

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—本文提出了 SNK——一个新颖的仿真平台，旨在评估用于全球互联网服务的星座系统的网络性能。SNK 提供实时通信可视化，并支持网络边缘节点之间路由的仿真。该平台能够在不同的网络结构和密度条件下评估路由和网络性能指标，如时延、路径拉伸比（stretch）、网络容量和吞吐量（throughput）。通过多个仿真实例验证了 SNK 的有效性，这些实例包括固定边缘站点或移动边缘站点之间的路由，以及对空间网络结构的分析。

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

I. 引言

在“新太空 (NewSpace)”范式下，卫星网络 (Satellite Networks, SN) 的发展近年来取得了显著进展，并在提供具有低时延和高吞吐量特征的全球互联网连接方面展现出巨大潜力。商业企业如 SpaceX[1]、Amazon[2]、OneWeb[3] 等，已纷纷加入这场激烈的“新太空”竞争。尽管上述激动人心的前景勾勒出未来天地一体化网络的美好图景，但当前学术界和工业界对现代巨型星座系统的结构特性以及可用网络性能仍了解有限。造成这一现状的主要原因之一，是缺乏合适的仿真平台。

与地面网络中广为人知的网络仿真不同，LEO (低地球轨道) 巨型星座中的网络仿真由于其巨大的时空尺度、高速移动性以及轨道运动特性而显著不同。这些独特的属性对传统网络仿真器构成了重大挑战。

迄今为止，社区中大多数仿真器均结合了空间仿真器 [4], [5], [6] 与网络仿真器 [7], [8]，以实现更好的可视化效果、天体运动模拟与网络仿真功能。SNS3[9] 是一个离散仿真器，结合了 NS3 与卫星通信链路模型，但主要支持 GEO (地球静止轨道) 之间的简单功能，缺乏网络层级的支持。Handley [10] 基于 Unity 3D 开发了一个平台，提供了令人印象深刻的可视化效果，能够直观展示端到端路

由过程中路径选择与时延。但由于其侧重于空间仿真，缺乏相应的网络仿真能力，无法满足数据包级 (packet-level) 的网络仿真需求。Benjamin[11] 提出了一个面向卫星网络的网络仿真器，具备强大的可视化功能，并通过 OpenGL 实现增强的图形支持。Celestial[12] 首次将虚拟化技术引入卫星网络仿真/仿真中，即为每颗卫星绑定一个微型虚拟机以仿真真实的网络环境。Hypatia[13] 则在空间仿真方面基于 Cesium[14]，在网络仿真方面基于 NS3 开发，从而支持数据包级仿真并具备可视化能力。尽管该工具在场景构建上不够灵活、可视化能力有限，且已停止更新，但它通过 Web 实现复杂可视化功能的实践仍具有重要价值。总体来看，多数仿真工具需要具备以下几个关键特性：1) 有效可视化空间运动与通信过程；2) 具备配置复杂网络场景的灵活性；3) 具备支持未来多方需求的可扩展性。

在本文中，我们介绍了 SNK——一个性能仿真平台，旨在帮助星座系统制造商与网络运营商评估和理解在不同星座选型下可达成的网络性能表现。

为了展示 SNK 在路由算法与卫星网络 (SN) 优化中的有效性，我们利用 SNK 对城市之间的 Dijkstra 路由算法性能进行评估与比较，并获得了有关优化星座设计以提升边缘到边缘 (edge-to-edge) 网络性能的洞见。评估结果表明，路由算法与网络结构的合理选择在时延、路径拉伸比 (path stretch)、吞吐量 (throughput) 和网络容量等通信过程指标上具有显著影响。

总体而言，本文的主要贡献有两点：

- 提出了 SNK，一个能够在多样化网络策略与结构下，支持巨型星座网络性能分析与理解的仿真平台。(§.III)
- 利用 SNK 揭示了在不同网络策略与结构下的性能差异，并提出了优化星座设计以改进网络性能的洞见。(§.IV)

我们将 SNK 的实现作为开源项目发布，以帮助未来研究人员验证其自身的应用与平台。我们希望这能使卫星网络 (SN) 研究领域更易于进入，并为学术界的系统研究提供一个起点。

II. System Model

A. 网络模型 (Network Model)

为应对卫星网络中的时间变化特性，我们定义一个有序时间集合为 $\mathcal{T} = \{t_1, t_2, \dots\}$ 。在不考虑遇遇链路 (eISLs) 的情况下，网络拓扑可视为在相邻时间戳之间保持不变。 $t_{i+1} - t_i$ 表示场景变化的最小时间粒度。于是，对于每个时间戳 $t \in \mathcal{T}$ ，网络拓扑可表示为一个无向图 $\mathcal{G}^t = (\mathcal{V}, \mathcal{E}^t)$ ，其中 \mathcal{V} 为网络节点集合 (卫星)， \mathcal{E}^t 为无向边集合 (可行星间链路，包括轨道内链路 iISL、轨道间链路 sISL 与遇遇链路 eISL)。

遇遇链路 (encountering inter-satellite link, eISL): 指两颗非固定邻接关系的卫星在相对运动过程中，临时满足视距 (LoS) 且距离不超过关联阈值 d_δ 时，按需建立的星间链路。其主要特征为：

- **临时性与时变性：**仅在满足 $d(s_i, s_j) \leq d_\delta$ 、指向与链路预算条件时存在，持续时间由相对轨道几何决定。
- **拓扑增益：**为拓扑上相距较远的卫星提供“捷径”，可降低最小跳数与路径拉伸比 (path stretch)，从而减少时延、提升吞吐。
- **区别于固定链路：**不同于轨道内固定链路 (iISL) 与轨道间固定链路 (sISL)，eISL 不预先固定邻居，按机会动态配对。
- **调度与开销：**需解决建立/拆除的时序控制、波束/指向资源分配与路由稳定性权衡，避免过度切换带来的信令与控制开销。

设 $\mathbf{r}^t(s_i)$ 表示在时刻 t 卫星 s_i 的笛卡尔坐标列向量，则两颗卫星 s_i, s_j 之间的欧几里得距离记为 $d^t(s_i, s_j)$ 。

假设地球为理想球体，两颗卫星之间的视距 (Line-of-Sight, LoS) 距离可表示为

$$LoS = 2\sqrt{h^2 + 2hR_E} \quad (1)$$

其中， R_E 表示地球半径。若 $d^t(s_i, s_j) \geq LoS$ ，则卫星 s_i 与 s_j 将被地球阻挡。

因此，我们定义可见矩阵 $\mathbf{V}^t = \{v_{ij}^t\}$ 。当 $d(s_i, s_j) \leq LoS$ 时， $v_{ij}^t = 1$ ；当 $d(s_i, s_j) \geq LoS$ 时， $v_{ij}^t = 0$ 。

进一步地，我们定义关联范围 d_δ ，其中 $0 \leq d_\delta \leq LoS$ ，表示当 $d(s_i, s_j) \leq d_\delta$ 时，两颗卫星之间可以建立遇遇链

路 (eISL)。其关联持续时间记为 $T(s_i, s_j, d_\delta)$ ，表示卫星 s_i, s_j 在关联范围内保持连接的时长。

设 $m_{ij}^t \in \{0, 1\}$ 为匹配指示变量 (matching indicator)，当在时刻 t ，卫星 s_i 与 s_j 之间建立了链路时， $m_{ij}^t = 1$ ；否则， $m_{ij}^t = 0$ 。进一步地，我们假设轨道内链路 (iISL) 与轨道间链路 (sISL) 为持续存在的固定链路，而遇遇链路 (eISL) 则为时变链路。匹配状态可表示为匹配矩阵 (Matching Matrix)：

$$\mathbf{M}_*^t = \mathbf{M}_+ + \mathbf{M}_- + \mathbf{M}^t \quad (2)$$

其中， $\mathbf{M}_-, \mathbf{M}_+$ 分别为常量矩阵，用以表示 iISL 与 sISL 的邻接状态； $\mathbf{M}^t = \{m_{ij}^t\}$ 表示在时刻 t ，两颗卫星之间是否建立 eISL。

因此，卫星网络的拓扑可由匹配矩阵 \mathbf{M}_*^t 表示，并等价于图 \mathcal{G} 的边集合 \mathcal{E}^t 。eISL 的构建问题可转化为一个匹配问题，其目标是在由固定链路模式定义的约束条件 \mathbf{M}_+ 和 \mathbf{M}_- 下，寻找一组合适的 \mathbf{M}^t 以最大化某一目标函数。

为在成本与连接性之间取得平衡，建议优先在拓扑距离较远的卫星之间建立遇遇链路 (eISLs)。我们定义在不考虑 eISL 的情况下，两颗卫星间的最小跳数作为其拓扑距离 $g(s_i, s_j)$ 。设 $\Delta p(s_i, s_j)$ 表示两颗卫星轨道面编号的最小差值，其定义为：

$$\Delta p(s_i, s_j) = \begin{cases} |p_i - p_j|, & \text{if } |p_i - p_j| \leq P/2 \\ \min\{p_i, p_j\} + P - \max\{p_i, p_j\}, & \text{otherwise} \end{cases} \quad (3)$$

其中， p_i 为卫星 s_i 所在轨道面的编号， P 为轨道面总数。类似地，相位差定义为：

$$\Delta f(s_i, s_j) = \begin{cases} |f_i - f_j|, & \text{if } |f_i - f_j| \leq N_P/2 \\ \min\{f_i, f_j\} + N_P - \max\{f_i, f_j\}, & \text{otherwise} \end{cases} \quad (4)$$

其中 N_P 为每个轨道面上的卫星数， f_i 为卫星 s_i 的相位编号。当已知卫星编号后，二者之间的拓扑距离可由下式计算：

$$g(s_i, s_j) = \Delta p + \Delta f \quad (5)$$

上述拓扑距离适用于使用 '+Grid' ISL 模式的星座结构 [?]. 若使用 '*Grid' ISL 模式，则拓扑距离定义为：

$$g(s_i, s_j) = \min\{\Delta p, \Delta f\} + |\Delta p - \Delta f| \quad (6)$$

B. 链路预算 (Link Budget)

在时刻 t , 卫星 i 发往卫星 j 的接收信噪比 (SNR) 表示为:

$$\text{SNR}^t(s_i, s_j) = \frac{\text{EIRP} \cdot G_r}{kT_s B_N L_f^t(s_i, s_j)} \quad (7)$$

其中, EIRP 表示等效全向辐射功率 (Equivalent Isotropically Radiated Power); G_r 为接收天线增益; k 为玻尔兹曼常数; T_s 为等效噪声温度 (单位 K); B_N 为信道带宽 (单位 Hz); L_f^t 为自由空间损耗 (Free Space Loss, FSL), 其计算公式为: $L_f^t(s_i, s_j) = \left(\frac{4\pi d_{ij}}{\lambda}\right)^2$, 其中 λ 为载波信号的波长。

基于上述公式, 卫星 s_i 与 s_j 之间链路的数据速率可表示为链路容量, 其表达式为:

$$R(s_i, s_j) = B \log_2(1 + \text{SNR}(s_i, s_j)) \quad (8)$$

C. 路径拉伸比与可达率 (Path stretch and reachable ratio)

路由算法的路径拉伸比 (path stretch) 定义为该算法返回的任意路径长度与对应节点间最短路径长度之间的最大比值。在卫星网络中, 我们将路径拉伸比定义为路径传播距离 L_{prop} 与对应两颗卫星间测地距离 L_{geo} 之比, 即:

$$\text{stretch} = \frac{L_{prop}}{L_{geo}} \quad (9)$$

该定义见于文献 [17]。由于光信号在光纤中的传播速度约为真空光速 c 的 $2/3$, 因此, 当 $\text{stretch} \leq 1.5$ 时, 说明卫星网络中的传播时延优于地面光纤连接。

不同于文献 [?], [?] 中通过最大化每条 ISL 的总容量进行链路分配的方法, 我们的主要目标是将具有较低复杂度的地理路由方法成功应用于卫星网络中。具体而言, 我们尝试通过引入 eISL 来解决地理路由中常见的“死胡同” (dead-end) 问题 [?]. 我们定义, 在时刻 t 、拓扑 \mathbf{M}_* 下, 路径 p_i^t 的可达性如下:

$$\text{Reach}(p_i^t, \mathbf{M}_*) = \begin{cases} 1, & \text{if } \min\{|\mathbf{M}_* - p_i^t|\} \geq 0 \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

其中, p_i^t 为在时刻 t 由路由算法生成的路径, 以邻接矩阵形式表示。

此外, 我们将匹配问题定义为最大化路径的可达率, 具体目标函数为:

$$\text{maximize} \quad \frac{1}{\|\mathcal{P}\| \|\mathcal{S}\|^2} \sum_{p_i \in \mathcal{P}} \sum_{\{s_i, s_j\} \subseteq \mathcal{S}} \text{Reach}(p_i^t, \mathbf{M}_*) \quad (11)$$

其中, $\|\mathcal{S}\|$ 表示卫星总数, $\|\mathcal{P}\|$ 表示路由算法生成的路径数量。考虑到 $\|\mathcal{S}\|^2$ 个卫星对的存在, 上述目标函数的计算复杂度为 $O(\|\mathcal{P}\| \|\mathcal{S}\|^2)$, 使其优化变得困难。为解决该问题, 我们提出了一种名为 DeC 的分布式算法, 用于在遇

D. 网络吞吐量 (Network Throughput)

为展示遇遇链路 (eISL) 对网络吞吐量的影响, 我们参考文献 [?], [?] 定义网络吞吐量, 暂不考虑地面侧的业务需求, 具体步骤如下:

- 1) 随机初始化若干端到端通信对, 表示为 (source, destination);
- 2) 利用路由算法获取每条端到端路径, 且约束任意两条路径之间不得共享链路 (即路径之间无公共边);
- 3) 计算每条路径上的瓶颈带宽, 作为其源卫星的吞吐量;
- 4) 计算各源卫星的平均吞吐量, 并将其累加作为整个系统的吞吐量。

III. SNK 平台

A. 系统概览与工作流程

SNK 是一个由多种语言共同开发的平台, 主要由四个关键模块组成: SNK-Scenario、SNK-Visualizer、SNK-Server 和 SNK-Analyzer。

SNK-Scenario 模块的功能是为 SN 生成场景数据, 包括卫星、链路、地面站、移动站以及其他相关元素。SNK-Visualizer 作为一个基于 Cesium 开发的 Web 应用, 不仅用于对构建的场景进行可视化展示, 更具备动态呈现网络过程、实时调整视角与图层, 并与 SNK-Server 进行交互的能力。SNK-Server 是一个基于 Python 的网络仿真器, 通过其 API 子系统与 SNK-Visualizer 建立同步数据传输。SNK-Analyzer 则作为一个分析工具, 可对端到端时延、路径拉伸率 (path stretch rate) 等统计结果进行可视化展示。

Fig.1 展示了 SNK 的工作流程。SNK 以配置文件作为输入, 用于生成描述星座组成与网络结构的场景文件。在运行时, 用户通过命令行与 SNK 交互, 加载已创建的场景文件并启动仿真计算过程; 启用“Watch”模式后, 可实时观察仿真过程。仿真结束后, 系统会加载生成的实例文件, 并测量如网络时延与吞吐量等性能指标, 进一步计算与量化整体网络性能。

B. SNK-Scenario

SNK-Scenario 由一组基于 Python 的脚本程序组成, 旨在生成仿真所需的基本场景实体。这些实体涵盖多个组成部分, 包括卫星 (SATs)、地面站 (GSes) 与移动站 (MSes),

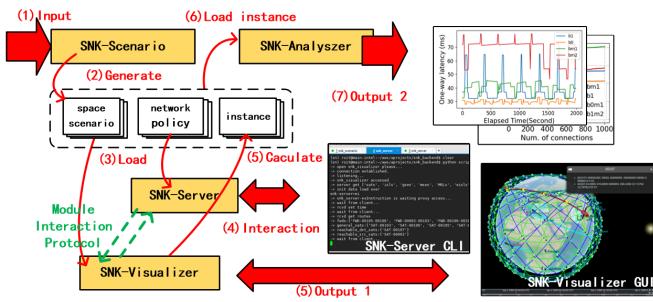


Fig. 1. SNK 的系统概览与工作流程。

其中 MSes 可包含在轨卫星或空中目标。它还包括各种链路类型，例如星间链路（Inter-satellite links, ISLs）、星地链路（Ground-satellite links, GSL）和移动站-卫星链路（Mobile-station-satellite links, MSL）。每一类实体均通过对 应脚本生成。

SNK-Scenario 具备高度灵活性，这得益于其采用了独立的 config.yaml 配置文件结构。该方式简化了多层卫星网络（multi-layer SNs）的构建流程，例如可配置不同轨道参数的 Walker 构型。关于 config.yaml 格式的模板，可参考附录文档。此外，我们还提供了一个简化脚本，允许用户通过一条 bash 命令生成完整场景：

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

部分实体的构建需依赖其他实体，因此推荐的构建顺序如下：

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

一旦场景构建完成，将以 *.sce 文件夹形式保存，通用结构如下：

```
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
```

其中，*.czml 格式类似于 JSON 文件，主要供 Web 界面解析使用，用于实体展示，在 Cesium[14] 中被广泛采用；

而 *.json 格式为抽象描述文件，仅包含标识符，不含具体细节。考虑到多层网络结构的支持，实体中均标记有与网络层级对应的数值 ID。除提供卫星的 czml 文件外，也提供 TLE 轨道数据文件，以满足不同的仿真需求。

C. SNK-Visualizer

SNK-Visualizer 是一个基于 JavaScript 开发的 Web 应用，运行于浏览器端，负责对由 SNK-Scenario 生成的实体进行可视化渲染，并通过基于 WebSockets[15] 的定制协议，与 SNK-Server 进行无缝交互。

由 config.yaml 脚本生成的场景与实体如 Fig.2 所示。Fig.2 (a) 展示了一个单层 walker δ 星座，倾角为 53°，共 20 轨道，每轨道含 20 颗卫星。SNK-Visualizer 支持多种星间链路（ISLs）的创建（见 Fig.2 (b)），包括轨道内链路（蓝色）、轨道间链路（绿色）与遭遇链路（黑白条纹）。

此外，SNK-Visualizer 支持显示卫星间或地面站/移动站等节点间的路由路径。在 Fig.2 (b) 中，红色线段表示使用最短路径算法（以卫星间距离为边权）建立的两颗卫星之间的路由路径。卫星可配置具备特定参数的天线，其覆盖范围可视化如 Fig.2 (c)，该功能亦可用于覆盖分析模块。当地面站处于卫星的覆盖角内时，即可建立 GSL（图中蓝白条纹连线，如 Fig.2 (i)）。第三方卫星（Fig.2 (g)）或飞行器（Fig.2 (f)）则通过黄色白条纹线建立 MSL 链路。

SNK-Visualizer 提供两种视图模式：三维视角（Fig.2 (a,b,c)）与二维视角（Fig.2 (e,d)），以支持对场景实体的全面分析与观察。

D. SNK-Server

SNK-Server 由 Python 开发，主要用于网络仿真与与 SNK-Visualizer 的交互，是整个系统中最复杂的模块，包含三个主要子系统：命令行子系统（CLI subsystem）、API 子系统（API subsystem）与流程子系统（Procedure subsystem）。CLI 子系统作为命令行交互接口，主要负责用户交互与操作逻辑。下文我们重点介绍 API 子系统与流程子系统。

1) API 子系统：SNK-Server 通过 API 子系统与各模块或外部模块进行交互，所采用的交互协议为模块交互协议（Module Interaction Protocol, MIP），该协议基于 WebSocket 开发，支持命令与数据的传输，从而实现数据共享与仿真过程的一致性。

2) 流程子系统（Procedure subsystem）：流程子系统在 SNK-Server 中负责调度与组织各类仿真流程的执行，其执行目的包括：

- **用于定量分析的数据采集。** 执行流程的主要目标之一是定期获取仿真数据，这些数据是评估通信策略或网

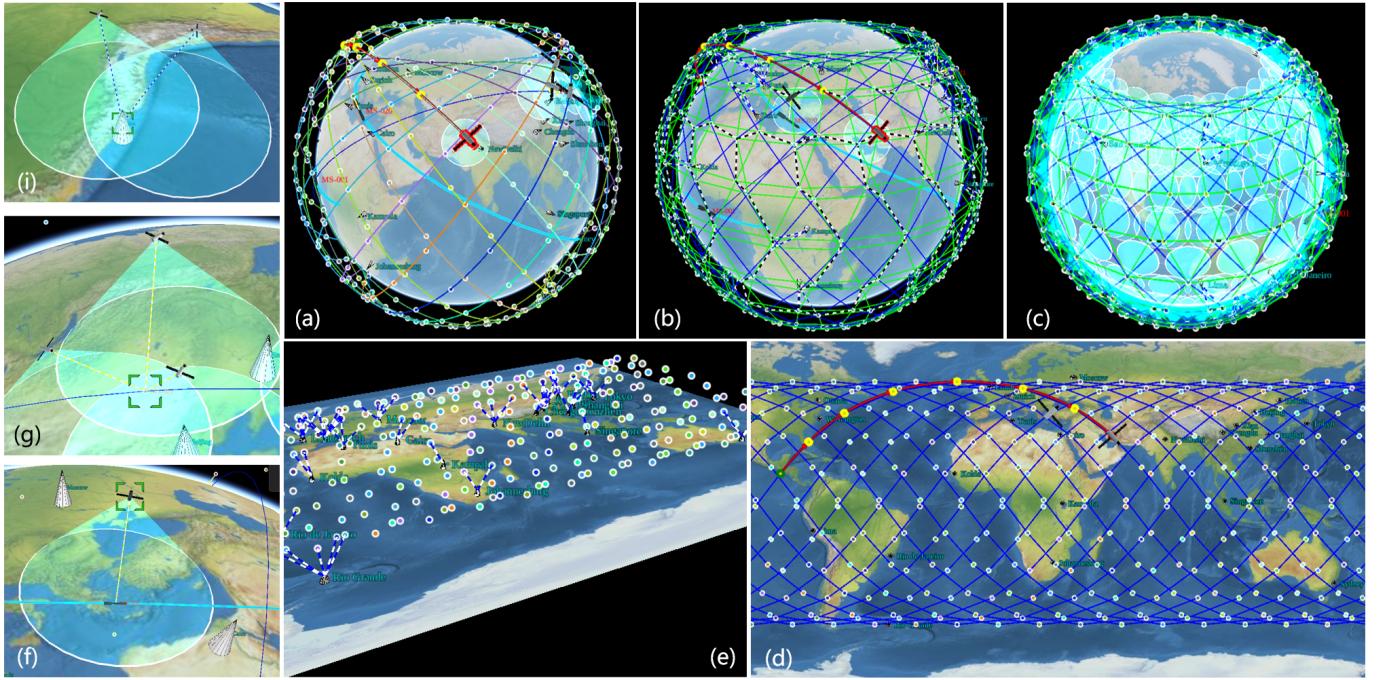


Fig. 2. 由 SNK-Scenario 构建的实体在 SNK-Visualizer 中进行展示。

络性能的关键依据，构成 SNK-Server 的核心功能之一。

- **用于定性分析的通信过程可视化。**通过动态可视化通信过程，用户可以便捷地调试与改进网络策略。

在该子系统中，我们提供了如下流程：

edge2edge 流程该流程执行边缘节点（地面站 GSes 与移动站 MSes）之间的通信过程，依据网络策略定义。激活流程后，将初始化网络策略文件与时间戳，并依照如下步骤执行：

- (1) SNK-Server 启动后，首先从网络策略文件中读取仿真时间戳，并发送 MIP 指令 “set time” 至 SNK-Visualizer 以保持时间一致性。
- (2) SNK-Server 向 SNK-Visualizer 发送请求命令以获取空间场景信息，包括网络拓扑状态、卫星位置与其他相关数据。
- (3) 基于上述数据，SNK-Server 计算源边缘节点与目标边缘节点所覆盖卫星集合之间的路由路径。
- (4) 当仿真到达预设的结束时间戳后，流程终止。
- (5) 流程信息将以 *.ins 文件形式保存，作为该次仿真的实例数据。

conTest 流程该流程用于执行任意两颗卫星节点间的通信过程，主要用于评估网络策略或空间场景。激活流程后，

将初始化内置的路由算法与时间序列列表，并依照如下步骤执行：

- (1) SNK-Server 启动后，首先从网络策略文件中读取仿真时间戳，并发送 MIP 指令 “set time” 至 SNK-Visualizer 以保持时间一致性。
- (2) SNK-Server 向 SNK-Visualizer 发送请求命令以获取空间场景信息，包括网络拓扑状态、卫星位置与其他相关数据。
- (3) 基于上述数据，SNK-Server 计算场景中任意两颗卫星之间的路由路径，使用内置算法进行仿真。
- (4) 若路径数量超过预设阈值 N ，则中断该流程并更新时间戳。
- (5) 当仿真到达预设的结束时间戳后，流程终止。
- (6) 流程信息将以 *.ins 文件形式保存，作为该次仿真的实例数据。

除上述两种流程外，我们还提供以下附加流程：**i) asyncReplay 流程：**该流程可将外部仿真程序生成的事件列表与数据包列表转换为动画数据，并以异步方式进行可视化；**ii) replay 流程：**该流程基于已生成的 *.ins 文件，回放其仿真过程；**iii) exInstruction 流程：**该流程通过 API 子系统支持与外部程序的联合调试。

IV. 仿真实验案例

本节展示 SNK 在刻画与理解新兴 LEO 巨型星座系统方面的有效性。

A. 局部视角实验 (*The Local View*)

该实验涉及固定边缘站点（代表地面站或用户终端）与移动边缘站点（代表功能性卫星或飞行器）之间的路由分析。实验场景如 Tab.I 所示，具体的站点信息列于 Tab.II 中。

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
α	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	0N,24W → 0N,20E
MS-020	MS-020	50N,0E → 50S,180E
MS-021	MS-021	-

1) 固定边缘站点路由：卫星网络（SN）能够有效降低城市间的网络时延，并可作为海底光缆的替代解决方案。因此，我们基于 SN 分析位于大城市的固定边缘站点之间的路由路径。当使用 *SNK-Scenario* 构建场景并运行 *edge2edge* 流程后，*SNK-Server* 将生成 *.ins 目录，并由 *SNK-Analyzer* 提取路由信息。

Fig.3 (a) 显示了 2000 秒仿真过程中，不同城市对在两种路由策略下的路径时延：最少跳数（虚线）与最短路径（实线）路由。SNK 已内置基础的最短路径算法，并支持用户简单开发更复杂的路由算法。Fig.3 (b) 左图展示了路径的拉伸比（stretch）分布，该指标为路由路径距离与地理距离之比，反映路径绕行程度。拉伸比越小，说明路径越接近地理弧线，表示路由效率越高。当 $stretch \leq 1.5$ 时，说明 SN 中路径优于城市之间的光纤直连 [13]。从图中可

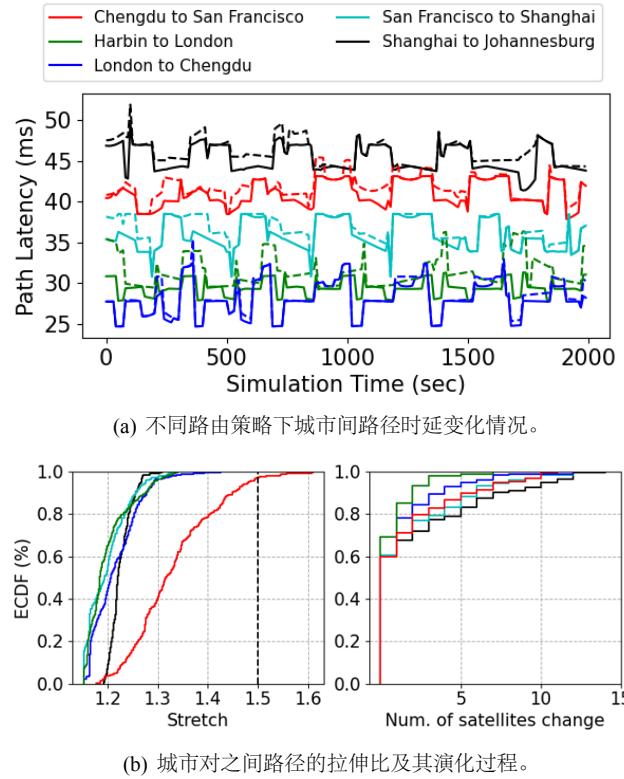


Fig. 3. 空间网络结构分析。

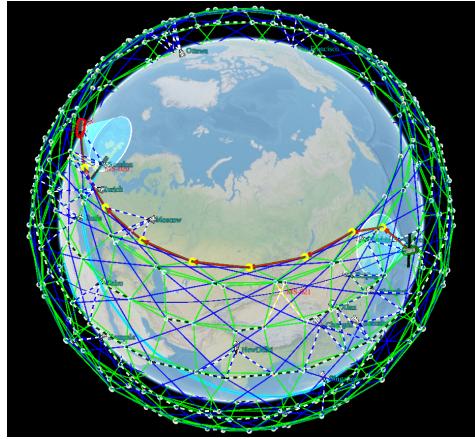
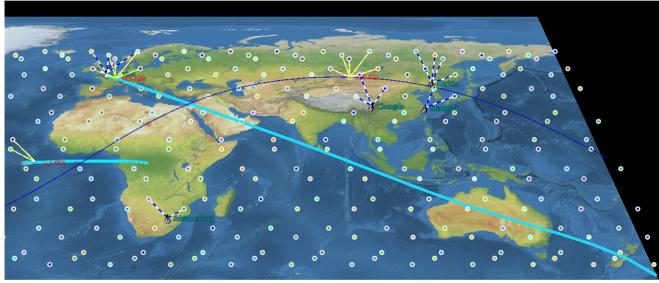


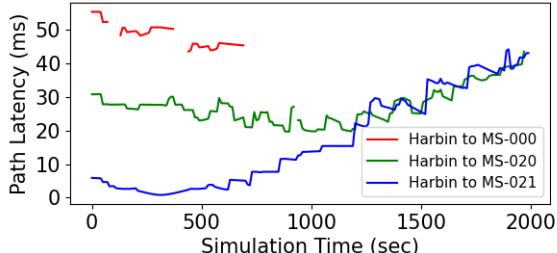
Fig. 4. 在启用“Watch”模式时，可在 *SNK-Visualizer* 中实时可视化路由路径。

以明显看出，大多数策略下，空间网络在性能上优于地面网络。Fig.3 (b) 右图则展示了路径中卫星切换次数的演化，值越小表示路由越稳定。此外，为便于开发路由算法时调试，SNK 支持在“Watch-enabled”模式下实时显示路由路径，如 Fig.4 所示。

2) 移动边缘站点路由：SNK 也支持对带有自定义轨迹的移动边缘节点（如飞行器或第三方卫星如地球观测卫星）之间的路由过程进行仿真。在本实验中，我们构建了多个



(a) 移动站点包括飞行器(MS-001, MS-020)或功能性卫星(MS-021)。



(b) 移动边缘站点的路径时延情况。

Fig. 5. 空间网络结构分析。

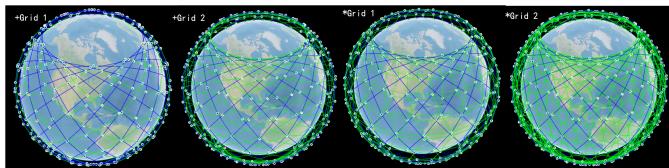


Fig. 6. 不同配置下构建的网络结构差异显著

轨迹不同的移动边缘站点，并评估了它们与哈尔滨之间在不同路由算法下的路径时延，如 Fig.5 (a) 所示。图中细蓝线表示第三方卫星轨道，粗蓝线表示飞行器轨迹，虚黄色线代表移动站-卫星链路 (MSLs)。Fig.5 (b) 展示了移动站点的路径时延情况。需要注意的是，MS-000 与 MS-020 在部分时间段存在路径中断，这是由于其移动过程中并非始终有可达卫星可用。

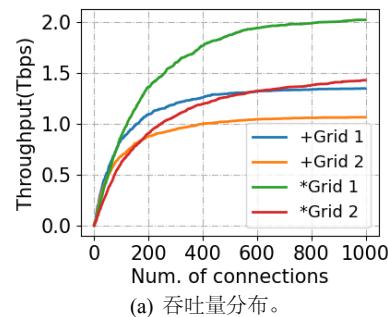
B. 全局视角实验 (The Global View)

现有星座设计主要侧重于覆盖优化或冲突规避，然而显而易见的是，在卫星网络 (SNs) 中设计具备更优网络性能的系统至关重要，这将直接影响其可用性与可行性。因此，SNK 提供了在不同网络结构与密度下量化网络性能的功能，支持评估全球范围内的时延、拉伸比、网络容量与吞吐量等指标。该能力可为 SNs 的结构设计提供依据，已在本实验中体现。场景配置见 Tab.III。

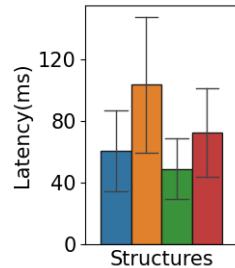
1) 结构分析 (Structure Analysis)：网络结构分析旨在评估特定网络结构下星座系统的网络性能，为提出更稳健的结构设计提供指导。Fig.6 展示了 SNK 中基于不同配

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



(a) 吞吐量分布。



(b) 时延分布。

Fig. 7. 不同网络结构下的吞吐量与时延分布。

置文件生成的网络结构，分别标记为 +Grid 1、+Grid 2、*Grid 1 与 *Grid 2，代表每颗卫星具有 4 或 6 条邻接星间链路 (ISLs) [17], [16]。在上述结构下，可基于自由空间损耗 (Free Space Loss) 计算链路容量，并利用最大流 (Maximum Flow) 算法获得网络吞吐量。Fig.7 展示了不同网络结构下的吞吐量与时延分布，直观反映了网络结构对这些关键性能指标的影响。

2) 密度分析 (Density Analysis)：不同密度的星座网络在性能表现上具有明显差异，密度越高，通常表现出更高的吞吐量与更低的时延。Fig.8 展示了在 *Grid 1 结构下，不同网络密度下的吞吐量与容量情况。可以看出，随着网络规模的提升，网络容量与吞吐量呈现明显增长。然而需要指出的是，由于路由算法仍基于最短路径策略，且未引入负载均衡机制，当前的容量利用率维持在约 25%。

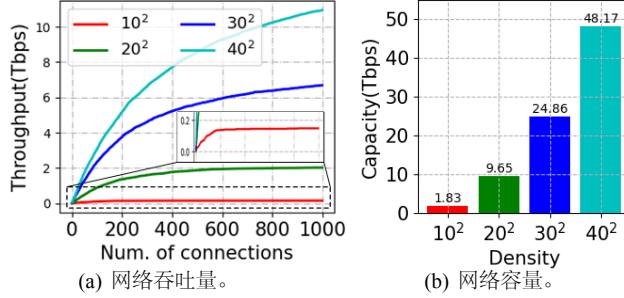


Fig. 8. 在密度为 10^2 , 20^2 , 30^2 , 40^2 的结构中, 基于最短路径路由算法下的网络容量与吞吐量表现。

V. 局限性与未来工作

优化并发处理以提升仿真流畅性。随着空间场景规模与网络复杂度的不断增长, 仿真过程的流畅性有所下降。未来, SNK 将通过优化并发处理机制以实现更平滑的仿真体验。

引入虚拟化技术以实现高保真网络仿真。与近年来专注于网络仿真 [8] 和空间网络仿真 [12] 的其他研究不同, SNK 目前主要依赖模型驱动的方式对网络性能进行估算。为提升仿真保真度, SNK 的最新版本将引入虚拟化技术, 通过将网络节点实体隔离于独立的网络命名空间, 并利用新型链路仿真器 [18], [19] 连接节点, 以构建逼近真实环境的网络仿真平台。该增强将支持系统级仿真, 并有助于与真实系统(如 Starlink 或 OneWeb)进行更准确的性能对比。

VI. 结论

本文提出了 SNK, 一个新型仿真平台, 使星座系统制造商与网络运营商能够在多种星座配置下评估可达成的网络性能。SNK 允许用户通过配置文件构建大规模复杂场景, 仅需一条 bash 命令即可执行仿真与通信过程可视化。基于 SNK, 我们对最短路径与最少跳数两种路由算法的性能进行了评估与对比, 并获得了有关巨型星座网络优化的重要洞见。该平台将以开源形式发布, 助力后续研究人员验证其应用系统与仿真平台。

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.
- [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. Kurjenniemi, “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. Bhattacherjee, A. B. Águas, 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. Bhattacherjee 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. Zeghache, “Dynamic link network emulation: A model-based design,” *arXiv preprint arXiv:2107.07217*, 2021.