

# Space Networking Kit: A Novel Simulation Platform for Emerging LEO Mega-constellations

Xiangtong Wang\*, Xiaodong Han†, Menglong Yang\*‡, Songchen Han\*, Wei Li\*‡

\*School of Aeronautics and Astronautics, Sichuan University, Chengdu, China

†China Academy of Space Technology, Beijing, China

‡Robotic Satellite Key Laboratory of Sichuan Province, Chengdu, China

Email: {li.wei,hansongchen}@scu.edu.cn

**Abstract**—This paper presents SNK, a novel simulation platform designed to evaluate the network performance of constellation systems for global Internet services. SNK offers real-time communication visualization and supports the simulation of routing between edge node of network. The platform enables the evaluation of routing and network performance metrics such as latency, stretch, network capacity, and throughput under different network structures and density. The effectiveness of SNK is demonstrated through various simulation cases, including the routing between fixed edge stations or mobile edge stations and analysis of space network structures.

**Index Terms**—Space network, global internet service, simulation platform

## I. INTRODUCTION

The development of satellite networks (SN) within the “NewSpace” paradigm has gained substantial traction in recent years and holds significant promise in delivering global internet connectivity characterized by low latency and high throughput. Commercial enterprises, such as SpaceX[1], Amazon[2], OneWeb[3], etc., have joined this competitive “NewSpace” race. While above exciting prospects depict a promising picture of these future integrated satellite-terrestrial networks, the community still has very limited understanding of the structural characteristic and the available network performance of modern mega-constellations. One of the major reasons for this is the absence of suitable simulation platform.

Networking simulation in LEO mega-constellations differs from well-known ones in terrestrial networks due to their sizeable spatiotemporal scale, high mobility, and orbital movements. These distinctive properties may pose significant challenges to traditional network simulators.

Until now, most simulators in community are combined with space simulators [4], [5], [6] and network simulators [7], [8], hence to obtain better functionality of visualization, celestial motion and network simulation. SNS3[9] is an discrete simulator that combines NS3 with satellite communications link model, but it primarily supports simple functionality among GEOs without networking. Handley [10] developed the platform through Unity 3D, which provides impressive visualization effects that intuitively show the path selection and delay of the end-to-end routing process, due to its emphasis on space simulation, it lacks the corresponding formation of the network simulation, and cannot satisfy the packet-level network simulation. Benjamin[11] presents a network simulator

for SNs with robust visualization features, leveraging OpenGL for enhanced graphics support. Celestial[12] first introduced the virtualization technology into SN simulation/emulation, i.e., attaching each satellite with a micro virtual machine to emulate the real networking environment. Hypatia[13] is developed based on Cesium[14] in space simulation and based on NS3 for networking simulation, thus supporting packet-level simulation and satisfying visualization. Although it is not flexible in building scenarios, has limited visualization capabilities, and is no longer receiving updates, it is a valuable practice to implement complex visualization features through Web. Overall, most simulation tools must possess several important qualities: 1) effective visualization of space motion and communication processes; 2) flexibility to configure complex networking scenarios; and 3) scalability to support future multilateral requirements.

In this paper, we introduce SNK, a performance simulation platform designed to assist constellation manufacturers and network operators in estimating and comprehending the attainable performance across various constellation options.

To demonstrate the effectiveness of SNK on routing algorithm and SN optimizing, we leverage SNK to evaluate and compare the performance of Dijkstra routing algorithm between cities and obtain insights on optimizing the constellation design to improve edge-to-edge network performance. The results of our evaluation indicate that the careful selection of a routing algorithm and network structure can significantly impact the communication process in terms of latency, path stretch, throughput and network capacity.

Overall, this paper makes two key contributions:

- Introduction of SNK, a simulation platform that facilitates the profiling and understanding the network performance of mega-constellations under a diversity of network policies and structure. (§.II).
- Utilization of SNK to reveal the different results under diverse network policies and structure, while also highlighting insights on optimizing constellation designs to improve network performance. (§.III)

We are releasing our implementation of SNK available as open source to help future researchers validate their own applications and platforms. Our hope is that this will make the field of SN more accessible and provide a starting point for systems research in the community.

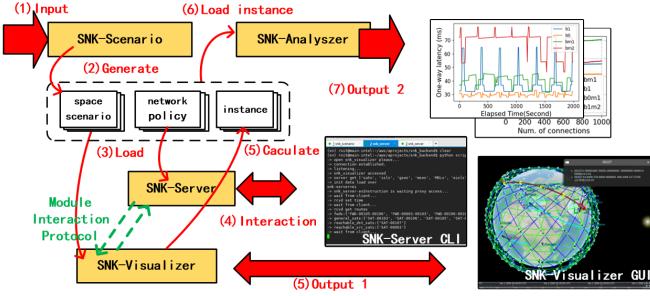


Fig. 1. The overview and workflow of SNK.

## II. THE SNK PLATFORM

### A. System overview and workflow

SNK is jointly developed in multiple languages and consists of four key modules: SNK-Scenario, SNK-Visualizer, SNK-Server and SNK-Analyzer.

The SNK-Scenario module serves the purpose of generating scenario data for SN, including satellites, links, ground stations, mobile stations, and other relevant elements. Developed as a web application rooted in Cesium, SNK-Visualizer goes beyond simple visualization of the constructed scenes. It dynamically showcases the networking processes, supports real-time adjustments of perspectives and layers, and facilitates interaction with the SNK-Server. The SNK-Server, a Python-based network simulator, establishes synchronous data transmission with SNK-Visualizer through its API subsystem. SNK-Analyzer functions as an analytical tool capable of visually representing statistical outcomes such end-to-end latency and path stretch rates.

Fig.1 shows the workflow of SNK. The SNK takes a configuration file as input to generate scenario files, which describe the composition of constellation and networking. During runtime, the user interacts with SNK via the command line to load the created scenario file and initiate the simulation computation, and the simulation process can be observed in real-time by enabling the "Watch" mode. At the end of the simulation, the generated instance file is loaded, and performance metrics like network latencies and throughput are measured. These metrics are then used to compute and quantify the network performance.

### B. SNK-Scenario

SNK-Scenario consists of a suite of Python-based script programs designed to generate essential scenario entities required for simulation. These entities encompass a range of components including satellites (SATs), ground stations (GSes), and mobile stations (MSes), which may include operational satellites or airborne objects. It also encompasses links such as Inter-satellite links (ISLs), satellite-to-ground links (GSL), and mobile-station-satellite links (MSLs). Each of these entities is created via the execution of individual scripts.

SNK-Scenario offers remarkable flexibility, achieved through the utilization of distinct config.yaml files. This approach streamlines the creation of complex multi-layer SNs, such as Walker configurations with varying orbit arguments. A template for the config.yaml format is available in the supplementary documentation. Additionally, we provide a simplified script that simplifies scenario generation with a single bash command:

```
bash run.sh config.yaml
```

The construction of certain entities requires reference to other entities. Therefore, the recommended order of construction is as follows:

```
SAT.py --> ISL.py--> eISL.py  
MSes.py --> MSL.py  
GSes.py --> GSL.py
```

Once the scenario is constructed, it is saved in the form of a \*.sce folder, with the general format being:

```
scenario_name.sce/  
| - config.yaml  
| - {layer_name}_sats.czml  
| - gses.czml  
| - mses.czml  
| - {layer_name}_isls.czml  
| - {layer_name}_eisl.czml  
| - {layer_name}_gsls.czml  
| - {layer_name}_msls.czml  
| - {layer_name}_tle.txt  
| - {layer_name}_eisl.json  
| - {layer_name}_isls.json  
| - {layer_name}_gsls.json  
| - {layer_name}_msls.json
```

Among these, the \*.czml format bears resemblance to a JSON file, primarily intended for interpretation by the web interface to create entities and is widely used in Cesium[14]. Conversely, the \*.json format functions as an abstract file, containing identifiers without essential details. Given the support for multi-layer networks, entities are tagged with numerical identifiers corresponding to network layers. In addition to furnishing satellite czml files, TLE data is also available to accommodate various requirements.

### C. SNK-Visualizer

SNK-Visualizer is a web application developed using JavaScript. It runs in web browser, provides visual rendering capabilities for entities generated by SNK-Scenario and enables seamless interaction with SNK-Server through the customized protocol that base on WebSockets[15].

The constructed scenario and entities are depicted in Fig.2, showcasing the outcomes of the config.yaml script we provide. Fig.2 (a) shows a satellite constellation, a single-layer walker  $\delta$  constellation with an inclination of  $53^\circ$ , consisting of 20 orbits, each containing 20 satellites. SNK-Visualizer allows the creation of various Inter-satellite Links (ISLs) (See Fig.2 (b)), encompassing intra-orbit ISLs (blue), inter-orbit ISLs (green), and encountering ISLs (black and white striped).

Furthermore, SNK-Visualizer supports showing routing paths between satellites or station nodes such as ground

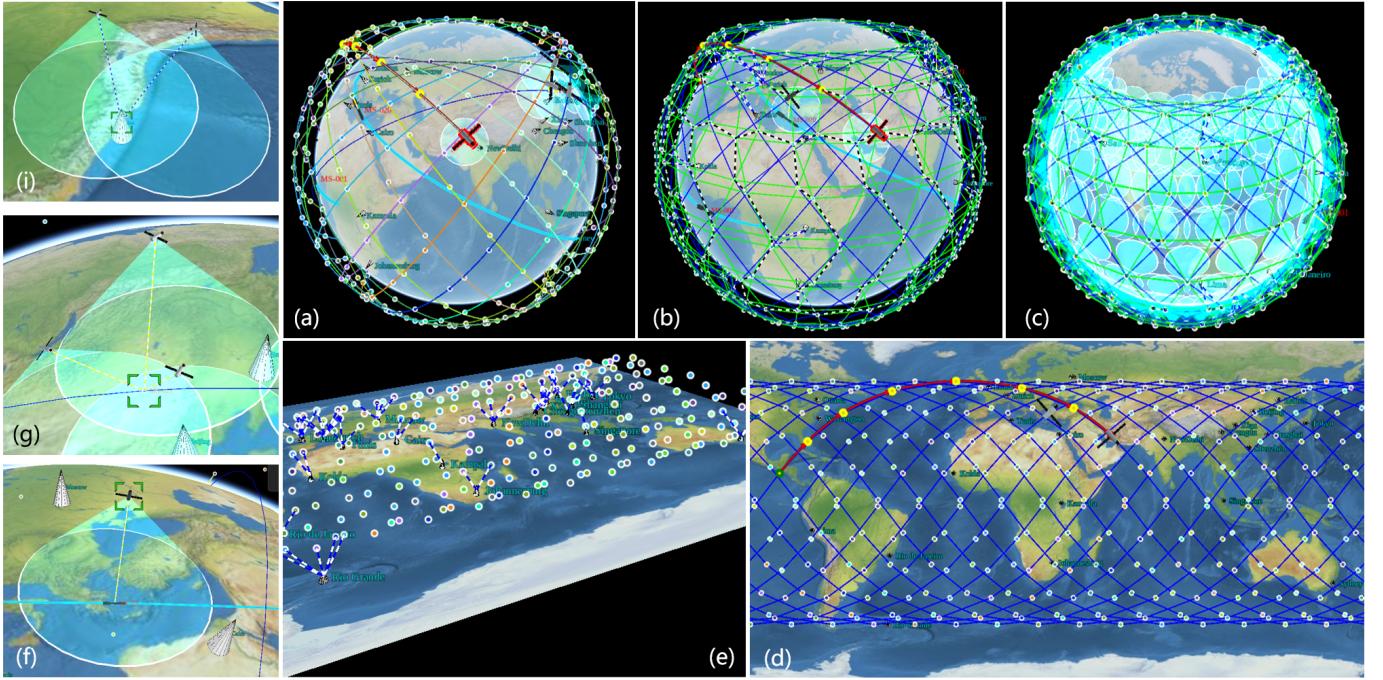


Fig. 2. The entities constructed by SNK-Scenario are displayed in SNK-Visualizer.

stations or mobile stations. In (Fig.2 (b)), the red lines represent routing paths between two satellites, established using the shortest-path routing algorithm, which takes the distance between satellites as edge weight. Satellites can be equipped with antennas with specific parameters, visualized by their coverage (See Fig.2 (c)). This visualization can be employed in other *SNK-Visualizer* modules for coverage analysis. Ground stations within a satellite's coverage angle can establish GSL (blue and white striped lines in (Fig.2 (i))). The 3rd-party satellites (Fig.2 (g)) or aircrafts (Fig.2 (f)) establish the MSL (yellow and white striped line) between the satellite and the mobile station. *SNK-Visualizer* offers two perspectives - 3D perspective (Fig.2 (a,b,c)) and 2D perspective (Fig.2 (e,d)) - facilitating a comprehensive analysis of scene entities.

#### D. SNK-Server

*SNK-Server* is developed by Python and is primarily designed with network simulation and interaction with *SNK-Visualizer*. It represents the most intricate module within the whole system, encompassing three major subsystems: the CLI subsystem, the API subsystem, and the procedure subsystem. The CLI subsystem functions as the command-line interface, predominantly responsible for user interaction and operational logic. Our current emphasis is on providing more insights into the API subsystem and procedure subsystem.

1) *API subsystem*: *SNK-Server* interacts with modules or external modules through API subsystem using the Module Interaction Protocol (MIP), a command and data transmission protocol developed based on WebSocket, facilitating the data sharing and simulation consistency.

2) *Procedure subsystem*: The Procedure subsystem plays a pivotal role in orchestrating the scheduling of each procedure within the *SNK-Server*. The execution of procedures serves two distinct purposes:

- Data acquisition for quantitative analysis. One primary objective of procedure execution is to acquire data periodically. Acquired data is a key factor used to evaluate the performance of a communication policy or network, and stands as the foremost mission of the Server.
- Observation for qualitative analysis. By visualizing the communication process in a dynamic network, it is easy for users to debug and improve the networking policy.

Within the such subsystem, we offer several procedures as following:

**The edge2edge procedure.** It performs the communication process between edge nodes (GSes and MSes) defined in network policy. Upon its activation, the networking policy and a timestamps are initialized and it runs as the following steps:

- (1) When *SNK-Server* starts up, it first gets the simulation timestamp from network policy file and sends the MIP command "set time" to *SNK-Visualizer* for consistency.
- (2) *SNK-Server* sends request commands to get the space scenario information in *SNK-Visualizer*, including network topology status, satellite positions, and other pertinent information.
- (3) Based on the acquired data, *SNK-Server* starts to calculate the routing path between the source and destination satellite sets. The source and destination satellite sets are the satellites covering the source edge node and destination edge node respectively.

- (4) Upon reaching the predefined end timestamp, the procedure is halted.
- (5) The procedure information is then preserved in \*.ins files as the instance data.

**The conTest procedure.** It performs the communication process between any two of satellites nodes, serving the purpose of evaluating the network policy or space scenario. Upon its activation, the routing algorithm and a time list are initialized and it runs as the following steps:

- (1) When *SNK-Server* starts up, it first gets the simulation timestamp from network policy file and sends the MIP command "set time" to *SNK-Visualizer* for consistency.
- (2) *SNK-Server* sends request commands to get the space scenario information in *SNK-Visualizer*, including network topology status, satellite positions, and other pertinent information.
- (3) Based on the acquired data, *SNK-Server* starts to calculate the routing path between any two satellites in the scenario under built-in algorithms.
- (4) if the number of paths exceeds a predefined threshold  $N$ , the procedure is interrupted, and the timestamp is updated.
- (5) Upon reaching the predefined end timestamp, the procedure is halted.
- (6) The procedure information is then preserved in \*.ins files as the instance data.

In addition to the first two procedures, we also provide: i) **asyncReplay procedure**, a procedure that can convert the event list and packet list generated by external simulation programs into animation data and visualize it asynchronously; ii) **replay procedure**, a procedure based on the obtained \*.ins file to replay the process of it and iii) **exInstruction** a procedure that supports joint debugging with external programs by API subsystem.

### III. SIMULATION CASES

In this section we demonstrate the effectiveness of SNK on characterizing and understanding emerging LEO mega-constellation systems.

#### A. The Local View

This Simulation involves routing analysis between fixed edge stations (representing ground stations or user terminals) and mobile edge stations (representing functional satellites or aircraft). The scenario is constructed as shown in Tab.I, and the stations are listed in Tab.II.

1) *Fixed edge station routing*: SN can effectively reduce network latency between cities and offers an alternative solution to submarine cables, therefore we will analyze the routing between fixed edge stations located in large cities with the SN. When the scenario is constructed via *SNK-Scenario* and the *edge2edge* process is executed, *SNK-Server* generates the \*.ins directory and retrieves routing information from *SNK-Analyzer*.

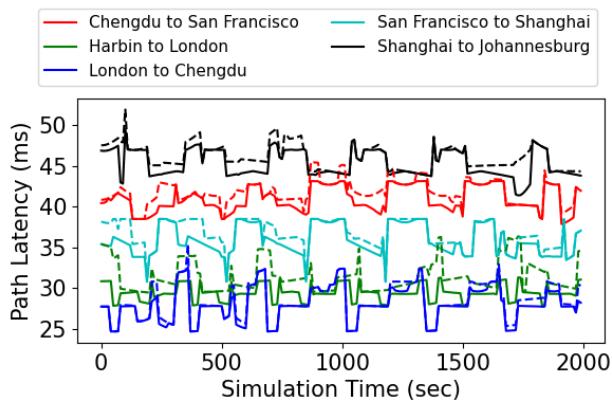
Fig.3 (a) displays the path latency under two different routing strategies: least-hop (dashed lines) and shortest-path

TABLE I  
SCENARIO INFORMATION.

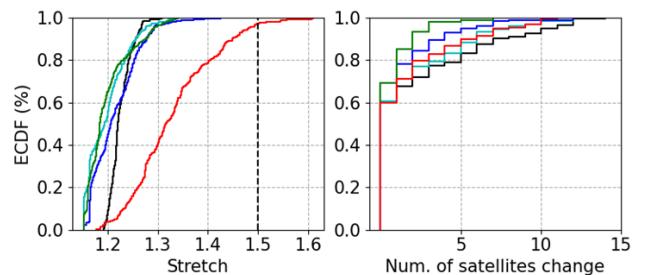
Symbols	description	value
$T/P/F/i$	parameters of constellation	400/20/0/53°
$h$	altitude of orbits	1000km
-	eISL building threshold[16]	2000km
-	ISL mode	*Grid
$\alpha$	steering angle of beams	45°
-	Simulation time	2000s
-	simulation time delta	10s

TABLE II  
EDGE STATIONS.

Edge Station Name	ID	Location \ Trajectory
Harbin	GS-0000	45N,127E
London	GS-0001	51N,0E
Chengdu	GS-0002	31N,102E
San Francisco	GS-0003	38N,122W
Shanghai	GS-0004	30N,122E
Johannesburg	GS-0005	26S,28E
MS-000	MS-000	ON,24W → ON,20E
MS-020	MS-020	50N,0E → 50S,180E
MS-021	MS-021	-



(a) Path latency over time between cities under different routing strategies.



(b) The path stretch and evolution between city pairs.

Fig. 3. Space network structures.

(solid lines) routing, for various city pairs over a 2000-seconds period. We have provided basic shortest path algorithms in SNK, and more complex routing algorithms can be constructed with simple development. In Fig.3 (b) on the left, the stretch distribution of the routing path is depicted. This metric measures the ratio of the routing path distance over the geographic

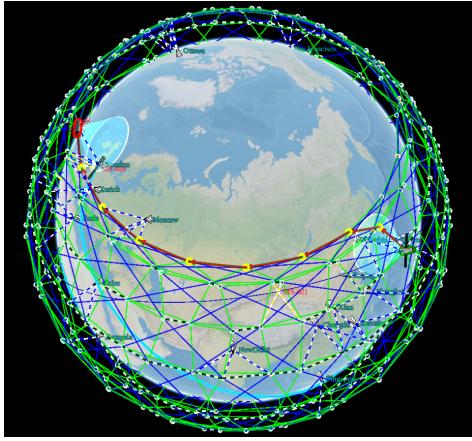
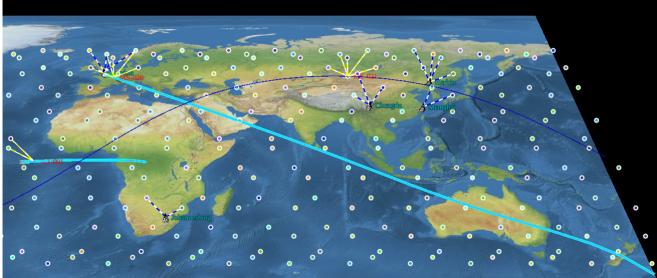
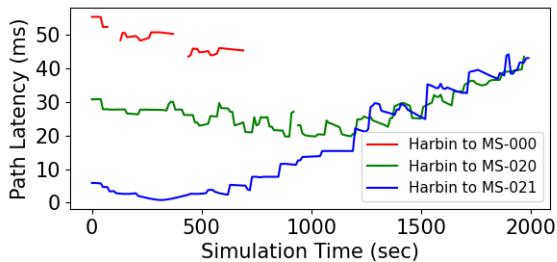


Fig. 4. The routing path can be visualized in *SNK-Visualizer* in real time when the enable "Watch" mode.



(a) Mobile stations include aircraft (MS-001, MS-020) or functional satellite (MS-021).



(b) Path latency of mobile edge stations.

Fig. 5. Space network structures.

distance, indicating the path detour. A smaller stretch indicates a path closer to the geographic arch, indicating a more efficient path. If  $stretch \leq 1.5$ , it indicates that the routing path in SNIs better than the optical fiber direct connection between two cities[13]. It is clear that the space network outperforms the terrestrial network in most routing strategies. On the right side of Fig.3 (b), the number of satellite changes in the path over time is shown, with smaller values indicating stable routing. In addition, in order to facilitate debugging during the development of routing algorithms, SNK has the capability to display the routing paths in real time in "Watch-enabled" mode, as is shown in Fig.4.

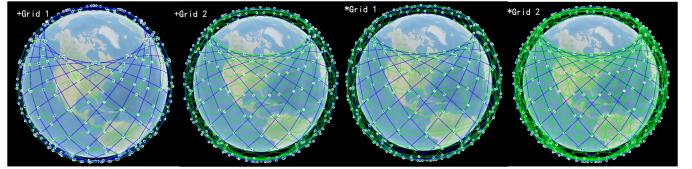


Fig. 6. Different configurations build vastly different network structures

2) *Mobile station routing:* SNK also supports to simulate the routing procedure between mobile edge nodes with a customized trajectory, such as aircraft or 3rd-party satellites such as Earth observation satellites. In this case, we constructed multiple mobile edge stations with different trajectories and evaluated the path latency between them and Harbin under different routing algorithms, as shown in Fig.5 (a). The thinner blue path is the orbit of 3rd-party satellites and thicker ones are the paths of the aircraft. The dashed yellow lines stands for mobile-station satellite links (MSLs). Fig.5 (b) shows the path latency of MSes. Note that there are gaps in the lines of MS-000 and MS-020 because there is not always an accessible satellite during the movement of these mobile stations.

#### B. The Global View

Existing constellations are primarily designed for coverage optimization or conflict avoidance, however, it is evident that designing systems with improved network performance in SNs is of paramount importance as it directly impacts the availability and feasibility of such systems. Therefore, SNK provides functions to quantify the network performance under different network structures and density, including global latency, stretch, network capacity and throughput. This capability supports the design of SNs and included in this simulation case. The scenario configuration is listed in Tab.III.

TABLE III  
SCENARIO CONFIGURATION OF GLOBAL ANALYSIS.

Description	value
Constellation scales	{ $10^2, 20^2, 30^2, 40^2$ }
Constellation structures	{+Grid 1, +Grid 2, *Grid 1, *Grid 2 }
Max num of connections	1000
Simulation time	1000s
Simulation time delta	100s

1) *Structure analysis:* The network structure analysis aims to evaluate the network performance of a constellation under a specific network structure, providing valuable insights for proposing robust network structures. Fig.6 shows the different structures under configuration files in SNK which are denoted as +Grid 1, +Grid 2, \*Grid 1 and \*Grid 2 respectively and represent 4 and 6 adjacent ISLs per satellites[17], [16]. In the above structure, the capacity of each link can be easily obtained based on the free space loss, and then the network throughput is calculated by the maximum flow algorithm. Fig.7 illustrates the throughput and latency distribution in different network configurations, providing insights into the impact of network structure on these performance metrics.

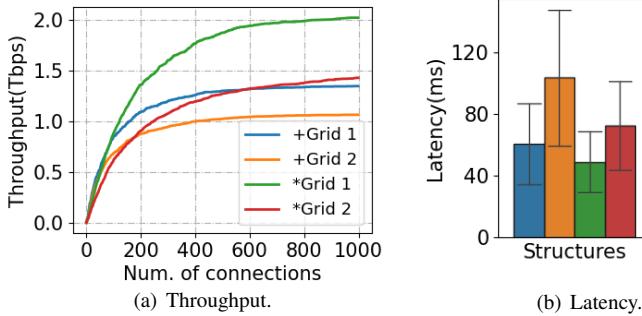


Fig. 7. Throughput and latency distribution in different network structures.

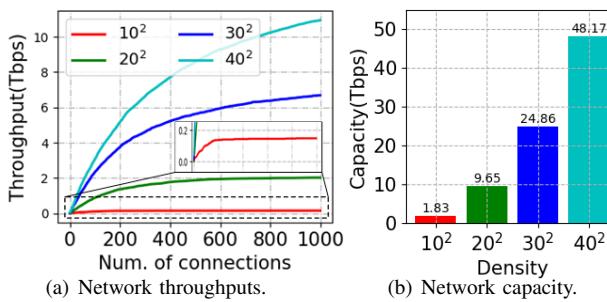


Fig. 8. Network capacity and throughput under shortest-path routing in structures with density 10<sup>2</sup>, 20<sup>2</sup>, 30<sup>2</sup>, 40<sup>2</sup>.

**2) Density Analysis:** Constellation networks of different density perform differently, with higher density networks exhibiting higher throughput and lower latency. Fig.8 shows the throughput and capacity of networks with different density under \*Grid 1 structure. It demonstrates that as the network size increases, the capacity and throughput are increasing significantly. However, it's important to note that since the routing algorithm relies on basic shortest-path without a load balancing mechanism, the capacity utilization remains around 25%.

#### IV. LIMITATIONS AND FUTURE WORKS

**Optimizing concurrent processing to enhance smoothness.** As the scale of space scenario and the complexity of network grow, the simulation process becomes less fluid. In the future, SNK will be further updated to achieve smoother simulations by optimizing the concurrency process.

**Introducing virtualization technology to SNK for high-fidelity emulating.** Unlike other recent works that focus on the network emulating[8] and space network emulating[12], SNK relies on model-based estimation for networking performance results. To improve fidelity, SNK's latest update combines virtualization techniques to emulate realistic networking environments by isolating node entities in network namespaces and connecting nodes using a novel link emulator[18], [19]. This enhancement aims to enable system-level emulation and

facilitate more accurate performance comparisons with real systems like Starlink or OneWeb.

#### V. CONCLUSION

This paper presents SNK, a novel simulation platform that enables constellation manufacturers and network operators to estimate the achievable network performance under a variety of constellation options. SNK allows user to build large-scale complex scenarios with configuration files by typing a single bash command, and evaluate or visualize the communication process. Leveraging SNK, we evaluate and compare the performance of shortest-path and least-hop routing algorithm and obtain the insights on network optimization for mega-constellations; This platform will be open-sourced to help future researchers validate their own applications and platforms.

#### REFERENCES

- [1] *For Approval of Orbital Deployment And Operating Authority for the SpaceX Gen2 NGSO Satellite System.* (2020), <https://fcc.report/IBFS/SAT-LOA-20200526-00055/2378669.pdf>, 2020.
- [2] *Kuiper NGSO constellation,* <https://fcc.report/IBFS/SAT-LOA-20190704-00057/2608982.pdf>, 2020.
- [3] *Modification to OneWeb U.S. Market Access Grant for the OneWeb Ku- and Ka-Band System.* (2020), <https://fcc.report/IBFS/SATMPL-20200526-00062/2379569.pdf>, 2020.
- [4] “System tool kit,” [ansys.com/products/missions/ansys-stk](https://ansys.com/products/missions/ansys-stk).
- [5] “General mission analysis tool,” <https://gmat.atlassian.net/wiki/spaces/GW/overview>.
- [6] “Savi: satellite constellation visualization,” <https://savi.sourceforge.io/>.
- [7] “ns3: A discrete-event network simulator for internet systems,” <https://www.nsnam.org/>, 2006.
- [8] B. Lantz, B. Heller, and N. McKeown, “A network in a laptop: rapid prototyping for software-defined networks,” in *ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010.
- [9] J. Puttonen, B. Herman, S. Rantanen, F. Laakso, and J. Kurjeniemi, “Satellite network simulator 3,” in *Workshop on Simulation for European Space Programmes (SESP)*, 2015.
- [10] M. Handley, “Delay is not an option: Low latency routing in space,” in *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, 2018, pp. 85–91.
- [11] B. Kempton and A. Riedl, “Network simulator for large low earth orbit satellite networks,” in *IEEE International Conference on Communications*. IEEE, 2021.
- [12] T. Pfandzelter and D. Bermbach, “Celestial: Virtual software system testbeds for the leo edge,” in *Proceedings of the 23rd ACM/IFIP International Middleware Conference*, 2022.
- [13] S. Kassing, D. Bhattacharjee, A. B. Agius, J. E. Saethre, and A. Singla, “Exploring the “internet from space” with hypatia,” in *Proceedings of the ACM Internet Measurement Conference*, 2020.
- [14] “Cesium: The platform for 3d geospatial.” <https://cesium.com/>, 2020.
- [15] “WebSocket,” <https://developer.mozilla.org/en-US/docs/Web/API/WebSocket>.
- [16] W. Xiangtong and L. Wei, “Enabling high-connectivity leo satellite networks via encountering inter-satellite links,” in *IEEE GLOBECOM*, 2023.
- [17] D. Bhattacharjee and A. Singla, “Network topology design at 27,000 km/hour,” in *CoNEXT*, 2019.
- [18] H. S., “Network emulation with netem,” <https://wiki.linuxfoundation.org/networking/netem>, 2005.
- [19] E. Petersen, J. López, N. Kushik, C. Poletti, and D. Zeghlache, “Dynamic link network emulation: A model-based design,” *arXiv preprint arXiv:2107.07217*, 2021.