Engineering 56 (2026) 62–78

Contents lists available at ScienceDirect

Engineering

journal homepage: www.elsevier.com/locate/eng

Research
6G: From Theory to Practice—Article

A High-Fidelity and High-Efficiency Simulator for 6G-Integrated Space–Ground Networks

Haibo Zhou [a,*], Xiaoyu Liu [a], Xin Zhang [a], Xiaohan Qin [a], Mengyang Zhang [a], Yuze Liu [a], Weihua Zhuang [b], Xuemin (Sherman) Shen [b,*]

[a] School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China
[b] Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada

ARTICLE INFO

Article history:
Received 2 December 2024
Revised 16 July 2025
Accepted 22 August 2025
Available online 20 September 2025

Keywords:
Low-Earth-orbit satellite networks
Discrete event simulation architecture
Parallel and distributed simulation
Performance evaluation

ABSTRACT

Mega-constellation networks have recently gained significant research attention because of their potential for providing ubiquitous and high-capacity connectivity in future sixth-generation (6G) wireless communication systems. However, the high dynamics of network topology and large scale of a mega-constellation pose new challenges to constellation simulation and performance evaluation. To address these issues, we introduce UltraStar, a high-fidelity and high-efficiency computer simulator to support the development of 6G wireless communication systems with low-Earth-orbit mega-constellation satellites. The simulator facilitates the design and performance analysis of various algorithms and protocols for network operation and deployment. We propose a systematic, scalable, and comprehensive simulation architecture for the high-fidelity modeling of network configurations and for performing high-efficiency simulations of network operations and management capabilities, while providing users with intuitive visualizations. We capture heterogeneous topology characteristics by establishing an environment update algorithm that incorporates real ephemeris data for satellite orbit prediction, sun outages, and link handovers. For a realistic simulation of software and hardware configurations, we develop a Network Simulator 3 based network model to support networking protocol extensions. We propose a message passing interface-based parallel and distributed approach with multiple cores or machines to achieve high simulation efficiency in large and complex network scenarios. Experimental results demonstrate the high fidelity and efficiency of UltraStar can help pave the way for 6G integrated space–ground networks.

© 2025 THE AUTHORS. Published by Elsevier LTD on behalf of Chinese Academy of Engineering and Higher Education Press Limited Company. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

The demand for global connectivity and seamless communication has become increasingly urgent in the sixth-generation (6G) era [1–3]. However, traditional ground-based networks have several limitations, including expensive deployment, insufficient coverage in remote areas, high latency, vulnerability to disruptions from natural disasters, and infrastructure damage [4,5]. Consequently, low-Earth-orbit (LEO) satellite networks have emerged and rapidly developed as vital complements to existing ground-based networks [6,7]. LEO satellites, which operate in orbits closer to the Earth's surface (~1200 km), minimize transmission latency. This makes them ideal candidates for real-time data transmission applications including video conferencing, online gaming, and autonomous vehicle communications [8,9]. Further, LEO satellite constellations, which are composed of numerous satellites working together, provide global coverage. This distributed network architecture ensures seamless connectivity even in remote and underserved areas, bridging the digital divide and granting Internet access to previously unreachable populations [10,11]. Companies and organizations such as SpaceX's Starlink and OneWeb have made substantial investments in LEO satellite networks. These initiatives aim to deploy thousands of satellites for creating a dense coverage mesh worldwide. Advancements in satellite miniaturization, launch capabilities, and inter-satellite communication have significantly accelerated the development of LEO satellite networks [12,13]. Therefore, driven by the promising potential of future LEO satellite constellations, research on topology simulation, network

* Corresponding authors.
E-mail addresses: haibozhou@nju.edu.cn (H. Zhou), sshen@uwaterloo.ca (X. Shen).

https://doi.org/10.1016/j.eng.2025.08.042
2095-8099/© 2025 THE AUTHORS. Published by Elsevier LTD on behalf of Chinese Academy of Engineering and Higher Education Press Limited Company.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

evaluation, and performance optimization has become essential. However, achieving high-fidelity and high-efficiency simulations for large-scale LEO satellite networks has three critical challenges.

The first challenge is accurately representing the dynamic and heterogeneous nature of topologies in high-dimensional complex environments. The space backbone of a LEO satellite network moves at a high orbital velocity relative to the Earth's surface unlike conventional terrestrial networks where routers and switches are static [14]. Ground–satellite links in a space–ground integrated network can experience frequent disruptions and re-associations because of LEO satellite dynamics, thereby resulting in frequent network topology fluctuations. In addition, in contrast to the backbone of conventional terrestrial networks deployed in closely protected environments, the space backbone is exposed to public and uncontrollable environments. Further, network failures, including node and link failures, are likely to occur because of a series of unique characteristics [15] such as satellite dynamics, environmental risks in complex outer space, and small satellite vulnerabilities. However, existing studies have focused on simplified simulation scenarios, neglecting critical factors such as high-precision satellite orbit prediction, complex space environment disturbances, inherent high mobility of space-based backbones, and predictable and unpredictable node/link failures. The high-fidelity modeling of network topology dynamics in high-dimensional complex environments is crucial for addressing this challenge.

The second challenge is replicating actual network models by realistically reproducing the communication environment, node hardware/software configurations, and diverse application behaviors. Channel modeling on non-terrestrial networks (NTNs), including channel impairments and propagation delays, is necessary to accurately simulate the communication conditions experienced by these networks. Further, capturing the diverse configurations of nodes, including their processing capabilities and software stack variations, is essential for creating realistic network simulations that can reflect the complexities of real-world systems. Finally, the diversity of application behaviors must be considered. Recognizing and accommodating these diverse behaviors can enable network designs and simulation models to better reflect real-world scenarios and provide more accurate insights into the system performance under varying conditions.

The third challenge is ensuring the time-efficient simulation of complex network scenarios. Discrete event simulations of large-scale networks in the serial mode are computationally prohibitive when modeling systems with numerous nodes, intricate networking protocols, and high computational complexity. One effective approach is leveraging parallel and distributed simulation techniques, which can significantly reduce simulation time by distributing the computational workload across multiple processors or computing resources. However, the application of traditional parallel synchronous algorithms to dynamic network simulations can generate errors in event ordering and causality. In addition, effective parallel task decomposition algorithms need to be developed to balance the load on various computational resources. Currently, effective parallel and distributed simulation schemes tailored for large-scale LEO satellite networks remain lacking.

In conclusion, the three aforementioned challenges in the design, analysis, and optimization of LEO satellite networks severely hinder the validation of new technologies and protocols, causing delays in the deployment of network infrastructure. To overcome these challenges, we developed a high-fidelity and high-efficiency simulator that provides users with a realistic environment of 6G integrated space–ground networks. Our contributions are summarized as follows:

- **Filling gaps in the simulation architecture:** We developed UltraStar, a high-fidelity and high-efficiency simulator designed to support a comprehensive analysis of large-scale LEO satellite networks. UltraStar features a systematic and scalable simulation architecture incorporating logical, control, and data planes. It supports high-fidelity modeling of target network configurations and the high-efficiency driving of simulation tasks while providing users with intuitive visualizations.
- **Enhancing system simulation fidelity:** We propose a custom discrete event-based satellite network simulation scheme for simulating various event-triggering and handling mechanisms in networking scenarios. To capture the topological characteristics, we designed an environmental model for highly dynamic large-scale LEO satellite networks by incorporating real ephemeris data for orbit prediction, sun outages, and link handovers. Further, we designed a Network Simulator 3 (NS-3) stack-based system-level and link-level integrated protocol simulation scheme to flexibly support protocol extensions to achieve realistic simulations of software and hardware configurations. Moreover, we maintained an actual data-trace-based audio and video application model to consider realistic traffic patterns.
- **Breaking through efficiency bottlenecks:** We designed a message passing interface (MPI)-based parallel and distributed simulation scheme with multiple cores or machines to improve simulation efficiency in large and complex scenarios. We use the MPI to build data communication between system processes. Subsequently, we designed a time-window-based parallel synchronization algorithm with a dynamic look-ahead to accommodate the high topology dynamics. The experimental results show that the proposed scheme provides significant acceleration with acceptable amplification factors compared to that of the serial scheme.

The remainder of this paper is organized as follows: Section 2 reviews the related work and identifies the limitations of existing approaches. Section 3 presents the overall architecture of the simulator and provides a macro-level view of its system components and operational workflow. Section 4 focuses on the module design of UltraStar kernel and details key functional modules that support high-fidelity simulations. Section 5 explores the implementation of parallel and distributed simulations, which serve as a foundation for ensuring simulation efficiency. Subsequently in Section 6, a series of case studies across the physical, link, and network layers are presented to demonstrate the application capabilities and performance advantages of the simulator. Finally, Section 7 concludes the thesis by summarizing its key contributions and outlining future research directions.

2. Related works

Several simulation platforms have been introduced to replicate the behavior and characteristics of LEO satellite networks in controlled environments. For example, a network simulation platform was proposed in Ref. [16] to support mobility tracking in space, aerial, and terrestrial networks, as well as to verify and research various protocols. Kassing et al. [17] developed Hypatia, an NS-3 based LEO satellite network simulator; analyzed the network characteristics of three different small-scale constellations; and provided a topological visualization. A simulation platform called StarPerf was developed in Ref. [18] for mega-constellations, enabling performance estimation and understanding across different constellation options. This platform integrated performance simulations for high-mobility networks and constellation scaling for exploring topological variations and operating conditions. Lai et al. [19] developed a testbed based on public constellation

information and deployed the OrbitCast prototype, which is a hybrid Earth observation data delivery architecture that utilizes LEO constellations and geodistributed ground stations (GSs). Low-Earth-orbit network evaluation and planning (LEO-NEaP) platform, which is an integrated platform for evaluating and planning large-scale LEO satellite constellation networks by analyzing key metrics such as network coverage, system throughput, and round-trip delay, was presented in Ref. [20]. This platform analyzed and optimized satellite constellation deployment, gateway station deployment, and radio resource management for enhancing system performance. Leosatellites, which is a LEO satellite constellation simulation model that uses OMNeT++ and INET, was designed and implemented in Ref. [21]. Leosatellites leveraged the open-source satellite simulator (OS^3) to successfully integrate into the INET framework. A hierarchical digital twin (DT) network for satellite communication networks, called HDTN-SCN, was presented in Ref. [22]. This network comprised central-DT and edge-DT models that offered differentiated DT services. Further, it included dynamic DT migration, quality of service (QoS)-aware DT synchronization slices, and a locator–identifier–isolation addressing mechanism to enhance the model synchronization efficiency. These studies provide preliminary insights into the high dynamics of heterogeneous network topologies, spatiotemporal complexity of network behaviors, and variability of diverse service demands. The system designs of existing platforms suggest the development of a new system that can consider all aspects of functionality, fidelity, efficiency, and scalability.

We identified six key aspects that are most crucial in large-scale LEO satellite network simulations for drawing a comprehensive comparison with existing platforms from academia and industry: ① terrestrial network integration, which involves modeling GS infrastructure and terrestrial network connectivity with LEO constellations; ② inter-satellite link (ISL) support, which involves modeling communication links between satellites within the constellation; ③ link-level simulation support, which involves simulating modulation and coding schemes (MCSs) for satellite–ground microwave links and inter-satellite laser links; ④ fidelity, which covers the systematic assessment of scenario modeling accuracy, environmental condition simulation, and network configuration realism; ⑤ efficiency, which involves quantitative measurements of time-to-solution and computational resource consumption in large-scale network simulations; and ⑥ scalability, which includes comprehensive evaluations of simulator capacity for scenario and protocol stack scaling.

These aspects can help ensure simulation fidelity, efficiency, and platform scalability. Platforms such as Hypatia, LEO-NEaP, and Satellite Network Simulator 3 (SNS3) [23] face challenges in simulating ground-based networks that interact with LEO satellite constellations. Fast mobility and frequent handovers are inherent characteristics of LEO satellite networks, and ignoring these features can lead to an inaccurate capture of network availability, stability, and service performance. Building ISLs is a promising networking approach [24,25] where each satellite becomes a node in the space network, achieving optimal path transmission and forming a global communication network. However, platforms such as the SNS3 and VVPSTK lack these capabilities [26]. In addition, the support for simulating the physical layer mechanisms for satellite–ground microwave links and inter-satellite laser links is essential for ensuring the realism of the simulation results and accurately reflecting the effect of the physical environment. Some platforms such as Satellite Tool Kit (STK) [27], SNS3, and VVPSTK focus on simulating and evaluating various physical layer mechanisms, whereas others [16–22] do not support this functionality.

These three abovementioned indicators evaluate the completeness of satellite network simulation functions. In addition, three additional indicators are used to assess the simulation capabilities at the system level. Fidelity quantifies the replication of topology dynamics, networking behaviors, and service demands in real networks. In contrast to traditional terrestrial networks, space dynamics models are more complex and involve factors such as satellite orbit predictions, space environment disturbances, and more stringent connectivity conditions. Providing simulation support for scalable networking protocols and diverse service demands can further enhance system fidelity. A limited number of existing platforms such as STK and VVPSTK can provide high-fidelity simulations. Simulation platforms must prioritize efficiency as satellite networks continue to scale and network interactions become more complex. Inefficient simulation systems result in significant resource and time consumption and can struggle to handle large-scale scenarios. Supporting parallel and distributed simulation capabilities is critical for improving the efficiency of discrete-event simulations. Unfortunately, most platforms do not focus on this capability. The ability of a simulation system to provide scalability for simulation scenarios, protocols, and evaluation criteria is essential for conducting in-depth research scenarios and optimized simulation processes; however, most platforms lack scalability. For example, although SNS3 offers a high-fidelity NS-3-based satellite communication simulator, it is limited to the static simulation configurations of a single geostationary satellite and cannot support the simulation of LEO constellations. Therefore, the design, analysis, and optimization of large-scale LEO satellite networks presents novel challenges. These challenges create a critical demand for advanced simulation tools that can deliver both high fidelity and computational efficiency to enable comprehensive network-behavior analysis.

3. Simulator architecture design

3.1. Architecture overview

We introduced a high-fidelity and high-efficiency simulation architecture designed for replicating real-world space network environments and behaviors as faithfully as possible in a time-efficient manner, thereby providing a reliable testing medium for network statistics and analysis. The system architecture of UltraStar includes logical, control, and data planes, as indicated in Fig. 1. The following section provides a detailed overview of these planes.

3.1.1. Logical plane

The target 6G integrated space–ground network includes a dynamic spatiotemporal environment, terminals, a satellite access network, a satellite backbone transmission network, and diverse service demands. Users must describe the network model using sufficient parameters before creating a simulation scenario. A more detailed description of the network model parameters established in the logical plane can provide a more realistic and accurate corresponding network simulation scenario.

3.1.2. Control plane

This plane receives the model description information of the target network from the logical plane and drives the data plane to complete the simulation. This plane performs two functions: human–computer interaction and data-plane control. The first function implements the network model input and storage through a website user interface (Web UI) and supports the visualization of simulation scenarios. The second function reads the parameters and generates instructions from the master controller to drive various simulation elements to simulate the target LEO satellite network.

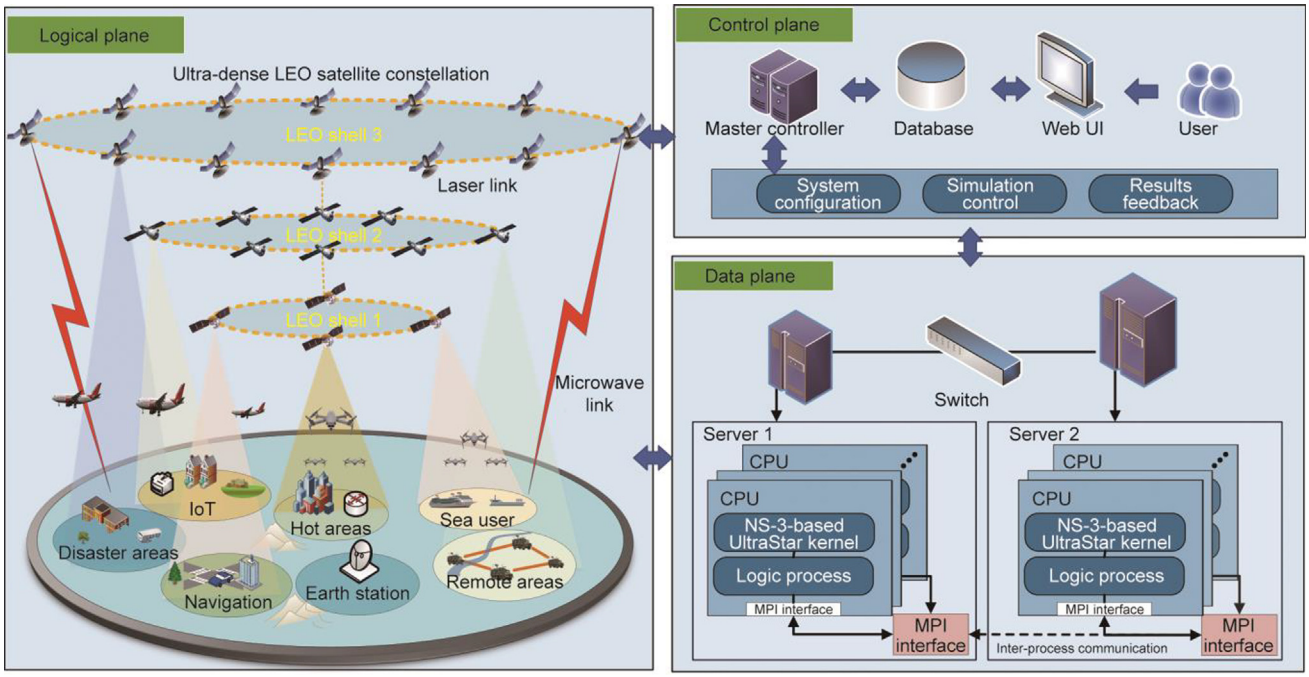

Fig. 1. Overview of the UltraStar architecture. IoT: Internet of Things; Web UI: website user interface; CPU: central processing unit.

3.1.3. Data plane

This plane provides the underlying simulation support for the entire simulation system. We build a 6G integrated space–ground networks simulation kernel based on NS-3 with a unified application programming interface (API) to maintain the construction of different scenarios, simulation of networking processes, and diverse service demands. The control plane issues instructions to the data plane and maintains the basic simulation process. The data plane feeds the simulation results back to the control plane and displays them through the westward human–computer interaction interface of the control plane. The data plane can also integrate and utilize different computing resources to perform parallel and distributed simulations through multi-core or multi-machine clusters to achieve high simulation timeliness.

3.2. System components

3.2.1. Web UI

The simulation system incorporates a Web UI to facilitate user interaction. The Web UI handles user requests, including the model parameter configuration, and initiates and terminates simulations. Further, it stores the configuration parameters in a database, communicates requests to the master controller to perform relevant calculations, and drives the simulation in the data plane. The Web UI also provides real-time visualization of the target network model and simulation data, offering users an intuitive understanding of the simulation process.

3.2.2. Master controller

The master controller interacts with the Web UI to execute core functions, including configuration file generation, simulation data precalculation, distributed system setting, and result summary. During the configuration file generation process, the master controller retrieves model parameters stored in the database and generates a configuration file in the JavaScript object notation (JSON) format. These configuration files served as inputs for the data plane simulation. The pre-calculation function runs the necessary computations in advance based on the simulation configuration, which involves invoking external tools or standard libraries as required. For example, it may call the National Aeronautics and Space Administration (NASA)'s Navigation and Ancillary Information Facility (NAIF) spacecraft, planet, instrument, camera-matrix, events (SPICE) toolkit to precalculate the trajectory of the Sun in space coordinates or calculate routing entries ahead of time. These pre-calculated data help optimize the simulation process and improve overall efficiency. In the distributed system setting function, the main controller configures multiple machines and organizes them into a cluster for executing distributed simulations. This function ensures that the simulation workload is efficiently distributed across machines, enabling parallel processing and enhancing overall performance. The resulting summary function involves the main controller aggregating and summarizing logs or tracked data recorded by multiple cores or machines. This information is consolidated into a final result document that provides a comprehensive overview of the simulation outcomes.

3.2.3. Database

The database stores comprehensive information on all simulated scenarios. We designed a scenario table with key fields to maintain a unique identifier, JSON-format configuration information, simulation status, output result, and creation time stamp for each scenario. The Web UI stores the configuration parameters of the scenarios in the database. Subsequently, the data plane updates the simulation status and records the output results in the database after the completion of the simulation using the provided configuration details.

3.2.4. UltraStar kernel

The NS-3-based UltraStar kernel integrates various simulation elements. The environment module addresses challenges associated with capturing the dynamics of the space environment and supports various constellation types and diverse heterogeneous network scenarios through topology and mobility modeling. The system-level module addresses the problem of Internet protocol

(IP) verification in emerging satellite networks. The kernel incorporates the NS-3 protocol stack and enhances it by introducing a wider range of standard open-source networking protocols through a protocol interface design. In addition, it provides simulation capabilities for audio and video applications using real data traces. The link-level simulation module overcomes the complexities of modeling complex satellite communication channels and numerous influencing factors. Further, the UltraStar kernel establishes a mathematical model for satellite-to-ground microwave and inter-satellite laser communications, enabling accurate and realistic simulations at the link level. This helps ensure the realistic restoration and simulation of node communication environments, as well as software and hardware configurations within the network model.

3.3. Operation process

As illustrated in Fig. 2, the operation process of the simulator progresses from a basic simulation configuration to a satellite Internet system-level simulation and finally to performance statistics. First, a basic simulation configuration that includes defining basic parameters, topology, and traffic for the simulation is established. The link-level simulation module then models the satellite-to-ground microwave and inter-satellite laser communication channels. This module supports the simulation of different coding and modulation techniques, provides an adaptive scheme selection strategy to optimize performance in terms of the block error rate (BLER), and provides the results to a system-level simulator based on MATLAB (MathWorks, Inc., USA). The satellite Internet system-level simulation is then conducted using NS-3, wherein communication protocols from the application layer to the physical layer are integrated and simulated to evaluate their performance. The interactions between these elements are modeled to simulate data transmission, routing protocols, and network load management. This stage includes assessing the capacity and reliability of the satellite network under different conditions such as traffic demands and link failures. Finally, performance statistics are generated using Python and MATLAB, which support routing evaluation and service analysis and provide comprehensive insights into the potential areas for optimization in protocol and algorithm design.

4. Module design of the UltraStar kernel

A detailed description of each module within the UltraStar kernel is shown in Fig. 3. We use "modules" to refer to the main functional components of the UltraStar kernel and "models" to represent the mathematical, physical, or computational models that constitute each module. The simulation management module is responsible for constructing a discrete event-based satellite network simulation technology and driving the scheduling and occurrence of events in the network. The environment module supports the construction of simulation elements such as satellites, GSs, terminals, and mobility and maintains the dynamic characteristics of the network topology through discrete events. The link-level simulation module is responsible for simulating the inter-satellite physical layer behavior, including MCSs such as digital video broadcasting (DVB) and Consultative Committee for Space Data Systems (CCSDS). The system simulation module based on the base network and link-level data constructed above simulates protocol interactions and networking processes in a relatively realistic network environment, including protocol stacks such as transmission control protocol (TCP)/IP and disruption-tolerant networking (DTN). In addition, the distributed simulation module is designed based on MPI parallel and distributed simulation technology, which further enhances the discrete event-based simulation efficiency.

4.1. Simulation management module design

The simulation management module based on the principles of discrete event simulation serves as the core component of the UltraStarkernel. Discrete event simulation is a modeling technique where events occur at distinct points in time and the system state changes only when these events occur. In this approach, the events are processed asynchronously. Each event has its own occurrence time, and the simulation engine handles them in the order of their occurrence. This asynchronous processing enables triggering and processing events at different points in time, accurately simulating the occurrence and response processes of events in the system. A discrete event simulation can faithfully model the behavior and interactions within the system by recording the timing and order of events.

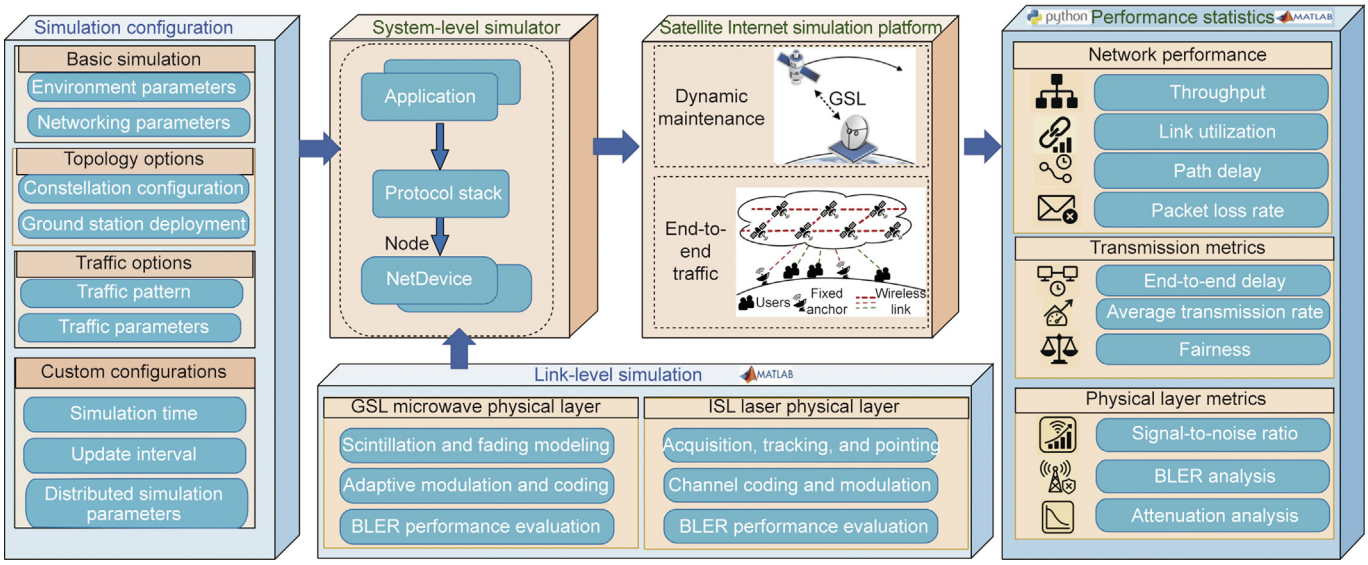

Fig. 2. Operation process of UltraStar simulator. GSL: ground-to-satellite link; BLER: block error rate.

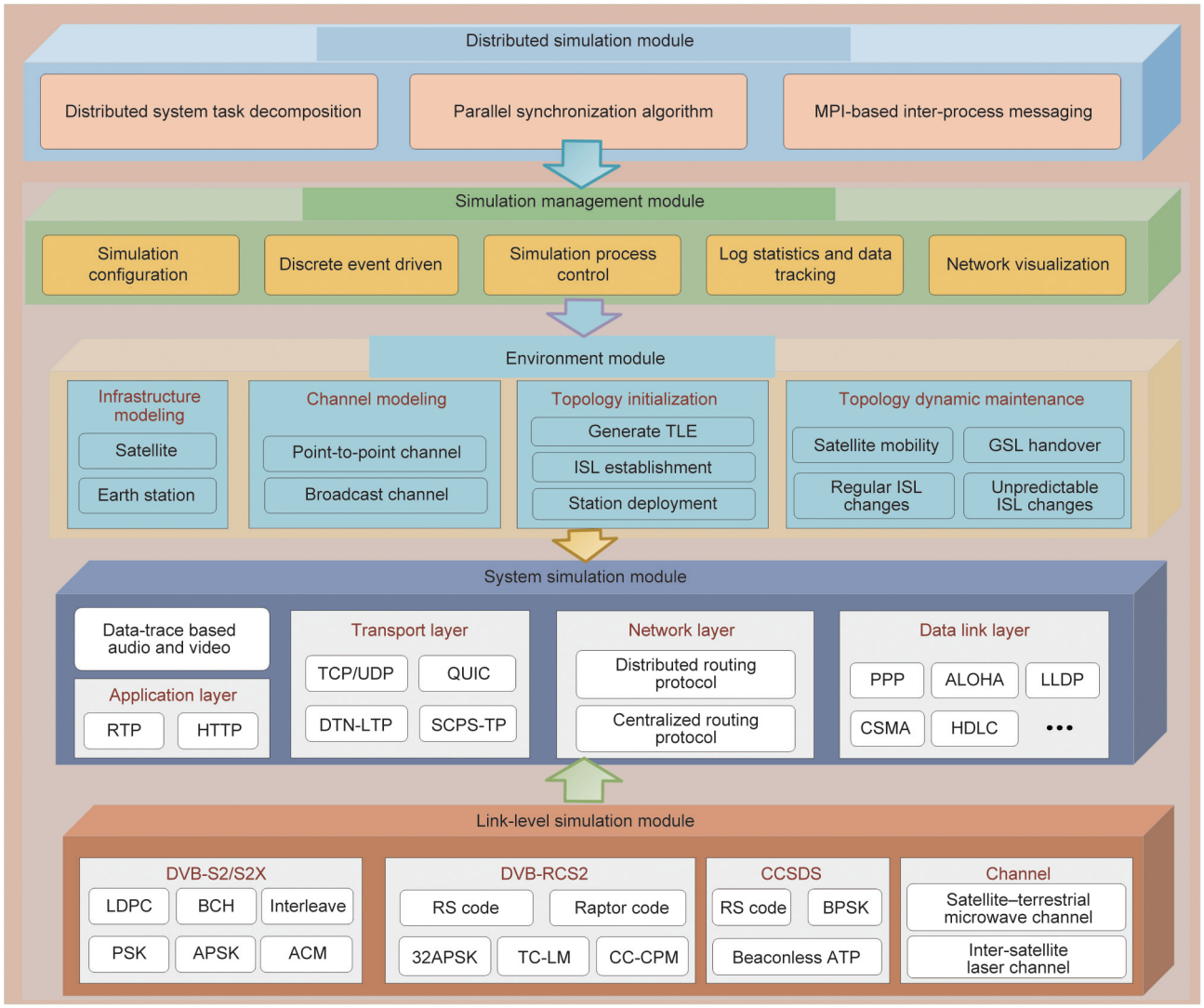

Fig. 3. Module design of UltraStar kernel. TLE: two-line element; RTP: real-time transport protocol; HTTP: hypertext transfer protocol; TCP: transmission control protocol; UDP: user datagram protocol; QUIC: quick UDP Internet connections; DTN: delay/disruption tolerant network; LTP: Licklider transmission protocol; SCPS-TP: space communications protocol standards transport protocol; PPP: point-to-point protocol; LLDP: link layer discovery protocol; CSMA: carrier sense multiple access; HDLC: high-level data link control; DVB-S2/S2X: digital video broadcasting-second generation satellite/second generation satellite extensions; DVB-RCS2: digital video broadcasting-return channel via satellite second generation; LDPC: low-density parity-check code; BCH: Bose–Chaudhuri–Hocquenghem code; PSK: phase shift keying; APSK: amplitude and phase shift keying; ACM: adaptive coding and modulation; RS: Reed–Solomon; TC-LM: telecommand link monitoring; CC-CPM: concatenated coded continuous phase modulation; BPSK: binary phase shift keying; ATP: automatic/advanced transport protocol; CCSDS: Consultative Committee for Space Data Systems.

In addition, the simulation management module within the UltraStar kernel is responsible for controlling the start and stop of the simulation process, as well as the allocation and release of resources. Further, this module maintains the selection of simulation modes, including both serial and parallel modes, as well as the algorithm selection in the parallel mode. The module manages essential simulation parameters, including the epoch in universal time coordinated (UTC) format, simulation duration, dynamic state update intervals, and random seed. It also maintains log records and tracks data to ensure the accurate reporting of simulation activities. In addition, it focuses on simulation statistics specific to the current central processing unit (CPU) core, providing a comprehensive summary of the simulation performance.

4.2. Environment module design

The environment module supports the creation of various simulation elements to facilitate a dynamic environmental topology, including nodes, structures, links, and mobility.

4.2.1. Node and structure model

As shown in Fig. 4, nodes represent different types of infrastructure or terminals, including satellites, aircraft, GSs, and user devices. Structures are collections of nodes with certain organizational structures and movement patterns such as constellations and drone swarms. We construct Walker constellations with symmetric and uniform configurations, including delta and star constellations. The delta type corresponds to inclined-orbit constellations designed for coverage in lower-latitude regions, whereas the star type corresponds to polar-orbit constellations designed for global coverage [28]. Users are provided the flexibility to define custom Walker constellation configurations. These configurations are uniquely represented by T, P, F, h, and u, where T, P, F, h, and u represent the total number of satellites in the constellation, number of orbits in the constellation, phasing factor ranging from 0 to $P-1$, orbital height, and orbit inclination. The phase offset (Δf) between satellites in adjacent orbit planes is determined by $\Delta f = 2\pi F/T$, which defines the relative positioning and phasing of the satellites within the constellation.

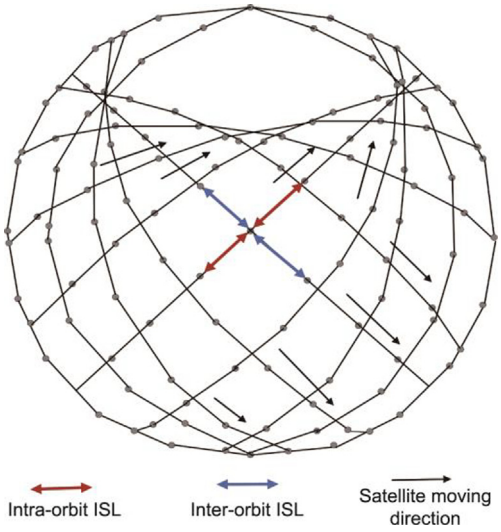

Fig. 4. Walker constellation.

We initialize a two-line element (TLE) [29] for all satellites to construct a complete Walker constellation. In addition to readily available parameters such as inclination, eccentricity, argument of perigee, and mean motion (revolutions per day), we also need to calculate the right ascension of the ascending node (RAAN) and mean anomaly (MA) based on the phase offset. The RAAN of the ith orbit (Ω_i) can be calculated as

$$\Omega_i = \Omega_1 + (i-1)\Delta\Omega, \quad \forall 1 \leq i \leq P \tag{1}$$

where $\Delta\Omega$ is the difference of the RAAN between adjacent orbit planes, and Ω_1 represents RAAN of the orbit where the seed satellite is located. For the delta type, the $\Delta\Omega$ is calculated as $\Delta\Omega = 2\pi/P$, and for the star type, it is calculated as $\Delta\Omega = \pi/P$. According to the MA of the seed satellite MA_{11} and phase offset Δf between satellites in adjacent planes, the MA of the first satellite in the ith orbit (MA_{i1}) can be expressed as

$$MA_{i1} = MA_{11} + (i-1)\Delta f, \ \forall 1 \leq i \leq P \tag{2}$$

Subsequently, owing to the evenly distribution of satellites in the same orbit, the phase difference ($\Delta\Phi$) between adjacent satellites can be given by $\Delta\Phi = 2\pi/(T/P)$, as shown in Fig. 5. The MA of the jth satellite in the ith orbit (MA_{ij}) can be expressed as

$$MA_{ij} = MA_{i1} + (j-1)\Delta\Phi, \quad \forall 1 \leq i \leq P, \ 1 \leq j \leq \frac{T}{P} \tag{3}$$

After generating the TLE of the initial constellation, we use it as input information for the mobility model to further maintain node mobility.

4.2.2. Mobility model

We maintained mobility models for different types of nodes to capture the high dynamics of the LEO satellite networks. We configured a stationary model for GSs characterized by

$$gnd_{postion} = \{longitude, \ latitude, \ altitude\} \tag{4}$$

For the satellites, we employed a simplified general perturbation 4 (SGP4) [17] orbit prediction model, which considers perturbations such as the Earth's nonspherical gravity, lunar and solar gravitational forces, solar radiation pressure, and atmospheric drag. This model provides relatively accurate orbit calculations based on the initial TLE and epoch, which can be represented as

$$sat_{postion} = SGP4_{model(initial_{TLE}, epoch)} \tag{5}$$

In addition, we integrated the NASA's NAIF SPICE toolkit [30], including the Jet Propulsion Laboratory Development Ephemeris 440 (JPL DE440) ephemeris, to maintain the position and motion of celestial bodies in the solar system, supporting sun outage simulations. Our system incorporates these mobility models, which enables it to accurately simulate the dynamic behavior of nodes, achieving realistic interactions and movements within the network.

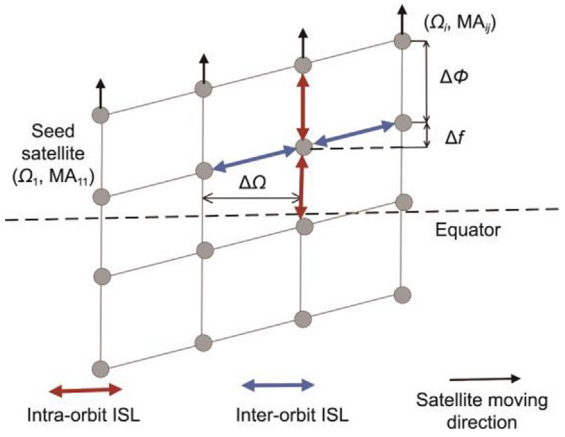

Fig. 5. Typical coordinate system orbit parameters. MA_{ij}: the MA of the jth satellite in the ith orbit.

4.2.3. Link model

The link simulation elements involve the simulation of ISLs and ground-to-satellite links (GSLs). According to the information outlined in the newly proposed constellation regulatory filing, there is a default configuration for building four ISLs for each satellite [28]. These ISLs connect adjacent satellites within the same orbital plane and adjacent satellites across neighboring orbital planes. A gap exists between the two opposing orbital planes in the Walker star-type constellations, and typically no links are established on either side of the gap.

Predictable and unpredictable ISL closures and reconstructions are modeled to simulate dynamic topology changes in the space environment. On the one hand, the inter-orbit ISLs are turned off in a polar region [31] because satellites in the same orbit are stationary relative to each other, whereas those in different orbitals are not. Antennas in two satellites in different orbits cannot always be aimed at each other because of physical limitations. This is predictable as the regular network topology changes. On the other hand, solar noise is a broadband noise, and the radiation intensity increases with frequency. Therefore, the impact of sun outage on the received signal-to-noise ratio (SNR) depends on the amount of solar noise, frequency bandwidth, and operating frequency. The laser ISL has a high frequency and large bandwidth, making it extremely susceptible to interference from sun outages. The laser link is interrupted for a certain period when the laser link is collinear with sunlight [32,33], as shown in Fig. 6. Sun-outage failures become predictable once the network topology changes. Unpredictable node and link closures refer to sudden topology changes caused by a variety of factors such as environmental risks in complex outer space and vulnerabilities of small satellites [15]. The increased constellation scale intensifies the switching of ISLs, which leads to greater network topology dynamics.

Establishing a GSL requires a sufficiently high elevation angle between satellites and GSs, and the range of this angle is constrained by the minimum ground elevation angle ε_{min}. A smaller minimum ground elevation angle can increase the number of visible satellites within the field of view of the GS, whereas a larger minimum ground elevation angle imposes stricter constraints on the establishment of GSLs. The GSLs undergo continuous handovers when satellites move at high speeds relative to the ground. With the minimum ground elevation angle ε_{min}, the maximum GSL

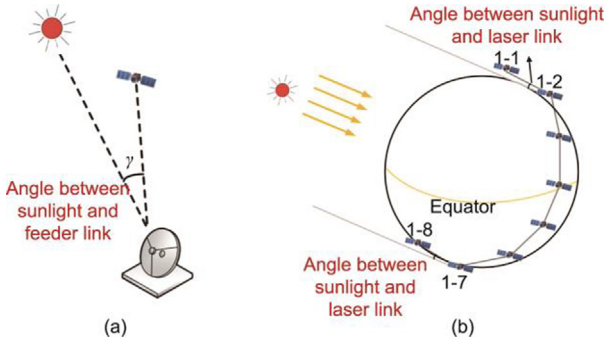

Fig. 6. Sun outage scenario. (a) Sun outage of feeder link; (b) sun outage of laser ISL. γ: angle between sunlight and feeder link.

distance (d_{max}) can be calculated by applying the Pythagorean theorem to a triangle, which results in

$$d_{max} = \sqrt{R^2 \sin^2(\varepsilon_{min}) + 2Rh + h^2} - R\sin(\varepsilon_{min}) \quad (6)$$

where R represents the Earth's radius. An object with an elevation angle $\varepsilon \geq \varepsilon_{min}$ toward the satellite can communicate with it; in other words, when the distance between them is $d \leq d_{max}$, the satellite can be considered to be within the line of sight.

4.2.4. Coordinate system model

Coordinate systems are used to represent the positions of objects on the Earth or in space. We introduced global and local coordinate systems and maintained conversions between typical coordinate systems [34]. Global coordinate systems can be classified into Earth-centered fixed (ECF) coordinate system and Earth-centered inertial (ECI) coordinate system. The ECF rotates along the central celestial body, whereas ECI does not. The J2000 epoch coordinate system was adopted for ECI. These systems use three-element coordinates to describe the positions of objects. In addition, we introduce the world geodetic system (WGS), which includes a set of standard geodetic coordinates for Earth, a reference ellipsoid used to calculate geodetic altitude data, and a set of gravity equipotential surface data used to define the mean sea level. Handling Julian dates and Earth orientation parameters, along with constants from the WGS, enable the conversion of ECI coordinates to ECF coordinates. Local coordinate systems include east–north–up, north–east–down, and azimuth–elevation–range systems. These systems require two sets of three-element coordinates: one to describe the position of the origin, and the other to describe the position of an object relative to the origin. This module maintains the coordinate conversion between these coordinate systems.

4.3. Link-level module design

The link-level module of UltraStar includes a channel for GSLs, channel for ISLs, MCSs, and link abstraction module [35].

4.3.1. Link budget for GSLs

The propagation loss of the GSLs is modeled using the NTN channel model defined in the 3rd Generation Partnership Project (3GPP) technical report (TR) 38.811. The NTN channel model provides three scenarios for simulating different propagation environments: dense urban, urban, and rural–suburban. Free-space loss, atmospheric attenuation, scintillation, and fast fading are considered in the 3GPP NTN channel model [36]. The link budget for the GSL is expressed as

$$P_R^{gsl} = P_T^{gsl} + G_T^{gsl} + G_R^{gsl} - PL_b - PL_A - PL_S - PL_F \quad (7)$$

where P_R^{gsl} represents the received signal power of GSL; P_T^{gsl} represents the transmitted power of GSL; G_T^{gsl} and G_R^{gsl} represent the transmitter and receiver antenna gains of GSL, respectively. Further, PL_b, PL_A, PL_S, and PL_F represent the free-space path loss, atmospheric attenuation, scintillation loss, and fast-fading loss, respectively.

4.3.2. Link budget for ISLs

The ISL channel model includes free-space path loss and pointing loss [37]. The link budget for the inter-satellite laser channel can be expressed as

$$P_R^{isl} = P_T^{isl} + G_T^{isl} + G_R^{isl} - PL_b + \lg(\eta_T \eta_R L_T L_R) \quad (8)$$

where P_R^{isl} represents the received signal power of ISL; P_T^{isl} represents the transmitter power of ISL; η_T and η_R represent the optical efficiencies of the transmitter and receiver, respectively; G_T^{isl} and G_R^{isl} represent the transmitter and receiver antenna gains of ISL, respectively; and L_T and L_R represent the pointing loss factors for the transmitter and receiver, respectively.

4.3.3. MCSs for GSLs and ISLs

The physical layer module of UltraStar supports various MCSs for both GSLs and ISLs tailored to DVB and CCSDS protocols. The link-level simulations in our simulator used the DVB standard, which is widely used in current LEO satellite constellations because it is robust against long delays and high-mobility satellite channels. Unlike orthogonal frequency-division multiplexing, which is sensitive to phase noise and Doppler shifts, DVB supports features such as beam hopping and flexible bandwidth usage, making it suitable for satellite networks. In addition, recent 3GPP TRs (TR 38.811 [38] and TR 38.821 [39]) have highlighted the feasibility of adopting DVB-like waveforms for NTNs, making it a practical and forward-compatible choice for future 6G systems. MATLAB was used to evaluate the BLER performance of each MCS across different signal-to-interference-and-noise ratios (SINR) and record the BLER curves for further system-level simulation in UltraStar [40].

4.3.4. Link-level abstraction

The link-level abstraction module simplifies the simulation of complex link conditions by approximating the performances of different MCSs and channel conditions. This module leverages the precomputed BLER curves using a link-level simulator to estimate the link quality without performing a detailed physical-layer MCS process in the system-level simulator. The link-level abstraction module in UltraStar first calculates the SINR for both ISLs and GSLs according to the channel model and maps the SINR to the BLER based on the pre-computed BLER results.

4.4. System-level module design

4.4.1. Network model

Our network simulation architecture is designed to support the NS-3 protocol stack, which comprises a reliable and ordered TCP/IP stack for traditional Internet communication as well as a store-and-forward delay/DTN stack tailored for delay-prone, intermittent, and high-latency environments. We combined the native TCP/IP stack of NS-3 to provide users with a convenient interface. Further, we incorporated additional open-source network protocols compliant with the request for comments standards, such as quick user datagram protocol (UDP) Internet connections (QUIC), border gateway protocol (BGP), open shortest path first (OSPF), and real-time transport protocol (RTP). For seamless integration, we developed data-exchange interfaces tailored to these protocols, which enabled smooth interoperability with the NS-3 stack. In addition, we incorporated a delay/DTN architecture. The core concept of a DTN is that it introduces a bundle layer between the transport and application layers, enabling asynchronous communication between nodes through the exchange of bundle messages. Unlike traditional Internet networks, DTN networks do not rely

on end-to-end paths and instead employ a store-and-forward message delivery approach. To support various types of DTN networks, the bundle protocol in the bundle layer supports multiple transport protocols through convergence layer adapters, including the TCP, UDP, and Licklider transmission protocol. The design and verification of new custom protocols in the stack and the implementation and comparison of traditional strategies can be implemented flexibly.

4.4.2. Traffic model

The traffic model comprises three components: synthetic codecs (syncodecs), traffic distribution, and applications.

Synthetic codecs: Syncodecs are a family of synthetic codecs used to generate synthetic video traffic. These methods are useful for evaluating real-time media-congestion controllers. The structure of the syncodec API is extensible; other codecs with extra functionality can be added in future versions by sub-classifying existing ones.

In syncodecs, a vector of bytes represents a payload (encoded frame) and can store useful information. The codec has a target bit rate that it attempts to output. At any time, the target bit rate can be read and set the target bitrate. Some typical codecs and audio and video application codes based on data traces are listed below:

(1) **PerfectCodec:** This codec provides the smoothest form of synthetic encoding by outputing packets or frames of constant size that match the configured maximum payload. The interval at which these packets or frames are provided remains constant but can be adapted when the user changes the target bitrate.

(2) **SimpleFpsBasedCodec:** This simple and effective implementation delivers a sequence of frames at specific intervals based on the reference frame rate. When necessary, the codec adjusts the frame size to achieve the configured target bit rate.

(3) **TraceBasedCodec:** This codec is an advanced synthetic implementation and generates a sequence of frames of realistic sizes corresponding to the output of a real codec from an offline video sequence. The codec uses multiple trace files as input, all of which refer to the same raw video sequence. Each trace file contains traces obtained by encoding the entire raw video sequence at a fixed resolution and target bitrate. The resolution refers to the fixed output resolution configured for offline encoding and can be one of the following strings: "90p," "180p," "240p," "360p," "540p," "720p," and "1080p." These strings correspond to the standard 16:9 pixel resolutions of 160×90, 320×180, 426×240, 640×360, 960×540, 1280×720, and 1920×1080 pixels. The target bitrate represents the fixed bitrate configured for the real video codec during offline encoding and is measured in kilobits per second.

Hence, users have the flexibility to select different syncodecs, which enables them to generate traffic with different patterns. Furthermore, they can encode real video sequences offline and employ them as trace file inputs to simulate audio and video applications in UltraStar.

Traffic distribution: The traffic distribution component of the model encompasses traffic distribution modeling and scheduling. After establishing the basic model of traffic distribution, the traffic scheduling component can be utilized to install corresponding applications for nodes with specific business needs. In this context, a typical flow f can be defined by an eight-tuple as

$$f = \{\mathrm{ID},\ \mathrm{SND},\ \mathrm{RCV},\ \mathrm{TGT},\ \mathrm{ST},\ \mathrm{DUR},\ \mathrm{SCD},\ \mathrm{SVC}\} \tag{9}$$

where ID represents the identification of each flow; SND and RCV represent the sending and receiving node numbers, respectively; TGT represents the target bitrate of this flow; ST and DUR represent the traffic start time and duration, respectively; and SCD and SVC represent the selected syncodec and Internet service, respectively. Finally, the traffic scheduling component installs specific sender and receiver applications on the sending and receiving nodes based on the information of each flow.

Applications: Applications are end–user applications that generate and consume synthetic traffic while maintaining application-layer protocol implementations for end-to-end data interaction. The system currently supports several typical application-layer protocols such as the hypertext transfer protocol (HTTP), file transfer protocol (FTP), and RTP to model typical applications such as web browsers, file transfers, and real-time audio/video streaming. These applications generate and consume traffic in simulated network environments. Further, they utilize supported application-layer protocols to facilitate data exchange and communication. For example, web browsers use HTTP to fetch and display web pages, file transfer applications utilize FTP to transfer files between clients and servers, and audio/video streaming applications rely on RTP for real-time transmission of media data. The system can simulate realistic traffic scenarios by supporting these application-layer protocols and modeling various applications.

Thus, the traffic model offers a comprehensive framework for generating and analyzing synthetic traffic that closely mimics real-world multimedia traffic patterns. This enables researchers and network engineers to evaluate and optimize the network performance under realistic traffic conditions, enhancing the effectiveness and accuracy of their analyses.

4.4.3. Transceiver model

Our simulation can adopt different antenna modules based on the type of device used. For satellites, a circular aperture antenna model, which is a reflective antenna that provides circular polarization, is recommended by 3GPP. For other flying vehicles such as high-altitude platforms (HAPs) or unmanned aerial vehicles (UAVs), a uniform planar array (UPA) antenna can be used, as defined by TR 38.901; this is the current standard for user equipment and evolved NodeB (eNB)/next generation NodeB (gNB) nodes in cellular networks. For ground terminals, 3GPP recommends using a UPA antenna or a very-small-aperture terminal (VSAT) antenna. The VSAT antenna includes a small (less than 1 m in diameter) circular parabolic reflector whose radiation pattern is similar to that of a satellite circular aperture antenna. This antenna is placed on a rooftop pointing toward the sky.

In addition, the error module can adopt various implementation methods according to different application scenarios and requirements. For example, the burst error module can simulate the burstiness of errors over time by introducing burst error sequences, which can indicate that errors occur in concentrated bursts during specific periods. A classical BLER model was used to simulate the mobility of the transceiver nodes, antenna gain, channel conditions, and other real-world environments using a system-level simulation for calculating the link SNR. Based on the SNR and physical layer standards employed, the corresponding BLER can be identified in the output curve of the link-level simulation with a specific MCS, which determines if data packets can be successfully received in the system-level simulation.

5. Module design of parallel and distributed simulation

5.1. MPI-based technical approach

NS-3 currently supports several parallel and distributed simulation approaches [41–46]. These approaches can be classified into local and distributed approaches. Local approaches involve deploying solutions for specific techniques using multithreading or graphics processing unit (GPU) technologies on a single machine.

Distributed methods use distributed computing resources available on high-performance computing (HPC) platforms. These methods rely on the MPI as a messaging protocol for efficient communication among distributed entities. The MPI, which serves as a standard library for message passing in parallel programming, plays a crucial role in facilitating communication and coordination among different processes in a distributed environment. NS-3 enables seamless integration with HPC platforms by leveraging the MPI, which enables the efficient utilization of distributed computing resources and enhances scalability and performance of network simulations. Considering the effectiveness and complexity of these approaches, we focused on designing a distributed simulation system based on MPI to conveniently and effectively utilize more distributed computing resources to increase simulation speed.

5.2. Parallel and distributed system design

5.2.1. System architecture

A parallel and distributed simulation system can be conceptualized as a composition of multiple interacting physical processes, where each physical process is abstracted as a logical process (LP). Each LP corresponds to a computational unit such as a CPU core, GPU, or distributed computing node, and it operates autonomously to simulate the behavior of its corresponding physical processes. These LPs communicate and synchronize with each other through a standardized messaging protocol, most commonly the MPI, which enables efficient data exchange across different computing devices.

Each LP maintains an event list in parallel and distributes discrete-event simulations. Computations performed by each LP include a series of event computations where each computation can modify the state variables and/or schedule new events for itself or for other LPs. Multiple LPs execute events in parallel to accelerate the simulation process while ensuring causality and processing events in the time stamp order. For simplicity, each process constructs a unified physical scenario while maintaining the same topological connections and address schemes. Through task decomposition among processes, the nodes can be divided into actual nodes maintained by the processes and shadow nodes that enable consistent scenario construction. Network operations such as protocol stacking and application installation are performed only on actual nodes, whereas shadow nodes solely contribute to scenario construction.

As shown in Fig. 7, we consider a simulation system with two LPs, namely LP_1 and LP_2. The grey-marked areas in LP_1 and LP_2 denote the regions of the actual nodes, whereas the other areas represent the shadow node regions. As an example, consider the flow demand from the sender to the sink, where the message sent from the sender undergoes intra-process communication and reaches the boundary node of the process. Then, it is forwarded through inter-process communication to the actual boundary node in LP_2. Inter-process communication triggers a remote event that represents a receiving event on a remote process node. The parallel and distributed systems maintain local causality constraints through time-synchronization algorithms, ensuring that each LP processes events in a non-decreasing time stamp order.

5.2.2. Synchronization algorithm

Existing time synchronization algorithms determine time advancement intervals to ensure event consistency by utilizing the time information exchanged between LPs to determine the range of time advancement. These helps ensure that no outdated messages are received during the time advancement process, avoiding inconsistencies in the system. The key aspect lies in calculating the time advancement interval, which is commonly referred to as the lower-bound time stamp (LBTS) in most algorithms. The high-level architecture (HLA) standard defines the concepts of the LBTS and look-ahead time management. The LBTS represents the maximum safe time advancement value for an LP, suggesting that no messages with time stamps smaller than this value are received in the future. The correct calculation of the LBTS is a crucial problem for the time synchronization algorithm. In distributed simulations, LBTS is commonly calculated using

$$T_{\text{lbts}} = \min_{\forall k \in N} \{T_k + \text{LA}_k\} \tag{10}$$

where T_{lbts} represents the time stamp of the next LBTS, N represents the LP set, LA_k represents the look-ahead of the kth LP (LP_k), and T_k represents the minimum time stamp of unprocessed events in LP_k. The look-ahead represents the predictive capability of a simulation node to anticipate its own future events, ensuring that no new events occur within the look-ahead period. This enables nodes to proactively notify other nodes of the time stamps of their upcoming events, enabling faster concurrent processing in the system.

The minimum inter-process channel delay is used as the look-ahead for network simulation scenarios. However, previous studies only considered static topology scenarios. Dynamic topology scenarios are updated in a snapshot manner based on the dynamic update time slot T_{slot}. Therefore, the minimum channel delay across processes changes over time. Failure to update the look-ahead in a timely manner can not only affect the efficiency of parallel simulations but also lead to local causality errors. Hence, we present a new LBTS calculation method tailored for dynamic topology updates.

$$T_{\text{lbts}} = \min_{\forall k \in N} \{T_k + \text{LA}_k,\ T_{\text{Next}}\} \tag{11}$$

where T_{Next} represents the next topology change time, which can be updated as $T_{\text{Next}} = T_{\text{Next}} + T_{\text{slot}}$.

Finally, as described in Algorithm 1, we propose a dynamic look-ahead value-based parallel synchronization algorithm for improving the simulation efficiency of large-scale integrated space–ground networks.

Algorithm 1. Dynamic look-ahead value-based parallel synchronization algorithm.

Require: Look-ahead LA_k of each LP_k, minimum time stamp T_k of unprocessed events in LP_k, minimum time stamp T_{lbts} to generate a new event, next topology change time T_{Next}, and dynamic update time slot T_{slot}
Ensure: Parallel simulation satisfying causality
1. Initialization: Update look-ahead LA_k to the minimum inter-process channel delay, $T_{\text{Next}} = T_{\text{slot}}$
2. **For** each LP_k **do**
3. **While** simulation in process **do**
4. Calculate lower bound on time stamp: $T_{\text{lbts}} = \min_{\forall i \in N} \{T_k + \text{LA}_k, T_{\text{Next}}\}$
5. Determine events safe to execute: S = set of events with time stamp $\leq T_{\text{lbts}}$
6. Process events in S
7. **If** $T_{\text{lbts}} = T_{\text{Next}}$ **then**
8. Calculate next topology change time: $T_{\text{Next}} = T_{\text{Next}} + T_{\text{slot}}$
9. Update look-ahead LA_k to the minimum inter process channel delay
10. **End if**
11. Enter barrier synchronization
12. **End while**
13. **End for**

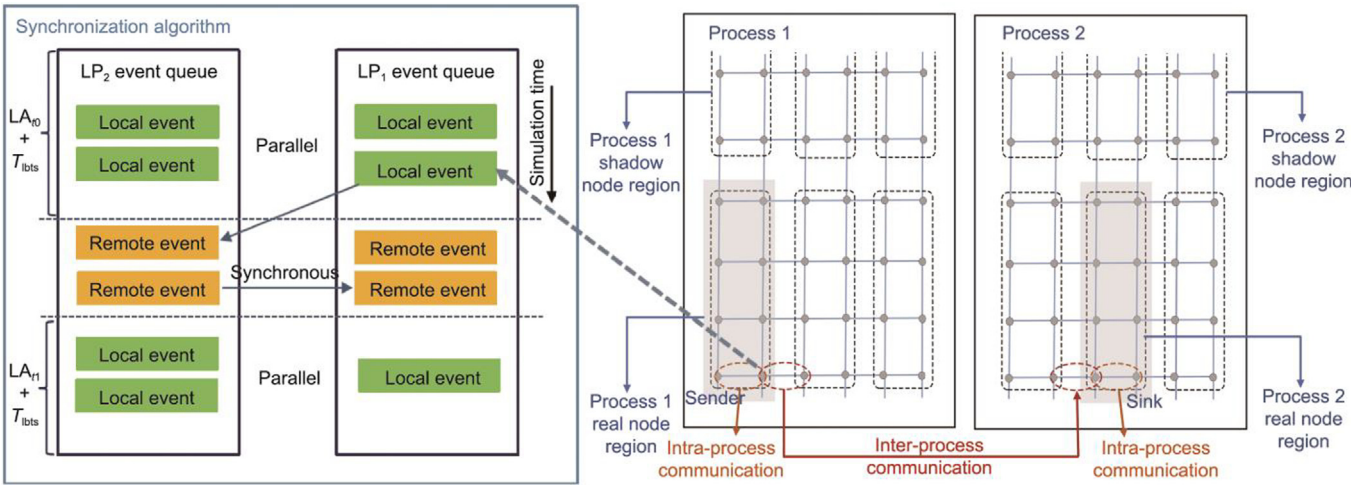

Fig. 7. Parallel and distributed simulation architecture. LA_{t0}: look-ahead during interval t_0; LA_{t1}: look-ahead during interval t_1; T_{lbts}: time stamp of the next lower-bound time stamp.

6. Case studies: 6G integrated space–ground networks via UltraStar simulation

The UltraStar simulator offers high fidelity, efficiency, and scalability, which enables multilayer protocol simulations and comprehensive performance analyses of 6G integrated space–ground networks. We conducted case studies on the physical, link, network, and transport layer in 6G integrated space–ground networks to showcase its capabilities. Furthermore, we conducted tests on the efficiency of the simulator.

6.1. Physical layer mechanism simulation

In this case study, we simulated the physical layer based on the StarLink phase I constellation configuration, which includes 72 orbital planes with 22 satellites per plane. Each satellite orbited at an altitude of 550 km and an inclination angle of 53°. The scenario considered five GSs: two located in Tokyo, Japan (Tokyo 1 and Tokyo 2; 35° 41′ N, 139° 41′ E), and the others situated in São Paulo, Brazil (23° 32′ S, 46° 38′ W), Mumbai, India (19° 04′ N, 72° 52′ E), and Shanghai, China (31° 13′ N, 121° 27′ E), respectively. The simulation employed a minimum-distance handover strategy in which a GS selects the nearest LEO to establish a GSL until the angle between the LEO and GS falls below the minimum elevation threshold, where it switches to the next-nearest LEO. The simulation integrated the link- and system-level modules by adopting the SINR and BLER curves provided by the link-level module.

For the satellite-to-ground link simulation, we set the center frequency as 20×10^9 Hz, with the scenario configured as a suburban/rural environment. The minimum ground elevation angle was set to 5°. The node transmission power was 50 dBm with a transmitting antenna gain of 40 dBi. For GS-to-LEO communications, we employed the digital video broadcasting-second generation satellite (DVB-S2) physical layer standard with the MCS set to SAT_MODCOD_8PSK_3_TO_4. For LEO-to-GS communications, the digital video broadcasting-return channel via satellite second generation (DVB-RCS2) physical layer standard was adopted, with waveform number three selected [36]. Fig. 8 shows how the SINR of the different GSL changed over time. "Satellite 727-GS 1588" had the highest SINR throughout the study period, which shows better performance than the other links. The other links start with similar SINR levels around 10 dB and gradually drop to around −4 dB by the end of the 500 s observation. This decrease can be attributed to factors such as increasing interference or changing satellite positions relative to GSs. The simulation results demonstrate the need to manage the link quality adaptively over time.

We set the transmission power to 50 dBm with transmit and receive antenna gains of 35 dB for the satellite-to-satellite link simulation. We configure the noise power spectral density as -174 dBm·Hz^{-1} with a channel bandwidth of 2 GHz. For both the transmitter and receiver, the optical efficiency was configured at 80% and radial pointing error angle was 0.00005 rad. For MCS, we configured the modulation order as 2 with a code rate of 239/255 [37].

In inter-satellite communication, the CCSDS protocol adopts a fixed MCS, and the observed BLER variations are driven by network dynamics such as signal degradation, interference, and environmental conditions. Fig. 9 illustrates the BLER variations over time for specific ISLs between satellite pairs: 0–1, 0–21, 252–253, and 253–254. Satellite pairs 0–1 and 0–21 exhibit a gradual increase in BLER, indicating a decline in link quality because of the relative movement or interference. In contrast, pairs 252–253 and 253–254 display a decreasing BLER trend, which suggests an improved link quality potentially because of alignment adjustments or reduced interference. The simulation results show that the BLER in ISLs is affected by dynamic conditions and satellite positioning.

For physical layer mechanism simulations, UltraStar can accurately model both GSLs and ISLs by incorporating factors such as free-space path loss, atmospheric attenuation, and solar scintillation.

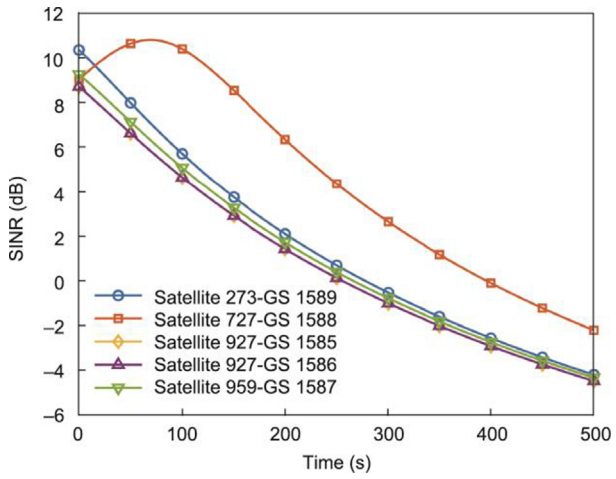

Fig. 8. Time-varying SINR of GSL.

In addition, UltraStar can dynamically compute key metrics such as SNR and bit error rate (BER) under time-varying channel conditions through integration with the link-level module. This capability provides crucial data support for researching adaptive MCSs in future space–air–ground integrated networks, which enables optimizing both spectral efficiency and anti-interference performance.

6.2. Link layer mechanism simulation

The proposed simulation platform is highly scalable and flexible, and it supports integration with multiple scenarios and algorithm frameworks. This provides a convenient experimental environment to develop a link-layer mechanism. In addition, the platform facilitates in-depth optimization of 6G technologies for various industrial applications. Among these, civil aviation is a key application domain. The platform enables the design of targeted technological solutions by simulating the handover performance in aviation scenarios, which ultimately enhances the service quality and operational efficiency of future 6G networks in industrial contexts.

6.2.1. Handover for civil aviation

We constructed a communication network scenario that included LEO satellites and civil aircraft using the simulation platform. The simulation obtained the network topology at each discrete time slot during flight by accurately modeling the satellite trajectories, aircraft flight paths, and satellite coverage parameters [47].

In the design of the handover strategy, we consider different priorities of civil aircraft; comprehensively consider the downlink rate, power overhead, and handover overhead indicators; and construct a handover evaluation function Q as a measure of the merits of the aircraft handover action evaluation [48]. The specific modeling is

$$Q = \mathrm{RRS} - \mathrm{PC} - \mathrm{LHC} \tag{12}$$

where RRS, PC, and LHC represent the aircraft receiver rate satisfaction function, power cost function, and link handover cost function, respectively. PC and LHC are constants for high-priority and low-priority aircraft, respectively. The aircraft receiver rate satisfaction function is modeled as

$$\mathrm{RRS} = 1 - \mathrm{e}^{\frac{R_{\mathrm{dr}}}{R_0}} \tag{13}$$

where R_{dr} and R_0 represent the downlink reception rate of the aircraft and downlink rate requirement value of the aircraft, respectively. The high-priority aircraft rate requirement is correspondingly higher than the low-priority aircraft rate requirement, and the power overhead cost and handover overhead value are lower than those of the low-priority aircraft.

A handover strategy is derived using a cooperative game-theory approach. At each discrete simulation time slot, the handover requested by the aircraft as a player obtains the relevant information required for calculating the handover evaluation function value through current access to the LEO satellite, such as the position information of the visible LEO satellite set and channel information. The handover-requested aircraft group plays the cooperative game to obtain the coordinated handover decision of the aircraft group; the goal is to maximize the handover evaluation function value of all handover cooperating players, namely,

$$\max \sum Q \tag{14}$$

The handover results are uploaded to the network via LEO satellites to update the information of the entire network.

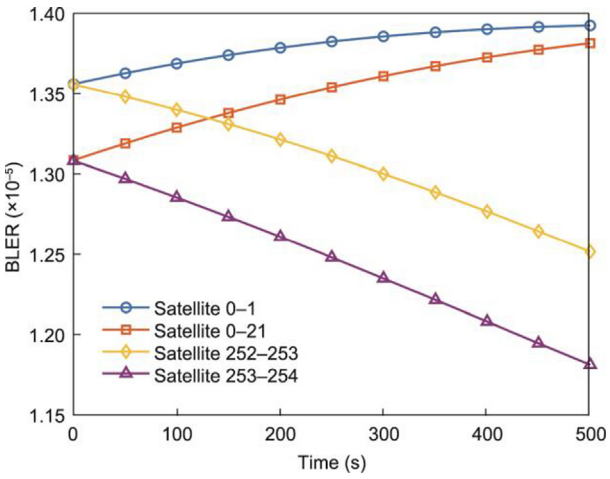

Fig. 9. Time-varying BLER of ISL.

6.2.2. Simulation results

We simulated Starlink different phase scenarios based on the simulation platform and input real flight data of 40 civil aviation aircraft located over the Indian Ocean into the simulation platform as civil aviation simulation objects [49]. The number of high-priority civil aircraft was set to 20, the number of low-priority civil aircraft was set to 20, and 180 discrete time slots were simulated with the duration of each discrete time slot being 10 s. We compared the proposed algorithm with two commonly used benchmark access strategies to evaluate handover performance: the minimum distance (MD) algorithm and the maximum rate (MR) algorithm. All evaluations were conducted within the UltraStar simulation framework.

Figs. 10 and 11 show the average received rate and number of handovers of civil aircraft for different priorities during the simulation period, respectively. The results confirm that the proposed algorithm outperforms the benchmark algorithms in terms of receiver rate and handover time for both high- and low-priority users in different constellation scenarios. For low-priority users, the proposed algorithm shows a strong service guarantee capability and can provide stable service quality in all scenarios, avoiding the significant experience gap attributed to the insufficient resource allocation. Handover times are core indicators of user experience and network stability. The proposed algorithm significantly reduces the number of handover times under different constellation scales using reasonable access decision and resource scheduling methods. Although the number of handovers is significantly reduced, the proposed algorithm can guarantee the optimal performance of the user-received rate, highlighting its ability to balance the efficient utilization of resources and service continuity.

For link-layer mechanism simulations, UltraStar can perform various handover types modeling, including real-world civil aviation flight scenarios. This simultaneously supports performance comparison and evaluation of different handover algorithms, which encompasses metrics such as handover frequency, handover delay, average reception rate, and QoS satisfaction levels. This provides a powerful tool to verify and optimize 6G space–air–ground integrated network handover management solutions.

6.3. Network routing simulation

The optimization schemes of interdomain routing protocols in satellite networks were evaluated using simulated satellite scenarios. We simulated LEO satellite networks mirroring standard Walker–delta LEO constellations with a uniform and symmetric setup: ① At an altitude of 215 km, 128 satellites were evenly distributed on 16 orbits at a 80° orbit inclination; ② at an altitude of

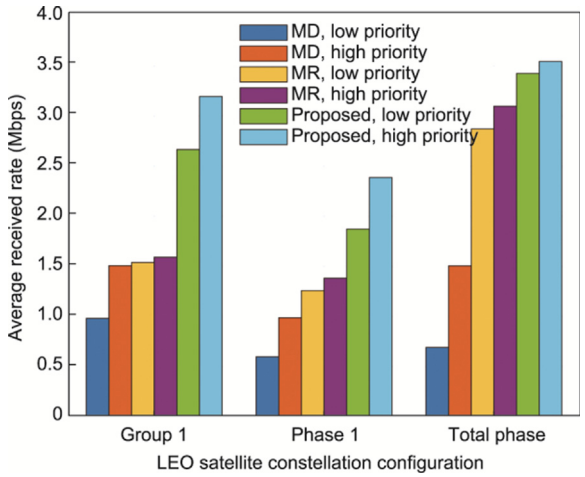

Fig. 10. Average received rate of civil aircraft. Mbps: megabits per second.

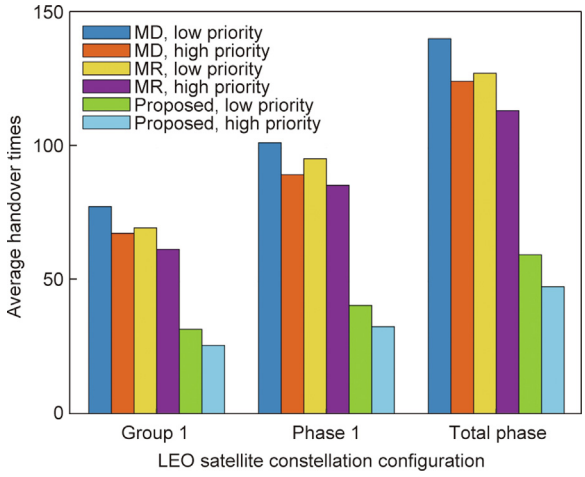

Fig. 11. Number of handover times of civil aircraft.

1015 km, 351 satellites were evenly distributed on 27 orbits at a 98.98° orbit inclination; ③ at an altitude of 570 km, 720 satellites were evenly distributed on 36 orbits at a 70° orbit inclination; ④ at an altitude of 550 km, 1584 satellites were evenly distributed on 72 orbits at a 53° orbit inclination. We uniformly configured the traffic as a constant flow conducted over a UDP at 20 megabits per second (Mbps) with each packet size set to 1 MB. Four metrics were selected to assess the BGP: the number of Internal border gateway protocol (iBGP) connections, BGP message overhead, network convergence delay, and packet loss rate. We compared performances of the full mesh (FM), route reflection (RR), and our proposed proactive hierarchical border gateway protocol (PH-BGP) at different network scales [50,51].

Although FM is barely manageable in small-scale satellite networks, and the number of iBGP connections expand to an intolerable level with an increase in the network scale in both real-world deployments and simulations. Although RR successfully controlled the number of iBGP connections across different constellation scales because of its scalability, and it was still significantly outperformed by the proposed scheme. In terms of the BGP message overhead, the BGP messages were directly correlated with the number of iBGP connections, which resulted in a similar trend between the three schemes, as shown in Fig. 12.

For the constellation of 720 satellites, we compared the variations among the three schemes in terms of convergence delay

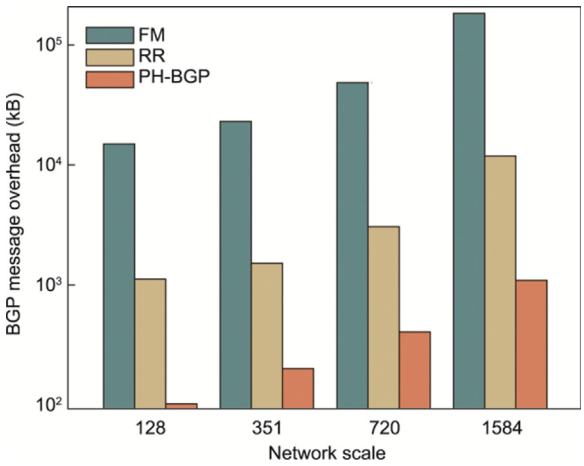

Fig. 12. Routing overhead of different schemes.

and packet loss. The cumulative distribution function (CDF) of the convergence delay is presented in Fig. 13. The average convergence delay of the proposed scheme (562 ms) is significantly lower than those of FM (969 ms) and RR (2972 ms). For packet loss, we varied the average data rate from 20 to 100 Mbps. The results confirm that the control plane performance of our proposed scheme directly improves the data forwarding efficiency.

The UltraStar simulation platform provides the ability to extend and improve network-layer protocols. Routing protocols can be simulated on various scales, particularly in large-scale satellite networks. The platform monitors and evaluates key performance indicators such as routing overhead, convergence delay, and packet loss rate. This enables assessing the adaptability and robustness of the protocol in scenarios with frequent topology changes, which is crucial for improving and developing more feasible and effective network-layer protocols for large-scale LEO satellite networks.

6.4. Transport protocol simulation

Based on the UltraStar simulation platform, we proposed a QUIC-enabled robust streaming transmission framework (QRST) in ultra large-scale LEO satellite networks (ULSLSNs) for real-time applications. Transmission latency critically affects the communication requirements of a user. Real-time applications have stringent delay constraints; a block becomes useless if it is received

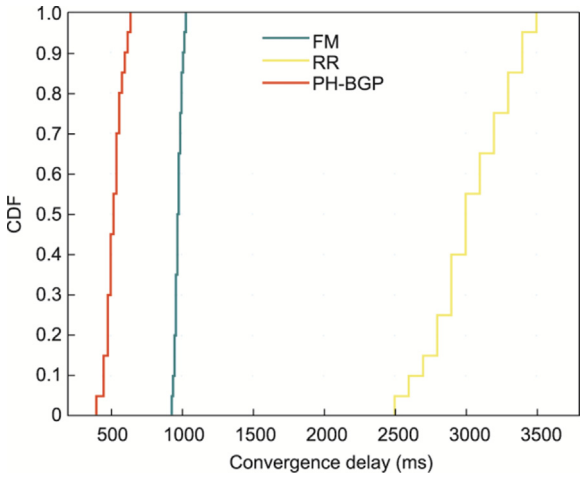

Fig. 13. Cumulative distribution function of the convergence delay.

after its deadline [52]. Some related studies focused on deadline-aware transmission by leveraging the QUIC protocol [53,54]; however, in ULSLSNs, the transmission of real-time applications was significantly affected by satellite mobility, limited bandwidth, and packet losses, making it challenging to satisfy the delay constraints for block transmission [55,56]. We proposed a framework using the QUIC protocol to enhance the transmission performance of real-time applications to reduce block transmission latency and deliver more blocks before the deadline.

6.4.1. Design of QRST

QRST ensure the timely delivery of blocks and improves the completion ratio, enhancing the transmission performance in ULSLSNs for real-time applications. The problem is divided into reducing the impact of packet losses on block transmission latency by adaptive forward error correction (FEC) coding and increasing the number of blocks delivered by block scheduling. QRST integrates these mechanisms and enhances the overall transmission performance for real-time applications. QRST proposes a transmission framework deployed on the sender side that comprises four components: a deadline-driven block scheduler, an adaptive FEC scheme, a transmission module, and a feedback module. On the sender side, the applications first deliver the blocks to the sender buffer. Subsequently, the block scheduler selects the block for transmission by considering multiple block properties to meet deadline requirements and improve the QoE of the user. Further, it considers the priority and deadline of blocks, employing a comprehensive scheduling approach to ensure that more high-priority blocks are delivered before the deadline. Once block scheduling is completed, the adaptive FEC scheme dynamically introduces repair symbols based on the current network conditions for enhancing data protection. This mechanism allocates an appropriate redundancy to simultaneously reduce block transmission latency and balance the bandwidth overhead according to the current network conditions. The block data are then encoded into source symbols and encapsulated within the QUIC streams for transmission over the network using the transmission module. Furthermore, the feedback module appends network condition details to other components through acknowledgement packets. At the receiver side, the process accounts for packet decoding and transmission.

6.4.2. Simulation results

We validated the performance of QUIC and other transport protocols in ultralarge-scale satellite networks and implemented the proposed architecture using the UltraStar platform. We incorporated the Kupiler K3 shell network model into our simulations. Using the FFmpeg library, we decoded real-time video streams into frames, assigning high priority to I-frames and low priority to B- and P-frames. Each frame was treated as a block, and the deadline was set to 350 ms. The following reference algorithms were selected:

- **QUIC**: Regular QUIC protocol with default scheduler and packet loss process mechanism.
- **Deadline-first scheduler (DDL)** [57]: The DDL selects the non-expired frame with the closest deadline for transmission.
- **Deadline-first scheduler with FEC (DDLF):** The DDLF simultaneously combines the DDL scheduler and FEC mechanism and operates by selecting the nearest non-expired block for transmission and encoding it with a fixed redundancy ratio.

The performance of different protocols on USLSNs is shown in Fig. 14. Fig. 15 shows the all and high-priority frames completion ratio for real-time applications; the bandwidth is set to 8 Mbps. For all frames, QRST achieved the highest completion ratio under all loss rates and demonstrated robustness, even with a higher loss rate. DDLF (8,6) and DDLF (8,7) exhibit suboptimal performances.

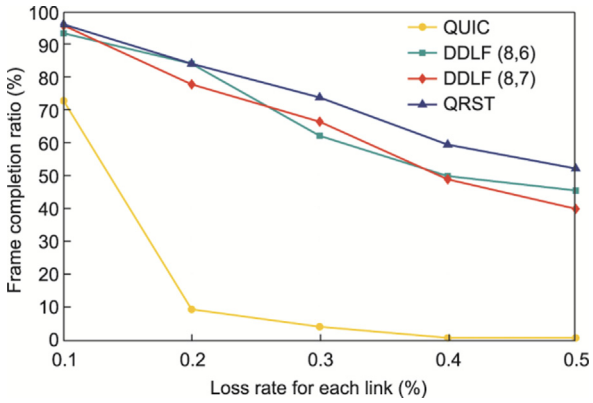

Fig. 14. Completion ratio of all frames.

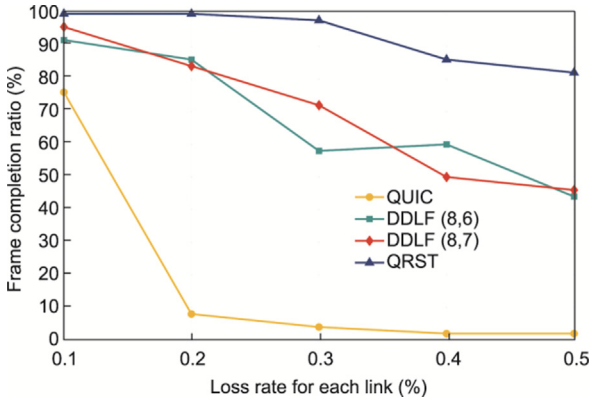

Fig. 15. Completion ratio of high-priority frames.

The FEC encoding can mitigate network packet loss and enhance delay performance to some extent; however, fixed redundancy cannot adapt to diverse loss scenarios, which results in a degraded completion ratio performance. The QUIC protocol obtains the poorest performance, which can be attributed to its inability to solve packet losses. Further, QRST exhibits a more pronounced advantage in terms of high-priority frame completion. The QRST can achieve a completion ratio of high-priority frames of more than 80% and 100% under higher and lower loss rates, respectively, which is significantly higher than that of all frames. For the other schemes, the high-priority frame completion ratio is similar for all frames because QRST considers the importance of critical frames sufficiently, whereas the other schemes treat all frames equally. Nearly identical completion ratios are displayed.

For the transport layer protocol simulation, UltraStar supports end-to-end transmission scenarios in ultralarge-scale satellite constellation networks. This supports the validation of the transport layer enhancement features in highly dynamic scenarios, which includes congestion control algorithm optimization, error recovery mechanism improvement, and QoS assurance schemes. In addition, UltraStar enables performance comparison and evaluation of different transport layer protocols, which includes key metrics such as bandwidth, packet loss rate, latency, and fairness.

6.5. Simulation efficiency of UltraStar

Experiments were conducted using two representative constellation scales to assess the simulation efficiency of UltraStar in large-scale network scenarios. GSs were deployed in the top 100

Table 1
Execution time under different constellations.

Constellation scale	Cores	Traffic load (Mbps)	Execution time (h)	Speedup	Factor
LEO-1000	1	998	7.04	Baseline	254
	8	998	1.07	6.57	39
	16	998	0.63	11.14	23
	24	998	0.46	15.22	17
LEO-2000	1	993	27.98	Baseline	1007
	8	993	3.96	7.06	143
	16	993	2.24	12.47	81
	24	993	1.57	17.83	57

cities globally based on population distribution. The minimum ground elevation angle was configured to ensure that dozens of GSL handovers occurred within the simulation duration. The simulation was configured with a 100 s duration and a 1 s topology update interval. Frequency-division multiple access was employed as the satellite-to-ground media access control protocol, while point-to-point was used for inter-satellite communication. The IPv4 and open shortest path first (OSPF) routing protocols were enabled, and traffic was transmitted via UDP and TCP with a traffic load of nearly 1 gigabits per second (Gbps). All experiments were performed using an Advanced Micro Devices, Inc. (AMD) server with 64 cores at 2.09 GHz.

The execution time represents the wall-clock time required by the CPU to complete a simulation. Throughput measures the total volume of traffic received by the entire constellation over the course of a simulation. Speedup represents a performance improvement over serial simulations, with higher values indicating the greater advantages of parallel architecture. The scale factor is calculated as the ratio of the execution time to the configured simulation duration.

Table 1 shows that increasing the number of computing cores leads to greater improvements in simulation speed; however, the synchronization overhead also increases with an increase in the number of cores, which indicates that adding more cores can result in diminishing acceleration gains. Considering the effect of interprocess communication overhead on the time efficiency and load balancing in task allocation is necessary when selecting the number of cores for parallel simulation. For each simulation scenario, there exists an optimal range of core numbers within which the simulation efficiency improves with an increase in the number of cores. However, beyond this range, the speedup starts to decline. The results in Table 1 indicate that the UltraStar simulation platform demonstrates support for system-level simulations involving up to 2000 satellites, maintaining a scale factor of no more than 60 under a traffic load of nearly 1 Gbps with parallel computing utilizing 24 cores. The advantages of parallel computing became more significant with an increase in the constellation scale. For LEO-2000, utilizing 24 cores achieved a speedup of up to 17.8 times compared to that with serial simulation. The parallel simulation approach significantly improved efficiency in complex scenarios to a level of performance unattainable with a serial simulation approach.

7. Conclusions

In this paper, we introduced a newly developed high-fidelity and high-efficiency computer simulator tailored to support the advancement of 6G integrated space–ground networks with LEO mega-constellation satellites. We designed a systematic, scalable, and comprehensive simulation architecture that enabled the precise modeling of network configurations and efficient simulation of network operation and management capabilities. We implemented an environment update algorithm that integrated real ephemeris data for precise satellite orbit prediction, sun outages, and link handover modeling to accurately capture the topology characteristics. Further, we developed an NS-3-based network model that flexibly accommodates the extensions of networking protocols to ensure realistic simulations of software and hardware configurations. We employed an MPI interface-based parallel and distributed approach that leveraged multiple cores or machines to achieve high simulation efficiency in the context of large and complex network scenarios. The simulator demonstrated its ability to effectively facilitate the design and performance analysis of algorithms and protocols tailored for network operation and deployment in future 6G integrated space–ground networks through extensive simulation experiments. In future work, we plan to apply this simulator to real network scenarios, enabling customized task simulation and analysis to meet specific requirements.

CRediT authorship contribution statement

Haibo Zhou: Software. **Xiaoyu Liu:** Software. **Xin Zhang:** Resources. **Xiaohan Qin:** Resources. **Mengyang Zhang:** Software. **Yuze Liu:** Resources. **Weihua Zhuang:** Resources. **Xuemin (Sherman) Shen:** Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the Major Program of the National Natural Science Foundation of China (62495021 and 62495020) and in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] Cheng N, Xu W, Shi W, Zhou Y, Lu N, Zhou H, et al. Air–ground integrated mobile edge networks: architecture, challenges, and opportunities. IEEE Commun Mag 2018;56(8):26–32.
[2] Liu X, Ma T, Tang Z, Qin X, Zhou H, Shen X. UltraStar: a lightweight simulator of ultra-dense LEO satellite constellation networking for 6G. IEEE/CAA J Automatica Sinica 2023;10(3):632–45.
[3] Wu H, Yang X, Bu Z. Task offloading with service migration for satellite edge computing: a deep reinforcement learning approach. IEEE Access 2024;12:25844–56.
[4] Chen JH, Kuo WC, Liao W. SpaceEdge: optimizing service latency and sustainability for space-centric task offloading in LEO satellite networks. IEEE Trans Wirel Commun 2024;23(10):15435–46.
[5] Zhao L, Liu Y, Hawbani A, Lin N, Zhao W, Yu K. QoS-aware multihop task offloading in satellite–terrestrial edge networks. IEEE Internet Things 2024;11(19):31453–66.

[6] Wei T, Feng W, Chen Y, Wang CX, Ge N, Lu J. Hybrid satellite–terrestrial communication networks for the maritime Internet of Things: key technologies, opportunities, and challenges. IEEE Internet Things 2021;8(11):8910–34.
[7] Ji Z, Wu S, Jiang C. Cooperative multi-agent deep reinforcement learning for computation offloading in digital twin satellite edge networks. IEEE J Sel Areas Commun 2023;41(11):3414–29.
[8] Tang Y, Zhou N, Yu Q, Wu D, Hou C, Tao G, et al. Intelligent fabric enabled 6G semantic communication system for in-cabin scenarios. IEEE Trans Intell Transp 2023;24(1):1153–62.
[9] Kodheli O, Lagunas E, Maturo N, Sharma SK, Shankar B, Montoya JFM, et al. Satellite communications in the new space era: a survey and future challenges. IEEE Commun Surv Tutor 2021;23(1):70–109.
[10] Nguyen DC, Ding M, Pathirana PN, Seneviratne A, Li J, Niyato D, et al. 6G Internet of Things: a comprehensive survey. IEEE Internet Things 2022;9(1):359–83.
[11] Lee Y, Choi JP. Connectivity analysis of mega-constellation satellite networks with optical intersatellite links. IEEE Trans Aerosp Electron Syst 2021;57(6):4213–26.
[12] Ma T, Zhou H, Qian B, Cheng N, Shen X, Chen X, et al. UAV–LEO integrated backbone: a ubiquitous data collection approach for B5G Internet of remote things networks. IEEE J Sel Areas Comm 2021;39(11):3491–505.
[13] Liu J, Shi Y, Fadlullah ZMD, Kato N. Space–ground integrated network: a survey. IEEE Commun Surv Tutor 2018;20(4):2714–41.
[14] Li Y, Li H, Liu L, Liu W, Liu J, Wu J, et al. "Internet in space" for terrestrial users via cyber–physical convergence. In: Proceedings of the 20th ACM Workshop on Hot Topics in Networks; 2021 Nov 10–12; New York City, NY, USA. New York City: Association for Computing Machinery (ACM); 2021. p. 163–70.
[15] Lai Z, Li H, Wang Y, Wu Q, Deng Y, Liu J, et al. Achieving resilient and performance-guaranteed routing in space–terrestrial integrated networks. In: Proceedings of IEEE Conference on Computer Communications; 2023 May 17–20; New York City, NY, USA. New York City: IEEE; 2023. p. 1–10.
[16] Cheng N, Quan W, Shi W, Wu H, Ye Q, Zhou H, et al. A comprehensive simulation platform for space–ground integrated network. IEEE Wirel Commun 2020;27(1):178–85.
[17] Kassing S, Bhattacherjee D, Águas AB, Saethre JE, Singla A. Exploring the "Internet from space" with Hypatia. In: Proceedings of the ACM Internet Measurement Conference; 2020 Oct 27–29; online. New York City: Association for Computing Machinery; 2020. p. 214–29.
[18] Lai Z, Li H, Li J. StarPerf: characterizing network performance for emerging mega-constellations. In: Proceedings of IEEE 28th International Conference on Network Protocols; 2020 Oct 13–16; Madrid, Spain. New York City: IEEE; 2020. p. 1–11.
[19] Lai Z, Wu Q, Li H, Lv M, Wu J. OrbitCast: exploiting mega-constellations for low-latency earth observation. In: Proceedings of IEEE 29th International Conference on Network Protocols; 2021 Nov 1–10; Dallas, TX, USA. New York City: IEEE; 2021. p. 1–12.
[20] Shen L, Zhou H, Wang X, Li W, Fan Y, Wu S, et al. Network evaluation and planning for large-scale low Earth orbit satellite constellation networks. In: Proceedings of the 2020 IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking; 2020 Dec 17–19; Exeter, UK. New York City: IEEE; 2020. p. 1235–40.
[21] Valentine A, Parisis G. Developing and experimenting with LEO satellite constellations in OMNeT++. In: Marek M, Nardini G, Veselý V, editors. Proceedings of the 8th OMNeT++ Community Summit Conference; 2021 Sep 8–10; online. Hamburg: Hamburg University of Technology; 2021.
[22] Zhou Y, Zhang R, Liu J, Huang T, Tang Q, Yu F. A hierarchical digital twin network for satellite communication networks. IEEE Commun Mag 2023;61(11):104–10.
[23] Sormunen L, Puttonen J, Kurjenniemi J. System level modelling of DVB-S2X in high throughput satellite system. In: Proceedings of the 36th International Communications Satellite Systems Conference; 2018 Oct 15–18; Niagara Falls, ON, Canada. New York City: IEEE; 2018. p. 1–4.
[24] Mao B, Zhou X, Liu J, Kato N. On an intelligent hierarchical routing strategy for ultra-dense free space optical low Earth orbit satellite networks. IEEE J Sel Areas Commun 2024;42(5):1219–30.
[25] Feng X, Sun Y, Peng M. Distributed satellite–terrestrial cooperative routing strategy based on minimum hop-count analysis in mega LEO satellite constellation. IEEE Trans Mobile Comput 2024;23(11):10678–93.
[26] VVPSTK [Internet]. Beijing: Beijing Aerospace Huihai System Simulation Technology Co., Ltd.; [cited 2025 Aug 7]. Available from: https://www.vvpstk.com/product.html.
[27] Satellite Tool Kit (STK) [Internet]. Exton: Ansys Government Initiatives (AGI); [cited 2025 Aug 7]. Available from: https://www.agi.com/products.
[28] Chen Q, Yang L, Zhao Y, Wang Y, Zhou H, Chen X. Shortest path in LEO satellite constellation networks: an explicit analytic approach. IEEE J Sel Areas Commun 2024;42(5):1175–87.
[29] Kelso TS. NORAD two-line element set format [Internet]. Maui: CelesTrak; [cited 2025 Aug 7]. Available from: https://celestrak.org/NORAD/documentation/tle-fmt.php.
[30] The SPICE toolkit [Internet]. Washington, DC: Navigation and Ancillary Information Facility (NAIF); 2022 Jan 3 [cited 2025 Aug 7]. Available from: https://naif.jpl.nasa.gov/naif/toolkit.html.
[31] Ekici E, Akyildiz IF, Bender MD. A distributed routing algorithm for datagram traffic in LEO satellite networks. IEEE/ACM Trans Netw 2001;9(2):137–47.
[32] Yong J, Wen F, Hu Z, Fan F, Qiu K. High-dynamic transmission modeling for laser inter-satellite links (LISLs). In: Proceedings of the 2022 Asia Communications and Photonics Conference; 2022 Nov 5–8; Shenzhen, China. New York City: IEEE; 2022. p. 590–4.
[33] Ma S, Chou Y, Zhao H, Chen L, Ma X, Liu J. Network characteristics of LEO satellite constellations: a Starlink-based measurement from end users. In: Proceedings of IEEE Conference on Computer Communications; 2023 May 17–20; New York City, NY, USA. New York City: IEEE; 2023. p. 1–10.
[34] Grewal MS, Andrews AP, Bartone CG. Fundamentals of satellite navigation systems. In: Grewal MS, Andrews AP, Bartone CG, editors. Global navigation satellite systems, inertial navigation, and integration. 4th ed. Hoboken: Wiley; 2020. p. 21–41.
[35] Zhang X, Wang Y, Qin X, Zhang Z, Zhou H, Shen X. Link-level performance analysis of DVB standards in ultra-dense LEO satellite–terrestrial networks. In: Proceedings of IEEE 99th Vehicular Technology Conference; 2024 Jun 24–27; Singapore. New York City: IEEE; 2024. p. 1–5.
[36] Sandri M, Pagin M, Giordani M, Zorzi M. Implementation of a channel model for non-terrestrial networks in NS-3. In: Henderson T, Imputato P, Liu Y, Gamess E, editors. Proceedings of the 2023 Workshop on NS-3; 2023 Jun 28–29, New York City, NY, USA. New York City: Association for Computing Machinery (ACM); 2023. p. 28–34.
[37] Polishuk A, Arnon S. Optimization of a laser satellite communication system with an optical preamplifier. J Opt Soc Am A Opt Image Sci Vis 2004;21(7):1307–15.
[38] TR 38.811: Study on new radio (NR) to support non-terrestrial networks (NTN) (version 15.4.0). 3GPP standard. Antibes: 3rd Generation Partnership Project (3GPP); 2020.
[39] TR 38.821: Solutions for NR to support non-terrestrial networks (NTN) (version 16.2.0). 3GPP standard. Antibes: 3rd Generation Partnership Project (3GPP); 2023.
[40] Singh K, Chebaane S, Ben Khalifa S, Benabdallah F, Ren X, Khemakhem H, et al. Investigations on mode-division multiplexed free-space optical transmission for inter-satellite communication link. Wirel Netw 2022;28(3):1003–16.
[41] Xia Y, Zhang X, Qin X, Zhang Z, Zhang Y, Ma T, et al. Leveraging deep-learning for adaptive coding and modulation for LEO satellite–terrestrial networks. In: Proceedings of International Conference on Communications in China; 2024 Aug 7–9; Hangzhou, China. New York City: IEEE; 2024. p. 774–9.
[42] Pelkey J, Riley GF. Distributed simulation with MPI in NS-3. In: Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques; 2011 Mar 21–25; Barcelona, Spain. Brussels: Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering (ICST); 2021. p. 410–4.
[43] Kunz G, Landsiedel O, Götz S, Wehrle K, Gross J, Naghibi F. Expanding the event horizon in parallelized network simulations. In: Proceedings of IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems; 2010 Aug 17–19; Miami Beach, FL, USA. New York City: IEEE; 2010. p. 172–81.
[44] Bilel BR, Navid N, Bouksiaa MSM. Hybrid CPU–GPU distributed framework for large scale mobile networks simulation. In: Proceedings of the IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications; 2012 Oct 25–27; Dublin, Ireland. New York City: IEEE; 2012. p. 44–53.
[45] Nikolaev S, Banks E, Barnes PD Jr, Jefferson DR, Smith S. Pushing the envelope in distributed NS-3 simulations: one billion nodes. In: Proceedings of the 2015 Workshop on NS-3; 2015 May 13–14; New York City, NY, USA. New York City: Association for Computing Machinery (ACM); 2015. p. 67–74.
[46] Renard K, Peri C, Clarke J. A performance and scalability evaluation of the NS-3 distributed scheduler. In: Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques; 2012 Mar 19–23; Desenzano del Garda, Italy. Brussels: Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering (ICST); 2012. p. 378–82.
[47] Fujimoto RM, Perumalla K, Park A, Wu H, Ammar MH, Riley GF. Large-scale network simulation: how big? How fast? In: Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems; 2003 Oct 12–15; Orlando, FL, USA. New York City: IEEE; 2003. p. 116–23.
[48] Ma T, Qian B, Qin X, Liu X, Zhou H, Zhao L. Satellite–terrestrial integrated 6G: an ultra-dense LEO networking management architecture. IEEE Wirel Commun 2024;31(1):62–9.
[49] Wang Y, Ma T, Qin X, Zhang X, Zhang Z, Zhou H. Satellite–aircraft handover in ultra-dense LEO satellite networks. IEEE Trans Veh Technol 2024;73:1–15.
[50] Chen Y, Sun J, Lin Y, Gui G, Sari H. Hybrid *N*-inception-LSTM-based aircraft coordinate prediction method for secure air traffic. IEEE Trans Intell Transp Syst 2022;23(3):2773–83.
[51] Da Silva RB, Souza ME. A survey on approaches to reduce BGP interdomain routing convergence delay on the Internet. IEEE Commun Surv Tutor 2017;19(4):2949–84.
[52] Buob MO, Lambert A, Uhlig S. iBGP2: a scalable iBGP redistribution mechanism leading to optimal routing. In: Proceedings of the 35th Annual IEEE International Conference on Computer Communications; 2016 Apr 10–14; San Francisco, CA, USA. New York City: IEEE; 2016. p. 1–9.
[53] Shi H, Cui Y, Qian F, Hu Y. DTP: deadline-aware transport protocol. In: Proceedings of the 3rd Asia-Pacific Workshop on Networking; 2019 Aug 17–

18; Beijing, China. New York City: Association for Computing Machinery (ACM); 2019. p. 1–7.
[54] Liu Y, Su W, Tan L. Tetris: near-optimal scheduling for multipath deadline-aware transport protocol. In: Proceedings of the 2021 International Conference on Networking and Network Applications; 2021 Oct 29–Nov 1; Lijiang, China. New York City: IEEE; 2021. p. 34–40.
[55] Tang F, Wen C, Luo L, Zhao M, Kato N. Blockchain-based trusted traffic offloading in space–ground integrated networks (SAGIN): a federated reinforcement learning approach. IEEE J Sel Areas Commun 2022;40 (12):3501–16.
[56] Habachi O, Shiang HP, Van Der Schaar M, Hayel Y. Online learning based congestion control for adaptive multimedia transmission. IEEE Trans Signal Process 2013;61(6):1460–9.
[57] Michel F, Cohen A, Malak D, De Coninck Q, Médard M, Bonaventure O. FlEC: enhancing QUIC with application-tailored reliability mechanisms. IEEE/ACM Trans Netw 2023;31(2):606–19.