POLITECNICO DI TORINO

POLITECNICO DI TORINO

Master Degree course in Communications and Computer Networks Engineering

Master Degree Thesis

Comparison of satellite networks simulators for mega-constellations

Supervisors
Prof. Paolo Giaccone
Prof. Roberto Garello

Candidate
Fatemeh Daneshvar

Academic Year 2023-2024

Acknowledgements

I am deeply thankful to my supervisors, Prof. Paolo Giaccone and Prof. Roberto Garello, for their invaluable advice on this thesis. I would like to express my gratitude to Gabriel Maiolini Capez, for his expert guidance and support throughout this process.

To my parents and to my beloved one, Hamed, for their constant support in every step of my life.

I want to extend heartfelt thanks to my 7-year-old son, Karen, whose infectious joy and patient understanding have added a bright spark to every step of this journey. Your enthusiasm has been a constant source of inspiration.

Abstract

Satellite communications have gained significant importance in recent years, due to their global coverage, versatility across multiple applications, and ability to meet the increasing demand for connectivity in today's interconnected society. Among different satellite communication systems, the design of the Low Earth Orbit (LEO) satellite system is becoming a hot spot in satellite communications, because of its capacity to deliver high-speed internet connectivity with minimal latency. Nowadays, large LEO satellite communication constellations (mega-constellations) are very popular.

The major obstacle to the progress of satellite communication technology is the high cost of research. There is a need for a suitable simulation that can accurately model satellite networks, especially capable of simulating a great number of satellite constellations, providing a cost-effective means for testing and refining new concepts without the need for expensive physical infrastructure. Moreover, simulating networks within LEO mega-constellations presents unique challenges compared to terrestrial networks, primarily due to their substantial spatiotemporal scale, high mobility, and orbital movements. These distinctive properties can pose significant hurdles for traditional network simulators. To provide insight into the available options, this thesis explores various simulators and emulators, evaluating their efficacy through a review of the literature. Moreover, the comparison of different tools is done.

It further examines the scalability of the MATLAB satellite simulation tool, by analyzing its performance in terms of execution time and memory utilization, while considering the simulation duration as well.

Contents

Chapter 1

Introduction

1.1 Introduction

1.1.1 Overview of satellite communications

The motivation for this research arises from the desire to simulate satellite communications for mega-constellations. Several satellite network simulators, emulators, and testbeds are compared in terms of their characteristics, functionalities, capability to support the modeling and designing of a large number of satellites, and their use cases. Satellite communications, which involves the transmission of information through artificial satellites orbiting the earth, serves as a crucial medium for delivering cellular, radio, television, broadband, and military applications to millions worldwide. With over three thousand communication satellites operating across various orbits, these satellites play a pivotal role in relaying information seamlessly across the globe.

How does satellite communication work?

Using microwaves, satellite communications leverage a network of orbiting satellites and ground stations to transmit and relay information between different points on the earth. Satellite communication has some important components and concepts, bellow there is a brief explanation:

Various types of satellites, including communication satellites, are artificial objects intentionally positioned in orbit around the earth. Communication satellites are specifically equipped with transponders that receive, amplify, and retransmit signals as part of their functionality. The next important objects are ground stations. Ground stations, equipped with antennas and other necessary gear, serve as earth-based facilities responsible for communicating with satellites. These stations transmit signals to satellites, which subsequently relay the signals either back to other ground stations or directly to terminals on the earth. Next are transponders. Transponders within communication satellites play a pivotal role in the transmission process. They receive signals from the earth, amplify them, and then retransmit them back to the earth. These transponders operate at precise frequencies within the radio frequency spectrum. Uplink and downlink are two main links that facilitate communication between ground stations and satellites. The

uplink refers to the transmission of signals from a ground station to a satellite, while the downlink involves the reception of signals from the satellite to a ground station. Other important objects are frequency bands. Satellite communications make use of different frequency bands that are allocated for specific purposes. Common frequency bands include C-band, Ku-band, and Ka-band. Each band presents unique advantages and is well-suited for particular types of communication applications. Moreover, satellite types of orbits are important in the communication systems. Satellites have the flexibility to be positioned in different orbits, including Geostationary Orbit (GEO), Medium Earth Orbit (MEO), or LEO. The selection of orbit depends on the specific purpose of the satellite and the desired coverage area.

1.1.2 Satellite communication link full connectivity

Access between two ground stations and a satellite in satellite communication is not always guaranteed; it is contingent upon several factors, including the satellite's orbit, the locations of the ground stations, and the design of the communication system. Here some factors that affect the connectivity is added.

First is the satellite orbit. The type of satellite orbit, whether it is LEO, MEO, or GEO, significantly influences the visibility and accessibility of a satellite from a ground station. Geostationary satellites maintain a fixed position above a specific point on earth's surface, ensuring continuous line-of-sight to a designated ground station. However, satellites in other orbits may experience varying visibility periods due to their orbital dynamics. The second factor is the orbital parameters. Non-geostationary satellites are influenced by parameters such as inclination, eccentricity, and other orbital characteristics, which determine when a satellite becomes visible to a particular ground station. Satellites in polar orbits, for instance, may be observable by a ground station during each orbit due to the nature of their orbital path. The third one is the ground station location. The geographical positioning of ground stations is paramount in satellite communication. These stations must be strategically situated to maintain uninterrupted communication coverage as satellites traverse the sky. Forth factor is the satellite constellation design. In constellations comprising multiple satellites, the design often incorporates handovers of communication responsibilities. These handovers can occur between satellites or between ground stations to ensure continuous connectivity is maintained. The fifth factor is link budget and antenna characteristics. The effective communication range between a satellite and a ground station is influenced by the link budget, which takes into account factors such as transmitted power, antenna gains, and path losses. The sixth factor is atmospheric conditions. Atmospheric conditions can impact the quality of the communication link. For instance, rain, clouds, and other weather phenomena can cause signal attenuation, affecting the reliability of the connection.

Note that advanced systems often incorporate multiple ground stations or satellites to ensure redundancy and continuous coverage. Moreover, they may feature designed handovers between ground stations or satellites to sustain connectivity as the satellite traverses its orbit.

1.1.3 Satellite Keplerian elements

To define a satellite orbit, certain elements are necessary, known as satellite orbital or Keplerian elements, named after Johann Kepler (1571-1630). The Keplerian model, though simpler than reality, provides a useful framework. It defines an ellipse, orients it around the earth, and positions the satellite on the ellipse at a specific time. These elements are collectively referred to as Classical Orbital Elements (COEs). The basic orbital elements include:

- **Semi-major axis**

  Gives the size of the orbit, denoted as a. It is the average distance from the center of the earth to the satellite. To understand this parameter, two terms are important, perigee and apogee. The perigee is the point in the orbit closest to earth's surface. The apogee is the point in the orbit farthest from earth's surface.

$$a = \frac{\text{perigee distance} + \text{apogee distance}}{2} \tag{1.1}$$

- **Eccentricity**

  Gives the shape of the orbit, denoted as e. Kepler's first law tolls as that all orbits are ellipses, either ellipses, circles, or part of one. It is the measure of how much the orbit deviates from a perfect circle. when e=0.0, the shape is a circle. The closer the eccentricity gets to 1.0, the flatter the orbit becomes. So it tells how round or flat the orbit is.

- **Orbital inclination**

  Represented by the symbol i. Inclination defines the angle of the orbit plane relative to the equator of the central body (usually the earth). It signifies the tilt of the orbit concerning earth's equator. The inclination is measured in degrees and ranges from 0 to 180 degrees. An inclination of 90 degrees indicates a polar orbit, where the satellite passes over the earth's poles.

- **Right Ascension of Ascending Node (RAAN)**

  It is denoted as Ω and measured in degrees, it specifies the orientation of the orbit in the plane of the earth's orbit. The right ascension of the ascending node is the angle between the line of nodes (where the satellite's orbit crosses the reference plane) and the vernal equinox. Vernal equinox direction is a line or vector from the center of the earth through the center of the sun on the first day of spring.

- **Argument of perigee**

  It is denoted as ω, and represents the orbital location. It represents the angle between the ascending node and the satellite's closest approach to the earth, which is known as periapsis.

- **Time of periapsis passage**

  It is the time when the element sets of Two Line Element (TLE) are accurate.

- **True anomaly**

  It is an angular parameter used in orbital mechanics to describe the position of a satellite. The true anomaly is the angle between the direction of periapsis (closest approach to the central body, usually earth for satellites) and the current position of the satellite along its orbit.

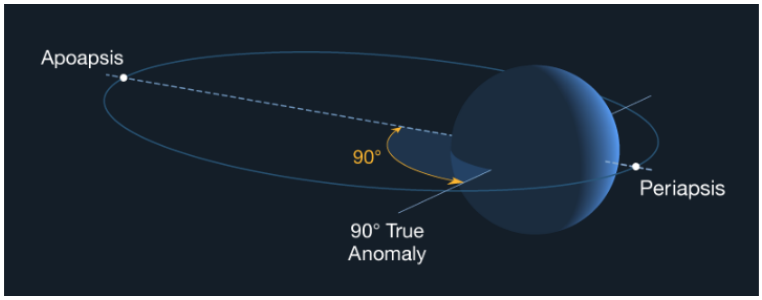


Figure 1.1. True anomaly (reproduced from [17])

- **Mean motion**

  The mean motion of a satellite is a key orbital parameter that defines the number of orbits completed by the satellite in a given unit of time. It is denoted by the symbol n and is typically measured in revolutions per day or degrees per day.

- **Mean anomaly**

  The mean anomaly represents the position of the satellite along its orbit at a specific time. Measured in degree, it is an angle that increases uniformly with time and serves as a time parameter for predicting the satellite's position.

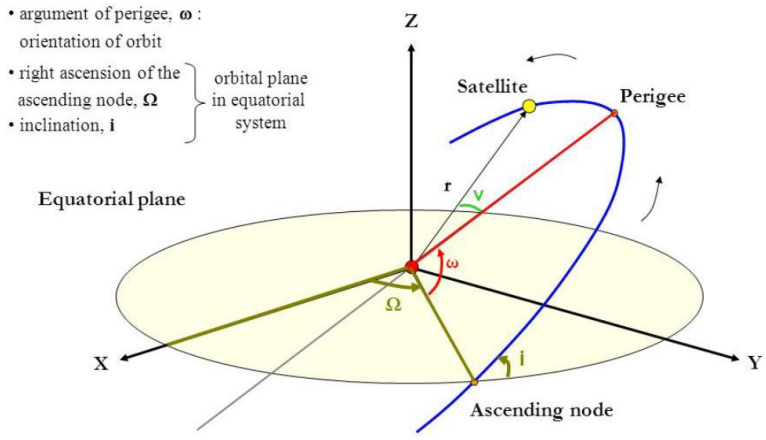


Figure 1.2. Satellite orbital elements (reproduced from [21])

Knowing these orbital elements we can see how they are used in real applications. Two line element sets are used throughout space industries, to list the COEs in a standard manner. Here is an example of a TLE:

1- **1000U** 16001A **17352.66420480** .00000000 00000-0 00000-0 0 11

2- 1000 **78.9943 212.9175 0001051 60.5293 54.8812 15.54194944** 2971

It consists of two lines, 1 and 2 and that's why we have the name two lines identifier!

First element (here is 1000U) is catalogue number. Everything that is tracked in the space has a number, using that it can be more easily managed. The next important element is the time where the element sets of TLE are accurate. In this example is 17352.66420480. The format is a 2-digit year (17), a 3-digit day (352), decimal, fraction the day format. Therefore, in this example will be 2017, 352th day (December 18th), at 4:00 pm Coordinated Universal Time (CUT).

In the second line, the second element is inclination which is 78.9943 degrees in this example.

The next one is RAAN. In this example it is 212.9175 degrees.

The next is the eccentricity, which is 0001051.

Next is argument of perigee, 60.5293 degrees.

Next one is the mean or true anomaly that is the satellite's location for the given time. It is 54.8812 degrees.

The semi-major axis is hidden in the next number, called mean motion. For this example, mean motion is 15.54194944 revolutions per day, which means the object will orbit the earth 15.5 times a day. We can divide 24 hours or 85.400 seconds by 15.30 revolutions per day, giving us a period of 5574 seconds or 93 minutes. Using Kepler's third law, we can determine that the semi-major axis is 6.795 km.

1.1.4 Satellite link budgets

When we design and simulate a satellite communication scenario, we need to be sure that our design will work. This design is under the effect of several parameters. These parameters including frequency, modulation, coding, bandwidth, data rate, required Eb/No, symbol rate, waveform, antenna size, gain, pointing, polarization, transmit power, receive sensitivity, noise figure, noise temperature, losses, margins, and availability. ITU which stands for International Telecommunication Union, which is the United Nations' specialized agency for information and communication technologies, [24] predicts the various propagation parameters needed in planning earth-space networks/systems operating in either the earth-to-space or space-to-earth direction. REC. ITU-R P.618-14 [25] is the latest published version of the Radiocommunication Sector(08/2023).

1.1.5 DVB-S2 and DVB-RCS2

European Telecommunications Standards Institute (ETSI) in 2005, standardized Digital Video Broadcasting Second Generation (DVB-S2), primarily for broadcast services, broadband applications [4]. On the other hand, Digital Video Broadcasting Second Generation Return Channel Satellite (DVB-RCS2)([5] represents the second generation of interactive satellite return channel specifications, aiming to enhance satellite technology's

competitiveness by offering modern IP-based services to customers. These standards provide frameworks for satellite communication systems, with DVB-S2 focusing on forward links and DVB-RCS2 addressing return links.

1.1.6 Types of orbits

When the satellite is launched, it will be placed in different kinds of orbits. There are different factors to decide which orbit to choose for the satellites. For the purpose of this thesis, the most important ones are GEO, MEO, and LEO.

GEO

Circling earth above the equator from west to east in sync with the earth's rotation, satellites in GEO complete one orbit every 23 hours, 56 minutes, and 4 seconds. They maintain their position relative to earth, giving the impression of being stationary over a fixed point. The speed of GEO satellites, approximately three km per second, matches earth's rotation rate, ensuring their synchronous motion. Positioned at an altitude of 35,786 km, GEO satellites orbit much farther from earth's surface than many other satellites. Fig. 1.3 shows the GEO.

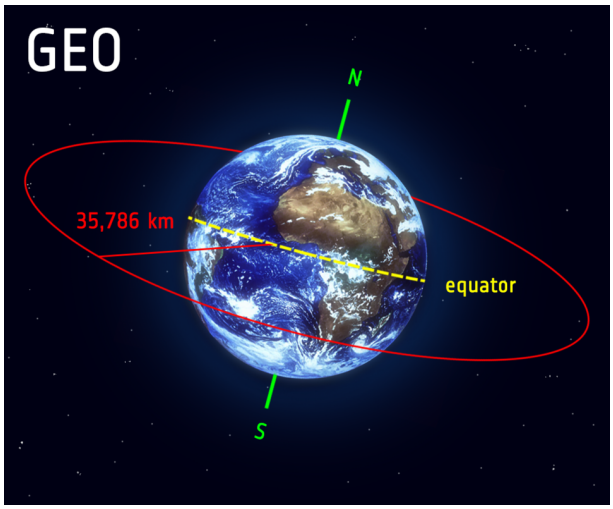


Figure 1.3. GEO (reproduced from [22])

MEO

MEO encompasses a diverse range of orbits situated between LEO and GEO. Satellites in this orbit do not follow specific paths around the earth, and it accommodates a variety of satellites with diverse applications. Navigation satellites, such as the European Galileo System, commonly utilize MEO. Galileo facilitates navigation communications across Europe and serves various navigation purposes, from tracking large aircraft to

providing directions on smartphones. The Galileo system operates through a constellation of multiple satellites, ensuring broad coverage across extensive areas simultaneously.

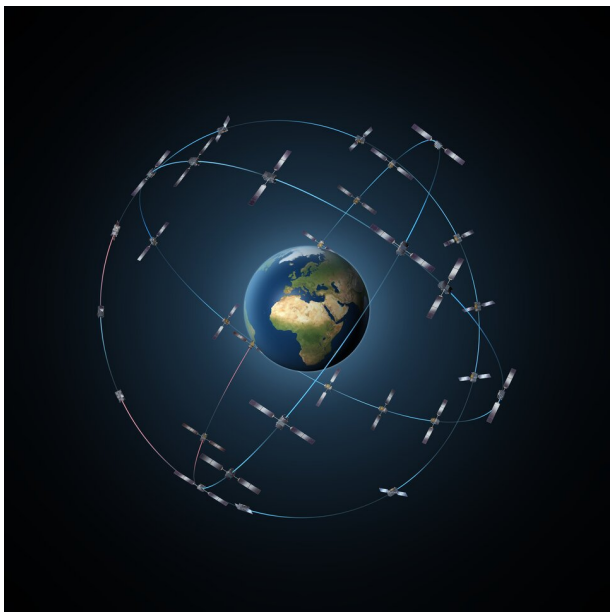


Figure 1.4. Galileo constellation (reproduced from [22])

LEO

A LEO is, as the name implies, an orbit situated relatively close to earth's surface. Typically, LEO is at an altitude of less than 1000 km, but it can be as low as 160 km above earth. A height considered low compared to other orbital altitudes, yet significantly distant from earth's surface. Most commercial airplanes operate at altitudes not much greater than approximately 14 km, making even the lowest LEO more than ten times higher. In contrast to satellites in GEO, which must orbit along earth's equator, LEO satellites are not confined to a specific path around earth, they can have tilted planes. Consequently, LEO offers more flexibility in terms of available orbital routes, making it a widely utilized orbit.

Nevertheless, individual LEO satellites are less practical for tasks like telecommunication due to their rapid movement across the sky, demanding considerable tracking efforts from ground stations. To overcome this limitation, communication satellites in LEO often operate within extensive constellations comprising multiple satellites, which are called mega-constellations, ensuring continuous coverage. To expand coverage further, constellations may consist of several identical or similar satellites launched simultaneously, effectively creating a 'net' around earth. This collaborative approach allows them to collectively cover extensive areas of the earth simultaneously.

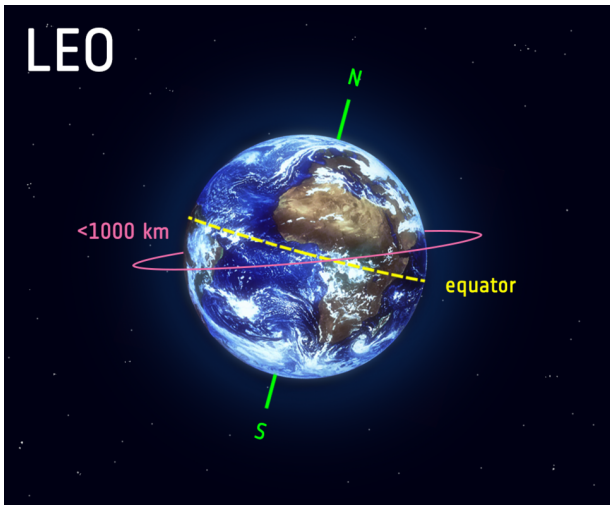


Figure 1.5. Low earth orbit (reproduced from [22])

Chapter 2

Satellite simulation analysis

Satellite network simulators, emulators, and real testbeds are tools used for different purposes in education, research, development, and testing. Simulators are software-based tools that replicate the behavior of satellite networks in a virtual environment. They simulate network conditions, satellite characteristics, and communication scenarios to analyze how the network performs under various conditions. Simulators provide a cost-effective and controlled way to study network behavior without the need for physical hardware. They are particularly useful for exploring theoretical concepts and conducting large-scale simulations.

On the other hand, emulators replicate the actual hardware and software of satellite networks, allowing researchers to test and validate their solutions in an environment that closely resembles the real system. Unlike simulators, emulators involve real hardware and software components, enabling more realistic testing and validation of protocols, applications, and services. Emulators bridge the gap between simulation and real-world deployment, offering a compromise between cost-effectiveness and realism. They allow for more accurate assessments of how solutions will perform in actual operational conditions.

Real testbeds involve physical implementation of satellite network components in a controlled environment to conduct practical experiments and assessments. In a real testbed, actual satellite equipment, ground stations, and communication links are set up to mimic a real-world scenario. This provides the most accurate representation of how a satellite network would behave in practice. Real testbeds offer the highest level of realism and are essential for validating solutions before deployment. They allow researchers to assess performance, identify issues, and fine-tune configurations based on real-world conditions.

2.0.1 Classifications

This chapter represents a comprehensive study of several simulators, emulators, and testbed in terms of their different features, capabilities, and availability. Not all the tools are suitable for the modeling of mega-constellations. The reason is that, the design, optimization, and operation of mega-constellations pose significant challenges due to their large-scale and complex nature. Simulation-based analysis plays a crucial role in evaluating the performance, reliability, and feasibility of mega constellation systems under

various scenarios and conditions.

The key features and requirements of simulators used for modeling mega-constellations, including **scalability** which is the capability of the simulator to support working with a large number of satellites, in terms of factors like time and memory. **Orbital dynamics modeling** to model the motion of the satellites, **communication modeling** which is a simulation of transmission and reception of signals. The ability to **model the different kinds** of links which are ground station to satellite, satellite to satellite, and satellite to ground station, **coverage analysis** to evaluate the extent of signal coverage, signal strength, and quality of service metrics such as Bit Error Rate (BER) or Signal-to-Noise Ratio (SNR) across different geographic locations, **constellation management** which refers to the strategic planning, operation, and optimization of a satellite constellation to ensure its effective and efficient performance in providing desired services, **environmental factors consideration**, and **visualization**.

This chapter provides the classification of the satellite simulation tools, first based on the type of the tool. The first group is simulators, Sec. 2.1, the second group is emulators, Sec. 2.2 and the third group is real testbeds, Sec. 2.3.

In each section, the simulators are two types. Satellite simulator without the support of mega-constellations and satellite simulator with the support of mega-constellations. It is started with the most powerful ones which in my idea are better choices for the simulation of mega-constellations based on the criteria mentioned above and then other satellite simulators are added.

Please note that a comprehensive table of the simulators mentioned in this thesis can be found at the following link: Link to the table

2.1 Simulators

2.1.1 MATLAB satellite simulation toolbox

Satellite communications toolbox [27] was introduced in March 2021, its first release, version 1.0, named R2021a, included satellite link budget analysis, introducing satellite communications toolbox, use of satellite scenario object for simulation, analysis and visualize satellites in orbit, standards-based waveform generation, channel models and Radio Frequency (RF) propagation loss, signal recovery, C and C++ code generation support. The last release, until now is R2023b, in which the bugs are fixed and the features are improved. In the last release, there exist more examples of support for Hardware Description Language (HDL) code generation and over-the-air testing. There are enhancements to the satellite scenario, support for Consultative Committee for Space Data Systems (CCSDS) waveform generation, and satellite link budget analyzer app.

MATLAB is a powerful tool, it has excellent capabilities for conducting statistical analysis and Monte Carlo simulations. A Monte Carlo simulation is used to model the probability of different outcomes in a process that cannot easily be predicted due to the intervention of random variables. It is a technique used to understand the impact of risk and uncertainty. MATLAB satellite simulation toolbox is not open source. It has perfect documentation online as well as in PDF format, describing in detail the toolbox and its functions. It contains various examples that can be run online on MATLAB online, which

is very effective and fast. There is also the possibility to copy at once the commands and run them offline.

Input parameters

In MATLAB satellite simulation, input parameters are in fact objects and functions, related to the location, transmitter, and receiver, characteristics of satellites and ground stations, as well as atmospheric conditions for links. As an example, if one wants to simulate a very simple end-to-end communication between two ground stations, via 2 satellites, the possible input parameters are time, satellite, gimbals, receivers, transmitters, antennas, and ground stations.

Time, including start time, stop time, and sample time. The start time is the time the simulation will start. Different start times will change the result because this changes the position of the satellite. Stop time is the time to stop the simulation and sample time is the resolution of your simulation. Satellite is another input object which is needed to be defined by its Keplerian orbital elements. Keplerian orbital elements correspond to the scenario start time, for example, semi-major axis, eccentricity, inclination, right ascension of ascending node, the argument of periapsis, and true anomaly. The satellites can be inserted with TLE files all at once. Gimbals are a set of pivoted supports that allow an object to rotate around its axes. Each satellite consists of two gimbals on opposite sides of the satellite. One gimbal holds the receiver antenna and the other gimbal holds the transmitter antenna. The mounting location is specified in Cartesian coordinates in the body frame of the satellite, which is defined by (x,y,z), where x,y, and z are the roll, pitch, and yaw axes respectively, of the satellite. For the receiver, possible properties are name, reacquired Eb/No, gain-to-noise temperature ratio (which is the ratio of the antenna gain to the system noise temperature), pre-receiver loss (which refers to losses that occur before the signal reaches the actual receiver or the front end of the receiving system), system loss, mounting location, mounting angles, coordinate axes. Transmitter properties are name, frequency, bit rate, power, link (the link object defines a link analysis object belonging to the transmitter), system loss, and antenna pattern. Antennas are another input. The antenna specifications can be defined, for example, type of antenna, dish diameter, and aperture efficiency. Ground stations are essential. Their properties are latitude, longitude, and name. Note that based on the scenario, different parameters need to be added, but these are the most important ones for a simple scenario.

Output

The output of the simulation includes statistics, analysis and evaluations of communication link budgets, orbital parameters, and overall system performance, including analyses of satellite constellations, ground station coverage, mission duration, and resource optimization.

Another output can be live satellite data, means receiving live satellite data streams. It is possible to use MATLAB's capabilities for real-time data processing and analysis to extract insights from the incoming data in real-time. Another outputs are logs, reports, and calculations. In the above example, the possible output can be link budget analysis,

including link margin, delays, received power and Doppler shift at the target ground stations. It can be the time duration at which the link between two ground stations is established, in a table with the exact start time, and stop time. Another possible output is the large high dimensional calculations, based on the function you used and the desired results, you can have complex computations as output. You can have plots, logs, 2D and 3D visualizations as well.

MATLAB satellite simulation toolbox features

The toolbox provides several applications and standards-based tools for designing, simulating, and verifying satellite communications systems and links. The toolbox is capable of simulation of a large number of satellites, which refers to mega satellite constellations with reasonable time and memory usage. There is the possibility to measure and visualize the coverage maps for the constellations. Another feature is satellite-satellite communication. This provides satellite-to-satellite communication links as well as earth-satellite and satellite-earth links. The next feature is having the link budget analyzer app, which can perform link analysis and access calculations. The toolbox provides RF propagation models as well to calculate atmospheric loss for a signal. It has a satellite waveform generator app as well. It offers functions to create various standard-based wave-forms, such as CCSDS, Telecommand (TC), Telemetry (TM), DVB-S2, Digital Video Broadcasting Satellite Second Generation Extended (DVB-S2X), Global Positioning System (GPS), for the design, modeling, and verification of satellite communications and navigation systems. Simulating aircraft-satellite communications is another feature.

The next attribute is channel modeling. Channel modeling refers to an over-the-air environment for communication systems. To understand the behavior of electromagnetic waves, it is essential to analyze channel modeling and propagation models. MATLAB provides the design of earth-space links using ITU-R P.618 [24] propagation loss model, the possibility to configure a deep space optical link with the Poisson channel. Simulating end-to-end satellite communications links enables you to configure, simulate, measure, and analyze end-to-end satellite communications links.

2.1.2 System Tool Kit (STK)

STK is a robust simulation tool enabling the construction and analysis of satellite constellations, exploration of air and spacecraft missions, and modeling of hybrid network performance. It can be used both for educational and research objectives.

Availability

STK is a commercial tool, it is not open source and it is not free. This is the website [19]. Ansys STK consists of 4 different versions, where each includes a set of tools. Different versions are [20]:

- PRO

List of tools: STK, STK communications, STK radar, STK Terrain Integrated Rough Earth Model (TIREM), STK urban propagation, STK analysis workbench, STK coverage, STK integration, STK engine, STK parallel computing.

- Premium (Air)

  PRO + STK analyzer, STK EOIR, STK Real-time Tracking Technology (RT3), STK Distributed Simulation (DSim), STK aviator.

- Premium (Space)

  Premium (Air) except STK aviator and including STK astrogator, STK SatPro, STK Space Environment and Effects Tool (SEET), STK Conjunction Analysis Tool (CAT)

- Enterprise

  This is the most complete version which includes all of the previous tools plus Test and Evaluation Tool Kit (TETK) and behavior execution engine.

Features

Ansys STK provides a wide-ranging set of multi-domain, physics-based analysis capabilities tailored specifically for industries like aerospace, defense, telecommunications, and beyond. Its extensive suite of tools stands out for its ability to simulate and analyze intricate systems with remarkable precision and accuracy. From space mission planning and satellite communication to radar systems and missile defense, Ansys STK offers advanced modeling and simulation features to assist in vital decision-making processes and enhance system performance across diverse domains. The most important features of Ansys STK is expressed in the next paragraph.

STK communication tool can be used to simulate RF and optical communications of real-world events. You can model and analyze link budget reports and graphs, signal performance contours, visualization tools for dynamic system performance assessment, detailed rain models, atmospheric loss calculations, RF interference source identification, flexibility to integrate custom models or interference sources. The system features advanced capabilities for multi-beam antennas, where each beam functions as an independent antenna with unique attributes including frequency, RF power level, polarization state, gain, and characteristic type. MATLAB seamlessly integrates with STK, allowing for two-way communication between the two platforms via TCP/IP sockets. With access to over 150 native MATLAB commands, users can model orbital, ballistic, and great arc trajectories and perform analytical functions within the MATLAB workspace. Additionally, MATLAB can parse dynamic data from STK for further mathematical analysis and optimize parameters using data providers for enhanced functionality.

STK leverages parallel computing to distribute complex analysis tasks across multiple computing cores. Furthermore, it provides SDKs for .NET, Java, and Python, enabling parallel execution of custom models and algorithms. STK facilitates real-time Geographic Information System (GIS) display and analysis by seamlessly integrating with ArcGIS.

With features like ArcGIS tracking analyst, users can receive, process, analyze, and visualize real-time data, including GPS data. This integration enables conducting temporal and spatial GIS analyses, visualizing data in both STK and ArcGIS, and understanding the spatial relationships of moving objects within the GIS environment. Moreover, it supports mega-constellation simulation.

Example scenarios implemented in STK

The newer versions of STK must be more powerful to implement more complex scenarios, but no example scenario was found. These examples are from [2] which is published in 2010. The first scenario is launching rockets. To launch a rocket from the Kennedy Space Center with the objective of rendezvousing with the International Space Station (ISS) for critical ration resupply, several steps must be followed. This includes mission planning to calculate the optimal launch window, trajectory, and rendezvous maneuver. Additionally, orbital dynamics and ISS position must be taken into account for successful docking. Coordination with ground control and adherence to safety protocols are crucial throughout the mission.

The second scenario is STK and the solar system. STK offers the capability to explore and understand the solar system comprehensively. Users can visualize planets and even simulate becoming planets to observe solar system orbits from various perspectives. Unique features enhance understanding of planetary motion around the Sun. In this scenario, the mission involves creating the solar system and observing the orbits of each planet, the Sun, and the Moon from different vantage points.

Third scenario relates to constellations and chains. During satellite operations, ground stations command and control satellites, but communication may be hindered when satellites are out of line of sight. In such cases, other satellites act as relays or cross-links, facilitating communication between ground stations and satellites. STK allows for the creation of entire constellations, starting with GEO and progressing to MEO and LEO, with increasing complexity and the need for more satellites as they move closer to the earth. Completing communication chains becomes more challenging as satellites move closer to the earth's surface.

Another example is space and a system of systems. The objective is to establish a continuous chain linking two ground stations on separate continents using various assets, including 1 LEO, 1 MEO, 1 HEO, and 1 GEO satellite, along with an aircraft, a ship, and a ground vehicle. The task involves launching a rocket into the orbital plane of the LEO satellite before creating it. The mission scenario revolves around relaying vital information about an ongoing conflict from one ground station to another, with potential locations including Afghanistan, North Korea, Taiwan, or a chosen site. The challenge lies in coordinating the assets to ensure seamless communication and data transmission across the chain.

Last example is using systems tool kit to model possible cubesat orbits. [35]

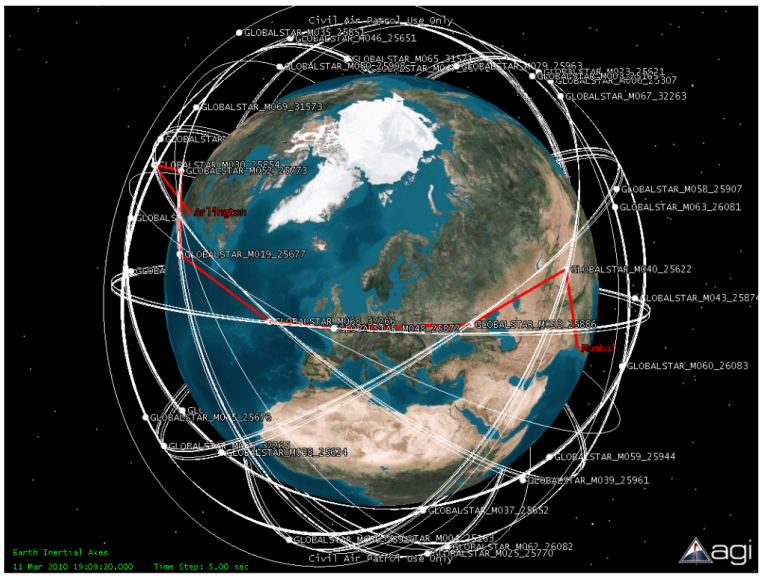


Figure 2.1. STK interface example (reproduced from [2])

2.1.3 Hypatia

Hypatia is a LEO framework for LEO satellite networks [36]. Its primary goal is to fulfill research needs for LEO satellite networks. Hypatia provides a packet-level simulation, built on top of NS3. It takes into account satellite trajectories, coverage constraints for ground station satellite connectivity, and the structure of inter-satellite connectivity. Hypatia consists of visualization to help intuitions. Hypatia uses Cesium to render views of the trajectories. One of the weak points of Hypatia is that it is not flexible to build different scenarios and has limited visualization capabilities.

Components

The system comprises three components. The first component is "satgenpy". This component is a Python framework designed for generating LEO satellite networks. It facilitates the creation of satellite networks and the generation of routing information over a specified period. Additionally, it offers various analysis tools to examine specific cases within the generated networks.

Next one is "ns3-sat-sim". Built upon the ns-3 simulation framework, this component leverages the output generated by satgenpy to conduct packet-level simulations over LEO satellite networks. It takes the network state provided by satgenpy as input and performs detailed simulations to analyze network behavior and performance.

And third component is "satviz". This component is a Cesium visualization pipeline that enables the creation of interactive visualizations of satellite networks. It utilizes Cesium, a platform for 3D mapping and visualization, to generate visually engaging representations of the satellite network topology and dynamics.

Input [36]

Five files are needed to analyze/simulate a LEO satellite network. The first file is "ground-stations.txt" which describes the ground station properties and locations. Next files are "tles.txt", which are TLEs describing the orbit of all satellites. The third file is "isls.txt", which is the topology of the inter-satellite links. Forth file is "gsl-interfaces-info.txt" which gives the number of Ground-to-Satellite Links (GSL) interfaces per node (both satellites and ground stations). Another file is "description.txt" which contains descriptive information (in particular, max. ISL/GSL length). And the last one is "dynamic-state", which is the dynamic state which encompasses (a) forwarding state (fstate) and (b) gsl interface bandwidth (gsl-if-bandwidth).

Output

Outputs of the Hypatia simulator include plots to analyze Round Trip Time(RTT), traffic, bandwidth, etc, as well as visualization, showing link utilization or traffic matrix.

Example scenario implemented in Hypatia

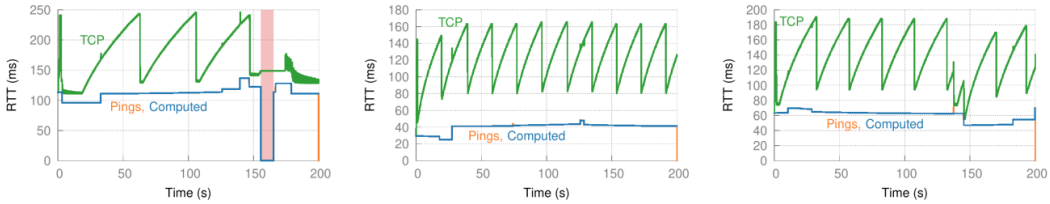
In [36] the scenario demonstrates Hypatia's utility in understanding the behavior of LEO satellite networks rather than solving the topology design, routing, or transport problems that arise due to the high dynamicity of LEO satellite networks. The experimental setup contains satellite network setup. The network will be first shell of kuiper(K1), Starlink(S1) or Telesat(T1). Another setup is to set the inter-satellite connectivity. The inter-satellite connectivity will be "+Grid", which is a north-south, east-west connectivity pattern. You need to set the ground station locations which are the top ten most populated cities in 2025. Another setup is time step intervals, which are the intervals at which the states are calculated are 50 ms, 100 ms (default), and 1s, and the paper has a comparison between the three. You need to set the network device rate. The bandwidth is set to 10 Mb/s for both the ground satellite and the inter-satellite network devices.

The report of the results of the scenario is presented in the following.

Results on RTT fluctuations:

It shows how the end-to-end RTTs vary over time. These experiments use the Kuipler K1 shell, which consists of 34 orbits and 34 satellites for each orbit, and operates at the height of 630 km and an inclination of 1.90 degrees. The analysis is run for 200 seconds and time resolution is 100 ms and the shortest path is computed using the Floyd-Warshall algorithm. They examined an end-to-end path from Rio de Janeiro in Brazil to Saint Petersburg in Russia. Fig 2.2 shows the result.

The x-axis is the elapsed time and the y-axis is the RTT. If you look at the blue line, st t=33 s, the path has changed, and the fluctuation is due to these path changes over time. At t = 166 s the path has a disruption, which is shown as the pink in the plot. In this period, Saint Petersburg does not have any visible satellites at sufficiently high angle of elevation, so the network path is disconnected. At the end, in ns3, a TCP newReno



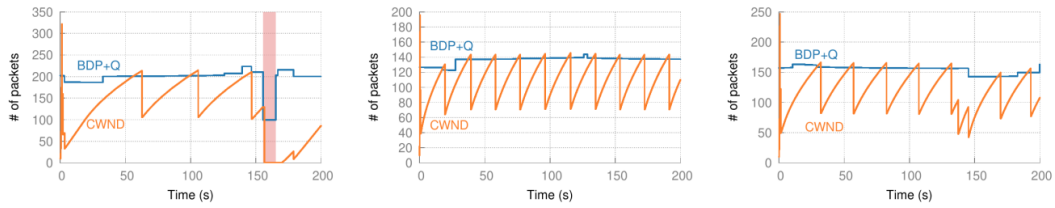
(a) Rio de Janeiro to St. Petersburg (b) Manila to Dalian (c) Istanbul to Nairobi

Figure 2.2. RTT fluctuations (reproduced from [36])

flow between the two locations is run. The TCP RTTs fluctuate over time (green curve), because of the changing of the queuing delay.

Congestion control, absent congestion

How does the congestion control work when the path is changed? First, the setting is congestion-free. The network is free and the one which is measured is only one sending traffic. In Fig 2.3, you can see the TCP congestion window.



(a) Rio de Janeiro to St. Petersburg (b) Manila to Dalian (c) Istanbul to Nairobi

Figure 2.3. TCP congestion window evolution (reproduced from [36])

This time, the x-axis is the elapsed time and the y-axis is the number of packets. The instantaneous Bandwidth Delay Product (BDP) plus the queue capacity, that is BDP+Q, is shown at each point in time. This is the maximum number of packets that can be transmitted without drops. Note that there exists one bottleneck. During periods when the product of the BDP and the number of packets in transit (Q) remains stable, TCP exhibits a predictable pattern of behavior. It consistently reaches this stable threshold, leading to congestion and subsequent packet drops, prompting TCP to reduce its transmission rate before gradually increasing it again. However, fluctuations in the RTT, and consequently in the BDP+Q metric, cause TCP to adapt its behavior accordingly. The disconnection is clear in the Rio de Janeiro to St.Petersburg.

The paper compares how loss-based and delay-based transport with the same experiment. Moreover, to analyze constellation-scale behavior, they utilize the initially planned deployments for Starlink and Kuiper, as well as the first shell for Telesat. They also investigate the configuration of the underlying paths. For each connection, they quantify the

frequency of path changes throughout the simulation period. A path change is recorded whenever the forwarding state computed in two consecutive time steps indicates different satellites comprising the path. They aggregate these path changes across connections and analyze their Cumulative Distribution Function (CDF). Furthermore, for each connection, they compute both the maximum and minimum number of satellite hops observed within the path throughout the simulation duration.

In addition to examining the structure and latency of paths and the response of individual TCP connections, the aim is to understand the outcomes of interactions between traffic flows within such networks. To achieve this objective, they conduct a straightforward experiment involving the transmission of long-running TCP flows between pairs of ground stations over their shortest paths.

2.1.4 Space Networking Kit (SNK)

SNK, is a novel networking platform for LEO mega-constellations, presented in January 2024.

SNK empowers constellation manufacturers and network operators to gauge network performance across diverse constellation options. Through SNK, users can easily construct complex scenarios via configuration files with a single bash command, facilitating evaluation and visualization of communication processes. Through the utilization of SNK, assessments and comparisons are carried out between the shortest-path and least-hop routing algorithms. This process enables the extraction of valuable insights crucial for optimizing mega-constellation networks.

Availability

The paper [32] mentioned that the simulator will be available open source, to help researchers evaluate their experiments. Nothing except this paper is available so far.

Platform scenario and workflow

SNK includes 4 main modules. They are SNK-Scenario, SNK-Visualizer, SNK-Server, and SNK-Analyzer. SNK scenario is used to generate data for the satellite network. The elements such as ground stations, links, satellites, mobile stations, etc. SNK-Visualizer is used to visualize the scenario. Developed in Cesium, and is a Java web application.

SNK-Server is a network simulator based on Python, that facilitates synchronous data transmission with SNK-Visualizer via its Application Programming Interface (API) subsystem. SNK-Analyzer is used to visualize the statistics such as latency of the end-to-end path and path stretch rates. Fig 2.4 shows an overview of the workflow in SNK.

In the SNK-Server, there exist several procedures. The main procedures are $edge2Edge$, $conTest$, $asyncReplay$, $replay$, and "$axInstruction$. The edge2Edge procedure is the procedure that performs the communication process between the edge points (nodes), which can be ground stations or mobile stations. When this procedure is active, a time stamp and a network policy will be activated and it will be run in 5 steps to find the best route and establish the communication. At the end, the procedure information will be kept in a .ins file as the instance data. The conTest procedure is used to establish a

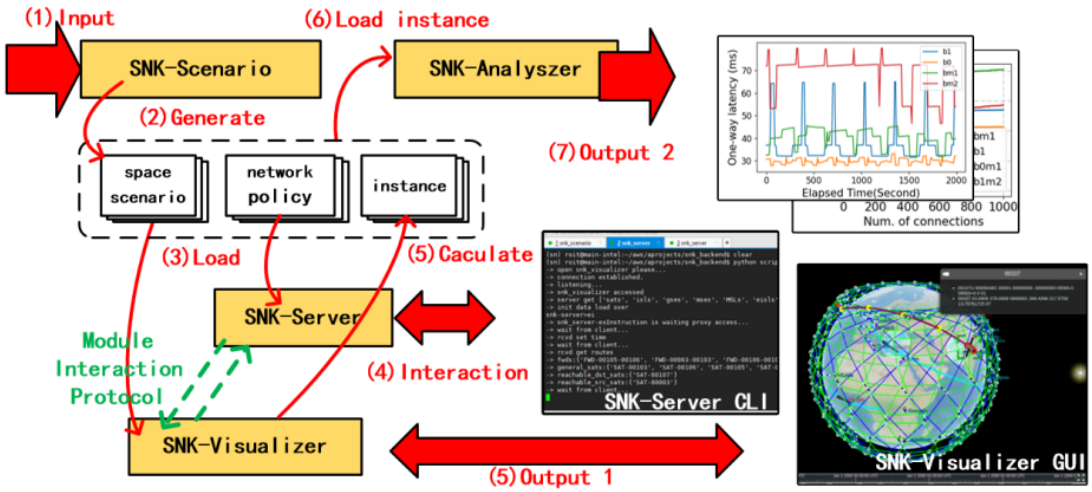


Figure 2.4. Workflow of SNK (reproduced from [32])

communication link between any 2 satellites. When this procedure is activated, a routing algorithm and a time list will be initialized, and by doing 6 steps, the communication is established. At the end, the procedure information will be kept in a .ins file as the instance data. AsyncReplay procedure is used to visualize the animation of the packet list and event list by converting the corresponding data. The replay procedure is used to enable collaborative debugging with external programs through the API subsystem. AxInstruction is a procedure utilizing the acquired *.ins file to replay its process.

Example scenario

To showcase how SNK enhances routing algorithms and satellite network optimization, they utilize SNK to assess and contrast the efficiency of the Dijkstra routing algorithm in intercity scenarios. This allows to gain valuable insights into enhancing constellation design for better edge-to-edge network performance. The findings underscore the substantial influence that deliberate choices of routing algorithms and network configurations can exert on communication dynamics, including factors such as latency, path elongation, throughput, and network capacity.

2.1.5 Satellite Network Simulator 3 (SNS3)

SNS3 was introduced in 2014, as a modular and flexible satellite model built upon the open source Network Simulator 3 (NS-3) [28], incorporating DVB-S2 and DVB-RCS2 specifications for forward and return links, respectively. The simulator is still under development. It was initially developed by Magister Solutions under ESA contract. SNS3 is a scalable and fast open source packet/system level network simulator for networking

research and development [31]. SNS3 simulates interactive multi-spot beam satellite networks featuring geostationary transparent star payload. It is used to perform simulations for a full-fledged system that is also used to perform protocol interactions. In terms of scalability, capable to carry out large-scale network simulations in an efficient way.

Availability

There exist two versions of SNS3. The first version is Magister internal version, which is used and developed in the research and development projects and is used to provide commercial services to the customers. The second version is an open source version hosted by the french space agency CNES.

This is the websites [16], and the GitHub link for the open source version [12], to access the source code and install. It has a nice documentation available [29].

General architecture of SNS3

The architecture of the satellite module comprises models for various components such as user terminals, satellites, gateways, network control centers, terrestrial nodes (end users), and their interactions. Each satellite node requires a new implementation of a NetDevice called SatNetDevice, inheriting from the NetDevice class, which incorporates logical link control, medium access control, and physical classes tailored to each node's specifications. Additionally, a new implementation of the channel class (SatChannel) is necessary to support signal power calculations for satellite systems.

For terrestrial links, the access technology (behind user terminals or gateways) can vary and may include options such as point-to-point, CSMA, or WiFi, with current helper structures assuming CSMA. The network control center is modeled as a shared module among all gateway nodes.

The satellite module accommodates both spherical and geodetic coordinate systems (WGS80 and GRS84) for latitude, longitude, and altitude, in addition to the default Cartesian coordinate system. Introducing a new coordinate system is essential for satellite domain nodes (user terminals, GEO satellites, and gateways). It supports both ground station-to-satellite and satellite-to-satellite communication. Fig 2.5 shows the architecture of this simulator.

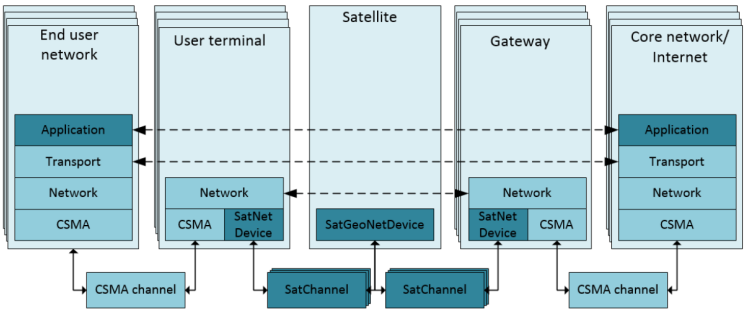


Figure 2.5. General architecture of SNS3 (reproduced from [16])

Other features

Key features encompass [16] first the simulation of the physical channel. The simulator can simulate land mobile satellite channels, Free Space Loss (FSL), propagation delay as well as antenna gain patterns, receiver power calculation, weather traces, noise, link budget parameters, and error modeling. In terms of Medium Access Control (MAC), In forward link, FWD link scheduler, and baseband (BB) frame container can be modeled as well as simulation of random access (slotted ALOHA, contention resolution diversity slotted ALOHA (CRDSA) in return link. Related to Logical Link Control (LLC), modeling of GSE encapsulator, RLE decapsulator, Automatic Repeat Request(ARQ), and packet queues in the forward link plus DAMA request manager, RLE encapsulator, GSE decapsulator, ARQ, and packet queues on the return link. It is capable of simulating packet classifier, and mapping Differentiated Services Code Point (DSCP) to flow identifiers. Modeling satellites is another feature. Transparent star, non-flexible payload can be modeled. In terms of Network Control Centre (NCC), it can model RTN link burst scheduler in return link. The simulator is capable of modeling the network layer. Model IPv4 and IPv6 , plus Internet message protocol (ICMP) and the support of Mobil IP and Open Shortest Path First (OSPF) through Direct Code Execution (DCE) framework of NS3.

SNS3 can model transport layer. Support of modeling TCP, UDP and stream control transmission protocol. Traffic models include modeling Hypertext Transfer Protocol (HTTP), Near-Real Time Video (NRTV), and Constant Bit Rate (CBR). Statistics, which is the support of calculation of the throughput, delay, error probability, Signal-to-Interference-plus-Noise Ratio (SINR), frame load, signaling load, requested resources, and granted resources, etc. Terrestrial network modeling encompasses modeling Wi-Fi, CSMA, Point-to-Point, LTE, Wimax. SNS3 can be integrated with a real-life testbed environment, achieved through specialized network layer to device interfaces. This integration enables the connection of the simulator to the network interfaces of the host machine, leveraging features inherent to ns-3 and ensuring compatibility with real network protocols like IP.

Example scenario implemented in SNS3

A simulation campaign was conducted to showcase example statistics derived from SNS3 [16]. The simulation case involved a single simulated spot-beam with a variable number of user terminals , each representing one end user, and varying traffic loads. The spot-beam was configured with a 125 MHz bandwidth in both forward and return links. The number of user terminals ranged from 25 to 250, with each user terminal offering a load of 1-3 Mbps CBR in either the forward or return direction. Key Performance Indicators (KPIs) such as spectral efficiency and average user throughput were collected. Channel conditions were modeled using channel (weather) time traces.

Example simulation results from SNS3 in the FWD link are shown in Fig 2.6 and in return link in Fig 2.7. The scenarios are the same, except in the RTN link, the frame configuration is different. These are the configuration for the RTN link:

1- 625 KHz carriers (in total 200 carriers)

2- 1.25 MHz carriers (in total 100 carriers)
3- 2.5 MHz carriers (in total 50 carriers)

FWD link

The maximum spectral efficiency in the FWD link, with a 125 MHz bandwidth, reached saturation at 3.3 b/s/Hz. This reflects the advantages of Adaptive Coding and Modulation (ACM), as well as the high probability of favorable channel conditions (Line-of-Sight), leading to a high likelihood of using 32APSK, which is the MODCOD with the highest efficiency. The FWD link was able to support more than 250 users at 1 Mbps, approximately 200 users at 2 Mbps, and around 125 users at 3 Mbps, with no significant degradation in user throughput.

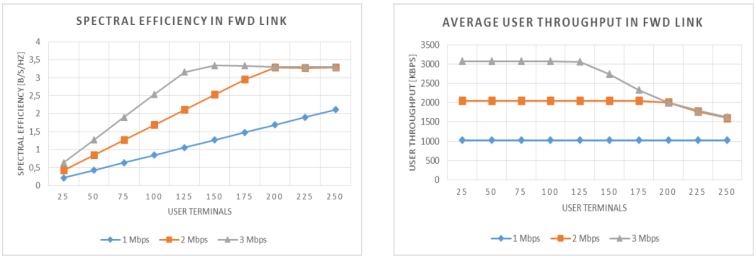


Figure 2.6. Spectral efficiency and throughput in FWD link (reproduced from [16])

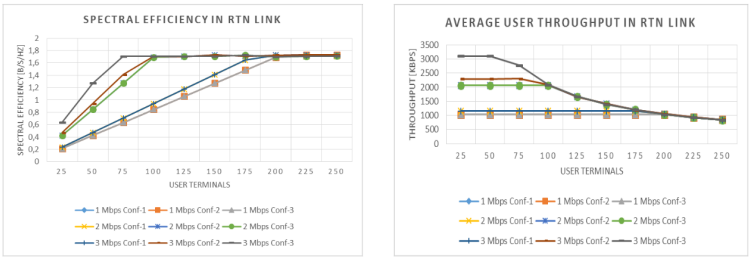


Figure 2.7. Spectral efficiency and throughput in RTN link (reproduced from [16])

RTN link

The spectral efficiency of the RTN link stabilizes at approximately 1.7 b/s/Hz. This indicates a predominant utilization of 16QAM, the most efficient MODCOD in the RTN link. Comparatively, the spectral efficiency of the RTN link is notably lower than that of the FWD link. This discrepancy arises due to several factors which are the utilization of 16QAM instead of 32APSK in the FWD link, the presence of carrier spacing in the RTN link to prevent interference between consecutive carriers, and the limitation of UT scheduling to a single carrier at a time, thereby restricting peak throughput for individual

user terminals. However, even with narrower carriers, the system maintains the same peak spectral efficiency when the number of user terminals is sufficiently increased.

2.1.6 OS3 OMNET++

OS3 [14], is an open source satellite simulator based on OMNET++, which was developed at the Communication Networks Institute in TU Dortmund, Germany. Its initial commit on GitHub was on 14th August 2013, and the development was finished in 2015.

OMNET++ is an extensible, modular, and component-base C++ simulator library [13]. It is expandable and easily adaptive. It is not objected to a special satellite system, but it can be used for arbitrary constellations, so it can be deployed for a variety of applications. The integration of up-to-date TLE files (a collection of parameters detailing a satellite's position over time), altitude (height above sea level), and live weather data gives good end effective connections.

Although OMNET++ has a clear website and documentation and tutorial, unfortunately, It does not have any specific specifications for OS3, and the website link is unreachable on the date of this thesis.

OS3 is used for systems-level simulation and testing procedures. Complementing the INET framework which allows the release under public license as well as a platform independent-implementation. Its focus has been on satellite mobility, satellite constellations, and the inclusion of weather data and channel models, but not on satellite communication protocols. The simulator is an event-driven, reliable TCP/IP stack with close to real systems networking models and it is often used as a reference for academic works.

OS3 enables testing new protocols or satellite orbits and evaluating the resulting performance pertaining to SNR, bit error rate, packet loss, round trip time, jitter, reachability, and other measures [37]. Os3 features a graphical user interface, including visualization options. It can be used to transfer VOIP over a satellite transmission.

Architecture

As the main goal of OS3 is to provide a modular simulation framework for a variety of satellite signal evaluation processes, the class hierarchy is also designed respectively. It includes a graphical user interface based on Java to help simulate set-up processes for the user.

Input

Inputs of the simulation include user-specific parameters which the user may select the satellites and parameters of interest for the whole simulation setup (for example future Galileo constellation, ordinary dipole receiver, and heavy rain). Simulation-specific parameters, up-to-date TLE data, up-to-date position-specific weather data, and high resolutive elevation data.

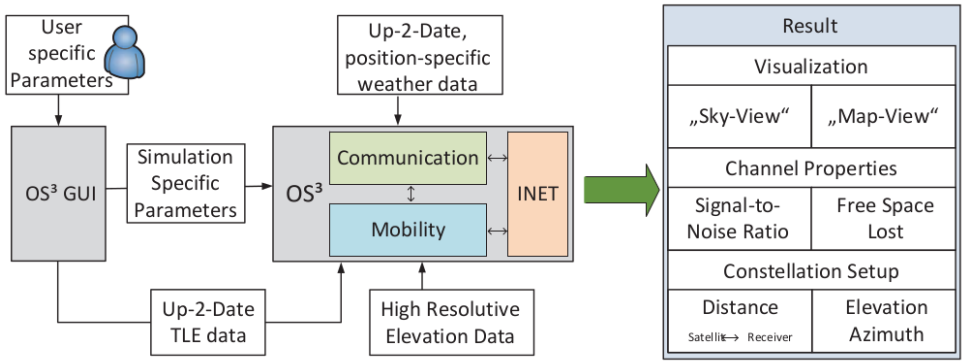


Figure 2.8. Visualization of OS3 simulation (reproduced from [37])

Output

Outputs of the simulation include visualization. There are two kinds of views, "sky-view", and "map-view" as well as channel properties. Channel properties that can be under analysis are signal-to-noise ratio and FSL. Another output relates to constellation setup. This refers to the distance of the satellite to the receiver and the elevation or azimuth of the satellite moving.

Link budget analysis

To evaluate the quality of the received satellite signal, link budgets are used. To calculate the signal-to-noise ratio $\left(\frac{C}{\mathrm{No}}\right)$, transmitter power and transmit antenna gain as well as receiver antenna characteristics need to be defined by the user for the given case and then free space loss can be calculated with the following formula:

$$FSL = \left(\frac{4\pi d}{\lambda}\right)^2 \tag{2.1}$$

with the atmospheric influences, atmospheric loss can be calculated as well. In addition, to consider the performance of the receiver equipment, figure of merit is used. The figure of merit depends on the antenna noise temperature, here is the formula.

$$\frac{G_{\mathrm{R}}}{T_{\mathrm{S}}} \tag{2.2}$$

Where the G_{R} is the receiver antenna gain and T_{s} is the noise temperature of the system.

Performance evaluation of OS3

In [37] is expressed that experimental and simulation tests were conducted to validate satellite movements, comparing real-world and simulated positions provided by DLR who is a German aerospace center, which involves storing first and last visible points of each

pass, considering time, elevation, and azimuth. Direct comparison using == operators is impractical due to inherent inaccuracies, so testing code verifies differences within specified tolerances. Fig 2.9 illustrates the probability density function of deviations in azimuth and elevation for the International Space Station (ISS) from 2012-09-15 to 2012-09-21, showing similarity between simulated and actual data. Further validation scenarios

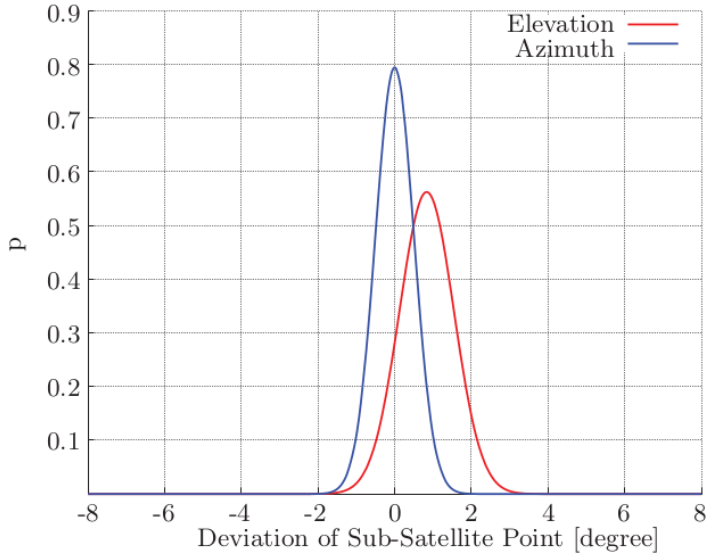


Figure 2.9. reproduced from [37]

are conducted to corroborate simulation accuracy, with similar results observed across different time frames despite inherent limitations.

Additionally, simulated channel characteristics were compared with actual measurements using high-end GPS receivers for experimental validation.

Example scenario implemented in OS3

As an example, [37] shows an example scenario involving using OS3 to simulate a voice transmission over a satellite. In this scenario, it is assumed that all the terrestrial networks are unavailable or damaged. The challenge lies in ensuring connectivity and assessing the impact of signal degradation factors like shadowing and diffraction on voice transmissions. The simulation, coupled with OMNeT++, evaluates SNR to calculate Packet Error Rate (PER), crucial for assessing the usability of satellite links for voice communication. To generate voice packets, as you can see in Fig 2.10, OMNET++ is used which is coupled with OS3 to get an estimation of the current SNR. Based on the SNR, the PER is determined, indicating the satellite link's suitability for voice communication. Jitter, caused by internal buffering, is addressed by increasing packet size to reduce its impact.

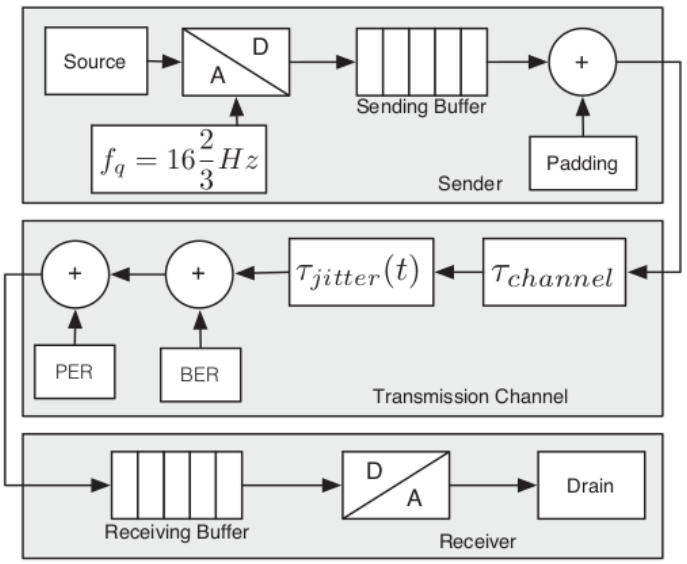


Figure 2.10. Communication model used in OMNET++ (reproduced from [37])

2.1.7 QualNet

It was developed in 2000, and it is still under development, developed by Scalable Network Technologies, Inc. It is widely used for modeling, simulating, and analyzing the performance of communication networks, especially in scenarios where communication endpoints, such as mobile phones, vehicles, aircraft, or satellites, are constantly changing their position relative to each other or to fixed infrastructure. QualNet can be used in modeling satellite networks. Paper [1] aims to tackle the network modeling challenge by integrating an open source satellite mobility model called SatMob into the QualNet network simulation tool. This integration specifically targets LEO/MEO satellites, enhancing the simulation capabilities of QualNet in the context of satellite-based communication networks.

Availability

QualNet is a commercial simulator.

QualNet's architecture

Fig 2.11 shows the architecture of the QualNet's platform. As can be seen, it consists of a command line interface, as well as a graphical user interface. The graphical user interface provides a range of visualization tools to aid in understanding and analyzing network behavior during simulation. It has the ability to visualize different types of packet flows as the scenario runs. This feature allows users to observe the movement of data packets through the network in real-time, providing an operational view of how

the network is functioning. Additionally, dynamic statistics can be displayed while the scenario is running. These statistics provide real-time information on various performance metrics such as throughput, packet loss, delay, and others. By monitoring these dynamic statistics as the simulation progresses, users can gain insights into the performance of the network under different conditions and identify any potential issues or bottlenecks.

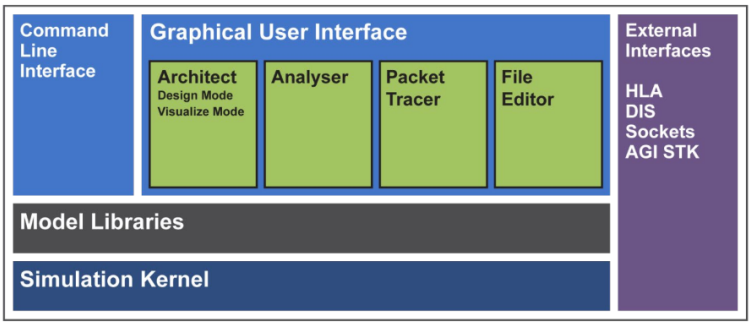


Figure 2.11. Architecture of QualNet (reproduced from [26])

QualNet offers extensive capabilities for post-simulation analysis through its analyzer tool. QualNet's analyzer allows users to plot hundreds of metrics post-simulation. This feature enables detailed analysis of various performance parameters such as throughput, latency, packet loss, path loss, connectivity among nodes, and more. Users can visualize the behavior of these metrics over time or across different simulation scenarios to gain insights into network performance. QualNet enables the generation of detailed time-stamped tables in an SQL database. These tables contain comprehensive information about the simulation, including metrics such as connectivity among nodes, throughput, latency, path loss, packet drop, and others. QualNet allows for seamless integration with third-party tools such as Tableau. Users can export data from the statistics database into formats compatible with external analysis tools like Tableau. This enables the creation of advanced reports, visualizations, and dashboards for in-depth analysis and presentation of simulation results.

QualNet offers versatile interfaces to enhance user interaction and integration with other simulation environments such as: Human-In-The-Loop (HITL) interface and VR-Link interface. QualNet's HITL interface allows for dynamic interactions during a simulation. Users can modify the operations of a running scenario in real-time by activating or deactivating nodes, adjusting traffic rates for specific applications, or making other changes through the interface.

The QualNet VR-Link interface enables seamless integration with other constructive simulators, virtual reality applications, and Computer-Generated Force (CGF) tools. This interface supports industry-standard protocols such as High-Level Architecture (HLA) or Distributed Interaction Simulation (DIS), allowing QualNet to network with external simulation environments like OneSAF Testbed Baseline (OTB) and One Semi-Automated Forces (OneSAF). Through VR-Link, QualNet can exchange data and synchronize simulation events with these external tools, enabling interoperability and collaboration across

different simulation platforms.

QualNet's example scenario

A typical scenario [26] is comprised of nodes, links, environment, mobility patterns and applications, and other sources of traffic operating on the network. Nodes represent the various network elements and endpoints that are part of the simulated network. These nodes can include a wide range of devices and components such as routers, switches, radios, sensors, PCs, servers, satellites, ground stations, mobile phones, access points, and more plus the protocols running on them. Links, connect the nodes in different types, buses, LAN segments, radio transmissions, Wi-Fi signals, LTE connections, etc. The environment in which the network operates (indoors, rural, or urban environment, weather, etc), and the mobility patterns (if any) of the communication devices.

2.1.8 GSSF (The Galileo System Simulation Facility)

It was developed in 2004 and its last update is on 30 September 2010. The simulation's environment is designed to replicate the functional and performance characteristics of the Galileo system, offering inherent flexibility to cater to the system's simulation requirements throughout its entire program life cycle. Primarily, it focuses on providing global coverage analysis for the upcoming Galileo navigation system. It was developed on behalf of the European Space Agency (ESA)/ESTEC, in collaboration with other partners involved in the program.

Availability

GSSF is not an open source software. No GitHub, specifications and software are available.

Features

It takes as input the start date and time (constellation reference time), simulation duration, time step, Galileo constellation, and GPS Constellation. GSSF SVS enables users to evaluate key performance indicators such as visibility, coverage, geometry, Dilution of Precision (DOP), navigation precision, integrity, and service (including critical satellites) on both global or regional grids and individual positions. Additionally, it provides associated metrics like availability and continuity figures for comprehensive assessment.

GSSF V2.0 offers support for performance analyses and early validation of ground segment algorithms. Additionally, it provides GPS/Galileo global interference analysis, Link Budget, and error budget analysis. With the export feature, users can ingest data produced by GSSF into other tools like RINEX/IGSSP3 for further analysis. GSSF offers a reporting feature allowing users to export simulation configuration reports in RTF and PDF formats. This feature enables the auto-generation of comprehensive documents containing the complete simulation/scenario definition and parameter settings for easy documentation and sharing. GSSF offers a modern and flexible user interface featuring

context-sensitive elements like the property grid, along with 2D and 3D figures for enhanced visualization and usability. Fig 2.12 shows an example of the GSSF User Interface.

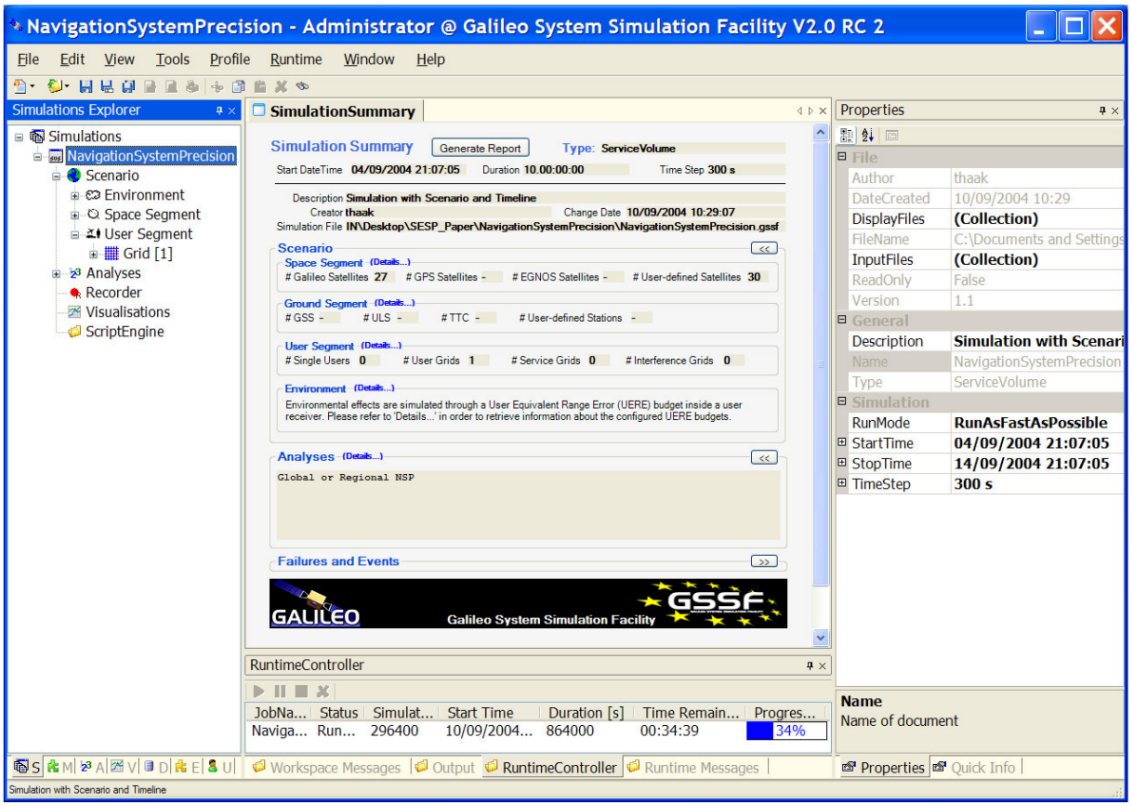


Figure 2.12. The GSSF user interface (workspace view)(reproduced from [6])

Example scenarios

- **Coverage analysis subject to ground station failures:** GSSF offers a coverage analysis that calculates the number of ground stations globally visible from each satellite position throughout the simulation period, typically displayed on a global map. Two simulations were conducted over ten days: the first with all specified Uplink Stations (ULSs) operational, and the second assuming the failure of ULS Hartebeesthoek. As a result, the visibility cone associated with this ground station disappears over South Africa in the second simulation, reducing the depth of coverage by one station in this area. This analysis aids in identifying regions where the density of the ground station network is insufficient for global coverage and informs station location configuration decisions.

- **Independent integrity path coverage analysis:** The independent integrity path coverage analysis in GSSF evaluates the number of available independent

integrity paths globally or regionally during each time step. It assesses whether satellites are within the control range of the ULS network and can receive integrity information. This analysis aims to identify regions with limited independent integrity paths, impacting the availability of Galileo integrity. By considering the reference Galileo constellation and ULS network, the analysis reveals the distribution of independent integrity paths across different regions. For example, Europe demonstrates coverage from 4 independent integrity paths, while South Africa maintains a minimum of 2. [6]

2.1.9 OPNET

It was developed in 2005 and It is still under development. It can be used to simulate any type of network [34]. For systems-level simulation and testing procedures, OPNET offers robust capabilities for modeling communication nodes, links, and network models of satellite communication systems efficiently and effectively. OPNET supports various types of nodes, including fixed, mobile, and satellite nodes, with the flexibility for one node to support multiple types simultaneously. Data generation and processing within nodes are seamlessly integrated into OPNET, facilitating comprehensive simulation. The node editor feature allows easy construction of the internal structure of nodes, enabling modeling of on-the-ground gateways, satellite communication nodes, and routing nodes. OPNET also provides several pipe phases to simulate the transmission process of actual data frames in the channel, enhancing the accuracy of simulation results. They mentioned that they developed more than 80 projects and the strength of OPNET is the flexibility, reliability, scalability, and mobility access. OPNET does not support mega-constellation. It can be used both for research and educational purposes.

Availability

OPNET is a free and open source software. The website [34] and the specifications are available.

OPNET features

In terms of modeling capabilities, the tool boasts a straightforward yet robust modeling approach, employing a hierarchical network model alongside support for finite state machines. Furthermore, it offers a hybrid modeling mechanism and operates as a fully open system, ensuring adaptability and versatility for users.

In terms of analysis tools, this software provides specialized applications tailored for statistical analysis, facilitating interactive exploration of data. Users benefit from graphical specifications and animation functionalities, enhancing their ability to visualize and interpret results effectively.

Referring to customization and integration, the tool excels with an API, empowering users to develop custom models with unparalleled flexibility. Specialization in network and systems enables tailored solutions to complex problems, while automatic simulation generation streamlines workflow processes for increased efficiency.

2.1.10 General Mission Analysis Tool (GMAT)

It was developed in 2007, the last version is R2022 and it is still under development. It is not suitable for simulation of terrestrial networks. It works like MATLAB. It is used for mission planning and orbit propagation analysis and optimization for satellites, being under development by NASA [38] and is used for real-world mission support, engineering studies, as a tool for education, and public engagement [33]. It offers a robust visualization tool tailored for modeling and analyzing missions, particularly focusing on space missions within earth's orbit or the solar system. GMAT supports a graphical user interface. It can be used both for education and research purposes. Users can interact with it using a scripting language similar to the MATLAB system.

Resources, which are objects in programming languages with properties, can be tailored to fulfill the requirements of specific applications and missions within GMAT. These resources encompass a wide array of functionalities, which can be categorized into physical model resources and analysis model resources.

Physical resources comprise spacecraft, thrusters, tanks, ground stations, formations, impulsive burns, finite burns, planets, comets, asteroids, moons, barycenters, and libration points.

Analysis model resources consist of differential correctors, propagators, optimizers, estimators, 3D graphics, x-y plots, report files, ephemeris files, user-defined variables, arrays, strings, coordinate systems, custom subroutines, MATLAB functions, and data. Commands within GMAT function akin to functions in programming languages, enabling users to execute various operations. The capability for evaluating communication link performance in GMAT is limited. Moreover, GMAT features analysis "objects" including propagators, plots, and reports.

Availability

GMAT is open source, extendable and customizable. The specification, GitHub, and website links are available.

Features [18]

GMAT offers external interfaces to MATLAB and Python, allowing for the execution of MATLAB functions within simulations. It finds application in orbit design and optimization, event detection and prediction, maneuver planning and calibration, fuel consumption monitoring, navigation, and more. It facilitates spacecraft mission design and navigation. GMAT provides full mission life-cycle support.

It enables optimized maneuver and trajectory design. Furthermore, supports operational orbit determination with measurement simulation capability.

Example scenarios implemented in GMAT

The first scenario is about mission simulation. The mission is to discover the presence of water in permanently-shadowed creators at the lunar south pole. The goal is optimal lunar flyby to perform phasing and plane change resulting in lunar impact.

The second example scenario is "safty ellips relative motion simulation". This shows a leader spacecraft, and a follower spacecraft, we want to design the relative motion such that if you lose control of the follower, it does not impact the leader.

2.1.11 Gpredict

It was developed in 2007, while the last release on GitHub was on January 21, 2018, and the last update was in 2022. Gpredict is a program for the Linux desktop that provides real-time satellite tracking and orbit prediction. It utilizes the SGP4/SDP4 propagation algorithms along with NORAD TLE to achieve this functionality.

Availability

Gpredict is open source, licensed under the GNU General Public License [23], with the specification available [7], website and GitHub are available as well.

Features

Key functionalities of Gpredict include utilizing multiple ground stations, predicting upcoming passes, displaying tracking data through various formats such as lists, maps, polar plots, or any combination thereof, the tracking numerous satellites, limited only by the computer's memory and processing capabilities. Supporting multiple modules simultaneously, whether within a notebook or in separate windows, including the option for full-screen mode. It can be adapted to provide information in both real-time and non-real-time modes. It is capable of operating in real-time, simulated real-time (with fast forward and backward options), and manual time control modes. It enables Doppler tuning of radios and the last feature is facilitating control of antenna rotators.

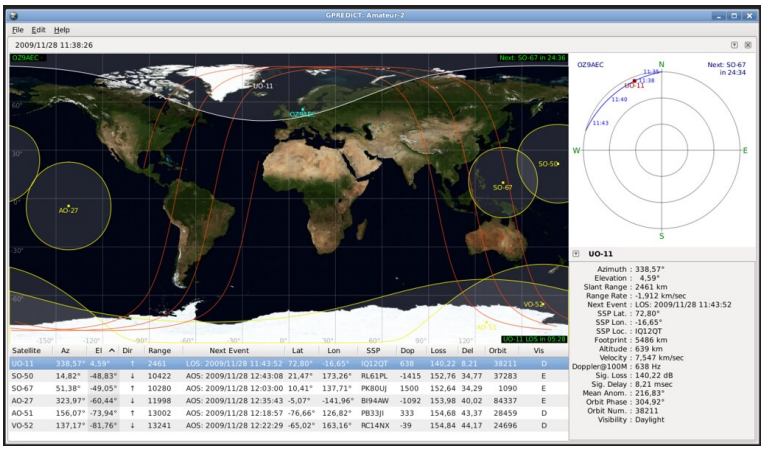


Figure 2.13. An interface of Gpredict tool (reproduced from [7])

2.1.12 NS2

It is a very old network simulator released in 1989, it is not under development anymore. NS2 is not a finalized and refined product, rather, it is the outcome of continuous research and development efforts.

NS2 is a discrete event simulator that offers significant support for simulating TCP, routing, and multicast protocols across wired and wireless networks, including both local and satellite networks. One drawback of NS2 is its absence of a graphical user interface, requiring users to navigate through a complex TCL script-based configuration. NS2 supports MAC, link layer, routing, and transport protocols. It is used in research objectives.

Availability

NS2 is available in open source, while the specification, website and source code are available as well [30].

Other Features

NS2 provides network emulation capabilities, allowing for the simulation of various networking scenarios and protocols. It facilitates the generation of network topologies, enabling users to create diverse and realistic network structures for simulation purposes. It provides meaningful studies of networking issues such as protocol interaction, congestion control, and scalability, requiring simulation of appropriate scenarios. NS2 supports the creation of scenarios encompassing topology size, density distribution, traffic generation, and more, allowing for comprehensive analysis. Another feature is topology animator. NS2 includes "Nam", a Tcl/TK-based animation tool for visualizing network simulation traces and real-world packet traces. Nam supports features like topology layout, packet-level animation, and various data inspection tools, enhancing the simulation experience.

Example scenarios implemented in NS2

Extensions within NS2 enable the simulation of satellite networks, allowing NS2 to model the traditional geostationary "bent-pipe" satellites with multiple users per uplink/downlink and asymmetric links, geostationary satellites with processing payloads, including regenerative payloads or full packet switching capabilities, polar orbiting LEO constellations like Iridium and Teledesic.

2.2 Emulators

2.2.1 Advanced IP Network Emulator (AINE)

AIEN is an IP network emulator that was developed in 2008 and is still under development.

Use-case

AIEN serves as emulation software designed to construct a laboratory environment for real-time performance characterization of networks and systems, with a particular focus on satellite communication systems. While initially conceived for satellite network emulation, AIEN functions as a generic IP emulation tool capable of emulating link layer algorithms as well. This versatility allows AIEN to accommodate various network configurations and system architectures, making it a valuable tool for evaluating the performance of both satellite and terrestrial communication systems.

Availability

AIEN is a commercial emulator. The GitHub, specification and websites are not available.

Features

AIEN was designed for the emulation of the satellite packet systems, the emulation of connection-oriented networks, and the emulation of different encapsulation schemes for DVB-RCS networks. In AIEN, on-demand custom-made modules are available.

It is scalable. AIEN's modularity and non-proprietary nature enable easy expansion with new block types and functions, making it adaptable to diverse testing scenarios. Additionally, it is based on an open source platform, so it facilitates cost-effective implementation of custom modifications to the standard IP protocol suite. AIEN is built using C/C++ language and operates on a Linux platform, leveraging open source software for optimal performance. Its architecture is optimized to maximize efficiency, ensuring streamlined IP packet handling and inter-block communication. It can emulate layer two protocols, although the external interfaces are at IP level. The emulator simulates the encapsulation of IP packets on lower layer protocols which are used on DVB-RCS networks.

AIEN's emulation library features blocks to simulate common communication link impairments like delay, asymmetric bandwidth, bit errors, and packet errors. These impairments allow for thorough testing of network performance under realistic conditions, ensuring system reliability. AIEN supports a distributed architecture for handling complex systems, allowing emulation models to be subdivided into independent subsystems.

2.2.2 OpenSand (PLATINE)

OpenSand was developed in 2005 and the GitHub activity shows that it is still under development. Its objective is to provide an engineering tool and a research tool that is capable of validating access and network functionalities. It can provide analysis tools and measurement points for performance evaluation. It also ensures connectivity with actual networks and applications for demonstration purposes [3]. The key factors of this platform are its user-friendly operation, flexibility to meet scientific needs, easy addition of new features, provision of various measurement points, and comprehensive documentation.

Availability

OpenSand is an open source free satellite network emulator. Its specifications, GitHub link and website are available.

Features

OpenSand supports network-to-network interconnections. Furthermore, it supports IPv4, IPv6 and ethernet connectivity. It is able to link with physical equipment and diverse IP-based networks, spanning terrestrial, satellite, or internet backbone setups. It can map IP-to-MAC queue. OpenSand is capable of implementing the Quality of Service (QoS) architecture to provide differentiated QoS on a satellite network.

An example scenario

The depicted scenario in Fig 2.14 offers a comprehensive view of an architecture than can be implemented in OpenSand. The left side represents the end-user aspect, while the

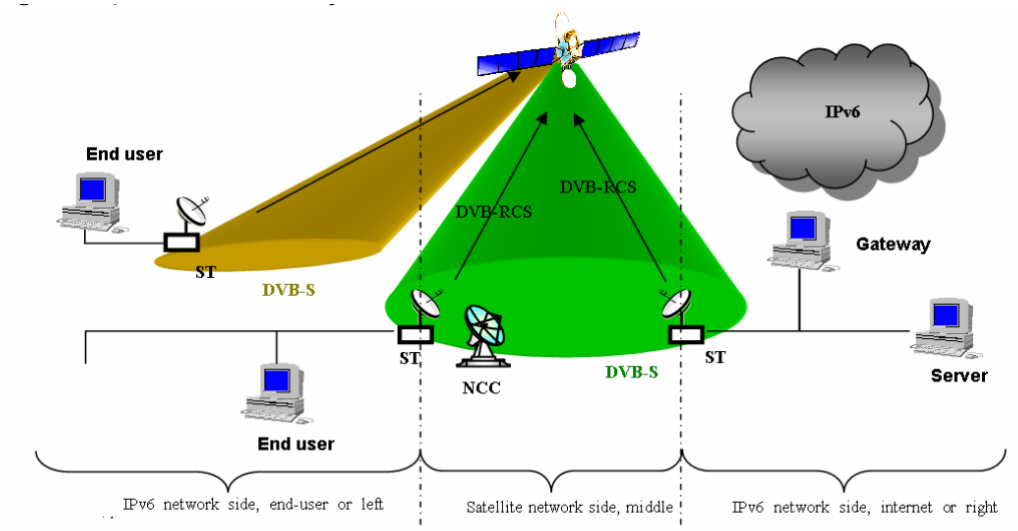


Figure 2.14. OpenSand example scenario (reproduced from [3])

right side depicts the provider/enterprise/internet aspect. Additionally, a distinction is made between the satellite network side (in the middle) and the IP network sides (on the left and right ends), which are interconnected by satellite terminals.

It comprises a geostationary satellite network equipped with onboard switching capabilities, utilizing Ka MF-TDMA (Multiple Frequency Time Division Multiple Access) for uplinks and Ku TDM (Time Division Multiplexed) for downlinks. With regenerative capabilities, the satellite enables direct interconnection between end users with just a single hop. Satellite Terminals grant access to the network for individual PCs or LANs, while Gateways facilitate connections to internet core networks. Uplink access from each RCST

is managed through DVB-RCS interfaces. Both Stations and gateways act as boundary devices between satellite and terrestrial links, crucial for ensuring QoS provisioning by efficiently utilizing satellite resources. These devices implement IP routing and possess an IP interface on the satellite segment, highlighting IP as the common protocol bridging the satellite and terrestrial networks. Consequently, the satellite network is perceived as a distinct link within a traditional network framework.

2.3 Real testbeds

2.3.1 OPENC3

It was founded in June 2022, and it is still under development. OPENC3 comprises a suite of applications designed for managing various embedded systems. These systems encompass a wide range, including test equipment like power supplies, oscilloscopes, and switched power strips, as well as development boards such as Arduino, Raspberry Pi, Beaglebone, and even satellites.

Availability

The tool is available both open source and in the commercial version, as an enterprise edition. The specification is available as well as the GitHub [10] for the source code and the website link [15].

Features

OPENC3 COSMOS is a tool for integration, testing and operations. In the case of working with satellite systems, the live satellite data can be tracked. Anything with a software interface can quickly be connected to COSMOS. TCP, UDP, Serial, MQTT and more are ready to go out of the box. It is the glue that allows you to interconnect and coordinate all the different pieces of the system and is perfect for companies of all sizes. You can command and control the satellites from the web browser of your PC. It has a GUI as well.

2.4 Comparison among satellite network simulators

In this section, we will present a summary of the simulators in different tables with some of their properties in order to group and summarize them.

In the Table 2.1, the first column is the name of the simulator that was presented in this thesis. The second column is the published year, which is found in the published paper or on their website. The third column discusses whether the simulator is still supported, updated, bug fixed and under development. If yes, it is ✓, if not, it is -. The forth column is the last update date, which for the open source simulators, can be derived from their activities and updates on their GitHub, while for others by checking the corresponding website and documentation. The next column expresses if the simulator

simulator	publish year	still under develop	last update date	open source	document support	source code link
NS2	1989	-	2015	✓	High	[30]
OPNET	2005	✓	-	✓	High	[34]
OpenSand	2005	✓	2023	✓	High	[9]
GMAT	2007	✓	2022	✓	High	[18]
Gpredict	2007	✓	2023	✓	High	[8]
NS3	2011	✓	2024	✓	High	[28]
OS3	2013	-	2015	✓	Low	[14]
Hypatia	2020	✓	2022	✓	High	[11]
SNK	2024	✓	-	✓	-	-
SNS3	2014	✓	2023	two versions open source-commercial	High	[12]
OpenC3	2022	✓	2024	two versions open source-commercial	High	[10]
QualNet	2000	✓	2022	No	Low	
GSSF	2004	-	2010	No	Low	
AIEN	2008	✓	-	No	Low	
STK	-	✓	2024	No	High	
MATLAB	2021	✓	2023	No	High	

Table 2.1. The practice guidance of network simulators covered in this thesis

is available open source and free of charge. If yes, it is ✓, and the link to download is available. If not, represented with *No* and without any link.

In the Table 2.2 expresses the general characteristic and use case of network simulators covered in this thesis. The first column is the name of the simulators that are present in this thesis. The second column is the programming language that is used to work with the simulator. The third column is the supported operating system, which will be found in the general requirements of the installation of the simulator. The last column briefly shows the use case.

Table 2.3 presents the support for some of the key simulation functionalities of existing simulators which was not mentioned in the previous tables.

The first column is the name of all the simulators in this thesis. The second column is the protocols and the network layer which can be modeled using these simulators. The third column, will define whether the simulator supports the simulation of mega-constellations or not. If yes, it is ✓, if not, it is -. The last column identifies the support of the simulator for the GUI.

The development timeline of the reviewed simulators in this thesis is shown in Fig 2.15.

simulator	programming language	supported OS	use-case
NS2	NS scripts	Linux	multicast, TCP and routing protocol simulations
QualNet	C++	Linux,Windows Solaris,Mac	evaluation of mobile communication networks, such as mobile phones, vehicles, aircraft, or satellites
GSSF	C, C++		Galileo system simulation
OPNET	C, C++	Windows,Linux	systems-level simulation and testing procedures
OpenSand	C++	Linux	simulate communication channels, protocol testing, DVB-RCS, IPv6 network features,Voip
GMAT	GMAT Syntax	Linux,Mac, Windows	mission planning, orbit propagation analysis
Gpredict	C,C++	Linux,Mac Windows	real time satellite tracking and orbit prediction program
AIEN	C, C++	Linux	generic IP emulation tool to emulate link layer algorithms
NS3	C++, Python	Linux,Mac	
OS3	C++	Linux,Mac, Windows	satellite mobility, satellite constellations channel models, systems-level simulation and testing procedures
SNS3	C++	Linux	geostationary multi-spot beam satellite networks, DVB-RCS2,DVB-S2 simulation
STK	Java,C,C++ .NET framework	Linux,Windows, Mac on STK cloud	build and analyse satellite constellations, model performance of hybrid networks
Hypatia	Python, C++	Linux	routing and visualization of mega-constellations
MATLAB	MATLAB script,C C++, VHDL	Linux,Mac, Windows	link budget analysis, visualization, routing, channel modeling, protocol simulation and testing
OpenC3	commands	Linux	to control a set of embedded systems, like satellites
SNK	modules developed in Python-Java	Linux	optimizing satellite networks, routing via mega-constellations visualization, build large-scale complex scenarios

Table 2.2. General characteristic and use case of network simulators covered in this thesis

simulator	network layer/protocols	support mega constellation	GUI module
MATLAB	physical layer	✓	✓
Hypatia	packet level	✓	✓
SNK	edge of network	✓	✓
STK	physical layer	✓	✓
NS2	link layer, routing, and transport protocols	-	-
QualNet	almost all layers	-	✓
GSSF	Galileo satellite navigation system	-	✓
OPNET	almost all layers	-	✓
OpenSand	physical layer		✓
GMAT	physical layer	-	✓
Gpredict	satellite tracking and prediction	-	✓
AIEN	IP, ethernet	-	✓
NS3	network layer	-	-
OS3	physical layer	-	✓
SNS3	network, link layer, packet level	-	-
OpenC3	command and control, no layers	-	✓

Table 2.3. Key simulation functionalities of existing simulators

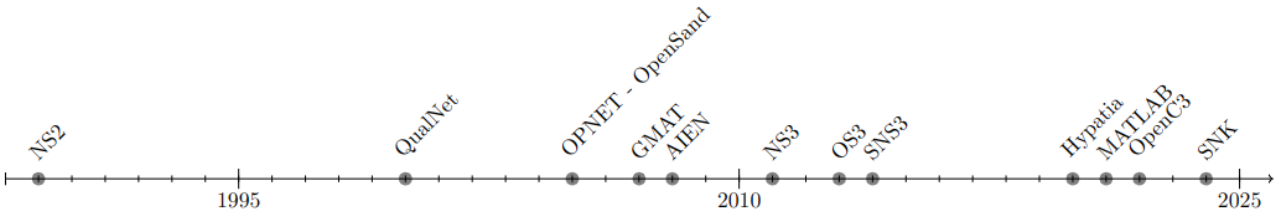


Figure 2.15. Development timeline from 1988 to 2024

Chapter 3

Experimental/Numerical evaluation

3.1 Methodology

- **Execution time measurement**

  To see the execution time of running the code, which is the time the code needs to be run on my PC, the $tic - toc$ function of MATLAB is used. The *tick* is added at the top of the code and the *toc* at the bottom of the code. This is not the CPU time. The simulation has run 10 times, and the minimum and average time is considered. Note that the CPU time is <= min execution time.

- **Memory usage measurement**

  It is not easy to know exactly how much memory is used by the program when running the code. The approach was to firstly, grabbing the process ID of MATLAB using the $pidof MATLAB$ command. Fig 3.1 shows the command with the result, which is the process ID.

Figure 3.1. Outcome of *pidof* command

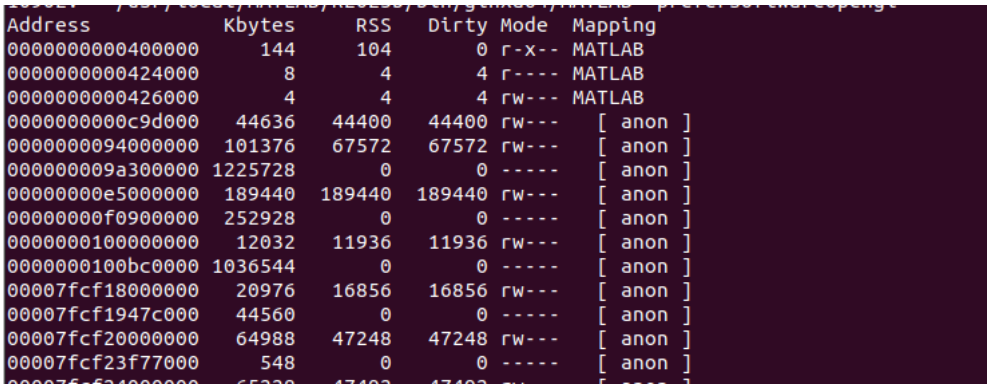
Then *pmap* is used. The *pmap* returns how a process is mapped in memory. The following command returns information in 6 columns:

sudo pmap PID -x

Fig 3.2 shows the result of the *pmap* command. The columns are:

Address: The beginning memory address allocation

Kbytes: Memory allocation in Kilobytes



```
Address           Kbytes     RSS   Dirty Mode  Mapping
0000000000400000     144     104       0 r-x-- MATLAB
0000000000424000       8       4       4 r---- MATLAB
0000000000426000       4       4       4 rw--- MATLAB
0000000000c9d000   44636   44400   44400 rw---   [ anon ]
0000000094000000  101376   67572   67572 rw---   [ anon ]
000000009a300000 1225728       0       0 -----   [ anon ]
00000000e5000000  189440  189440  189440 rw---   [ anon ]
00000000f0900000  252928       0       0 -----   [ anon ]
0000000100000000   12032   11936   11936 rw---   [ anon ]
0000000100bc0000 1036544       0       0 -----   [ anon ]
00007fcf18000000   20976   16856   16856 rw---   [ anon ]
00007fcf1947c000   44560       0       0 -----   [ anon ]
00007fcf20000000   64988   47248   47248 rw---   [ anon ]
00007fcf23f77000     548       0       0 -----   [ anon ]
```

Figure 3.2. Outcome of *pmap* command

RSS: Resident Set Size, is the portion of memory occupied by a process that is held in main memory (RAM)

Dirty: Is memory representing data on disk that has been changed but has not yet been written out to disk.

Mode: Access mode and privileges

Mapping: The user-facing name of the application or library

The last row returns the total in Kilobytes in three columns. The first number is total amount of memory mapped to files, the second is the amount of private address space and the third number is the amount of address space this process is sharing with others. The memory map before running the code and then the memory map during and after running the code was captured, then the difference between the two was computed. Here are the exact commands were used:

- pidof MATLAB
- sudo pmap -x PID | tail -n 1

Memory allocation in MATLAB [27]

Memory allocation in MATLAB is quite sufficient for arrays. It stores the array data into contiguous blocks of memory along with important information of class and, in a header block. Header blocks bring overheads, but it is possible to reduce the number of headers by consolidating large data into smaller ones. Although the overhead is negligible for most arrays, still there could be some benefits.

When elements are removed, MATLAB maintains contiguous storage by compacting the array within its original memory lactation.

Moreover, when new elements are added to an existing array, MATLAB maintains contiguous by finding a large memory block, then copies the array content in the new memory block and deallocates the original memory.

- **Workspace size**

  The *whos* function, returns the number of bytes each variable occupies in the workspace. The following code example was used to return the size of the workplace in Bytes:

  ListOfVariables = whos ;

  NumberOfBytes = 0 ;

  for k = 1:length(ListOfVariables)

  NumberOfBytes = NumberOfBytes + ListOfVariables(k).bytes;

Experimental setup

The computer on which the following numerical results have been obtained was equipped with the processor Corei7, CPU @ 1.80GHz x 8, 8 GB of memory and was running on Linux, Ubuntu 22.04.

3.2 Numerical results

3.2.1 MATLAB performance analysis

In this section, the scalability and performance of the satellite simulation toolbox are under analysis. The goal is to analyze MATLAB satellite toolbox performance in terms of execution time, memory usage and simulation duration, using a scenario in mega-constellation system.

Scenario: Path selection through large satellite constellation

Overview of the scenario

In this scenario, two ground stations, one in Boston in the United States and the other in India, will be connected through satellite constellations with different numbers of satellites. A TLE file is used to add the constellation to the scenario. On the simulation start time, the path will be determined and then we must determine the times over the next hours (which is the simulation time) when the path can be used. After determining the path, gimbals, transmitters, receivers, and antennas are added to the satellites in the path and then we plot the link margin at the receiver antenna. Here is the input and output objects and their properties used in this scenario:

Input objects

A satellite scenario is an object in the toolbox, which is a 3D arena consisting of satellites, ground stations, and the interactions between them. Times are input, including start time, stop time, sample time, and simulation time. All in Universal Time Coordinated (UTC). You can set "auto simulate" to true or false, which is an option to simulate the satellite scenario automatically. If false, the scenario will be running only by calling the "advance" function. In our scenario, at the start time the "auto simulate" is false

to prevent the scenario from advancing automatically through the time steps, when the path is found, it will become true again. Satellites are another input. A satellite is an object with properties such as name, ID, conical sensors, gimbals, and orbit object, etc. To add satellite "satellite" function is used. For large constellations, a TLE file is used.

The Ground station is an input object. Ground station object has properties such as name, ID, latitude, longitude, altitude, minimum elevation angle (values between -90 and 90 degrees, the minimum elevation angle for the satellite to be visible from the ground station, and for the ground station to be visible from the satellite, specified as a scalar or row vector [27]). To add ground station, the "groundStation" function is used. Auto show setup is an option to show the graphics automatically. With the help of the graphics, you can visualize the constellation and the path. Other inputs are gimbals, transmitters, receivers, and antennas.

Output

The outputs are nodes of the path, tables, and link margin plot. After the nodes are determined, you will output the satellites that are included in the path from Gs1 to Gs2. Using the "link" object, which defines a link analysis object, the nodes in the path are linked. Then it returns a table of intervals during which the times when ground station Gs1 can send data to ground station Gs2 via the satellites. The columns of the table are source (first node), target (last node), interval number, start time, end time, duration (in seconds), start orbit (if the source or target are satellite or an object which is connected to the satellite directly or indirectly, start orbit and end orbit are associated to that), end orbit. At the end, the Eb/No in dB is calculated as [27].

$$\begin{aligned} \text{EbNo} = {} & \text{txPower} + \text{txAntennaGain} - \text{txSystemLoss} - \text{pathloss} \\ & + \text{rxAntGaintoNoiseTempRatio} - 10\log_{10}(K) - \text{rxSystemLoss} \\ & - 10\log_{10}(\text{bitRate}) - 60 \end{aligned} \tag{3.1}$$

where:

txPower is the transmitter power in dBW.

txAntennaGain is the transmitter antenna gain in dB.

txSystemLoss is the transmitter system loss in dB.

pathloss is the path loss in dB.

rxAntGaintoNoiseTempRatio is the receiver antenna gain to noise temperature ratio.

K is the Boltzmann constant.

rxSystemLoss is the receiver system loss in dB.

bitRate is the bit rate in Mbps.

The link margin is computed at the receiver, as the difference between "energy per bit to noise power spectral density ratio (Eb/No)" and the "required EbNo". To be sure that the link is closed between GS1 and GS2, the link margin must be positive at all receiver

nodes. The quality of the link depends on the link margin, the higher the link margin, the better the quality.

To determine the link margin at the final node which is ground station 2 receiver, the Eb/No history at the ground station 2 receiver is computed and then subtracted from the required Eb/No from this value to obtain the link margin. Then it can be plotted.

Explanation of the scenario in detail with the parameters used

To explain the scenario steps are used.

- **Step 1: Create satellite scenario**

  Using the "satelliteScenario" function, the scenario will be created. Start time, end time, and sample time are shown in the table shown in Fig 3.3:

Simulation parameters	
Start time	10 December 2021, 6:27:57 PM UTC
simulation duration	1 - 6 - 12 - 18 - 24 hours
sample time	15 s

Figure 3.3. Simulation parameters of the scenario

- **Step 2: Add large constellation of satellites**

  Using a TLE file, different numbers of satellites are added for each experiment, including 10, 20, 40, 80, 100, etc.

- **Step 3: Add ground stations**

  Two fixed ground station are added, the parameters are shown in tables of the Fig 3.4 and Fig 3.5:

- **Step 4: Determine elevation angles of satellites concerning ground stations**

  Why this step is needed? Because for access to exist between **two satellites**, the Line Of Sight (LOS) must exist. For access to exist between **a ground station and a satellite**, the LOS must exist and in addition, the elevation angle of the satellite and the ground station must be greater than the minimum elevation angle of the ground station. So, we need to find the elevation angles of the satellite concerning the source and target ground stations. It is assumed that for the initial routing, the elevation angle of the first satellite in the path with respect to source ground station and the last satellite in the path with respect to target ground station must be at least 30 degrees. Therefore, the elevation angles greater than 30 degrees are retrieved.

- **Step 5: Determine best satellite for initial access to constellation**

parameters of Gs1	
Latitude	42.3001
Longitude	-71.3504
tx at GS1	
frequency	30e9 Hz
power	40 dbW
bitrate	20 Mbps
Antenna at GS1	
type	GaussianAntenna
dishdiameter	5 m

Figure 3.4. Parameters of source ground station

parameters of Gs2	
Latitude	17.4351
Longitude	78.3824
rx at GS2	
requiredEbNo	1 dB
GainToNoiseTemperatureRatio	3 decibels/Kelvin
Antenna at GS2	
type	GaussianAntenna
dishdiameter	5 m

Figure 3.5. Parameters of target ground station

Between all the satellites with elevation angles greater than 30 degrees, the best ones will be chosen. But how? The best satellite needs to satisfy 2 conditions simultaneously.

- Has the elevation angle greater than 30 with respect to source ground station
- Has the closest (minimum) range to target ground station

This satellite will be the first satellite in the path. Thus up to here, two nodes are determined, the source ground station and the first satellite.

- **Step 6: Determine remaining nodes in path to target ground station**

  Next nodes are chosen using similar logic. It needs to satisfy these conditions:

  - Has the elevation angle greater than or equal to -15 degrees concerning the current satellite
  - Has the closest (minimum) range to target ground station

  The elevation value of -15 degrees is chosen because the horizon with respect to each satellite in the constellation is about -21.9813 degrees. This value can be derived by assuming a spherical earth geometry and the fact that these satellites are in near-circular orbits at an altitude of roughly 500 km. Note that the spherical earth assumption is used only for computing the elevation angle of the horizon.

The satellite scenario simulation itself assumes a WGS84 ellipsoid model for the earth [27]. Other satellites will be added continuously to the pass until it reaches a satellite whose elevation angle is at least 30 degrees with respect to the target ground station. This is the last node and the routing is finished. Note that this is not the shortest path between 2 ground stations.

- **Step 7: Determine intervals when calculated path can be used**

  The intervals over the next hours during which the calculated path can be used need to be determined.

- **Step 8: Add gimbals, transmitters, receivers, and antennas to the satellites of the path**

  When the path is determined, for each satellite involved in the path, the gimbals, transmitters, receivers, and antennas are added. Fig 3.6 specifies the parameters.

satellites parameters	
#of satellites	10,20,40,80,100,200,400,800,1000,...
type of satellite	regenerative repeater
rx on the satellites	
GainToNoiseTemperatureRatio	3 decibels/Kelvin
requiredEbNo	4 dB
rx on the satellites	
frequency	27e9 Hz
power	20 dbW
bitrate	20 Mbps
SystemLoss	3
Antenna on satellites rx and tx	
type	GaussianAntenna
dishdiameter	0.5 m
ApertureEfficiency	0.5

Figure 3.6. Satellite parameters of the scenario in the determined path

Experiment results

The scenario is done for different numbers of satellites in the constellation, different simulation times, 1-6-12-18-24 hours, and for each of them, 10 times to see the average time. Note again that the time is not the exact CPU time, but the execution time of executing the code in my PC, we can conclude that CPU time <= minimum execution time calculated using $tic - toc$ function in MATLAB.

Table 3.1 shows the minimum time and average time, when increasing the number of satellites, and with a simulation duration of 1 hour. Time units are in seconds. When the number of satellites increases, the times will increase as well. The number of satellites is increased up to 4000 satellites in the TLE file.

Table 3.2 shows the minimum time and average time, when increasing the number of satellites in a simulation duration of 6 hours.

Simulation duration = 1 hour

number of sats	min time(s)	average time(s)
10	0.82	0.92
20	1.12	1.18
40	1.88	2.70
80	3.66	4.73
100	4.84	6.22
200	8.76	9.82
400	15.30	18.17
800	36.70	41.72
1000	39.56	43.30
1600	84.15	82.53
2000	117.84	108.20
3000	177.16	193.61
4000	315.42	327.56

Table 3.1. Min and avg time for simulation duration = 1 hour

Simulation duration = 6 hours

number of sats	min time(s)	average time(s)
10	2.94	3.92
20	3.42	4.34
40	5.03	6.70
80	9.68	12.02
100	10.89	14.13
200	19.82	22.08
400	38.26	44.81
800	87.25	91.84
1000	95.17	95.64
1600	155.95	170.61
2000	209.94	215.43
3000	344.51	359.79
4000	515.33	531.00

Table 3.2. Min and avg time for simulation duration = 6 hours

Table 3.3 shows the minimum times and average times, when increasing the number of satellites in a simulation duration of 12 hours. Note that similar to other tables, when the number of satellites increases, the times will increase as well.

Simulation duration = 12 hours

number of sats	min time(s)	average time(s)
10	5.37	6.47
20	6.02	7.54
40	9.58	11.46
80	15.22	18.32
100	20.24	21.96
200	31.60	35.042
400	63.59	69.144
800	137.15	143.99
1000	153.21	155.43
1600	230.89	270.20
2000	357.26	369.30
3000	433.26	464.25
4000	707.75	741.00

Table 3.3. Min and avg time for simulation duration = 12 hours

Table 3.4 is similar to the Tables 3.1, 3.2, 3.3. The difference is, the number of satellites could be increased up to 2000 satellites. More than that, The system would run out of memory. This is evident because the simulation duration affects the execution time and memory usage as well as the number of satellites in the constellation.

Simulation duration = 18 hours

number of sats	min time(s)	average time(s)
10	7.51	9.26
20	8.47	10.34
40	12.91	15.00
80	21.55	24.74
100	23.20	29.35
200	42.43	46.74
400	79.59	87.18
800	173.44	179.59
1000	198.24	201.48
1600	338.85	345.59
2000	357.26	369.30
3000	-	-
4000	-	-

Table 3.4. Min and avg time for simulation duration = 18 hours

Simulation duration = 24 hours

number of sats	min time(s)	average time(s)
10	10.9	12.84
20	11.34	13.89
40	15.51	18.91
80	27.59	33.52
100	33.41	36.11
200	57.77	62.13
400	109.12	113.63
800	209.74	220.19
1000	240.8	260.46
1600	400.72	449.69
2000	-	-
3000	-	-
4000	-	-

Table 3.5. Min and avg time for simulation duration = 24 hours

Table 3.5 shows the simulation duration of one day. The threshold to add the satellites to the constellation was 1600 satellites.

Memory usage in simulation duration = 24 hours

number of sats	memory usage(MB)
10	436.77
20	451.12
40	473.91
80	596.46
100	611.21
200	935.46
400	1198.58
800	2034.72
1000	2316.65
1600	3485.20
2000	Out of memory
3000	Out of memory
4000	Out of memory

Table 3.6. Performance evaluation in terms of memory usage

Table 3.6 is different from the previous tables. This table shows the amount of memory usage, calculated in the way explained in Sec 3. What we learn from the data is that, as the number of satellites increases, memory usage grows as well. Note that, with my

system's configuration, and the aforesaid scenario, the threshold to add satellites is 1600. More than that, leads to running out of memory.

When working with a large constellation of satellites, the increased memory usage can be attributed to several factors. Here are some reasons:

- **Data size:** A larger constellation typically involves more data points, and each satellite might have its own set of parameters, such as position, velocity, and communication characteristics. Storing and processing this information for a large number of satellites can lead to increased memory requirements.

- **Simulation complexity:** Simulating the behavior and interactions of a large number of satellites can be computationally intensive. The complexity of the simulation algorithms, especially if they involve detailed physical models, can contribute to higher memory usage.

- **Visualization:** If your simulation includes visualization of the satellite constellation, the graphical representation can consume additional memory. This is especially true if you're dealing with 3D graphics or complex visualizations.

- **MATLAB workspace:** In MATLAB, variables stored in the workspace consume memory. If you are storing information about each satellite as separate variables or if your simulation involves large matrices or arrays, it can significantly increase the memory footprint.

- **Data structures:** The choice of data structures for storing information about each satellite can impact memory usage. For instance, using cell arrays or structures might be more memory-efficient than separate arrays or matrices.

3.2.2 Result graphs

In this section, some graphs will be reported. These graphs are interesting since they show how the execution time and the memory usage of the MATLAB satellite toolbox change when:

- **The number of satellite increases**

- **The simulation duration increases**

Fig 3.7 shows the minimum and average execution time in function of number of satellites, in a simulation duration of 24 hours and with sample time of 15 seconds. The x-axis is the number of satellites and the y-axis is the execution time. We could simulate up to 1600 satellites and greater than this number the PC ran out of memory.

The orange curve shows the average execution time and the blue curve shows the minimum execution time.

What we learn?

As it is shown in the graph, the execution time is proportional to the number of satellites added to the scenario. When we increase the number of satellites by a factor of two, the execution time will grow almost proportionally. Note that again, the CPU time <= minimum execution time.

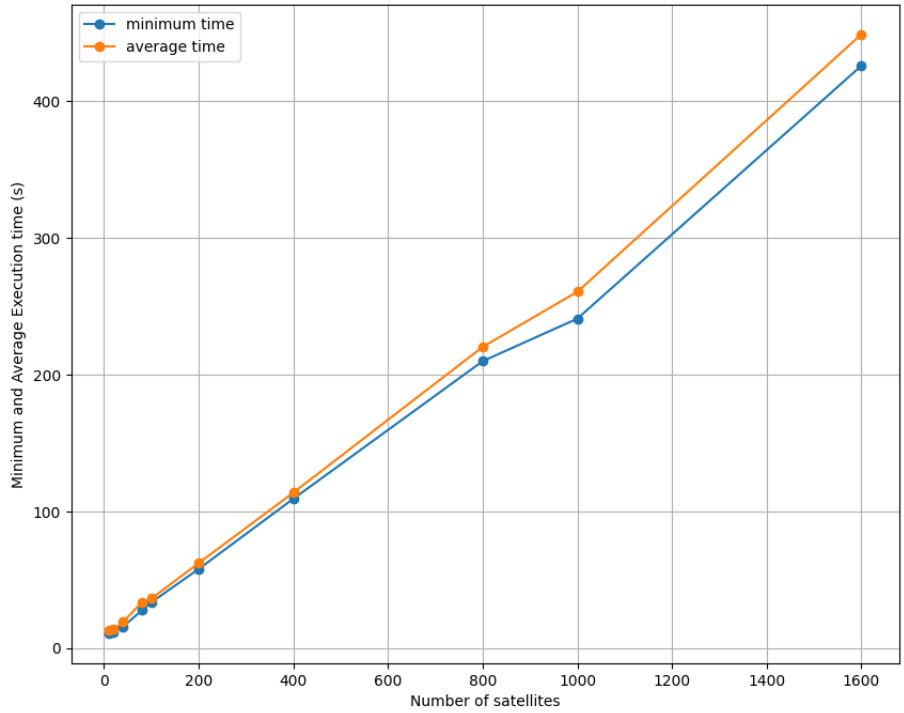


Figure 3.7. Min and avg execution time in function of number of satellites

Fig 3.8 shows the memory usage in MB with respect to the number of satellites. To plot, we started from 100 and we increased the number of satellites up to 1600 which is the saturation point in my PC. The simulation duration is 24 hours and the sample time is 15 seconds.

What we learn?

The x-axis is the number of satellites, the points are important since we tried to double the number of satellites. In this way, it can be easy to estimate what should be the next point, even if is not depicted here. The y-axis is the memory usage, in MB. When the number of satellites increases, memory usage increases as well. Memory is not increasing proportionally.

Fig 3.9 shows the minimum execution time in function of simulation duration for different numbers of satellites. The simulation duration includes 1 hour, 6 hours, 12 hours, 18 hours, and 24 hours. The x-axis is the simulation time(h) and the y-axis is the

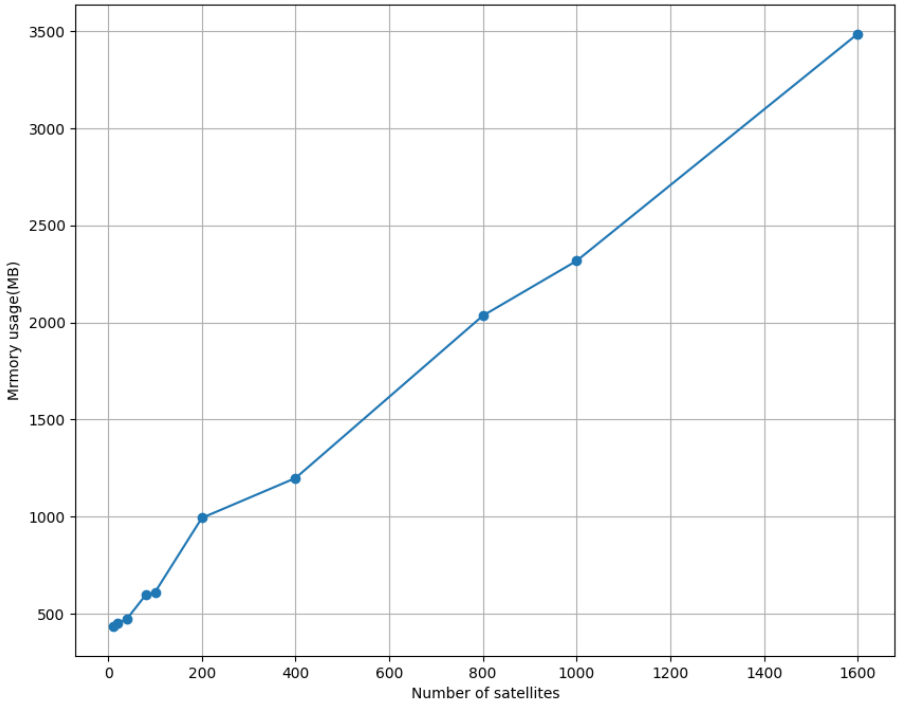


Figure 3.8. Memory usage in function of number of satellites

minimum execution time(s).

What we learn?

The plot shows that the execution time depends on both the number of satellites and the simulation duration. As we increase the number of satellites, the execution time increases. When we increase the simulation duration, the execution time increases as well. Note that the sample time is always the same and equal to 15 seconds. If we change the resolution, it will be similar to increasing the simulation duration, so again it will affect the execution time.

Note that, for greater than 1600 satellites, you can see the curves are missing some points. This is because for bigger amounts of the number of satellites, the system has run out of memory and no results obtained. However, the results can be estimated, based on the behavior of other curves.

To ensure that the link is closed between ground station 1 and ground station 2, we used the linkIntervals() object. This outputs the intervals on which the link is closed. Moreover, we compute the Link margin. Fig 3.10 is the output from linkIntervals() function of my scenario.

Fig 3.11 is the plot for link margin. It is calculated as explained in Formula 3.1. The reason why the plot has this shape is that during 24 hours, the link is closed two times. So there are two curves. The first curve (left side) started from 10-Dec-2021 at

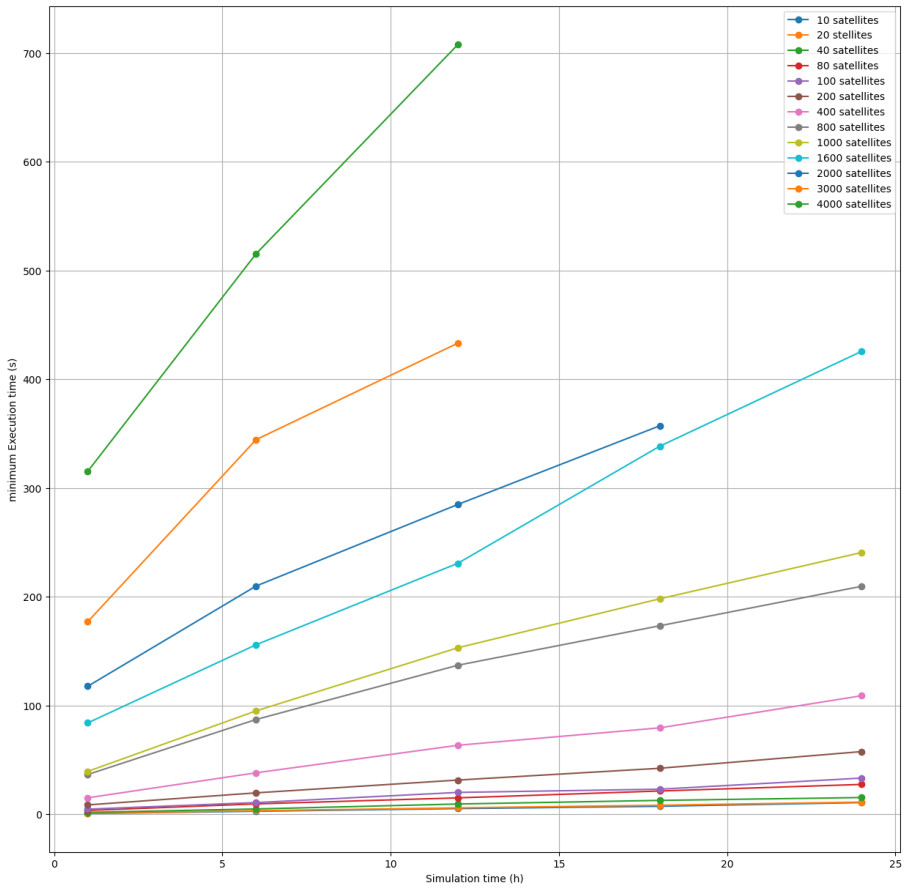


Figure 3.9. Execution time in function of simulation duration of 1 - 6 - 12 - 18 - 24 hours

```
ans =

  2×8 table
```

Source	Target	IntervalNumber	StartTime	EndTime	Duration	StartOrbit	EndOrbit
"Ground Station 1 Transmitter"	"Ground Station 2 Receiver"	1	10-Dec-2021 18:27:57	10-Dec-2021 18:30:27	150	NaN	NaN
"Ground Station 1 Transmitter"	"Ground Station 2 Receiver"	2	10-Dec-2021 20:01:12	10-Dec-2021 20:05:27	255	NaN	NaN

Figure 3.10. Output of linkIntervals() function

18:27:57 until 10-Dec-2021 at 18:30:27 for 150 seconds. Note that the quality of the link depends on the link margin, the higher the link margin, the better the quality. During this interval, the link margin decreases. It is because the satellite is getting further from the ground station, so the distance increases, and the link margin decreases.

Second curve the second interval is from 10-Dec-2021 at 20:01:5712 until 10-Dec-2021 at 20:05:27 for 255 seconds. During this interval, the link margin increases. It is because the satellite is getting closer to the ground station, so the distance decreases, and the link margin increases.

At all other times, the link is not closed, so nothing can be calculated at the receiver

side for the link margin.

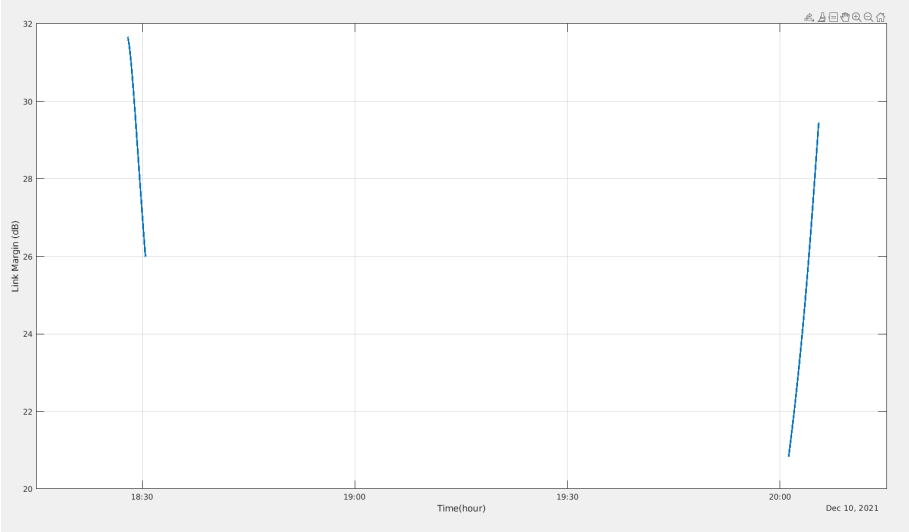


Figure 3.11. Link margin computed at the receiver at ground station 2

Chapter 4

Conclusion

In this thesis a comprehensive summary of existing network simulators used for simulating satellite networks, considering the capability of mega constellation LEO satellite networks is presented. They were categorized in 3 tables in case of their availability, general simulator characteristics, and their use cases.

Moreover, the scalability of the MATLAB satellite simulation toolbox was discussed in a mega-constellation scenario. It was shown that the execution time of the scenario will increase almost proportionally. As expected, memory usage will increase when we increase the number of satellites and the simulation time.

It is essential to emphasize, as per the findings of this study, that there exists no single simulator capable of meeting all modeling requirements. Each simulator is designed for a particular purpose. Therefore, if a new problem arises and the available simulators are inadequate, the options are either to combine multiple simulators or to develop one tailored to the specific needs at hand.

Bibliography

[1] Daniel Havey , Elliot Barlas , Roman Chertov , Kevin Almeroth , Elizabeth Belding. A satellite mobility model for qualnet network simulations. *University of California Santa Barbara Santa Barbara, Ca.*, 2008.

[2] Alabama Civil Aair Patrol United States Air Force Auxiliary Maxwell Air Force Base. *Civil Air Patrol-Satellite Toolkit (Cap-STK) AerspAacEe Program.* Civil Air Patrol Aerospace/STEM, 2010.

[3] O. Alphand , P. Berthou , T. Gayraud , F. Nivor , S. Combes. Platine: Dvb-s/rcs tesbed for next-generation satellite networks. pages 242–243, 2006.

[4] European Telecommunications Standards Institute (ETSI). Digital video broadcasting (dvb); second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications (dvb-s2). 2005.

[5] European Telecommunications Standards Institute (ETSI). Digital video broadcasting (dvb); second generation dvb interactive satellite system (dvb-rcs2); part 2: Lower layers for satellite standard. 2012.

[6] F. Zimmermann, T. Haak, and C. Hill. "galileo system simulation facility". *8th International Workshop on Simulation for European Space Programmes*, October 2004.

[7] http://gpredict.oz9aec.net/documents.php.

[8] http://gpredict.oz9aec.net/index.php.

[9] https://github.com/CNES/opensand.

[10] https://github.com/OpenC3.

[11] https://github.com/snkas/hypatia.

[12] https://github.com/sns3/sns3-satellite.

[13] https://omnetpp.org/.

[14] https://omnetpp.org/download-items/OS3.html.

[15] https://openc3.com/.

[16] https://satellite-ns3.com/sns3.

[17] https://science.nasa.gov/learn/basics-of-space-flight/chapter5-1/.

[18] https://sourceforge.net/projects/gmat/.

[19] https://www.agi.com/products/stk/.

[20] https://www.ansys.com/content/dam/amp/2023/november/asset-creation/stk-capabilities.pdf.

[21] https://www.astronomicalreturns.com/.

[22] https://www.esa.int/.

[23] https://www.gnu.org/licenses/gpl-3.0.html.
[24] https://www.itu.int/en/ITU-R/Pages/default.aspx.
[25] https://www.itu.int/rec/R-REC-P.618/en.
[26] https://www.keysight.com/us/en/assets/3122-1395/technical-overviews/QualNet-Network-Simulator.pdf.
[27] https://www.mathworks.com/help/satcom.
[28] https://www.nsnam.org/.
[29] https://www.sns3.org/doc/satellite.html.
[30] The Network Simulator-ns2, Project web page and source code :http://www.isi.edu/nsnam/ns.
[31] Jani Puttonen(1), Budiarto Herman(1), Sami Rantanen(1), Frans Laakso(1), Janne Kurjenniemi(1). Satellite network simulator 3. *Workshop on Simulation for European Space Programmes (SESP)*, 24-26 March 2015.
[32] Xiangtong Wang , Xiaodong Han , Menglong Yang , Songchen Han , Wei Li. Space networking kit: A novel simulation platform for emerging leo mega-constellations. 2024.
[33] NASA. General mission analysis tool (gmat). *https://go.nasa.gov/3aHhX21*, 2019.
[34] OPNET Technologies, https://opnetprojects.com/.
[35] Michael Wahl Kehla Meacham CJ Rojas. Using systems tool kit to model possible cubesat orbits. *University of Alabama in Huntsville*, April 2022.
[36] Simon Kassing , Debopam Bhattacherjee , Andre Baptista Aguas , Jens Eirik Saethre and Ankit Singla (equal contribution). Exploring the internet from space with hypatia. *nternet Measurement Conference (IMC)*, 2020.
[37] B. Niehoefer , S. Subic and C. Wietfeld. The cni open source satellite simulator based on omnet++. *6th International Workshop on OMNeT++*, March 2013.
[38] A. Pitz, C. Teubert, and B. Wie. Earth-impact probability computation of disrupted asteroid fragments using gmat/stk/codes. *paper AAS*, 2011.