution space to explore. We design a heuristic to shrink the solution space. Our heuristic is based on the insight that in a STIN, to attain the goal of low latency (*i.e.,* Eq (5)), satellites that are too far away from the ground projection between a communication pair $a \to b$ is not expected to carry f_{ab}. We thus estimate the path stretch p^{ij}_{ab} between a and b if the demand f_{ab} is carried by a certain link (i, j), by summing up the great circle distance between a and i, the length of link (i, j), and the great circle distance between b and j. We do not consider link (i, j) to carry traffic demand f_{ab}, if the estimated $p^{ij}_{ab} \geq \beta \cdot \mathcal{D}_{ab}$. By this method, we eliminate many variables of $\tau_{ab}(i, j, t)$.

Merging traffic demands. Converting the original traffic matrix $\mathcal{F}$ into $\hat{\mathcal{F}}$ increases the matrix size and also imposes a large number of additional constraints in the ILP problem. We reduce the number of such constraints by merging the traffic

Algorithm 3: Location-Guided Protection Routing.

Input : A set of received data packets $\mathcal{P}$.
Output : The outgoing link for each packet (if any).

1 **for** *each* $pkt_{dst} \in \mathcal{P}$ **do**
2 pkt_{dst}.protection==TRUE;/* pkt_{dst} *to dst.* */;
3 **if** *available_links* - pkt_{dst}.in_link == $\varnothing$ **then**
4 /* *there is only one available link.* */ ;
5 LoopAvoidance(), **Return** NULL;
6 pkt_{dst}.out_link $\leftarrow$ **argmin**$_{n \in to_adjacent_nodes}$distance$(n, dst)$;
7 **Return all** pkt_{dst}.out_link.

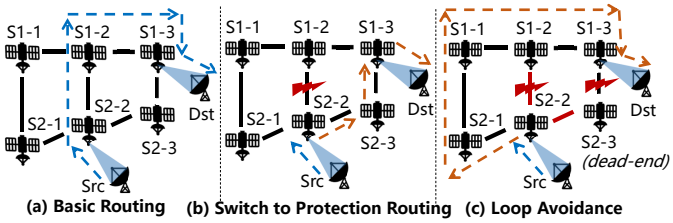


Fig. 4: Protection routing deals with unexpected failures.

demand with the same source destination pair in the same slot. For example, two traffic demands f_{ab}^1 and f_{ab}^2 with the same source-destination pair can be merged in a single demand from a to b, in which the traffic size is the sum of f_{ab}^1 and f_{ab}^2.

2) StarCure*'s protection routing strategy:* Note that the basic routing requires knowing the traffic matrix $\hat{\mathcal{F}}$ to solve the P2. If an unexpected failure occurs, the TSM adds a new burst traffic demand in $\hat{\mathcal{F}}$, and the basic routing has to re-compute new routing decisions. During such a re-computation period, StarCure switches to a location-guided protection routing (LGPR) to accomplish fast local recovery and switches back to the basic routing once its re-computation completes.

Algorithm details. LGPR leverages the unique mesh-like topology of STINs to calculate the position of each vertex in the network and guide packet forwarding. Each satellite vertex can be uniquely described by its relative location in the constellation topology, via an orbit index (p) and intra-orbit (q) satellite index. For example, a location index (p, q) indicates the qth satellite in the pth orbit. Because satellites in the same shell have the same altitude, their relative positions are stable in failure-free scenarios. For each satellite router, if it knows the destination location, it can estimate the path distance from the current vertex to the destination. In other words, a satellite router with a certain number of ISLs can estimate which adjacent ISL connects to a vertex closer to the destination. Algorithm 3 shows the details of LGPR for failure protection. For each packet received, LGPR calculates its outgoing link to the next adjacent node that is closer to the destination (line 6). Note that LGPR returns `NULL` if there is only one available link, *i.e.,* the ingress of the packet in current satellite (line 3-5). Figure 4 plots an example illustrating how StarCure deals with failures. Assume there is a basic routing from Src to Dst as shown in Figure 4(a). An unexpected failure occurs in link (S_{12}, S_{22}). When packets arrive at S_{22} (*i.e.,* the second satellite in orbit 2), because S_{22} knows the relative location of other satellites and S_{23} is more close to the destination, then S_{22} forwards packets to S_{23}, not S_{21}, as shown in Figure 4(b).

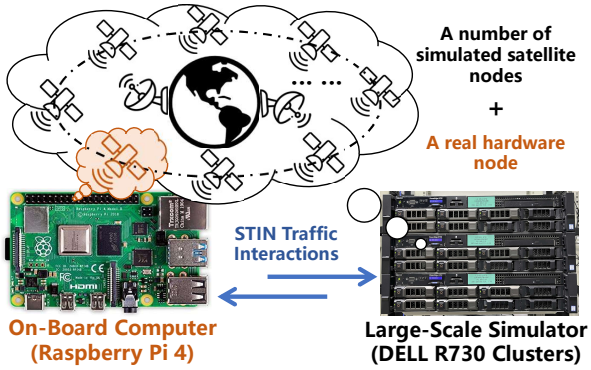


Fig. 5: Our STIN testbed combines a real hardware and a simulator to collaboratively build the experiment environment.

Loop avoidance. Since LGPR uses constellation-wide location information together with local link state, in certain scenarios it may suffer from forwarding loops. For example, in Figure 4(c), if link (S_{13}, S_{23}) also fails, S_{23} becomes a *"dead end"* with only one available link. In this situation, if S_{22} forwards packets to S_{23}, it suffers from a loop on (S_{22}, S_{23}), since S_{23} has only one available link. We design a *local assessment* strategy in each satellite router. If a satellite has only one available link, it temporally disables this link to avoid dead end in the network. For example, in Figure 4(c) S_{23} notifies S_{22} that it has only one available link. In this situation, S_{22} should not forward packets to S_{23}. Instead, it reroutes packets to S_{21} and finally to the destination as plotted in Figure 4.

V. Implementation

We implement a StarCure prototype on Linux, together with a hardware-in-the-loop STIN experimental environment that can simulate large scale constellations, load realistic network protocols and traffic.

Prototype. The control plane of StarCure's basic routing strategy, which pre-computes routing decisions for predictable failures is implemented based on `Gurobi` [28]. Once decisions have been made, the control plane leverages Linux `route` to update routing tables in the data plane. The protection routing strategy calculates distance to destination of each adjacent node via `Geopy` [29]. To inform routers which routing strategy should be used, we add a one-bit `protection` flag in IPv6 `Hop-by-Hop` extension header [30]. A satellite router forwards packets by basic routing if this flag is false, and by protection routing when this flag is true.

STIN testbed. We implement a hardware-in-the-loop STIN testbed as depicted in Figure 5 which integrates: (i) a container-based STIN simulator that allows us to simulate large-scale STINs and load realistic routing software as well as network traffic; and (ii) a real Raspberry Pi 4 computer which has been tested in a real space environment [31], and thus allows us to evaluate real system overhead of routing software. Specifically, we build our simulator based on a number of inter-connected `Mininet` [32] nodes, and each node simulates a satellite or ground station. We use orbit analysis tools [33] to

calculate inter-visibility and time-varying geo-locations of each node based on their public orbital information collected from `CeleTrak` [34]. The simulator is implemented and deployed on a cluster with three DELL-R730 PowerEdge servers. Each server has two Intel four-core 3.6GHz processors and 256G DDR4 RAM, running Ubuntu 20.04 and Mininet 2.3 software.

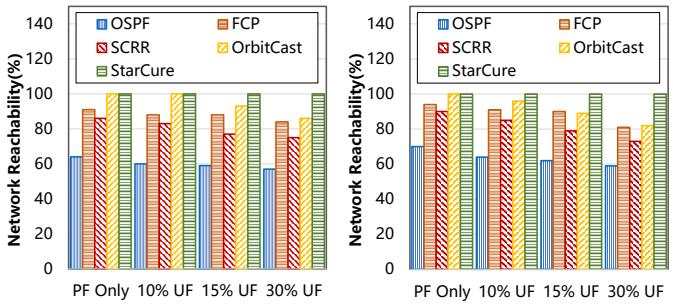
VI. Performance Evaluation

A. Experiment Setup

Constellation setting. Our evaluations are conducted based on the public details of two state-of-art satellite Internet constellations, SpaceX's Starlink [1] and Amazon Project Kuiper [2]. Starlink is currently the largest commercial LEO constellation under heavy deployment. In our STIN testbed, we simulate the complete Starlink first shell with 1584 satellites in 72 orbits at 550km altitude as the space backbone, together with SpaceX's ground stations obtained from [35]. Similarly, we simulate Amazon Project Kuiper together with its ground facilities, *i.e.,* AWS ground stations [36]. We set the link capacity following a previous study [37] and FCC fillings [1].
Traffic pattern. We generate the traffic demand matrices following the population-based methodology proposed in [38]. The traffic matrices in our experiment describe the traffic demands between the populous cites [39] covered by STIN's ground stations. The traffic volume of each demand is scaled in proportion to the population products of the city pairs.
Comparison. We also evaluate other representative resilience solutions for comparison: (i) OSPF [24], as one of the most widely deployed routing protocols in today's terrestrial Internet; (ii) Failure-Carrying Packets (FCP) [10], which uses the packet header to gather and carry the list of failed links required for routing that packet. Each packet carries the information of encountered link failures, and routers re-compute a new forwarding path on receiving that failure-carrying packet; (iii) Source-Controlled Resilient Routing (SCRR). A recent work [3] proposed to exploit source-controlled routing in STINs, and SlickPackets [15] is a source-controlled re-routing solution that allows packets to slip around failure by specifying alternate paths in their headers in the source controller. We thus combine these two works to build a SCRR solution; (iv) OrbitCast [18], a recent geographical location-based routing approach which forwards packets purely based on local location information and is convergence-free under any types of failures.

B. Network Reachability

Figure 6 plots the network reachability of different solutions, under various failures. "PF only" here indicates that the network only contains predictable failures caused by periodical LEO dynamics. UF indicates unexpected failures occur randomly according to a certain proportion. For example, 10% UF refers that about 10% of links in the STIN suffer from an unexpected outright failure. The network reachability is calculated as the ratio of the duration when routes are reachable, to the running time of the experiment. In addition, Table I shows the routing restoration time of each solution for different failure types. Collectively, we make several observations. First, as expected,



(a) Starlink topology. (b) Kuiper topology.

Fig. 6: Network reachability for different constellation topology under various failures. **PF:** predictable failures. **UF:** unpredictable failures for a portion of satellite links.

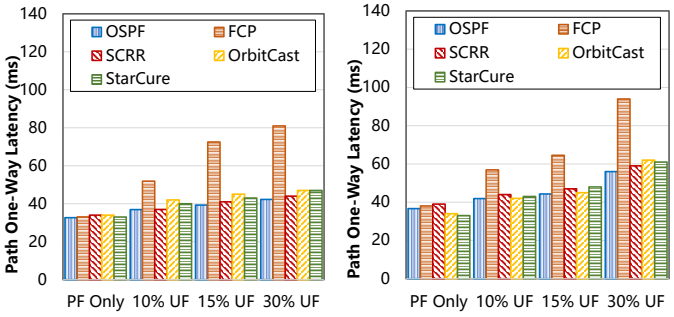
Failure Scenario	OSPF [24]	FCP [10]	SCRR [3], [15]	OrbitCast [18]	StarCure (this paper)
Single PF	7.4s	1.4s	1.8s	1.3s	0.6s
10% UF	35.2s	17.4s	6.4s	1.2s	1.1s
15% UF	79.1s	21.2s	11.7s	1.2s	1.2s
30% UF	153.8s	29.4s	16.2s	1.2s	1.3s

TABLE I: Routing restoration time for various failures.

OSPF which detects failures and performs routing restoration in a reactive manner suffers from long restoration delay. Since failures are common and frequent in a STIN environment, the network reachability of OSPF is only about 55% on average. FCP and SCRR achieve about 80-90% network reachability on average under different scenarios, which is much higher than OSPF. This is because they do not broadcast failure information globally to advertise all other routers to re-compute the routing tables. Instead, they pack the failure information into the packet header, and only these routers in the affected paths identify the failures and re-calculate the route, eliminating the network-wide reconvergence to improve reachability. However, since both FCP and SCRR still need to re-compute new routes after the failure and transit failures frequently occur, FCP and SCRR fail to accomplish 100% reachability due to the endless online re-computation process. Third, we observe that OrbitCast achieves 100% reachability when there are predictable failures only or with a small fraction of unexpected failures, but its reachability decreases as the failure rate increases. This is because OrbitCast routes packets based on local information together with the geo-location of the destination, and it can not avoid routing loops when the high failure rate results in dead ends in the network topology. Finally, StarCure achieves close-to-100% network reachability, by exploiting the adaptive hybrid routing scheme under various failure scenarios and under different constellation topologies.

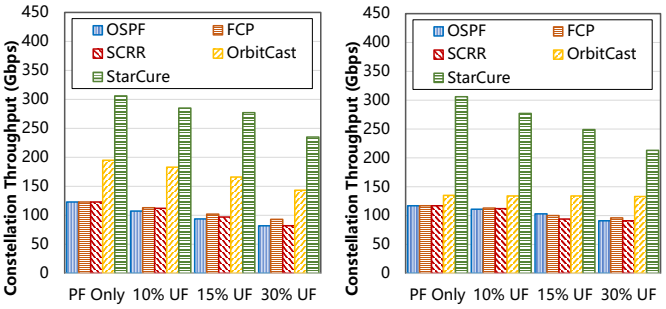
C. Latency Comparison

Figure 7 plots the one-way path latency of those new valid routes recovered from certain failures. We observe that the path latency increases as the failure rate increases. This is because when many links fail, packets have to travel a long distance to find another valid path to the destination. Specifically, OSPF and SCRR exploit the Dijkstra algorithm to find the shortest path in the network, but suffer from frequent



(a) Starlink topology. (b) Kuiper topology.

Fig. 7: Latency comparison for different resilience solutions.



(a) Starlink topology. (b) Kuiper topology.

Fig. 8: Throughput comparison for different resilience solutions.

re-calculations and limited network reachability. FCP attains higher latency because it calculates the shortest path based on current incomplete graph information, until the packets have traversed all failures. As a result, FCP has to re-compute the shortest path for several times for multiple link failures, and suffers from meandering routes. StarCure achieves similar latency results compared to OrbitCast, and achieves near-to-optimal latency while maintaining high network reachability.

D. Constellation Throughput

To understand how the achievable throughput changes after routing restoration, for each scenario when the STIN has recovered from unexpected failures, we enlarge the traffic demand, until the maximum link utilization in the network reaches 100%. Then we calculate the sum of all traffic demands as the achievable constellation-wide throughput. Figure 8 plots the constellation-wide throughput after failure restoration under different resilience solutions. We observe that as the failure rate increases and more links become unavailable, the throughput of each solution decreases because the amount of valid path that can carry traffic demands decreases. OSPF, FCP, SCRR and OrbitCast achieve about 100-200Gbps throughput under various failures, as they reroute traffic on the shortest path, resulting in high link utilization in these paths. StarCure is more flexible in dealing with failures and re-routing traffic under various failures, and it accomplishes up to 197% throughput improvement as compared to other four solutions.

E. System-level Overhead

Figure 9 plots the average CPU usage on the Raspberry Pi 4 computer in our hardware-in-the-loop experimentation which runs different resilient solutions under different failure scenarios. We do not plot SCRR as its major computation overhead is

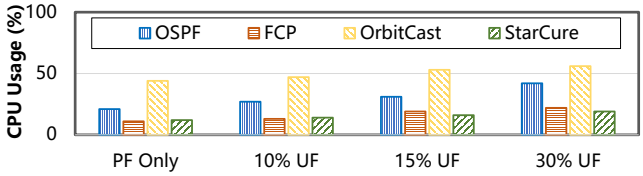


Fig. 9: CPU consumption of different resilient solutions.

aggregated in the source controller, not in each satellite router. OrbitCast requires the highest usage as it performs routing calculation for each packet. As compared to other solutions, StarCure accomplishes acceptable computation overhead.

VII. Related Work

Routing in STINs. The network community has many prior efforts focusing on satellite routing [40], [41], [42], [43], [44], [14], [3], [45], [46], [47]. Authors in [44] proposed a multi-tier space routing mechanism in a hybrid constellation with LEO and geostationary satellites. DEEPER [42] is an efficient transmission approach based on dynamical cooperation among satellite and ground networks. All these efforts acknowledge the low-latency, high-throughput capability of STINs, but most of them do not consider how to deal with various failures in the complex space environment. OPSPF [14] is a routing protocol for satellite networks that leverages satellite prediction to resist failures by periodically pre-computing routing tables. However, OPSPF is designed dedicated to polar orbit constellations. StarCure addresses the limitation of existing space routing mechanisms and achieves resilient and performance-guaranteed routing in STINs under various failure scenarios.

Routing under various traffic demands. To deal with high dynamics of Internet traffic, previous works have proposed robust traffic engineering that works well under highly variable traffic [48], [49]. Some solutions of them maintain a history of observed traffic demand matrices, and optimize for the representative traffic demand matrices. Other solutions are oblivious routing [50], [51], [52], aiming at optimizing the worst-case performance over all possible traffic demands. These works complement our study in this paper, as they focus on traffic variability but do not consider topology variability due to various network failures.

VIII. Conclusion and Acknowledgment

This paper presents StarCure, a resilient and performance-guaranteed routing mechanism for emerging STINs. StarCure incorporates a new network model, called topology-stabilizing model (TSM) to convert the frequent topology fluctuations to traffic fluctuation upon a logically stable topology. Then, StarCure adopts a hybrid routing algorithm that efficiently tackle large-scale time-varying traffic in STINs. Our extensive evaluations demonstrate that compared to existing resilience solutions, StarCure can accomplish close-to-100% reachability and better performance restoration.

We thank all INFOCOM reviewers for their feedback which greatly improved our paper. This work was supported by the National Key R&D Program of China (No. 2022YFB3105202), National Natural Science Foundation of China (NSFC No. 62132004) and Tsinghua University-China Telecom Joint Research Institute for Next Generation Internet Technology.

REFERENCES

[1] Application for fixed satellite service by space exploration holdings, llc. https://fcc.report/IBFS/SAT-MOD-2020041700037.

[2] Application for fixed satellite service by kuiper systems llc. https://fcc.report/IBFS/SAT-LOA-20190704-00057.

[3] Mark Handley. Delay is not an option: Low latency routing in space. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks (HotNets)*, page 85–91, 2018.

[4] Inigo Del Portillo, Bruce G Cameron, and Edward F Crawley. A technical comparison of three low earth orbit satellite constellation systems to provide global broadband. *Acta Astronautica*, 159:123–135, 2019.

[5] Yuanjie Li, Hewu Li, Lixin Liu, Wei Liu, Jiayi Liu, Jianping Wu, Qian Wu, Jun Liu, and Zeqi Lai. "Internet in Space" for Terrestrial Users via Cyber-Physical Convergence. In *Proceedings of the 20th ACM Workshop on Hot Topics in Networks (HotNets)*, pages 163–170, 2021.

[6] Kessler syndrome. https://en.wikipedia.org/wiki/Kessler_syndrome.

[7] 2009 irudium-cosmos collision fact sheet. https://swfound.org/media/6575/swf_iridium_cosmos_collision_fact_sheet_updated_2012.pdf.

[8] Spacex says a geomagnetic storm just doomed 40 starlink internet satellites. https://www.space.com/spacex-starlink-satellites-lost-geomagnetic-storm.

[9] Krishna P Gnawali, Heather M Quinn, and Spyros Tragoudas. Developing benchmarks for radiation testing of microcontroller arithmetic units using atpg. *IEEE Transactions on Nuclear Science*, 68(5):857–864, 2021.

[10] Karthik Lakshminarayanan, Matthew Caesar, Murali Rangan, Tom Anderson, Scott Shenker, and Ion Stoica. Achieving convergence-free routing using failure-carrying packets. *In Proceedings of the ACM SIGCOMM conference*, 37(4):241–252, 2007.

[11] Ye Wang, Hao Wang, Ajay Mahimkar, Richard Alimi, Yin Zhang, Lili Qiu, and Yang Richard Yang. R3: resilient routing reconfiguration. In *Proceedings of the ACM SIGCOMM conference*, pages 291–302, 2010.

[12] Junda Liu, Aurojit Panda, Ankit Singla, Brighten Godfrey, Michael Schapira, and Scott Shenker. Ensuring connectivity via data plane mechanisms. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 113–126, 2013.

[13] Baohua Yang, Junda Liu, Scott Shenker, Jun Li, and Kai Zheng. Keep forwarding: Towards k-link failure resilient routing. In *IEEE Conference on Computer Communications*, pages 1617–1625. IEEE, 2014.

[14] Tian Pan, Tao Huang, Xingchen Li, Yujie Chen, Wenhao Xue, and Yunjie Liu. Opspf: orbit prediction shortest path first routing for resilient leo satellite networks. In *IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

[15] Giang TK Nguyen, Rachit Agarwal, Junda Liu, Matthew Caesar, P Brighten Godfrey, and Scott Shenker. Slick packets. *ACM SIGMETRICS Performance Evaluation Review*, 39(1):205–216, 2011.

[16] Brent Stephens, Alan L Cox, and Scott Rixner. Plinko: Building provably resilient forwarding tables. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks (HotNets)*, 2013.

[17] Musk confirms laser crosslinks for starlink satellite network. https://smartmaritimenetwork.com/2021/01/27/musk-confirms-laser-crosslinks-for-starlink-satellite-network/.

[18] Zeqi Lai, Qian Wu, Hewu Li, Mingyang Lv, and Jianping Wu. Orbitcast: Exploiting mega-constellations for low-latency earth observation. In *IEEE 29th International Conference on Network Protocols (ICNP)*, 2021.

[19] Yannick Hauri, Debopam Bhattacherjee, Manuel Grossmann, and Ankit Singla. "Internet from Space" without Inter-Satellite Links. In *Proceedings of the 19th ACM Workshop on Hot Topics in Networks (HotNets)*, page 205–211, 2020.

[20] Bradley Denby and Brandon Lucia. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems(ASPLOS'20)*, page 939–954, 2020.

[21] Starlink launch statistics. Jonathan's Space Report. 19 December 2022. Retrieved 19 December 2022.

[22] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *Proceedings of ACM SIGCOMM*, page 161–174, 2002.

[23] Yin Zhang, Matthew Roughan, Nick Duffield, and Albert Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads. *ACM SIGMETRICS*, 31(1):206–217, 2003.

[24] OSPF Version 2. https://www.rfc-editor.org/rfc/rfc2328.

[25] OSI IS-IS Intra-domain Routing Protocol. https://datatracker.ietf.org/doc/html/rfc1142.

[26] Sanghwan Lee, Yinzhe Yu, Srihari Nelakuditi, Zhi-Li Zhang, and Chen-Nee Chuah. Proactive vs reactive approaches to failure resilient routing. In *IEEE INFOCOM 2004*, volume 1. IEEE, 2004.

[27] Satellite constellation wikipedia. https://en.wikipedia.org/wiki/Satellite_constellation.

[28] Gurobi optimization. https://www.gurobi.com/.

[29] Geopy. https://github.com/geopy/geopy.

[30] IPv6 Hop-by-Hop Options. https://www.ietf.org/archive/id/draft-hinden-6man-hbh-processing-00.html.

[31] Raspberry pi in space! https://www.raspberrypi.com/news/raspberry-pi-in-space/.

[32] Mininet. http://mininet.org/.

[33] Systems Tool Kit (STK). https://www.agi.com/products/stk.

[34] Celestrak. Norad two-line element set format description. https://www.celestrak.com/NORAD/documentation/tle-fmt.php.

[35] Starlink sx. https://starlink.sx/.

[36] AWS ground station antenna locations. https://aws.amazon.com/ground-station/locations/.

[37] Inigo del Portillo, Bruce Cameron, and Edward Crawley. Ground segment architectures for large LEO constellations with feeder links in EHF-bands. In *IEEE Aerospace Conference*, 2018.

[38] Giacomo Giuliari, Tommaso Ciussani, Adrian Perrig, and Ankit Singla. Icarus: Attacking low earth orbit satellite networks. In *USENIX Annual Technical Conference (ATC)*. USENIX Association, 2021.

[39] Greater London Authority (GLA). Global city population estimates. https://data.london.gov.uk/dataset/global-city-population-estimates.

[40] Jun Sun and Eytan Modiano. Routing strategies for maximizing throughput in leo satellite networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 22(2):273–286, 2004.

[41] Ales Svigelj, Mihael Mohorcic, Gorazd Kandus, Ales Kos, Matevz Pustisek, and Janez Bester. Routing in ISL networks considering empirical IP traffic. *IEEE Journal on Selected areas in Communications (JSAC)*, 22(2):261–272, 2004.

[42] Feilong Tang. Dynamically adaptive cooperation transmission among satellite-ground integrated networks. In *IEEE Conference on Computer Communications (INFOCOM)*, 2020.

[43] Xu Li, Feilong Tang, Yanmin Zhu, Luoyi Fu, Jiadi Yu, Long Chen, and Jiacheng Liu. Processing-while-transmitting: Cost-minimized transmission in sdn-based stins. *IEEE/ACM Transactions on Networking*, 2021.

[44] Yi Ching Chou, Xiaoqiang Ma, Feng Wang, Sami Ma, Sen Hung Wong, and Jiangchuan Liu. Towards sustainable multi-tier space networking for leo satellite constellations. In *IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*, 2022.

[45] Mark Handley. Using ground relays for low-latency wide-area routing in megaconstellations. In *Proceedings of the 18th ACM Workshop on Hot Topics in Networks (HotNets)*, page 125–132, 2019.

[46] Giacomo Giuliari, Tobias Klenze, Markus Legner, David Basin, Adrian Perrig, and Ankit Singla. Internet Backbones in Space. *ACM SIGCOMM Comput. Commun. Rev.*, 2020.

[47] Yuan Yang, Mingwei Xu, Dan Wang, and Yu Wang. Towards energy-efficient routing in satellite networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 34(12):3869–3886, 2016.

[48] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. Cope: Traffic engineering in dynamic networks. In *ACM SIGCOMM*, pages 99–110, 2006.

[49] David Applegate and Edith Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *ACM SIGCOMM*, pages 313–324, 2003.

[50] Harald Racke. Minimizing congestion in general networks. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 43–52. IEEE, 2002.

[51] Praveen Kumar, Yang Yuan, Chris Yu, Nate Foster, Robert Kleinberg, Petr Lapukhov, Chiun Lin Lim, and Robert Soulé. Semi-oblivious traffic engineering: The road not taken. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 157–170, 2018.

[52] Mohsen Ghaffari, Bernhard Haeupler, and Goran Zuzic. Hop-constrained oblivious routing. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1208–1220, 2021.