SERENADE: A Digital Twin Emulator for LEO Satellite Networking At-Scale

Roberto Chamorro Martinez
Georgia Institute of Technology
Atlanta, GA, USA
r.chamorro@gatech.edu

N. Cameron Matson
Georgia Institute of Technology
Atlanta, GA, USA
ncmatson@gatech.edu

Karthikeyan Sundaresan
Georgia Institute of Technology
Atlanta, GA, USA
karthik@ece.gatech.edu

Abstract

In this work we propose SERENADE: a large-scale, real-time, and flexible emulator for satellite networking. While existing tools rely on heavier container-based node emulation, SERENADE is written in Go, and models nodes using ultra-lightweight threads. This design enables SERENADE to realistically emulate up to several 100,000s of nodes (both satellite and ground nodes) on a single machine, a first of its kind. We ensure that scale does not come at the expense of responsiveness; SERENADE starts up in a few seconds and tracks satellite trajectories in real time (rather than being precomputed), making it ultra-responsive to changes on-the-fly. Additionally, SERENADE is designed to interface easily with external applications, allowing for arbitrary and dynamic network inputs and measurements. These features make it the first satellite network emulator suitable to operate as a digital twin for satellite networking. Our evaluations highlight SERENADE's ability to efficiently scale while maintaining a high degree of networking realism and remaining adaptable. We demonstrate SERENADE's digital twin capabilities through several real-world case studies, including disaster relief and satellite constellation updates. We also use SERENADE to illustrate how many proposed frameworks for user-satellite association fail to effectively scale to large deployment scenarios and result in grossly under-utilized network resources.

CCS Concepts

• **Networks** → **Network emulation**; *Wireless networks*; *Network performance analysis*; • **Computing methodologies** → **Model development and analysis**.

Keywords

LEO Satellite Networks, Network Emulation, Digital Twin, Starlink, Go, Scalability

ACM Reference Format:
Roberto Chamorro Martinez, N. Cameron Matson, and Karthikeyan Sundaresan. 2026. SERENADE: A Digital Twin Emulator for LEO Satellite Networking At-Scale. In *ACM/IEEE International Conference on Embedded Artificial Intelligence and Sensing Systems (SenSys '26), May 11–14, 2026, Saint Malo, France.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3774906.3802777

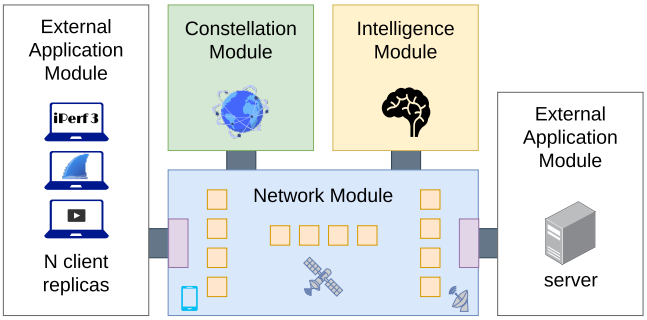


Figure 1: Container-level architecture for SERENADE

1 Introduction

In recent years, there has been a rapid proliferation of satellites in Low Earth Orbit (LEO). One of the primary advantages of LEO over other, higher-altitude orbits, is that its proximity to Earth results in low enough latencies to enable broadband networks. While SpaceX's Starlink [11] is the current leader in LEO Satellite Networks (LSN) for communication systems, there are other (either already launched or planned) constellations as well (OneWeb [6], Iridium [15], Telesat [7], Amazon Kuiper [45]). These deployments and others (very low earth orbit [37], cis-lunar systems [19]) represent a new frontier for a range of applications from personal [16, 27], industrial [10, 12, 18] and governmental [22, 39] use cases.

With these quickly growing deployments, there is a need for rapid innovation in this "space". However, outside of a few select entities (i.e., the commercial enterprises deploying the LSNs) there is little opportunity to conduct networking experiments. Many researchers have attempted to reverse engineer the popular Starlink network by purchasing user terminals and conducting "outside-in" experiments [26, 38]. These approaches can yield interesting insights, but they tend to be descriptive rather than innovative. An alternative is to rely on simulations, but this lacks the realism of real networks and traffic. A middle ground is emulation, which attempts to combine simulation of satellite constellation dynamics with real (virtualized) networking platforms. A LSN emulator must realistically model *both* the LEO satellite aspects (i.e., satellite mobility, global scale, etc.) and the networking aspects (i.e., traffic dynamics, realistic congestion, etc.).

A related but distinct tool is the Digital Twin (DT). DTs are a framework that pairs a real-world system with a virtual counterpart and enables a continuous flow of information between the two. Data from the real-world flows to the virtual one either to serve as input or to update models of the underlying real system; these models can then be used to conduct experiments or drive

SenSys '26, Saint Malo, France

ACM ISBN 979-8-4007-2309-4/26/05
https://doi.org/10.1145/3774906.3802777

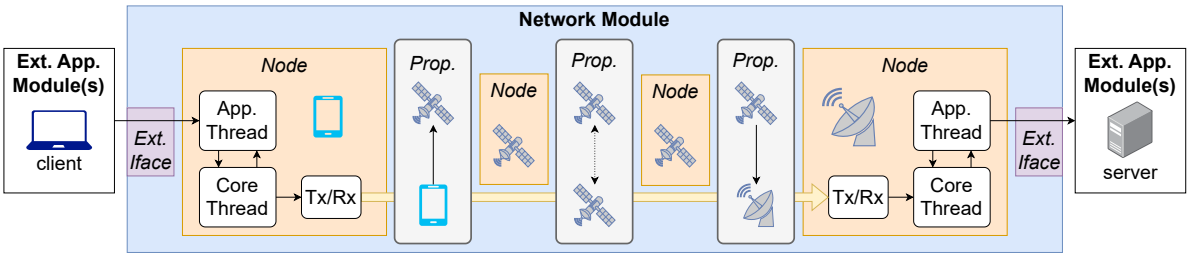


Figure 2: Flow of traffic through SERENADE

decision making to update the real-world system. Originally from the domain of industrial systems and manufacturing [24], DTs have recently been applied to many fields [32, 34, 48] including communication networks [17]. Since there is an immense cost in deploying, operating, and updating LSNs, Network Digital Twins (NDTs) can be leveraged to reduce the deployment risk by enabling controlled "what-if" testing, providing significant value for LSN operators.

DTs go beyond emulators in that they must also be responsive to real-world dynamics. To function as a true NDT, an LSN emulator must meet three key requirements. First, it must support **large-scale** deployment, encompassing thousands of satellites *and* ground nodes to capture full-system interactions. Second, it must reproduce **realistic** traffic dynamics across *all* nodes, since collective behaviors, such as congestion, routing adaptation, and load balancing, emerge only when the network operates at scale. Finally, the emulator must remain **responsive**, enabling rapid reconfiguration and continuous operation without lengthy startup or reinitialization times, as any tool that requires minutes or hours to restart cannot function as a true digital twin.

While existing satellite emulators (e.g., [28, 30, 49]) have advanced the state of the art, they remain unable to deliver the scale and responsiveness required of a true LSN digital twin, as they are constrained by the heavy virtualization layers in which their designs are based. To bridge this gap, we introduce SERENADE[1], a lightweight and massively scalable emulator designed from scratch in Go [14]. Its architecture replaces heavy virtualization layers (e.g., containers) with lightweight threads (*goroutines*), where each network node (i.e., satellite, gateway, or user terminal) is represented by a group of cooperating threads. This design minimizes per-node overhead, enabling SERENADE to emulate scenarios several orders of magnitude larger than existing tools on a single workstation, while reducing startup time to only a few seconds regardless of constellation size.

Beyond scalability, SERENADE introduces a hybrid traffic model that balances realism and efficiency. All nodes can generate synthetic traffic that reproduces network-level load, congestion, and contention, while a configurable subset of nodes connects to External Application Modules: Docker-based environments running unmodified applications with full networking stacks. This integration allows real and synthetic traffic to coexist and interact, enabling realistic end-to-end evaluations under representative network conditions. By combining scale, responsiveness, and traffic realism within a single unified framework, SERENADE hopes to serve as a foundation for next-generation LSN digital twins and to offer researchers a powerful tool for analyzing and designing large-scale satellite networks. Moreover, SERENADE's control plane enables the introduction of orchestration intelligence (both heuristic and ML based) into the emulation, key to its use as a NDT.

Our evaluation of SERENADE demonstrates its ability to scale to massive numbers of nodes (500,000+) while maintaining networking realism and a fast response time, and compares it to existing emulators. Additionally, a Starlink measurement campaign was carried out to validate our realism claims. To highlight the importance of massive-scale, we consider an exemplary, important problem in LSN: maximizing network utilization. We illustrate that the commonly assumed models of user-satellite connectivity lead to severe network under-utilization when applied across multiple thousands of users. Additionally, we illustrate that SERENADE's responsiveness makes it usable as the emulator component of a LSN DT system through several case studies (disaster relief, constellation updates).

Our contributions in this work are as follows:

- We highlight the limitations of existing LSN emulation tools to serve as the platform for LSN Digital Twins.
- We present our solution: SERENADE, which enables flexible, realistic LSN emulation at massive scale (500k+ emulated nodes) while remaining responsive enough to serve as a DT (<2 second startup/reconfiguration time). All of this is achieved on a single machine thanks to its thread-based approach, eliminating the need for distributed emulation.
- We use SERENADE to unveil the network inefficiencies of standard access assumptions in LSNs and illustrate how SERENADE can be used to evaluate multiple network orchestrators (e.g., heuristic, optimized, ML-based.)
- We showcase how SERENADE might be used as a DT in several real-world case studies (disaster relief, constellation updates) highlighting its responsiveness.

We believe SERENADE can serve as an important toolkit for the scientific community and aid researchers in their contributions to the rapidly evolving domain of LSNs. SERENADE has been made available at https://github.com/serenade-project/SERENADE.

2 Background and Motivation

2.1 Motivation

Scale: LSNs are inherently large scale. Modern mega constellations have multiple thousands of satellites and are designed to support global coverage. The constellations are large to support a large number of users—with Starlink (by far the biggest player at the time) recently reporting over 8 million active users [46]. Researchers and operators need a tool which can capture this level of network

[1]Satellite Environment for Realistic Emulation, Network Analysis and Design Exploration

Table 1: Comparison of Satellite Network Simulators/Emulators

Tool	Category	Technology	Focus	# Emulated Nodes	Traffic	Responsiveness[a] (Start/Stop, 1k nodes)
StarPerf [31]	Flow-Level Simulator	Custom	Network	-	-	-
LEOCraft [20]		Custom	Network	-	-	-
Hypatia [29]	Discrete-Event Simulator	ns3	Network	-	Synthetic	-
Plotinus [23]			PHY + Network	-		-
CosmicBeats [43]		Custom	MAC + Network (IoT only)	-	Synthetic	-
StarryNet [30]	Virtual-Network Emulator	Docker	Computation + Network	<1024 (distributed)	Real	∼1500 seconds
RHONE [49]			Power + Thermal + Comp. + Network			∼3600 seconds
OpenSN [35]			Network			∼100 seconds
Celestial [40]		microVMs	Comp. + Network (within bounding box only)	HW resource limited	Real	-
LeoEM [21]		Mininet	Network (single active path only)	4096+	Real	-
xeoverse [28]			PHY + Network			∼300 seconds
SERENADE		**Custom (Go)**	**Network**	**500,000+**	**Background + Real**	**∼1 second**

[a] As tested in § 5.

scale if they want to accurately reflect the underlying system. While qualities like machine thermal profiling or atmospheric effects may factor into device- or link-level evaluation, they largely do not affect the overall behavior of the *network*. There are many aspects of a LSN which have large-scale network effects: handover decisions [51], routing [47], traffic engineering [50], etc.

Network utilization, i.e., the amount of a network's available capacity being used at any time, is an important metric in any network, but especially large-scale ones such as LSNs. The set of network "assignments", i.e., which users connect to which infrastructure or access nodes, is critical to the overall network utilization. In small-scale networks, this choice is trivial—there may be only one infrastructure node available. Even in larger, more complex networks like terrestrial cellular networks, user-base station (BS) association is largely driven by proximity-based metrics like received signal power. In other words, these systems take a "greedy" approach to association. By contrast, in very large networks, and *especially in a complex system with infrastructure mobility like LSNs*, we need intentional, network orchestration to maximize utilization. An emulator without the ability to capture this **scale** with respect to not only the number of *nodes* in the network, but also to the amount of *traffic* they generate, would not allow researchers to faithfully evaluate novel network orchestration controllers.

Responsiveness: In addition to size, another identifying feature of modern LSNs is the degree of their dynamics. Satellites are constantly being added to and decommissioned from the constellation, resulting in changing network topologies. As new users are added to the system, demand patterns change. LSNs are envisioned to be a key component of future 6G networks, so-called non-terrestrial networks (NTN) [27], and will need to be integrated within the existing terrestrial networks, adding further complexity. A network digital twin solution would allow researchers and network operators to continuously experiment and test new network-level solutions, but to do so, the underlying emulation platform must be **responsive** enough to adapt to such changes without incurring significant interruption.

Realistic Traffic: Finally, we note that in order to be an effective network-scale emulator, the tool must deal with real traffic. That is, it must be able to work with packets generated in real time by real applications. As with scale, network effects can only be accurately modeled if there is a high degree of realism.

As we show in the next section (§ 2.2), many of the existing works are *fundamentally* incapable of meeting this large-scale requirement while maintaining usability let alone remaining sufficiently responsive to function as a network digital twin.

2.2 Related Work

Because of the challenges in experimenting on real satellite networks, there have been several recent attempts to create LSN emulators. These emulators can be broadly classified into three categories: flow-level simulators, discrete-event simulators, and virtual-network emulators. Since our work focuses on the emulation of large-scale network behavior, we limit our analysis to those dimensions which enable large-scale LSN digital twin emulators: maximum number of emulated nodes, supported traffic, and responsiveness.

Flow-Level Simulators: In the first category, we have tools like StarPerf [31] and LEOCraft [20], which abstract away packet-level dynamics and instead model traffic as aggregate flows, enabling high-level design exploration, sensitivity studies, and scenario analysis where scalability is more critical than protocol fidelity. Because they do not generate or handle real traffic, they cannot represent packet-level effects, congestion dynamics, or application behavior, making them valuable complementary tools but insufficient as substitutes for real-traffic network emulation.

Discrete-Event Simulators: Platforms in this category, encompassing ns3-based solutions such as Hypatia [29] and Plotinus [23], or custom discrete-event simulators like CosmicBeats [43], advance time by processing scheduled events such as packet transmissions, link updates, or mobility changes, enabling detailed modeling of PHY/MAC behavior and precise control over timing and propagation effects. Because they operate on simulated events rather than real traffic, they are not designed to run in real time or to sustain continuous, indefinite execution. Their event-driven nature also leads to rapidly increasing computational cost as scenario complexity grows, making them impractical for evaluating live applications, end-to-end performance, or long-running operational behaviors characteristic of real LEO systems.

Virtual-Network Emulators: These differ fundamentally from simulators in that they execute real network stacks and real traffic in real time, enabling end-to-end evaluation of applications, protocols, and congestion behavior. Unlike simulators, which can slow

down, speed up, or batch-process events and simply produce results upon completion, emulators must keep pace with wall-clock time, which inherently constrains their scalability. As the size of the emulated scenario grows, the underlying system must sustain a growing amount of concurrent network activity while maintaining real-time execution. Existing LSN emulators typically rely on heavyweight virtualization technologies such as Docker containers, Mininet processes, or microVMs, each introducing substantial per-node overhead.

Docker-based emulators, such as StarryNet [30], RHONE [49], or OpenSN [35], replicate LSNs by deploying one Docker container (or multiple, in the case of RHONE) for each node in the network. StarryNet was the first to adopt this approach for LEO constellations, demonstrating the feasibility of container-based emulation and motivating subsequent systems. RHONE extends this paradigm beyond networking by modeling each node's power, thermal, and computational constraints, which substantially increases its startup time (approximately 2.4 times higher than StarryNet) due to additional container initialization steps. OpenSN, in contrast, introduces several architectural optimizations aimed at improving emulation efficiency and vertical scalability, reducing container overhead while maintaining compatibility with real applications and protocol stacks. Docker-based approaches are attractive because each emulated node can run a fully isolated software environment, enabling unmodified applications and protocol stacks to execute directly. **Limitations:** There are several inherent limitations with this approach: 1) Despite Docker containers being considered "lightweight", the incurred overhead (e.g., per-container filesystem, caching, and processes) limits its scalability. 2) Docker networking is inherently limited in size (max 1024 total nodes) [3, 28], requiring distributed emulation for larger scenarios. 3) Setting up and updating the network topology requires a large volume of per-node and per-link kernel operations, some routed through Docker's centralized control path and the rest applied as per-interface `tc` shaping, significantly impacting responsiveness. 4) The way satellite mobility is handled is to update the virtual network in discrete steps. This i) takes a significant amount of time that scales with the number of nodes in the network and ii) restricts network measurement to either side of the update step; this point is a deal breaker because it prevents us from observing what happens during satellite handovers, a critical point in designing satellite networks.

MicroVM-based emulators, exemplified by Celestial [40], instantiate each network node as a lightweight virtual machine using technologies such as Firecracker [13]. This provides strong isolation and high per-node realism, particularly for studying interactions between networking and on-board computation. **Limitations:** 1) Each microVM requires a non-trivial minimum memory allocation (512 to 1024 MB per node) and supporting OS footprint, causing total resource usage to grow rapidly with constellation size. 2) Although arbitrarily large emulations are theoretically possible, the resulting memory requirements quickly become prohibitive in practice. To mitigate this effect, Celestial makes use of a bounding box, limiting their emulation to a small region. 3) Such baseline costs make microVM-based approaches impractical for the thousands of nodes typical of modern LEO constellations.

Mininet-based emulators rely on lightweight Linux processes and network namespaces to emulate thousands (over 4096) of network nodes within a shared kernel environment [5]. This design allows each virtual node to run real kernel networking code while avoiding the heavier isolation and resource costs of container- or VM-based approaches. LeoEM [21] adopts this model in a narrowly scoped manner, emulating only the active end-to-end path between two ground nodes to study how LEO links affect a single connection. Xeoverse [28] similarly builds on Mininet but targets larger LEO scenarios, improving scalability and responsiveness relative to Docker-based emulators. **Limitations:** 1) As with Docker-based solutions, Mininet still requires substantial per-node and per-link kernel reconfiguration for topology changes, which can limit responsiveness at scale. 2) LeoEM's single-path focus limits its fidelity, preventing analysis of constellation-wide dynamics, multi-user interactions, or realistic congestion patterns. 3) Xeoverse requires all satellite positions and routes to be precomputed before the emulation begins, which i) leads to long preprocessing times (nearly an hour for ~1,500 satellites and 100 ground nodes), ii) prevents from adjusting to changes in the emulation environment during runtime, and iii) limits the emulation to the pre-computed time period.

Table 1 summarizes how the previously discussed categories, enabling technologies, and individual emulators compare across key dimensions. This analysis shows that while several emulators exist for large-scale satellite networks, none exhibit the properties required of a true network digital twin. Existing systems are constrained by architectural limits or by the high per-node virtualization overhead which restricts scalability and resource efficiency. Responsiveness is likewise poor, with deployment and reconfiguration taking over a hundred seconds for thousand-node scenarios (for the most responsive emulator), which remains far too slow for systems meant to evolve in real time. These great delays are a common issue among all existing virtual-network emulators due to the amount of kernel operations required for setting up, updating, and dismantling the network elements. Finally, their limited scalability prevents realistic network dynamics such as congestion or contention from emerging, as only a small fraction of the constellation and total user traffic can be actively represented.

3 SERENADE Emulator Design

To overcome these limitations, we designed SERENADE, a thread-based LEO network emulator purpose-built to meet the requirements of a network digital twin. Its design philosophy centers on maximizing scalability, responsiveness, and realism by eliminating unnecessary abstraction layers and minimizing per-node overhead. In contrast to container- or VM-based emulators that duplicate entire operating environments, SERENADE is written primarily in Go [14] and leverages ultra-lightweight threads (*goroutines*) to represent the majority of network nodes. This approach dramatically reduces memory and CPU consumption per node, enabling the emulation of hundreds of thousands of satellites, gateways, and terminals on modest hardware. The reduction in structural overhead not only improves **scalability** but also yields a secondary benefit: **responsiveness**. Without the latency introduced by container orchestration, virtual-network device management, interprocess communication, or kernel virtualization, SERENADE can initialize,

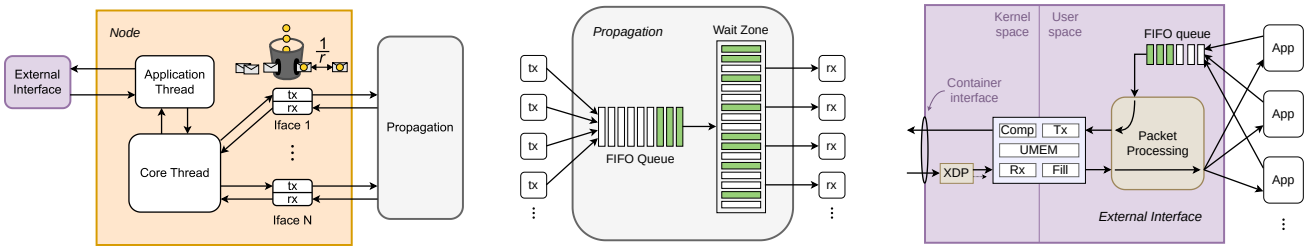


Figure 3: SERENADE Network Module component architectures: a) Emulation node, b) Propagation block, c) External Interface

reconfigure, or teardown large-scale scenarios in seconds rather than minutes. This level of agility allows it to maintain synchronization with real-world systems and supports rapid iteration of experiments—essential properties for a network digital twin.

At the same time, achieving **realism** requires more than scale and speed. Because a fully thread-based design cannot execute arbitrary software in the nodes, SERENADE introduces a hybrid emulation model that integrates two complementary modes of operation. Most nodes are implemented as lightweight thread-based entities capable of producing synthetic background traffic, enabling realistic network load, congestion, and contention patterns to emerge naturally at scale. A smaller subset of nodes (i.e., the External Application Modules, § 3.4) are instantiated as containerized environments capable of running unmodified applications and protocol stacks. This hybrid design combines large-scale load modeling with real-traffic experimentation, supporting both scalability and application-level realism: thousands of virtual nodes can interact dynamically with live application traffic, accurately reflecting end-to-end performance under realistic conditions.

In the following sections, we provide an architectural overview of SERENADE's modules followed by detailed descriptions of each, highlighting the key features which contribute to SERENADE's scale, realism, and responsiveness.

3.1 Architecture Overview

To realize these design objectives, SERENADE adopts a modular architecture that separates core emulation functions from constellation dynamics, and control logic into independent Docker containers (shown in Fig. 1). This separation improves clarity, scalability, and extensibility, allowing each subsystem to evolve independently while maintaining tight integration through shared interfaces. This subsection provides a high-level overview of the four core modules of SERENADE: the Networking, Constellation, Intelligence, and External Application Modules; and outlines how they interact to form a coherent large-scale, hybrid LEO network emulator.

The **Networking Module** (§ 3.2, § 3.3) provides the core packet-forwarding environment in SERENADE (see Fig. 2). It models satellites, user terminals (UTs), and gateways (GWs) as groups of cooperating lightweight threads (Fig. 3a), incorporates a propagation component for realistic one-way delay (Fig. 3b), and exposes interfaces for connecting with external applications (Fig. 3c). This structure enables scalable, packet-level emulation of both synthetic and real traffic while keeping per-node overhead minimal. SERENADE also includes a configurable set of **External Application Modules** (§ 3.4), each implemented as an isolated Docker container corresponding to a UT or GW. These containers can run unmodified applications (e.g., `iperf`, `ping`) and interface directly with the Networking Module, enabling selective integration of real traffic into the emulation.

Several modules contribute to SERENADE's control plane (§ 3.5): The **Constellation Module** supplies the system's time-varying orbital state. Using Skyfield [42], it periodically computes satellite positions and provides them to the **Link-State Engine**, which determines visibility, propagation delay, and channel conditions. A configurable time-dilation factor decouples constellation evolution from networking time, supporting long-horizon mobility studies without long wall-clock execution. The **Intelligence Module** builds on this information to implement flexible control-plane behaviors, as custom models can be deployed in the Intelligence Module to control networking behavior (e.g., UT-Satellite association).

3.2 Large-Scale Emulation via Threads

SERENADE adopts a thread-based emulation architecture designed to extend scalability far beyond existing network emulators while maintaining agility and responsiveness, two properties essential for digital-twin operation. The primary design objective is to minimize per-node overhead by implementing only fundamental networking behaviors (e.g., routing, transmission, and reception) and eliminating dependence on heavyweight virtualization layers such as containers, microVMs, or the Linux kernel networking stack. These elements have been identified as main bottlenecks regarding scalability in existing emulators. By contrast, a lightweight, thread-based design allows SERENADE to scale to upwards of 500k nodes on a single machine. (See § 5 for implementation and experimental details.)

Within this architecture, network entities are abstracted as User Terminals (UTs), Satellites, or Gateways, each instantiated as a collection of cooperating threads (*goroutines*) organized into three functional components: the Core Thread, the Application Thread, and the Transmitter/Receiver Interfaces (Fig. 3a). The Core Thread serves as the Node's primary processing unit, forwarding packets according to decisions issued by the Control Plane (§ 3.5). Packets destined for the Node are directed to the Application Thread, which handles synthetic traffic or interacts with external interfaces (§ 3.4). The Application Thread is also capable of node-level intelligence, such as generating and receiving control messages (§ 3.5). Each Node has multiple Transmitter/Receiver (Tx/Rx) Interfaces. The number of these depends on the number of antennas emulated at

that node—i.e., a UT Node may have one Tx/Rx pair while a satellite Node might have six (one each for uplink, downlink, and four ISL neighbors).

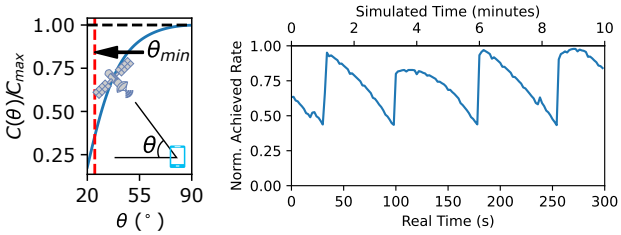
By forgoing traditional virtualization and kernel-based networking, SERENADE achieves improved scalability and responsiveness, but doing so necessitates new solutions for high-throughput packet processing and realistic network behavior. To address the first challenge, SERENADE employs a shared-memory transfer model, where ownership of packet buffers—not their contents—is passed between threads. Each packet, whether generated synthetically or injected externally (§ 3.4), is allocated once and advanced through the emulation pipeline via pointer transfer rather than data copying. The Core Thread performs the required operations (e.g., routing, forwarding, and propagation time calculation) before transferring ownership to the next component. This approach establishes a packet-size-invariant, low-overhead processing framework which contributes to the system's scalability and responsiveness. The complementary challenge of preserving realistic network dynamics is discussed in the following section.

3.3 Networking Realism in User Space

Building on the packet-processing framework introduced above, the next design challenge lies in preserving realistic network dynamics while operating entirely in user space. Because SERENADE executes outside the kernel, it cannot rely on built-in mechanisms for precise timing, queueing, or rate enforcement. Moreover, its multi-threaded execution depends on the scheduling behavior of both the operating system and Go's runtime, introducing nondeterminism that can distort temporal fidelity if left unaddressed. To ensure that large-scale emulations remain both scalable and temporally accurate, SERENADE's design focuses on modeling key timing-related properties—transmission and reception rates, channel capacities, and propagation delays—using mechanisms explicitly built for user-space operation.

3.3.1 Transceiver Rates. Each Transmitter/Receiver Interface in a Node is implemented as two dedicated threads: a transmitter and a receiver, one pair for each antenna. These threads serve a single purpose, enforcing ingress and egress rate limits. Without loss of generality, consider the Transmitter Interface. A naive approach would be to apply inter-packet delays to packets arriving from the Core Thread to control the outbound rate; however, such delays (e.g., 120 μs for 1500-byte-long packets at 100 Mbps) are difficult to enforce reliably in user space. Due to the non-deterministic nature of user-space execution, invoking a sleep operation for a specific time only guarantees the minimum delay, with the actual delay potentially exceeding the target by an unpredictable margin [4]. This unpredictability leads to inconsistent throughput and undermines precise rate control.

Instead, we use a token bucket as a rate limiter. A token bucket is a traffic shaping mechanism commonly used in communication networks to enforce a maximum average transmission rate and maximum burst size. Upon a packet's arrival, a number of tokens, equal or proportional to its length in bytes, must be available in the bucket for the packet to be transmitted. If there are not enough tokens, the packet is delayed until enough tokens accumulate. Token buckets should be refilled at a uniform rate for them to behave as



(a) (b)

Figure 4: a) Normalized Channel Factor, b) Impact of Channel Quality ($k = 0.1$, $\theta_0 = 15°$, $\alpha = 0.3$)

expected; however, due to the maximum frequency of the Linux clock interrupt being 1 kHz, the burst size must be increased in order to achieve the desired maximum average rate, increasing the burstiness of the resulting traffic. Linux's traffic control utility (`tc`), used by Mininet [5] and StarryNet [30], also implements a token bucket filter and is thus subject to the same limitations; therefore, SERENADE's traffic behavior is the same as the one caused by `tc`.

3.3.2 Link Channel Capacities. LSNs are fundamentally wireless networks, where link capacities vary stochastically with environmental and user factors (e.g., distance, weather, multipath). While SERENADE is focused on large-scale network emulation, it provides researchers with a mechanism to approximate large-scale link-level variation across ground-to-satellite links by adjusting the "exchange rate" between tokens and packets in each Tx/Rx interface's token bucket. Intuitively, tokens represent channel uses/attempts required for transmission; increasing the tokens or number of channel uses/attempts per packet is equivalent to reducing the channel capacity.

By default, SERENADE models channel degradation as a function of the elevation angle θ, which reflects both distance-related free-space loss and atmospheric attenuation [8]. The resulting capacity is given by:

$$C(\theta) = C_{max}\left(\frac{2}{1+e^{-k(\theta-\theta_0)}} - 1\right)\sin(\theta)^{\alpha}, \tag{1}$$

where k, θ_0, and α are user-configurable parameters. Fig. 4 shows an example of $C(\theta)$ and the corresponding normalized rate for a single user maintaining a link to the nearest visible satellite. This function was selected because a similar relationship between ergodic capacity and elevation angle for LEO satellites has been demonstrated in [36]. While the exact parameters may differ, this approximation provides a practical mechanism to capture large-scale PHY effects — absent in tools like StarryNet and OpenSN; and can be configured for cases where constrained channel conditions are of interest. Overall, it enables researchers to abstract complex propagation phenomena that would otherwise make large-scale emulation computationally infeasible.

Currently, SERENADE employs a simple first-in-first-out (FIFO) medium access scheme. When multiple UTs transmit to the same satellite, their packets queue at the satellite's receiver interface, which enforces the target receive rate. As long as the aggregate demand remains below the satellite's capacity (i.e., uplink token bucket rate), no losses occur; once exceeded, the buffer fills and packets are dropped proportionally to the offered load, maintaining

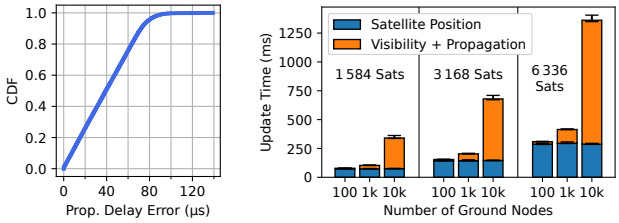


Figure 5: a) Propagation Delay Error, b) Constellation Update Times

the original demand ratios. Thanks to node-level intelligence (§ 3.5), alternative access policies can be integrated in the future.

3.3.3 Propagation Delays. Accurately emulating propagation delay is a fundamental challenge in satellite network emulation. Since emulating such delay requires temporarily storing packets for the duration of their expected propagation time, this mechanism can become a critical bottleneck due to the memory demands it imposes; specifically, the total memory required is given by the aggregate bandwidth-delay product (BDP) across all active links, defined as:

$$BDP_{total} = \sum_{\forall i \in A} R_i d_i \,, \tag{2}$$

where A denotes the set of active links and R_i and d_i are the transmitted rate and propagation delay for the i-th link, respectively. Given this relationship, pre-allocating memory buffers for all possible links in a large-scale emulation becomes impractical, particularly when many of these links may remain underutilized due to the non-uniform geographical distribution of users and gateways.

To emulate propagation delays consistently while preserving packet order and supporting high-rate nodes, SERENADE employs the **Propagation Module** (Fig. 3b). It uses a FIFO queue that aggregates packets from all transmitters and feeds them into a *Wait Zone*, a dedicated thread that iteratively scans an array of packet pointers. Each iteration fills empty slots with new packets and releases those whose transmission timestamps, set by the sender's Core Thread (§ 3.2), have expired. The iteration rate of the Wait Zone defines the delay resolution, enabling precise yet scalable timing control while maintaining efficient resource usage.

Fig. 5a shows the CDF of excess delay for 100,000 packets traversing the Propagation Module, generated by 5,000 users transmitting at 1 Mbps. The observed delay is nearly uniform with a 95th percentile of 79 μs, indicating an average iteration time of about 80 μs for the Propagation Module. Because the delay error is independent of the target propagation time, the relative excess delay exhibits a similar distribution (95th percentile = 4.19%), demonstrating that SERENADE maintains precise delay emulation even under heavy load. A key tunable parameter is the **Wait Zone size**, which determines the effective bandwidth–delay product (BDP). If the Wait Zone becomes full and is subject to highly bursty traffic, packet reordering may occur. Smaller zones limit the achievable BDP and increase the likelihood of reordering occurring, whereas larger ones lengthen each iteration, lowering delay resolution and risking FIFO overflow when processing falls behind. To maintain performance, SERENADE splits propagation into uplink, downlink, and inter-satellite segments so that each operates within capacity using smaller, faster Wait Zones. Ultimately, the optimal size is platform-dependent; see § 5 for details on our implementation.

In summary, SERENADE's transition to a fully user-space, thread-based architecture unlocks massive scalability but necessitates careful treatment of time-sensitive network behavior. To preserve realism under these conditions, its design combines token-bucket mechanisms for transceiver rate control, an angle-based formulation for channel capacity, and a multi-queue delay model with tunable wait zones. Together, these components reproduce the essential dynamics of LEO communication links while maintaining the responsiveness required for real-time digital-twin operation.

3.4 Hybrid Traffic Model

A defining feature of SERENADE is its hybrid traffic model, in which *synthetic* and *real* traffic coexist and interact within the same emulated network. Synthetic traffic, generated by internal nodes, provides large-scale background load representative of thousands of users, while real traffic, originating from External Application Modules, enables end-to-end evaluation of actual applications and protocols. This combination balances scalability with realism, allowing researchers to observe system-level behavior under realistic congestion and routing dynamics.

Synthetic traffic is produced by the Application Thread of each user node at variable rates defined in the configuration file and enforced via token buckets. Packets are then passed to the Core Thread and propagated through the network as described in § 3.2.

To incorporate *real traffic*, SERENADE connects External Application Modules, each mapped to a specific UT, through dedicated external interfaces (Fig. 3c). External Ethernet frames enter and exit the emulation via AF_XDP sockets [1], which bypass the kernel stack to achieve high-speed, low-latency communication with the user-space network.

While the underlying mechanics of packet exchange are handled transparently, the key implication is that real and synthetic packets traverse the same links, queues, and buffers, influencing each other's latency and throughput. Unlike existing emulators, which must artificially inflate delays to approximate the effects of congestion, SERENADE reproduces these effects naturally: synthetic background traffic competes for shared resources, creating genuine congestion and additional buffer-induced latency. This enables realistic performance evaluation of real-world applications (e.g., video streaming, file transfer) under conditions that accurately reflect large-scale LEO network behavior.

3.5 Satellite Modeling and Control Plane Architecture

To support the packet processing architecture outlined in § 3.2, SERENADE incorporates a modular and extensible control plane designed to orchestrate the dynamic behavior of the emulated satellite network. As shown in Fig. 6, it consists of three components: the **Constellation Module** and **Intelligence Module**, both external containers, and the **Link-State Engine**, which runs internally within the Networking Module. This separation isolates orbit propagation, node-role assignment, and link-state evaluation, improving scalability and easing customization of the emulation environment.

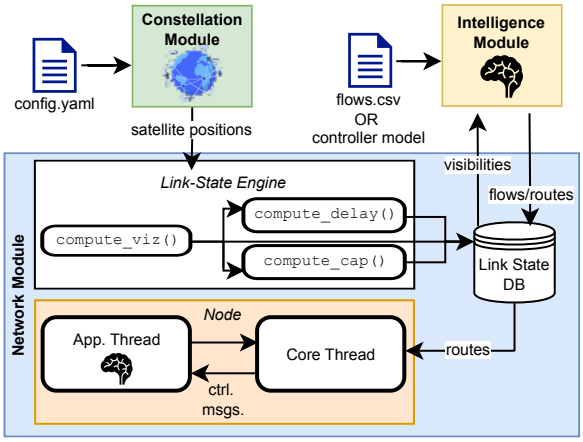


Figure 6: SERENADE Control Plane Architecture

Constellation Module: The Constellation Module is based on the Skyfield Python astronomy package [42]. Throughout the emulation, the module updates satellite positions on demand by computing new ITRF (International Terrestrial Reference Frame) coordinates at "dilated" timestamps. These modified timestamps are calculated as:

$$T[n] = T[n-1] + \Delta t \cdot \eta\,, \tag{3}$$

where $T[n-1]$ is the previous timestamp, Δt is the real time elapsed since the last update, and η is a speedup factor. This enables long-term satellite dynamics, such as visibility shifts and inter-satellite link (ISL) routing changes, to be simulated on a compressed timescale while maintaining real-time packet-level interactions.

A core advantage of SERENADE is this *decoupling of constellation motion from real-time traffic emulation*. Whereas traditional emulators advance mobility and traffic synchronously, limiting scale and scenario duration, SERENADE's time-dilation approach reproduces multi-hour/day orbital behavior within minutes without added computational cost. This allows researchers to study long-term connectivity patterns alongside fine-grained packet-level events.

Once the coordinates for all nodes have been established, the Link-State Engine determines which satellite-to-ground links (UTs, GWs) are currently viable, based on the minimal elevation angle. For each visible satellite-to-ground link and inter-satellite link, the engine then computes the corresponding propagation delay and link channel capacity. These values are written to a shared thread-safe data structure accessible by all nodes, ensuring a consistent and synchronized update of the network state across the emulation environment.

Fig. 5b shows the update time for the Constellation Module and the Link-State Engine, for three different constellation sizes—1584 (72 orbits x 22 satellites per orbit), 3256 (72 x 44), and 6336 (72 x 88)—and varying number of ground nodes. Note that the Constellation Module update time itself does not vary with the number of Ground Nodes, only the size of the constellation and in a linear manner. The Visibility+Propagation calculation by the Link-State Engine is proportional to the number of ground nodes times the number of satellites. The total time between these updates represents the minimum step size for SERENADE's emulation. For the 1584 constellation (Starlink phase 1 shell), the total update time for 10,000 ground nodes is roughly 340 ms, meaning SERENADE could update

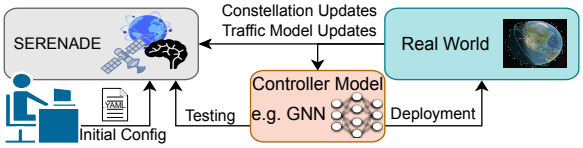


Figure 7: SERENADE as a Network DT

the positions of satellites, together with the wireless channel parameters (latency and rate) and any handovers, roughly three times per second—this is more than sufficient to capture fine-grained satellite behavior. SERENADE's decoupling of the constellation and networking modules allows speeding up or slowing down the satellite dynamics with respect to real time, enabling larger emulations to run at the desired constellation-level granularity without requiring any precomputation.

Intelligence Module: The Intelligence Module is responsible for the generation of sophisticated user-satellite-gateway assignments based on the calculated visibility, delay, and capacity values provided by the Network Module. While SERENADE defaults to a nearest-neighbor policy for routing (i.e., selecting the closest satellite and gateway), the modular design of the Intelligence Module allows for integrating custom ground-to-space assignments and ISL routing logic. This flexibility enables the exploration of customized decision-making policies, allowing for various performance goals, operational constraints, and deployment scenarios, as we will later see in our network utilization use case § 4.

Node-Level Control: In addition to the Intelligence Module, control plane functionality can originate from within the Application Threads of the emulated nodes. For example, consider using SERENADE for emulating 3GPP Non-Terrestrial Networks (NTN) [27]. One of the critical challenges for NTNs is how to handle the large quantity of handovers that happen between UEs on the ground and satellites. Different proposed NTN architectures define distinct gNB functional splits, which in turn dictate the type and amount of control messages required during handover. In SERENADE, the Application Thread can generate these control messages directly, allowing satellites to exchange information and implement a variety of handover algorithms within the emulation.

Design Summary: The primary motivation for SERENADE is to emulate LSNs at massive scales demanded by practical deployments but not supported by existing solutions. The key design components of SERENADE are to enable massive LSN emulation while maintaining realistic networking behavior. Additionally, SERENADE offers incredible flexibility in designing and conducting experiments to facilitate LSN networking research at scale as we will see in the following section.

4 Using SERENADE: From Emulation to Digital Twin

In the previous section, we have laid out SERENADE's core design principles as well as detailed its modular architecture. We now describe how researchers and operators can use SERENADE, both as a realistic, large-scale emulator and as a real-time network digital twin for LSNs. Fig. 7 illustrates how users interact with SERENADE in both cases. The first step in using SERENADE is in providing initial configuration files. These YAML files collectively describe the

satellite constellation parameters, UT and GW locations, communication constraints, and constellation speed-up factor. Users also configure the demand model which, as described in § 3.4, has two components. First, users define the background traffic model—i.e., how much traffic is generated synthetically by nodes. Second, users configure the **External Application Module** which creates client and server containers and attaches them to the appropriate nodes. Using these containers, researchers can inject real-time, dynamic traffic, observe the network behavior, and generate packet traces.

Large Scale Emulation: An important component of SERENADE's control plane is the **Intelligence Module**, which defines how users, satellites, and GWs connect to one another. Recall that certain network-level behavior, like network utilization, can only be properly observed at scale, as discussed in § 2.1. Through its thread-based design, SERENADE gives us a platform to evaluate different network behavior at the appropriate (*realistic*) scale.

The Intelligence Module allows for flexible implementations, and as such, we have included several different controller baselines. This includes two heuristic controllers: the "nearest-neighbor" as described in § 2.1 and "hold" which maintains association as long as visibility constraints are met. We have also formulated the network utilization problem as a mixed-integer linear program, and used a commercial-grade solver [25] to maximize network throughput through load balancing. While this is the optimal controller, solving the full optimization cannot be done within the timescales of satellite dynamics. In this scenario, the assignments are fed to the Intelligence Module through a pre-computed CSV file. To bridge the gap between scalable but sub-optimal heuristics and an optimal-but-slow controller, we have also designed a graph neural network to make network-wide orchestration decisions. ML-based inference is faster than traditional optimization, and the GNN is a natural fit for satellite problems due to the structure of satellite constellations, and has been used in similar applications such as handover [33] and routing [52]. In § 5, we use SERENADE to evaluate these different network orchestrators on the task of network utilization.

Network Digital Twin: While SERENADE can be used by researchers as a stand-alone platform for generating data and testing, ultimately network operators need a tool that not only models an LSN, but also reflects the dynamic reality of the true system. This second mode of operation for SERENADE is the network digital twin. As illustrated in Fig. 7, SERENADE can receive information from the real world and update *on-the-fly* to reflect these changes. In the following section, we will demonstrate multiple instances of real-time dynamic updates to the underlying emulation.

5 Evaluation

In this section we evaluate SERENADE across a range of performance metrics and experimental case studies. We proceed by comparing SERENADE to several state-of-the art emulators on key scalability and responsiveness metrics across diverse emulation scenarios (§ 5.1). Out of the emulators listed in § 2.2, we have not included RHONE [49] and xeoverse [28] due to them not being publicly-available, Celestial [40] for lack of scalability and use of bounding boxes to limit emulation size, and LeoEM [21] for limiting its emulation strictly to satellites in the current active path. Thus reducing our comparison to StarryNet [30], OpenSN [35], and Mininet [5]

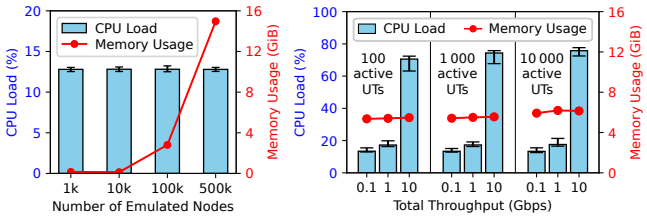


Figure 8: Scalability results for a) number of nodes (memory constrained), and b) total traffic (CPU constrained).

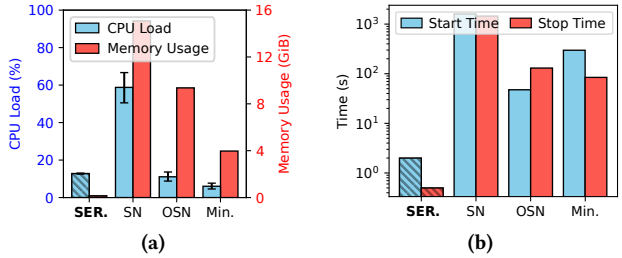


Figure 9: a) Hardware and b) start/stop time (log scale) comparison for SERENADE (SER.), StarryNet (SN), OpenSN (OSN), and Mininet (Min.) in 1,000-node scenarios.

as a stand-in for the closed-source xeoverse. We then benchmark SERENADE's Network Module ability to emulate network traffic realistically, including a comparison to real world Starlink measurements (§ 5.2). Finally, we return to the motivating use case of network orchestration (§ 5.3) and operation as a LSN-DT (§ 5.4).

Implementation: All the following evaluations are executed on a single desktop Linux workstation with a 12-core Intel i9 CPU at 3.5 GHz with 32 GB of RAM. Based on the discussion in § 3.3.3, we set the size of the Wait Zone array to 2^{14}. Unless specified, we emulate a 1584-satellite constellation based on the Starlink Phase 1 deployment.

5.1 Scalability

We demonstrate SERENADE's capability to emulate extremely large numbers of nodes within a large-scale network (LSN), highlighting both its scalability and the limits imposed by physical and computational constraints. As illustrated in Fig. 5b, there exists an inherent limit to the number of nodes in the emulation for a desired physics-update period (i.e., the granularity at which SERENADE can emulate the movement of satellites). Nonetheless, SERENADE's unique ability to dynamically accelerate or decelerate satellite dynamics with respect to the network emulation softens this limitation. While these design mechanisms alleviate some of the timing constraints, hardware resources, in contrast, impose hard and unavoidable boundaries on the achievable emulation scale.

Scaling the Nodes: To characterize these effects, Fig. 8a presents a series of experiments where we progressively increase the number of non-traffic-generating nodes, including satellites, gateways, and user terminals. The results clearly show that CPU load remains nearly constant as the node count grows, demonstrating that the computational overhead of maintaining additional idle nodes is minimal. Meanwhile, the memory footprint of SERENADE increases

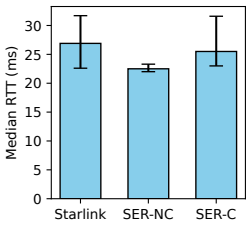


Figure 10: Latency comparison between Starlink and SERENADE in non-congested (SER-NC) and congested (SER-C) scenarios.

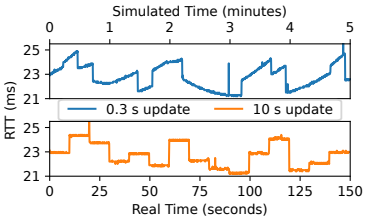


Figure 11: Latency comparison in SERENADE for different constellation update times.

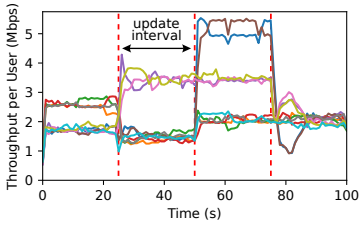


Figure 12: User throughput through multiple switch events in SERENADE.

predictably with scale, reaching approximately 15 GiB for 500,000 nodes, which represents a practical upper bound given our available hardware resources.

Scaling the Traffic: Fig. 8b explores the second dimension constraining SERENADE's scalability: the total amount of emulated traffic. To examine this, we deploy three scenarios with 100, 1,000, and 10,000 active user terminals (UTs), respectively. Each scenario is configured such that the aggregate throughput reaches 0.1, 1, and 10 Gbps, corresponding, for example, to 1,000 UTs each transmitting at 10 Mbps for the 10 Gbps case. Across all configurations, the hardware footprint remains nearly identical, with memory utilization showing minimal variation regardless of total traffic volume. However, CPU utilization scales with the offered load, eventually becoming the dominant limiting factor once the total throughput surpasses approximately 10 Gbps (~75 % CPU load). Thus, while SERENADE can faithfully represent the full size and geometry of large constellations, its scalability in practice also depends on the aggregate workload carried by the constellation rather than just on the total number of emulated nodes.

Takeaway: SERENADE *supports realistic LSN experiments by addressing both dimensions of scalability: (i) emulating large constellations and ground deployments with modest per-node overhead, and (ii) generating realistic, network-wide traffic patterns.*

Remaining Efficient: We now compare, for a fixed emulation size, the resource usage and startup time of SERENADE and the existing baselines. All emulators are configured to run a 1000-node topology; well within the documented limits of StarryNet, OpenSN, and Mininet. Figure 9a reports the CPU load and memory usage required by each emulator, while Figure 9b shows the corresponding start and stop times.

StarryNet exhibits the highest CPU load and the slowest initialization and teardown times, with delays reaching up to 20 minutes even at this moderate scale. OpenSN, which also adopts a container-based design, significantly reduces both CPU usage and start/stop latency, improving performance by roughly an order of magnitude for the latter, yet remains restricted by its 1,024-node cap per machine, requiring distributed deployments for larger constellations, increasing deployment complexity. Acting as a baseline for Mininet-based emulators, we can observe our simple Mininet example uses the least amount of hardware resources among pre-existing alternatives, with its startup and shutdown delays falling in the multi-minute range[2]. By comparison, SERENADE sustains low CPU load, minimal memory usage, and near-instantaneous start/stop times under these same conditions.

Takeaway: SERENADE*'s lightweight, thread-based architecture eliminates the resource and orchestration overhead of containerized and network namespace-based emulators, enabling large-scale LEO constellation experimentation on a single machine with far superior efficiency and practicality.*

5.2 Networking Realism

Matching Real-World Latency: To assess the latency fidelity of SERENADE, we conducted a measurement campaign using a REV-4 Standard Starlink user terminal located in the continental United States. We collected RTT measurements by issuing ICMP echo requests directly toward the serving ground station/PoP, and we compare these observations against those produced by SERENADE under equivalent operational conditions.

Fig. 10 shows the measured RTT from the Starlink measurements as well as two configurations of SERENADE: without (SER-NC) and with congestion (SER-C). A common limitation of existing LSN emulators is their inability to reproduce the inherent delay dispersion observed in real systems. Prior work (e.g., [30]) has shown that simplified or fully virtualized networking stacks tend to produce unrealistically low variance in RTT. In practice, however, LEO network latency fluctuates due to a combination of satellite handovers, constantly changing propagation distances, and queuing dynamics driven by traffic demand.

SERENADE addresses this challenge through two complementary mechanisms. First, the Intelligence Module supports more advanced satellite selection policies, beyond the naive "closest satellite" heuristic, capturing realistic variations in both link distance and handover timing. Second, SERENADE's ability to emulate many thousands of nodes enables controlled generation of synthetic background traffic, allowing us to accurately model congestion effects and the resulting non-uniform queuing delays.

As shown in Fig. 10, SERENADE closely matches Starlink's observed RTT distribution in the congested scenario (SER-C), including the increased dispersion under load. This demonstrates that SERENADE can faithfully capture not only baseline LEO propagation delays but also the dynamic latency characteristics arising from realistic satellite mobility and traffic-induced contention.

Emulator Latency: Figure 11 illustrates the importance of supporting sub-second constellation updates in an LSN emulator. With a 0.3-second update interval, SERENADE clearly captures the characteristic "U-shaped" RTT pattern that arises as satellites approach

[2]These results don't take into account the additional delay due to the topology and satellite mobility pre-computation performed by xeoverse.

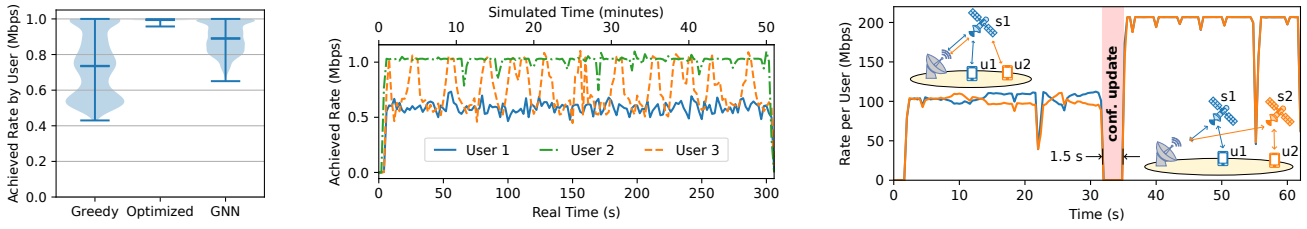


Figure 13: a) Average Throughput Distributions b) Throughput Traces for Greedy Strategy c) Environment Update (1.5 seconds)

and recede from a UT over a 15-second assignment window (matching Starlink's behavior). This fine-grained update cadence preserves the smooth, continuous delay evolution dictated by orbital motion. In contrast, using a 10-second update interval effectively quantizes satellite movement: the RTT curve becomes piecewise-constant, masking the natural delay dynamics and significantly reducing fidelity for mobility-sensitive evaluations.

Crucially, SERENADE can sustain these sub-second updates indefinitely because satellite positions are computed on demand rather than precomputed. By contrast, other LSN emulators face fundamental limitations: those that rely on precomputed orbital trajectories (e.g., xeoverse) cannot update the constellation faster than the granularity of their stored traces, while those with heavier update pipelines or less efficient control-plane architectures cannot process state changes quickly enough to support sub-second ground-to-satellite reassignment [35]. Regardless of the specific cause, these constraints force competing emulators to operate at coarse update intervals, reducing the temporal fidelity of the delay dynamics they can reproduce. SERENADE's lightweight, real-time update mechanism therefore enables a level of fine-grained, mobility-driven realism that alternative platforms cannot match.

Satellite Switches: In this experiment, we wish to observe how SERENADE handles satellite mobility from a networking perspective. We consider 10 UTs located in relatively close proximity to one another. All UTs are connected to External Application modules which run an `iperf` client to send TCP traffic to a server located at a nearby GW. The satellites are configured to have a capacity of 10 Mbps for the purpose of illustration. The measured throughput for each UT is shown in Fig. 12. In the beginning of the emulation, the UTs are distributed between two satellites: six on one and four in the other. The UTs connected to the first satellite can achieve a throughput of roughly $\frac{10}{6} = 1.66$ Mbps; likewise the UTs connected to the second satellite achieve around 2.5 Mbps. After 25 s, the assignment[3] changes such that the distribution is now 3/7 for the first/second satellite. The UT throughputs adapt accordingly. After another 25 s, a new/third satellite is utilized such that the distribution is 3/2/5. Finally, the first satellite leaves, and the remaining two satellites share load equally.

This experiment highlights two aspects of SERENADE. First is its ability to capture many traffic traces simultaneously. While SERENADE's hybrid traffic model relies on synthetic traffic to scale to massive numbers of nodes, we can still collect real traces from a non-trivial number of users as needed. Second, it demonstrates how SERENADE faithfully handles dynamics in LSNs. As satellites move, and assignments change, the per-user throughput adapts as congestion reduces the capacity available per node.

Takeaway: SERENADE *is a high-fidelity LSN emulator, matching real-world and theoretical performance, despite the challenges of user-space-based networking.*

5.3 Case Study: Network Utilization at Scale

We now return to the important problem of LSN network utilization introduced in § 2.1. In this experiment, we consider 5000 UTs randomly distributed throughout the United States, 54 GWs placed at crowd-sourced Starlink GW locations [9], and the 1584-Starlink phase 1 constellation. Each UT attempts to transmit at 1 Mbps, each GW has eight antennas, and each satellite has a 100 Mbps capacity. We test both the Greedy (i.e., "nearest neighbor") satellite selection strategy as well as the Optimized and GNN controllers. The optimized and GNN-based assignments are uploaded to SERENADE's Intelligence Module; however, in the future, the controller will run the GNN model directly. Recall that SERENADE uses a "time-dilation factor" to speed up satellite mobility while retaining real-time networking measurements. This allows us to capture the behavior of long-term satellite mobility (e.g., multiple handovers, many different satellites, etc.) in a short amount of time. We run the emulation for 300s with a time dilation factor of 10, for a total simulated time of 50 minutes.

Fig. 13a shows the distributions of the average throughput for each UT over the entire emulation for each strategy. As expected, when using the Optimized controller, the total network load is balanced across the satellite constellation. The result is that there is little congestion, and all 5000 users are able to achieve nearly the full rate of 1 Mbps nearly all the time. When using the GNN controller, we lose a bit of efficiency, but the majority of UTs are above 80% of the desired throughput. And while the Optimized Controller takes on average 4-5 seconds to compute a solution, the GNN inference is effectively instantaneous by comparison, making it a promising candidate for real-time LSN network orchestration. By contrast, when UTs deploy the Greedy strategy, the UT experience is much more varied. While some users are able to achieve their full rate, there are what appear to be three modes in the distribution, with the two other modes achieving only around 0.75 and 0.55 Mbps.

To inspect these three modes further, we also connect External Application Modules to 10 random UTs in the experiment and run `iperf` transmitting 1 Mbps of UDP traffic to measure their achievable rate. The traces from three of these UTs using the Greedy connection strategy are shown in Fig. 13b. User 3 is able to achieve the full 1 Mbps rate, indicating that over the course of the experiment, its chosen satellites are never overloaded; meanwhile for

[3]Note that we use a custom assignment to illustrate the above point.

User 1, the opposite is true. Its satellites are always subject to heavy load, thus it never achieves more than 0.75 Mbps. On the other hand, User 2's rate oscillates back and forth between these two modes.

This case study seeks to illustrate how researchers may use SERENADE in the future to diagnose, prepare, and test network-scale solutions for LSNs. Emulators which are incapable of modeling massive numbers of nodes would be unable to fully capture the network dynamics of a full scale LSN deployment. Network utilization is one example of a vitally important metric that is only really observable on an emulator the size of SERENADE. Additionally, using SERENADE as a DT means that as the real world changes (either through constellation updates or traffic pattern shifts), researchers can easily and rapidly test the proposed GNN-based controller in a controlled environment.

Takeaway: SERENADE *supports numerous, customizable network orchestration controllers through its Intelligence Module.*

5.4 Case Study: Responsiveness for Digital Twin

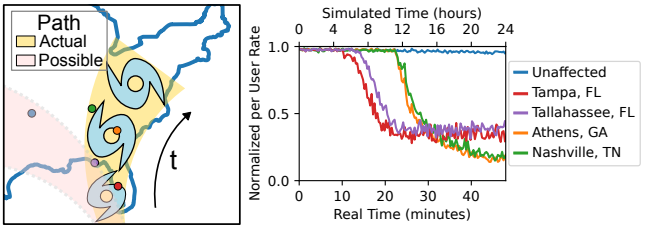
As mentioned, digital twin systems rely on a bi-directional relationship between the real and emulated systems. This requires the emulation platform to be responsive to information gained from the real world. To highlight SERENADE's responsiveness, we consider the following case studies.

5.4.1 Constellation Updates. LSNs are rapidly changing, with new satellites being deployed, satellites going online, coming offline, etc. on a nearly daily basis. Existing LSN emulation tools are designed to run pre-defined experiments for a single satellite topology in mind. If an update to the topology is needed, the emulation must be shut down entirely and then restarted which may require significant turn-around time (this was highlighted in Fig. 9b).

Owing to SERENADE's lightweight and modular design, researchers can respond to changing constellation topologies on-the-fly. Consider the following scenario, depicted in Fig. 13c. Researchers are using SERENADE to monitor the throughput of two users (u1, u2) located in close proximity to one another and connected to the same satellite (s1). They use SERENADE to measure the throughput achievable by each user by running `iperf` client in the External Application Modules. The operators of the constellation then deploy a new satellite (s2) in a nearby orbit to s1. The researchers update SERENADE's config with the new topology, and restart only the Networking, Constellation, and Intelligence Modules of SERENADE, leaving the External Applications running. The total time to apply the update is approximately 1.5 s. After the update, u2 connects to s2 rather than s1. The result is that both UTs are able to utilize the full satellite capacity of 200 Mbps. It is important to note that the TCP congestion control algorithm used for this test is BBR. BBR continually probes the available bandwidth during transmission and periodically drops its throughput to 2% of the achieved rate to accurately measure the roundtrip delay [2]. This drop in throughput is what can be observed in Fig. 13c at $t \approx 20$ s and $t \approx 55$ s.

Takeaway: LSN configuration updates (e.g., demand model and constellation config) can be seamlessly modified without disrupting ongoing applications

5.4.2 Demand Dynamics: Disaster Relief. As LEO satellite networks become more and more prevalent, more and more users will begin



(a) (b)

Figure 14: a) Real/Possible Path of Hurricane, b) User Throughput

to access the network. The locations from which these users come online and the nature of their traffic are unknowable *a priori*. The benefit of SERENADE as a digital twin is that we can add or modify the behavior of user nodes *on-the-fly*. This is in stark contrast to all of the existing systems which can only run pre-defined experiments.

To illustrate this, we consider the very real case study of hurricane disaster relief. In the fall of 2024, Hurricane Helene made landfall in the southeastern United States, knocking out a significant portion of the terrestrial network infrastructure in the area. During this period of disaster, many people turned to satellite networks (e.g., Apple's direct to cell [41] and Starlink [44]) as a means of connecting to emergency services and contacting loved ones. These services represent an important resource to those in distress, and it's important that they are able to respond quickly.

We conduct an experiment inspired by this scenario: Upon learning of an oncoming storm, we prepare SERENADE by allocating 5000 UTs throughout the region the LSN provides service to (i.e., the US mainland), which includes two potential hurricane trajectories, shown in Fig. 14a. Initially each UT generates a demand of 1 Mbps. Of these, we set up a number of them with external interfaces to monitor their throughput via `iperf` (in this example, four of them within the eventual trajectory). When the hurricane hits, as we learn its exact trajectory, we model the previously mentioned expected increase in users by increasing the demand of the synthetic users within the storm's path by a factor of 10. With SERENADE, it is possible to push this new demand configuration to the targeted users, *on-the-fly*, without needing to stop, reconfigure, or restart the emulation, unlike existing tools. Fig. 14b shows the achieved rates for the five test users. The "unaffected" user is able to maintain its full rate throughout the storm, while the other four see their rate drop predictably as the damage sweeps over them and more users begin to rely on the LSN for connectivity. Note that this is a targeted experiment to highlight the adaptability of SERENADE's traffic model. More sophisticated experiments are also possible, for example: studying the potential impact this increase in demand may cause on the rest of the network as different load-balancing strategies are attempted.

6 Discussion

Distributed Scalability: SERENADE does not currently support distributed emulation (i.e., dividing the emulation workload between several machines). We argue that such horizontal scalability in other emulators (e.g., Docker or microVM-based) is not an inherent

advantage, but rather a necessity imposed by their architectural limitations, specifically: a) strict caps on the number of nodes per host, b) excessive per-node resource overhead, or c) both. In contrast, SERENADE's lightweight thread-based architecture enables single-machine scalability of up to 500,000 nodes while maintaining realistic network behavior, a scale that systems like StarryNet or OpenSN can only achieve by distributing the load over hundreds of machines. While SERENADE empowers even students/researchers with limited compute to run realistic at-scale LSN emulation, nonetheless, if future research demands distributed operation, SERENADE could be extended (through multiple instances with appropriate interfaces) to incorporate this feature.

Lower-Level Realism: A key design decision in SERENADE is to focus on large-scale network behavior rather than low-level PHY/MAC interactions. This choice reflects a deliberate trade-off between low-level fidelity and scalability. Emulating detailed physical-layer phenomena can provide valuable insights, but it also imposes significant computational overhead, making it impractical for the scale and responsiveness that a network digital twin demands. To maintain efficiency, SERENADE models link capacities through a simplified, parameterized channel abstraction without incurring the cost of full signal-level emulation. This level of abstraction is sufficient to reproduce the effects most relevant to application and user performance, which evolve at coarser time scales than the instantaneous dynamics of individual channels. More detailed PHY/MAC modeling could be incorporated in future work, but doing so would shift SERENADE's scope away from its current emphasis on massive-scale, responsive emulation toward finer-grained physical simulation.

7 Conclusion

In this work, we presented SERENADE, a scalable and highly responsive emulator for large-scale LEO satellite networks. By leveraging lightweight thread-based emulation and a hybrid traffic framework, SERENADE can realistically model dynamic constellations exceeding 100,000 nodes, orders of magnitude beyond the capabilities of existing tools. It does this while remaining ultra-responsive, enabling on-the-fly changes to parameters such as constellation dynamics or user traffic patterns. Together, these properties—scale, responsiveness, and realism—position SERENADE not only as a powerful platform for analyzing the behavior of large satellite constellations, but also as a foundation for network digital twins capable of mirroring live system dynamics in real time. By combining accurate large-scale emulation with rapid adaptability to evolving conditions, SERENADE enables the study of system-level effects such as congestion, connectivity evolution, and network utilization under realistic and time-varying network conditions. Ongoing work focuses on extending SERENADE's framework to support 3GPP non-terrestrial network (NTN) contexts and protocols.

Acknowledgments

This work was supported in part by NSF (CNS 2208761, CNS 2414197).

References

[1] [n. d.]. AF_XDP — The Linux Kernel documentation. https://docs.kernel.org/networking/af{_}xdp.html
[2] [n. d.]. BBR Congestion Control. https://www.ietf.org/archive/id/draft-cardwell-iccrg-bbr-congestion-control-01.html
[3] [n. d.]. Bridge network driver. https://docs.docker.com/engine/network/drivers/bridge/
[4] [n. d.]. Golang time library documentation. https://pkg.go.dev/time/
[5] [n. d.]. Mininet: An Instant Virtual Network on Your Laptop (or Other PC) - Mininet. https://mininet.org/
[6] [n. d.]. OneWeb Network. http://oneweb.net/about-us
[7] 2020. Telesat Lightspeed LEO Network | Telesat. https://www.telesat.com/leo-satellites/
[8] 2023. Propagation data and prediction methods required for the design. https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.618-14-202308-I!!PDF-E.pdf
[9] 2023. Starlink Ground Station Locations (2024). https://starlinkinsider.com/starlink-gateway-locations/
[10] 2024. Satellite IoT solutions for smart agriculture in Latin America. https://www.satellitetoday.com/content-collection/satellite-iot-solutions-for-smart-agriculture-in-latin-america/
[11] 2024. Starlink | Specifications. https://www.starlink.com/specifications?spec=4
[12] 2024. A Whole New Orbit: Why Shipping Should Embrace the Connectivity Boom. https://maritime-executive.com/editorials/a-whole-new-orbit-why-shipping-should-embrace-the-connectivity-boom
[13] n.d.. Firecracker: Secure and fast microVMs for serverless computing. https://firecracker-microvm.github.io/
[14] n.d.. The Go Programming Language. https://go.dev/
[15] [n. d.]. Iridium Network. https://www.iridium.com/network/
[16] [n. d.]. Viasat to Provide In-Flight Connectivity to Additional Delta Aircraft, Including Widebodies. https://www.viasat.com/news/latest-news/aviation/2023/viasat-to-provide-in-flight-connectivity-to-additional-delta-aircraft-including-widebodies/
[17] Paul Almasan, Miquel Ferriol-Galmés, Jordi Paillisse, José Suárez-Varela, Diego Perino, Diego López, Antonio Agustin Pastor Perales, Paul Harvey, Laurent Ciavaglia, Leon Wong, Vishnu Ram, Shihan Xiao, Xiang Shi, Xiangle Cheng, Albert Cabellos-Aparicio, and Pere Barlet-Ros. 2022. Network Digital Twin: Context, Enabling Technologies, and Opportunities. *IEEE Communications Magazine* 60, 11 (Nov. 2022), 22–27. doi:10.1109/MCOM.001.2200012
[18] Katherine Anderson, Sonntag Barbara, Ryan William, Kavvada Argyro, and Lawrence Friedl. 2017. Earth observation in service of the 2030 Agenda for Sustainable Development. *Geo-spatial Information Science* 20, 2 (apr 2017), 77–96. doi:10.1080/10095020.2017.1333230 Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10095020.2017.1333230.
[19] Brian Baker-McEvilly, Surabhi Bhadauria, David Canales, and Carolin Frueh. 2024. A comprehensive review on Cislunar expansion and space domain awareness. *Progress in Aerospace Sciences* 147 (May 2024), 101019. doi:10.1016/j.paerosci.2024.101019
[20] Suvam Basak, Amitangshu Pal, and Debopam Bhattacherjee. 2025. LEOCraft: towards designing performant LEO networks. In *Proceedings of the 2025 USENIX Conference on Usenix Annual Technical Conference* (Boston, MA, USA) *(USENIX ATC '25)*. USENIX Association, USA, Article 47, 25 pages.
[21] Xuyang Cao and Xinyu Zhang. 2023. SaTCP: Link-Layer Informed TCP Adaptation for Highly Dynamic LEO Satellite Networks. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*.
[22] Sandra Erwin. 2025. U.S. Space Force forecasts $2.3 billion in commercial satellite services contracts. http://spacenews.com/u-s-space-force-forecasts-2-3-billion-in-commercial-satellite-services-contracts/
[23] Yue Gao, Kun Qiu, Zhe Chen, Wenjun Zhu, Qi Zhang, Handong Luo, Quanwei Lin, Ziheng Yang, and Wenhao Liu. 2024. Plotinus: A Satellite Internet Digital Twin System. doi:10.48550/arXiv.2403.08515 arXiv:2403.08515 [cs].
[24] Michael Grieves. 2014. Digital Twin: Manufacturing Excellence through Virtual Factory Replication. (2014). https://www.3ds.com/fileadmin/PRODUCTS-SERVICES/DELMIA/PDF/Whitepaper/DELMIA-APRISO-Digital-Twin-Whitepaper.pdf
[25] Gurobi Optimization, LLC. 2024. Gurobi Optimizer Reference Manual.
[26] Liz Izhikevich, Manda Tran, Katherine Izhikevich, Gautam Akiwate, and Zakir Durumeric. 2024. Democratizing LEO Satellite Network Measurement. *Proc. ACM Meas. Anal. Comput. Syst.* 8, 1 (Feb. 2024), 13:1–13:26. doi:10.1145/3639039
[27] Joern Krause. 2024. Non-Terrestrial Networks (NTN). https://www.3gpp.org/technologies/ntn-overview
[28] Mohamed M. Kassem and Nishanth Sastry. 2024. xeoverse: A Real-time Simulation Platform for Large LEO Satellite Mega-Constellations. In *2024 IFIP Networking Conference (IFIP Networking)*. 1–9. doi:10.23919/IFIPNetworking62109.2024.10619898 ISSN: 1861-2288.
[29] Simon Kassing, Debopam Bhattacherjee, André Baptista Águas, Jens Eirik Saethre, and Ankit Singla. 2020. Exploring the "Internet from space" with Hypatia. In *Proceedings of the ACM Internet Measurement Conference (IMC '20)*. Association for Computing Machinery, New York, NY, USA, 214–229. doi:10.1145/3419394.3423635
[30] Zeqi Lai, Hewu Li, Yangtao Deng, Qian Wu, Jun Liu, Yuanjie Li, Jihao Li, Lixin Liu, Weisen Liu, and Jianping Wu. 2023. {StarryNet}: Empowering Researchers

to Evaluate Futuristic Integrated Space and Terrestrial Networks. 1309–1324. https://www.usenix.org/conference/nsdi23/presentation/lai-zeqi

[31] Zeqi Lai, Hewu Li, and Jihao Li. 2020. StarPerf: Characterizing Network Performance for Emerging Mega-Constellations. In *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. 1–11. doi:10.1109/ICNP49622.2020.9259357

[32] Massimo Lauria and Maria Azzalin. 2024. Digital Transformation in the Construction Sector: A Digital Twin for Seismic Safety in the Lifecycle of Buildings. *Sustainability* 16, 18 (Jan. 2024), 8245. doi:10.3390/su16188245 Number: 18 Publisher: Multidisciplinary Digital Publishing Institute.

[33] Ji-Woon Lee, Byungju Lim, Ki-Hun Kim, Jong-Man Lee, Young-Seok Ha, Young-Jin Han, and Young-Chai Ko. 2025. Handover strategy for LEO satellite communication using graph neural network. *ICT Express* 11, 2 (April 2025), 239–244. doi:10.1016/j.icte.2025.01.009

[34] Xin Li, Min Feng, Youhua Ran, Yang Su, Feng Liu, Chunlin Huang, Huanfeng Shen, Qing Xiao, Jianbin Su, Shiwei Yuan, and Huadong Guo. 2023. Big Data in Earth system science and progress towards a digital twin. *Nat Rev Earth Environ* 4, 5 (May 2023), 319–332. doi:10.1038/s43017-023-00409-w Publisher: Nature Publishing Group.

[35] Wenhao Lu, Zhiyuan Wang, Hefan Zhang, Shan Zhang, and Hongbin Luo. 2025. OpenSN: An Open Source Library for Emulating LEO Satellite Networks. *IEEE Transactions on Parallel and Distributed Systems* 36, 8 (2025), 1574–1590. doi:10.1109/TPDS.2025.3575920

[36] Wenhao Lu, Zhiyuan Wang, Hefan Zhang, Shan Zhang, and Hongbin Luo. 2025. OpenSN: An Open Source Library for Emulating LEO Satellite Networks. *IEEE Transactions on Parallel and Distributed Systems* 36, 8 (2025), 1574–1590. doi:10.1109/TPDS.2025.3575920

[37] Ciara McGrath, Christopher Lowe, Malcolm Macdonald, and Steven Hancock. 2022. Investigation of very low Earth orbits (VLEOs) for global spaceborne lidar. *CEAS Space J* 14, 4 (Oct. 2022), 625–636. doi:10.1007/s12567-022-00427-2

[38] Nitinder Mohan, Andrew E. Ferguson, Hendrik Cech, Rohan Bose, Prakita Rayyan Renatin, Mahesh K. Marina, and Jörg Ott. 2024. A Multifaceted Look at Starlink Performance. In *Proceedings of the ACM Web Conference 2024 (WWW '24)*. Association for Computing Machinery, New York, NY, USA, 2723–2734. doi:10.1145/3589334.3645328

[39] Wajeeha Nasar, Ricardo Da Silva Torres, Odd Erik Gundersen, and Anniken T. Karlsen. 2023. The Use of Decision Support in Search and Rescue: A Systematic Literature Review. *ISPRS International Journal of Geo-Information* 12, 5 (May 2023), 182. doi:10.3390/ijgi12050182 Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

[40] Tobias Pfandzelter and David Bermbach. 2022. Celestial: Virtual Software System Testbeds for the LEO Edge. In *Proceedings of the 23rd ACM/IFIP International Middleware Conference* (Quebec, QC, Canada) *(Middleware '22)*. Association for Computing Machinery, New York, NY, USA, 69–81. doi:10.1145/3528535.3531517

[41] Felecia Wellington Radel. 2024. 'Lifesaver': How iPhone's satellite mode helped during Hurricane Helene. *USA TODAY* (Oct. 2024). https://www.usatoday.com/story/tech/2024/10/11/iphone-satellite-mode-att-cell-service/75616990007/

[42] Brandon Rhodes. 2019. Skyfield: High precision research-grade positions for planets and Earth satellites generator. *Astrophysics Source Code Library* (July 2019), ascl:1907.024. https://ui.adsabs.harvard.edu/abs/2019ascl.soft07024R ADS Bibcode: 2019ascl.soft07024R.

[43] Jayanth Shenoy, Om Chabra, Tusher Chakraborty, Suraj Jog, Deepak Vasisht, and Ranveer Chandra. 2024. CosMAC: Constellation-Aware Medium Access and Scheduling for IoT Satellites. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking* (Washington D.C., DC, USA) *(ACM MobiCom '24)*. Association for Computing Machinery, New York, NY, USA, 724–739. doi:10.1145/3636534.3690657

[44] Kanishka Singh and Kanishka Singh. 2024. Dozens of Starlink systems deployed for Hurricane Helene by Biden administration. *Reuters* (Sept. 2024). https://www.reuters.com/world/us/trump-says-he-spoke-musk-about-getting-internet-areas-hit-by-hurricane-2024-09-30/

[45] Amazon Staff. 2025. 15 photos from Project Kuiper's first launch of low Earth orbit satellites. https://www.aboutamazon.com/news/innovation-at-amazon/project-kuiper-photos-internet-satellite-first-launch

[46] Starlink [@Starlink]. 2025. Starlink is connecting more than 8M active customers with high-speed internet across more than 150 countries, territories, and many other markets. Thank you to all our customers around the world! http://starlink.com/stories-8M https://t.co/tfz96rdfMD. https://x.com/Starlink/status/1986168985453490449

[47] Gregory Stock, Juan A. Fraire, and Holger Hermanns. 2022. Distributed On-Demand Routing for LEO Mega-Constellations: A Starlink Case Study. In *2022 11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop (ASMS/SPSC)*. 1–8. doi:10.1109/ASMS/SPSC55670.2022.9914716 arXiv:2208.02128 [cs].

[48] Fei Tao, He Zhang, Ang Liu, and A. Y. C. Nee. 2019. Digital Twin in Industry: State-of-the-Art. *IEEE Transactions on Industrial Informatics* 15, 4 (April 2019), 2405–2415. doi:10.1109/TII.2018.2873186

[49] Liying Wang, Qing Li, Yuhan Zhou, Zhaofeng Luo, Donghao Zhang, Shangguang Wang, Xuanzhe Liu, and Chenren Xu. 2025. Emulating space computing networks with RHONE. In *Proceedings of the 2025 USENIX Conference on Usenix Annual Technical Conference* (Boston, MA, USA) *(USENIX ATC '25)*. USENIX Association, USA, Article 48, 17 pages.

[50] Hao Wu, Yizhan Han, Mohit Rajpal, Qizhen Zhang, and Jingxian Wang. 2025. SaTE: Low-Latency Traffic Engineering for Satellite Networks. In *Proceedings of the ACM SIGCOMM 2025 Conference (SIGCOMM '25)*. Association for Computing Machinery, New York, NY, USA, 896–916. doi:10.1145/3718958.3750524

[51] Jiasheng Wu, Shaojie Su, Xiong Wang, Jingjing Zhang, and Yue Gao. 2024. Accelerating Handover in Mobile Satellite Network. In *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*. 531–540. doi:10.1109/INFOCOM52122.2024.10621115 ISSN: 2641-9874.

[52] Peng Xu, Mingjie Feng, Jiaxi Zhou, Lixia Xiao, Pei Xiao, and Tao Jiang. 2024. Inter-Satellite Routing for LEO Satellite Networks: A GNN and DRL Integrated Approach. In *2024 IEEE/CIC International Conference on Communications in China (ICCC)*. 1346–1351. doi:10.1109/ICCC62479.2024.10681995 ISSN: 2377-8644.