# StarryNet: Empowering Researchers to Evaluate Futuristic Integrated Space and Terrestrial Networks

Zeqi Lai and Hewu Li, *Tsinghua University and Zhongguancun Laboratory;* Yangtao Deng, *Tsinghua University;* Qian Wu, Jun Liu, and Yuanjie Li, *Tsinghua University and Zhongguancun Laboratory;* Jihao Li, Lixin Liu, and Weisen Liu, *Tsinghua University;* Jianping Wu, *Tsinghua University and Zhongguancun Laboratory*

https://www.usenix.org/conference/nsdi23/presentation/lai-zeqi

This paper is included in the Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation.

April 17–19, 2023 • Boston, MA, USA

978-1-939133-33-5

# StarryNet: Empowering Researchers to Evaluate Futuristic Integrated Space and Terrestrial Networks

Zeqi Lai[†‡], Hewu Li[†‡*], Yangtao Deng[†], Qian Wu[†‡], Jun Liu[†‡], Yuanjie Li[†‡], Jihao Li[†], Lixin Liu[†]
Weisen Liu[†], Jianping Wu[†‡]
[†]*Tsinghua University,* [‡]*Zhongguancun Laboratory*

## Abstract

Futuristic integrated space and terrestrial networks (ISTN) not only hold *new opportunities* for pervasive, low-latency Internet services, but also face *new challenges* caused by satellite dynamics on a global scale. It should be useful for researchers to run various experiments to systematically explore *new problems* in ISTNs. However, existing experimentation methods either attain realism but lack flexibility (*e.g.,* live satellites), or achieve flexibility but lack realism (*e.g.,* ISTN simulators).

This paper presents StarryNet, a novel experimentation framework that enables researchers to conveniently build credible and flexible experimental network environments (ENE) mimicking satellite dynamics and network behaviors of large-scale ISTNs. StarryNet simultaneously achieves constellation-consistency, networked system realism and flexibility, by adopting a real-data-driven, lightweight-emulation-aided approach to build a digital twin of physical ISTNs in the terrestrial virtual environment. Driven by public and real constellation-relevant information, we show StarryNet's acceptable fidelity and demonstrate its flexibility to support various ISTN experiments, such as evaluating different inter-networking mechanisms for space-ground integration, and assessing the network resilience of futuristic ISTNs.

## 1 Introduction

Thanks to the resurgence in the space industry [41, 46, 48], big competitors such as SpaceX and Amazon are actively planning and deploying hundreds or even thousands of broadband satellites in low earth orbits (LEO). Such emerging mega-constellations (*e.g.,* Starlink [34], Kuiper [9]) can be integrated into existing terrestrial Internet, *i.e.,* constructing an integrated space and terrestrial network **(ISTN)** to: (1) provide pervasive last-mile network access; (2) enable low-latency and high-bandwidth Internet transit [40, 52, 56]; and (3) facilitate efficient acquisition and delivery for big data from space (*e.g.,* earth observation images) [44, 74, 77, 78].

[*]Hewu Li is the corresponding author.

While holding great promise, several unique characteristics of LEO satellites (*e.g.,* high LEO dynamics) impose new challenges at various layers of the ISTN networking stack, and open a door to many new research problems, such as: (1) how should LEO satellites and ground facilities be interconnected to provide low-latency and continuous network services? (2) how should satellite routers be integrated into existing terrestrial Internet routing? and (3) are current constellation and protocol designs resilient enough to satellite failures in complex and harsh space environments? With many *unexplored* problems facing the "NewSpace" industry, it is thus foreseen that in the near future, there will be a surge of new ideas on the system and networking research relevant to ISTNs. But, *how can researchers build an experimental network environment* ***(ENE)*** *to test, evaluate and understand their new thoughts?*

Typically, existing approaches for creating an ENE can be classified into three categories: (1) live networks or platforms [7, 20, 34, 75, 81], which allow experiments in real deployments; (2) network simulation [60, 61, 76], which uses discrete events to model and replicate the behavior of a real network; and (3) network emulation [6, 55, 68, 69], which can test real applications/protocols in a virtual network. However, as will be illustrated in §3, all existing approaches have their limitations in creating a desired ENE for ISTNs: (1) the *feasibility* and *flexibility* of live satellite networks are *technically* and *economically* limited for normal researchers; (2) the abstraction level of simulation might be too high to capture low-level system effects, hiding practical issues such as the resource competition under heavy workload, energy drain or software errors; (3) existing emulators fail to characterize the high dynamicity of LEO satellites and thus are insufficient to build an experimental environment with acceptable fidelity.

The key challenge of building an expected ENE for ISTN research is: it is difficult to *simultaneously* achieve *realism* and *flexibility* in the experimental environment. First, terrestrial devices inherently lack the ability to reasonably mimic the high dynamics, system and network behaviors of realistic satellites. Second, mega-constellations typically consist of thousands of satellites. Thus the network scale required by an

ENE for mega-constellations might be far more than the extent supported by existing ENE methods (*e.g.,* [54, 55, 69]). Third, as a large number of satellites simultaneously move at a high velocity, continuously mimicking such frequent variations at scale could involve significant system overhead on the ENE.

This paper presents STARRYNET, an integrated experimentation framework that empowers researchers to conveniently build ENEs with acceptable realism, flexibility and cost (*e.g.,* requiring only a few number of local/cloud machines) to satisfy various experimental requirements of ISTNs. The design of STARRYNET is inspired by a key insight obtained from the satellite Internet ecosystem: many organizations or communities in this ecosystem have released and shared their constellation-relevant data, including regulatory information [2, 73], satellite trajectories [16, 38], ground station distribution [26, 39] and measurements from user terminals [32, 33], *etc.* Therefore, the key idea behind STARRYNET is to build an experimental *digital twin*, *i.e.,* a virtual presentation of a physical ISTN, in terrestrial environments by: (1) leveraging terrestrial machines to virtualize a large number of lightweight virtual nodes to emulate satellites in mega-constellations; and (2) exploiting a *crowdsourcing* approach to collect, combine and use realistic constellation information to drive the emulation of *spatial* and *temporal* characteristics of ISTNs.

To achieve acceptable realism, STARRYNET employs a *constellation synchronizer* based on realistic constellation-relevant information to make the virtual ENE as consistent as possible to a real ISTN, such as: (1) *constellation consistency:* the ENE is built with the same scale of a physical mega-constellation, where each node emulates a satellite, a ground station or a terrestrial host. The spatial and temporal characteristics, such as time-varying satellite locations and inter-visibility, are also configured and updated in each node based on our data-driven model-based analysis; (2) *system and networking stack consistency:* the ENE can support the run of unmodified applications as in real deployments; and (3) *capability consistency:* network and computation capabilities in the ENE are configured based on real hardware specifications. Further, to flexibly support various ISTN experiments and mimic *large-scale* and *highly-dynamic* mega-constellations, STARRYNET adopts a *constellation orchestrator* that judiciously schedules and manages system resources on multiple machines to collaboratively construct ENE on demand.

We evaluate the ability of STARRYNET based on real constellation information in two steps. First, we show the acceptable fidelity of STARRYNET by comparing the experiment results obtained by STARRYNET with live satellite networks and other state-of-the-art ISTN simulators. Second, facing futuristic ISTN scenarios, we demonstrate STARRYNET's flexibility by conducting three case studies to: (1) explore the trade-space of various space-ground inter-networking mechanisms; (2) evaluate the resilience of routing protocols in various constellation designs; and (3) perform hardware-in-the-loop tests to measure system effects under various workloads.

Summarily, this paper makes the following key contributions: **(1)** we design STARRYNET, a data-driven, emulation-aided ISTN experimentation framework (§4); **(2)** we implement STARRYNET with a collection of open APIs for creating and manipulating user-defined ENEs (§5); **(3)** we evaluate and analyze STARRYNET's experimentation overhead and fidelity (§6), and show STARRYNET's flexibility (§7) by conducting various case studies driven by realistic constellation information. STARRYNET is now available at: `https://github.com/SpaceNetLab/StarryNet`.

## 2 Preliminaries

**Integrated space and terrestrial networks (ISTN).** Recent satellite operators/organizations are actively developing their mega-constellations [4, 8, 9, 25, 34, 36], with hundreds to thousands of low earth orbit (LEO) satellites working together as a system. These satellites can be equipped with high-speed inter-satellite links (ISLs), and construct an LEO satellite network **(LSN)**. An LSN can further be integrated into existing terrestrial Internet via globally distributed ground stations [3, 26, 39] and very-small-aperture terminals (VSAT) [31], constructing an *integrated space and terrestrial network (ISTN)* that promises to provide pervasive, low-latency, broadband Internet services [40, 52, 56, 57] for terrestrial users globally.

**Unique characteristics of ISTNs, as well as the new challenges.** Two critical characteristics differentiate LSNs from existing terrestrial networks, and involve new challenges on the integration of satellites and terrestrial Internet. First, LEO satellites are moving at a high-speed with the respect to the earth surface. An LEO satellite might be visible for a certain ground vantage point only within *a few minutes* in one pass. Such high dynamics could inevitably result in technical challenges (*e.g.,* frequent connectivity disruptions and routing re-convergence) at the networking stack of ISTNs. Second, while evolved, resources (*e.g.,* bandwidth, CPU, energy) are still limited and costly in space, as compared with terrestrial network systems. Resource-intensive technologies might not be doable for resource-constrained satellites to sustain good network performance (*e.g.,* applying sophisticated network coding techniques for packet recovery in remote space).

**Call for new research for futuristic ISTNs.** The above characteristics and challenges accordingly raise a series of *unexplored* research problems in ISTNs, such as: (1) ***topology:*** how should LEO satellites and ground facilities be interconnected under the high space-ground dynamicity? (2) ***routing:*** how should we integrate hundreds or thousands of LEO satellites into Internet routing and tackle the potential performance degradation due to intermittent connectivity? (3) ***system effect:*** how much energy would a new functionality consume in space under various workloads? It is foreseeable that in the near future, in parallel with the evolution of real mega-constellations, there would be a surge of new research focusing on emerging satellite network systems. But, how should researchers test, assess and understand their new

| Category / Tools | | (i) Constellation Consistency | (ii) System and Networking Stack Realism | (iii) Flexible and Scalable Environment | (iv) Low-cost and Easy-to-use |
|---|---|---|---|---|---|
| Live LSNs or platforms | Live Starlink ( [34]) | ✓ | ✓ | ✗ | ✗ |
| | PlanetLab ( [20]) | ✗ | ✓ | ✗ | limited |
| | Emulab ( [7]) | ✗ | ✓ | ✗ | limited |
| Simulators and orbit analysis tools | STK ( [35]) | ✓ | ✗ | ✓ | limited |
| | GMAT ( [11]) | ✓ | ✗ | ✓ | ✓ |
| | SNS3 ( [76]) | for GEO only | ✗ | ✓ | ✓ |
| | Hypatia ( [60]) | ✓ | ✗ | ✓ | ✓ |
| | StarPerf ( [61]) | ✓ | ✗ | ✓ | ✓ |
| Emulators and variations | MiniNet ( [55,68]) | ✗ | ✓ | ✓ | ✓ |
| | DieCast ( [54]) | ✗ | ✓ | limited at scale | ✓ |
| | Etalon ( [69]) | ✗ | ✓ | limited at scale | ✓ |
| STARRYNET (this paper) | | ✓ | ✓ | ✓ | ✓ |

Table 1: Comparison of popular network experimentation platforms across different ENE requirements for ISTNs.

thoughts? The community requires a *technically* and *economically* feasible approach to construct ***Experimental Network Environments (ENE)*** and advance futuristic ISTN research.

## 3 How Can Researchers Evaluate Their New Thoughts for ISTNs?

### 3.1 ENE Requirements

Ideally, an ENE built for ISTN research is expected to simultaneously accomplish *acceptable* realism and flexibility. We summarize four baseline requirements as follows.

- **(1) Constellation-consistency.** The ENE is expected to be *spatially* and *temporally* consistent to the characteristics of real mega-constellations. For example, the ENE is desired to mimic a large number of network nodes at the same scale of a real mega-constellation, and can characterize the high dynamicity of LEO satellites, as well as its impact on network conditions (*e.g.,* connectivity, delay variations).
- **(2) System-level and networking stack realism.** The ENE is expected to run user-defined system codes and network functionalities like in a real system and networking stack.
- **(3) Flexible and scalable environment.** Emerging mega-constellations are evolving rapidly. As most state-of-the-art constellations are still not in their final stage, the ENE is expected to flexibly support various network topologies at different scales to meet diverse research requirements.
- **(4) Open, low-cost and easy-to-use interface.** Finally, it is expected that the ENE could be open to the community, and can provide low-cost and easy-to-use programmable interfaces for researchers to carry out various experiments.

### 3.2 Why Existing ENEs are Insufficient?

Existing approaches for building an ENE can be classified into three categories, differing in their realism, flexibility and cost: (1)live LSNs/platforms, (2)simulators, and (3)emulators.

**Live LSNs or platforms.** A straightforward approach for ISTN experimentation is to construct an ENE based on *live LSNs*, *e.g.,* recently SpaceX's Starlink has started its initial services in certain regions. Although this approach guarantees good realism, directly manipulating and inspecting a live LSN might be *technically* and *economically* difficult for a common research group. Current live LSNs are also limited in their *flexibility* when they face diverse, exploratory research requirements. Realistic mega-constellations are still under-constructed and evolving rapidly, and their regulatory information in practice cannot be flexibly modified for *what-if* analysis. Further, the network community has many public experimentation platforms [7, 20] that can be shared among researchers. However, these platforms are originally designed for tests in terrestrial networks, not for ISTNs, and thus cannot characterize the unique network behaviors under large-scale LEO dynamics.

**Simulators and orbit analysis tools for ISTNs.** Numerical or discrete-event-based simulation presents another extreme as compared with live LSNs and platforms. STK [35] and GMAT [11] are representative *orbit analysis tools* that can perform complex analysis of spacecrafts as well as ground stations. However, both STK and GMAT mainly focus on orbit and spacecraft analysis and have limited support for network simulation. More recently, SNS3 [76], Hypatia [60] and StarPerf [61] are emerging simulators for ISTNs. SNS3 is an extension to the ns-3 platform, and it models a full satellite network with a geostationary (GEO) satellite and bent-pipe payload. Hypatia is a framework for simulating and visualizing the network behavior of emerging mega-constellations. Similarly, StarPerf is a simulator that enables users to characterize, estimate and understand the achievable network performance under a variety of constellation options. Although the above simulators can *flexibly* simulate various satellite characteristics as well as the impact of high dynamics on network behaviors, a fundamental limitation of those simulators is that they can not support the run of system codes/functionalities and interactive network traffic as in real deployments. The abstraction-level of simulators might be too high to capture system-level effects, and could hide other practical issues (*e.g.,* software overhead under heavy workloads) in real systems.

**Network emulators, and their variations.** Emulation is a hybrid approach that integrates real applications, protocols and operating systems in a synthetic network environment. Similar to live networks, emulators can load and run real codes with interactive network traffic. Similar to simulators, emulators can support controllable and diverse topologies and their virtual hardware requires fewer resources as compared with live networks. The community has many prior efforts focusing on emulated environments, *e.g.,* VM- or container-based emulation [6, 45, 54, 55, 68–70, 72, 79–81, 84, 85].

However, existing emulators suffer from two limitations when they are applied for generating ENEs for ISTN research. First, they are not constellation-consistent, since existing emulators inherently lack the ability of mimicking planet-wide LEO dynamics and time-varying network behaviors in ISTNs. Second, the network scale for mega-constellations could be significantly larger than that in prior experimentation. For example, authors in [54] use 10 physical machines (25 VMs on each) to support a networked cluster with 250 nodes. Etalon [69] is a container-based emulator and its local testbed uses four servers to emulate 48 hosts in a data center network. Different from prior scenarios, ISTN experiments require a much larger network environment: only the first shell of Starlink Phase-I includes about 1584 LEO satellites. Since both VMs and containers can involve software overhead on the physical host machine, it is difficult for existing emulators (*e.g.,* [54, 55, 68]) to support such large-scale and dynamic emulations for mega-constellations.

**Our motivation.** Table 1 summarizes the landscape of existing experimentation approaches that can be used to build ENEs. Collectively, we find that none of existing approaches can simultaneously satisfy the four expected features. Limitations of existing approaches thus motivate us to seek for a constellation-consistent, credible, flexible, and low-cost methodology to advance the test and evaluation of new research for futuristic ISTNs. We present such a framework, namely STARRYNET, aiming at empowering researchers to build ENEs accomplishing the four goals as described in §3.1.

## 4 STARRYNET Design

### 4.1 System Overview

**Key idea.** The design of STARRYNET is inspired by an important insight obtained from the satellite Internet ecosystem: many organizations (*e.g.,* regulators and satellite operators) and end users have shared a collection of public data for the community, including constellation regulatory information [2, 73], orbital data observed from realistic satellites [16, 38], ground station distributions [3, 26, 39], and performance results measured from terrestrial user terminals [32, 33], *etc.* Based on this important fact, STARRYNET creates ISTN experimental environments on demand by judiciously combining: (1) crowd-sourced real data trace; (2) model-based orbit and network analysis; and (3) large-scale network system emulation, to construct a real-data-driven digital twin, *i.e.,* a virtual presentation synchronized to a real

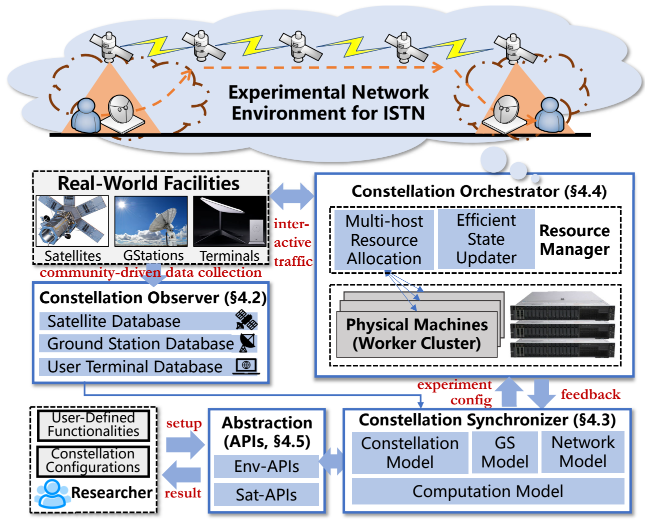


Figure 1: STARRYNET system architecture.

physical ISTN in terrestrial environments. In particular, the key idea behind our STARRYNET design can be summarized as follows: (1) leveraging a crowd-sourcing approach to collect, combine and explore realistic constellation-relevant information to calculate the *spatial* and *temporal* characteristics consistent to real mega-constellations; then (2) driven by such realistic information, exploiting a large number of networked *virtual nodes and links* to flexibly emulate a customized experimental environment, which characterizes system-level effects and network behaviors consistent to a real ISTN.

**System architecture.** Figure 1 depicts STARRYNET's architecture, including four core components as described below.

- A ***Constellation Observer*** (§4.2) that leverages a *crowd-sourcing* approach to collect public constellation information, network performance, ground station distributions *etc.*, from the satellite ecosystem. It maintains several databases to support, guide and drive the construction of ISTN experimental environments for various research requirements.
- A ***Constellation Synchronizer*** (§4.3) which exploits a collection of hybrid models to calculate the *spatial and temporal characteristics* of a specific mega-constellation, based on collected data as well as user-defined configurations. Specifically, such characteristics include constellation scale, visibility, connectivity, time-varying propagation delay, *etc.*, which are further used to configure the network emulation.
- A ***Constellation Orchestrator*** (§4.4) for automating the management, coordination and allocation for resources used to build the experimental environment upon multiple physical machines. The orchestrator can also interact with real-world facilities (*e.g.,* real satellite hardware) to support network experiments with interactive Internet traffic.
- A ***Unified Abstraction*** (§4.5) offering flexible and easy-to-use *APIs* for researchers to create and manipulate the configurations of the ISTN experimental environment.

### 4.2 Constellation Observer

The constellation observer is designed as a collector for constellation-relevant information, and it maintains three databases related to satellite, ground station and user terminal information respectively. Specifically, the observer searches and collects the latest: (1) regulatory information (*e.g.,* from [2, 10, 73]) which describes frequency and orbital coordination of mega-constellations; (2) operating satellites information (*e.g.,* from [16, 38, 71]); (3) ground station distributions (*e.g.,* from [3, 26, 39]); (4) Internet user statistics (*e.g.,* from [59]) which can be used to generate the distribution of terrestrial users; and (5) network measurements from end users (*e.g.,* from [32, 33]). The constellation observer classifies the above information, and saves them in the databases, which then can be used to drive other components and build ENEs to flexibly support various research experiments.

### 4.3 Constellation Synchronizer

StarryNet's synchronizer leverages a collection of models to calculate various constellation characteristics and network behaviors, and accordingly generate a virtual network presentation synchronized to a real ISTN. At runtime, values in these models are assigned primarily based on real ISTN information collected by the constellation observer. In addition, to improve the flexibility, StarryNet also allows researchers to manually configure model values based on his or her customized experimental requirements.

#### 4.3.1 Hybrid models

**Constellation model.** StarryNet's synchronizer characterizes a satellite system in both *constellation-granularity* and *orbit-granularity* descriptions. First, StarryNet uses the Walker notation [82] to describe a constellation: $N/P/p$, where $N$ is the number of satellites per plane, $P$ is the number of planes, and $p$ is the number of distinct phases of planes to control spacing offsets in planes. Second, the orbit-granularity description enables a fine-grain notation for orbits in a certain constellation via specifying several primary orbital elements, including: (1) inclination, which is the angle between an orbit and the Equator as satellites move; (2) altitude, which is the height above sea level and determines the orbital velocity; (3) phase offset [56], which is a factor between $[0,1]$, describing the relative position between two satellites in adjacent orbits.
**Ground station model.** StarryNet leverages the following parameters to describe a ground station: (1) geographic location; (2) the number of available antennas for ground communication; (3) the elevation angle, which determines the light-of-insight (LoS) of the ground station and can affect the available duration of space-ground communication.
**Network model.** The inter-satellite or ground-satellite network connectivity is mainly affected by following factors: (1) the visibility between two communication ends; (2) the amount of available ISLs or antennas in satellites or ground stations; and (3) the connectivity policy, which decides how to establish a connection between two communication ends. StarryNet enables two prefabs of connectivity policies for inter-satellite connection: (1) +Grid [58], where each satellite connects to two adjacent satellites in the same orbit and to two in adjacent orbits; (2) Motif [40], which is a repetitive pattern where each satellite connects to multiple visible satellites and each satellite's local view is the same as that of any other. For ground-satellite connectivity, StarryNet offers multiple optional schemes that control a ground station to connect to a visible satellite, *e.g.,* selecting the one with the shortest distance or the longest remaining visible time. Since the delay is typically determined by the network topology and speed of the light, StarryNet calculates the propagation delay of each link based on the physical distance between two ends. As network capacity might be too speculative in practice, link capacity is set by user-specific configurations.
**Computation model.** The computation capability of satellites varies greatly in different real-world deployments. Generally, conventional space-grade processors have limited capability [53, 65] as compared with that used in terrestrial computer systems. For example, the operating frequency of spaceborne processors (*e.g.,* BAE-RAD series [21,22]) ranges from 110 to 466MHz per core. Recently, satellite operators and researchers start to use commercial off-the-shelf (COTS) processors in space stations [14] or in LEO small satellites [23, 44] to reduce the manufacturing cost. To support various experimental requirements, StarryNet allows researchers to manually configure the CPU capability of each satellite node through approximating the frequency and number of available cores of each emulated node, by scaling download CPU frequency and enforcing a maximum time quota for each node.

#### 4.3.2 Constellation consistency

**Spatial consistency.** Based on constellation-wide information, StarryNet first determines the amount of required containers. Each container runs realistic system environments and networking stack to emulate a satellite in the constellation, a ground station or a terrestrial user terminal. Second, using orbit-granularity information, StarryNet calculates the latitude, longitude and height (*i.e.,* LLH information) of each satellite at any given time slot. Finally, exploiting the above LLH information and minimum elevation angles, StarryNet calculates the visibility between arbitrary two satellites, or between satellites and ground facilities.
**Temporal consistency.** Since emerging LEO satellites are inherently moving at a high velocity, the locations, intervisibility, and network conditions (*e.g.,* connectivity and propagation delay between two nodes) of an ISTN are changing over time. To achieve realism, these states should be temporally consistent to real scenarios. StarryNet splits time into slots, calculates LLH information in each slot, and follows the hybrid models to determine time-varying network conditions in different slots. Such dynamic states will further be used for driving the dynamic emulation operated by the orchestrator.

## 4.4 Constellation Orchestrator

The orchestrator is designed for four goals. First, the orchestrator exploits container-based emulation to construct an emulated ISTN environment upon one or many physical machines (depending on the constellation size). Second, the orchestrator configures the computation and network capability of each emulated node, according to users' configurations and the spatial and temporal results calculated by the constellation synchronizer. For example, a space-ground connectivity should be dynamically updated based on the time-varying visibility between its two ends. Third, the orchestrator can connect the emulated ISTN to real-world network facilities (if any, *e.g.,* real satellite prototypes or terminals) and support interactive Internet traffic. Finally, at runtime the orchestrator leverages a measurement agent to monitor and report the runtime resource usage (*e.g.,* CPU/memory/bandwidth usage) to the user as a feedback of the experiment for further analysis.

### 4.4.1 Multi-machine support for constellation emulation

Since each emulated satellite consumes computation, network and storage resources in practice, it is challenging to support the emulation of large-scale constellations on a single machine, especially when additional user-defined payloads (*e.g.,* a new satellite routing protocol) need to be loaded for experimentation. STARRYNET addresses this limitation by integrating resources on multiple machines to support large-scale, time-varying constellation emulations. When deployed on multiple machines, STARRYNET's orchestrator divides these machines into two parts. First, one machine, selected as the *resource manager*, manages, schedules and allocates resources upon all machines to jointly create and maintain the ENE. Second, other machines, working as *worker clusters*, receive and follow commands from the resource manager. For each node in the ISTN (*e.g.,* a satellite or ground station), the orchestrator creates a container to emulate it, and creates a virtual bridge connecting two nodes to emulate a link.

**Topology creation on multiple machines.** One big challenge made by extending an emulated mega-constellation to multiple machines is to *ensure that the emulated constellation is topologically consistent to the real constellation*. Figure 2 shows an example of emulating an LEO satellite constellation including two adjacent orbits on two physical machines. Specifically, Figure 2a plots two Starlink inclined orbits, each of them having 22 satellites evenly spaced with available ISLs. Assume that we use two machines for this emulation, and each machine creates 22 containers to emulate 22 satellites. In a real constellation, each satellite connects its two neighbors in the same orbit, and to another satellite in the adjacent orbit. Ideally, an emulated topology for the constellation in Figure 2a can be easily created if each machine has more than 22 physical network interfaces. However, in practice the number of available interfaces of a machine is likely to be limited. As shown in Figure 2b, if we directly bridge the emulated network interface of each satellite to the physical interface, due to the lack of traffic isolation, emulated satellites in two machines will establish an "all-to-all" topology which is inconsistent to the grid-like topology in Figure 2a .

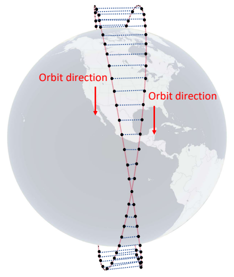
STARRYNET addresses this inconsistency on multiple machines by creating multiple VLANs to emulate independent ISLs, interconnect satellites in two machines and isolate inter-satellite traffic as that in a real constellation. As plotted in Figure 2c, we build a VLAN for each ISL crossing different machines (*e.g.,* vlink A1-B1 interconnects emulated satellite A1 and B1), and thus STARRYNET obtains the correct virtual network topology consistent to the real constellation topology as illustrated in Figure 2a upon multiple machines.

**Topology update on multiple machines.** Due to the high dynamics of ISTN, the network topology fluctuates over time. If a connectivity change occurs on a single machine, *i.e.,* all affected nodes are located on the same machine, STARRYNET deletes the old virtual link, and creates a new link connecting corresponding nodes. Otherwise, if a connectivity change involves nodes on multiple machines, STARRYNET exploits a VLAN-based approach to limit the link update operation to a single machine. Figure 3 plots an example of the emulation of space-ground handovers upon multiple machines. Assume there are three satellites, two ground stations in two sequential time slots. In the first slot, satellite S2/S3 connects to ground station GS1/GS2 respectively. As satellites move, the space-ground connectivity changes, and S1/S2 connects to GS1/GS2 respectively in the second slot. STARRYNET emulates the ground-satellite links (GSLs) by two VLAN-based virtual links, and performs the correct handover by properly adjusting the connectivity change between S1/S2/S3 and vlink-GS1/GS2, in different slots on machine A.

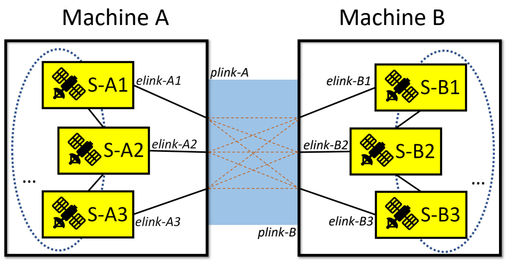
### 4.4.2 Efficient time synchronization and state update

Another challenge made by multi-machine extension is the *time synchronization and link update overhead.* Specifically, to achieve temporal consistency, STARRYNET's constellation synchronizer assigns a sequence of update events to the orchestrator to inform each emulated satellite *when* an update (*e.g.,* a location/connectivity change) should happen. To trigger such events, a baseline approach is to let the centralized manager send commands to all emulated satellites in every slot, and trigger corresponding update events. However, such a real-time approach has limited scalability as the amount of emulated satellites increases, since each event requires to update the state of a certain virtual interface/container, and continuously and simultaneously performing a large number of updates can overload the manager, result in delayed update, and invalidate the temporal consistency of the emulation.

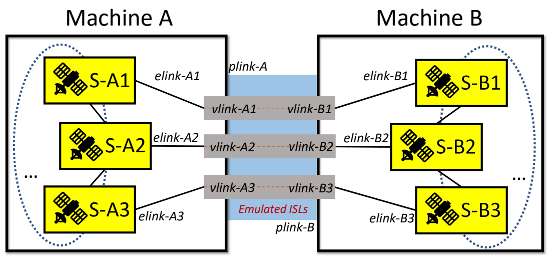
To reduce the state update overhead caused by mimicking temporal dynamics in mega-constellations, STARRYNET leverages a prediction-based *multi-thread event memorization* approach. We define the synodic period as the amount of time that it takes for the constellation to reappear at the same projection upon the earth surface. In the emulated environment,



(a) Two orbits with ISLs.



(b) Emulated topology without traffic isolation.



(c) Emulated topology with emulated ISLs for traffic isolation.

Figure 2: Multi-machine extension to support consistent topology emulation of satellite networks. Machine A and B interconnect by their physical interfaces plink-A/B (plink: physical link, elink: emulated link, vlink: virtual network link created by `VLAN`).

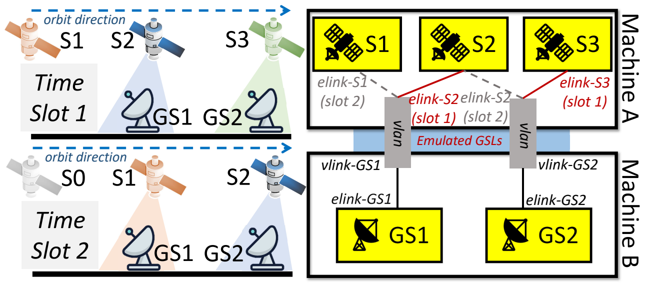


Figure 3: Emulated ground satellite links (GSLs) and space-ground handover in two time slots on multiple machines.

at the beginning of each *synodic period*, the orchestrator pre-generates an event list for each satellite that includes all its update events during the current synodic period. At runtime, each emulated satellite adopts an independent *event update thread* to read the local event list in each slot, and triggers the corresponding event scheduled in the current slot. Time clocks upon all machines are synchronized by the NTP [17].

## 4.5 Open APIs for ISTN Experiments

**Environment APIs.** STARRYNET provides the environment APIs for a researcher to load trace from the database, and create/control/run an ENE upon one or multiple machines with user-defined ISTN configurations. Once constellation and GS information are completely loaded, these configurations are delivered to the synchronizer to calculate spatial and temporal characteristics, which are further used by the orchestrator to construct the ENE. The environment APIs also allow users to configure the interval of discrete time slot to adjust the dynamicity. Note that the ENE not only maintains the run-time of emulated constellations, but also needs to run the test workload deployed by the researcher. We design a resource threshold $\Delta$ to control the percentage of CPU/memory used by the framework. In other words, at least (100%-$\Delta$) of the total CPU/memory should be left for the researcher's test cases.

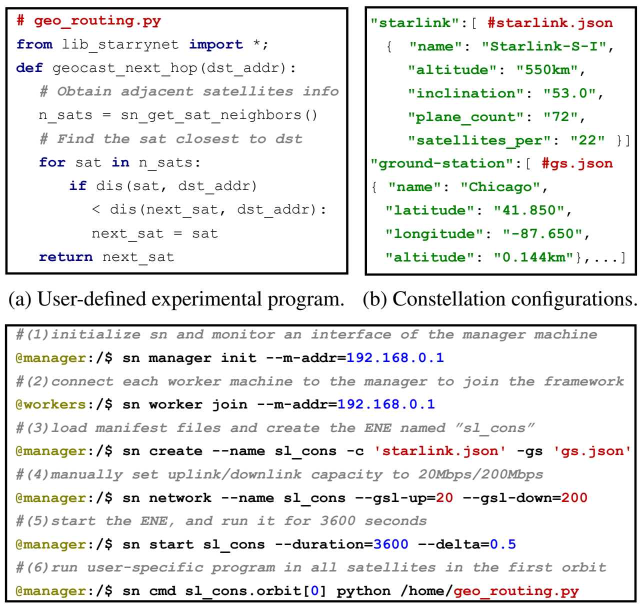
**Self-node APIs.** STARRYNET's self-node APIs are designed to be called by the researcher's programs on each emulated satellite. These APIs expose underlying satellite-related information to user programs. Specifically, in each emulated satellite, user program can obtain the index of current satellite/orbit, sunlight state, time-varying geo-location information, current satellite velocity and the index of adjacent reachable satellites, *etc.* Such satellite-specific information can be used for developing new on-board capabilities in ISTNs.

## 5 Implementation and Usage

We highlight the salient aspects of STARRYNET's implementation and usage in this section.

**Framework implementation.** STARRYNET's observer is implemented as a combination of a crawler based on `Scrapy` [27], together with a MySQL-based data store. We implement STARRYNET's synchronizor based on `SkyField` [29], an astronomy library that supports the calculation of high precision research-grade positions for satellites. STARRYNET's orchestrator is implemented upon `Docker` [6] and it spans the emulated constellation across multiple machines. We use `OpenvSwitch` [19] to emulate and configure links, and use `tc` [67] to dynamically set artificial network conditions according to the numeric results calculated by STARRYNET's synchronizer. Specifically, we optimized the link management module in `tc` to satisfy the requirement of light-weight state update. To accomplish flexibility, STARRYNET's abstraction is implemented as a combination of a `lib`-STARRYNET library and a collection of shell commands. Collectively, the core components of STARRYNET are implemented in about 6500 lines of Python codes and scripts.

**Framework usage.** We illustrate the usage of STARRYNET with a concrete example as plotted in Figure 4: a researcher wants to evaluate a novel geo-location-based routing mechanism based on [63], under the Starlink constellation. In particular, this experiment can be conducted with STARRYNET in three steps. First, leveraging STARRYNET's APIs, the researcher writes a user-defined experimental program (Figure 4a) for test. In this example, we show a geo-routing policy similar to [63], which runs on each satellite, and forwards received packets to the adjacent satellite that is the geographically closest to the destination. Second, the researcher prepares a set of manifest files describing the constellation configurations, *e.g.,* orbital information and ground station distribution (Figure 4b). Finally, the researcher runs a batch of shell commands exposed by STARRYNET to load manifest files (*e.g.,* starlink.json and gs.json), create experimental en-



```
# geo_routing.py
from lib_starrynet import *;
def geocast_next_hop(dst_addr):
   # Obtain adjacent satellites info
   n_sats = sn_get_sat_neighbors()
   # Find the sat closest to dst
   for sat in n_sats:
      if dis(sat, dst_addr)
         < dis(next_sat, dst_addr):
         next_sat = sat
   return next_sat
```

(a) User-defined experimental program.

```
"starlink":[ #starlink.json
  { "name": "Starlink-S-I",
    "altitude": "550km",
    "inclination": "53.0",
    "plane_count": "72",
    "satellites_per": "22" }]
"ground-station":[ #gs.json
{ "name": "Chicago",
  "latitude": "41.850",
  "longitude": "-87.650",
  "altitude": "0.144km"},...]
```

(b) Constellation configurations.

```
#(1)initialize sn and monitor an interface of the manager machine
@manager:/$ sn manager init --m-addr=192.168.0.1
#(2)connect each worker machine to the manager to join the framework
@workers:/$ sn worker join --m-addr=192.168.0.1
#(3)load manifest files and create the ENE named "sl_cons"
@manager:/$ sn create --name sl_cons -c 'starlink.json' -gs 'gs.json'
#(4)manually set uplink/downlink capacity to 20Mbps/200Mbps
@manager:/$ sn network --name sl_cons --gsl-up=20 --gsl-down=200
#(5)start the ENE, and run it for 3600 seconds
@manager:/$ sn start sl_cons --duration=3600 --delta=0.5
#(6)run user-specific program in all satellites in the first orbit
@manager:/$ sn cmd sl_cons.orbit[0] python /home/geo_routing.py
```

(c) Load configurations to initialize the ENE and run experiment.

Figure 4: A getting-started example of STARRYNET (`sn`).

vironment (*e.g.,* "sl_con") on multiple machines, configure network parameters (*e.g.,* uplink/downlink capacity), and run the user-specific program on emulated satellites (Figure 4c).

## 6 Framework Evaluation

In this section, we evaluate STARRYNET by exploring two important aspects related to the framework. **Q(i):** Can STARRYNET flexibly scale to various experimental requirements, with acceptable system and configuration overhead? **Q(ii):** How faithful are the results obtained by STARRYNET, as compared with other state-of-the-art simulators, and live network performance? Our framework evaluations are conducted on a typical enterprise cluster, including eight DELL R740 servers connected to a LAN. Each server is equipped with two Intel Xeon 5222 Processors (4-core, 3.8GHz for each processor), 8*32G DDR4 RAM, and Ubuntu20.04-LTS.

### 6.1 Ability to Satisfy Various Experimental Requirements for ISTNs

**Elastic scaling to various constellation configurations.** In reality, satellite operators *incrementally* deploy their satellite mega-constellations, which consist of multiple *shells*. As depicted in Table 2, STARRYNET is able to flexibly create a user-defined experiment environment for different shells, or multi-shell combinations of representative mega-constellations to satisfy various research requirements. The emulated constellation size can scale from about 300 (*e.g.,* the T1 shell of Telesat) to 4408 (*e.g.,* the full-scale Starlink Phase I with five shells) following different users' configurations.

**Environment setup overhead.** STARRYNET's APIs have concealed complex underlying processing for trajectory calculation and resource orchestration for the emulation. Thus a researcher can easily establish each ENE listed in Table 2, by writing about a dozen lines of code based on constellation prefabs (*e.g.,* like Figure 4b) predefined in STARRYNET's database. The creation time of a certain ENE upon STARRYNET tightly depends on the experiment scale, and the hardware capability of these machines used for experiments. Concretely, as shown in Table 2, the total creation time, including both node and link creations, increases as the constellation size scales up, and ranges from several minutes (for small size ENE) to tens of minutes (for large size ENE) in our current STARRYNET implementation.

**System overhead.** Table 2 also plots the average CPU and memory overhead consumed on each machine by running various ENEs. We make several observations. First, as expected, when the constellation size increases, STARRYNET requires more worker machines, consuming more CPU/memory resources to emulate ISTN nodes, links, and their constellation-wide dynamics. Second, if STARRYNET updates satellite dynamics more frequently (*i.e.,* with shorter update intervals), it consumes more resources to accomplish fine-granularity updates. Note that in this experiment we limit the CPU usage below $\Delta = 50\%$ in each machine. This is because in an ENE, the runtime overhead of the underlying STARRYNET should not use up all CPU/memory resources. It is reasonable to leave sufficient resources for the tested workloads and functionalities running upon the ENE.

### 6.2 Fidelity Analysis

Next we analyze the fidelity of STARRYNET by comparing the experiment results obtained by STARRYNET with live satellite networks and other state-of-the-art simulators.

**Network performance under a live Starlink topology.** We leverage STARRYNET to establish an ENE following the network topology of a recent live Starlink test conducted in Europe in 2021 [33]. Specifically, this real-world Starlink topology involves several key components as illustrated in Figure 5: (1) a user terminal together with a Starlink satellite dish located at the campus Klagenfurt Primoschgasse; (2) a SpaceX's ground station located in Frankfurt, Germany; (3) a Point of Presence (PoP) connecting the ground station to terrestrial Internet; and (4) a Web server deployed in Vienna. This experiment publicly reports the `ping` and `iperf` results measured between user terminal and the Web server, over the ISTN integrating Starlink satellites and terrestrial Internet. We use STARRYNET, Hypatia [60] and StarPerf [61] to generate network performance under the same topology configuration. The latter two are state-of-the-art ISTN simulators. Figure 6 plots the comparison for the latency results. First, we find that existing simulators *underestimate* the latency, since their latency estimations are based on a high-level abstraction without considering system effects like packet processing overhead. Second, STARRYNET achieves acceptable fidelity, as it attains similar latency performance in each case (*i.e.,* aver-

| Constellation \ Metrics | Height (km) | Constellation Size (number of satellites) | Creation Time (min) Nodes/Links/Total | | | Avg. CPU (%) Interval = 1/2/3 (s) | | | Avg. Memory (%) Interval = 1/2/3 (s) | | | Minimum # of Required Workers |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Starlink S1 (72*22, 53°) | 550 | 1584 | 5.9 | 4.6 | 10.5 | 7.2% | 7.0% | 6.3% | 3.9% | 3.5% | 3.4% | 2 |
| Starlink S2 (72*22, 53.2°) | 540 | 1584 | 5.9 | 4.6 | 10.5 | 7.2% | 7.0% | 6.3% | 3.9% | 3.5% | 3.4% | 2 |
| Starlink S3 (36*20, 70°) | 570 | 720 | 3.0 | 2.1 | 4.9 | 1.2% | 1.1% | 1.0% | 2.7% | 2.6% | 2.6% | 1 |
| Starlink S4 (6*58, 97.6°) | 560 | 348 | 1.9 | 1.3 | 3.2 | 1.0% | 1.0% | 1.0% | 2.7% | 2.6% | 2.4% | 1 |
| Starlink S5 (4*43, 97.6°) | 560 | 172 | 1.6 | 1.2 | 3.2 | 1.0% | 1.0% | 1.0% | 2.3% | 2.3% | 2.3% | 1 |
| Starlink Full (4408 satellites) | hybrid | 4408 | 13.3 | 7.9 | 21.2 | 39.6% | 37.0% | 34.3% | 10.4% | 9.1% | 8.9% | 7 |
| Kuiper K1 (34*34, 51.9°) | 630 | 1156 | 4.4 | 3.8 | 8.2 | 2.6% | 2.4% | 2.3% | 3.8% | 3.5% | 3.2% | 2 |
| Kuiper K2 (36*36, 42°) | 610 | 1296 | 4.7 | 4.2 | 8.9 | 3.9% | 3.6% | 3.2% | 4.0% | 3.6% | 3.5% | 2 |
| Kuiper K3 (28*28, 33°) | 590 | 784 | 3.2 | 2.4 | 5.6 | 1.3% | 1.2% | 1.2% | 2.7% | 2.6% | 2.6% | 2 |
| Kuiper Full (3236 satellites) | hybrid | 3236 | 5.7 | 4.8 | 10.5 | 24.6% | 23.9% | 23.2% | 6.3% | 6.2% | 6.2% | 6 |
| Telesat T1 (27*13, 98.98°) | 1015 | 351 | 1.9 | 1.3 | 3.2 | 1.0% | 1.0% | 1.0% | 2.6% | 2.5% | 2.4% | 1 |
| Telesat T2 (40*33, 50.88°) | 1325 | 1320 | 4.8 | 4.2 | 9.0 | 3.9% | 3.7% | 3.3% | 4.0% | 3.6% | 3.5% | 2 |
| Telesat Full (1671 satellites) | hybrid | 1671 | 3.1 | 2.4 | 5.5 | 7.2% | 7.0% | 6.4% | 4.2% | 3.7% | 3.6% | 3 |

Table 2: Ability to support mega-constellation emulation with various experimental configurations and system overheads.

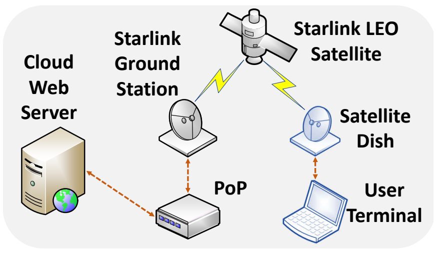


Figure 5: A live Starlink topology.

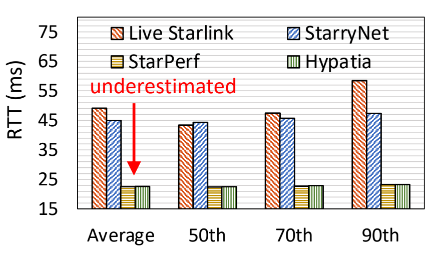


Figure 6: Latency comparison.

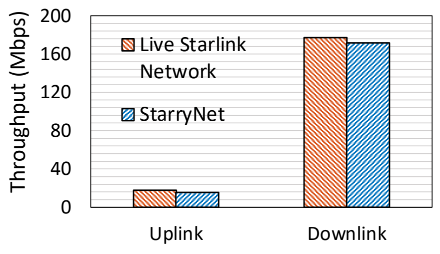


Figure 7: Throughput comparison.

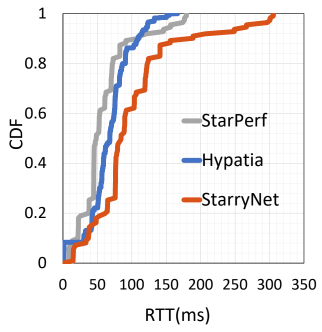


Figure 8: Latency comparison in an ISL-enabled topology.

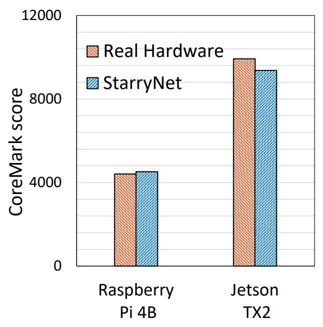


Figure 9: Flexible computation capability.

age/50th/70th/90th percentile) as compared with the real measured data from live Starlink. This is because STARRYNET jointly combines model calculation, data-driven calibration and real networking stack to create the ENE.

Bandwidth is a metric that can be affected by many operational factors. Therefore, in a research experiment STARRYNET allows the researcher to manually configure the link capacity on demand. For example, we follow the realistic Starlink trace in [33] to set the uplink/downlink capacity, and run `iPerf` to measure the TCP throughput in each direction. Since Hypatia and StarPerf can not load real network traffic by `iPerf`, we compare the throughput results of live Starlink and STARRYNET. Specifically, evaluation results in Figure 7 demonstrate that STARRYNET can be tuned to accurately emulate the bandwidth of a live ISTN.

**Network performance under an ISL-enabled topology.** As of the date of this paper submission, most real mega-constellations like Starlink and Kuiper are still in their early stage and under heavy construction. Although Starlink has started to deploy laser ISLs on its LEO satellites, those ISLs are still under internal test, and it is difficult to directly compare the network performance estimated by STARRYNET with a real ISL-enabled satellite network. To analyze the fidelity of STARRYNET when ISLs are activated, we compare the performance results obtained by STARRYNET with other ISTN simulators. Figure 8 plots the CDF of latency between a collection of real ground-station pairs [26] with the same constellation configuration based on the ISL-enabled Starlink network. The latency results of STARRYNET are measured by `ping` test in the emulated ENE, while the results of other simulators are generated by numeric or event-driven calculation. As shown in Figure 8 the latency obtained by STARRYNET is slightly higher than other simulators, because STARRYNET incorporates realistic system-level overhead (*e.g.,* packet processing) which could be neglected in simulators.

**On-demand computation capability.** Researchers may need to conduct their experiments on different satellite hardware with various computation capabilities. For example, authors in [53] studied the application performance achieved by two space-grade processors RAD-5545 [21] and HPSC [13]. Recent works like [23, 44] explored new satellite functionalities running upon commercial low-power processor such as Raspberry Pi [24] and Jetson TX2 [18]. STARRYNET is able to flexibly adjusting the computation capability on each emulated satellite to satisfy various experimental requirements. To validate the computational flexibility, we use `CoreMark` [5], a well-known processor benchmark to measure the performance of the real hardware and its facsimile created by StarryNet. As plotted in Figure 9, CoreMark score is a metric that quantifies the computation capability. Higher scores indicate stronger computing capability. For various computation requirements, StarryNet can mimic similar processor capability based on concrete experimental requirement.

## 7 Evaluating Futuristic ISTN Research with StarryNet: Case Studies

Next we conduct several case studies to show how StarryNet can be used to advance futuristic ISTN research.

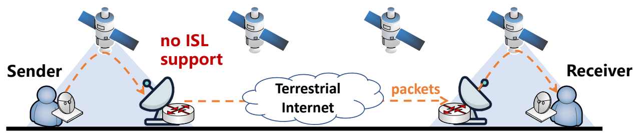
### 7.1 Exploring the Design Space of Integrating LEO Satellites and Terrestrial Facilities

To realize the promise of low-latency and pervasive accessibility of ISTN, the first step should be interconnecting LEO satellites and terrestrial facilities (*e.g.,* ground stations and user terminals). While many existing studies have proposed different space-ground topology designs, it still lacks a systematically analysis and comparison on these integration paradigms, in terms of their network performance and corresponding cost. We leverage StarryNet to explore how different design choices of space-ground integration (as illustrated in Figure 10) could affect the performance and cost of an ISTN.
**(1) Satellite relays for last-mile accessibility (SRLA).** Satellites and ground facilities can be integrated based on the classic "bent-pipe" architecture to provide last-mile network accessibility as illustrated in Figure 10a, which is the *status-quo* of many today's satellite constellations like OneWeb. Data from the ground are first transmitted to the satellite, which then sends it right back down again like a bent pipe. The only processing performed by satellites is to retransmit the signals.
**(2) Satellite relays for ground station networks (SRGS).** Figure 10b depicts another "bent-pipe"-based integration paradigm originally introduced in [57], where geo-distributed ground stations work as routers to construct a network. Each satellite switches packets between two ground stations connected to the satellite. Packets from the sender are routed to the receiver by paths built upon satellites and ground stations.
**(3) Ground station gateway for satellite networks (GSSN).** Figure 10c shows an ISL-based internetworking approach proposed by [52]. ISL-enabled satellites can build space routes for long-haul communication, without the need of a large number of ground station relays, as well as user-side satellite dishes. Satellites and ground stations build a Layer-3 network. During an end-to-end transmission, packets from the sender are first routed to a ground station via terrestrial Internet, then to the receiver side ground station over ISL-enabled satellites, and finally to the receiver over the terrestrial Internet.
**(4) Directly accessed satellite networks (DASN).** Figure 10d plots a paradigm where users' satellite dishes directly connect to ISL-enabled satellite networks, and two users can establish long-haul communication without the assistance of geo-distributed ground stations [51, 56]. Satellites work not only as routers, but also as access points/gateways allocating addresses for different terrestrial users.
**Experiment setup.** We leverage StarryNet to build an ENE for each paradigm, analyze their cost, and evaluate their network performance. Specifically, we establish an ENE based on Starlink's first constellation shell and its ground station distribution. We randomly pick geo-distributed user-pairs and establish communication sessions between these pairs over the satellite network. On each emulated satellite, we load `BIRD` [37] routing software and run `OSPF` as the routing protocol to achieve the shortest path for data transmission.
**Observations.** Table 3 summarizes the average end-to-end latency and the latency breakdown of different space-ground topology designs. We observe an obvious latency reduction accomplished by laser ISLs, and DASN obtains the lowest end-to-end latency on average. Since ground stations typically can not be deployed upon oceans (70% earth surface), SRGS suffers from the highest latency as compared with other schemes due to the insufficient deployment of ground stations.

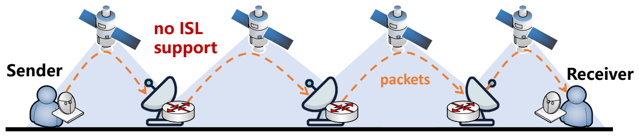
As satellites move, two main factors affect the end-to-end network reachability. First, users in certain regions may lose available satellite access due to the LEO dynamics. For example, users in high latitude areas may suffer from intermittent satellite access. Second, frequent connectivity changes can trigger *routing re-convergence*. As plotted in Table 3, SRGS suffers from the lowest reachability due to the combination of LEO dynamics and limited coverage of insufficient ground stations. GSSN obtains low reachability because frequent satellite-ground handovers result in continuous re-convergence, during which routes are fluctuating and unstable.

For SRLA, SRGS and GSSN, the IP address of user's satellite dish is allocated by the fixed user-side ground station, and the addresses of senders or receivers do not change during the communication. However, for DASN, each satellite works not only as a router, but also as an access point/a gateway which allocates IP addresses for terrestrial dishes connect to it. Due to the LEO dynamics, terrestrial dishes have to frequently change access satellite as well as their subnet. Consequently, addresses are frequently updated, which may further disrupt high layer transport connections and application sessions.

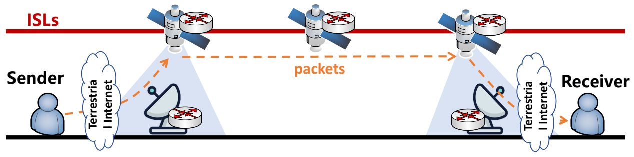
The above four topologies have different deployment and operating costs in addition to LEO satellites. SRLA and SRGS require a large number of available ground stations near users to guarantee continuous satellite coverage. For SRGS, it also requires sufficient geo-distributed ground stations to ensure stable and low-latency routes over satellites and ground stations. GSSN and DASN require the extra deployment of ISLs. Users in SRLA, SRGS and DASN have to purchase a dedicated dish to access satellites. In GSSN, users connect to the ground station gateway via terrestrial networks, and do not need to install additional satellite dishes.
**Implications.** As summarized in Table 3, there is no clear win-



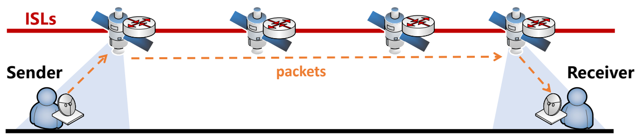
(a) **SRLA**: satellite relays for last-mile accessibility.



(b) **SRGS**: satellite relays for ground station networks.



(c) **GSSN**: ground station access for satellite networks.



(d) **DASN**: satellite networks directly accessed by terrestrial users.

Figure 10: The design-space for various space-ground integration methodologies.

| Design | Average end-to-end latency and its breakdown (ms) | | | | Reachability | Frequent Address Update | Operating Cost | | |
|---|---|---|---|---|---|---|---|---|---|
| | Inter-Satellite | Space-Ground | Ground | Toal | | | GS | Terminal | ISLs |
| SRLA | 0 | 76.25 | 107 | 183.25 | 97.00% | ✗ | ✓ | ✓ | ✗ |
| SRGS | 0 | 313.39 | 0 | 313.39 | 51.00% | ✗ | ✓ | ✓ | ✗ |
| GSSN | 48.46 | 38.45 | 20 | 106.91 | 57.40% | ✗ | ✓ | ✗ | ✓ |
| DASN | 48.46 | 37.65 | 0 | 86.11 | 97.50% | ✓ | ✗ | ✓ | ✓ |

Table 3: Comparison for different space-ground integration methodologies.

ner for all four integration methodologies. "Bent-pipe"-based approaches achieve simplicity without ISL requirements, but they fail to fully unleash the low-latency potential of ISTNs. Approaches relying on ISLs can form near-optimal spaces routes to attain wide-area low-latency communication, but they involve extra ISL cost, and suffer from higher route instability and connection disruptions, due to the route re-convergence and address update caused by LEO dynamics. All integration approaches have their limitations, and satellite operators are suggested to choose a proper topology based on their specific performance requirements and cost budgets.

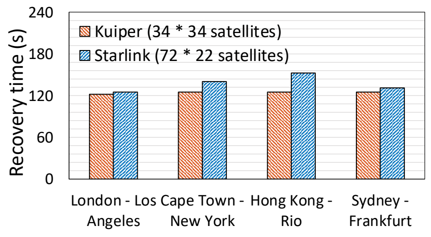
## 7.2 Evaluating ISTN Resilience

Satellites are operated in complex outer space environments. Small satellites deployed in emerging mega-constellations typically have a short lifetime (*e.g.,* 3-5 years [30]) due to their low manufacturing cost. Many space factors or events, such as space debris [15], geomagnetic storms [12], and single event upset [28], *etc.*, can cause immediate satellite failures. For example, in February 8, 2022, about 40 Starlink satellites are doomed by a geomagnetic storm [1]. Therefore, given the harsh and error-prone space environment, it should be important for satellite operators and researchers to evaluate and analyze how resilient an ISTN is, and what kind of system/network factors affect the resilience.

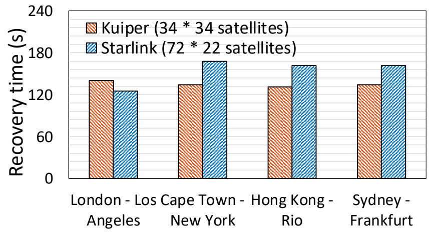
**Experiment setup.** We thus conduct an experiment with STARRYNET to evaluate the network resilience with different routing configurations. Specifically, we mimic the impact of a space failure (*e.g.,* due to a geomagnetic storm) which makes a fraction of satellites in the constellation inactive and can not forward network traffic. We load `BIRD` [37] in our ENE and run `OSPF` as the routing protocol in this experiment.

**Observations.** Figure 11 plots the routing recovery time for a set of representative city-pairs under various failure ratios. An on-path satellite failure can cause a network disruption, and the routing recovery time increases as the constellation size and failure rate increase. Figure 12 plots the comparison for the end-to-end latency before the constellation failure and after the routing re-convergence. We observe that the latency increases slightly under low failure rate, and the latency dramatically increases when the failure rate reaches 30%.

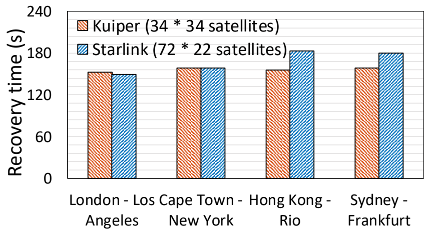
**Implications.** We summarize three key implications from this experiment. First, we find the mesh-like network based on a large number of satellites can maintain low latency in case of low failure rate. This is because the mesh-like satellite network has high *path diversity*, offering backup routes for communication pairs in case of network failures. Second, the inherent high dynamicity of LEO satellites is a double-edge sword for the service restoration in an ISTN. On one hand, for terrestrial users whose access satellite above them fails, the dynamicity helps because faulty satellites will soon move out of their line-of-sight. On the other hand, the dynamicity hurts, as it spreads the failure globally, and could affect the network accessibility of other users. Finally, while improving redundancy in physical connectivity and applying robust mechanisms in protocol design are two critical directions to improve the ISTN resilience, it is challenging to attain a "win-win" integration of them in practical systems. Increasing the satellite density indeed improves the resilience of satellite accessibility in case of sudden failures, but it also involves much more nodes and links in the network, and thus imposes new challenges and requirements on the protocol scalability and recovery efficiency under various failure scenarios.



(a) Route recovery time under 10% failure ratio.

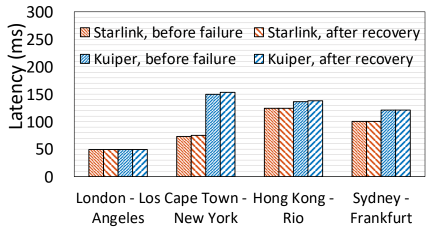


(b) Route recovery time under 20% failure ratio.

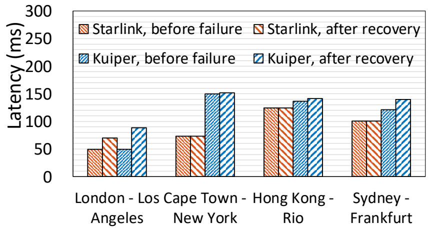


(c) Route recovery time under 30% failure ratio.

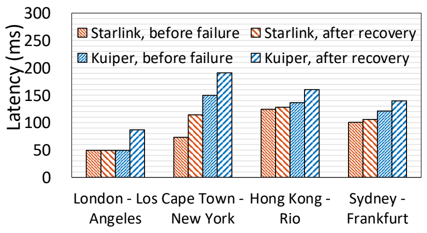
Figure 11: Route recovery time under different constellation-wide failures.



(a) Increased latency under 10% failure ratio.



(b) Increased latency under 20% failure ratio.



(c) Increased latency under 30% failure ratio.

Figure 12: Increased latency after route reconvergence under various constellation-wide network failures.

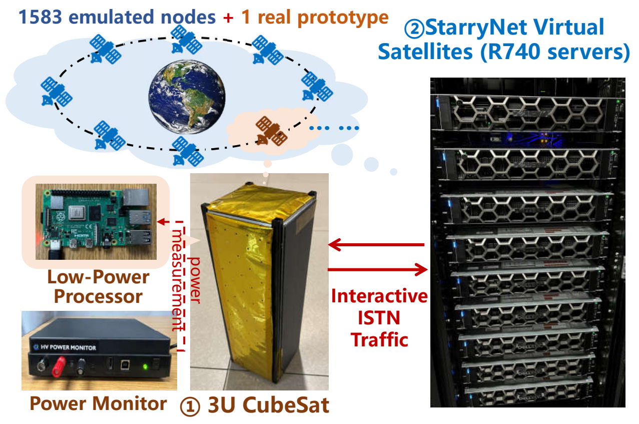


Figure 13: Hardware-in-the-loop testing with STARRYNET.

| State | Idle | Routing convergence | Data transmission rate (Mbps) | | | | |
|---|---|---|---|---|---|---|---|
| | | | 100 | 250 | 500 | 750 | 1000 |
| Power consumption (W) | 2.83 | 3.22 | 4.6 | 4.99 | 5.36 | 5.45 | 5.46 |

Table 4: Power consumption under various ISTN workloads.

## 7.3 Hardware-in-the-loop Testing

In real satellite deployments, it is very important to accurately estimate how much energy a new system or network function will consume before the launch. STARRYNET enables researchers to conduct *hardware-in-the-loop* testing to accurately evaluate the low-level system effects under various workloads. As a case study to demonstrate STARRYNET's ability, we connect a 3U CubeSat prototype, equipped with a low-power processor [23,24] running real routing protocols to the virtual satellite network emulated by STARRYNET, as illustrated in Figure 13. Collectively, the 3U CubeSat prototype together with the emulation creates a virtual constellation network with 1584 Starlink satellites. We follow the satellite traffic model proposed in [51] to inject traffic and use a power monitor to measure the satellite prototype. Table 4 summarizes the power consumption in different states (*e.g.,* calculating route convergence and forwarding traffic in various data rates). Our hardware-in-the-loop test demonstrates STARRYNET's ability to create a hybrid ENE and evaluate real system effects for user-defined functionalities.

## 8 Limitation and Future Work

**Experimental scope and limitations.** Our STARRYNET framework mainly targets at various network-level experiments for ISTNs, *e.g.,* evaluating a new routing/transport-layer protocol, or assessing the network performance of a new topology design in a highly-dynamic, resource constrained virtual ISTN environment. The scale of the experiment supported by STARRYNET is closely related to the underlying resources provided by physical machines. In its present form, the key limitation of STARRYNET can be summarized as follows. First, STARRYNET is essentially a data-driven framework combining constellation-relevant modeling and network emulation. Thus its fidelity tightly depends on the availability and accuracy of the public information shared by the satellite ecosystem. For example, in practice, public TLE data may provide inaccurate orbit information, which can have errors up to 12 km, and such errors can affect the calculation of network performance (*e.g.,* inter-satellite visibility and propagation delay). Second, some parameters are hard to obtain from a practical satellite system today, because most mega-constellations are still in their early stage with limited access. For example, it is difficult to obtain the real ISL-enabled Starlink performance right now, since Starlink's laser ISLs are still under internal test. Thus, STARRYNET allows researchers to manually configure the ISL parameters (*e.g.,* link capacity) and customize their experiments based on various experimental requirements. Third, our framework is primarily based on virtualization-based network-level emulation, and thus it has limited ability to emulate physical layer (PHY) characteristics that can be observed in a live network experiment, *e.g.,* spectrum adaptation and multiplexing [42], or the time consumed by a real satellite dish to detect PHY connectivity changes.

**Future work.** Satellite Internet mega-constellations are still evolving rapidly. New constellation designs are constantly being proposed, and existing constellation schemes are constantly being updated. In our future work, we will follow the evolution and deployment of realistic satellite Internet constellations. In particular, we will track the latest constellation information to update STARRYNET's open database, calibrate the constellation models and further improve the fidelity of the STARRYNET framework. Moreover, based on these implications obtained from our case studies in §7, we will explore new network techniques tailored for ISTNs, *e.g.,* practical and resilient satellite routing protocols in the future. Our latest research progress on STARRYNET will be updated on the website: https://github.com/SpaceNetLab/StarryNet.

## 9 Related Work

Section 3.2 has discussed most existing efforts relevant to the method of building ENEs. In this section we briefly introduce other ISTN works related to our study in this paper.

The network community has many recent efforts studying on the topology design [40, 58], routing [47, 52, 56, 57, 83], transport-layer congestion control [60, 64], new satellite applications [62] and security issues [51] for emerging ISTNs. For instance, Motif [40] is a recent topology design for LSNs, in which each satellite is dynamically connected to other visible satellites to achieve low latency under various traffic configurations. Works in [40, 56, 57] suggest the use of pre-calculated shortest-path-based routing and traffic engineering schemes for ISTNs. On one hand, these pioneering studies indeed outline the promising network potential of futuristic ISTNs. On the other hand, the above new thoughts are evaluated by simulations with a high-level abstraction. STARRYNET can stimulate new research and advance existing ISTN works by evaluating them in a more realistic ENE to obtain practical insights for further optimizations.

The rapid evolution of ISTNs also attracted the attention of the system community. Specifically, orbital edge computing (OEC) [43, 44, 66] is a new computation architecture which leverages computational satellites to pre-process earth observation (EO) data, and save the data download overhead. Studies in [49, 50] explored the feasibility of applying deep neural networks to process on-board satellite data. Since STARRYNET creates real system runtime and networking stack in an experimental ISTN environment, it can also help to evaluate system-level effects of these new algorithms, implementations and programming models designed for ISTNs.

## 10 Conclusion

To advance futuristic research on ISTNs, this paper presents STARRYNET, an experimentation framework that empowers researchers to conventionally and flexibly build ENEs for ISTN research. STARRYNET simultaneously achieves constellation-consistency, network realism, and flexibility, by integrating real constellation-relevant information, orbit analysis and large-scale emulations to construct ENEs. By comparing STARRYNET's results with live network performance and conducting diverse case studies, we demonstrate STARRYNET's fidelity and flexibility for various ISTN experiments. We are confident that the open-source STARRYNET can help the network and system community to flexibly conduct various ISTN evaluations with more credible results.

## Acknowledgments

We thank our shepherd Behnaz Arzani and all anonymous NSDI reviewers for their feedback which greatly improved the paper. This work was partially supported by the National Natural Science Foundation of China (NSFC No. 62132004) and Tsinghua University-China Mobile Communications Group Co., Ltd. Joint Institute.

## References

[1] 40 Starlink satellites doomed by geomagnetic storm. https://earthsky.org/space/40-starlink-satellites-doomed-by-geomagnetic-storm/.

[2] Application of Kuiper Systems LLC for Authority to Launch and Operate a Non-Geostationary Satellite Orbit System in Ka-band Frequencies. https://licensing.fcc.gov/myibfs/download.do?attachment_key=1773885.

[3] Azure Orbital: Satellite ground station and scheduling service connected to Azure for fast downlinking of data. https://azure.microsoft.com/en-us/services/orbital/.

[4] China's megaconstellation project establishes satellite cluster in chongqing. https://spacenews.com/chinas-megaconstellation-project-establishes-satellite-cluster-in-chongqing/.

[5] Coremark: a benchmark designed specifically to test the functionality of a processor core. https://www.eembc.org/coremark/.

[6] Docker: Empowering app development for developers. https://www.docker.com/.

[7] Emulab: a time- and space-shared platform for research, education, and development in distributed systems and networks. https://www.emulab.net/.

[8] FCC authorizes boeing broadband satellite constellation. https://www.fcc.gov/document/fcc-authorizes-boeing-broadband-satellite-constellation.

[9] FCC Authorizes Kuiper Satellite Constellation. https://docs.fcc.gov/public/attachments/FCC-20-102A1.pdf.

[10] FCC International Bureau Filings. https://fcc.report/IBFS/.

[11] General mission analysis tool. https://gmat.atlassian.net/wiki/spaces/GW/overview?mode=global.

[12] Geomagnetic storm. https://en.wikipedia.org/wiki/Geomagnetic_storm.

[13] High performance spaceflight computing (HPSC). https://www.nasa.gov/directorates/spacetech/game_changing_development/projects/HPSC/.

[14] HPE spaceborne computer. https://www.hpe.com/us/en/compute/hpc/supercomputing/spaceborne.html.

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

[16] Norad two-line element sets current data. https://www.celestrak.com/NORAD/elements/.

[17] NTP: The Network Time Protocol. http://www.ntp.org/.

[18] Nvidia Jetson TX2 Module. https://developer.nvidia.com/embedded/jetson-tx2.

[19] Open vswitch manual. http://www.openvswitch.org/support/dist-docs/ovs-vsctl.8.txt.

[20] PLANETLAB: an open platform for developing, deploying and accessing planetary-scale services. https://planetlab.cs.princeton.edu/.

[21] RAD5545 SpaceVPX single-board computer. Multi-core single-board computer. https://www.baesystems.com/en-media/uploadFile/20210404061759/1434594567983.pdf.

[22] RAD750 family of radiation-hardened products. https://www.baesystems.com/en-media/uploadFile/20210404044504/1434555689265.pdf.

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

[24] Raspberrypi fundation. https://www.raspberrypi.org/.

[25] Roscosmos for space flights, cosmonautics programs, and aerospace research. http://en.roscosmos.ru/.

[26] SatNOGS – Open Source global network of satellite ground stations. https://network.satnogs.org/.

[27] Scrapy. https://scrapy.org/.

[28] Single-event upset. https://en.wikipedia.org/wiki/Single-event_upset.

[29] Skyfield. https://rhodesmill.org/skyfield/.

[30] SpaceX is Giving the Internet Lift With Starlink. https://subspace.com/resources/spacex-is-giving-the-internet-lift-with-starlink.

[31] SpaceX's Starlink user terminal. https://arstechnica.com/information-technology/2020/12/teardown-of-dishy-mcflatface-the-spacex-starlink-user-terminal/.

[32] Speed check: Starlink performance. https://www.speedcheck.org/starlink-performance-2021/.

[33] Starlink analysis at the carinthia university of applied sciences (CUAS). https://forschung.fh-kaernten.at/roadmap-5g/files/2021/07/Starlink-Analysis.pdf.

[34] Starlink: high-speed, low latency broadband Internet. https://www.starlink.com/.

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

[36] Telesat. https://www.telesat.com/.

[37] The BIRD Internet Routing Daemon. https://bird.network.cz/.

[38] UCS satellite database. https://www.ucsusa.org/resources/satellite-database.

[39] Amazon. AWS Ground Station. https://aws.amazon.com/ground-station/.

[40] D. Bhattacherjee and A. Singla. Network topology design at 27,000 km/hour. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies (CoNEXT)*, pages 341–354. ACM, 2019.

[41] K. C. Castonguay. Additive manufacture of propulsion systems in low earth orbit. Technical report, Air Command and Staff College, Air University Maxwell AFB United States, 2018.

[42] R. Chen and W. Gao. TransFi: emulating custom wireless physical layer from commodity wifi. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services (MOBISYS)*, 2022.

[43] B. Denby and B. Lucia. Orbital edge computing: Machine inference in space. *IEEE Computer Architecture Letters*, 18(1):59–62, 2019.

[44] B. Denby and B. Lucia. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, page 939–954. ACM, 2020.

[45] G. Di Lena, A. Tomassilli, D. Saucez, F. Giroire, T. Turletti, and C. Lac. Distrinet: a mininet implementation for the cloud. *ACM SIGCOMM Computer Communication Review*, 51(1):2–9, 2021.

[46] L. Dreyer. Latest developments on SpaceX's Falcon 1 and Falcon 9 launch vehicles and Dragon spacecraft. In *Aerospace conference*. IEEE, 2009.

[47] E. Ekici, I. F. Akyildiz, and M. D. Bender. Datagram routing algorithm for leo satellite networks. In *Proceedings of International Conference on Computer Communications (INFOCOM)*, volume 2, pages 500–508 vol.2. IEEE, 2000.

[48] W. Frick and C. Niederstrasser. Small launch vehicles - A 2018 state of the industry survey.

[49] G. Giuffrida, L. Diana, F. de Gioia, G. Benelli, G. Meoni, M. Donati, and L. Fanucci. Cloudscout: a deep neural network for on-board cloud detection on hyperspectral images. *Remote Sensing*, 12(14):2205, 2020.

[50] G. Giuffrida, L. Fanucci, G. Meoni, M. Batič, L. Buckley, A. Dunne, C. van Dijk, M. Esposito, J. Hefele, N. Vercruyssen, G. Furano, M. Pastena, and J. Aschbacher. The sat-1 mission: The first on-board deep neural network demonstrator for satellite earth observation. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–14, 2022.

[51] G. Giuliari, T. Ciussani, A. Perrig, and A. Singla. Icarus: Attacking low earth orbit satellite networks. In *USENIX Annual Technical Conference (ATC)*, pages 317–331. USENIX, 2021.

[52] G. Giuliari, T. Klenze, M. Legner, D. Basin, A. Perrig, and A. Singla. Internet backbones in space. *ACM SIGCOMM Computer Communication. Review.*, 50(1):25–37, Mar. 2020.

[53] E. W. Gretok, E. T. Kain, and A. D. George. Comparative benchmarking analysis of next-generation space processors. In *Aerospace Conference*. IEEE, 2019.

[54] D. Gupta, K. V. Vishwanath, M. McNett, A. Vahdat, K. Yocum, A. Snoeren, and G. M. Voelker. Diecast: Testing distributed systems with an accurate scale model. *ACM Transactions on Computer Systems (TOCS)*, 29(2):1–48, 2011.

[55] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown. Reproducible network experiments using container-based emulation. In *Conference on emerging Networking Experiments and Technologies (CoNEXT)*. ACM, 2012.

[56] M. 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.

[57] M. 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. ACM, 2019.

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

[59] Internet World Stats. World internet usage and population statistics. `https://www.internetworldstats.com/stats.htm`, 2021.

[60] S. Kassing, D. Bhattacherjee, A. B. Águas, J. E. Saethre, and A. Singla. Exploring the "Internet from Space" with Hypatia. In *Proceedings of the Internet Measurement Conference (IMC)*, page 214–229. ACM, 2020.

[61] Z. Lai, H. Li, and J. Li. StarPerf: Characterizing Network Performance for Emerging Mega-Constellations. In *28th International Conference on Network Protocols (ICNP)*. IEEE, 2020.

[62] Z. Lai, W. Liu, Q. Wu, H. Li, J. Xu, and J. Wu. SpaceRTC: Unleashing the Low-latency Potential of Mega-constellations for Real-Time Communications. In *Proceedings of International Conference on Computer Communications (INFOCOM)*, pages 1339–1348. IEEE, 2022.

[63] Z. Lai, Q. Wu, H. Li, M. Lv, and J. Wu. OrbitCast: Exploiting Mega-Constellations for Low-Latency Earth Observation. In *29th International Conference on Network Protocols (ICNP)*. IEEE, 2021.

[64] X. Li, F. Tang, J. Liu, L. T. Yang, L. Fu, and L. Chen. AUTO: Adaptive congestion control based on multi-objective reinforcement learning for the satellite-ground integrated network. In *USENIX Annual Technical Conference (ATC)*, pages 611–624. USENIX Association, 2021.

[65] T. M. Lovelly. *Comparative Analysis of Space-Grade Processors*. PhD thesis, University of Florida, 2017.

[66] B. Lucia, B. Denby, Z. Manchester, H. Desai, E. Ruppel, and A. Colin. Computational nanosatellite constellations: Opportunities and challenges. *GetMobile: Mobile Computing and Communications*, 25(1):16–23, 2021.

[67] man7.org. tc(8) — Linux manual page. https://man7.org/linux/man-pages/man8/tc.8.html.

[68] MININET. An Instant Virtual Network on your Laptop (or other PC). http://mininet.org/.

[69] M. K. Mukerjee, C. Canel, W. Wang, D. Kim, S. Seshan, and A. C. Snoeren. Adapting TCP for reconfigurable datacenter networks. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, 2020.

[70] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan. Mahimahi: Accurate record-and-replay for HTTP. In *USENIX Annual Technical Conference (ATC)*. USENIX Association, 2015.

[71] W. M. Organization. Observing systems capability analysis and review tool. https://space.oscar.wmo.int/satellites.

[72] D. Pediaditakis, C. Rotsos, and A. W. Moore. Faithful reproduction of network experiments. In *Proceedings of the tenth ACM/IEEE symposium on Architectures for networking and communications systems (ANCS)*, pages 41–52, 2014.

[73] S. Services. Petition of Starlink Services, LLC for Designation as an Eligible Telecommunications Carrier. https://www.mass.gov/doc/dtc-21-1-starlink-final-order/download, 2021.

[74] V. Singh, A. Prabhakara, D. Zhang, O. Yağan, and S. Kumar. A community-driven approach to democratize access to satellite ground stations. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking (MOBICOM)*. ACM, 2021.

[75] A. Singla. SatNetLab: a call to arms for the next global internet testbed. *ACM SIGCOMM Computer Communication Review*, 51(2):28–30, 2021.

[76] SNS3. Satellite network simulator 3. https://www.sns3.org/content/home.php.

[77] D. Vasisht and R. Chandra. A Distributed and Hybrid Ground Station Network for Low Earth Orbit Satellites. In *Proceedings of the 19th Workshop on Hot Topics in Networks (HotNets)*, page 190–196. ACM, 2020.

[78] D. Vasisht, J. Shenoy, and R. Chandra. L2D2: low latency distributed downlink for LEO satellites. In *Proceedings of the ACM SIGCOMM Conference*, pages 151–164. ACM, 2021.

[79] E. Weingärtner, F. Schmidt, H. Vom Lehn, T. Heer, and K. Wehrle. SliceTime: A Platform for Scalable and Accurate Network Emulation. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, 2011.

[80] P. Wette, M. Dräxler, A. Schwabe, F. Wallaschek, M. H. Zahraee, and H. Karl. Maxinet: Distributed emulation of software-defined networks. In *IFIP Networking Conference*, 2014.

[81] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. *ACM SIGOPS Operating Systems Review*, 36(SI):255–270, 2002.

[82] L. Wood. Satellite Constellation Design for Network Interconnection Using Non-Geo Satellites., 2002. https://openresearch.surrey.ac.uk/esploro/outputs/bookChapter/Appendix-A-Satellite-Constellation-Design-for/99513302802346.

[83] Y. Wu, Z. Yang, and Q. Zhang. A Novel DTN Routing Algorithm in the GEO-Relaying Satellite Network. In *The 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, pages 264–269, 2015.

[84] F. Y. Yan, J. Ma, G. D. Hill, D. Raghavan, R. S. Wahby, P. Levis, and K. Winstein. Pantheon: the training ground for Internet congestion-control research. In *USENIX Annual Technical Conference (ATC)*. USENIX Association, 2018.

[85] Y. Zheng and D. M. Nicol. A virtual time system for openvz-based network emulations. In *Workshop on Principles of Advanced and Distributed Simulation*. IEEE, 2011.