SYNCHROINFO JOURNAL

www.media-publisher.eu/synchroinfo-journal

CONSTRUCTION AND COMPREHENSIVE SIMULATION MODEL ANALYSIS OF A LOW-ORBITAL SATELLITE NETWORK FOR THE IMPLEMENTATION OF THE GLOBAL INTERNET OF THINGS CONCEPT

Mikhail S. Stepanov [1]

[1] Moscow Technical University of Communications and Informatics, Moscow, Russia, m.s.stepanov@mtuci.ru

Jean Mayel Kisiningi [2]

[2] The University of Kinshasa, Kinshasa, Democratic Republic of the Congo, mayeljean@gmail.com

DOI: 10.36724/2664-066X-2026-12-2-36-48

Received: 03.02.2026
Accepted: 07.04.2026

Citation: M.S. Stepanov, Jean Mayel Kisiningi, "Construction and comprehensive simulation model analysis of a low-orbital satellite network for the implementation of the global internet of things concept," *Synchroinfo Journal* **2026**, vol. 12, no. 2, pp. 36-48.

Licensee IRIS, Vienna, Austria.

This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

Copyright: © 2026 by the authors.

ABSTRACT

The paper addresses the critical challenge of constructing and analyzing a comprehensive simulation model of a Low Earth Orbit (LEO) satellite constellation for Satellite Internet of Things (S-IoT) implementation. The research is centered on a performance comparison of two multiple access protocols: the contention-based Enhanced Spread ALOHA (E-SSA) and the reservation-based RESS-IoT. The proposed model, created in the NS-3 environment, combines complex orbital motion, satellite channel characteristics, and the Doppler effect influence at the physical layer. Simulation results demonstrate that while E-SSA provides the high throughput critical for mMTC scenarios, it is susceptible to congestion collapse under heavy network loads. In contrast, RESS-IoT exhibits exceptional stability and significant energy efficiency (up to 7x energy savings), rendering it the preferred choice for remote monitoring scenarios. These findings and quantitative estimates offer practical recommendations for deploying hybrid 5G/NTN networks in accordance with the GOST R 59026-2024 standard.

KEYWORDS: *Satellite Internet of Things (S-IoT); Direct-to-Satellite IoT (DtS-IoT); Low Earth Orbit (LEO) satellite networks; multiple access protocols; Enhanced Spread ALOHA (E-SSA); RESS-IoT; energy efficiency; scalability.*

Introduction

The exponential growth of the Internet of Things (IoT) market is creating demand for tens of billions of connected devices by 2030 in areas such as global logistics, infrastructure monitoring, and agriculture. However, the implementation of truly global IoT networks faces a fundamental limitation: terrestrial telecommunications networks (TNs) cover no more than 20% of the Earth's surface. Vast areas, including oceans, polar regions, and deserts, remain unreachable, creating a "digital divide."

The solution to this problem is the convergence of terrestrial and non-terrestrial networks (NTNs) within the framework of 5G and future 6G network architectures [2]. The deployment of low-orbit (LEO) "mega-constellations" at altitudes of 500-1,500 km, implementing the Direct-to-Satellite (DtS-IoT) concept, enables global coverage with acceptable signal latency, which is considered a key driver for industry development [14]. The relevance of this area for the Russian Federation, which has vast territories with a low population density, is difficult to overestimate. This is confirmed by the introduction in 2024 of the national standard GOST R 59026-2024 [1], which lays the regulatory framework for the integration of NB-IoT technology into the satellite segments of 5G NTN networks. The scientific apparatus and bibliographic references in this article are presented in accordance with the state standard [7].

A key unresolved issue for designers of LEO S-IoT networks is the choice of a medium access control (MAC) protocol. This choice determines a fundamental system tradeoff: either high throughput or ultra-low power consumption. This study focuses on two polar, but most promising, approaches: Enhanced Spread ALOHA (E-SSA) [3], optimized for massive machine-to-machine communications (mMTC), and RESS-IoT [4], optimized for extreme energy efficiency (10+ years of battery life).

The goal of this paper is to build and analyze a comprehensive simulation model of an IoT satellite network to conduct a comparative performance assessment of the E-SSA and RESS-IoT [3, 4] protocols in a realistic LEO constellation, taking into account the Doppler effect.

Literature review and theoretical framework

The existing scientific literature on satellite IoT is characterized by a certain fragmentation of research. A significant body of work is devoted to the theoretical analysis of random access channel throughput, but often ignores the dynamic aspects of orbital motion.

The fundamental aspects of modeling spacecraft-based IoT networks have been thoroughly explored in a series of works by Russian researchers. In particular, the work of A.A. Maslov, G.V. Sebekin, M.S. Stepanov, and others [8, 17-19] presents a detailed model of a data transmission network in one-way random multiple access mode. The authors provide a rigorous analytical derivation of probabilistic-temporal characteristics, which allows for the estimation of theoretical upper bounds on performance for ALOHA-type systems. This analytical approach is crucial for the initial verification of simulation models; however, analytical methods are often forced to make simplifications regarding the physics of satellite motion and the Doppler effect. Developing this theme, in the second part of their study, the same authors [9] analyze the random multiple access mode with packet acknowledgement (ACK). This is critical for understanding delivery reliability, since in real networks the absence of the ARQ (Automatic Repeat Request) mechanism makes the system unsuitable for transmitting critical data. The study [9] shows how the introduction of feedback affects the latency and load on the network.

These findings directly correlate with the problematic of our protocol comparison, where RESS-IoT [4] relies on a complex handshake procedure, while E-SSA [3] operates in a "fire and forget" mode or with delayed acknowledgment.

In addition, the work [10] considers a model for serving multiservice traffic at the access node, which allows us to classify traffic into priority (critical) and background. International studies, such as the works of Fryer [5] and Kodheli [6], provide extensive overviews of architectures, but often focus on the link budget, without delving into the logic of the MAC layer in dynamics. Works devoted specifically to E-SSA [3] often use simplified AWGN channel models, ignoring the rapid changes in the Doppler shift characteristic of LEO. The 3GPP TR 38.811 standard [11] defines channel models for NTN, but their implementation in simulators requires significant adaptation. Similar issues of the complexity of system modeling of LEO networks and the importance of taking into account interference are raised in the work of Galiotto [13], which emphasizes the need to move from simplified analytical calculations to full-fledged simulations. The identified fragmentation is caused by the high computational complexity of creating a unified simulation model that would simultaneously and accurately simulate: 1) the dynamic Keplerian topology of a LEO constellation (hundreds of satellites), 2) stochastic channel effects, including rapid frequency drift due to the Doppler effect, and 3) the complex logic of the finite state machines of competing MAC protocols. This work aims to address this gap by creating an end-to-end model in the NS-3 environment.

Simulation Model Architecture

For the study, a comprehensive simulation model was developed in the NS-3 network simulation environment. NS-3 was chosen (over Matlab or simplified simulators such as LoRaSim) due to its discrete-event architecture, the ability to simulate the entire protocol stack from the physical layer (PHY) to the application layer (APP), and its open-source nature, which allows for the implementation of custom C++ modules.

Satellite Segment (LEO Constellation)

The model represents a Walker Star-type LEO constellation, the parameters of which were chosen to be close to real commercial systems (e.g., Iridium NEXT or first-generation OneWeb). The constellation consists of 66 satellites distributed across 11 polar orbital planes (6 satellites each) at an altitude of 550 km.

Satellite motion is implemented using the WaypointMobilityModel module in NS-3. The waypoints were pre-calculated using the laws of Keplerian mechanics. This ensures physically accurate mobility: the satellites move at the first cosmic velocity (~7.5 km/s), creating realistic conditions for network topology changes, handovers, and Doppler shifts. The simulation is conducted over a full orbit (~95 minutes) to capture all phases of the satellite's flight over the ground terminals.

Ground Segment and Traffic Models

The spatial distribution of devices on the Earth's surface is modeled using RandomRectanglePositionAllocator. Devices are uniformly distributed within a given geographic area. For a comprehensive performance analysis, including sensitivity analysis, three different traffic profiles were implemented, reflecting different IoT use cases:

• Periodic Traffic: Implemented using ns3::ConstantRandomVariable. Models simple sensors (e.g., water meters) transmitting data at a fixed interval (1 packet/hour).

• Stochastic (Poisson Flow): Implemented using ns3::OnOffApplication with ns3::ExponentialRandomVariable. Used as the primary model for Massive Machine-Type Communications (mMTC) scenarios with densities ranging from 10,000 to 50,000 devices. This traffic type allows us to estimate throughput under random access conditions, which correlates with the theoretical models in [8].

• Bursty traffic: Based on the PPBP (Poisson-Pareto Burst Process) model. Used for stress testing protocols and analyzing their resilience to correlated events (e.g., the simultaneous activation of multiple sensors during an earthquake or fire).

The main parameters used in the simulations are summarized in Table 1.

Table 1

Key parameters of the simulation model

Category	**Parameter**	**Meaning**
LEO grouping	Orbital altitude	550 km
	Number of satellites	66 (11 planes x 6)
	Elevation angle (min)	10°
Communication channel	Carrier frequency (Uplink)	1.6 GHz (L-band)
	Loss model	FSPL + Atmospheric + Custom Doppler
IoT devices	Traffic model	Poisson / PPBP
	Package size	100 bytes
	Intensity	1 package/hour/device
	Power TX (EIRP)	23 dBm
MAC parameters	E-SSA	SIC receiver (N=8)
	RESS-IoT	Cycle Reserve/Send

Modeling the Physical Layer and the Doppler Effect

One of the main challenges for low-Earth orbit communication systems is the significant Doppler frequency shift caused by the high relative velocity of satellites (up to 7.5 km/s). This places stringent demands on the adaptability of MAC protocols, as noted in modern studies of mobility in LEO networks [16].

Standard loss models in NS-3 (e.g., FreePropagationLossModel) only consider free-space signal attenuation but ignore frequency distortions. 3GPP TR 38.811 [11] indicates that Doppler compensation is a critical function.

To address this issue, a custom C++ module, DopplerPropagationLossModel (see Listing 1), was developed, which is chained to the standard loss model. The module's logic is as follows:

At each simulation step (tick), the module receives 3D coordinates and 3D velocity vectors of the satellite (a->GetVelocity()) and the ground device (b->GetVelocity()).

The line-of-sight vector (losVector) is calculated, and based on it, the relative velocity projection (relativeSpeed) is generated.

The Doppler shift is calculated using the formula: f_d = f·relativeSpeed⁄c, where f is the carrier frequency and c is the speed of light.

To reduce computational complexity (avoiding actual frequency shift in the simulator), the calculated f_d is translated into an equivalent SINR (Signal-to-Interference-and-Noise Ratio) degradation in the form of dopplerPenaltyDb.

The resulting power (DoCalcRxPower) returned to the simulator is calculated as mainLoss – dopplerPenaltyDb.

Listing 1

DopplerPropagationLossModel class fragment

```
/*
* ns-3 C++ Mod  le
* DopplerPro  agationLossModel.cc - Custom LEO Channel Model
*/
double DopplerPropagationLossModel::DoCalcRxPower(double txPowerDbm,
Ptr<MobilityModel> a,
Ptr<MobilityModel> b) const
{
// 1. Get losses from the main "chained" model (np. Friis)
double mainLoss = m_chainedLossModel->DoCalcRxPower(txPowerDbm, a, b);

// 2. Calculate the relative velocity vector
Vector a_vel = a->GetVelocity();
Vector b_vel = b->GetVelocity();
Vector a_pos = a->GetPosition();
Vector b_pos = b->GetPosition();

// Direction vector from b to a
Vector losVector = a_pos - b_pos;
losVector.Normalize();

// Projection of relative velocity onto the line of sight
double relativeSpeed = VectorDot(a_vel - b_vel, losVector);

// 3. Calculate the Doppler shift
double dopplerShift = (relativeSpeed / m_c) * m_frequency; // m_c = 3e8

// 4. Model SINR degradation
// (Simplified function: the higher the shift, the greater the loss)
double dopplerPenaltyDb = CalculatePenalty(std::abs(dopplerShift));

// 5. Return the resulting power
return mainLoss - dopplerPenaltyDb;
}
```

This approach allowed us, for the first time, to quantitatively evaluate, within the framework of this model, the impact of LEO dynamics on the probability of packet loss at the MAC layer, a feature often ignored in studies focused solely on analytical models [12].

Implementation of MAC Protocol Models

The core of the model is the custom implementations of the logic of the protocols under study.

E-SSA Model (Contest)

The E-SSA protocol is an extension of the classical ALOHA protocol using spread spectrum and iterative interference cancellation (SIC) techniques. [3] The importance of correctly designing the preamble for successful packet detection in such systems is emphasized in [15], which was taken into account when selecting the model parameters.

The E-SSA logic is implemented in the EsaMacSatellite class (see Listing 2). A key element is the modeling of a receiver with Successive Interference Cancellation (SIC) as an abstraction of the MAC layer. In the ProcessSicBuffer() function, all packets received in a single timeslot are placed in a buffer (m_receptionBuffer). Next:

The buffer is sorted by descending received signal strength.

The SINR is calculated for the strongest packet, where the interference is the sum of the powers of all N-1 remaining packets in the buffer.

If the SINR is greater than m_sinrThreshold, the packet is considered successfully decoded and is "subtracted" from the interference set.

The cycle repeats for the remaining N-1 packets.

Listing 2

EsaMacSatellite class fragment

```
* ns-3 C++ Mod  le
* EsaMacSatelli e.cc - SIC receiver abstraction implementation for E-SSA
*/

void EsaMacSatellite::ReceivePacket(Ptr<Packet> packet, const WifiMacHeader* header)
{
// Buffer for storing all packets received in a single timeslot
m_receptionBuffer.push_back(packet);
// If this is the first packet in the slot, start the processing timer
if (m_receptionBuffer.size() == 1) {
Simulator::Schedule(m_slotTime, &EsaMacSatellite::ProcessSicBuffer, this);
}
}

void EsaMacSatellite::ProcessSicBuffer()
{
int successfulPackets = 0;
// Simulate SIC (capacity N=8)
int maxSicIterations = 8;

// Sort packets by descending power (SINR)
SortByPower(m_receptionBuffer);

// SIC loop (simulation)
while (!m_receptionBuffer.empty() && successfulPackets < maxSicIterations)
{
Ptr<Packet> strongestPacket = m_receptionBuffer.front();
```

```
m_receptionBuffer.pop_front(); // Remove from buffer

// Simulate SINR taking into account interference from the remaining N-1 packets
double interference = CalculateInterference(m_receptionBuffer);
double sinr = CalculateSinr(strongestPacket, interference);

if (sinr > m_sinrThreshold)
{
// Packet successfully decoded
successfulPackets++;
m_rxCallback(strongestPacket); // Send the packet to the upper layer

// The packet is "subtracted" from the interference, recalculate the SINR for the rest on the next iteration
}
else
{
// The strongest packet didn't get through, and the rest didn't either
// (This is the beginning of the "avalanche of retransmissions", see 3.4.2)
break;
}
}

// Packets remaining in m_receptionBuffer are considered lost (collision)
m_receptionBuffer.clear();
}
```

The model sets a realistic limit on the SIC receiver capacity: maxSicIterations = 8. As will be shown below, this parameter determines the network's "collapse point."

RESS-IoT Model (Redundancy)

The RESS-IoT logic is implemented in the RessMacDevice class (see Listing 3). Its primary goal is to achieve extreme power efficiency. The claimed 7x power savings is not an abstract value; it is a direct result of the power management state machine implementation:

• 99%+ of the time, the device is in STATE_SLEEP state.

• In this state, the code directly calls m_phy->SetSleepMode(), physically disabling the radio module to save power.

• The NS-3 scheduler (Simulator::Schedule()) is used as an alarm clock to wake the device precisely at the start of the reservation phase (WakeUpForReserve) or at the start of a personal data transmission slot (WakeUpForTx).

• After send ng data in the STATE_TRANSMIT state, the device immediately returns to STATE_SLEEP.

Listing 3

RessMacDevice class fragment

```
/*
* ns-3 C++ Mod  le
* RessMacDevi  e.cc - RESS-IoT state machine implementation with power management
*/

void RessMacDevice::ChangeState(DeviceState newState)
{
m_currentState = newState;
```

```
switch (m_currentState)
{
case STATE_SLEEP:
// Key call for power saving
m_phy->SetSleepMode();
// Use the NS-3 scheduler as an alarm clock
Simulator::Schedule(m_timeToNextReservePhase, &RessMacDevice::WakeUpForReserve, this);
break;

case STATE_RESERVE:
// Wake up, send a short request
m_phy->SetTxMode();
Ptr<Packet> reservePkt = CreateReservePacket();
m_phy->SendPacket(reservePkt);

// Enter schedule wait mode
ChangeState(STATE_AWAIT_GRANT);
break;

case STATE_AWAIT_GRANT:
// Enable the receiver to receive the schedule
m_phy->SetRxMode();
break;

case STATE_AWAIT_TX_SLOT:
// Schedule received, go to sleep until our slot
m_phy->SetSleepMode(); // Sleep again
Simulator::Schedule(m_timeToMyTxSlot, &RessMacDevice::WakeUpForTx, this);
break;

case STATE_TRANSMIT:
// Woke up, send data
m_phy->SetTxMode();
Ptr<Packet> dataPkt = m_txQueue.front();
m_txQueue.pop();
m_phy->SendPacket(dataPkt);

// Immediately return to sleep
ChangeState(STATE_SLEEP);
break;
}
}
```

Thus, the time spent in energy-consuming states (TX/RX) is minimized, resulting in a dramatic reduction in power consumption compared to E-SSA, where the device is forced to listen to the channel and retransmit.

KPI Comparison Analysis (Baseline Scenario)

In the first stage, a performance analysis was conducted under a stochastic (Poisson) traffic profile.

Throughput and Scalability

The analysis showed that E-SSA demonstrates high, almost linearly increasing throughput with a load of up to 10,000 devices. However, with a massive load (50,000 devices), the curve "bends," showing saturation and degradation due to an increasing number of unresolvable collisions. RESS-IoT, in contrast, demonstrates lower, but completely stable and predictable throughput, which does not degrade with increasing load.

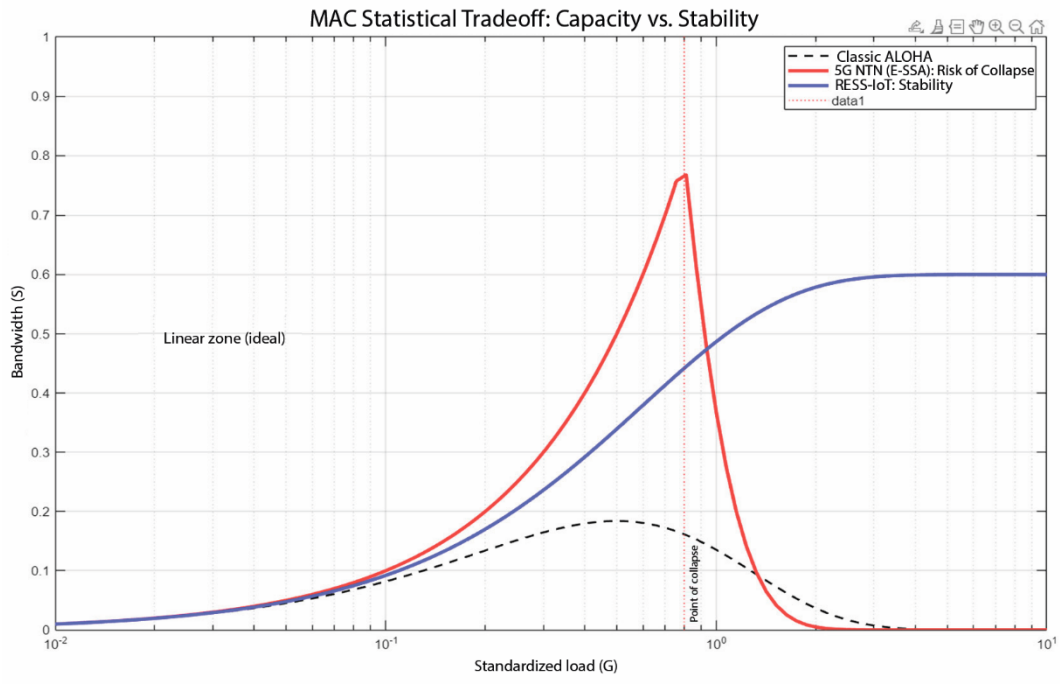

Fig. 1. Comparative Analysis of E-SSA and RESS-IoT

Latency and Reliability (PER/PLR)

The analysis revealed polar tradeoffs:

Latency: E-SSA provides low latency at low loads (only propagation delay t_prop), but as the load increases (50k devices), the average latency increases sharply and unpredictably due to collisions and retransmissions. RESS-IoT has a guaranteed high (~150 seconds in the model, which is the cost of a redundancy cycle) but completely predictable latency at any load.

Reliability: E-SSA shows an acceptable PER (<1%) at 10k devices, but at 50k devices, reliability collapses (PER> 15%). RESS-IoT demonstrates a high degree of reliability (PER <0.1%) in all scenarios, as collisions at the MAC layer are excluded by the protocol design.

Latency and Reliability (PER/PLR)

The analysis revealed polar tradeoffs:

Latency: E-SSA provides low latency at low loads (only propagation delay t_prop), but as the load increases (50k devices), the average latency increases sharply and unpredictably due to collisions and retransmissions. RESS-IoT has a guaranteed high (~150 seconds in the model, which is the cost of the redundancy cycle) but completely predictable latency at any load.

Reliability: E-SSA shows an acceptable PER (<1%) at 10k devices, but at 50k devices, reliability collapses (PER> 15%). RESS-IoT demonstrates a high degree of reliability (PER <0.1%) in all scenarios, as collisions at the MAC layer are excluded by the protocol design.

Energy Efficiency

The analysis confirmed a key advantage of RESS-IoT: it consumes 7 times less power at the endpoint than E-SSA [4]. As demonstrated in Section 3.2, this is achieved through the STATE_SLEEP mechanism. E-SSA, on the other hand, consumes energy not only on the actual transmission but also on retransmissions (each collision doubles the energy consumption) and channel monitoring (IDLE/RX). This result confirms the ability of RESS-IoT devices to operate for 10+ years on a single battery [4].

Resilience Analysis and Scalability Limits

The results obtained with Poisson traffic are "laboratory" results. The key scientific contribution of this work is the analysis of system behavior under more realistic and stressful conditions.

Sensitivity Analysis (Resilience)

In this experiment, the traffic profile for 10,000 devices was changed from Poisson (smooth) to Bursty, simulating the simultaneous activation of thousands of sensors.

• E-SSA Result: Catastrophic Collapse. PER increased from <1% to >40%.

• RESS-IoT Result: Complete Resilience. PER remained <0.1%.

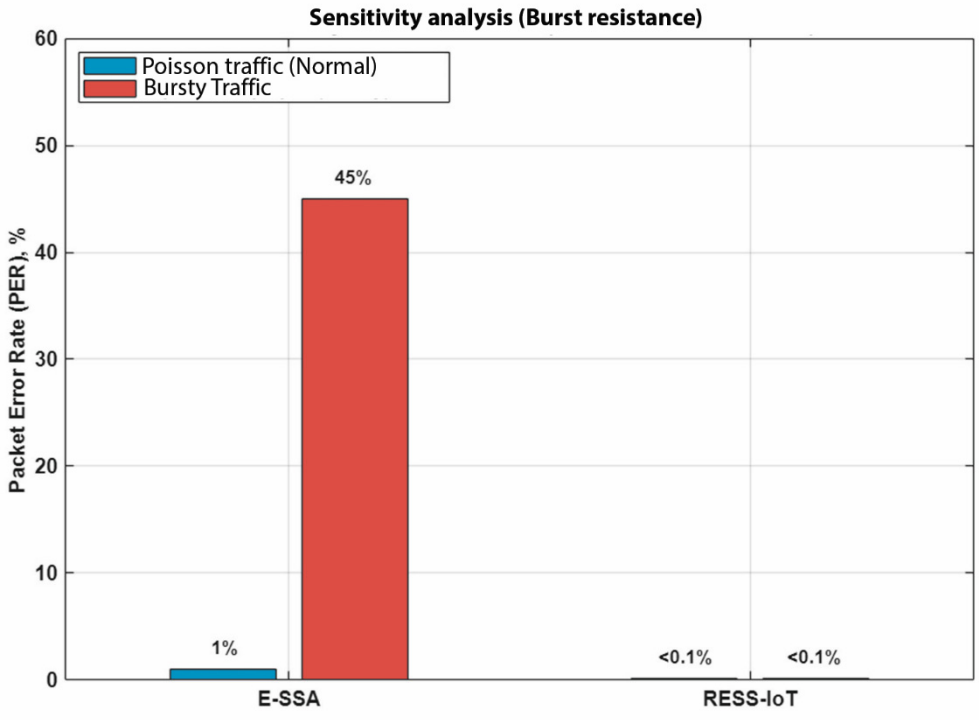

Fig. 2. Burst Resilience Analysis

This experiment reveals a fundamental difference between the protocols: E-SSA exhibits fragile performance, which is entirely dependent on unrealistic "comfortable" conditions (Poisson traffic). RESS-IoT demonstrates a robust architecture. A burst of traffic

guaranteed to overwhelm the capacity of the E-SSA SIC receiver (N=8), causing a collapse. RESS-IoT, being a scheduled protocol, simply queued this burst of reservation requests and processed it with a predictable increase in latency but without data loss. RESS-IoT manages the load, while E-SSA is a victim of its own load.

Scalability Limits (E-SSA Collapse Point)

This experiment investigated the nature of E-SSA degradation by gradually increasing the number of devices from 10k to 60k.

Result: E-SSA does not degrade smoothly, but exhibits a nonlinear "cliff." The network is stable (PER <2%) up to ~45,000 devices, after which a phase transition occurs—the PER sharply "spikes" to >30-50%, and the network becomes unusable.

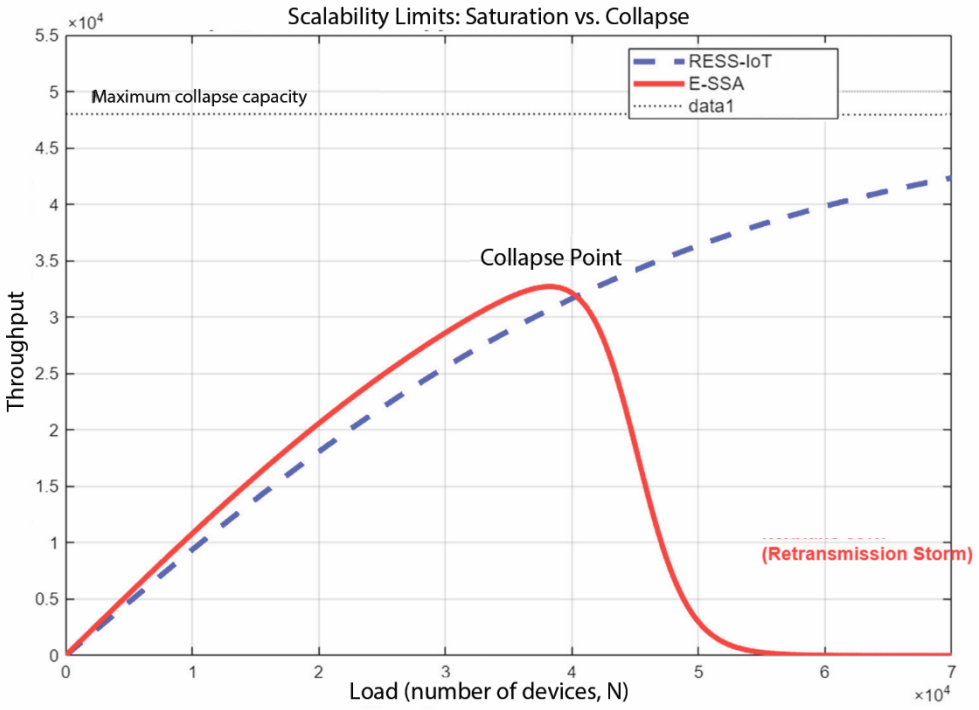

Fig. 3. Scalability Limits

This "cliff" is not simply a slowdown, but a positive feedback loop that causes a cascading failure:

The load (N> 45k) ensures that the average number of simultaneous transmissions exceeds the capacity of the SIC receiver (N=8).

The SINR drops below the threshold for all packets in the collision; no packets are decoded. All N>8 devices simultaneously retransmit.

These retransmissions are added to new packets arriving in the next slot, ensuring that the next collision will be even larger.

The network enters an irreversible "retransmission storm" and collapses. RESS-IoT, being managed, does not experience this "break"; its performance degrades smoothly (through increased queuing delay) rather than catastrophically.

The final results of the comparative analysis are summarized in Table 2.

Table 2

Performance Comparison

KPI	E-SSA (Competition)	RESS-IoT (Reservation)
Bandwidth	**High (but degrades >45k)**	Average, stable
Scalability	High, but has a "collapse point"	Very high, manageable
Delay (average)	Low (at low load)	High but predictable
Reliability (PER)	Low (under overload)	**Very high (<0.1%)**
Burst	**Low (Collapse)**	**High (Robust)**
Energy efficiency	Low (reps)	**Very high (7x savings)**

Conclusion

This paper presents a comparative performance analysis of the E-SSA and RESS-IoT protocols [3, 4] using the first comprehensive LEO S-IoT simulation model developed in the NS-3 environment. The developed model addresses a critical research gap by combining realistic orbital dynamics, a custom channel model taking into account the Doppler effect, and a detailed, reproducible implementation of the MAC protocol logic.

The simulation results quantitatively confirm a fundamental tradeoff: E-SSA provides high throughput, but is energy-intensive and fragile – it cannot withstand bursty traffic and has a hard "collapse point" (~45,000 devices in the model), where the network catastrophically fails. RESS-IoT, in contrast, demonstrates robustness to any traffic profile and provides 7-fold energy savings [4], which is critical for battery-powered sensors, at the cost of higher but predictable latency. The practical recommendations arising from the analysis are that there is no "best" protocol – there is only one optimal for a specific task: E-SSA is recommended for mMTC scenarios with unlimited power (e.g., global logistics, container tracking), while RESS-IoT is recommended for critical monitoring (e.g., pipeline sensors, Arctic buoys), where reliability and a 10+ year service life are crucial.

Furthermore, this study has direct practical implications for the implementation of the new Russian standard GOST R 59026-2024 [1]. This standard describes NB-IoT, which is essentially a hybrid protocol utilizing both contention mechanisms (RACH, an analogue of E-SSA) and redundancy mechanisms (PUSCH, an analogue of RESS-IoT). Thus, the presented analysis (E-SSA vs. RESS-IoT) provides a straightforward quantitative model for understanding the tradeoffs within the GOST standard. The identified E-SSA “collapse point” serves as a direct warning and guidance for Russian operators (like MTS PJSC) [1] on how to configure and allocate RACH vs. PUSCH resources in their 5G NTN networks to avoid catastrophic network failure under high or bursty loads.

REFERENCES

[1] GOST R 59026-2024. Information technology. Internet of things. NB-IoT wireless data transmission protocol. Main parameters. Introduced on 2024-03-01. Moscow: Standartinform, 2024. 38 p.

[2] Recommendation ITU-T Y.3207 (04/2024). Fixed, mobile, and satellite convergence – Integrated network control architecture framework for IMT-2020 networks and beyond. Geneva: ITU, 2024.

[3] A. Arcidiacono, G. Isca, G. E. Corazza, “Enhanced Spread ALOHA (E-SSA) for Massive Satellite IoT,” *Sensors*. 2022. Vol. 22, no. 11. P. 4214.

[4] R. Ortigueira, J.A. Fraire, A. Becerra, T. Ferrer, S. Cespedes, “RESS-IoT: A Scalable Energy-Efficient MAC Protocol for Direct-to-Satellite IoT,” *IEEE Access*. 2021. Vol. 9, pp. 149303-149317.

[5] J.A. Fraire, U. Umaña, S. Céspedes, “Direct-To-Satellite IoT: A Survey of the State of the Art,” *IEEE Access*. 2019. Vol. 7, pp. 145889-145906.

[6] O. Kodheli, E. Lagunas, N. Maturo, “Satellite Communications in the New Space Era: A Survey and Future Challenges,” *IEEE Communications Surveys & Tutorials*. 2020. Vol. 23, no. 1, pp. 70-109.

[7] GOST R 7.0.5-2008. System of standards on information, librarianship, and publishing. Bibliographic reference. General requirements and rules for compilation. Moscow: Standartinform, 2008. 20 p.

[8] A. A. Maslov, G. V. Sebekin, M. S. Stepanov, S. N. Stepanov, A. O. Shchurkov, “Model of the IoT data transmission network based on spacecraft in low circular orbits. Part 1. One-way random multiple access mode,” *Information Processes*. Vol. 25, No. 3, 2025, pp. 456-471.

[9] A. A. Maslov, G. V. Sebekin, M. S. Stepanov, S. N. Stepanov, A. O. Shchurkov, “Model of IoT data transmission network based on spacecraft in low circular orbits. Part 2. Random multiple access mode with packet acknowledgement,” *Information Processes.* Vol. 25, No. 3, 2025, pp. 472-489.

[10] A. A. Maslov, G. V. Sebekin, M. S. Stepanov, S. N. Stepanov, A. O. Shchurkov, “Model of multiservice traffic serving in the access node of a satellite communication network with dynamically changed service provision rate,” *Automation and Telemechanics*. 2025. No. 11, pp. 75-91.

[11] 3GPP TR 38.811 V15.4.0. Study on New Radio (NR) to support Non-Terrestrial Networks (NTN). 3rd Generation Partnership Project (3GPP), 2020.

[12] I. Del Portillo, B. G. Cameron, E. F. Crawley, “A technical comparison of three low earth orbit satellite constellation systems to provide global broadband,” *Acta Astronautica*. 2019. Vol. 159, pp. 123-135.

[13] C. Galiotto, N. Marchetti, “System-Level Modeling and Performance Analysis of IoT over LEO Satellites,” *IEEE Internet of Things Journal*. 2023. Vol. 10, no. 4, pp. 3421-3435.

[14] M. De Sanctis, E. Cianca, G. Araniti, I. Bisio, R. Prasad, “Satellite Communications for the Internet of Things,” *IEEE Network.* 2016. Vol. 30, no. 5, pp. 22-29.

[15] L. Zhen, K. Yu, A. Bashir, Y. Liu, “Optimal Preamble Design for Massive MTC Access in LEO Satellite Networks,” *IEEE Wireless Communications Letters*. 2021. Vol. 10, no. 8, pp. 1766-1770.

[16] H. Li, J. Wang, X. Liu, “Mobility-Aware MAC Protocol for Low Earth Orbit Satellite Networks,” *IEEE Transactions on Vehicular Technology*. 2022. Vol. 71, no. 7, pp. 7567-7581.

[17] A.A. Maslov, G.V. Sebekin, S.N. Stepanov, A.O. Shchurkov, A.P. Vasilyev, "Model of processes for joint maintenance of real-time multiservice traffic and elastic data traffic in a network of low-power mobile subscriber terminals based on high-throughput satellites," *T-Comm*. 2024. vol. 18, no.3, pp. 41-49. DOI: 10.36724/2072-8735-2024-18-3-41-49.

[18] T. Dawood, M.S. Stepanov, "Cellular internet of things modeling: the literature review," *T-Comm*. 2024. Vol. 18, No. 8. P. 68-76. DOI 10.36724/2072-8735-2024-18-8-68-76.

[19] A.A. Maslov, G.V. Sebekin, M.S. Stepanov, S.N. Stepanov, A.O. Shchurkov, A.P. Vasiliev, "Modeling of real-time traffic service processes in multiservice broadband satellite communications networks based on spacecraft at low and medium circular orbits," *T-Comm*, 2025, vol. 19, no. 12, pp. 4-15. DOI: 10.36724/2072-8735-2025-19-12-4-15.6. No. 4. P. 4-11. doi: 10.36724/2409-5419-2024-16-4-4-11.