New Security Paradigms for Spacecraft and Networks: Metrics, Testbeds, and Scalable Solutions

Alexander L. Kedrowitsch

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science & Application

Danfeng (Daphne) Yao, Chair
Jonathan T. Black, Co-chair
Jin-Hee Cho
Wenjie Xiong
Samuel Jero

May 12, 2025
Blacksburg, Virginia

Keywords: Space System Security, Space Networks, Resilient Networks, Security Testbeds, Space-Specific Algorithms, Energy-Efficient Security, Scalable Solutions

New Security Paradigms for Spacecraft and Networks: Metrics, Testbeds, and Scalable Solutions

Alexander L. Kedrowitsch

(ABSTRACT)

The rapid expansion of commercial spaceflight, driven by reduced launch costs and increased private investment, has transformed the landscape of space-based communications. Proposed mega-constellations comprising thousands of satellites promise global connectivity, linking directly to handheld devices on Earth. However, as government and commercial actors push further into space, the emerging heterogeneous networks that interconnect these systems face significant challenges in security and resilience—areas often discussed but infrequently demonstrated. This dissertation addresses these challenges through four primary contributions. First, it presents a novel logical topology for mega-constellations that enables scalable and resilient dynamic routing, significantly reducing network and computational overhead in large-scale satellite networks. Second, it introduces techniques for enabling robust, secure communication on energy-harvesting spacecraft, balancing security requirements with constrained power budgets. Additionally, this work evaluates the suitability of NIST-certified encryption algorithms for deep-space platforms, ensuring compliance with established standards while considering space-specific constraints. Third, it presents several holistic frameworks for systematic performance evaluations of dynamic network routing resilience and intermittent cryptography in space environments. Lastly, the dissertation describes contributions made to several open-source network emulators, enhancing their utility for future space networking research.

New Security Paradigms for Spacecraft and Networks: Metrics, Testbeds, and Scalable Solutions

Alexander L. Kedrowitsch

(GENERAL AUDIENCE ABSTRACT)

Space exploration and satellite technology are undergoing a dramatic shift. With lower launch costs and surging private investment, we now see vast constellations of satellites orbiting Earth, providing internet access and other services directly to the devices in our hands. Meanwhile, companies are planning ambitious missions like asteroid mining and interplanetary exploration. This wave of innovation brings tremendous opportunity—but also new risks. As these space networks grow and become more interconnected, ensuring their security and resilience becomes essential. This research explores how to keep these systems running smoothly and safely, even as they face challenges from limited power supplies, long distances, and potential cyber threats. We investigate ways to improve satellite network designs, making them more robust and efficient, and explore how encryption methods can protect sensitive data traveling across space. We also contribute tools and technologies to help other researchers study and improve these systems, working toward a future where space-based services are as secure and reliable as those on Earth.

Dedication

To Melissa,

For your unwavering love, patience, and encouragement. Your belief in me has pushed me to achieve more than I ever thought possible.

To Alexis and Lilly,

For your endless love, understanding, and the joy you bring to my life. You are my greatest inspirations.

This work is a testament to the strength of family and the support that makes all things possible. Any success I claim from this journey is not mine alone, but the result of shared dedication, resilience, and love.

Acknowledgments

I would like to express my deepest gratitude to those who have supported me throughout my doctoral journey. First and foremost, I extend my sincere thanks to my co-advisors, Dr. Daphne Yao and Dr. Jonathan Black, for their invaluable guidance and mentorship. Dr. Yao has been a source of unwavering support, encouragement, and keen insight since my Master's studies; her expertise and guidance has profoundly shaped my academic growth. I am also deeply grateful to Dr. Black, whose mentorship and perspective was instrumental as I transitioned my research focus toward space network security. I am thankful to my dissertation committee members—Dr. Jin-Hee Cho, Dr. Wenjie Xiong, and Dr. Samuel Jero—for their thoughtful feedback, encouragement, and commitment of time that have enriched this work.

I also wish to acknowledge the faculty and students of Space@VT (The Center for Space Science and Engineering Research), whose collaboration and engagement contributed greatly to my research experience. In particular, I am grateful to Dr. Samantha Kenyon for her insights, and to the fellow students of Space@VT for their camaraderie and support on various projects. I gratefully acknowledge the financial support provided by the Office of Naval Research, the National Science Foundation, and the Walts Fellowship Award, which made this research possible. Finally, I am deeply thankful to my family and friends for their unwavering support and encouragement throughout this journey.

Contents

List of Figures

List of Tables

List of Abbreviations

3GPP 3rd Generation Partnership Project

BP Bundle Protocol

BPSec Bundle Protocol Security

CGR Contact Graph Routing

DTE direct-to-Earth

DTN Delay/Disruption Tolerant Network

FPGA Field-Programmable Gate Array

GEO Geostationary or Geosynchronous Earth Orbit

IC Intermittent Computing

IoT Internet of Things

ISL Inter-Satellite Link

JPL Jet Propulsion Laboratory

LCT Lunar Communication Terminal

LEO Low Earth Orbit

LNSP LunaNet service provider

LTP Licklider Transmission Protocol

MEO Medium Earth Orbit

NASA National Aeronautics and Space Administration

NIST National Institute of Standards and Technology

NTN Non-terrestrial networks

NVM non-volatile memory

PKI Public Key Infrastructure

TLE Two-Line Element

TTL Time-to-live

xGEO Beyond Geostationary Earth Orbit

ZKP Zero-Knowledge Proof

TLE is the abbreviation for 'Two-Line Element', a standardized format for data describing the orbit of an Earth-orbiting object.

xGEO is a new term coined by officials to refer to the region of space beyond Geostationary Earth Orbit (GEO).

3GPP is a group of organizations that develop and establish protocols for wireless, mobile communication

LEO is used to label a class of Earth orbits at altitudes below 1,000 - 2,000 km.

MEO is used to label a class of Earth orbits with altitudes between 2,000 and 36,000 km.

GEO is used to label a class of Earth orbits with periods that match Earth's rotational period.

ISLs are a new technology that allow satellites in space to directly communicate or pass data between each other.

NASA is the lead civilian government organization that is responsible for science and technology related to air and space.

NIST is the lead civilian government organization that is responsible for developing technology standards used in government and commercial industries.

JPL is a government funded research and development center under the direction of NASA.

DTN is a class of networks whose normal operating conditions include extreme latencies and frequent link interruptions. For the purposes of this work DTN refers exclusively to deep space DTNs.

BP is a higher-level protocol developed to operate within the conditions of DTN. Uses 'store-and-forward' mechanism to relay message data between nodes to reach its destination. The current version of the Bundle Protocol is version 7.

LTP is a lower-layer protocol that combines aspects of UDP and TCP network protocols for use on DTNs. A portion of the LTP traffic can be designated as verified-delivery and the rest of the traffic is sent as 'best-effort'.

PKI is the term used to describe a form of public asymmetric key management, where public keys are distributed via digital certifications to verify registered user identity.

IoT is the term applied to miniature electronics that typically operate as sensors without overt human interfaces. IoT devices rely on network connectivity to receive the data and may be deployed in large numbers. Sensed data is transmitted to a network and received by sever for consolidation and processing.

BPSec refers to the publication outlining the mechanisms of Bundle Protocol security, as established by Bundle Protocol version 7.

FPGA refers to a type of configurable integrated circuit whose logic gate structure is mal-

leable for large numbers of configuration changes, allowing developers to customize a processor's layout and configuration to suite their needs.

DTE refers to message transmissions originating away from Earth but sent in the proper direction and with sufficient energy to be received by an Earth-born recipient.

LNSP describes an orbiting component of NASA's LunaNet architecture; LNSPs are intended to communicate with end-user devices to provide resource access.

LCT describes a surface component of NASA's LunaNet architecture; LCTs operate as relays between mobile and vehicular hosts and orbiting or DTE communication links.

CGR describes the routing method derived by NASA for advising DTN nodes on where to route bundles based on the target destination and the current time. CGRs are the de facto standard for DTN and BP routing.

ZKP are a form of mathematical proofs that allow recipients to verify the integrity of a received message by the sender demonstrating knowledge of a shared secret without actually revealing the secret itself. Zero-knowledge is used to describe the lack of preparatory actions such as distributing a shared-secret.

TTL is a parameter used to indicate how long a packet of data should be attempted for delivery before discarding it as undelivered.

IC is used to describe a program execution paradigm where computation work is performed over the course of several power cycles.

NVM is the term that describes a computers long-term memory, where data is retained even in the event of a power loss.

NTN is the broad term used to describe networks that have some or all portions operating in space.

Chapter 1

Introduction

Space is the most hostile operating environment currently known and places significant constraints on computer hardware. Historically, operators in this domain have primarily consisted of government entities and the large aerospace industry contractors those governments hire. However, recent commercial access has lowered the bar to environments beyond Earth's atmosphere and created an exciting age of new frontiers for computation in space. Lower cost and more ready access are changing the landscape of space platforms away from only a handful of massive government contractors to a multitude of smaller vendors, each trying to break into the economic growth projected to be twice the current global GDP [47].

Many operating conditions taken for granted on terrestrial systems and networks are highly modified or simply absent in space. Out of all regions in space, the near-Earth environment is the cheapest and most readily-accessible, and benefits from some natural shielding. However, the further from Earth a spacecraft is designed to operate, the greater the cost of access [121], along with increased exposure to the harmful effects of the space environment [114]. As such, the challenges and constraints placed on computing in the space domain are non-uniform and depend heavily on the conditions of the target environment.

Despite nearly a quarter-century of government and military think-tanks using the phrase "Space Pearl Harbor" to describe a possible surprise attack against critical space infrastructure [123], security remains an oft-touted priority with actual implementations severely lacking and much less straightforward [41, 108]. Commercially developed security hardware technologies are available for spacecraft designers to implement; however the details on im-

plementation are highly proprietary and unavailable for public scrutiny and their cost tends to be extremely high due to the cost of development and lack of competition.

Additionally, the perceived difficulty of adversarial access to spacecraft communication may lull new commercial operators into a false sense of security, believing space to still be the sanctuary environment it once was. While specialized equipment and knowledge are required to communicate with spacecraft, governments and large corporations are hardly the only entities possessing such equipment and skills. Numerous resources are available online to instruct individuals on how to communicate with available spacecraft, for both cooperative and non-cooperative scenarios [107, 144].

Due to the size, cost, and specialized nature of space missions, hardware and software utilized on spacecraft have traditionally either been developed in-house for the specific mission at hand, or purchased from vendors who have developed their own. As the operating environment of space places strong limitations and constraints on traditional spacecraft hardware, software would need to be developed at a very low-level using limited development environment tools and require re-education whenever a hardware platform was updated. Likewise, the growing complexity of modern spacecraft has further challenged software development [65]. International organizations like the Consultative Committee for Space Data Systems have emerged to help standardize component interface protocols and aid component interoperability at all levels. However, significant challenges remain, which has prompted a new direction in space platform development: utilizing general-purpose hardware and software tools to provide companies with lower-cost hardware, using well-known instruction sets, and compatible with a wide-range of preexisting commodity software development tools. Such systems still require specialized fabrication techniques, which keeps the cost of such hardware high, but competition and the adoption of existing hardware architectures continue to push costs down. As an example, the company CySec specializes in secure computing built with ARM's Trust-Zone capability for use on spacecraft. Other spacecraft vendors such as GomSpace, EnduroSat, and Alba Orbital, all offer ARM-based processors. Ad-

ditionally, NASA recently selected RISC-V as the architecture for their next generation High-Performance Spaceflight Computing (HPSC) processor for upcoming space, lunar, and planetary missions [92].

With the rapid entrance of new commercial vendors introducing previously unavailable capabilities to the global population, institutions and government entities will likely begin to increase reliance on these systems; however, in order to protect a return on investment, most of these services are provided using closed-source and proprietary technology. Aside from hardware and software security concerns, space-based platforms supporting critical infrastructure and emergency services must be well studied and understood in academia to provide the public with the necessary understandings of potential risks, dangers, or threats (if any). We are seeing such scenarios unfold today, as cell phone carriers begin to integrate space-based capabilities into their communication and emergency infrastructure. While organizations such as 3GPP develop the actual communication protocols used to enable such capabilities, the "proprietary-curtain" infrastructure that these services rely on, specifically their robustness and resiliency, are not well understood. Internal architectural changes heavily influence the operational capabilities of these networks, yet changes are made without public awareness. Additionally, any security-related impacts derived from changes that are made publicly, such as Starlink's previous requests to modify the approved configuration of their satellite constellation, are not easily apparent due to a lack of architectural understanding.

The research we conducted divided itself into three areas of interest: Near-Earth orbits, cislunar, and interplanetary space. Near-Earth is the easiest to access and is currently the most contentious among commercial entities; cislunar operations are being prepared as interest from international governments in semi-permanent lunar operations is increasing. Interplanetary space is the most far-reaching of regions, but still of great commercial interest due to the potential for substantial (and unprecedented) payoffs through either asteroid mining or accessing other planetary body resources [145].

1.1 Research Challenges

The academic study of spacecraft and space network security is an emerging and difficult field for numerous reasons: Commercial spacecraft are often not available for academic testing, spacecraft hardware is extremely expensive, and many spacecraft security implementations are proprietary, to name a few. Our research seeks to address the following challenges we have identified:

- **Limited implementations of robust security in spacecraft design despite anticipated threats**. Since 2001, the security concerns regarding space-based assets has been advertised in government findings [96, 123], however, the actual implementation of security measures has lagged behind for a multitude of reasons [41, 108].
- **Inflexibility and risk of hardware cryptographic modules in spacecraft security**. The rigidity in implementation of hardware cryptographic modules, along with risk of component failure during every firmware update, is not an ideal solution in the current cybersecurity environment. Adoption of new post-quantum cryptographic schemes, along with the necessary flexibility that accompanies the implementation of new cryptographic schemes ([9, 88, 142]) makes adaptability, energy-efficiency, and certifiably-secure cryptographic solutions essential for long-term spacecraft missions.
- **High computational complexity of dynamic routing in mega-constellations**. The computational requirements to consistently maintain the shortest path between all nodes in a highly-connected time-dependent network currently exceeds available technological solutions [1], making the development of dynamic routing protocols suitable for mega-constellations and an area of ongoing study. As energy-efficiency is the prized goal for most solutions, the resiliency of newly proposed algorithms is often overlooked.
- **Methodology gaps in evaluating space network performance and security**.

As with many emerging fields of study, a lack of consistency in metrics and an absence of defined frameworks make it difficult for the academic community to compare one proposal's performance to another. Additionally, the lack of robust testbeds that accurately model the unique operating environment and environmental constraints of space networks increases the barriers faced by researchers when seeking to make meaningful contributions in this field.

1.2 Research Contributions

In summary, the research contribution made within this work are as follows:

- We introduce a resilient and scalable dynamic routing approach for LEO mega-constellations that leverages a novel logical topology and proof-of-concept routing algorithm with significantly reduced per-hop computational overhead while outperforming existing solutions. Our proposed logical topology simplifies dynamic routing calculations by exploiting satellite relative mobility patterns. Our presented dynamic routing algorithm that utilizes this logical topology to demonstrate superior resilience and scalability with minimal real-time computation.

- We advance low-energy security operations for resource-constrained spacecraft by investigating the use of intermittent computing and recommending the integration of security considerations at all levels of spacecraft design. We applied intermittent computation techniques adapted from IoT environments to enable AES encryption on microcontrollers and FPGA processors intended for cislunar missions that reduces instantaneous energy costs. We proposed a low-energy security design framework that incorporates cryptographic operations into the earliest phases of mission design, thereby reducing overall mission costs. We analyzed several contemporary and post-quantum cryptographic schemes at a range of security strengths, identifying potential benefits

of applying intermittent computation to cryptographic operations critical for next-generation deep space delay-tolerant networks.

- We develop frameworks for holistic performance assessments that systematically evaluate both network resilience and the suitability of intermittent cryptography in space environments. We proposed a network resilience analysis framework that combines scenario-specific guidelines, adversarial categorizations, and performance metrics for robust protocol evaluations. Additionally, We proposed "IC-CAPE" (Intermittent Cryptographic Circum-Architectural Performance Evaluations) to measure the potential benefits and compatibility of cryptographic operations under intermittent computing conditions.

- We develop realistic, non-terrestrial network testing tools, including a Bundle Protocol version 7 module for NS-3 and critical enhancements to the SpaceNet emulator, to support hardware-in-the-loop integrations and large-scale simulations. We developed and publicly released an NS-3 Bundle Protocol version 7 module that supports TCP, UDP, and LTP layers, filling a critical gap in delay-tolerant network simulations. We expanded SpaceNet's capabilities with hardware-in-the-loop integration, scalable "optimizer" execution modes, and node-level autonomy for dynamic routing. We also supported the development and public release of 'SpaceNet' as a flexible research testbed for analyzing the impact of orbital mechanics on communication performance in LEO mega-constellations.

Chapter 2

Background

2.1 Unique Characteristics of Space Domain

2.1.1 Regions of Interest in the Space Domain

While space is vast, the areas of human and commercial interest are proportionally modest. Although not commonly divided along such lines, for the purpose of studying security in space systems and networks, we divide space into the following regions of interest:

- Near-Earth
- Cislunar
- Deep Space / Interplanetary / xGEO

The Near-Earth region encompasses LEO, MEO, and GEO orbits; however, the primary interest in recent years has been LEO due to the emergence of SpaceX's mega-constellation. Near-Earth operations are generally categorized as low-latency, permitting the use of many terrestrial-based protocols. However, due to the rapid transit across Earth's geography, near-Earth networks of interest tend to be massive in size, ranging from several hundred nodes to several thousand creating complexity unique to this region. Near-Earth space is the least expensive to access and companies are able to launch large numbers of spacecraft into it.

The region of space referred to as 'cislunar' covers all lunar orbits as well as any spacecraft and networks between the Earth and the Moon. The final region, called deep space /

interplanetary space, or (more recently by government and defense officials) xGEO, which describes the region of space beyond the Lunar orbit. Cislunar operations have the widest range in characteristics compared to other categorized regions due to the area of interest this region covers, spanning between Near-Earth to objects in Lunar orbit. Latencies can be short or long and network sizes can be small or large. Objects in Lunar orbit are also more erratic than Earth-orbiting objects due to the varied nature of Lunar orbital mechanics; stable Lunar orbits are uncommon and highly valued, likely becoming areas of international interest/tension due to the strategic advantages stable orbits provide. NASA has proposed the establishment of a heterogeneous communication network called LunarNet within cis-lunar space which will encompass both orbital and lunar surface relay nodes to build an integrated communication network. This region is more expensive to access, with launch costs potentially 20x more expensive than Near-Earth launches [66].

The terms Deep space / interplanetary space / xGEO (referred to in the rest of this document as simply 'deep space') describes the region beyond the Lunar orbit. Defining characteristics of this region are low density networks and extremely high data latency links, potentially having values up to minutes and hours due to light-speed lag. The size of the region, sparsity of occupying networks, and the expense to reach it, generally precludes ubiquitous point-to-point connectivity and/or dedicated relay nodes, requiring spacecraft to cooperate by relaying messages in a peer-to-peer model. Deep space networks are also characterized as lacking constant end-to-end connectivity, with messages often requiring extended, temporary storage on a relaying node before the node's mobility brings it within view/range of of the next relaying node. This region has the highest cost to access with the strictest constraints on mass. Future spacecraft launched into deep space will prioritize small sizes, tight system integration, and low-power operating modes to achieve the most performance in the lightest possible package.

Traits to anticipate with deep space networks and hardware are:

- Extreme packet latency and non-interactive communication.
- Node mobility impacting link availability; frequent network partitioning.
- Lack of centralized coordination beyond ground control.
- Messages structured as large, single-transmission 'bundles'.
- Messages relayed between nodes using a 'store-and-forward' mechanic to provide message delivery; node mobility becomes a component of data transmission.

2.1.2 Space Environmental Impacts on Computing Hardware

Space is the harshest imaginable operating environment for computational systems. Aside from the heat and cold exposures, the disruption of constant heavy-particle bombardments, and the ever-persistent ionizing radiation, the vacuum of space itself can disrupt hardware with component off-gassing. Computing platforms operating in space face numerous design constraints not present for their terrestrial counterparts and has been outlined in prior studies [1, 14, 108].

Across all of these constant dangers, the most significant long-term threat to space systems is the exposure to a variety of radiations. From the perspective of computational hardware, the two radiation types of primary concern are ionizing radiation and particulate radiation, in the form of heavy-particle bombardment. Despite the heavy-sounding name of heavy-particle bombardments (HPB), this radiation only causes 'soft' errors in the form of occasional bit-flips in certain types of volatile memory. Long-running terrestrial systems also experience this phenomenon, but with much less frequency due to the dual-protection of Earth's atmosphere and magnetosphere. Ionizing radiation, however, is a killer of commodity hardware. This radiation effects the polarity of the material used in most terrestrial processors, ultimately overwhelming the processor's logic gates and permanently locking

their states. Additionally, advanced processors manufactured at smaller scales are more susceptible and impacted sooner by ionizing radiation than processors forged at larger scales [36].

Generally, commercial spacecraft harvest solar energy to power the onboard electronics. While solar energy is plentiful, a spacecraft's access to this energy is dictated by the size of its solar panels, the orientation of those panels, the distance of that spacecraft to the sun, and how often the spacecraft passes in the shadow of a body that blocks solar energy from reaching it. Due to these factors, spacecraft are deployed with onboard battery to power equipment when solar energy is insufficient or absent.

Spacecraft batteries are often the life-blood of a satellite's utility, with the operational capability of the spacecraft directly tied to the fitness of the battery. As such, mission planners meticulously develop energy budgets that attempt to maximize battery longevity through practices such as minimizing depth-of-discharge (DoD). This can produce scenarios where, despite the onboard battery having a sufficient charge to perform a computational operation, the power budget policy may not allow that operation to be performed until a later time.

Lastly, the cost of launching a satellite into space, while significantly cheaper now than it has ever been, is still a considerable expense for even the largest of companies, with the cost of a launch dictated by the weight of the spacecraft. This cost also increases significantly the further away from Earth a mission is located. Given this, launch expenses can become prohibitively expensive, producing a design scenario with competing requirements: Spacecraft must be engineered for redundancy and robustness due to hazardous operating factors and the overall expense of getting it into space, and the pressure to keep the spacecraft's mass at an absolute minimum–a factor that often impedes the simple inclusion of redundant components for increased operational resilience.

2.1.3 Spacecraft Mass vs Launch Cost

SpaceX has significantly disrupted the cost of launching spacecraft to Low-Earth orbit, resulting in a significant decrease in the cost per kilogram, with the reduction being further enhanced for ride-sharing launches. Based on recently published data, private organizations and universities are now able to launch spacecraft using SpaceX ride-sharing services for approximately $4,800 per kg into Earth orbit [52]. Unfortunately, launching spacecraft to lunar orbit or beyond still remains prohibitively expensive, with mass having the single-largest impact on cost. NASA launched its first lunar rideshare mission in 2022 when Artemis I deployed 10 cubesats near lunar space; using the disclosed mission expenses as a basis for estimate (and applying a x2 factor for realistic expectation of future costs), we project a cost of approximately $110,000 per kg to reach Lunar orbit. While this is only an 23-fold increase in price for an average of 100-fold increase in distance, this does highlight the very real fiscal burden of each gram of spacecraft mass when reaching past near-Earth orbits.

This tyranny of mass[1] continues to be a significant challenge for spacecraft mission designers. While an expensive, inaccessible system with an extended operating lifespan would likely have redundant physical components and larger-than-necessary capacity for energy reserves, the cost of each gram necessitates novel solutions and clever engineering to achieve resiliency goals without adding additional, redundant components.

2.2 Paradigm Shifts in the Space Industry

After SpaceX's disruption of the commercial space industry, access to space by commercial entities has never been cheaper or easier. Through innovative and bold technologies, SpaceX has lowered the cost of Low Earth Orbit launches by over a factor of 10 [110]. This unforecasted shift in space access flipped the existing industry nearly overnight. No longer the

[1] Non-aerospace variant for the expression 'tyranny of the rocket equation'.

domain of governments and massive industry corporations, technology companies of all sizes suddenly had access to space with a need for all of the novel commercial applications they can get.

SpaceX's next magic trick was the disruption of global, satellite-based communication in the form of Starlink. By launching a previously unimaginable number of small Low Earth Orbit satellites, Starlink covered the globe in a communications network that offered *low latency* Internet coverage to under-served locations around the world. First generation satellites connected customers to nearby ground stations using a traditional 'bent pipe' method, where satellites relayed data between two points on the globe that were within its view. Newer generation satellites are equipped with inter-satellite links (ISL) that enable satellites to pass data between each other, removing the requirement for Starlink customers to be within a specific distance of a Starlink ground station. Usable and accessible Internet is now potentially within everyone's grasp nearly anywhere on the globe.

While SpaceX has been the catalyst for many disruptions in the space industry, they are not the only company attempting to seize the initiative. Across 14 companies, there are 17 planned and developing broadband internet satellite constellations targeting consisting of over 50,000 satellites. Along with demonstrating how seriousness of commercial investments, these launches also represent an opportunity for companies to loosen the grip on the global broadband market as only one of the 13 top broadband providers is developing/collaborating a LEO satellite constellation (Hughes Network Systems) [77, 118]. The absence of entrenched broadband providers in this rapidly evolving sector means this market is open to old and new companies alike.

2.2.1 Evolving Architectures of Spacecraft Processors

During this revolutionary access to space, spacecraft are also adopting more capable processors (e.g., CYSEC's ARM Cortex-A, NASA's multicore RISC-V [2, 3, 139, 140]), re-

ducing development barriers and supporting commodity software such as Linux. This shift in architecture provides numerous advantageous to spacecraft designers, but the execution performance and impact of commodity, general-purpose hardware in the space environment is not thoroughly tested (beyond assessing projected operational lifespans); this literature gap provides opportunities for future research regarding novel benefits as well as potential risks.

2.2.2 Evolution of Spacecraft Storage and Memory

Modern spacecraft architectures are adopting commodity components that include new storage technologies to increase both capacity and capabilities [2, 32, 139, 140]. High-performance spacecraft platforms often employ traditional DDR3/DDR4 dimms to support fast and efficient data processing; some or all of this memory is equipped with ECC to combat high rates of in-memory bit flips from heavy-particle bombardments. In addition to fast-speeds, DDR4 supports self-refresh modes when placed into "stand-by" to significantly reduce power requirements to maintain resident data when the system is idle.

Spacecraft are also more frequently built with ultra-low-energy non-volatile solid state drive technologies, such as magnetoresistive random-access memory (MRAM) and ferroelectric random-access memory (FRAM). In addition to native radiation-resistance, these SSDs provide non-volatile storage with extremely low read and write energy costs. While low-energy SDDs such as MRAM and FRAM are increasing in storage capacity, the maximum storage sizes generally range from only several kilobytes up to several megabytes. To augment these storage limitations, spacecraft often include slower, more energy consuming flash memory, such as OSPI, SD, and eMMC for larger data storage. These capabilities are illustrated in AMD's line of Versal SoCs and the processing modules that use them [2, 138].

2.2.3 Evolving Spacecraft Communication Architectures

Spacecraft are becoming increasingly integrated into networks at all layers of communication, ranging from ground station networks, to LEO mega-constellations, and to deep space delay tolerant networks. This increased connectivity increases the availability of spacecraft to the assets that need them. LEO mega-constellations are now implementing inter-satellite links (ISLs) that utilize directional RF antennas/optical transceivers to route packets between multiple satellites before being sent to ground terminals, resulting in large, highly-connected orbiting networks [39, 122]. However, environmental, manufacturing, and launch constraints make ISL packet routing a non-trivial problem for LEO mega-constellations that precludes the re-application of existing, terrestrial-based solutions [1]. Dynamic packet routing between highly-connected satellite infrastructures is an area of ongoing study. However, while sparse networks, such as deep space DTNs, may see an increase in available network connections, these nodes still anticipate limited or lacking end-to-end connectivity for any given data message and must still be structured and organized around this expectation.

2.3 Spacecraft Security Implementations

Given each mission's cost and importance, spacecraft security is critical [10, 96], yet robust implementations often face environmental, power, and bandwidth constraints [41]. In 2022, NASA mandated FIPS 140 cryptographic module protection for civilian and government missions (Space System Protection Standard [96]) and recommended it for commercial ventures; however, this policy permitted deep space missions to opt out—an approach we find unsuitable given the longevity of their missions. Although specialized hardware modules can be used to meet security strength goals while reducing certain overheads, they are inflexible, harder to update, and pose a single point of failure. By contrast, software-based cryptography is easier to patch and evolve but draws more energy than is commonly permissible for

space missions.

Transmission Mediums Because deep space networks rely on broadcast mediums, all messages can potentially be intercepted by unauthorized or unintended recipients. In the case of RF, signal emissions are made from both main-lobe and side-lobe emissions; although optical transmissions have tighter main lobes and no side lobes, these signals remain susceptible in principle.

Inaccessibility of Space A core challenge is protecting in-memory or on-disk data from attackers who gain physical access to a device. In deep space contexts, however, physical inaccessibility confers a distinct advantage: barring any side-channel leakage, the knowledge and resources required to physically reach orbiting or interplanetary platforms are typically far beyond the monetary value of the data itself. Furthermore, state-level actors with such capabilities would likely target larger government missions whose system design may not align well with the benefits of our proposed approach.

Longevity of Deep Space Missions Due to the expense of reaching deep space, the capacity for over-the-air updates to extended-mission platforms is crucial. Such updates may be driven by missions requirements or for security vulnerability fixes. The significance of these updates is increasing as advances in computer hardware may dictate the implementation of emerging 'post-quantum' encryption schemes (see discussion in Section 5.5.2); however, recent research on the security of post-quantum schemes highlights the expectation of "growing pains" for these new schemes as vulnerabilities are identified [9, 88, 142]. Hardware-implemented cryptography will either be unable to update to correct identified fixes due to hard-wired behavior, or require firmware updates that risk unrecoverable errors in single-point-of-failure components.

2.4 Related Work

2.4.1 Resilient Routing

As satellite position and geographical visibility of LEO mega-constellations are in continual flux, the use of traditional routing methods that rely on maintaining full network routing tables becomes impractical due to the issues outlined in Sections 2.1.1 and 2.1.2. Terrestrial routers typically use computationally expensive algorithms to find the shortest path to any known destination network with each topology change and are not expected to handle constant and continual routing table updates (colloquially known as 'route flapping'). Previous works, listed in Table 2.1, have explored suitable and efficient routing alternatives for mega-constellations [14, 29, 56, 155].

Bhattacherjee, et al. made several proposals of various dynamic routing algorithms that could be applied to mega-constellations in [14]. However, the authors' basic assumptions rely on the use of omnidirectional inter-satellite link interface, achievable only in RF-based implementations and not adaptable with the current technology's adoption of directional, optical-based interfaces. The machine learning approach to dynamic routing proposed by Cigliano, et al. [29] did show promising results, but was only demonstrated in a model less than 4% of Starlink's Shell 1 size and lacked sufficient clarity for reproducibility. Grislain, et al. made a proposal focused on the reduction of traffic congestion in [56] by spreading user-traffic across a broader number of links rather than prioritize traffic to links that are on the shortest path. While a promising presentation, the authors used link utilization as the primary evaluation metric and compared the performance of their proposed algorithm to a single shortest-path algorithm (Floyd-Warshall), rather than performing an evaluation that included traditional metrics or to other state-of-the-art proposals. The dynamic routing algorithm proposal by Zhang, et al. in [155] also sought to reduce traffic load, but on satellites themselves rather than links. Uniquely, their routing algorithm did not utilize inter-satellite

Date	Authors	Title	Key Contributions
Dynamic Routing in LEO Mega-Constellations			
2019	Bhattacherjee, et al.	Network Topology Design at 27,000 km/hour [14]	Initial proposal of using 'motif's to improve dynamic routing efficiency
2020	Cigliano, et al.	A Machine Learning Approach for Routing in Satellite Mega-Constellations [29]	Demonstrated use of Double Deep Q Networks (DDQN) machine learning-based routing in small-scale representation of LEO mega-constellation
2020	Qi, et al.	A Distributed Survivable Routing Algorithm for Mega-Constellations with Inclined Orbits [113]	Proposed grid-based path selection for efficiency and multi-path calculation for fault-tolerance
2022	Zhang, et al.	Segment Routing for Traffic Engineering and Effective Recovery in Low-Earth Orbit Satellite Constellations [155]	Proposes segment-based traffic-splitting algorithms in bent-pipe constellation to minimize satellite traffic load and two alternate-reroute mechanisms for greater fault tolerance
2022	Stock, et al.	Distributed On-Demand Routing for LEO Mega-Constellations: A Starlink Case Study [131]	Proposes bounded sub-graphs for improved dynamic routing efficiency
2022	Zhao, et al.	Self-Healing Motif-Based Distributed Routing Algorithm for Mega-Constellation [156]	Demonstrates use of motif-based dynamic routing for path selection and link failure recovery
2022	Grislain, et al.	Rethinking LEO constellations Routing with the Unsplittable Multi-Commodity Flows Problem [56]	Demonstrates routing algorithm with improved link congestion rates over shortest path-based algorithms
Energy-Aware Computation in Spacecraft			
2019	Denby, et al.	Orbital Edge Computing: Machine Inference in Space [34]	Proposed use of image filters on formation of nanosatellites to down-select uninteresting portions of image frame to minimize downlink bandwidth requirements
2021	Li, et al.	Towards Sustainable Satellite Edge Computing [80]	Proposes energy-aware operation scheduling to reduce battery depth-of-discharge
2023	Bleier, et al.	Space Microdatacenters [17]	Proposes use of dedicated in-orbit data processing to address downlink communication bottlenecks
2024	Li, et al.	Battery-Aware Energy Optimization for Satellite Edge Computing [81]	Enhances energy-aware battery preserving scheduling with satellite collaboration to further reduce battery wear
Space Network Simulators			
2019	Yang, et al.	NPVT: Network Protocol Validation Testbed for Integrated Space-Terrestrial Network [149]	Introduces NPVT space-terrestrial network emulator to validate diverse network protocols.
2020	Liu, et al.	Large-Scale Small Satellite Network Simulator: Design and Evaluation [82]	Introduces a low-overhead and scalable simulation platform for satellite networking and routing mechanism verification
2020	Kassing, et al.	Exploring the "Internet from space" with Hypatia [67]	Introduces Hypatia as a framework for simulating temporal variations in path structure and latency of LEO networks
2021	Kempton, et al.	Network Simulator for Large Low Earth Orbit Satellite Networks [74]	Presents tool for simulating arbitrarily sized and structured satellite constellations for topology and path dynamic routing analysis.
2021	Valentine, et al.	Developing and experimenting with LEO satellite constellations in OMNeT++ [143]	Presents a LEO satellite constellation simulation model using OMNeT++ and INET for developing and experimenting with orbital networks
2022	Hou, et al.	A Realistic, Flexible and Extendible Network Emulation Platform for Space Networks [60]	Presents a space network emulator focusing on authenticity, flexibility, and extendibility
2022	Pfandzelter, et al.	Celestial: Virtual Software Systems Testbeds for the LEO Edge [111]	Introduces Celestial as LEO emulation tool using microVMs to evaluation arbitrary software performance on orbiting networks
2023	Lai, et al.	StarryNet: Empowering Researchers to Evaluate Futuristic Integrated Space and Terrestrial Networks [78]	Presents StarryNet as emulation-aided framework to build realistic integrated space and terrestrial networks.
2024	Gao, et al.	Plotinus: A Satellite Internet Digital Twin System [49]	Introduces Plutonis as microservice-based digital twin system for satellite Internet emulation to handle complex dynamics and offer in-depth analysis with modularity and real-time capabilities

Note: The relatively small number of prior efforts represented here reflects the early state of research in this domain. This table highlights the foundational nature of ongoing contributions, including the present work.

Table 2.1: Related work aligned with research efforts identified in Section 3.3.

links and instead relied on hopping data between satellite and ground-stations. Despite not requiring the use of still-immature inter-satellite link interfaces, the success of this proposal relies on global and uniform deployment of operator ground-stations; an unpractical scenario in the real world.

The bulk of such works have focused on the computational efficiency of route selection with only a few publications addressing routing resiliency [113, 131, 156]. In [156], Zhao et al. builds overlapping network 'motifs' where nodes use local routing tables to manage multiple routes to near neighbors. While demonstrating routing resilience, this method produces computational and network traffic overheads at every link state change, which occur frequently under normal conditions. While being significantly less than maintaining local models of the entire network state, these overheads are non-trivial and must be accounted for in hardware design. In [131], Stock et al. propose a routing method that narrows the network region under consideration to identify near-optimal paths between nodes without the computational overhead of Dijkstra's algorithm. However, these authors did not perform resilience analysis under link disruption, preventing any nominal basis of comparison against other proposed methods.

Additionally, across all of the discussed dynamic routing proposals, no author performed evaluations against other state-of-the-art mega-constellation dynamic routing proposals, instead demonstrated effectiveness against baseline/'naive' implementations or terrestrial-based algorithms unsuitable for non-terrestrial network use, such as Open Shortest Path First (OSPF), Dijkstra, and Flloyd-Warshall.

The topic of resilient, dynamic routing in mega-constellations is explored further in Chapter 4, where we propose the use of a logical topology to simplify dynamic network routing calculations on each node.

2.4.2 Intermittent Computing

As mentioned in Section 2.1.2, a spacecraft's battery health is directly tied to its operational capabilities. Spacecraft battery life is traditionally maximized by strict energy-budgeting; however, recent research has proposed several novel approaches to assist with "battery-friendly" operations, such as scheduling computationally intensive operations to run during periods of high energy availability, with state-of-the-art proposals listed in Table 2.1.

Recent research on reducing spacecraft energy usage has focused on scheduling power-intensive tasks during periods of high energy availability, typically leveraging predictable orbits in Low Earth Orbit (LEO) missions [80, 81]. However, these methods rely on stable and periodic solar exposure, which may not apply to deep space platforms operating at greater distances from the Sun with less predictable sunlight and less overall energy reserves. Additionally, these proposals assume a homogeneous task operation and data size to perform lightweight estimates for optimal times of task execution

Similarly, Denby et al. introduced an 'Orbital Edge Computing' framework, which enabled small LEO satellites in close formation to collectively perform machine-learning tasks using 'batteryless' execution [34]. Their work aligned with intermittent computing principles—shifting compute-heavy tasks to harvested energy windows—but targeted swarms of low-power satellites with communication windows defined by predictable LEO dynamics rather than lone spacecraft operating in sparse network conditions. Additionally, only tasks with sufficiently low complexity to fully execute on limited energy reserves are compatible with this solution. Bleier et al. similarly explore the feasibility of shifting compute into space using orbital micro data centers [17], also targeting the specific data processing and transmission challenges of earth observation satellites.

An approach discussed in Chapter 5 explores the use of intermittent computing to address limited power reserves by executing high-computational tasks in a piece-meal fashion. Intermittent computing encompasses a range of techniques aimed at preserving forward progress

through frequent power cycles, typically focusing on checkpoint overhead and correct execution. Some existing methods include continuous checkpointing approaches that track state changes at regular intervals [148] while other works seek to shift the program execution model for efficiency gains [30, 83, 85, 147]. Alternatively, other proposals suggest the use of 'Just-in-Time' checkpoints, where system states are checkpointed only when sensing imminent power failure, achieving the minimal amount of checkpoint overhead [11, 84, 146].

Most prior efforts focus on ultra-low-power microcontrollers; in contrast, we adapt intermittent computing strategies for emerging deep space platforms, whose computing designs increasingly include 'application-class' processors.

2.4.3 Space Network Simulators

Recent advances in inter-satellite communication technologies, as outlined in Section 2.2.3, has expanded the need for research into these new satellite networks. As research lacks robustness without experimental environments to test new and existing proposals, academic researchers have recently contributed to the development of space network simulators by either adapting existing solutions or developing new solutions from scratch.

Table 2.2 outlines a summary of space network testbeds identified in literature, providing a general description of each simulator's intended use, any common platforms or protocols supported, backend simulation components, and various capability support. The listed testbeds encompass a wide-range of focuses and capabilities and includes the 'SpaceNet' testbed, which we discuss in detail in Section 6.2.

Several of the included testbeds leverage NS-2 and NS-3 for network stack and link performance simulation. However, these testbeds execute sequentially and single-threaded, scaling poorly to mega-constellation sized networks and challenge real-time modeling of networks greater than a few dozen nodes.

Additional testbeds utilize other tools, such as native Linux capabilities or containerized environments, such as Docker. However, not all tools are open source or do not incorporate several key features, such as weather-based signal attenuation and computational impact assessments.

Testbed 1st Author	Problem Scope	Supported Protocols	Emulator Backend	SDN	ISTN	HIL	WXM	OSS	CIA
NPVT Yang [149]	Validate network protocols in ISTN	Many	NS-2	✓	✓	✓	✗	✗	✗
LSNS Liu [82]	Two-layer network architecture	DTN-inspired protocol	Custom Java-based	✗	✗	✗	✗	✓	✓
Hypatia Kassing [67]	General LEO network emulation	Linux kernel	NS-3	✗	✗	✗	✗	✓	✗
Untitled Valentine [143]	LEO constellation latency evaluation	None	OMNeT++	✗	✗	✗	✗	✓	✓
SILLEO-SCNS Kempton [74]	LEO constellation design evaluation	None	Custom Python-based	✗	✗	✗	✗	✓	✗
Celestial Pfandzelter [111]	Edge computing in LEO networks	Linux kernel	Augmented [74]	✗	✗	✗	✗	✓	✓
Untitled Hou [60]	Dynamic space network emulation	TCP/IP, UDP, CCSDS, DTN	Kubernetes/ NetEm	✓	✓	✓	✗	✗	✗
StarryNet Lai [78]	Analyze space-terrestrial networks	Linux kernel	Docker container	✓	✓	✓	✗	✓	✗
Plotinus Gao [49]	Dynamic SAGIN network emulation	Linux kernel	NS-3	∼	✓	✓	✗	✗	✗
SpaceNet (*this research*)	Realistic and low footprint LEO network emulation	Linux kernel	Mininet	✓	✓	∼	✓	✓	✓

Table 2.2: Summary of mega-constellation network testbeds in the literature, sorted by publication date. Capability abbreviations are defined as follows: SDN: Software-defined network controller support; ISTN: Integrated space-terrestrial network support; HIL: Hardware-in-the-loop support; WXM: Weather modeled attenuation support; OSS: Open-Source Software; CIA: Computational Impact Assessments. ✓- indicates feature support; ∼- indicates partial support; ✗- indicates no support.

Absent from Table 2.2 is the NASA/JPL environment Interplanetary Overlay Network on Delay/Disruption Tolerant Networking (ION-DTN). It was not included due to this tool serving as an open-source testbed for mission planners to implement software and protocol modules in an 'operating environment' that mirrors the conditions of deep-space as opposed to tools that flexibly permit modeling space networks under a variety of operating conditions. While capable of cross-compilation, ION-DTN is intended for use on the target hardware architectures, with the resulting binaries and configurations capable of being directly loaded onto actual mission hardware. While this low-level testing capability is critical to the success

of deep-space missions, its scope limits the versatility required for effective academic exploration and research. The configurations necessary for ION-DTN are complex and intended to precisely match those of a planned mission; varying environmental conditions is not a native capability of ION-DTN. The end result is that ION-DTN is ill-suited for purely simulation-based space environment research, where a large number factors and implementations can be explored.

The topic of robust space network simulators is discussed further in Chapter 6, where we discuss the advances we have made to several open-source testbeds.

2.4.4 Methodology Gaps

A recurring theme in this research is the necessity for novel measurement frameworks tailored specifically to evaluate unique performance aspects of space network security and efficiency, that cannot be adequately addressed by conventional frameworks. Table 3.1 lists the major themes of research conducted in this work, along with the methodological contributions made; specific frameworks were developed for each of these topics, with details of their designs included in the respective chapters.

Resilient Routing When comparing mega-constellation routing algorithm resilience, we identified a lack of common metrics and test scenarios in current literature. Within the related works, such as those listed in Table 2.1, there is considerable variation in how each proposal measured their evaluation, lacking any consensus for algorithm-agnostic analysis. Additionally, no known works have proposed a categorization of adversarial disruption capabilities, nor incorporated what we believe to be the most likely form of adversarial disruption: geographically-based denial-of-service attacks. This lacking of a common framework for resilience analysis has made it difficult for researchers and industry to accurately compare the capabilities and successes of proposed works.

This methodology gap is addressed in Chapter 4, where we introduce a novel logical topology and propose a common measurement framework to evaluate dynamic routing resilience under adversarial disruptions.

Intermittent Computing When considering the adoption of intermittent computing concepts to spacecraft platforms, measuring effectiveness compared to continuous execution required careful consideration. Existing methods to evaluate continuous execution efficiency, (e.g. cache hit/miss rates), do not account for partial task completion as performed by IC. At the same time, prior work on IC primarily focuses on contexts where continuous execution is *impossible*, while prioritizing the reduction of overhead penalties to reduce their impact. As a result, there is little in-depth comparison of the two as alternate, competing approaches to software execution. For our investigations into program execution in deep space, energy consumption and efficiency are the primary concerns. However, even the definition of 'energy efficiency' is not uniform across all domains and manufacturers [133], warranting careful delineation when using the term in Chapter 5, where we propose the IC-CAPE framework to evaluate cryptographic execution efficiency in energy-constrained environments.

Given these gaps in existing literature due to the novelty of the field of study, we will propose our own methodologies where appropriate to establish a standardized approach for future research utilize. This standardization will focus on the salient traits of the respective field and common metrics that permit easier comparison between future proposals.

Chapter 3

Unified Research Framework

3.1 Framework Key Attributes

From the holistic consideration of this emerging domain, several key attributes were identified as pillars essential to effective, secure, and robust space networks: 1) Security, 2) Energy-Efficiency, 3) Resilience, and 4) Realistic Emulation and Testing.

Security The the cost of launching to near-Earth orbits has been considerably reduced, the expense of modern spacecraft development, assembly, and launch still represent significant monetary investments by government or commercial owners. As such, a high value is placed on the data being sent to and from the spacecraft.

Energy-Efficiency Modern spacecraft are highly autonomous systems and not physically connected to any other systems or supporting infrastructure; this means they must satisfy the entirety of their energy needs through harvesting. To counter the simple solution of utilizing large solar panels or possessing a sizable battery, the mass of a spacecraft directly impacts the cost of its launch, as discussed in Section 2.1.3. If the mission requires launching multiple spacecraft, or sending spacecraft into deep space, these costs grow exponentially. As such, any sustainable mission must seek to harvest the minimal amount of energy required to perform its task in order to minimize the size and weight of energy harvesting and energy storing components. Additionally, these components lose efficiency and capacity over time,

dictating specific usage patterns to maximize their lifespans, further constraining the amount of energy available for immediate use.

Resilience Space systems and networks must remain highly resilient on both implementation and component levels operate in space conditions for extended mission durations. Easily considered the harshest operating environment known, computational platforms are subjected to vacuum, intense cold/heat, and multiple forms of radiation, all conditions that terrestrial-based systems are protected from.

Realistic Emulation and Testing Effective solutions for secure and resilient space communications must simultaneously address resilience, energy efficiency, robust security, and practical validation. Due to competing requirements and constraints of the space environment, any proposed solutions must seek to meet the identified key attributes through *innovation* and thorough, methodical *testing and validation.*

The attributes of resilience, energy-efficiency, security, and realistic emulation and testing, as we have outlined them have guided our overall research approach and methodology.

3.2 Threat Model

As the space domain is undergoing a period of rapid change, so too do the perceived threats of this region. Previously considered the purview of only the most developed countries, bad actors that threaten the normal operation of space systems can now come in a variety of forms. While the threat of small groups or individual adversaries is not trivial, the most likely threats will come from larger, well-funded groups of highly technical and highly trained operators. For our research, we assume an adversary that is well-funded, staffed by experienced personnel in deep space communications and network/host penetration, and likely to be backed by major commercial ventures or nation-states [135]. Such adversaries

would likely be state or large corporation-backed actors with the goals of:

- Disrupting availability (DoS attacks / signal jamming / damaging device)
- Compromising confidentiality (eavesdropping communications / break encryption)
- Compromising data integrity (injecting, modifying, or replaying messages)
- Accessing legitimate assets (spoof/masquerade attacks)

Many of these attacks focus on identifying and exploiting security vulnerabilities on satellite modems. As presented in the foundational study performed by Yu et. al., satellite modems have different design considerations than their terrestrial counterparts that are not commonly addressed in current academic literature [152]. This lack of focus leaves the most vulnerable component of a spacecraft, operating as an open receiver, poorly researched and poorly understood.

3.2.1 Disruption of Availability

Considering the cost of spacecraft launches, adversaries will most likely seek to interfere with the operation of space networks using ground-based equipment [79, 108, 109, 126]. While radio-frequencies are the dominant form of ground to satellite signaling, projects devoted to the advancement of optical ground stations have begun in the last few years by nearly all major space agencies [4, 26, 27, 94]. Despite current research being government supported and focusing on large site installation using cutting-edge technology, ground-to-space optical communication can still be achieved using extremely modest equipment [120], and should not be excluded when considering potential adversarial transmission methods.

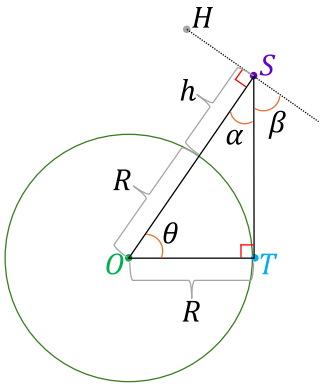
Modern optical communication apertures, unlike RF communication, use an extremely limited Field of View (FoV), of just a few degrees wide. Such a limited FoV does significantly reduce the physical location of attackers, it does not preclude the possibility of ground based

disruptions, which would require novel approaches to perform. While the narrow FoV of optical interfaces does increase component resilience to signal "spoofing" as any illegitimate optical signals would have an extremely limited range of physical locations they can be generated to produce an impact, this limitation increases the precision in which legitimate signals must be aimed and pointed. Such precision comes at a cost of increased attitude control complexity to offset atmospheric and orbital perturbations. Ongoing research exists in the development of high-bandwidth, wide-angle optical apertures that drastically increase the FoV, permitting a much greater range of angles to receive signals [8]. A widened FoV improves aiming tolerances and lowers attitude control complexity and cost; however, mission planners must balance the FoV range with potential sources of adversarial signals to minimize any attack surface. An example in estimating the minimum FoC range necessary to be susceptible to ground-based attacks, assume a LEO satellite orbiting at an altitude of 520km and using an inter-satellite optical communication interface perpendicular to nadir; optical signals from Earth's surface are capable of being observed by a satellite if the ISL interface FoV is at least 46°.

If deep space relay networks adopt optical communication, interfaces are fully expected to possess a wide FoV as the distance between nodes is orders of magnitude more distant than near-Earth spacecraft. This increases the necessity for mission planners to account for potential reception of disrupting signals.

When adversaries are seeking to disrupt space networks by emitting signals, we assume adversaries will be capable of disrupting links between satellites or disrupting a satellite's ability to send/receive data when a satellite is passing over an adversary's geographical location. We assume these proposed adversarial emitters are capable of producing effects that either last only while a satellite is within the area of disruption or permanently damage satellites so that functionality continues to be disrupted after the satellite has left the disruption area.

While traditional networking literature defines adversarial disruptions in terms such as jam-



$$R = 6,378km \qquad h = 520km$$
$$\overline{OT} \perp \overline{ST} \qquad \overline{OS} \perp \overline{HS}$$
$$\alpha = 90° - \theta \qquad \beta = 90° - \alpha$$
$$\beta = 90° - (90° - \theta) = \theta$$
$$\cos(\theta) = \frac{\overline{OT}}{\overline{OS}} = \frac{6,378km}{6,898km} \approx 0.9247$$
$$\theta = \arccos(0.9247) = 23°$$
$$\beta = 23°$$

Figure 3.1: Estimation of Low-Earth Orbit inter-satellite optical interface minimum half-cone field of view necessary to view Earth horizon. O: Earth's center; T: Tangential horizon point on Earth's surface (where $OT \perp OT$); S: Satellite position; H: Reference point that defines local horizontal at S (so $OS \perp SH$); h: Altitude of satellite; R: Distance from Earth's center to surface; α: Compliment of β; β(half-cone FoV): Angle between local horizontal (HS) and tangent line (TS).

ming, denial-of-service, and node destruction, these do not directly account for the unique dynamics of space-based networks and orbital motion. The framework we propose here maps broadly to these established categories, but adapts them for relevance to satellite-specific architectures, including inter-satellite link (ISL) topologies, orbital visibility windows, and constellation-side coordination. Applied disruption *categories* align with traditional notions of localized, targeted, and distributed attacks, while disruption *types* reflect both transient and permanent adversarial impacts. We categorize such adversarial capabilities according to the following outline:

I: Disrupt ISL capabilities for satellites passing over a specific geographical area. Disruption intensity is defined as the percent of ISL interfaces disrupted out of the total number of ISL interfaces available on affected satellites. Satellite ISL functionality is restored after leaving the disruption region.

II: Disrupt all packet routing capability for satellites passing over a specific geographical area. Disruption intensity is defined as the percent of satellites over the disruption region affected. Satellite functionality is restored after leaving the disruption region.

III: A permanent disability of packet routing capabilities for individual satellites that prevents any future packet routing along any ISL interfaces starting from the moment of disruption.

IV: A disruption of all packet routing capabilities for a number of satellites across the entire constellation. Disruption intensity is defined as the percent of all satellites in the constellation affected. Satellite functionality is fully restored when the disruption duration has elapsed.

V: A disruption of ISL capabilities for all satellites across the entire constellation. Disruption intensity is defined as the percent of ISL interfaces disrupted out of the total number of ISL interfaces available on each affected satellite in the constellation. Satellite functionality is restored when the disruption duration has elapsed.

These adversarial disruptions are divided into the following categories: Geographically-based disruptions, Targeted disruptions , and General disruptions. Disruption Types I and II are geographically-based, Types IV and V are general disruptions and Type III disruptions are targeted. General disruptions are less realistic than geographically-based and targeted disruptions but are still effective for comparing resilience measurements across various routing algorithms. Disruption intensities and duration can be modified when considering strong vs weak adversarial models.

3.2.2 Compromise of Confidentiality

When not attempting to redirect or disrupt the flow of traffic, we assume adversaries will eavesdrop on satellite transmissions for the purpose of breaking confidentiality. In this work, we consider threats both to spacecraft hardware and to communication emissions going to and from the spacecraft. Such confidentiality breaches would come in the form of breaking weak or exploitable data encryption schemes, either symmetric (transmission of

encrypted data) or asymmetric (transmission of encrypted shared secret), imposed by energy and hardware constraints. Additionally, we consider an adversary that attempts to influence the flow of traffic by transmitting false data that is masquerading as legitimate by breaking weak or insecure digital signatures.

As referenced in Section 2.3, deep space communication signals may be intercepted by unintended recipients. Given this, we consider an adversary capable of injecting messages directly into the network. With sufficient monitoring resources, an adversary could track the path of each data bundle as it is relayed through the Delay Tolerant Network (DTN), thereby increasing the need for robust key sizes and signing/encryption schemes-especially if an adversary assumes a significant portion of retransmissions contain recognizable plaintext. Meanwhile, emerging quantum-safe cryptographic methods offers new key sizes and computational complexities but also introduce novel vulnerabilities.

Many existing government or commercial spacecraft that support classified missions require the implementation of NSA-approved cryptography [104]. While NSA mandated cryptographic schemes and physical modules are typically robust (and very expensive), no system is immune to all forms of attack, particulary as government and industry shift to Post-Quantum encryption schemes. Although side-channel attacks are not our focus, recent work highlights numerous fault-injection and side-channel exploits targeting lattice-based signature schemes [9, 88, 142]; our discussion on post-quantum algorithms is in Section 5.5.2, but these findings underscore the importance of cryptographic implementations that are flexible, updatable, and extensible in the face of new threats.

3.2.3 Compromise of Data Integrity

An adversary seeking to compromise the integrity of transmitted or stored data in a space network aims to alter legitimate information in transit without being detected. Integrity breaches threaten the trustworthiness and legitimacy of mission-critical commands, teleme-

try, and science data. Such threats extend beyond mere eavesdropping or disruption by corrupting or altering data to mislead or degrade mission operations.

Adversarial manipulation could take various forms, including modification of transmitted commands, alteration of software updates, or injection of falsified data for the purpose of provoking incorrect decisions. In DTN scenarios, integrity violations pose a pronounced risk due to data being relayed across multiple intermediate nodes, increasing potential opportunities for attack.

Considering spacecraft hardware constraints, cryptographic integrity checks (such as digital signatures, cryptographic hashes, or authenticated encryption modes) must balance energy efficiency, computational complexity, and robustness against evolving cryptographic threats over a long period of time.

3.2.4 Unauthorized Access

Adversaries seeking unauthorized access aim to establish illicit access to legitimate commands for onboard systems, potentially enabling follow-on attacks or exploitation. Such attacks got beyond eavesdropping or integrity alteration as they involve active manipulation of spacecraft commands and components. These assets may include communication modules, navigation systems, power management systems, propulsion, or data storage systems.

Unauthorized access presumes preexisting compromise of confidentiality, or hijack of command and control system lacking adequate cryptographic protections. Additionally, adversaries must be familiar with the target architecture in order to produce meaningful effects. Emerging spacecraft often employ commodity hardware and standardized software stacks, simplifying development efforts while also simplifying attacker exploitation efforts. Additionally, with the goal of using reprogrammable spacecraft platforms to increase operational flexibility, vulnerabilities related to unauthorized reconfiguration are introduced.

3.3 Alignment of Identified Threats to Research Attributes and Contributions

Each of the threats identified in Section 3.2 seek to invalidate one or more of the defined key attributes: A space network's resilience is degraded when adversaries disrupt or influence the normal flow of data or data messages themselves, security of both the network and attached devices is impacted when adversaries compromise the encryption and authentication schemes selected for data handling in constrained energy environments, and adversaries will seek to achieve their goals through exploiting vulnerabilities not easily managed or defended on modern spacecraft due to a lack of tools and testing environments available to academic and industry researchers.

***Attribute* / Research Theme:**	**Core Problem Addressed:**	**Applied Contributions:**	**Methodological Contribution:**	**Key Results & Impact:**
Resilience: Resilient Routing for LEO Mega- Constellations (Chapter 4)	Scalable, dynamic, resilient routing	Novel logical topology and routing algorithm	Framework for evaluating dynamic routing resilience	Enhanced network resilience; highly scalable [73]
Energy-Efficiency and Security: Energy-Efficient Security for Cislunar and Deep Space (Chapter 5)	Secure communications under severe resource constraints	Novel use of Intermittent Computing to enable efficient secure communication	Energy and performance metrics for evaluating intermittent cryptographic execution	Enable certifiable-secure cryptography at low power [68] (*)
Realistic Evaluation and Testing: Realistic Evaluation of NTN Computing Environment via Simulation (Chapter 6)	Lack of robust testing utilities for academia	SpaceNet Mininet-based emulator, NS-3 Bundle Protocol emulation	Flexible simulation tools supporting researcher-defined evaluation metrics	Enabled accurate and scalable testing environments [37, 71, 72, 129, 130] (†)

* Manuscript being updated for submission to *SpaceSec 2026*
† Manuscript currently being updated for re-submission to *IEEE Access Journal*

Table 3.1: Overview of Research Contributions

Table 3.1 outlines the research we performed that addresses each of these identified threats, along with the domain's aligned attributes. Our work on improving the resilience of future space networks through scalable and effective dynamic routing suitable for LEO mega-constellations is discussed in Chapter 4. Additionally, our research into energy-efficient implementations of symmetric and asymmetric cryptographic schemes is presented in Chap-

ter 5. Lastly, our contributions to the evaluation of current and future NTN proposals using realistic simulations is presented in Chapter 6.

Chapter 4

Resilient and Dynamic Logical Topologies for LEO Mega-Constellations

This chapter contributes:

- A novel mega-constellation network topology based on Starlink's Shell 1, with each satellite using six directional ISLs, called the TriCoordinate logical topology. This topology reduced real-time calculation complexity for LEO mega-constellation routing without the use of internal network state models.
- A novel, proof-of-concept dynamic routing algorithm using the TriCoordinate logical topology, named TriCoordinate Axis Priority routing. This resilient, lightweight, and scalable algorithm maintains higher packet delivery rates than current state-of-the-art proposals without significant computational or network overheads.
- A general framework to study mega-constellation routing resilience that includes categorization of adversarial disruption capabilities, standardized performance metrics, and requirements for analyzed routing scenarios.

Novelty and Significance The TriCoordinate logical topology introduces a novel coordinate system explicitly designed around the architectural characteristics of LEO mega-

Metric	Scenario/Condition	**TriCoordinate** (Ours)	**Self-Healing Motif** [156]	**DisCoRoute** [131]
Packet Delivery Rate %	Nominal Conditions	**100%**	95-100%	46-89%
	High Adversarial Disruption (75% Intensity, Type II)	**93-98%**	67-94%	22-86%
	North-South Routing (Ave)	**85%**	79%	59%
	East-West High Lat. Routing (Ave)	**66%**	54%	21%
Latency (ms)	Nominal	**60-80 ms**	60-100 ms	60-110 ms
	Adversarial Conditions	60-160 ms	60-130 ms	**60-120 ms**
Jitter (ms)	Nominal	6-18 ms	**3-8 ms**	2-14 ms
	Adversarial Conditions	6-52 ms	4-30 ms	**0-29 ms**
Computational Overhead	Link-state changes	**None**	Moderate (frequent)	**None**
	Per-packet overhead	**Constant (low)**	**Constant (low)**	Variable (path-dependent)
Network Traffic Overhead	Routing table distribution	**None**	Moderate (frequent)	**None**
Methodology Novelty	Routing Topology	Novel TriCoordinate	Local routing tables	Region-based search

Note: Packet delivery rates vary by scenario, disruption type, reflecting constellation design and adversarial disruption intensity. Our TriCoordinate method consistently achieves higher resilience, particularly under severe disruptions.

Table 4.1: Overview of Resilient and Dynamic Logical Topologies for LEO Mega-Constellations contributions

constellation inter-satellite links. Unlike previous approaches that rely on computationally expensive routing tables or regional route-finding (as in Self-Healing Motifs or DisCoRoute), our method enables real-time route calculation with minimal computational overhead, no routing table exchanges, and inherently strong resilience characteristics. This topology simplifies complex real-time dynamic routing into straightforward algebraic operations, enabling the first demonstrated scalable, resilient routing framework specifically suited to LEO mega-constellations under realistic adversarial disruption scenarios.

4.1 Introduction

As seen in recent news, the promise of wide-spread, satellite-based network connectivity can be a double-edged sword as space hardware is widely accessible to attempted adversarial disruption [19, 55, 115, 117]. While resistance and counter-measures should be employed at all levels of design, when utilizing dynamic routing to pass data between inter-satellite

links, the resilience of the routing algorithm is a critical factor. Challenges related to the the calculation of dynamic routes in a LEO mega-constellation have been previously outlined in Section 2.4.1 with limitations and constraints placed on computing in space that further increase the challenge and complexity of such routing have been outlined in Section 2.1.

Much of the content in this chapter has been presented in [73]. While this document includes expanded explanations and discussion, the experiment and data analyzed remain the same. The code used to generate the data for this study is located at [69].

We addressed the challenge of dynamically routing packets through a massive, highly-connected, time-dependent network by taking advantage of aspects dictated by the network's constellation type. We achieved this goal by overlaying a logical topology on the physical constellation to reduce the complexity of determining a packet's next-hop interface, where our proposed solution uses an architecture-dependent two-dimensional poly-coordinate topology. As the number of packet-movement axes of our two-dimensional logical plane is dictated by the number of ISL interfaces available on each satellite, several design and implementation assumptions need to be made.

4.2 Logical Topology

4.2.1 Satellite Relative Mobility and Design Assumptions

As ISL technology is still in development, with any active deployments considered highly proprietary, details surrounding its implementation for any other LEO mega-constellation provider is not publicly available. As such, several design assumptions and conclusions had to be made during this investigation, but were made with effort to be sound and reasonable based on available data.

For this investigation, we model Starlink's Shell 1 constellation, as Starlink operates the

largest LEO constellation in the world and the topic of many academic studies. Shell 1 is deployed as a Walker Delta constellation to permit maximal ground coverage using minimal satellites, while simplifying constellation design and management, making this a commonly employed configuration [75, 105]. According to FCC documentation, Starlink's Shell 1 operates using 72 orbital planes, with each orbital plane populated with 22 satellites, comprising a total Shell 1 population of 1,584 satellites [31].

Regarding the unknown aspects of Starlink's ISL implementation, we assume satellites will preference ISLs that are shorter and have increased orientation stability, tolerating an acceptable ISL interface alignment angle of +/-15°. We also assume a constellation phase offset of $\frac{\pi}{2}$ for Starlink's Shell 1 to maximize satellite spacing and even ground coverage distribution. From these assumptions, we identify two viable intra-plane satellite neighbors and four inter-plane satellite neighbors (two satellite neighbors from each adjacent orbital plane). Figure 4.1 shows ISL availability within our assumed parameters for all six potential neighbors of a single satellite: Intra-plane neighbor orientation remains relatively stable throughout a satellite's orbit; inter-plane neighbor orientations fall within stated interface alignment tolerances when the satellite traverses latitudes between 38°N and 38°S, providing full ISL connectivity for approximately 70% of a satellite's traversed latitude range. Given this observed behavior, we opted to model satellites equipped with six ISL interfaces oriented along three axes.

Topology and ISL interface directions are oriented using the following terms: Ram, wake, port, and starboard. Ram is the direction of the satellite's motion in its orbital plane with starboard being 90° clockwise from ram when looking down from a location immediately zenith to the spacecraft. Wake is the direction opposite of ram and port is the direction opposite of starboard.

As a final assumption, we presume a data packet's originating ground terminal can identify the target destination satellite at the time of transmission, permitting this work to focus

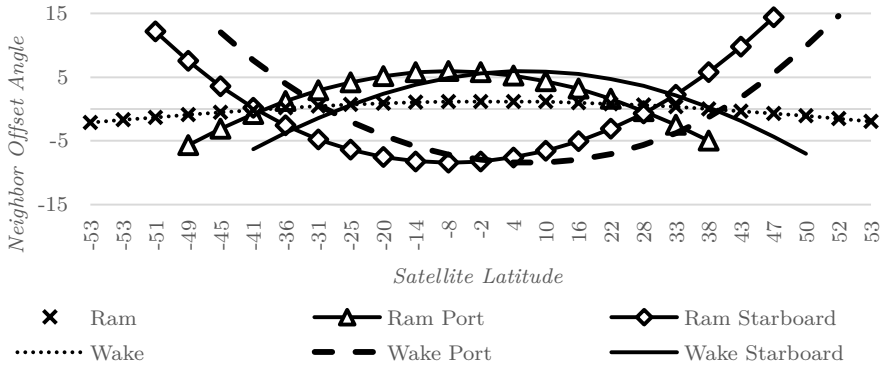


Figure 4.1: Neighbor angle offset from fixed-direction ISL interfaces for satellites in a modeled Walker Delta LEO constellation. Orientations for the six nearest neighbors are explored (ram, wake, port, and starboard pairs). Offset angles are plotted across orbital latitude over one complete period, illustrating dynamic neighbor alignment relative to each interface axis.

on the problem of inter-satellite routing, and not include the logic of efficient addressing. This assumption is reasonable as satellite-based ISPs have the potential to disseminate associations between logical addresses and ground terminal physical locations to end-point hardware, then have ground terminals perform physical location look-ups for non-cached destinations and identify overhead spacecraft using TLEs (or proprietary equivalent) in real-time.

4.2.2 Definitions

Several definitions and formulas are used to describe our proposed logical topology. To maintain conciseness of our definitions, common satellite constellation characteristic definitions used in our descriptions and formulas are located in Appendix A.1 with definitions for the variables P, O_n, S, i, s_n, v as equations A.1–A.6.

Using our novel approach of deriving ISL architecture-dependent two-dimension poly-coordinate topologies, we use six-direction ISLs to define three logical axes of packet travel. Using these three axes to represent inter-satellite links, where each intersection of all three axes represents the logical location of an individual satellite aligned with its physical neighbors, we form a lattice using equilateral triangles as the smallest basic shape of the topology; a geometric

shape well-suited to approximate curved surfaces.

Axis are labeled as A, B, and C, with the A-axis aligned along the orbital planes and B/C-axes aligned along inter-plane ISL interfaces. Each satellite possess a unique address, consisting of the three axis coordinates, and remain fixed once established. Coordinate values range from 0 to $P-1$ and roll over to 0 if increased past their maximum value (P defined as the number of orbital planes per Appendix A.1). A-coordinate values use base-10 number with B and C coordinates using base-S, for 'standard case' constellations, where we define 'standard case' constellations as Walker Delta using the criteria outlined in Appendix A.2.1. The rational to use base-S addressing for B and C coordinates is provided in Section 4.2.3.

From an arbitrary satellite number, i (defined in equation A.4), we calculate its A, B, and C coordinate values using the following developed equations:

$$A = O_n \tag{4.1}$$

$$B = \left(\left\lfloor \frac{O_n - (O_n \ mod\ 2)}{2} \right\rfloor - i \right) \ mod\ P \tag{4.2}$$

$$C = \left(\left\lfloor \frac{(O_n + (O_n \ mod\ 2)) \ mod\ P}{2} \right\rfloor + i \right) \ mod\ P \tag{4.3}$$

The first satellite in the constellation, in orbital plane 0 with orbit index 0, is assigned satellite number 0 and is located at the grid origin with coordinates (0, 0, 0). The next satellite in orbital plane 0 has orbit index 1 and satellite number 1 and is located along axis A, immediately forward of satellite 0 in the ram direction.

4.2.3 Prime Meridian and B/C Coordinate Translation

TriCoordinate logical topology's *Prime Meridian* is the line $A = 0$. In the standard case, a translation of B/C coordinate values is required when comparing coordinates across the Prime Meridian due to lines along B/C axes not terminating at their point of origin on

the Prime Meridian. This can be visualized as lines along axis A depicted as closed loops and lines along B and C axes depicted as spirals. We use base-S numbering for B/C axes to simplify this translation to a static offset value for a 'standard case' Walker Delta constellation. Offset values are specific to constellation characteristics. For the example constellation used for analysis, using constellation characteristics listed in Appendix A.2.1, the derived B/C coordinate translation offset is 8.

A partial visual representation of the TriCoordinate logical topology is depicted in Figure 4.2.

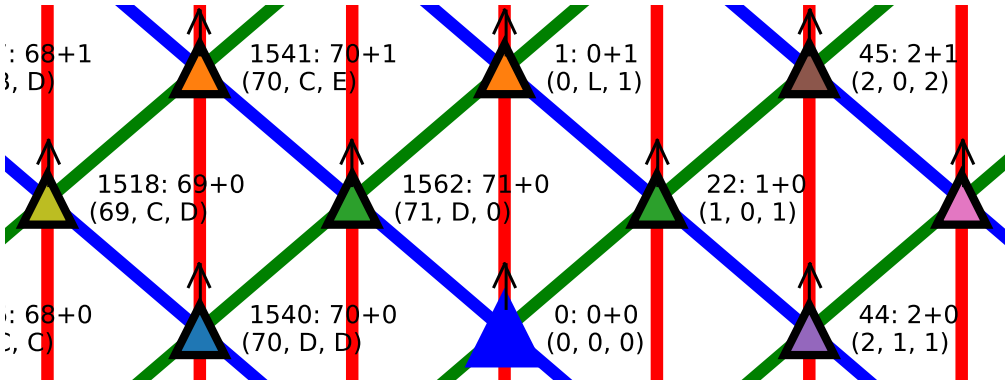


Figure 4.2: Partial view of the TriCoordinate logical topology developed for low Earth orbit mega-constellations. Each triangle represents a satellite located at the intersection of three logical axes: A (red), B (green), and C (blue). Arrows indicate direction of motion within the satellite's orbital plane (ram). Satellites are labeled with: (1) a unique identifier within the topology, (2) their orbital plane and index within that plane (e.g., 70+0), and (3) their TriCoordinate location in the grid (e.g., (0, 0, 0)). This topology simplifies spatial reasoning and routing calculations by enabling coordinate-based neighbor resolution across three directions.

4.3 TriCoordinate Priority Axis Routing

Traditional Manhattan-grid location and distance calculations are not directly applicable in our proposed topology due to having more than two axis in a two-dimensional plane and the presence of a single diagonal path between four adjacent neighboring nodes in the triangle lattice.

The use of the TriCoordinate logical topology does not inherently dictate particular routing strategies; however, several routing approaches can maximize the utility of TriCoordinate logical typology's design. For our initial study, we route packets according to axis priority using a method we call *TriCoordinate Axis Priority* routing. When a satellite receives a packet to forward, we compare coordinate values between the current satellite and the packet's destination and select a forwarding interface using only local ISL status data. This approach identifies reasonably short paths with minimal computational and storage overheads while also avoiding network traffic overhead and propagation delays. Nodes can optionally report to their neighbors when they have only a single ISL interface available to avoid unnecessary forwarding to a node that cannot relay, but otherwise nodes do not require any knowledge of link statuses beyond their own. TriCoordinate Axis Priority Routing is calculated on-board each satellite upon receipt of a forwarding packet using the following steps:

1. Identify the smallest coordinate difference between each current and destination coordinate value using the following equations:

$$A_{diff} = Min(|A_{dest} - A_{curr}|, |A_{curr} - A_{dest}|) \tag{4.4}$$

$$B_{diff} = Min(|B_{dest} - B_{curr}|, |B_{curr} - B_{dest}|) \tag{4.5}$$

$$C_{diff} = Min(|C_{dest} - C_{curr}|, |C_{curr} - C_{dest}|) \tag{4.6}$$

2. Label $A/B/C$ axes along the following criteria:

   $Major$: Axis with the largest coordinate value difference

   $Inferior$: Axis with the smallest coordinate value difference

   $Minor$: Remaining axis

3. Designate forwarding interface as the highest available priority using the following criteria:

Priority	Axis to Reduce	Reduce Along Axis
1	Major	Inferior
2	Major	Minor
3	Minor	Inferior
4	Inferior	Minor
5	Minor	Major
6	Inferior	Major

To prevent small-scale routing loops, packets are not routed to satellites with no other links available unless they are the destination satellite. This does introduce a trivial amount of administrative network traffic, but no more than several kilobytes every few seconds. Additionally, interfaces that route back to the previous hop are automatically assigned the lowest priority.

We gain further computational efficiency by having nodes automatically forward incoming packets along the current axis of travel unless one of two events occur: the forwarding interface is unavailable, or an axis difference other than the current axis of travel is 0.

4.4 Routing Resilience Measurement Framework

In this section, we describe our proposed common measurement framework to establish effective and consistent comparisons of LEO mega-constellation routing algorithm performance and resilience within this and future works. Our framework outlines specific test scenario goals, baseline performance measurement metrics, and categorization of expected adversarial capabilities.

4.4.1 Terminology

Given the unique nature of LEO mega-constellations as network topology perpetually undergoing a state of change, various definitions of common terms may not be seamlessly transferred from terrestrial networks to their orbiting counterparts, particular regarding terms that may possess multiple definitions based on context and author.

In this work, our use of the term 'resilience' aligns with the definition provided by Yacov Haimes in [58] which is defined as 'the ability of the system to withstand a major disruption within acceptable degradation parameters and to recover within an acceptable time and composite cost and risks'. In the context of LEO mega-constellation networks, resilience continually remains a relevant factor even when no adversarial disruption or hardware fault is present - perpetual link 'churn' and large regions of reduced connectivity are all normal aspects and the nodes impacted by these characteristics change over time.

When using Haimes' definition of resilience, specific components like 'acceptable degradation parameters', 'acceptable time', and 'acceptable composite cost and risks' are implementation specific when it comes to orbiting networks and will be determined by the service provider when there is no established industry standard; however, factors that influence customer experiences can be informally-defined and estimated. For this study, we determine 'acceptable degradation parameters' by measuring packet delivery rates and packet latency. Delivery rate and latency also overlap with the definition of 'Availability', defined by Cho, et al. as a sub-metric of 'Security' in [28], where it is described as the state of the service possessing full functionality without using failure or recovery processes.

4.4.2 Resilience Measurement Framework Scenario and Adversarial Disruptions

To maximize the diversity of data collection, we propose the following goals for resilience scenario testing:

1. Perform a minimum of one packet-traffic scenario that maximizes anticipated link availability.

2. Perform a minimum of one packet-traffic scenario that anticipates reduced link availability due to satellite mobility (if applicable for a given satellite constellation).

3. Perform a minimum of one packet-traffic scenario using a path not covered by previous scenarios.

Scenarios should maximize the distance between packet traffic end points when not exploring near-distance routing.

Along with these scenario goals, we use the adversarial disruption categories outlined in Section 3.2.1.

4.4.3 Resilience Measurement Framework Performance Metrics

We propose the use of the following network performance metrics for future routing algorithm resilience studies. These metrics are user-traffic focused and seek to identify: How much traffic was successfully routed, how long did it take for packets to be routed, and what was the variation in delivery time for a given packet stream.

Using our established definition of resilience from Section 4.4.1, we measure acceptable degradation parameters in our framework using **packet delivery rate** and **packet latency**.

From these primitive metrics, considerable additional composite or aggregate metrics can be derived, such as **packet stream jitter**, which we include in our measurement framework. Additional composite metrics that can be produced are:

- **Delivery Ration Degradation**: Ratio of packet delivery rates without and without disruptions.
- **Latency Increase Factor**: Comparison of packet latencies with and without disruptions.
- **Delivery/Latency Stability**: Composite measurements utilizing established threshold values.
- **Service Availability**: Comparison of network performance against established availability thresholds.
- **Attack Success Rate**: Comparison of adversarial disruptions against service availability.

Advanced composite metrics such as these are most useful when employing either real-world networks for experimentation or using high fidelity network simulators. The network connectivity-based simulator used for our experimentation is best suited for an initial investigation, with advanced metrics being compiled in future experimentation.

Users experience negative impacts when packet delivery rates fall below 98% or packet latency rises above 200ms, with an ideal latency being less than 100ms. Packet stream jitter is based on the definition outlined in RFC 4689 as the absolute value between forwarding delays of two consecutively received packets belonging to the same data stream [112]. Jitter negatively impacts user experience for real-time applications such as voice-over-IP when values are 30ms or greater, assuming a packet delivery rate of 98% or better. The negative impact from packet delivery rates below 98% outweighs the impact from high jitter values [137].

Additional metrics may be incorporated for specific studies as needed, such as network reconvergence time when analyzing algorithms that maintain internal models of network link availability.

4.5 Evaluation

To compare routing algorithm performance, we developed a satellite orbit simulator in Python using the Skyfield API for all orbital positioning calculations and the available SGP4 API to derive an experimental constellation from a single, real-world satellite TLE [119].

For this study, we extrapolated realistic orbital characteristics for the Starlink Shell 1 constellation using a TLE for satellite STARLINK-1071 [25], retrieved using celestrak.org. we modeled a Walker Delta constellation configured with 72 orbital planes, 22 satellites per orbital plane, a satellite altitude of 550km, and an orbital inclination of 53°, giving a total of 1,584 satellites [44]. We assume a constellation phase offset of $\frac{\pi}{2}$.

Our simulator transmits ten "packets" during each time interval with packet origin divided evenly between the two route endpoints. Simulated time is advanced 2 minutes at each interval to balance appreciable planet rotation and satellite mobility without sacrificing overall fidelity.

Satellite ISLs beacon link statuses every time interval; a satellite marks an ISL as unavailable if it does not receive a signal during the current time interval. Routing table advertisements, updates, and propagation are also simulated as appropriate. All trials and scenarios were run using the same time epoch. Trials captured data for 74 simulated minutes to allow influential constellation characteristics to manifest while ensuring adequate satellite coverage at all packet route endpoints for all scenarios.

4.5.1 Experiment Trial Selection

Using our proposed routing resilience measurement framework, we developed three scenarios to run trials impacted by four adversarial capability types while observing three performance metrics.

4.5.2 Trial Scenarios

From our proposed resilience measurement framework scenario goals and target constellation configuration, we developed the following three trial scenarios:

- East–West Equator scenario: Packet traffic traverses an area along the equator, anticipating maximal ISL availability along the route's length, with a longitudinal separation of approximately 75°. Packet traffic endpoints are Kapenguria, Kenya and Pontianak, Indonesia.

- East–West High Latitude scenario: Packet traffic traverses an area between 45°N and 50°N latitude, anticipating limited inter-orbit ISL availability along the route's length, with a longitudinal separation of approximately 140°. Packet traffic endpoints are Seattle, WA, USA and Krakow, Poland.

- North–South Americas scenario: Packet traffic traverses a primarily North/South route with a latitudinal separation of approximately 118°. Packet traffic endpoints are Montreal, Canada and Comodoro Rivadavia, Argentina.

4.5.3 Selected Adversarial Disruption Types

To investigate the performance of TriCoordinate Priority Axis Routing, we selected disruption types I, II, IV, and V, as defined in Section 3.2.1. Type III disruptions were omitted

due to the disruption intensity required to generate appreciable impacts on packet traffic for our given scenarios produced effects that mirrored Type II disruptions. Type III disruptions were omitted for brevity.

Disruptions were applied at 30, 40, and 75% intensities to explore routing algorithm performance against both weak and strong adversarial models as well as explore routing algorithm sensitivity to small vs large changes in disruption strength.

Geographically-based disruptions impact satellites with at least 30° elevation at the disruption site. Placement considerations of geographically-based disruptions included both end-points and mid-point of each packet stream. However, disruptions applied at the packet stream mid-point did not reliably produce significant packet delivery impacts and were omitted for brevity. As simulated packet traffic is generated evenly at both end-points, adversarial disruptions were placed near a single end-point only. End-point disruptions were positioned along packet routes approximately 1,000km away from end-points to allow sizable disruptions without impacting delivery to endpoint ground stations as endpoint routing resilience was outside the scope of this investigation.

Once applied, disruptions endure for the duration of the trial.

Type I and II geographically-based disruptions were applied at the following locations:

- East–West Equator scenario: Baraawe, Somalia
- East–West High Latitude scenario: Reims, France
- North–South Americas scenario: Malargue, Argentina

4.5.4 Performance Metrics Selected to Observe

The resilience performance metrics we selected from our proposed framework to observe for each trial are: packet delivery rate, packet latency, and packet stream jitter.

Packet latency is calculated as a combination of hop count and link distance for a packet's route. The physical distance of each hop is multiplied by $3.3 * 10^{-6}$ sec/km to approximate light speed delay in a vacuum, with an additional 100 microseconds applied at each hop to approximate satellite processing logic and delays in forwarding packets to the appropriate interface. Note that this is an approximation of idle latency as opposed to latency during work load, as link/router congestion was not a component of this simulation.

Our approximation of jitter deviates from the traditional definition by measuring the difference in packet latency *between* time intervals as opposed to packet latency variation *within* a time interval due to the lack of simulated influencing factors. Additionally, as RFC 4689 measures jitter using consecutively received packets, not all trial results include a value for packet stream jitter when there are insufficient packet deliveries to measure.

Each trial delivers 50 packets prior to the application of adversarial disruptions (13.5% of a trial's total packet attempts) to allow for algorithm initialization, if necessary. The initialization period was used by the routing method Distributed Self-Healing Motif (discussed in Section 4.5.5) to populate local routing tables prior to processing consumer traffic. Given this "warming up" period of packet delivery, trial results with a packet delivery rate of 14% indicates no packets were delivered after the start of adversarial disruption. Packet delivery results less than 14% indicate packet loss was observed during the initialization period, before adversarial disruptions were applied.

4.5.5 Routing Algorithm Comparison

TriCoordinate Axis Priority Routing's resilience performance is compared against the following state-of-the-art, dynamic mega-constellation routing algorithms: DisCoRoute [131] and Self-Healing Motif-based Distributed Routing [156].

We selected DisCoRoute for analysis due to its proposal as a highly efficient alternative

to Dijkstra's shortest path. Additionally, we selected Self-Healing Motif-based Distributed Routing for analysis for its emphasis on efficient and resilient routing. Both algorithms were adapted rom their original proposal to a six-ISL interface model, but no modification of design logic was required.

Additionally, we implemented a naive model for analysis comparison as a variant of 'CoinFlip' presented in [131]. This implementation performs an inter-orbit hop if one is available, rather than performing a notional 'coin flip'. This modification was made as a trivial update to increase routing performance by biasing link selection towards less stable inter-orbit hops whenever possible; we named this algorithm 'Biased-CoinFlip'.

A list of routing algorithms selected for inclusion in this study, along with the labels used during the experiment, are as follows:

Routing Method	**Experiment Label**
Biased-CoinFlip	'Naive'
Self-Healing Motif-based Distributed Routing [156]	'Motif'
DisCoRoute [131]	'DisCoRoute'
TriCoordinate Axis Priority (this work)	'TriCoord'

Various overheads impact the performance of a given routing algorithm when it operates within a specific network structure. These overheads can be implementation specific, but several characteristics are typically tied closely to the algorithm design, specifically computational overhead and network traffic overhead. The size of these overheads can influence network managers depending on the amount of negative impact they have on network performance.

Computational Overhead

Two events prompt computational effort among the selected algorithms: receipt of a link-state change and receipt of a packet. Biased-CoinFlip, Self-Healing Motif-based Distributed Routing, and TriCoordinate Axis Priority calculate a received packet's forwarding interface at constant computational complexities, regardless of conditions. However, DisCoRoute's per-packet computational complexity is dependent on the number of intervening hops.

Additionally, Self-Healing Motif-based Distributed Routing performs additional computation upon the receipt of a link-state change to update internal routing tables. We define the term 'latitude state change' to describe the anticipated loss of inter-orbit satellite links that occurs when a satellite reaches latitude extremes, which we annotate as $\phi_a \rightarrow \phi_b$ (see Figure 4.1 for Latitude values that result in link losses). For the constellation used in our analysis, Self-Healing Motif-based Distributed Routing makes $(\nu - 1) * (\nu - 2)$ routing table updates at each latitude state change, with four latitude state changes occurring during each orbital period.

Network Traffic Overhead

To inform neighbors of a link's state in our evaluation, satellites beacon their ISL interfaces at regular intervals . Self-Healing Motif-based Distributing Routing also transmits routing tables to all neighbors in the event of any link-state change to maintain a consistent view of satellite 'motifs'. As with routing table calculations, the routing table advertisements are made at each link-state change, having a minimum of four routing table advertisements each orbital period for latitude state changes ($\phi_a \rightarrow \phi_b$).

4.5.6 Results

We performed this experiment to answer the following questions:

1. Which dynamic routing algorithms provide the highest packet delivery rate across all scenarios and disruption intensities?

2. Does any correlation exist between packet delivery rate and latency and/or jitter that may characterize resilient/non-resilient algorithm performance?

3. Were there any notable differences in algorithm performance across the tested scenarios?

4. Were there any notable differences in algorithm performance across adversarial disruption types?

The descriptions of trial results include illustrative figures depicting notable highlights. Trial result figures not included in this section are located in Appendix A.3. Notable results are also summarized in Table 4.1

Packet delivery rates are plotted as a point graph on each figure, packet latency is plotted as a bar graph, and packet stream jitter is represented by the bar graph fill. Cut-off lines are included in each figure at 13.5% packet delivery rate (red dashed line) and 30ms jitter value (blue dashed line). Trials with insufficient packet delivery will lack plotted latency or jitter values.

Results highlights for each scenario are as follows:

All Scenarios with no Adversarial Disruptions

Figure 4.3 shows routing algorithm performance in all three scenarios without adversarial disruptions applied. Most algorithms performed well except DisCoRoute in East–West High Latitude with a packet delivery rate less than 50%.

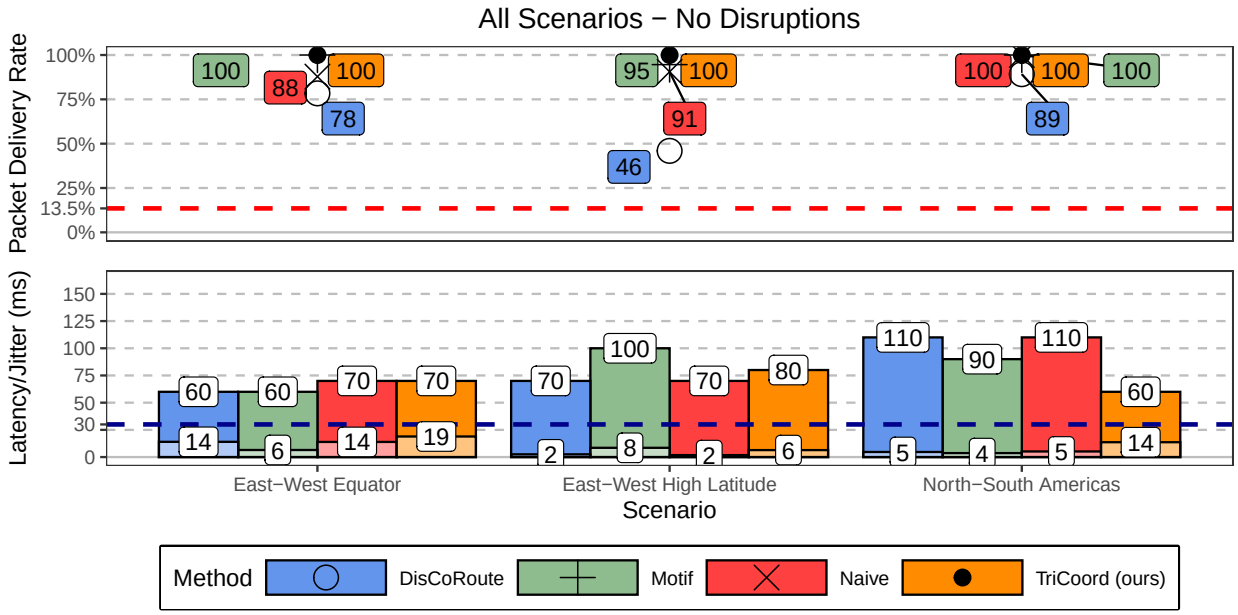


Figure 4.3: Comparative performance of four dynamic routing algorithms across three routing scenarios in the absence of adversarial disruptions. Packet delivery rate is shown in upper point graph, with packet latency and latency jitter is represented in lower bar graph and bar fill pattern, respectively. Red dashed line (top) marks packet delivery threshold below which routing is ineffectual during adversarial disruption (no disruptions depicted). Blue dashed line (bottom) marks jitter threshold above which degrades user experience.

East–West Equator Scenario

Anticipating maximal ISL availability, East-West Equator scenario results showed solid packet delivery rates for two of the four tested routing algorithms. TriCoordinate Axis Priority maintained or tied for the highest packet delivery rate across all adversarial disruption types and rates, followed by Self-Healing Motif-based Distributed Routing, which had matching rates for Type I and II disruptions at 30 and 40% intensities. DisCoRoute and Biased-CoinFlip demonstrated markedly reduced packet delivery rates across all disruption Types and intensities, with Biased-CoinFlip generally outperforming DisCoRoute by a slight margin. All packet latencies remained below 100ms except during Type V disruption at 40% intensity, where TriCoordinate Axis Priority had a latency of 140ms; however, this latency occurred when TriCoordinate Axis Priority had a delivery rate more than twice the next highest result. TriCoordinate Axis Priority did typically demonstrate the greatest amount

of jitter overall, it remained at or below 30ms except during disruption intensities that impacted the delivery rates of all routing algorithms. Figure 4.4 illustrates the combination where Self-Healing Motif-based Distributed Routing performed nearly as well as TriCoordinate Axis Priority for packet delivery rates and consistently demonstrated the lowest latency across all tested algorithms.

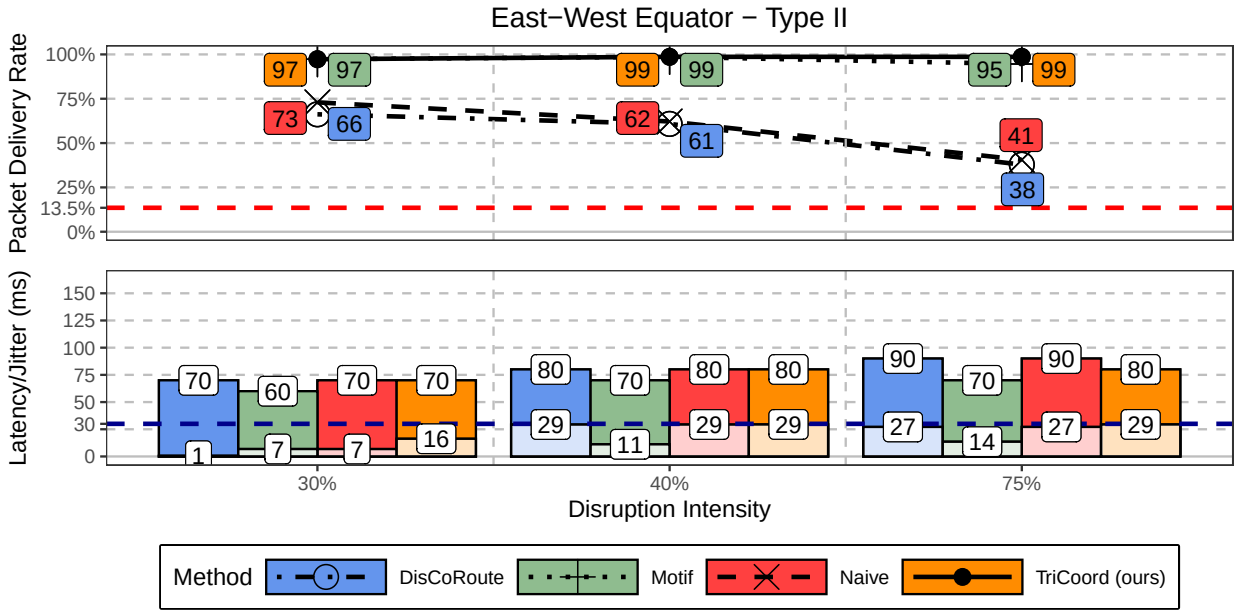


Figure 4.4: Comparative performance of four dynamic routing algorithms in the East–West Equator scenario under Type II adversarial disruptions. Packet delivery rate is shown in upper point graph, with packet latency and latency jitter represented in lower bar graph and bar fill pattern, respectively. Red dashed line (top) marks packet delivery threshold below which routing is ineffectual during adversarial disruption. Blue dashed line (bottom) marks jitter threshold above which degrades user experience. Results are shown for 30%, 40%, and 75% disruption intensities.

East–West High Latitude Scenario

Packet delivery rates were reduced over all adversarial disruption types and intensities due to the preexisting reduction of ISL link availability, as illustrated in Figure 4.5. TriCoordinate Axis Priority maintained the highest packet delivery rate for all adversarial disruption types except Type IV, where DisCoRoute demonstrated unusual packet delivery success, beating

all other routing algorithms during 30 and 40% disruption intensities.

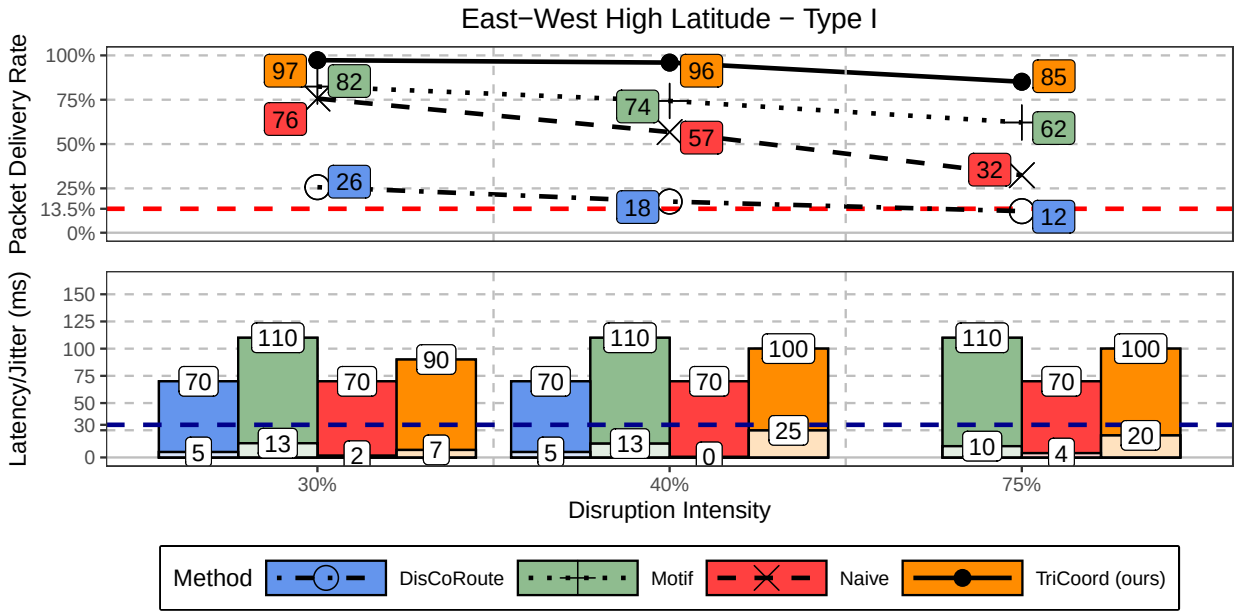


Figure 4.5: Comparative performance of four dynamic routing algorithms in the East–West High Latitude scenario under Type I adversarial disruptions. Packet delivery rate is shown in upper point graph, with packet latency and latency jitter represented in lower bar graph and bar fill pattern, respectively. Red dashed line (top) marks packet delivery threshold below which routing is ineffectual during adversarial disruption. Blue dashed line (bottom) marks jitter threshold above which degrades user experience. Results are shown for 30%, 40%, and 75% disruption intensities.

North–South Americas Scenario

All routing algorithms showed improved packet delivery performance compared to the two prior scenarios. TriCoordinate Axis Priority routing maintained the highest packet delivery rate with a minimum of 99% during Type I and II adversarial disruptions at all intensities. However, TriCoordinate Axis Priority had unusually low packet delivery for Type IV adversarial disruption at 30% intensity, being the second lowest at only 78% delivery, but recovered at higher disruption intensities, securing the highest delivery rates at 95% or above. Of note, Figure 4.6 shows Type V adversarial disruptions produced significantly reduced packet delivery for all routing algorithms. TriCoordinate Axis Priority had the high-

est packet delivery rates at 30 and 40% disruption intensities, with rates of 78% and 22% respectively; no algorithms successfully passed traffic at 75% intensity.

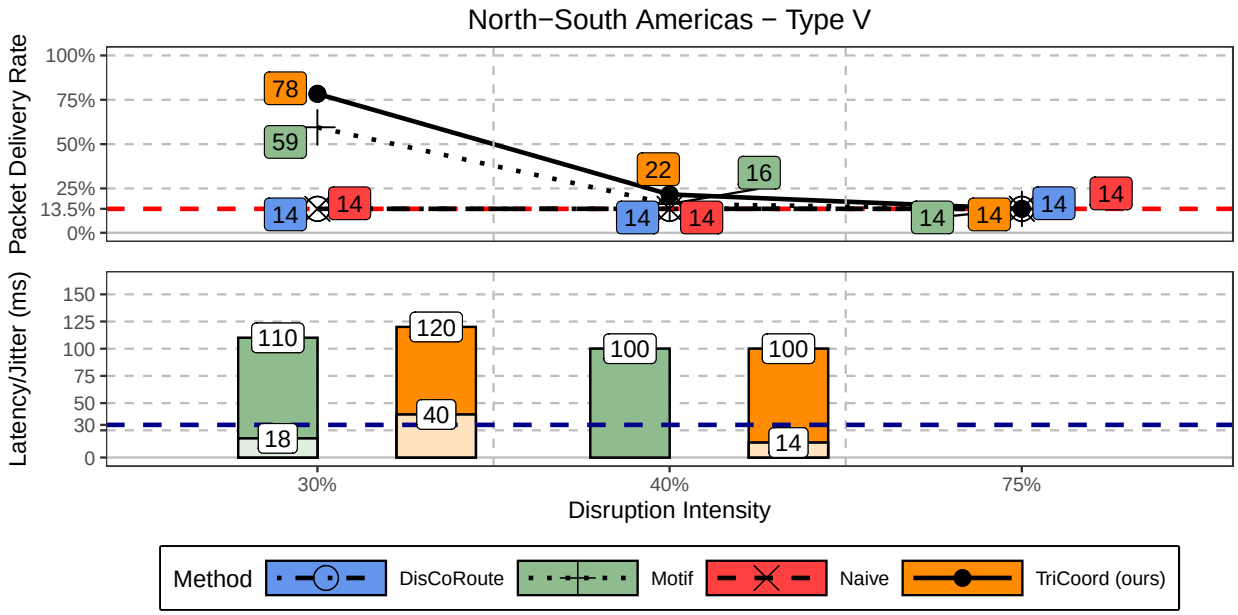


Figure 4.6: Comparative performance of four dynamic routing algorithms in the North-South Americas scenario under Type V adversarial disruptions. Packet delivery rate is shown in upper point graph, with packet latency and latency jitter represented in lower bar graph and bar fill pattern, respectively. Red dashed line (top) marks packet delivery threshold below which routing is ineffectual during adversarial disruption. Blue dashed line (bottom) marks jitter threshold above which degrades user experience. Results are shown for 30%, 40%, and 75% disruption intensities.

4.6 Discussion

From analysis of the performed trials, a number of findings regarding routing algorithm resilience can be drawn:

4.6.1 Mega-constellation Routing Algorithm Resilience

TriCoordinate Axis Priority demonstrated itself as the most resilient algorithm tested by maintaining the highest mean packet delivery rate across all disruption types and disruption

intensities, as shown in Figures 4.7 and 4.8. While Self-Healing Motif-based Distributed Routing had similar packet delivery rates at low levels of adversarial disruption, TriCoordinate Axis Priority demonstrated larger delivery rate gains at higher disruption intensities, as illustrated in Figure 4.9. Comparing TriCoordinate Axis Priority to Self-Healing Motif-based Distributed routing across all scenarios, we see TriCoordinate Axis Priority delivered more than 13% packets for Type I and II adversarial disruptions at 75% intensity and Type V adversarial disruptions at 30 and 40% intensities. The lack of delivery difference for Type V adversarial disruptions at 75% intensity was due to the inability of any routing algorithm to successfully deliver packet traffic after the start of disruption.

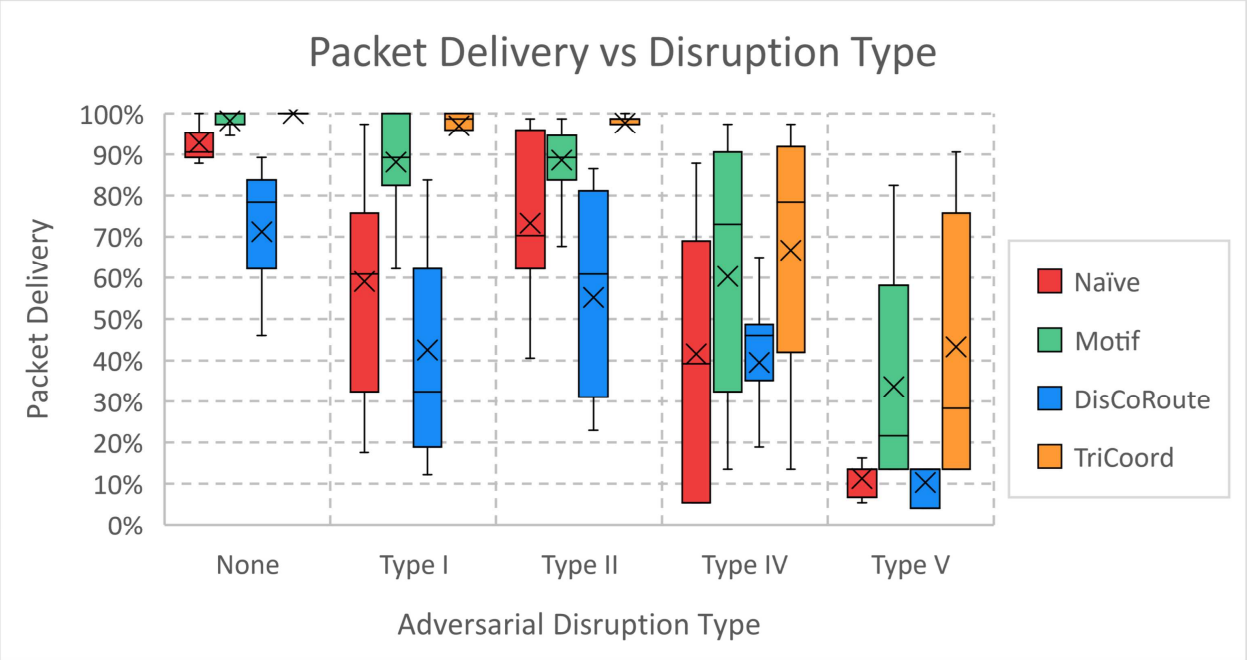


Figure 4.7: Algorithm packet delivery rates vs disruption type.

4.6.2 Routing Algorithm Flexibility and Packet Stream Jitter

Biased-CoinFlip and DisCoRoute demonstrated their design emphasis to minimize hop counts with consistently low latency and minimal jitter, but at the expense of significantly reduced

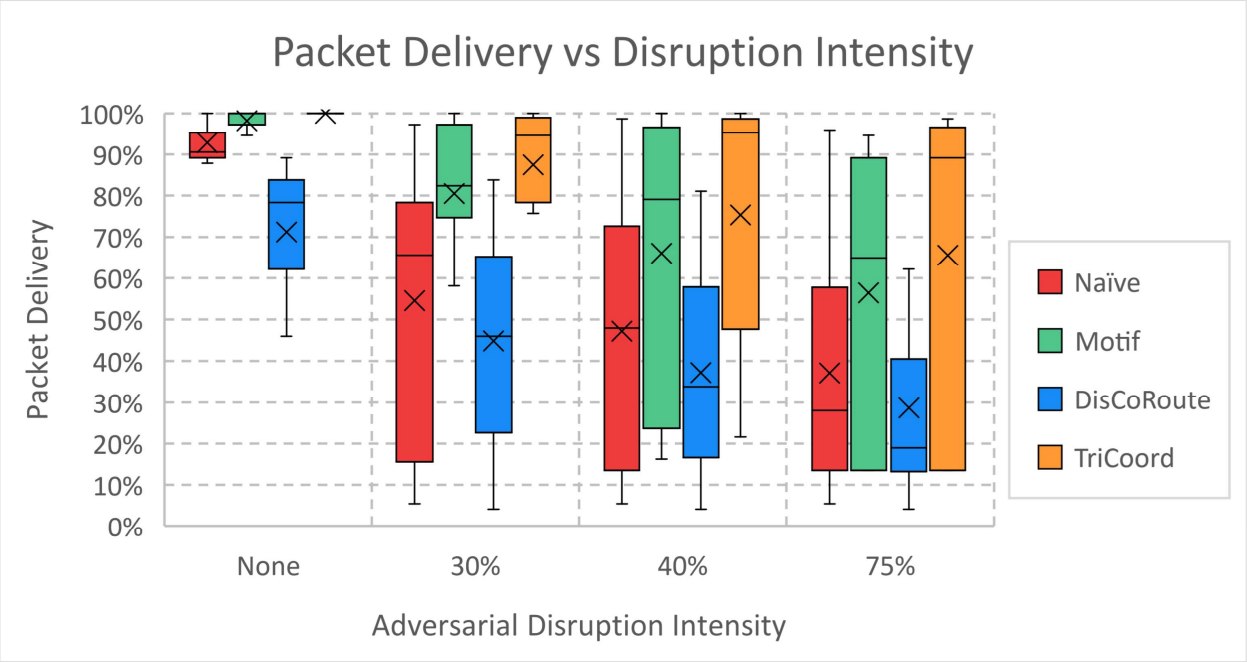


Figure 4.8: Algorithm packet delivery rates vs disruption intensity.

packet delivery rates. Self-Healing Motif-based Distributed Routing and TriCoordinate Axis Priority consistently had higher packet delivery rates, but with an increase in packet stream jitter. In fact, when comparing average packet delivery rate and jitter across all trials, these two values produce a correlation coefficient of 0.86, indicating a strong positive relationship between these two properties. This suggests increased jitter may be an inherent characteristic of volatile orbital network packet routing and should be an area of future study as validation of this potential relationship may influence future routing protocol and hardware design to reduce the potential impact to sensitive network applications.

4.6.3 The Necessity for Testing Multiple Routing Scenarios

Algorithm packet routing performance demonstrated notable variations when routing packets North/South compared to East/West. Additionally, nearly all tested algorithms had lower delivery rates routing packets East/West at higher latitudes due to the reduction in available

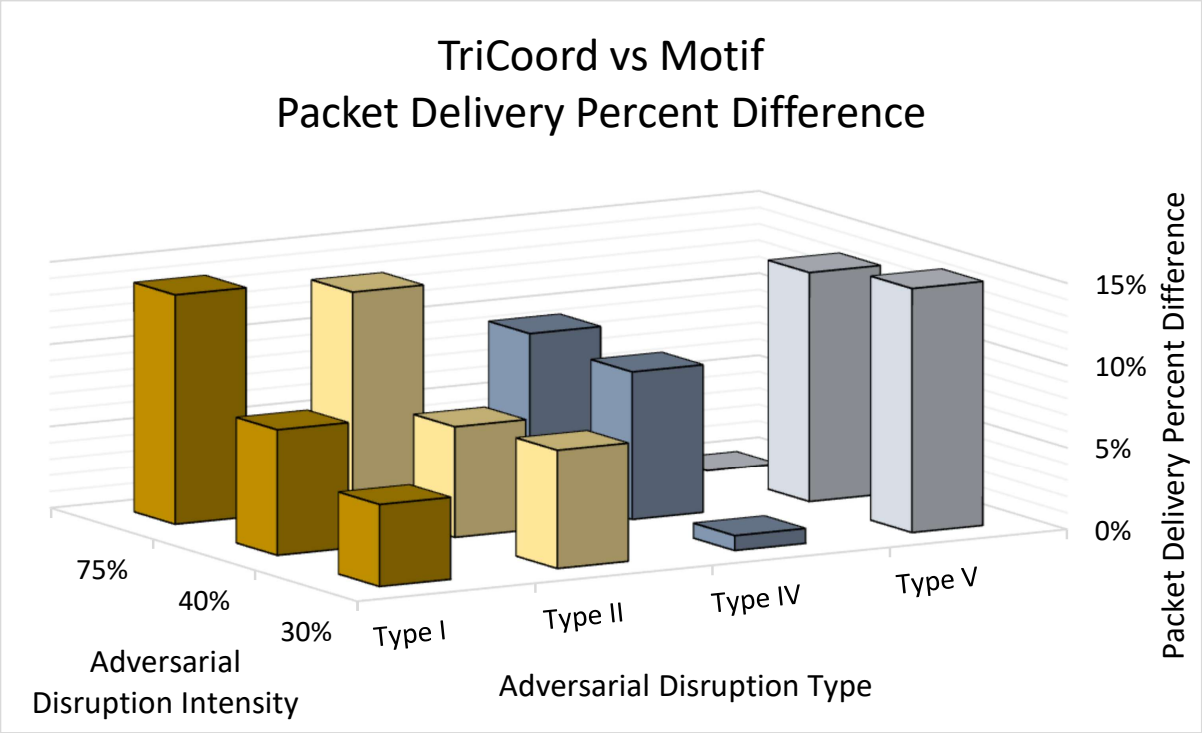


Figure 4.9: TriCoord packet delivery rate average gains over Motif-based dynamic routing across all disruption types and intensities.

ISLs inherent from constellation design. Considering the results shown in Figure 4.3, where DisCoRoute demonstrated a packet delivery rate difference of 43% between East-West High Latitude and North-South Americas scenarios, this illustrates the need of multiple packet traffic scenarios to ensure sufficiently representative results are collected in resilience analysis. Future LEO mega-constellation routing algorithm proposals should address routing under a variety of packet delivery scenarios to demonstrate real-world practicality.

4.6.4 Adversarial Disruption Models

On average, all routing algorithms had greater difficulty routing packets under disruptions to individual ISLs than disruptions to entire satellites. As shown in Figure 4.10, routing

algorithms delivered 12.25% less packets during Type I disruptions compared to Type II and 28% less packets during Type V disruptions compared to Type IV at 75% intensities. While additional investigation is necessary to further validate this observation, this indicates partial satellite disabling may produce greater disruptions to packet traffic than complete disabling. If confirmed, this finding will likely influence future adversarial disruption mitigation strategies.

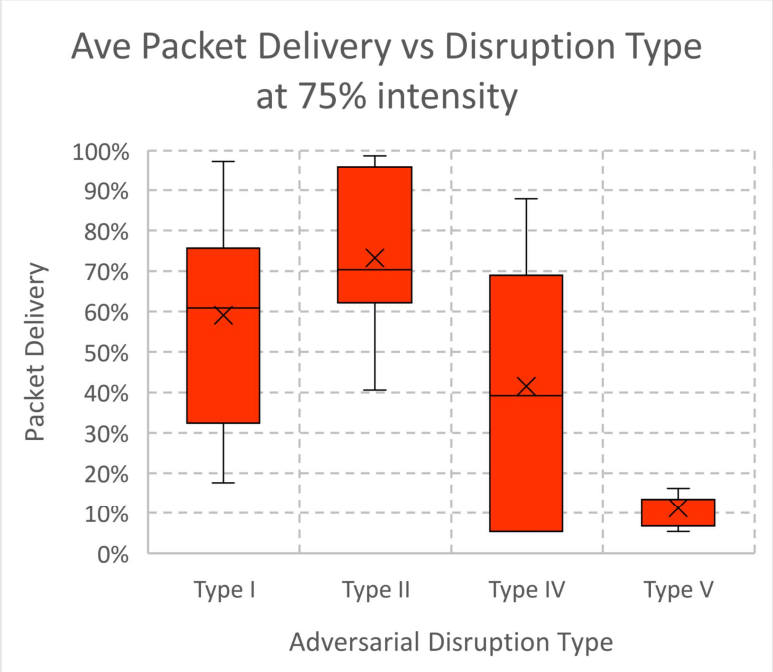


Figure 4.10: Packet delivery rate averages for all algorithms by disruption type using 75% intensity.

4.6.5 Security Analysis and Guarantees

This study was an initial investigation into the use of ISL architecture-dependent logical topologies and its benefits to dynamic routing. Several aspects of interest were not addressed and will be topics of future work. TriCoordinate Axis Priority's lack of routing table updates reduces potential attack vectors, but authentication of actionable data received is still an essential component. Adversaries may attempt spoofing ISL beacons during periods of

non-availability in order to influence packet routing. Additionally, ISL implementations and constellation designs that differ from the analyzed example will influence the logical topology's design and performance; further investigation into poly-dimensional logical topologies is currently ongoing.

4.7 Findings Summary

In summary, our findings for this study include:

- TriCoordinate Axis Priority routing demonstrated greater resilience to a variety of adversarial disruptions and intensities, with comparative performance increasing at higher levels of disruption intensity. Maximum observed packet delivery was over 12% higher than the next highest algorithm during periods of maximum adversarial disruption intensity.
- Mega-constellation dynamic routing algorithms demonstrate a correlation coefficient of 0.86 between resilience and jitter, indicating a potential inherent relationship.
- Multiple testing scenarios are necessary to determine routing algorithm suitability in real-world applications. We observed up to 43% difference in packet delivery rates between scenarios prior to any adversarial disruptions.
- Disruptions that impact a portion of satellite ISL interfaces was observed to cause greater disruptions to packet delivery than disabling entire satellites, producing an additional 28% packet loss at the highest intensities.

Chapter 5

Secure and Energy-Efficient Cryptography for Cislunar and Deep Space Platforms

In this chapter, the key contributions include:

- A novel application of using pre-generating one-time pads (OTPs) to significantly reduce energy consumption during AES-CTR encryption on microcontroller and FPGA-based cislunar spacecraft.
- A comprehensive analysis demonstrating measurable energy and cost benefits by comparing the OTP pre-generation approach against conventional AES-CTR methods without OTP pre-generation.
- An in-depth examination of intermittent computing methods for performing cryptographic operations on spacecraft, evaluating the potential energy savings versus traditional continuous execution models.
- Detailed quantification of peak power demand reductions achievable via intermittent computing.

Novelty and Significance This chapter presents two key methodological innovations addressing critical power and energy constraints on spacecraft: energy-opportunistic pre-

Metric	Cryptographic Operation	FPGA OTP Pre-Gen (Ours)	Standard AES-CTR	Intermittent Computing (Ours)	Continuous Execution
On-Demand Execution Time Reduction (%)	AES-CTR Encryption	**95.7-99.5%**	Baseline	-	-
Energy Budget Consumption Reduction (%)	AES-CTR Encryption	**98.2-99.8%**	Baseline	-	-
Est. Peak Power Demand Reduction (relative)*	Signature Generation (all schemes)	-	-	**53%**	Baseline
	Signature Verification (all schemes)	-	-	**71%**	Baseline
	Key Pair Generation (RSA, DSA, DH)	-	-	**79%**	Baseline
Suitability for Cislunar/DTN Platforms	Resource Constraints	**Highly Suitable**	Unsuitable	**Highly Suitable**	Unsuitable

* Estimated by Ex-Situ analysis results at highest key strength assessed for a given cryptography scheme.
Note: Energy savings in OTP pre-generation and intermittent computing depend on specific cryptographic operations and execution platform. OTP pre-generation significantly reduces real-time encryption workload, yielding high operational savings ideal for constrained cislunar spacecraft systems. Energy-reactive checkpointing is estimated to notably reduce peak power consumption of select operations for select schemes, providing substantial total energy reduction benefits, particularly suitable for irregular power availability and strict energy budgets of deep space.

Table 5.1: Comparative Performance of Energy-Efficient Cryptographic Approaches

generation of one-time pads (OTPs) for AES-CTR encryption and energy-reactive checkpointing for cryptographic operations. Unlike traditional cryptographic execution, which are continuous and thus incur high peak power and energy consumption, energy-opportunistic OTP pre-generation significantly shifts computational loads to periods of high energy availability. Concurrently, energy-reactive checkpointing innovatively harnesses inherent operational downtime to perform cryptographic tasks incrementally, greatly reducing estimates of peak energy consumption. These methods represent the first comprehensive analysis clearly quantifying the energy, power, and operational cost advantages of such techniques specifically tailed to resource-constrained, deep-space and cislunar missions.

5.1 Introduction

Spacecraft operating in cislunar space and beyond typically have mission lifetimes up to a decade or more without the possibility of physical servicing. Whatever data collection or

service performance that warrants such engineering feats and monetary investments automatically flags their scarce data streams - which may be intercepted by unintended third parties - as highly valuable [38]. These spacecraft face all of the challenges outlined in Section 2.3, along with increasing energy harvesting difficulties as these spacecraft travel further from the Sun (due to the inverse square law for electromagnetic radiation), elevating the importance of efficient, resilient cryptographic measures [24]. Cryptographical solutions on current spacecraft often use proprietary hardware modules that minimize power consumption, but complicate periodic updates with firmware patches that run the risk of unrecoverable component failure [108]. Even with implementations that seek to minimize this risk, the introduction of a possibly unrecoverable fault is still significant. The risks of firmware updates can be avoided by implementing cryptographic operations in software; however, the complexity of these schemes typically requires more power/energy to perform than is permissible within spacecraft design constraints. To address this, we propose leveraging *intermittent computing* (IC) for software-implemented cryptographic operations.

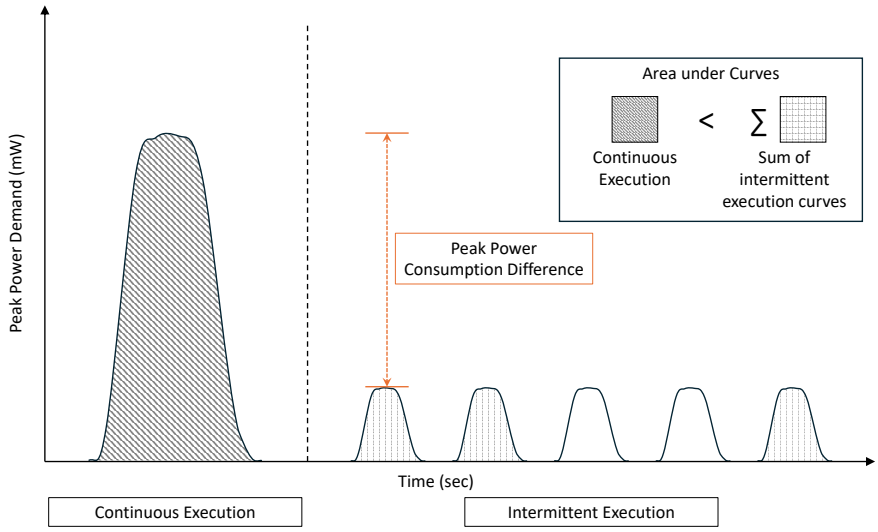


Figure 5.1: Illustration of instantaneous power demands (peak power) over time for continuous and intermittent computational execution modes. Continuous execution requires higher sustained peak power, but with lower total energy consumption (area under curve). Intermittent execution requires significantly less peak power, but with higher overall energy consumption due to additional overhead and extended duration. Reduction of peak power requirements enables task progression under limited battery reserves.

Intermittent computing can significantly reduce the spacecraft's peak power demand to execute computationally heavy tasks, enabling operations to proceed incrementally using limited battery reserves, as illustrated in Figure 5.1. However, this execution mode typically increases the total energy consumption compared to continuous execution due to overheads associated with storing and retrieving partial execution data. Despite the increased cumulative energy expense, intermittent computing remains advantageous for spacecraft scenarios where the energy source (solar power) is abundant over time but local energy storage capacity (battery reserves) and the rate of energy collection (harvesting by solar panel arrays) is significantly constrained.

Previous proposals addressing the energy consumption of computationally heavy tasks on spacecraft have been already presented: As discussed in Section 2.4.2, these works either schedule homogeneous tasks for execution during predicted periods of energy availability or divide operations into small enough tasks that can be performed in parallel on 'nanosatellites'. However, such proposals do not account for heterogeneous operations or varying data size, as can be expected in deep space operations. Additionally, the proposal of Bleier et al. to use micro data centers [17] in space simply shifts the task load to centralized processors and the deployment of sufficient systems to adequately support sparse and frequently segmented deep space DTNs would likely be prohibitively expensive.

Through the use of IC, the occurrences and execution patterns of heavy computational tasks associated with software-based cryptographic operations can be closely aligned with the operational constraints of space missions, such as power availability. This permits the use of flexible, adaptable, and secure software-based cryptographic libraries within deep space operational and cost constraints, without the risks and expense associated with hardware-based cryptographic solutions.

We explore two different approaches of IC under different mission scenarios to demonstrate use cases in which IC-supported software-based cryptography can be used to satisfy mission

requirements. These selected scenarios merely 'set the stage' for the potential benefits of using IC and do not encompass all scenarios for which the proposed solutions may be applied.

To perform energy-efficient software-based cryptographic operations on cislunar and deep space spacecraft using intermittent computing, we propose the following approaches:

- Energy-opportunistic precomputation: Pre-calculating a cache of "one-time pads" during periods of high energy availability to reduce the overall energy storage requirements related to encryption execution on cislunar spacecraft.
- Energy-reactive checkpointing: Performing "Just-in-Time" process checkpoint and restoring to execute computationally-heavy cryptographic operations intermittently to reduce the instantaneous energy consumption on deep space platforms.

For cislunar spacecraft, where only some data latency may be tolerated, we perform *energy-opportunistic precomputation* by leveraging a technique previously proposed for terrestrial IoT sensors that pre-computes a cache of one-time pads (also known as encryption coupons) 'offline' when energy is plentiful, storing them until they are consumed during data encryption [134]. The material presented for this approach is greatly expanded from its original original presentation in [68]. While additional experimentation has not been performed, the analysis and discussion resulting from the original data has been enhanced.

For deep space platforms, we utilize *energy-reactive checkpointing* by performing "Just-in-Time" execution state captures of unmodified binaries when energy reserves are nearly depleted, then restoring execution on demand at a later time when energy reserves are replenished or energy is abundant. This approach allows the use of robust, adaptable, and updatable software encryption algorithms - such as those standardized and certified by NIST - while drastically reducing instantaneous power requirements compared to continuous execution of software-based cryptography, with the additional benefit of mitigating mission-threatening risks of hardware module malfunctions due to firmware updates. Energy-reactive check-

pointing does increase task latency, but it is not expected to impact mission requirements as deep space is a latency-laden environment and such delays will already be a component of a Delay-Tolerant Network's (DTN) mission design.

All code used to generate data for this study is located [70].

5.2 Security Considerations of Intermittent Computing

Security considerations regarding the use of intermittent computing, particularly when used for cryptographic operations, centers around physical access to the given device [124]. Fortunately, this is where the lack of physical access to cislunar and deep space platforms serves as a positive trait where this concern is reasonably made invalid.

Additionally, as cislunar and deep space missions generally continue for a minimum of several years, the necessity of selecting cryptographic schemes suitable for such long durations guides mission planners to consider newly standardized post-quantum cryptographic schemes [103]. However, as much of the research regarding the impacts of quantum computing on cryptography is still theoretical, NIST acknowledges inherent uncertainties regarding the protections these schemes provide, as can be seen in NIST's post-quantum security categories [93]. As will be discussed in Section 5.5.2, we are including several of NIST's newly standardized post-quantum cryptographic schemes in our analysis, ML-KEM and ML-DSA, as well as pre-standardized versions of SLH-DSA and Falcon [100, 101, 102]. As with any newly standardized cryptographic schemes, it is reasonable to assume a period of frequent revisions as additional security and real-world deployment analyses dictate [64, 95]. Given the tumultuous nature of current and near-future cryptography, adaptable and flexible approaches are crucial to maintaining sufficient data protection over long periods of time. Such malleable approaches are best implemented in software, with the rigid constraints and risky updates

of hardware implementations likely to become covert and understated threats to spacecraft security.

5.3 Energy-Opportunistic Precomputation in Cislunar space

Traditional cislunar and deep space missions typically categorize communication security as a supplemental requirement and is delegated as a sub-component to the communications module. Under this approach, the security of transmitted data is considered and selected only after a host of other design decisions have occurred, such as the spacecraft bus and power system. As such, cryptographic mechanism selection becomes a design decision primarily driven by cost and energy expense.

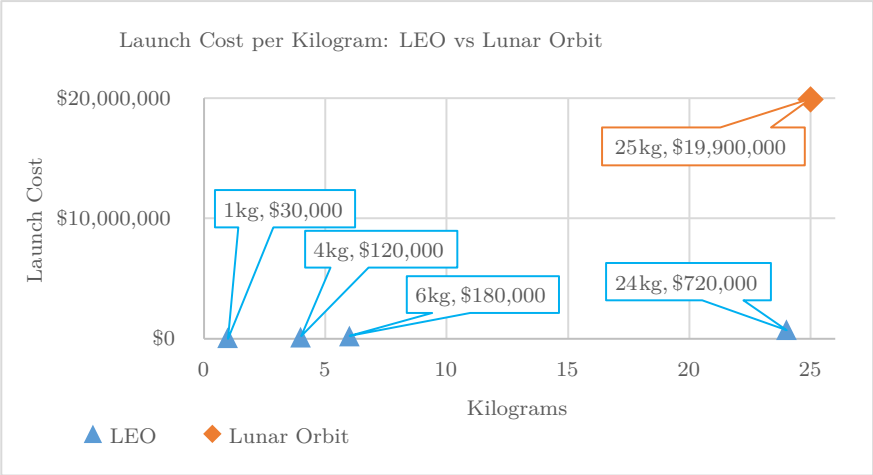


Figure 5.2: Estimate of spacecraft launch costs by weight.

5.3.1 Energy-Opportunistic Security Framework Description

In this work, we proposed a spacecraft design framework that incorporates security as an initial system design component to save costs and improve overall operational efficiency.

In its entirety, our proposed framework incorporates typically disparate spacecraft components into a complimentary, security-focused configuration. By combining offline encryption token caching with low-power transmitters, efficient power budgeting, and a supporting infrastructure, such as NASA's proposed LunaNet, we intend for this framework to guide future cislunar spacecraft designs that adopt industry-standard security encryption while simultaneously reducing overall power budget costs and subsequent launch costs (at the time of writing, NASA has not yet published security standards for LunaNet, but do state that data confidentiality is a requirement [98]).

5.3.2 Encryption-Coupon Caching

A key element of this framework is leveraging an IoT energy harvesting technique of precomputing 'offline' encryption-coupons, or one-time pads, that translates energy storage requirements into data storage requirements [134]. As in IoT devices, data storage on spacecraft is commonly cheaper than energy storage, with the additional benefit of data storage mediums also being considerably lighter than energy storage mediums. Spacecraft mass is a direct contributor to launch costs, increasing significantly the further from Earth a spacecraft mission resides.

'Offline' Encryption-Coupon Caching is a technique that can be used when future data will need to be transmitted securely, but the value of that data is not yet known. Rather than applying cryptographic operations on plain text to derive the cipher text, as done in typical schemes, the portion of cipher text that is not dependent on the plain text is pre-generated and cached as a one-time pad to be combined with plain text at a later time; this approach generates cipher text on demand with much less computational effort once the data becomes available. This technique can be performed using AES in either Counter Mode (CTR) or Galois/Counter Mode (GCM), where a portion of the cipher text is generated by combining the private key with the appropriate initialization vector value. The resulting one-time pad

(or 'encryption coupon') can be stored for later use and simply XOR'ed with the plain text data just prior to transmission.

As storing one-time pads and retrieving them later will consume more energy than simply generating the values when needed, the material benefit is gained when employed on "energy harvesting" devices. Spacecraft, and some IoT devices, harvest energy using solar panels and use it to perform operations or store in a battery for later use. Periodically, these devices are in a situation where their energy storage mediums are full, but energy is still available to be harvested; without any work to be performed, this "excess energy" goes unutilized. Generating encryption-coupons during these periods of "excess energy" allow for the device to perform work towards data encryption without consuming energy from the battery, thereby supplementing energy storage with data storage. Figure 5.3 provides an illustrative example of time periods when encryption-coupons could be generated without subtracting from the mission power budget.

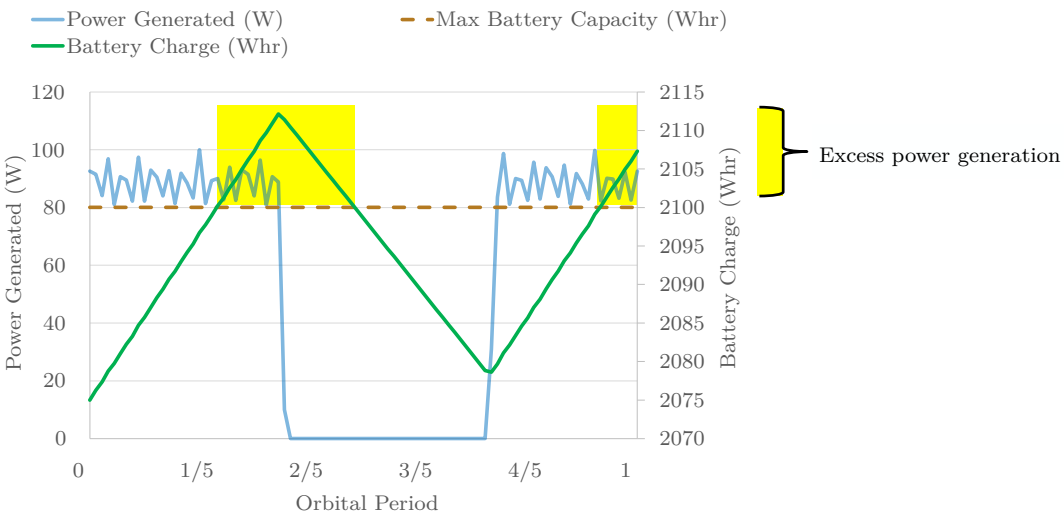


Figure 5.3: Illustration demonstrating battery level during periods of solar collection and discharge. Periods of time when additional energy can be harvest after the battery is at maximum is highlighted.

5.3.3 Encryption-Coupon Caching Evaluation

The applicability of encryption-coupon caching on cislunar spacecraft was evaluated through estimating energy savings for two 'flight-heritage' processors, the MSP430FR5969-SP and the RTG4 FPGA, using two methods of AES implementation.

The Texas Instruments MSP430FR5969-SP microcontroller has been used on numerous space missions such as Spacemanic's onboard computer and the Agile Micro Satellite payload controller. The authors of [134] demonstrated energy savings of encryption-coupon caching using a Texas Instrument MSP430FR5994 microcontroller, which is closely related to the MSP430FR5969-SP, and whose results can be representative for both microcontrollers.

For comparison, we also implemented encryption-coupon caching on an FPGA processor for analysis. FPGAs are often utilized on spacecraft due to naturally higher radiation tolerances as a result of their manufacturing compared to microcontrollers, as well as their ability to be fully reprogrammed on the fly, potentially extending the operational lifespan of a spacecraft through reconfiguration follow-on missions. The FPGA evaluation was performed on a Microchip IGLOO2 M2GL010T FPGA, which is architecturally similar to Microchip's space-hardened RTG4 FPGA. IGLOO2 M2GL010T power estimates were collected using Microchip's SmartPower v2023.2 software with generated .vcd files for increased accuracy.

The MSP430 microcontroller and IGLOO2 FPGA both support vendor-provided hardware-accelerated AES along with standard software implementations. Both modes of AES implementations were included in the evaluation.

The amount of power saved by implementing coupon-caching for AES-CTR to encrypt 1 MB of data in one second is listed in Table 5.2.

Processor	**Implementation**	**On-Demand Execution Time Reduction** (%)	**Energy Reduction** (%)	**Power Savings** (Watts)
MSP430FR5994	Software AES	-	96.4%	1.93
	Hardware AES*	-	51.1%	0.07
IGLOO2 M2GL010T	System Services AES	99.5%	99.8%	1.22
	CoreAES128 Macro*	95.7%	98.2%	0.15

Table 5.2: Power savings from encryption-coupon caching using AES-CTR mode to encrypt 1MB of data per second. Implementations marked with (*) are hardware accelerated.

Processor	MSP430FR5994		IGLOO2 M2GL010T	
AES Implementation	**Hardware**	**Software**	**Hardware**	**Software**
Battery Cost Savings	$126.99	$3,501.21	$252.12	$2,213.20
Launch Cost Savings	$1,671.60	$46,088.40	$3,582.00	$29,133.60
Total Cost Savings	$1,798.59	$49,589.61	$3,854.12	$31,346.80

Table 5.3: Cost Savings for ten 1MB transmissions per hour calculated as reduction in battery expense from reduced wattage and reduction in launch expense from reduced mass.

5.3.4 Encryption-Coupon Caching Discussion

While the amount of energy reduction identified in Section 5.3.3 appears minute, mapping these savings to an energy budget allows us to quantify the impact. For a basis for comparison, we considered a generalized spacecraft design and mission profile with the following features (modeled from the CAPSTONE lunar cubesat mission):

- 12U cubesat with 24 kg mass (maximum per CDS Rev14).
- Average bus/payload power load: 40W.
- 100W peak solar power generation.
- Battery capacity (occupying 6U/3.5kg): 2100 Watt-hours.
- Positioned in a Lunar circular orbit at 50km altitude with an orbital period of approximate 113 minutes (a circular orbit was modeled for simplicity).
- Transmitting an average of ten 1MB messages per hour.

Power generation and consumption estimates are made using the Isispace Group 12U cubesat bus [57]. Cost estimates are made using the TITAN-1 350Whr cubesat battery matrix [86] and launch pricing from [52]. The frequency and size of spacecraft transmissions is dictated by individual mission parameters, so we consider a conservative scenario in which a cislunar spacecraft is performing ten 1MB transmissions per hour. As shown in Table 5.3, mission designers can save up to nearly $50,000 through encryption-coupon caching when not paying for the use of hardware-accelerated AES operations.

5.3.5 Energy-Opportunistic Precomputation Supporting Infrastructure

In addition to encryption-coupon caching, energy efficient communications are enabled through the use of NASA's planned LunaNet infrastructure to avoid the need for direct-to-Earth (DTE) communication, which would otherwise require large antennas, powerful transmitters, and sophisticated attitude controllers. LunaNet supports S-band for its relay communication network, permitting lunar orbiting spacecraft to transmit low-energy messages to orbiting LunaNet Service Providers (LNSP) for eventual retransmission to Earth through the LunaNet infrastructure. In the event of LNSP nonavailability, spacecraft may also transmit messages to surface Lunar Communication Terminals (LCTs) [97].

5.4 Energy-Reactive Checkpointing Design Description

For deep space missions, where all energy consumption is at a premium and data latency tolerances are inherently high, we consider the application of Energy-Reactive Checkpointing to software-based cryptography for all cryptographic operations.

To assess the effectiveness and suitability of energy-reactive checkpointing, we measure dif-

ferences in instantaneous energy consumption of cryptographic operations under continuous execution and intermittent computing. This measurement is performed to estimate the reduction of instantaneous energy consumption needed to make operational forward progress. To simulate limited energy reserves, we constrain CPU usage to predetermined time "thresholds," allowing us to observe task performance across various levels of energy availability. This approach highlights any potential peak power energy savings that might arise from using intermittent execution. Because the class of spacecraft this study targets is still emerging (as seen by the current variety of commercial spacecraft processors [2, 3, 139, 140]), we deemed taking direct energy measurements on current "analogous" platforms to be premature. Instead, we execute specific cryptographic operations using both continuous and intermittent execution methods and compare the execution performance of both. This approach yields performance insights that should be broadly applicable to future deep space computing platforms.

5.5 Energy-Reactive Checkpointing Measurement Approach

When considering the time periods for which storing data in spacecraft non-volatile memory costs less energy than keeping the data resident in volatile memory, we can perform a quick analysis using energy consumption rates for both components. For comparison, we selected a Samsung 4GB DDR4-2400 DIMM as our exemplar volatile memory [40] and a Micro Electronics 64Mb MRAM module as our exemplar non-volatile memory [61]. DDR4 is commonly used for system memory (generally with a portion configured for Error-Correcting Code (ECC) to account for single-event upsets) and MRAM is used to store critical data in advanced spacecraft. Using worst-case energy values provided in the component datasheets for conservative estimates, but without other component impacts (such as CPU, system bus,

etc...), we determined that energy required to read and write 8MB of data from DRAM plus read and write 8MB of data to MRAM requires only as much energy that is needed to hold the same data resident in DRAM for less than 4 seconds, even when using DRAMs lowest energy states (self-refresh 'standby'). These extremely short break-even times for spaceflight-configured systems makes the application of intermittent computing much more palatable and, under certain conditions, very desirable.

Measuring IC's effectiveness compared to continuous execution required careful consideration as existing methods used to evaluate continuous execution efficiency (e.g. cache hit/miss rates) do not account for partial task completion as performed by IC. At the same time, prior work on IC primarily focuses on contexts where continuous execution is *impossible*: as a result, there is little in-depth comparison of the two as competing approaches for executing the same software. As this study is investigating program execution in deep space, energy consumption and efficiency are our primary concerns. However, the definition of 'energy efficiency' itself is not uniform across all domains and manufacturers [133]. To fill this gap, we define our own framework named Intermittent Cryptographic Circum-Architectural Performance Evaluations (IC-CAPE) to explore the reduction in instantaneous energy consumption (if any) of performing cryptographic operations using IC instead of continuous execution.

5.5.1 IC-CAPE Description

The IC-CAPE cryptographic operation performance analysis framework is used to assess whether a particular cryptographic operation could potentially reduce the amount of instantaneous energy required for execution through the use of intermittent computing. No assessment or evaluation is made on the security properties of cryptographic schemes and operations selected for evaluation.

IC-CAPE measures cryptographic execution under two conditions: using micro-architecture-

dependent event counts (in-situ) and using CPU time of the target process (ex-situ). These separate metrics are intended to contrast and compliment each other to gain a more complete understanding of IC's impact on program execution: In-situ measurements are useful for understanding performance characteristics in the context of native architectural penalties. Ex-situ measurements produce general execution profiles for reasonable comparison across multiple CPU architectures. Through the optics of both metric lenses, mission designers can determine the general amount of impact (ex-situ) and whether such impacts derive from straight computational effort or micro-architectural bottlenecks (in-situ).

In-Situ Measurement

IC-CAPE performs in-situ measurements to observe execution differences between intermittent and continuous execution within a single environment. The influence of instruction execution, along with CPU architecture, on CPU energy consumption has been long-established [63]. As stated earlier, typical efficiency measurement methods, such as cache hit rates, do not account for partial forward-progress, making them insufficient for the purposes of our study. Our measurement approach uses occurrences of specific system events that negatively impact execution efficiency (the process 'effort') along with the amount of storable output that is produced during execution (the process 'work'). This value is normalized for processor frequency variation and labeled as our 'Work-to-Effort' ratio (R), defined in Equation 5.1 with the uncertainty of R calculated using Equation 5.2. R values of both IC and continuous execution are used to calculate IC-CAPE's **Point of Inflection** (PoI). PoI values greater than 0 indicate the maximum CPU Time Threshold where IC provides a benefit, where PoI values of 0 indicates IC provides no benefit at any CPU Time Threshold.

A holistic view of execution performance is calculated as an aggregate Work-to-Effort ratio $R_{aggregate}$ and uncertainty $\sigma_{aggregate}$, defined in Equations 5.3 and 5.4, using weighted contributions of all collected events. Weight values are assigned to be roughly analogous to the

significance of impact each event occurrence has on CPU energy consumption through stalled execution, with assigned weight values listed in Table 5.4 and guided by recent discussions on the topic [133, 153] and general/historical knowledge [15, 18]. We temper the weights of highly penalizing events (e.g. Page Faults) with the expected use of aggressive energy-saving OS and system architecture features that reduce processor energy consumption during periods of long CPU delay. Our assigned weight values constitute a preliminary selection for initial investigation and can be refined in subsequent studies.

Event	**Weight**
Page Faults	5
LLC Load Misses	5
Branch Misses	4.5
L1-D Cache Load Misses	4
dTLB/iTLB Load Misses	4
L1-I Cache Load Misses	3
L1-D Cache Prefetch Misses	3
General Cache Misses	3
L1-D Cache Loads (Hits)	2
L1-I Cache Loads (Hits)	1

Table 5.4: Weighted event costs with respect to processor energy consumption. Weight assignments assume events with longer idle periods permit processors to partially-mitigate energy consumption through reduced-energy states. Shorter waits do not permit state changes to mitigate their power consumption, thereby narrowing the range of energy costs for the listed events.

$$R = O \cdot \frac{C}{E} \tag{5.1}$$

$$\sigma_R = R \cdot \sqrt{(\frac{\sigma_C}{C})^2 + (\frac{\sigma_E}{E})^2} \tag{5.2}$$

Where:

O**:** Output-per-trial (amount of total output produced / number of trials); $\varnothing \leq O \leq 1$.

C**:** Mean CPU cycles.

E: Mean event count.

$\frac{C}{E}$: indicates execution 'efficiency' with respect to penalizing events E; higher values indicate better efficiency

σ_C: Standard deviation of CPU Cycles.

σ_E: Standard deviation of event counts.

$$R_{aggregate} = \frac{\Sigma_i(w_i \cdot R_i)}{\Sigma_i w_i} \tag{5.3}$$

$$\sigma_{aggregate} = \sqrt{\sigma_C^2 + \frac{\Sigma_i(w_i \cdot \sigma_{E_i}^2)}{\Sigma_i w_i}} \tag{5.4}$$

Where

R_i: Work-to-Effort ratio for event i.

w_i: Weight assigned to event i.

σ_{E_i}: Standard deviation of event i.

σ_C: Standard deviation of shared CPU cycle count (applied only once).

Ex-Situ Measurement

IC-CAPE complements micro-architecture dependent analysis with task-centric execution metrics that permit a comparison of execution efficiency between intermittent and continuous executions across different CPU architectures (ex-situ measurements), using a processes' utilized CPU time as the sole metric. CPU energy consumption dominating system power

for CPU-bounded workloads has been previously established, along with the CPU's continue influence on power consumption even when idle [51] CPU time was collected as the combination of perf's *user_time* and *system_time* events [43].

A target operation's consumed CPU time is collected after task completion or termination, along with the CPU time consumed by any assisting background processes for IC execution. IC-CAPE then uses equations 5.7-5.8 to estimate a relative energy consumption value for a single IC execution cycle.

Individual Handler Operation Cost Modeling To model the per-operation CPU time cost $T_{op}(n(t))$, for both the target operation and any checkpoint handler process, as a function of the total number of operations n, we perform a curve-fitting procedure on empirical CPU time measurements. For each operation-scheme combination, we apply nonlinear least squares minimization using Python's scipy.optimize.curve_fit to fit both a power-law model and a log-logistic model to the data, providing appropriate initial guesses for each. Once parameter sets for both curves were obtained, we calculated R^2 to measure goodness of fit and used the curve with the higher R^2. The selected curve was then used as the basis for computing the per-operation cost for that operation-scheme combination. The initial guesses used to calculate fit-curves are:

- Power-Law: $A_0 = 1e10$; $B_0 = 1$
- Log-Logistic: $A_0 = max(y)$; $B_0 = median(x)$; $C_0 = 1$

CPU times with the shortest execution cycles that produce forward progress for intermittent and continuous execution are used to calculate IC-CAPE's **Forward-Progress Cost** (FPC) ratio. FPC ratio values of **1 or greater** indicates no benefit is provided by IC, where FPC ratio values **less than 1** indicating potential benefits of varying significance from IC.

$$T_{cycle}^{continuous}(t) = T_{task}(t) \tag{5.5}$$

$$T_{cycle}^{intermittent}(t) = T_{task}(t) + T_{handler}^{idle} + \begin{cases} T_{op}(n(t)), & \text{if } n \leq 2 \\ 2 \times T_{op}(n(t)) & \text{if } n > 2 \end{cases} \tag{5.6}$$

$$T_{op}(n(t)) = \frac{T_{handler}^{raw}(n(t)) - T_{handler}^{idle}}{n}$$

$$T_{min}^{continuous} = \min_{t \in T} \left(T_{\text{cycle}}^{\text{intermittent}}(t) \right) \tag{5.7}$$

$$T_{min}^{intermittent} = \min_{t \in T_{\text{reliable}}} \left(T_{\text{cycle}}^{\text{continuous}}(t) \right) \tag{5.8}$$

$$\text{Forward-Progress Cost (FPC) Ratio} = \frac{T_{min}^{intermittent}}{T_{min}^{continuous}} \tag{5.9}$$

$$\sigma_{T_{min}^{cont}} = \sigma_{T_{task}}$$

$$\sigma_{T_{min}^{IC}} = \sqrt{\sigma_{T_{task}}^2 + \sigma_{T_{handler}}^2 + (k \cdot \sigma_{T_{op}})^2}, k = \begin{cases} 1, & n(t) \leq 2 \\ 2, & n(t) > 2 \end{cases}$$

$$\sigma_{FPC} = FPC \cdot \sqrt{\left(\frac{\sigma_{T_{min}^{IC}}}{T_{min}^{IC}} \right)^2 + \left(\frac{\sigma_{T_{min}^{cont}}}{T_{min}^{cont}} \right)^2} \tag{5.10}$$

Where:

T_{task}**:** CPU time taken to execute assigned task.

$T_{cycle}^{continuous}(t)$**:** Total CPU time per cycle for continuous execution at threshold t.

$T_{cycle}^{intermittent}(t)$**:** Total CPU time per cycle for intermittent execution at CPU time threshold t.

$T_{handler}^{idle}$**:** CPU time for the intermittent execution handler process when not performing operations (idle).

$T_{op}(n(t))$**:** CPU time required for a single intermittent execution operation (checkpoint or restore) for CPU time threshold t. Obtained by fitting CPU-time measurements to

power-law/log-logistic models using scipy.optimize.curve_fit and selecting best fit based on R^2.

$n(t)$**:** Number of intermittent execution operations performed for task completion for CPU time threshold t.

T**:** The set of all CPU time thresholds considered,
$T = \{500, 750, \ldots, 9750\}$ ms.

$T_{reliable} \subseteq T$**:** The subset of CPU time thresholds where continuous execution reliably produces output.

$\sigma_{T_{task}}$**:** Standard deviation of CPU time utilized during task execution.

$\sigma_{T_{handler}}$**:** Standard deviation of CPU time utilized by IC handler during task execution.

$\sigma_{T_{op}}$**:** Standard deviation of CPU time taken for single IC operation with 'n' operations.

Uncertainty Measurement

Capturing the variance for both in-situ and ex-situ evaluations is crucial for accurately interpreting IC-CAPE results. As the in-situ PoI is determined through linear interpolation, direct error propagation methods for variance estimation are not feasible. Instead, we employ a Monte Carlo simulation to estimate variance, providing statistically robust insight into the variability of in-situ results. The Monte Carlo simulation parameters used to estimate the variance for our collected data are:

- Number of simulations: 2,000
- Zero offset: 100 (1% of typically lowest aggregate Work-to-Effort ratio)
- Fixed seed value (for reproducibility): 42

Variance for ex-situ measurements, which is computed directly from collected execution times, allows for straightforward uncertainty propagation using Equation 5.10.

5.5.2 Cryptographic Scheme Selection

The National Institute of Standards and Technology (NIST) is the U.S. government agency that develops technology standards used in government and commercial industries and is the leading publication agency that standardizes and certifies cryptographic schemes appropriate for specific uses. Per NIST guidelines, appropriate algorithm selection and key size must provide protection for the lifetime of the data *and* the lifetime of the system [13]. As manufacturing and launching spacecraft intended for beyond cislunar exploration are considerable monetary investments, the life expectancy of any such vehicle is also likely to be considerable [41]. As such, we selected NIST-standardized algorithms and key sizes that reflect this timescale, with the addition of several 'traditional' cryptographic schemes for comparison and completeness.

For evaluation, we selected Elliptical Curve Cryptography (ECC) due to its emphasis on energy efficiency. Additionally, we included the traditional schemes RSA, DSA, and Diffie-Hellman (DH) for completeness. Utilized key sizes are listed in Table 5.5 and selected in accordance with NIST's guidelines [13]. DSA and RSA key sizes greater than 7,680 bits were omitted due to performance and compatibility limitations. This list of algorithms is not intended to be all-inclusive and merely representative for our initial investigation.

In light of advancing computational capabilities, NIST has selected RSA, DSA, Elliptic Curve DH, and other contemporary cryptography schemes for deprecation after 2030 and disallowed for use after 2035 [93] - a time period within expected mission life for many newly launched inner solar system spacecraft. As such, we expanded our analysis and evaluation to include the newly NIST-standardized post-quantum algorithms ML-KEM (for shared secret generation/encapsulation), and ML-DSA, SLH-DSA, and Falcon (for digital signing/verifi-

cation). The selected post-quantum cryptographic schemes and key sizes are listed in Table 5.6.

Recognizing uncertainties surrounding future development and capabilities of quantum computers, NIST updated its framework for estimating the maximum security strength of post-quantum algorithms to use new categories, as detailed in Section 4.1 of NIST Internal Report (IR) 8547 Initial Public Draft (published in November 2024). To aid clarity and facilitate alignment with existing standards in this work, we have adopted the following custom security labels: 'Standard', 'Enhanced', and 'Advanced'.

We consider 'Standard' to represent security suitable for current needs of non-critical data, aligning with SP 800-57's 128-bit security strength and IR 8547 Category 1. 'Enhanced' reflects security intended to address near-future requirements or critical data protection, corresponding to SP 800-57's 192-bit security strength and IR 8547 Category 3. 'Advanced' reflects security meeting the highest foreseeable security demands for long-term protection of highly critical data, aligning with SP 800-57's 256-bit security strength and IR 8547 Category 5.

To reiterate, these labels are only intended to aid readers with relating to the new post-quantum security categories established and assist with the understanding and application of these novel algorithms.

SLH-DSA, as specified in FIPS 205 [102], was not available for testing and experimentation in Open Quantum Safe, the development library used in this work, at the time of publication; the foundational implementation, SPHINCS+ v3.1, was substituted for this study while using the security parameters outlined in FIPS 205. SLH-DSA/SPHINCS+ includes two modes of operation: fast and small; both of which were included in our evaluation.

The digital signing algorithm Falcon has been selected for standardization by NIST (and is considered a potentially superior selection for digital signing and verification on constrained systems), but does not yet have a supporting FIPS publication; security parameters used

Key Lengths with common labeling:	128-bits (**Standard**)	192-bits (**Enhanced**)	256-bits (**Advanced**)
DSA†	3072	8192	
RSA	3072	7680	
DH	3072	4096	
ECC	256	384	512
AES	128	192	256
Digest	SHA2-256	SHA3-384	SHA3-512

Table 5.5: Cryptographic schemes and key lengths (in bits) aligned with security strength as published in NIST SP 800-57 Pt1, Rv5 along with custom security labels to assist reader identification of roughly equivalent security ratings with NIST's Post-Quantum Cryptographic categories. Elliptic-Curve Cryptography key sizes provide the lowest accepted value in defined key size range. †Note: FIPS 186-5 approves DSA for signature verification only [99]. Key sizes greater than 7680-bit are not considered for DSA/RSA/DH algorithms due to reduced performance and compatibility.

PQC Categories with common labeling:	1 (**Standard**)	3 (**Enhanced**)	5 (**Advanced**)
ML-KEM	512	768	1024
SPHINCS+†	SHA2-128	SHAKE-192	SHAKE-256
ML-DSA	44††	65	87
Falcon	512		1024

Table 5.6: Post Quantum Cryptography (PQS) algorithm parameter length (in bits) by security category aligned with NIST FIPS 203/204 and IP 8413 Update 1 along with custom security labels to assist reader identification of roughly equivalent security ratings with NIST SP 800-57 Pt1, Rv5. †SPHINCS+ was tested using parameters specified for SLH-DSA. ††ML-DSA-44 is claimed to provide Category 2 security; we align it with Category 1 for purposes of comparison in this study.

for evaluation were implemented according to PQC Standardization Process, Third-Round Status Report [5].

5.5.3 Trial Descriptions

For evaluation, we selected five cryptographic operations we consider to be essential for any near-future space network PKI, consisting of: Key pair generation, digital signing and verification, shared secret generation, and encryption/decryption.

Key Pair Generation

Key Pair Generation is the first operation performed by asymmetric encryption schemes, as it generates the public-private key pair used in all subsequent operations. Key storage and dissemination is handled by PKI key management policies. Producing keys for symmetric cryptography schemes, such as AES, can occur in a variety of ways and is generally much simpler than asymmetric key generation. For brevity, symmetric key generation was omitted from this study.

Digital Signing and Signature Verification

Digital signing validates the integrity and authenticity of transmitted messages and is critical for robust PKI implementations. We consider this operation to be a key component for deep space networks as platforms may receive messages sent by any transmitter within sight of their receiving apertures. These operations protect limited system resources by verifying message integrity and authenticity prior to committing further system resources for processing.

As digitally signing a large message is computationally expensive for contemporary asymmetric cryptography schemes, most traditional schemes require the generation of small, cryptographic message digests to be signed instead of the entire message. To maintain intended security strengths, message digests must be created using approved algorithms with hash strengths at least as strong as the signing algorithm. Hash algorithm strengths used in this study are included in Table 5.5.

Of our selected schemes, RSA, ECC, ML-DSA, SPHINCS+, and Falcon all support digital signing and verification. While DSA is capable of generating digital signatures, NIST no longer recommends it for this operation, and is included for signature verification only.

Per published specifications, NIST recommends ML-DSA and Falcon post-quantum algo-

rithms perform signing and verification operations on raw messages, rather than message digests, to maintain theoretical security strengths [48, 101].

To simulate message processing on constrained platforms, we perform digital signing and verification using a data buffer of 4MB. However, due to ML-DSA and Falcon not using a pre-digest, and do not support using fixed-length data buffers, these two algorithms execute using all system memory available.

Encryption/Decryption

Asymmetric encryption and decryption are categorically much more 'expensive' than symmetric encryption schemes, and are often utilized by PKIs to share only small quantities of data to be used by two parties to then perform symmetric encryption/decryption on the larger message data. Of our selected asymmetric algorithms, only RSA natively supports encryption and decryption. While AES supports both encryption and decryption, these operations are 'symmetric' with no performance variance between the two operations, so only one operation is performed for our study. For analysis, AES encryption is performed on data message size as outlined in Section 5.5.5.

The maximum data block size that can be used for RSA encryption/decryption depends on key size and implementation; we perform RSA encryption using Optimal Asymmetric Encryption Padding (OAEP), which accepts plaintext input with a maximum size of $(keylength - 2) * (hashlength - 2)$ bytes.

Generate Shared Secret

Generating a shared secret allows separated parties to agree on cryptographic key material without interactive communication, provided all parties possess each other's appropriate public keys. DH and ECC schemes both perform this operation. ML-KEM performs a

similar operation, but through use of encapsulation and decapsulation operations.

5.5.4 Energy-Reactive Checkpointing Implementation

Existing intermittent computation frameworks target highly constrained sensor systems running bare-metal firmware or a lightweight Operating System (OS) tailored for intermittent operation [150, 151], which lack the OS utilities and development interfaces expected on typical higher-power computing systems. For proof-of-concept on more feature-complete near-future systems, we utilize Checkpoint/Restore In Userspace (CRIU), a Linux utility widely used in numerous academic publications [7, 53, 132], to perform process checkpoint and restore operations necessary for reactive intermittent computation. As CRIU wasn't optimized for our selected task set, we are confident the values produced in our evaluation are conservative estimates, demonstrating an upper-bound on checkpoint/restore resource consumption.

CRIU imposes several computational overheads with varying levels of expense. We utilized the typical installation of CRIU that runs a background monitor daemon to orchestrate and broker checkpoint/restore operations with the host OS. We account for this background process's overhead both when idle and when performing checkpoint/restore operations. At the time of publication, CRIU's incremental checkpointing method was not recommended for widespread usage, and thus was not implemented. Incremental checkpoints are intended to increase performance of subsequent checkpoints by recording only the differential changes from the initial checkpoint image; as this feature is developed further, a re-investigation in potential benefits for energy-reactive checkpointing may be required.

The computational cost of starting and stopping the CRIU monitor service is not included in our analysis as it is only performed once during every powercycle and amortized rapidly to near-zero over the operational period of the system.

IC typically requires non-volatile storage to record intermediate progress between periods of execution. Depending on implementation, the energy costs associated with data storage and retrieval can vary considerably. Given this case, along with the availability and current use of low-energy non-volatile storage mediums like FRAM [62] and MRAM [61] in space missions, we opted to omit this energy cost from our analysis. For completeness, any specific use-case investigations should incorporate this energy expense.

5.5.5 Data Description

Given the focused intent of near-future, deep space platforms, we anticipate the use of delay-tolerant network (DTN) protocols for data transmission, specifically the Bundle Protocol [20, 46]. Under these assumptions, we anticipate large data bundles to be relayed in single bursts (primarily) between network nodes, until reaching the intended destination. As such, and given the growing size of sensor data (as indicated in [35, 76, 154]), we used file sizes of 500MB for appropriate cryptographic operations to simulate large data bundle movements in near-future delay tolerant networks. Cryptographic operations that use a large data bundle are: Digital signature generation, digital signature verification, and symmetric encryption/decryption. Additionally, as we anticipate deep space platforms to be conservative in design and prioritize redundancy and energy-efficiency, cryptographic operations operating on large data bundles do so using a 4MB data buffer.

5.5.6 Execution Environment

Evaluations were performed on an AMD64 SoC A10-8700P running Ubuntu 24.04.4 with 8GB configured for 1600 MT/s. Our test binaries were compiled with CRIU's libraries to internally initiate process checkpoint/restores through the library interface with high timing precision. This interface introduced a small, but non-zero, amount of overhead for the

No Benefit	Minimal Benefit	Moderate Benefit	Significant Benefit	Maximum Benefit
$PoI = 0$	$0 < PoI < 2000$	$2000 \leq PoI < 5000$	$5000 \leq PoI < 7500$	$7500 \leq PoI < 10000$
$FPC > 1$	$1.0 \geq FPC > 0.8$	$0.8 \geq FPC > 0.5$	$0.5 \geq FPC > 0.25$	$0.25 \geq FPC > 0.0$
Maximum Variance	High Variance	Moderate Variance	Low Variance	Minimal Variance
$CV \geq 1.0$	$1.0 > CV \geq 0.7$	$0.7 > CV \geq 0.4$	$0.4 > CV \geq 0.1$	$0.1 > CV$

Table 5.7: Intermittent Computing suitability categories for in-situ Point of Inflection (PoI) results and ex-situ Forward Progress Cost (FPC) results [top] along with Coefficient of Variance (CV) categories for IP and FPC results [bottom].

operation process itself. Evaluation is performed by monitoring process execution with the Linux 'perf' utility and collecting the system event counters listed in Appendix Table 5.4. CPU time, the metric utilized for ex-situ analysis, is a combination of perf's *user_time* and *system_time* events. Additional software utilities and libraries used to perform our evaluation are listed in Table B.1 in the Appendix.

Trials are executed 50 times for variability tracking and to permit IC forward progress through subsequent executions.

5.6 Evaluation

Following the measurement techniques introduced in Section 5.5.1, we present the trial outcomes for intermittent computing (IC) versus continuous execution across multiple cryptographic schemes, key strengths, and operations. As previously stated, in-situ results leverage micro-architecture-dependent events to assess per-operation performance within a single environment, while ex-situ results use CPU time to enable cross-platform comparisons.

Table 5.7 defines the color-coded scale we used for IC suitability and uncertainty, with results for scheme-operation combinations found suitable for using IC summarized in Table 5.8. IC suitability categories are labeled as: *No Benefit*, *Minimal Benefit*, *Moderate Benefit*,

Key Strength		Standard Strength						Highest Strength						
Metric		*In-Situ*				*Ex-Situ*			*In-Situ*				*Ex-Situ*	
Scheme:	Operation*:	Mean:	MC† Mean:	MC† Std Dev:	(CV category)	Mean:	Std Dev:	(CV category)	Key Strength:	Mean:	MC† Mean:	MC† Std Dev:	(CV category)	Mean:	Std Dev:	(CV category)
RSA	KP	778	660	54		0.48	0.34		(E)	*Max*	–	–		0.15	0.01	
	SG	2129	2021	32		0.51	0.16			3860	3662	150		0.24	0.32	
	SV	2316	2041	36		0.46	0.14			3614	3487	147		0.23	0.07	
DSA	KP	3932	3257	1677		0.16	0.10		(E)	*Max*	–	–		0.24	0.01	
	SV	2292	2139	68		0.47	0.14			3555	3488	174		0.29	0.09	
DH	KP	4539	1036	541		0.18	0.02		(E)	3034	1038	564		0.24	0.19	
ECC	SG	3923	4718	1593		0.21	0.07		(A)	5096	4989	177		0.17	0.08	
	SV	2322	2054	39		0.47	0.15			5053	4723	1028		0.21	0.12	
ML-DSA	SG	4436	5081	1161		1.08	0.10		(A)	4809	5089	1188		0.97	0.05	
	SV	4694	4686	1786		0.11	0.08			4423	4438	1453		0.48	0.21	
SPHINCS+ (fast)	SG	2477	2254	47		0.51	0.19		(A)	7047	4459	258		0.2	0.02	
	SV	2093	2017	32		0.5	0.12			8383	4738	152		0.16	0.04	
SPHINCS+ (small)	SG	2537	2506	52		0.4	0.36		(A)	6320	5967	354		0.2	0.14	
	SV	2223	2006	36		0.49	0.06			7121	4732	153		0.18	0.03	
Falcon	SG	4437	5059	1224		1.06	0.13		(A)	4478	5118	1137		1.07	0.28	
	SV	4414	5054	1098		0.84	0.21			4474	3912	705		0.48	0.19	

Table 5.8: Evaluation results for cryptographic scheme-operation combinations found suitable to use IC at 'Standard' and maximum ('Enhanced' or 'Advanced') key sizes. In-situ results are PoIs (in ms) with ex-situ results as FPC ratios; suitability and variability categories used are listed in Table 5.7. PoI values assessed as '*Max*' indicate IC outperformed continuous execution at all tested CPU Time Thresholds. Key strengths are labeled according to Table 5.5. CV - Std Dev / Mean. *Operation abbreviations: KP - Key Pair generation; SG - Digital Signing; SV - Signature Verification. †MC - Monte Carlo distribution.

Key Strength		Standard Strength							Highest Strength						
Metric		*In-Situ*				*Ex-Situ*				*In-Situ*				*Ex-Situ*	
Scheme:	Operation*:	Mean:	MC† Mean:	MC† Std Dev:	(CV category)	Mean:	Std Dev:	(CV category)	Key Strength:	Mean:	MC† Mean:	MC† Std Dev:	(CV category)	Mean:	Std Dev:	(CV category)
RSA	EA	0^1	–	–	–	24.11	4.94		(E)	0^1	–	–	–	22.25	4.19	
	DA	0^1	–	–	–	14.38	2.45			0^1	–	–	–	3.12	0.15	
DH	GS	0^1	–	–	–	9.93	1.23		(E)	0^1	–	–	–	1.94	∞^2	–
ECC	KP	0^1	–	–	–	22.62	∞^2	–	(A)	0^1	–	–	–	19.05	∞^2	–
	GS	0^1	–	–	–	23.91	5.56			0^1	–	–	–	17.68	3.81	
AES	ES	0^1	–	–	–	1.62	0.06		(A)	0^1	–	–	–	1.63	0.06	
ML-KEM	KP	0^1	–	–	–	13.1	2.36		(A)	0^1	–	–	–	14.39	2.39	
	SD	0^1	–	–	–	17.8	3.56			0^1	–	–	–	16.89	3.36	
	SE	0^1	–	–	–	14.55	2.63			0^1	–	–	–	16.12	2.76	
ML-DSA	KP	0^1	–	–	–	13.67	2.63		(A)	0^1	–	–	–	13.28	2.22	
SPHINCS+ (fast)	KP	0^1	–	–	–	12.69	2.26		(A)	0^1	–	–	–	10.77	1.47	
SPHINCS+ (small)	KP	0^1	–	–	–	5.18	0.56		(A)	0^1	–	–	–	3.41	0.3	
Falcon	KP	0^1	–	–	–	9.4	1.66		(A)	0^1	–	–	–	6.03	1	

Table 5.9: Evaluation results for cryptographic scheme-operation combinations found non-suitable for IC at 'Standard' and maximum ('Enhanced' or 'Advanced') key sizes. In-situ results are PoIs (in ms) with ex-situ results as FPC ratios; suitability and variability categories used are listed in Table 5.7. PoI values assessed as '*Max*' indicate IC outperformed continuous execution at all tested CPU Time Thresholds. Key strengths are labeled according to Table 5.5. CV - Std Dev / Mean. *Operation abbreviations: KP - Key Pair generation; EA/DA - Encryption/Decryption, Asymmetric; ES - Encryption, Symmetric; GS - Generate Shared Secret; SE - Shared Secret Encapsulation; SD - Shared Secret Decapsulation. †MC - Monte Carlo distribution.[1]At no time did IC execution outperform continuous/Monte Carlo distribution could not be generated. [2]Calculating propagated error produced DIV0; assigning as ∞

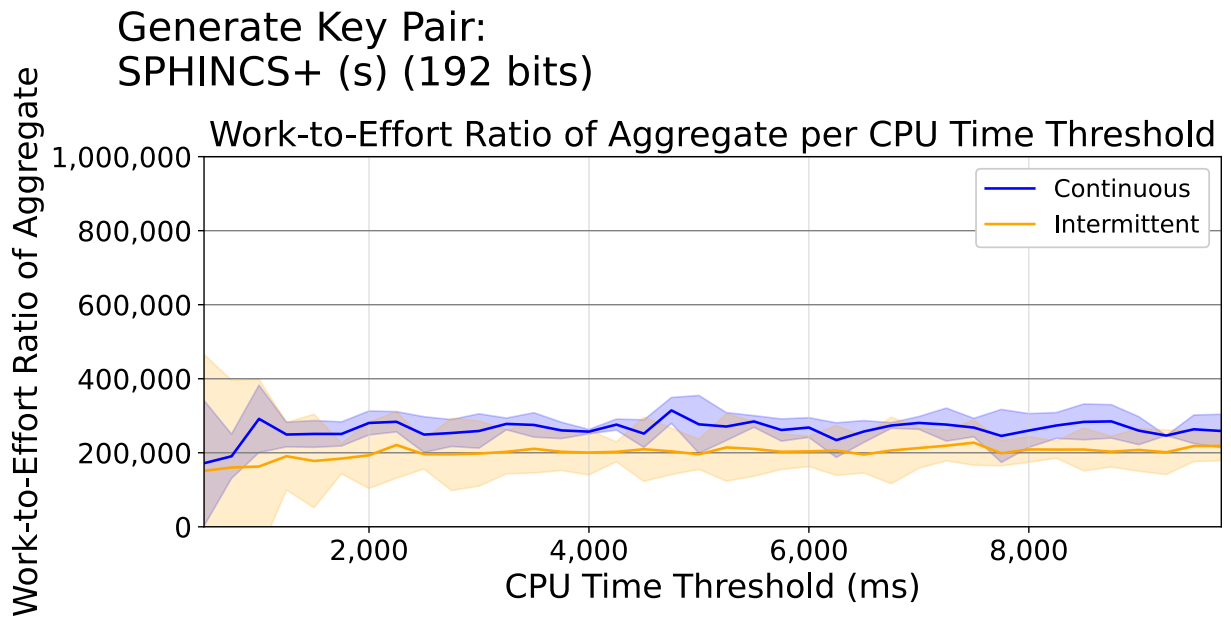


Figure 5.4: Aggregate Work-to-Effort ratio per CPU Time Threshold (in-situ measurement, from Section 5.5.1) of SPHINCS+ (small) key pair generation using 'Enhanced' key strength. Continuous execution maintains a higher aggregate Work-to-Effort ratio, indicating it outperformed intermittent computing at all evaluated CPU Time Thresholds. Solid lines denote average values with shaded regions denoting variance.

Significant Benefit, and *Maximum Benefit*. In-situ results are given as **Points of Inflection** (PoI) measured in milliseconds (ms) with ex-situ results provided as **Forward Progress Cost** (FPC) ratios. Coefficient of Variance (CV) denotes Standard Deviation divided by Mean.

Non-suitable scheme-operation combination results have been consolidated into Table 5.9. For each of these scheme-operation combinations, at no time did IC outperform continuous operations, resulting with In-Situ PoI values of 0 and prevented Monte Carlo distribution uncertainty estimates. Ex-Situ results for non-suitable scheme-operation combinations all had FPC values greater than 1 and typically also had uncertainty values greater than 1. Select cases had uncertainty less than one or resolved to infinity due to the calculations dividing by 0.

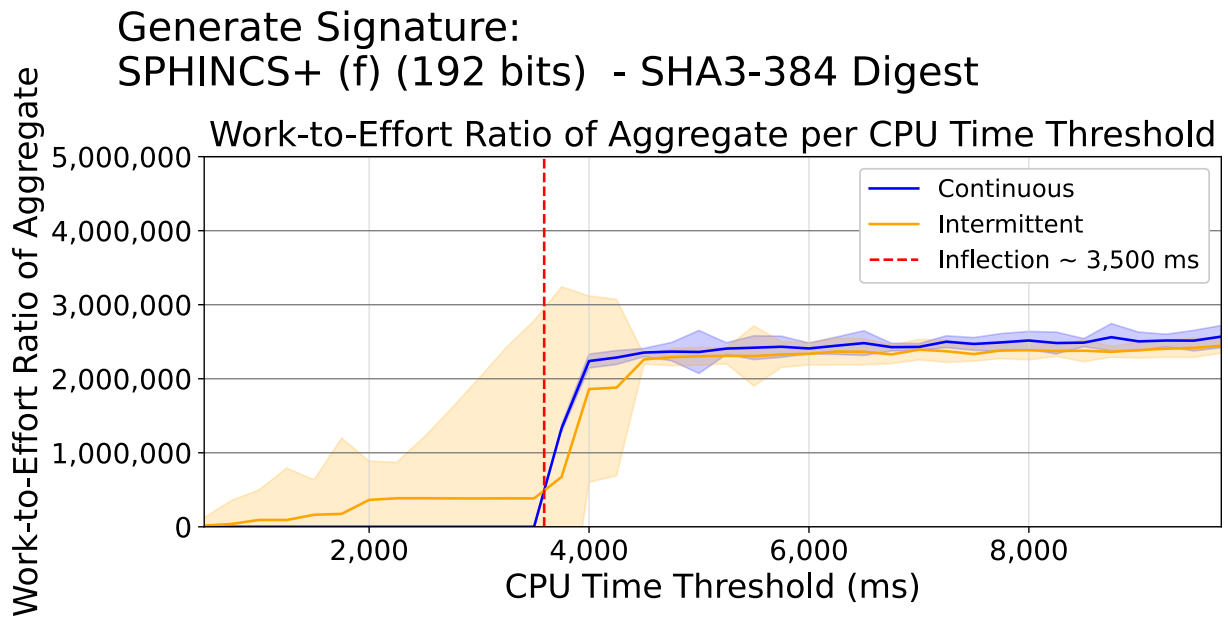


Figure 5.5: Aggregate Work-to-Effort ratio per CPU Time Threshold (in-situ measurement, from Section 5.5.1) of SPHINCS+ (fast) digital signature generation using 'Enhanced' key strength and 'Enhanced' message digest with PoI of 3595 ms. Solid lines denote average values with shaded regions denoting variance.

5.6.1 In-Situ Evaluation

We calculate a *Work-to-Effort* ratio for each cryptographic operation under a range of CPU Time Thresholds. A point of inflection (PoI) is found whenever ICs ratio overtakes that of continuous execution. Values greater than zero in Table 5.8 indicate the maximum CPU time for which IC maintains a higher work-to-effort ratio. This table also includes uncertainty estimates calculated using Monte Carlo simulations.

Key Pair Generation As shown in Table 5.8, IC execution of the traditional cryptographic schemes RSA and DSA stayed ahead of continuous execution across all tested CPU thresholds at higher security strengths ('Enhanced' or equivalent), as indicated by PoIs marked as 'Max'. At 'Standard' security strengths, DSA and DH had moderate benefit, but RSA's advantage declined to minimal. While DH maintained moderate benefit for both 'Enhanced' and 'Standard' security strengths, DH showed substantial variability for Key Pair

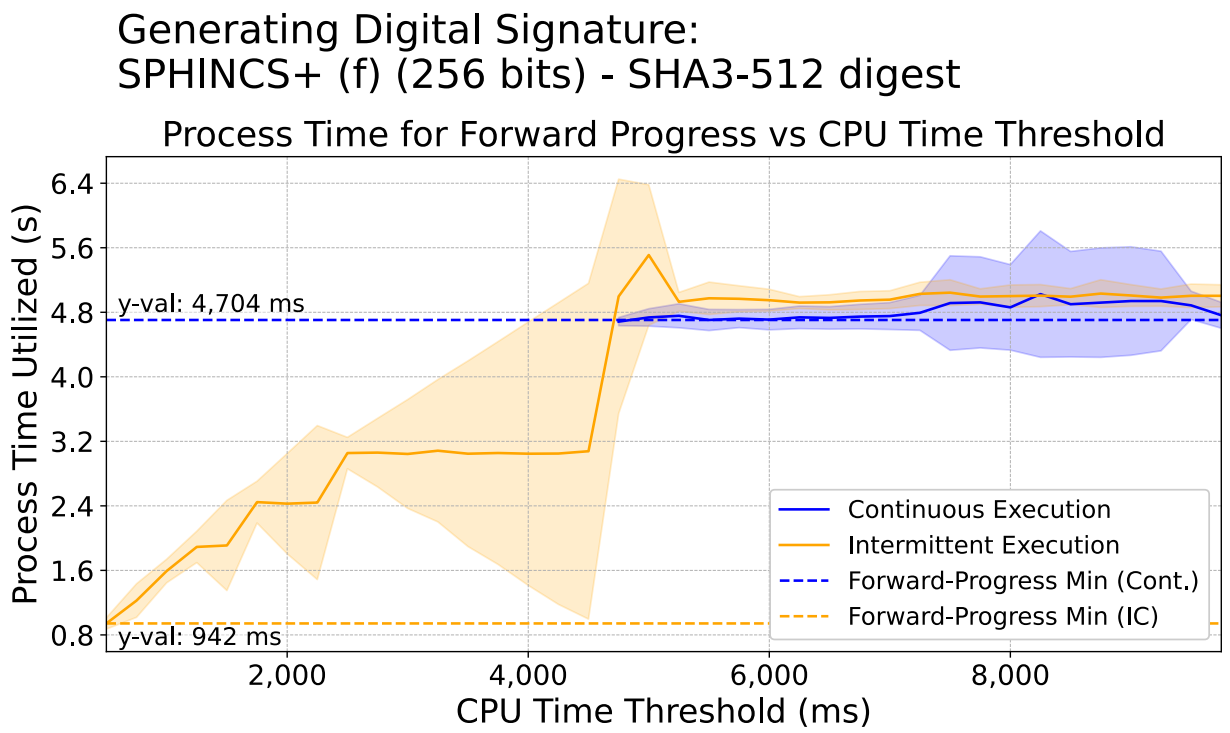


Figure 5.6: Process time utilization per CPU Time Threshold, with minimum values used to compute FPC ratio (ex-situ measurement, from Section 5.5.1) of SPHINCS+ (fast) digital signature generation using 'Advanced' key strength and message digest. Intermittent computation (IC) made forward progress using 3,765 ms less processor time than continuous execution, producing a FPC ratio of 0.2 and indicating maximum benefit from the use of IC. Solid lines denote average values with shaded regions denoting variance.

Generation as completion time was sensitive to random starting parameter seeds, producing random delays that lead to more scattered PoIs.

No post-quantum cryptographic schemes produced a PoI greater than zero for key pair generation–either completing too fast for IC to matter or IC execution incurring an overhead that negates potential gains. Looking closer at the results for SPHINCS+ (small) at 'Enhanced' security strength, Figure 5.4 shows the aggregate Work-to-Effort ratio for continuous execution maintaining higher values than IC for all plotted CPU time thresholds.

These results match expectations and reflect advancements in cryptographic scheme efficiencies over the decades.

Digital Signing and Verification Most explored digital signing schemes demonstrated moderate to significant benefits from IC, with PoIs often exceeding 2,000 ms. SPHINCS+ (fast) in particular showed especially high crossover thresholds, reflecting extended windows where IC had better Work-to-Effort ratios, as illustrated in Figure 5.5. At maximum security strength, the average PoI across all tested schemes was over 5,200 ms, with an average PoI over 3,000 ms at 'Standard' security strength.

As expected, less benefit from IC was observed at smaller key strengths due to the reduced 'difficulty' of the underlying cryptographic principles. However, the reduction in IC benefit was not synchronized, with individual schemes demonstrated different reductions in IC benefit when comparing the highest security strength to 'Standard'. For example, ECC demonstrated a 38% average drop in sign/verify benefit performance, with SPHINCS+ (fast) showing a 70% drop over the same interval. Interestingly, the lattice-based post-quantum cryptographic schemes, ML-DSA and Falcon, demonstrated minimal performance difference between 'Advanced' and 'Standard' security strengths with 'Advanced' results of both schemes within 2% of results at 'Standard'. Additionally, these schemes had significantly greater variance in their results across all security strengths compared to the other schemes investigated, as shown in Table 5.8.

These results generally match expectations as operations performed on sufficiently large data messages should take longer to execute than efficient, purely computational operations. The uneven execution performance between signature generation and signature verification for lattice-based schemes was greater than expected and it is unclear whether this imbalance is due to underlying cryptographic fundamentals or performance issues with their initial implementations.

Encryption, Decryption, and Shared Secret Generation For all tested key sizes, these operations completed prior to the first CPU Time Threshold, giving IC no window to provide an advantage.

5.6.2 Ex-Situ Evaluation

For this evaluation, we measure the *Forward-Progress Cost* (FPC) ratio–how much of continuous execution time is 'recoverable' under IC–using total CPU time of the main and helper processes. FPC ratios below 1 indicate IC could reduce instantaneous energy usage; results ≥ 1 are determined as not suitable for IC. FPC variances are also included in Table 5.8.

Key Pair Generation As with in-situ measurement, only the traditional cryptography schemes consistently achieved FPC ratios that indicate a benefit from IC at select key sizes. FPC values could not be evaluated for RSA and DSA at 'Enhanced' security strength as none of these schemes reliably produced forward-progress using continuous execution across the CPU time thresholds evaluated. To estimate a minimum benefit, we substituted the CPU time for continuous execution at the longest CPU time threshold for RSA key pair generation at 'Enhanced' strength, we find IC will use at most only 10% of the CPU time required by continuous execution to make forward progress. At 'Standard', DSA and DH maintained maximum IC benefit with FPC ratios of 0.16 and 0.18 respectively, with RSA achieving a more modest FPC ratio of 0.48. Note that values for 'Enhanced' are estimated using the substitution described above, resulting in DSA and DH demonstrating a reduced benefit from IC at this strength, which is inconsistent with the true result.

Digital Signing and Verification RSA, ECC, and SPHINCS+ consistently produced FPC ratios approximately at, or below, 0.5 for all considered security strengths, indicating substantial IC benefits. An example is illustrated in Figure 5.6 which shows an approximately 3,500ms difference between minimum process time needed for IC forward progress and minimum process time for continuous execution forward progress. By contrast, Falcon and ML-DSA generally maintained a result of 1.0 for signature generation, making them unsuitable for IC in that operation. However, their signature verification results were more promising: ML-DSA at 'Standard' strength saw process time reduced by 89%, whereas

Falcon's FPC ratio of 0.84 indicated only minimal gain.When comparing the highest key strength to 'Standard', RSA, ECC, and SPHINCS+ each saw an approximate 50% drop in IC advantages but still maintained an FPC value below 0.5. Falcon and ML-DSA remained unsuitable for signature generation at either strength, yet both benefited significantly from IC for verification at higher strengths–underscoring how these lattice-based schemes exhibit high variability depending on the specific cryptographic operation and security level.

Encryption, Decryption, and Shared Secret Generation As with in-situ results, encryption, decryption, and shared secret generation all completed prior to the first CPU Time Threshold for all tested key sizes, giving IC no window to demonstrate a benefit. To illustrate the performance gap, Figure 5.7 shows ML-KEM generating a shared secret at 'Advanced' security strength, where at no point did IC make forward progress using less processor time than continuous execution's minimum processor time for consistent forward progress.

5.6.3 Data Size Sensitivity

We explored the performance sensitivity of IC to continuous operation using a message size 1/5 of the baseline used in other evaluations (Table 5.10). Combinations considered *linearly sensitive* exhibited a performance factor of 0.20, indicating that they required 20% of the baseline execution time for a message 20% of the original size, suggesting the operation's cost scales in proportion to message size. Conversely, combinations with a factor of 0.0 finished sooner than the lowest CPU Time Threshold measured in our evaluation. Overall, most cryptographic scheme-operation pairs maintained consistent factors across both in-situ and ex-situ metrics, with notable outliers from lattice-based ML-DSA and Falcon schemes. For digital signing, these schemes retained at least 80% of the ex-situ baseline's execution cost—even though the message was 1/5th the size—across both 'Standard' and 'Advanced'

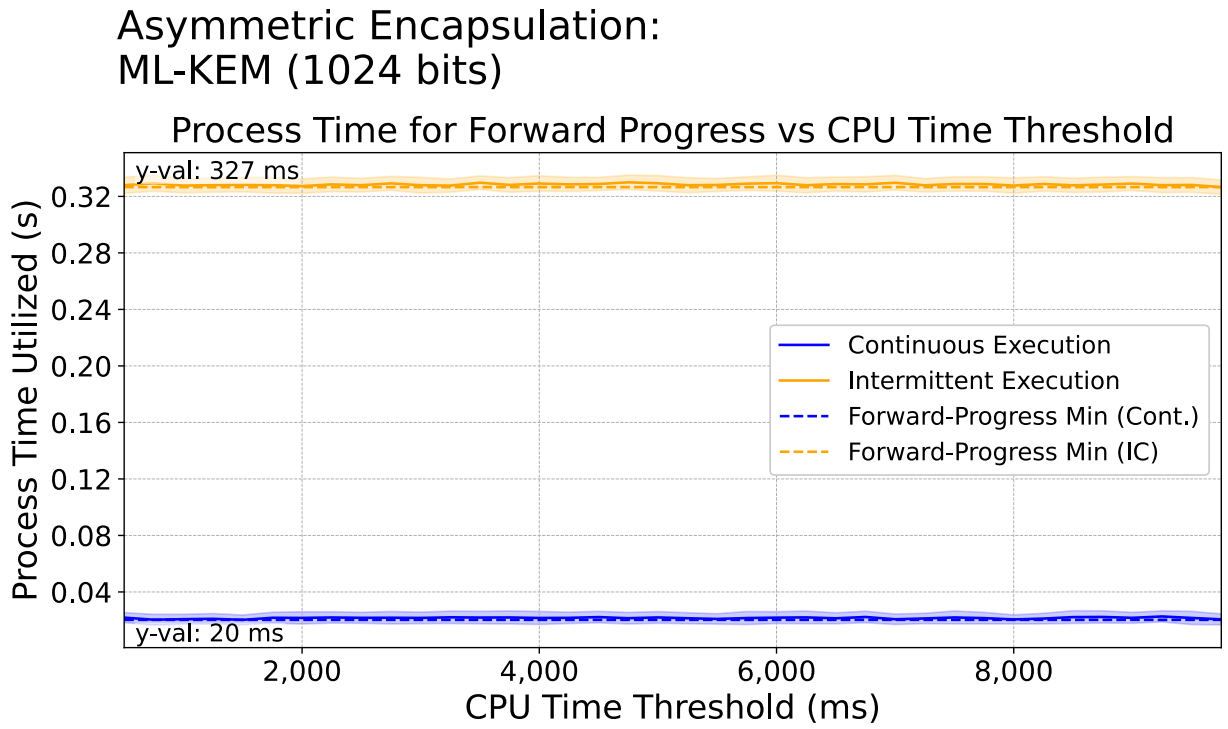


Figure 5.7: Process time utilization per CPU Time Threshold, with minimum values used to compute FPC ratio (ex-situ measurement, from Section 5.5.1) of ML-KEM shared secret generation with 'Advanced' key strength. The execution of this scheme-operation-strength combination was efficient enough that an FPC ratio could not be calculated, indicating a clear unsuitability for IC implementation. Solid lines denote average values with shaded regions denoting variance.

key sizes. In contrast, signature verification performance for 'Standard' key sizes showed inconsistent behavior: ML-DSA had an ex-situ factor of 0.08 vs 0.26 in-situ, whereas Falcon's ex-situ factor was 0.64 vs. 0.26 in-situ. These discrepancies may be due to algorithm-specific operations or inefficiencies in the still-evolving implementation codebase.

5.7 Discussion

Our experimental analyses highlight the potential for significant instantaneous energy reductions when executing certain cryptographic scheme operations using intermittent computing (IC). Specifically, we identified clear performance benefits for several cryptographic

	Key Strength	Digital Signing		Signature Verification	
		In-Situ	**Ex-Situ**	**In-Situ**	**Ex-Situ**
RSA	(S)	0.28	0.32	0.0	0.28
	(E)	0.21	0.20	0.23	0.17
DSA	(S)			0.0	0.28
	(E)			0.25	0.22
ECC	(S)	0.0	0.16	0.0	0.28
	(A)	0.31	0.27	0.21	0.21
ML-DSA	(S)	0.69	0.90	0.26	0.08
	(A)	0.30	0.80	0.23	0.37
SPHINCS+ (fast)	(S)	0.33	0.43	0.31	0.32
	(A)	0.20	0.22	0.13	0.16
SPHINCS+ (small)	(S)	0.56	0.66	0.30	0.31
	(A)	0.37	0.45	0.15	0.18
Falcon	(S)	0.30	0.86	0.26	0.64
	(A)	0.30	0.87	0.26	0.36

Table 5.10: Performance Sensitivity to Message Size. Table values represent relative performance similarity between operations executed with two message sizes, where the smaller message is 1/5th the size of the baseline used in other measurements. Color coding highlights performance stability: lower values (orange to red) indicate higher sensitivity, with smaller messages executing faster, while higher values (yellow to green) indicate reduced or minimal sensitivity.

scheme-operation combinations executed on general-purpose processors commonly adopted by modern spacecraft designs. Among the operations analyzed, only key pair generation, digital signature generation, and signature verification exhibited meaningful potential energy benefits when executed intermittently. These operations represent the minimum operation list expected to benefit from IC. Asymmetric encryption-decryption is also computationally heavy, but have strict size limitations placed on plain text data for many scheme implementations, keeping the continuous execution of this operation brief enough to not benefit from intermittent computation. Given the nuanced findings discussed in Section 5.6, our results highlight several key considerations for deploying energy-aware cryptographic strategies within future deep space communication network implementations.

5.7.1 Insights for Future Deep Space PKI

The operations identified as benefiting from IC are cornerstone components for a future deep space Public-Key Infrastructure.

Future spacecraft operating beyond Earth orbit will likely need to autonomously manage their Public Key Infrastructure (PKI), including on-demand generation and dissemination of public-private key pairs without routine intervention from Earth-based Certificate Authorities (CAs). Whereas only traditional, computationally intensive schemes demonstrated benefit from IC, continued use of these schemes within mission-critical hybrid cryptographic designs is a viable and adaptable approach to the uncertain threat of post-quantum adversaries [42, 54].

Digital signing and signature verification operations are expected to become increasingly critical within deep space networks and are already integral components of advanced space communication standards [16, 45]. Communication platforms will likely incorporate digital signing extensively to authenticate incoming messages prior to resource-intensive processing. Among the cryptographic operations analyzed, digital signing and verification demon-

strated the most consistent and significant potential benefits from intermittent computing (IC). Additionally, cryptographic algorithms using message pre-digests (e.g., RSA, ECC, SPHINCS+) exhibited clear scalability in computational effort relative to increasing key sizes and hash strengths.

Regarding symmetric and asymmetric encryption/decryption, trial results indicate these operations are unsuitable for JIT intermittent computing, as their relatively low computational complexity leads to minimal or no benefit from intermittent execution. However, alternate intermittent computing approaches, such as proactively generating one-time-pads (OTPs) demonstrated by prior studies [134], have shown potential benefits for symmetric encryption tasks. These methods merit consideration in future spacecraft cryptographic designs to further optimize energy usage under constrained conditions.

5.7.2 Suitability of Lattice-Based Cryptography

The performance variability introduced by intermittent computing (IC) warrants specific consideration, particularly for cryptographic schemes anticipated to become prevalent in the post-quantum era, such as lattice-based algorithms. Our findings show that the lattice-based algorithms evaluated, ML-DSA and Falcon, exhibit the largest performance outliers compared to traditional cryptographic schemes. For instance, while typical execution reveals little difference between the costs of signature generation and signature verification on large message sizes (as demonstrated by several of the explored algorithms), ex-situ results for ML-DSA and Falcon deviate substantially from this pattern: verification displays higher benefits from IC, whereas signature generation shows minimal or no gain. Further, in-situ analysis reveals particularly high variability for both signature generation and verification in these algorithms, a pattern not observed with non-lattice-based cryptography. Lastly, when assessing IC's sensitivity to different message sizes, signature verification in ML-DSA and Falcon is especially impacted by larger messages, whereas generation remains consistently

unsuitable for IC. Together, these results highlight the unique sensitivities of lattice-based cryptography to IC dynamics. Mission planners relying on these schemes should anticipate broader performance variance under intermittent execution, particularly for signature verification on varying message sizes. Properly accounting for this variability ensures more accurate energy planning and secure, robust integration of lattice-based methods into deep space missions. Additionally, these lattice-based operations are implemented as 'single-shot' and cannot be incrementally executed over the message body. As such, lattice-based signature algorithms may have an upper limit on maximum message sizes when implemented on memory-constrained spacecraft. Unless future algorithm implementations include streaming interfaces or fixed-size buffer operations, practical application of these schemes may remain limited on memory constrained spacecraft platforms.

While the introduction of IC results in additional performance variability, this trait was greatly pronounced for post-quantum lattice-based algorithms. In our evaluation, ML-DSA and Falcon yielded the largest performance outliers relative to traditional schemes. While signature generation and verification costs scale symmetrically for non-lattice-based schemes, ex-situ measurements of ML-DSA and Falcon diverge: signature verification demonstrates a benefit from IC, while generation sees little or no gain. Likewise, in-situ analysis shows higher variability than observed in non-lattice-based cryptography. Further, signature verification in ML-DSA and Falcon is highly sensitive to larger messages under ex-situ metrics, whereas generation remains unsuitable for IC across message sizes. These findings highlight the unique sensitivities of lattice-based cryptography under IC, underscoring the importance of anticipating broader performance variance—particularly for signature verification—when planning energy budgets and cybersecurity integration into deep space missions.

5.7.3 In-Situ vs Ex-Situ Evaluations

While not offering any definitive conclusions due to the small sample size included in our study, this work demonstrates how neither metric alone fully captures IC suitability and suggests a promising avenue for deeper investigation. Our evaluation indicates that results from in-situ and ex-situ measurement methodologies generally aligned closely for most cryptographic schemes, but notable discrepencies were observed in several lattice-based post-quantum schemes. Specifically, in-situ measurements suggested minimal performance sensitivity to changes in key size, while ex-situ evaluations indicated many scheme–operation combinations were unsuitable for intermittent computing (IC). This discrepancy underscores the necessity of dual-methodology evaluations to fully understand the intricate interactions between cryptographic algorithm design, intermittent execution dynamics, and processor architecture.

5.7.4 Battery Depth of Discharge

While not previously discussed in this work, using IC for cryptographic operations where appropriate will likely have a positive impact on spacecraft longevity. The life-expectancy of a spacecraft battery is heavily influenced by the amount it is drained before recharging during normal operations, referred to as the 'Depth of Discharge'. The impact of depth of discharge on a battery is further exacerbated at extreme temperatures, such as those in deep space [81]. Through significant reduction of draws on energy reserves between execution cycles of computationally intensive operations, the adoption of IC potentially prolongs the operational lifespan of the battery, thus the spacecraft itself. By significantly reducing instantaneous power draw during computationally intensive cryptographic tasks, IC can help moderate a spacecraft's DoD, potentially prolonging battery lifespan and enhancing overall mission endurance under harsh environmental conditions.

5.8 Findings Summary

In summary, our findings for this study include:

- Demonstration on the use of Just-in-Time intermittent computation to reduce the instantaneous energy consumption of software-based, NIST standardized cryptographic schemes and operations.
- Identifying the cryptographic schemes (RSA, DSA, and DH), along with the cryptographic operations (signature generation and signature verification), that consistently demonstrated a strong potential benefit by using intermittent computing, with up to 89% reduction of projected instantaneous power consumption.
- Proposed the cryptographic analysis framework, named IC-CAPE, to measure the execution of various schemes and operations using micro-architecture dependent and agnostic conditions. Initial results demonstrate the utility of dual-methodology evaluation frameworks to analyze process execution performance for multi-architecture domains.

Chapter 6

Realistic Emulation and Validation through Network Simulation

In this chapter, the key contributions include:

- Design and implementation of SpaceNet, a comprehensive two-phase testbed that uniquely integrates orbital dynamics with real-time network emulation, specifically targeting LEO mega-constellation networks.
- Integration of realistic terrestrial-to-terrestrial segments, significantly enhancing the ability to replicate real-world data flows and paths within satellite networks.
- Autonomous satellite node functionality, enabling dynamic routing behavior, self-monitoring, and inter-node communication within the emulated constellation.
- Development of an optimized execution mode, substantially reducing the computational complexity required for realistic emulation of large-scale constellations.
- Implementation of a versatile hardware-in-the-loop (HIL) capability, allowing incorporation of real spacecraft communication hardware to measure and model link-quality variations.
- Creation of a drop-in extensibility framework to facilitate easy and modular future enhancements without sacrificing code accessibility and maintainability.

- Development of a Bundle Protocol version 7 (BPv7) module for NS-3, enabling accurate emulation of deep-space delay-tolerant network behaviors.

- Improvements to the NS-3 Licklider Transmission Protocol (LTP) module, specifically to ensure compatibility and effective integration with the developed BPv7 module.

Contribution Area	Specific Capability Introduced	Existing SotA/ Prior Work	Enhancement or Novelty
Realistic Terrestrial Integration	Terrestrial-to-terrestrial (t2t) segment integration and routing	NPVT integrates terrestrial segments via CERNET2; minimal or no explicit t2t emulation elsewhere	Extensive, realistic emulation of terrestrial Internet server-to-gateway routing, directly integrated with satellite network
Node Autonomy and Inter-node Messaging	Satellite nodes execution autonomous logic, dynamic routing decisions, and inter-node administrative messaging (via gRPC)	Mostly static or limited autonomy without full dynamic messaging capability (StarryNet, SILLEO-SCNS lack detailed autonomy)	Full autonomous decision-making and dynamic inter-node routing communication
Optimized Execution Mode	Scalable optimized execution reducing topology size and computational overhead, enabling large-scale simulations on consumer hardware	Standard execution typically instantiates full constellations with heavy resource consumption (StarryNet, NPVT lack specific optimizations)	Significant topology reduction enabling efficient and realistic large-scale emulation
Hardware-in-the-Loop (HIL) Implementation	Seamless integration of real spacecraft communication hardware as network links within emulated topology	Limited or non-existent; StarryNet supports HIL with specific hardware focus, minimal elsewhere	General-purpose, easily configurable HIL integration with SDR-based links
Extensibility Framework (Drop-in Modules)	Dynamic, drop-in code extensibility framework to maintain codebase simplicity and facilitate easy future enhancements	Most existing testbeds lack modular, drop-in extensibility; new features typically require significant core code modifications	Novel extensibility framework enabling easy modular enhancements without modifying core architecture
Bundle Protocol v7 Implementation (NS-3)	Complete, fully functional BPv7 protocol module with support for UDP, TCP, and LTP convergence layers	Partial, incomplete implementations of older protocols (e.g., BPv6); no BPv7 implementations available	First publicly available, full functional BPv7 implementation integrated with LTP for deep-space DTNs

Note: This table highlights contributions explicitly detailed within this chapter, clarifying the distinct advancements relative to the existing state-of-the-art. State-of-the-art testbeds are listed in Section 2.4.3.

Table 6.1: Comparative Performance of Energy-Efficient Cryptographic Approaches

Novelty and Significance This chapter introduces targeted, novel enhancements in space network emulation and delay-tolerant network simulation. Key among these are a detailed terrestrial-to-terrestrial network emulation integrated directly with satellite network topologies and full node autonomy with dynamic inter-node communication using gRPC. Additionally, the scalable optimized execution mode significantly reduces computational demands,

enabling previously unattainable realistic large-scale simulations on consumer hardware. The innovative hardware-in-the-loop implementation allows easy integration of real spacecraft communication hardware, providing unmatched flexibility and realism. Furthermore, the developed drop-in extensibility framework ensures easy maintenance and feature expansion without complex core modifications. Lastly, the first publicly available BPv7 module integrated with LTP significantly advances deep-space DTN research capabilities. Collectively, these contributions significantly enhance the realism, flexibility, and applicability of space network simulation research tools, addressing crucial gaps in current state-of-the-art methodologies.

6.1 Introduction

With the growing commercial and academic interest in space, the demand for improved data transmission and routing also continues to rise — advancing space network architectures, as discussed in Section 2.2.3, as well as necessitating corresponding advances in data network testing environments. For example, consider the physical characteristics of a low-Earth orbit (LEO) mega-constellation network, where data is relayed across inter-satellite links, as described in Chapter 4: These systems feature unique, time-dependent topologies composed of hundreds to thousands of energy-harvesting nodes, each traveling at approximately 27,000 kilometers per hour while the Earth rotates beneath them. This produces an exceptionally dynamic and complex network environment, fundamentally different from terrestrial architectures and requiring novel solutions to meet both operator and end-user needs.

Delay-tolerant networks (DTNs), such as those used in deep space, are similarly distinctive — even if their scale is topologically smaller than that of LEO constellations. While near-Earth networks benefit from low-latency conditions that permit use of well-established terrestrial protocols, deep-space DTNs experience light-speed delays over vast interplanetary distances,

making many common protocols infeasible. The absence of high end-to-end availability, a near-complete lack of interactive communication, and the reliance on extremely constrained hardware built for the harsh conditions of space further necessitate new approaches when advancing state-of-the-art network technologies.

Unfortunately, existing network testbeds are terrestrial-based and fail to capture the unique aspects found only in space-based data networks. To produce practical and impactful research, tools are needed that can effectively bridge the gap between experimental setups and the unique operational conditions of space. Testbeds designed to support this role must be robust, scalable, and adaptable—capable of modeling a wide variety of protocols, constellations, and hardware constraints relevant to space applications.

In support of this need, we have developed and extended space network testing tools using two well-established, academically-supported network emulators: NS-3 and Mininet. These platforms represent two complementary ends of the networking spectrum. NS-3 excels at modeling link and channel characteristics, providing fine-grained control over lower-layer network behavior. In contrast, Mininet focuses on constructing and analyzing large-scale network topologies, with strengths in modeling upper-layer protocol interaction and software-defined networking.

This effort supports academic contributions centered on tool development for space networking research, composed of two distinct parts:

- Supporting the study of protocol behavior under dynamic link conditions, including the types of impairments and variability encountered in deep-space DTNs.
- Enabling exploration of large-scale constellation performance and routing behavior across complex, time-varying topologies relevant to LEO mega-constellations.

This effort is inherently interdisciplinary, bridging spacecraft and orbital dynamics from aerospace engineering, network and protocol design from computer science, and topological

modeling and graph theory from mathematics.

All of our contributions are made public to the research community by way of open source repositories. The Mininet-based network emulator named SpaceNet supports a public repository in with with its core initiatives, which includes providing open source tools to the research audience at large [129, 130]. The NS-3 module emulating Bundle Protocol v7 has been made available to academic researchers through personal GitHub repositories [71, 72].

6.2 Design and Implementation of a Multi-Phase LEO Satellite Network Testbed

Space@VT, with collaboration from the University of Surrey, has developed the SpaceNet testbed to support ongoing and future research, foster innovation, and advance non-terrestrial network (NTN) technologies. The resulting testbed, named SpaceNet, is currently capable of modeling any LEO mega-constellation deployed in a Walker-Delta configuration from TLEs alone. SpaceNet is capable of performing a variety of research studies, but primarily focuses on observing overall network performance under a variety of constellation and topology conditions and designs, and is well suited for the type of exploration presented in Chapter 4. As several existing works have already been published on SpaceNet [12, 37], this section will focus on the contributions made by the author and any supporting information required for reader clarity.

The SpaceNet testbed operates in two phases: Phase I collects constellation parameter definitions through either real-world or synthetic TLEs and models orbital dynamics for the target constellation over time to produce link state topology files. Optionally, static routes between all nodes over time may also be generated during Phase I. Phase II performs real-time network emulation using the software-defined network emulator Mininet. This phase reads the topology files produced by Phase I (along with the optional Phase I static routes)

and generates a real-time network instance of nodes and links from the modeled constellation. Network topology changes are made over time while various networking tools are used to measure performance statistics.

6.2.1 SpaceNet Capabilities

As discussed in Section 2.4.3, Table 2.2 provides a summary of academic space network testbeds. The included testbeds encompass a wide-range of focuses and capabilities.

SpaceNet is unique among all these listings as it is currently the only simulator in literature that supports the entire suite of network simulator capabilities listed, open source, and includes the ability to assess computational impacts. SpaceNet also possesses the capability of comparing simulation output directly to real-world data through its integration with the LEOScope measurement testbed [141].

Testbed Degrees of Realism Comparison

Table 6.2 provides a comparison between popular academic space network testbeds along our defined degrees of realism. Each feature listed indicates a capability used to more accurately model space networks either as they currently operate or are able to operate in the near future. We divide realism features into two main categories: basic and advanced.

Basic realism features represent the minimum capabilities we feel a testbed requires to model space-based networks distinctly separate from terrestrial networks while executing in such a manner as to generally benefit the research community. These features consist of:

- Lower-layer model accuracy
- Higher-layer model accuracy
- Application Support

	Celestial [111]	OMNeT ++ [143]	NPVT [149]	Plutonis [49]	Modular Emulator [60]	Hypatia [67]	Starry-Net [78]	SILLEO-SCNS [74]	LSNS [82]	SpaceNet (ours) [37]
Lower-layer Model Accuracy (Layers 1-2)	2	2	2	3	2	3	2	3	1	2
Higher-layer Model Accuracy (Layers 3-7)	3	2	3	3	3	1	3	0	2	3
Application Support	3	1	2	3	3	0	3	0	1	3
Concurrent Execution	2	1	2	3	2	1	3	1	1	3
Temporal Topology Changes	3	3	3	3	3	3	3	3	3	3
Autonomous Node Behavior	3	2	3	3	3	1	3	0	2	3
Individual Node Sensing	2	2	3	3	3	1	3	0	2	3
Inter-Node Messaging	3	3	3	0	3	1	3	0	3	3
Directional Inter-Satellite Link Modeling	1	2	1	2	1	2	1	1	1	2
Weather Impact Modeling	0	1	1	0	1	0	0	0	0	2
Terrestrial Segment Modeling	3	2	3	2	2	2	3	1	0	3
Hardware -in-the-Loop	1	0	2	2	2	0	3	0	1	2
External Validation	0	1	2	1	2	1	3	0	0	3
TOTALS:	**26/39**	**22/39**	**30/39**	**28/39**	**30/39**	**16/39**	**33/39**	**9/39**	**17/39**	**35/39**

Table 6.2: Measurement of space network degrees of realism; basic realism features are listed in top portion of the table with advanced realism features listed in bottom portion. Values indicate the level of support for each capability that contributes to overall testbed realism. 3 - Strong support; 2 - Good support; 1 - Basic/limited support; 0 - Not supported / insufficient information

- Concurrent Execution
- Temporal Topology Changes

Lower-layer model accuracy represents the fidelity in which physical and data-link layers of the network model are simulated. This includes accurate modeling of modem, antenna, and channel behavior specific to a given set of environment conditions. **Higher-layer model accuracy** represents the fidelity in which the remaining higher layers of the network model are simulated. Testbeds that utilize network stacks similar or identical to those used in real-world missions have the greatest level of accuracy where testbeds that abstract these portions of the network stack will have reduced accuracy. **Application support** represents the capabilities and realism of applications that can be employed on the testbed during experimentation. These applications can consist of either network utilities, such as ICMP requests/responses, or actual customer and mission traffic fitting the network being modeled. **Concurrent execution** represents the amount of networking that can be performed concurrently. Discrete-event simulators, such as NS-3, score low as network conditions are analyzed and updated sequentially and take longer periods of time to complete when modeling a large number of network connections. **Temporal topology changes** represent the fidelity in which the network topology can change over time. These changes can include link states (up or down), latency, and bandwidth. Fidelity is additionally impacted by the frequency in which topology updates are/can be made.

Advanced realism features represent additional testbed capabilities that model features which either enhance the realism of existing capabilities or model space network capabilities not captured in the Basic feature set. Advanced realism features consist of:

- Autonomous Node Behavior
- Individual Node Sensing
- Inter-Node Messaging

- Directional Inter-Satellite Link Modeling
- Weather Impact Modeling
- Terrestrial Segment Modeling
- Hardware-in-the-Loop (HIL)
- External Validation

Autonomous node behavior represents whether network nodes are able to perform calculations and/or actions independent of dictated behavior or a central controller. When combined with individual node sensing, this feature includes dynamic routing decisions. **Individual node sensing** represents whether a network node is able to detect, or be alerted, to any changes that occur to their interfaces or the links connected to their interfaces. This includes the ability to detect when a link state has changed to UP/DOWN. When combined with autonomous node behavior, this feature permits dynamic routing decisions. **Inter-node messaging** represents whether nodes are able to send administrative messages to neighbors or nodes further away. This feature includes a node's ability to notify controlling/authority nodes when experiencing disruptions or failure. When combined with dynamic routing decisions, this feature may include the propagation of routing tables or other similar administrative datasets. **Directional inter-satellite link modeling** represents whether node inter-satellite link interfaces determine connectivity based on the direction and orientation of their attached interfaces and the neighboring nodes. This feature more accurately represents connectivity requirements for directional optical interfaces being employed in real-world satellite mega-constellations as opposed to establishing inter-node connectivity using more primitive methods, such as distance. **Weather impact modeling** represents whether a given testbed accounts for weather conditions occurring at a given geographic place at a given time when determining link quality between satellites and terminals/gateways on the ground. Weather impact modeling permits realistic simulation of network behavior dur-

ing a variety of real-world weather conditions. **Terrestrial segment modeling** represents whether a given testbed models a terrestrial topology segment. High fidelity terrestrial segment modeling allows packet traffic to be realistically transferred between space and terrestrial network segments as dictated by real-world topology, conditions, or governing regulations. **Hardware-in-the-loop** (HIL) represents whether a testbed is able to incorporate hardware/real-world components into its simulated topology. This feature permits researchers to leverage the behavior of real-world spacecraft components, such as processors, interfaces, modems, and radios to model realistic performance impacts when exposing actual hardware to a variety of experiment conditions. **External validation** represents whether a given testbed has incorporated the ability to validate their simulated data with real-world data. Such examples include SpaceNet's use of LEOScope data [125] to compare simulated results of a given scenario against real-world data observations.

Among the explored academic space network testbeds, SpaceNet produced the highest score for degrees of realism, with StarryNet as a close second. StarryNet closely mirrors SpaceNet's capability list, to include incorporating actual spacecraft hardware into their testbed, but lacked any form of weather impact modeling. Additionally, StarryNet does not include orientation data in their description of inter-satellite links which greatly impacts inter-satellite connectivity at latitude extremes.

6.2.2 Phase I: Orbital Modeling and Link State Generation

While the bulk of our attention and effort was not focused on SpaceNet's Phase I, we did make several key contributions that includes robust orbital plane classifications to aid the use of NORAD-generated TLEs of real-world constellations along with the development of terrestrial-to-terrestrial (t2t) links.

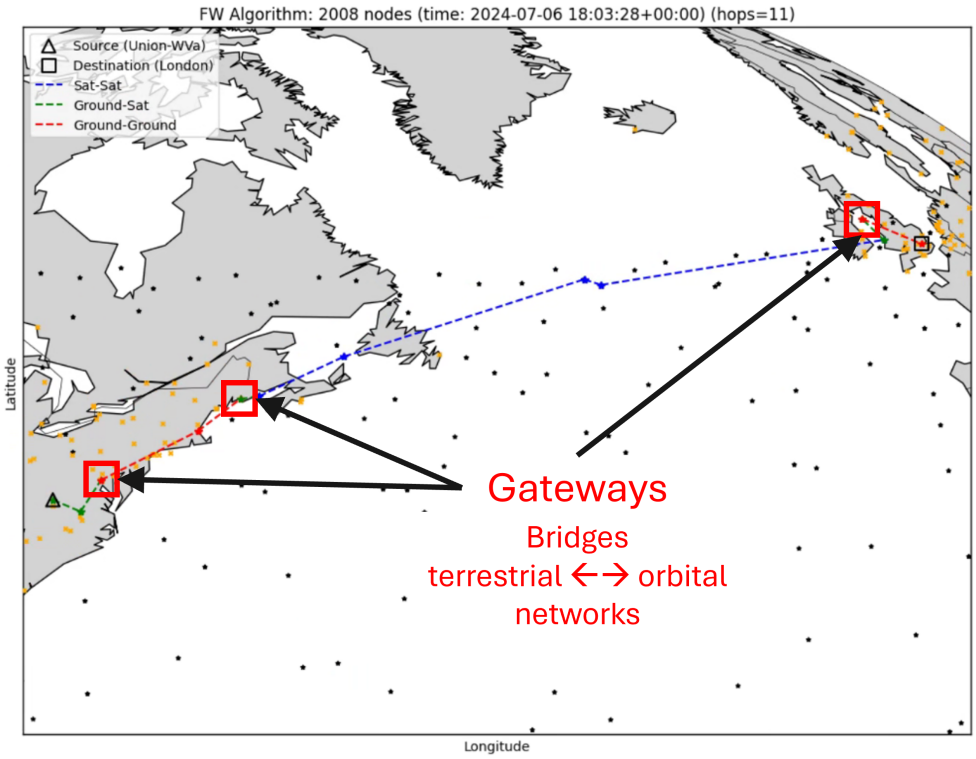


Figure 6.1: Example of packet route using terrestrial-to-terrestrial links between two endpoints.

Terrestrial-to-Terrestrial Links

Terrestrial infrastructure remains a critical component of real-world, high-performance satellite networks, particularly when considering network traffic performance from a customer perspective - rarely are customers communicating with nodes that either reside within the space network topology itself or are a single-hop away. As such, terrestrial network components must be incorporated into a testbed to effectively compare results with real-world data collection. Of the explored, existing space network testbeds, only NPVT [149] deeply integrated a terrestrial network segment into their overall design and connected their space network testbed to CERNET2.

Rather than simply connecting SpaceNet to a live terrestrial network, we sought to emulate an end-to-end connection that is easily reproducible. Our approach required an extension of our simulated topology to incorporate a terrestrial-specific segment that connects to the

space network through various 'gateways'. To realistically incorporate t2t links in SpaceNet, we needed to acquire real-world data on large Internet server locations, Starlink gateway locations, and latency values between them. To collect Internet server locations, we used two sources: Microsoft's Azure server latency website [91] and Wonderproxy's server data posting [116].

For unstated reasons, Microsoft publishes the approximate server locations for data centers outside of the United States [90], but does not post approximate U.S.-based server locations, labeling their locations using only generic regions. Because of this, the locations of U.S.-based servers had to be crowd-sourced from a variety of unofficial, online public forums, which we deemed sufficiently accurate to provide a measurable benefit to the realism of generated results [33, 106].

In another instance of commercial "strategic ambiguity", Starlink also does not post world-wide gateway locations, preventing the use of officially verified data sources to collect this information. However, SpaceX enthusiasts have already conducted crowdsourcing to compile a list of allegedly active and upcoming gateways using a public Google 'My Map' [128]. We downloaded this map's contents in a KMZ file extracted all alleged SpaceX gateway locations at the time of file retrieval.

We combined the three terrestrial data sources using the following process:

- If an Azure and Wonderproxy server pair were within 500km of each other, we added a "link" between the two locations using a conservative latency value of 10 ms. This intertwined the two terrestrial server networks.

- If a Starlink gateway was within 2,000 km of an Azure or Wonderproxy server, we considered the two nodes to be "adjacent". When a Starlink gateway was assigned an adjacency, we formed links the gateway and all of the adjacent server's distant-ends, adding a conservative 10ms latency buffer to the original link latency values.

This established a t2t dataset containing:

- 471 total terrestrial sites comprised of: 291 Internet servers / 180 Starlink gateways.
- 72,502 unique links between sites, with averages of:
  - 216 terrestrial links for each Internet servers.
  - 229 terrestrial links for each Starlink gateways.
- An average link latency of 160 ms.

Starlink gateways provide bridges between the orbiting satellite network and this skeletal terrestrial network. While the sparse terrestrial network is merely representative of major Internet traffic sources, terrestrial-to-terrestrial modeling permits highly realistic network traffic routing for meaningful comparison to real-world data sampling where real Starlink traffic will utilize both terrestrial and orbital networks. This capability is not known to exist in any other academic space network emulator.

6.2.3 Phase II: Real-Time Topology Emulation with Mininet

Real-time satellite constellation emulation performed by SpaceNet leverages the software-defined network testbed Mininet. Mininet serves as a popular, open-source software-defined network emulator that provides a convenient Python (and graphical UI) interface to build network topologies consisting of LXC Container nodes and Linux network stack links that are modified to meet specific throughput and latency specifications using Linux's Traffic Control (tc) utility. In SpaceNet Phase II, we do not use Mininet's SDR capabilities but rather generate fully parallelized network topology emulations that we manage over time according to the products of SpaceNet Phase I.

Using topology files produced by Phase I, Phase II generates a 'super-topology' in Mininet possessing all of the links that will be utilized during the execution of the target simulation,

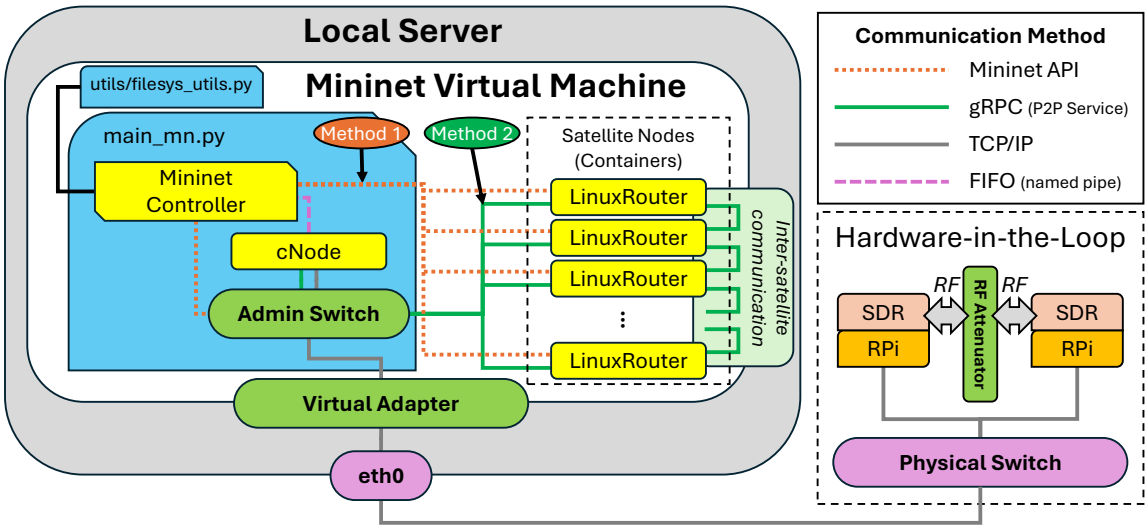


Figure 6.2: SpaceNet Phase II administrative data traffic overview. Mininet's API is the primary communication method from the Mininet control script to each of the network nodes (Method 1). When executing with external nodes or internal nodes operating autonomously, gRPC messages are sent and received through an administrative Mininet switch (Method 2). Inter-Node communication is also performed via gRPC messaging. Data sent to/from Hardware-in-the-Loop (HIL) travels external to the host machine and traverses the RF link before returning to Mininet.

regardless of link-states at specific times (links cannot be added or removed from a Mininet topology once the testbed network is started). All nodes within the topology use Mininet's LinuxRouter class, with each node operating as an LXC container. All non-available links for a given time period are taken administratively 'down' and, if using static routing (the default configuration), the route tables for each node are updated with the appropriate routes for a given time period. Once the topology is established, the simulator begins by running a designated utility, such as ping or perf. The output is collected and displayed at the conclusion of the simulation.

To illustrate the capabilities of SpaceNet, Figures 6.3 and 6.4 show two examples of experimental Starlink constellation simulations being compared to real-world data. Each plot shows three points of data for both latency and jitter: Theoretical 'best-case' values ('Derived'), simulated values ('Emulated'), and real-world values ('Actual Starlink'). In both cases, real-world latency values generally remained 60-70 ms above the simulated and 'best-

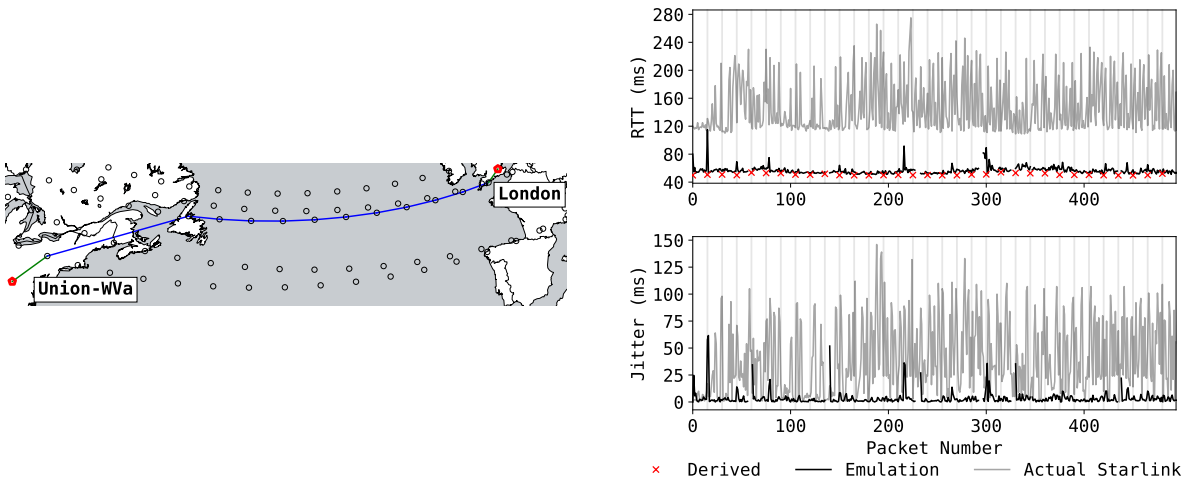


Figure 6.3: Latency and Jitter results for Starlink traffic from Union, West Virginia, U.S.A. to London, U.K..

case' estimates. This indicates that additional elements within Starlink's infrastructure are not yet properly modeled and is a critical detail for academic researchers attempting to reverse-engineer Starlink's architecture. Real-world jitter values remained higher than 'best-case' and simulated values as well, but with less of a difference than with latency;. this is most likely due to the abstractions made in a simulated environment not sufficiently modeling the amount of real-world factors that agitate Starlink traffic latency.

Among the many software engineering contributions we made to SpaceNet's Phase II, we have made the following academic contributions to the emulator's capabilities and design:

- Satellite node autonomy / inter-node communication
- Scalable "optimized" execution mode
- "Drop-in" extensibility design
- "Hardware-in-the-loop" design implementation

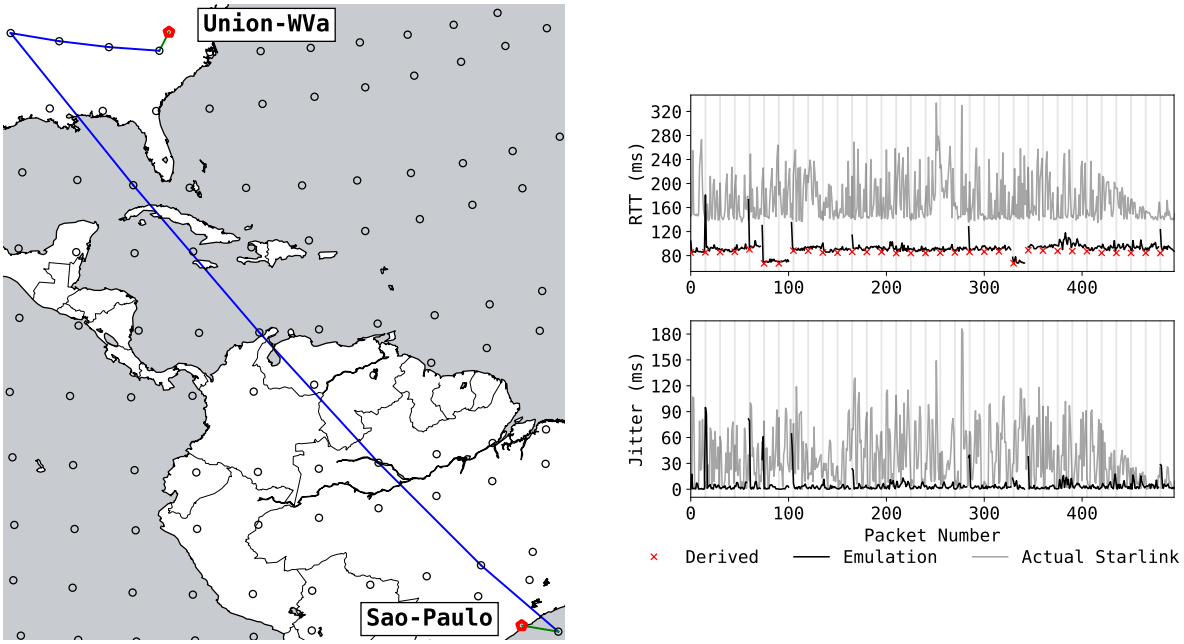


Figure 6.4: Latency and Jitter results for Starlink traffic from Union, West Virginia, U.S.A. to Sao Paulo, BR.

Satellite Node Autonomy / Inter-Node Communication

Following the complete rebasing we performed on the original emulator's code, we immediately began to expand its capabilities in pursuit of greater academic fidelity, specifically regarding the ability for SpaceNet to emulate dynamic routing. In order to do so, satellite nodes would need to 1) execute logic autonomously, and 2) pass administrative data between each other.

Satellite node autonomy was easily implemented by having each satellite execute a Python script (named 'sNode') with a given set of start-up parameters. However, meaningful behavior requires the sNode script to permit nodes to:

1. Monitor internal own network interfaces for any changes.

2. Manage internal routing tables.

3. Manually process packets with destinations not in routing tables.

4. Pass administrative traffic to neighboring nodes (or to a centralized, controller node).

We have satellites nodes utilize Python's netifaces module [6] for notification whenever the state of any network links change, permitting the execution of follow-on actions or logic, such as notifying neighboring nodes and updating internal routing tables. To maintain high performance packet routing, we continue to leverage iptables; but enable nodes to edit their internal iptables using various bash commands performed from within the sNode script.

We then configure the Python module NetFilterQueue [89] to trigger an interrupt whenever it identifies a packet with a destination not within the node's current iptable configuration. Any necessary logic is performed by the node, the node's iptables is updated to reflect the new destination network (if necessary), and the packet is either re-submitted for forwarding or dropped.

Lastly, we implemented inter-node communication using Google's remote procedure call protocol buffer, named gRPC. Nodes are able to communicate with direct neighbors using gRPC to pass administrative traffic, such as routing table updates. To aid administrative communication, we also created a separate network with all nodes belonging to a single subnet and only one 'hop' away. This administrative network, illustrated in Figure 6.2, allows nodes to receive non-customer traffic such as direct-action commands from the simulation's main controlling script. we selected gRPC for implementation due to its high performance, small size, and cross-platform support.

Scalable 'Optimized' Execution Mode

As a LEO mega-constellation network of 1,500 satellites would have between 3,000-4,500 links, the resources of any system executing a topology of this size would have limited resources remaining to devote to dynamic node behavior. To implement autonomous and dynamic node behavior with minimal delays in response time, system resources needed to

be made available when simulating networks. As a solution, we implemented SpaceNet's 'Optimized' execution mode where, prior to building the topology, all Phase I files (topology and routing) are parsed to identify the minimal list of nodes required to simulate the desired topology over the specified region of time. 'Optimized' execution then implements only these nodes and links, vastly reducing the size and scale of the modeled topology (often using less than 1% nodes that would have been instantiated during full execution when using pre-generated, static routing), thereby freeing resources that can be utilized for dynamic node behavior. In scenarios in which pre-generated static routing is not performed (thus not able to identify the nodes and links utilized throughout the simulated time prior to execution), alternate strategies can be employed to reduce the size of the network emulated in Phase II. Alternate strategies have not yet been implemented, but can include geographic boundaries, or use pre-generated routes and include k-hop neighbors (1-hop neighbors, 2-hop neighbors, etc...). Optimized execution node selection is study specific and a balance between maximizing system resource availability for dynamic node behavior while ensure sufficient topology is available to capture the full range of dynamic decisions. As a collateral benefit, 'optimized' execution allows large LEO mega-constellations to be simulated on consumer systems, permitting the study of networks on a much larger range of available systems.

Phase II Drop-In Feature Extensibility

With the addition of each new feature, SpaceNet's Phase II codebase increases in size and complexity. To maintain SpaceNet's core attribute of accessibility for the wider academic community, we sought to simplify Phase II's overall code structure while continuing to support existing and future capability. The end-result is a fully 'drop-in', extensible implementation of Phase II. This implementation is not yet public facing, but designated as a future version 3.0.

The current development branch containing extensible Phase II, which we have labeled 'Dy-

namic Library Loading' consists of the following structure: a minimal implementation of Phase II, performing only the base set of topology emulation capabilities within the overall initialize→execute→update program structure. This code base also contains a list of 'feature-hook' locations (currently 31), that provide designated windows for additional features and code to execute. Most feature-hooks are centered around a particular action in the base code, such as 'configure hosts', where feature-hooks are positioned before, during, and after each action. Any code registered to execute at a specific feature-hook listing can do so as either 'in-addition-to' (IAT) or 'in-replace-of' (IRO) the original base code. Depending on the nature of the action, feature-hooks will support calling only one or many potential feature methods (IAT feature-hooks typically permit many feature methods where IRO feature-hooks typically support only one).

For feature methods to execute at a feature-hook, they must be registered and initialized during Phase II start up. To be registered, feature libraries must be saved as Python files in the designated SpaceNet library directory (./lib) and contain the method register_hooks. This method registers the feature-hook locations the library wishes to execute and the name of the method to execute. Additionally, this method states the simulator configuration file options required to enabled this feature. All registered feature libraries have the option to perform initialization actions at the start of execution.

When a feature method is called at a feature-hook location, it is passed the current 'context' of the calling method to maximize extensibility and replicate having the feature method code be called within the current scope. This context contains all current local and global variables, along with their values, at the time of calling. When executing a feature method, the local and global context is requested, captured, and passed to the called method; the feature method then updates the context (if applicable) and returns it when complete. The returned context then updates the local and global values for any variables that were updated during feature method execution.

This design and implementation supports the accessibility of SpaceNet by keeping the core codebase simplified and streamlined to maximize readability. The dynamic library loading supports extensibility by providing a wide range of code locations for augmentation/enhancement without any modifications to the core codebase. However, there is room for improvement to resolve two main areas of concern: lack of feature sequencing and resolving feature conflicts. While feature methods can be positioned to execute in the core codebase as desired, there currently is no method to sequence the order of feature method calls. Currently, feature methods are simply called in the order they were registered for a particular feature-hook, which is dictated by alphabetical naming of the feature library file. Additionally, conflicts between feature library code currently does not have any method of identification or deconfliction. Both of these open issues should be resolved in future iterations to improve the robustness and versatility of SpaceNet's dynamic library loading.

Hardware-in-the-Loop Implementation

The integration of hardware-in-the-loop (HIL) has been a core component of Virginia Tech's contributions to the SpaceNet project. This capability, not supported in any other space network testbed at this time, permits the use of actual hardware components to serve as transmission links within the SpaceNet topology. The current hardware configuration consists of two Raspberry Pi 4 single-board computers, each connected to a software-defined radio (SDR) via USB connection, and an RF link connecting the two SDRs that passes through a digital signal attenuator, as illustrated by Figure 6.2. With this configuration, various modems, protocols, and modulation types can be explored at varying levels of signal attenuation allowing researchers to observe the impact on higher-level protocols within an IP-based network. As the initial configuration and establishment of data packet passing between the two Raspberry Pi SBCs via SDRs was performed by someone else, we contributed by developing the scheme of integration into Mininet and assisting with the implementation process.

Rather than implement each Raspberry Pi SBC as a satellite or ground-station node in the network topology, as was the original design intent, we opted to treat the two Raspberry Pi SBCs as a pair of interfaces that can be applied to two separate network nodes. This approach streamlined HIL integration into our topology and allows for easier dynamic changes in which a topology link is represented by actual hardware.

Our design adds a virtual switch to our Mininet topology with an interface bridged to a network adapter on the virtual host we use to execute SpaceNet. This virtual network adapter is bridged to a second hardware network adapter on the physical computer running SpaceNet. This network adapter connects to a hardware switch that is also connected to each of the Raspberry Pi SBC's Ethernet adapters. When designating a Mininet node to use HIL, the routing table of the designated node is updated for all packets intended for a virtual interface to be forwarded to the IP of a Raspberry Pi SBC. The Raspberry Pi SBC is configured to forward all packets with source IPs within the Mininet topology to the TUN/TAP interface instantiated for the SDR. The receiving Raspberry Pi SBC will forward all packets coming from its TUN/TAP interface to a designated IP address within the Mininet topology.

This configuration has been successfully tested with small-scale topologies and we are aiding its integration into the full Phase II codebase. This integration involves centralizing all execution script and commands to originate from the SpaceNet virtual host that provides all commands and scripts to the Raspberry Pi SBCs over secure network communication.

6.3 Design and Implementation of BPv7 and Convergence Layer Support in NS-3 for Delay-Tolerant Network Research

NS-3 is an open-source, C++ network simulator commonly used for academic studies that excels at characterizing link states to model impacts on network traffic [59]. Researchers and developers add capabilities to NS-3 through the writing of well-structured user modules. While modules for a wide range of protocols and simulated interfaces exist through community support, no working modules are publicly available that supported the Bundle Protocol. A partial implementation for Bundle Protocol version 6 (BPv6) does exist online [50], but it is incomplete and non-functional in its published state.

NASA and JPL have developed the Bundle Protocol as the de facto higher-level communication protocol used for deep space DTN networks that operates within the unique constraints of this topology. Superseding version 6 [127], the version 7 Bundle Protocol (BPv7) incorporates lessons learned from prior versions and seeks to increase simplification of the protocol, improve operating robustness, while also permitting greater flexibility to users [20].

BPv7 is much more "future-facing" than previous versions due to design updates such as compartmentalizing specific capabilities for independent growth such as Bundle Protocol Security (BPSec) [23] and Bundle-in-Bundle-encapsulation (BIBE) [21]. Mandatory actions outlined by the protocol also support future capabilities as unknown canonical/extension blocks no longer need to be stripped from a bundle when encountered by nodes that lack implementation. With only a partially functional implementation of BPv6 and no implementation of BPv7 available for the NS-3 network simulator, a serious gap prevented academic and industry researchers from leveraging NS-3's accurate channel modeling to study and advance DTNs and the Bundle Protocol. We addressed this gap by implementing a fully functional BPv7 module that emulates much of the protocol's specified higher functions.

Researchers are able to use this module to accurately simulate space-based DTNs, utilizing many features of the most current Bundle Protocol, and better understand the nature and constraints of deep-space communication networks to aid future advancement.

As a higher-level protocol, Bundle Protocol is intended to communicate with lower-level protocols via 'convergence layers' that encapsulate specific mechanisms each lower-layer protocol requires to manage connections. For our BPv7 module, we implemented convergence layer adapters supporting UDP, TCP, and the Licklider Transmission Protocol (LTP) [22]. LTP is a space-centric protocol that provides 'partially-reliable' transportation mechanisms for a portion of the data (typically the data headers) then uses 'best effort' for the remaining data portion; LTP is the most commonly employed lower-layer protocol for use with the Bundle Protocol [136]. An existing and functional LTP module is available for NS-3, but required some corrections and augmentation to fully operate with our BPv7 module.

6.3.1 Bundle Protocol version 7 (BPv7) Module Implementation

Existing implementations of BPv6 have been published by NASA and JPL, but exclusively for NASA's Interplanetary Overlay Network (ION) on Delay/Disruption Tolerant Networking (DTN). ION-DTN serves as an open-source testbed for mission planners to implement software and protocol modules in an 'operating environment' that mirrors the conditions of deep-space; the program binaries and configurations generated for and used in ION-DTN testing can be copied directly on to mission hardware.

ION-DTN does not model real-world behavior and is not a suitable environment for effective experimentation with variations of transmission components, antenna characteristics, channel configurations, and environmental impacts. Having a functional and realistic BPv7 implementation for NS-3 available to academic and industry researchers greatly benefits current and future research into DTNs.

BPv7 Primary Module Capabilities

The primary capabilities we implemented into our BPv7 module include:

- Per-node addressing
- Static node registration
- Bundle routing interface
- Implementation of Bundle Protocol static routing
- Bundle transmission, forwarding, and receipt of unicast traffic
- Bundle fragmentation

Ancillary capabilities, such as Concise Binary Object Representation (CBOR) encoding, proper structure and processing of primary, payload, and primary canonical blocks, and Cyclic Redundancy Checks (CRC) are also implemented to follow RFC 9171 specifications, but do not largely impact module functionality.

Per-node addressing and node registration are critical functionalities in which Bundle Protocol nodes are informed of the presence of other nodes within the DTN, and authorizes the receipt and processing of bundles originating from them. We implemented static node registration by which messages originating from a pre-authorized source directly informed nodes of additions to the existing topology - this approach mirrors the operation of current delay tolerant networks where the ground station serves as the trusted authority and disseminates administrative messages to all nodes in the network. Registration must occur prior to a node receiving bundles from a new node.

Our implementation also provides a routing interface for various bundle routing methods to connect. Additionally, we implement a static routing method that integrates with the routing interface so that a pre-authorized source can directly advise nodes of routes to other

nodes within the DTN. Researchers are able to implement alternative routing methods, such as Contact Graph Routing (CGR), as needed for their studies.

Lastly, our implementation supports bundle fragmentation, which is a mechanism outlined in the BPv7 specifications that allow bundles to be fragmented by a relaying node if the size of a bundle exceeds the maximum transmission size of available links and protocols (if permitted by administrative flags). This fragmentation divides the bundle into smaller bundles, each containing a fragment of the original bundle payload, along with appropriate header information, to be received by the intended destination node and reassembled once all fragments arrive. This capability is critical to the implementation of BPv7 on heterogeneous networks where data can pass through a variety of transmission means and mediums, each permitting potentially different maximum 'datagram' sizes.

BPv7 Convergence Layer Adapters

The primary mechanisms and actions performed by BPv7 are made irrespective of the underlying protocols used to transmit data bundles. The integration of BPv7 with those lower-layer protocols is performed at the 'convergence layer' level. Each supported underlying protocol must have a Convergence Layer Adapter (CLA) that interfaces between the lower-layer protocol mechanisms and the higher-layer bundle protocol actions. This interface can include manual implementation of non-native actions that must be performed to meet BPv7 requires as well as additional data retention, such as session information, dependent on the lower-layer protocol being interfaced.

The CLAs we implemented in our BPv7 module support TCP, UDP, and LTP. Additional CLAs can be implemented by other researchers for our module, depending on the needs of their study. TCP and UDP are well-studied and ubiquitous protocols, making their CLA implementations a straight-forward process. However, LTP is less-studied and has its own operations and data types needed to perform successful data passing. This CLA

implementation was more involved and is described in further detail in Section 6.3.2.

6.3.2 Licklider Transport Protocol (LTP) and Convergence Layer Extension and Integration

As previously stated, NS-3 possess an existing LTP implementation [87]; however, additional effort was needed to have a BPv7 CLA effectively interface with LTP. During our updates, we improved/enhanced some aspects of the LTP module as well as corrected some issues that were observed when the module was under heavy use.

Since LTP is a 'partially-reliable' protocol, it possess traits of both TCP and UDP protocols, while implementing traits of its own, such as LTP identification using 'Engine IDs'. To effectively interface with our BPv7 LTP CLA, we had to enhance IP address and Engine ID binding and session management, providing methods for the BPv7 LTP CLA to receive and query the status of necessary details.

Additionally, we enhanced the LTP module functionality to make it easier to deploy in NS-3 networks as well as improved the convergence layer LTP uses to interface with UDP, increasing administrative data passing for better communication management. Lastly, we resolved LTP behavior that erroneously discarded non-reliable data when an attempted data connection was not immediately available, improving the robustness of this implementation.

At the completion of our effort, our BPv7 module was able to effectively utilize LTP as the underlying protocol for data transmission, permitting network simulation implementations that mirror those of real-world deep space DTNs.

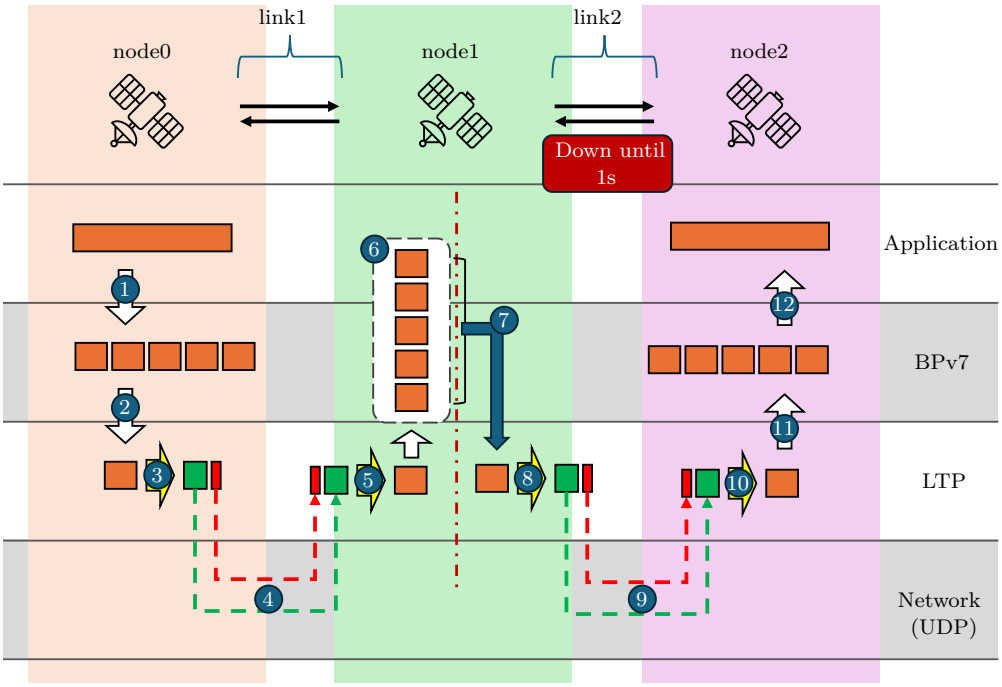


Figure 6.5: Overview of NS-3 scenario with multi-hop and link status change using Bundle Protocol version 7 (BPv7) with Licklider Transmission Protocol (LTP) as described in Section 6.3.3. Link2 is 'down' until second 1. Steps **1,12**: BPv7 fragmenting/assembling application message to/from fragmentation bundles. Steps **2,11**: BPv7 passes/receives bundle fragment to/from LTP. Steps **3,5,8,10**: LTP split/assemble bundle fragment to/from red-green segments. Steps **4,9**: LTP transmits red segment 'reliably' and green segment 'best effort'. Steps **6,7**: BPv7 queues bundle fragments for forwarding until link 2 is available.

6.3.3 Validation Scenario

To validate the implementation of our BPv7 module with our updated LTP module, we construct an NS-3 scenario that utilizes multi-hop connectivity with link-status change. This scenario verifies BPv7 bundle fragmentation, static routing, bundle storing, and forwarding with LTP's red-green segmentation transmission, and session management. Figure 6.5 provides a diagram describing the NS-3 scenario with the full scenario code listed in Appendix C. The main steps of the BPv7 and LTP actions and interactions from this scenario (after node creation and initialization), along with select log snippets (logs lightly edited for readability), are as follows:

Step 1: **Application submitting data to node0 BPv7 for transmission to node1**

BPv7 identifies data size is too large for single bundle, begins fragmentation, and passes the first bundle fragment to the LTP convergence layer adapter for transmission.

```
+0.300000000s BundleProtocol:Send_data("Received PDU of size: 1998; max bundle size is: 400,
↪  Fragmenting")
+0.300000000s BundleProtocol:Send_data("Current fragment offset: 0. Current fragment size: 400")
+0.300000000s BundleProtocol:Send_data("Send bundle: src eid dtn:node0, dst eid dtn:node2,
↪  payload size 400, serialized bundle size 493")
+0.300000000s BpLtpClaProtocol:SendBundle(0x55c99b8b3fe0 "0x55c99b899830")
```

Step 2: **Convergence Layer Encapsulation (node0, LTP)**

Bundle fragments are queued for transmission. Each bundle is divided into red(reliable) and green(unreliable) segments. This scenario sets LTP segmentation to label the administrative bundle portion as red data and the bundle payload as green data.

```
+0.300000000s BpLtpClaProtocol:SendBundle(0x55c99b8b3fe0, "Internal engine id: 0, Queuing bundle
↪  from eid: dtn:node0 to nextHopEid: dtn:node1 with nextHopEngineId: 1 to send.")
+0.300000000s BpLtpClaProtocol:SplitBundle(0x55c99b8b3fe0, "redDataMode: 1")
+0.300000000s BpLtpClaProtocol:SplitBundle(0x55c99b8b3fe0, "Original bundle CBOR size is 493
↪  bytes")
+0.300000000s BpLtpClaProtocol:SplitBundle(0x55c99b8b3fe0, "split CBOR vector has size: 510 with
↪  red part size of: 76")
```

Step 3: **LTP Segment Transmission and Reception (node0 → node1)**

Node0 LTP sends red data to node1; node1 LTP responds with acknowledgment of red data receipt; node0 LTP then sends rest of bundle as green data; node1 LTP receives green data, assembles red and green segments, and notifies node1 BPv7 of bundle receipt.

```
+0.300000000s BpLtpClaProtocol:SendIfLinkUp(0x55c99b8b3fe0, "Route and link available to
↪  LtpEngineId: 1")
```

```
+0.300000000s BpLtpClaProtocol:SendBundleFromTxQueue(0x55c99b8b3fe0, "Internal engine id: 0,
↪  Sending bundle from txQueue to 1 with redSize of 76 immediately")
+0.300000000s LtpProtocol:StartTransmission(0x55c99b8b1770, 1, 1, 76)
+0.331904000s LtpProtocol:ReceiveFrom(0x55c99b8812b0, "packet 0x55c99b8bea90, iaddr: 10.1.1.1")
+0.331904000s LtpProtocol:ReceiveFrom(0x55c99b8812b0, "internalEngineId: 1, received packet from
↪  10.1.1.1")
+0.331904000s LtpProtocol:Receive(): [DEBUG] LtpEngine: 1 Received Red data segment, checkpoint
↪  and End of Red Part: LtpHeader ( Version: 0, Type : 3, (SessionID ( Originator : 0, Number:
↪  1388083456), HeaderExtensions: 0, TrailerExtensions: 0))LtpContentHeader ( Dst. Client
↪  Service ID: 1 Offset: 0, Length: 76, CP Serial Num: 0, RP Serial Num: 0)
+0.331904000s LtpProtocol:ReportSegmentTransmission(0x55c99b8812b0, 1388083456 0, 0)
+0.347616000s LtpProtocol:Receive(): [DEBUG] LtpEngine: 1 Received a Green data segment, End of
↪  Block: LtpHeader ( Version: 0, Type : 7, (SessionID ( Originator : 0, Number: 1388083456),
↪  HeaderExtensions: 0, TrailerExtensions: 0))LtpContentHeader ( Dst. Client Service ID: 1
↪  Offset: 76, Length: 434)
+0.347616000s BpLtpClaProtocol:NotificationCallback(0x55c99b886c00, "Receiver informed a Green
↪  Segment has been received. Internal engine ID: 1, Session Id: 1388083456; Data offset: 76;
↪  Segment length: 434; End flag: 1")
+0.393280000s BpLtpClaProtocol:AssembleBundle(0x55c99b886c00, "Bundle assembled from both red and
↪  green parts; bundle payload has size: 400 bytes")
```

Step 4: **Bundle Forwarding and Queuing (node1 → node2 (unavailable))**

Node1 identifies bundle is addressed to node2, verifies it knows of path to node2, and attempts to forward bundle; node1 BPv7 confers with LTP for link availability; link from node1 to node2 is not available, so bundle is queued and link status check is scheduled.

```
+0.393280000s BundleProtocol:ProcessBundle("Received bundle for eid: dtn:node2. Current eid is:
↪  dtn:node1. Forwarding bundle")
+0.393280000s BpLtpClaProtocol:SendBundle(0x55c99b886c00, " Internal engine id: 1, Queuing bundle
↪  from eid: dtn:node0 to nextHopEid: dtn:node2 with nextHopEngineId: 2 to send.")
+0.393280000s BpLtpClaProtocol:SendIfLinkUp(0x55c99b886c00, "No interface or device available for
↪  interface index 2")
+0.393280000s BpLtpClaProtocol:AddBundleToTxQueue(0x55c99b886c00, "Unable to make connection
↪  request to destination address: 2. Scheduling link status check")
```

Step 5: **Link Status Check and Segment Transmission (node1 → node2)**

Node1 BPv7 confers with LTP for link availability; link is available and queued for

immediate transmission.

```
+1.393280000s BpLtpClaProtocol:CheckForBundleToSendFromTxQueue(0x55c99b886c00, "Checking for
↪  available link to destination engine id: 2")
+1.393280000s BpLtpClaProtocol:SendIfLinkUp(0x55c99b886c00, "Route and link available to
↪  LtpEngineId: 2")
+1.393280000s BpLtpClaProtocol:SendBundleFromTxQueue(0x55c99b886c00, "Internal engine id: 1
↪  Sending bundle from txQueue to 2 with redSize of 76 immediately")
```

Step 6: **Node2 Fragment Reception**

Node2 identifies bundle is addressed to self and that received bundle is fragment; node2 stores fragment while waiting to receive remaining.

```
+1.486560000s BundleProtocol:ProcessBundle("Recv bundle: src eid dtn:node0, dst eid dtn:node2")
+1.486560000s BundleProtocol:ProcessBundle(0x55c99b8a2780, "Bundle is part of fragment:
↪  timestamp= 800740883, total ADU length: 1998, offset= 0")
+1.486560000s BundleProtocol:ProcessBundle(0x55c99b8a2780, "First fragment for bundle:
↪  dtn:node0_800740883)
+1.486560000s BundleProtocol:ProcessBundle(0x55c99b8a2780, "Checking for complete bundle")
+1.486560000s BundleProtocol:ProcessBundle(0x55c99b8a2780, "Found fragment with block data size:
↪  400. CurrentBundleLength: 400")
+1.486560000s BundleProtocol:ProcessBundle(0x55c99b8a2780, "Have 400 out of 1998. Waiting to
↪  receive rest")
```

Step 7: **Node2 Bundle Reassembly and Passing to Application**

Node2 BPv7 receives all bundle fragments and reassembles data; node2 BPv7 calls registered callback to notify application of data receipt and necessary retrieval.

```
+1.777248000s BundleProtocol:ProcessBundle(0x55c99b8a2780, "Have complete bundle of size 1998.
↪  Reassembling")
+1.777248000s BundleProtocol:ProcessBundle(0x55c99b8a2780, "Reconstructed ADU:
↪  dtn:node0_800740883 with size 1998")
+1.777248000s BundleProtocolMultihopLinkStatusChangeLtp:RecvCallback("RecvCallback called for ",
↪  0x55c99b8a2780)
+1.777248000s BundleProtocolMultihopLinkStatusChangeLtp:Receive_char_array(): [INFO ] Receive(..)
↪  called.
```

6.4 Contributions Summary

In summary, the academic contributions we have made for space network research are:

- Advanced research tools for testing Low-Earth Orbit mega-constellation networks through the following SpaceNet efforts:
  - Integrated realistic terrestrial nodes into the satellite network infrastructure to accurately reflect paths taken by real-world network traffic.
  - Implemented satellite node autonomy, self-monitoring, and inter-node communication.
  - Developed a scalable 'optimized' execution mode to reduce network topology size by an order of magnitude that frees computational resources for autonomous node behavior modeling.
  - Aided the implementation of 'Hardware-in-the-Loop' capabilities for link-quality variation using real-world spacecraft communication hardware.
  - Developed a 'drop-in' extensibility framework to support future capability development while maintaining an accessible codebase.
- Advanced research tools for testing delay tolerant networks in deep-space by developing a Bundle Protocol version 7 module for the NS-3 network simulator.
- Enhanced delay tolerant network research by updating the NS-3 Licklider Transmission Protocol module to operate with our developed Bundle Protocol module.

Chapter 7

Future Work and Conclusions

7.1 Future Work

7.1.1 Resilient Network Topologies and Scalable Dynamic Routing

Fast, low-latency internet delivery via LEO mega-constellations is poised to usher the world into a new age of digital connectivity. Dynamically routing packets along inter-satellite links, while being a difficult problem that is still being solved, will vastly improve satellite network usage and flexibility. However, while developing effective and scalable routing algorithms, constellation operators must focus on routing resilience to mitigate inevitable adversarial disruption attempts. In this work, we discussed by proposal of an ISL architecture-derived coordinate system that simplifies dynamic routing decisions and developed the TriCoordinate logical plane modeled for Starlink's Shell 1 using six ISL interfaces per satellite. we then presented the proof-of-concept TriCoordinate Axis Priority routing algorithm that demonstrated packet delivery rate improvements over existing state-of-the-art mega-constellation routing algorithms with minimal computational and traffic overhead.

Additionally, the framework we proposed for studying dynamic routing resilience in LEO mega-constellations stands as a first attempt to formalize analysis metrics, disruption categories, and scenario requirements needed for meaningful comparison between proposed algorithms. Future proposals lacking an evaluation along any standardized guidelines will continue to be exceedingly difficult for researchers to effectively compare performance as-

pects against existing solutions.

Future effort on this work includes the implementation of the TriCoordinate logical plane and Axis Priority routing into high-fidelity space network simulators, such as SpaceNet, for improved performance analysis and comparison to more state-of-the-art proposals. Additionally, exploration of lower and higher-dimensional axes, along with non-Walker Delta constellations, to identify enduring logical plane framework factors will be the subject of future work.

In addition to studying performance impacts from varying adversarial disruptions, performing experiments using SpaceNet will permit future evaluations that incorporate real-world uncertainties, such as weather. HIL incorporation will also permit the inclusion of variance from actual device hardware and systems.

Future development of our routing resilience measurement framework will include exploring additional derived metrics, incorporation of robustness measurements, and framework validity analysis. A comprehensive proposal on evaluating the overall security and fitness of terrestrial networks is made by Cho, et al. in [28] and an exploration of which aspects are directly relevant to space networks and which aspects require adaptation and augmentation would greatly benefit the community.

7.1.2 Energy-Efficient Cryptography

The research we performed for energy-efficient cryptography spanned both proactive and reactive approaches to assess opportunities available to spacecraft from a variety of perspectives. Proactive approaches allow the design of the spacecraft itself be influenced by the selection and implementation of cryptographic schemes, and possibly resulting in significant cost savings through reduced energy needs. The integration of encryption-coupon caching into early-stage spacecraft design illustrates the potential of proactively aligning security

mechanisms with energy-aware engineering. By treating security not as an afterthought but as a power-aware design input, this work demonstrates how opportunistic precomputation can offload cryptographic effort to non-critical periods—preserving mission power budgets without compromising data integrity. The demonstrated reductions in energy demand, though individually small, translate into meaningful cost savings when accumulated across mission lifetimes, especially when factoring in the steep scaling of launch and storage costs in cislunar operations. Incorporating such precomputation methods alongside energy-efficient communications infrastructure, like LunaNet, enables a new class of small, secure, and cost-effective spacecraft.

Reactive approaches allow designers to leverage the benefits of unmodified and certified, cryptolibraries in delay-tolerant environments that permit implementations to be robust, flexible, and adaptable without the expense of dedicated, fixed-implementation cryptographic hardware modules. In this work, we proposed the use of IC as a viable solution for executing cryptographic operations, enabling reduced instantaneous power requirements via 'Just-in-Time' checkpointing while preserving the integrity of certified cryptographic libraries, to provide robust and flexible cryptographic capabilities. Our findings highlight how IC can facilitate secure, updatable cryptographic operations with reduced energy impact, enabling spacecraft to dynamically allocate computational resources based on available power.

Overall, we confirmed that intermittent computing can significantly reduce instantaneous power requirements for select software-implemented cryptographic tasks (e.g., RSA, ECC digital signing and verification), enabling robust security operations without specialized hardware. However, by including recently standardized quantum-safe algorithms into our study, we identified lattice-based schemes demonstrate inconsistent execution profiles under intermittent conditions, underscoring the necessity of careful scheme selection based on mission-specific energy and security requirements. Future work will explore optimizing IC-based cryptographic execution across various processor architectures and continually assessing long-term viability under real mission constraints.

Additionally, to address an identified gap in comparative research, we developed the IC-CAPE framework for analyzing cryptographic execution performance using IC under two separate environmental assumptions (in-situ and ex-situ). Our proposal, specifically addressing the unique factors of process execution in a deep-space environment, provide a portable and impactful utility for future research to benefit not just the target architecture of the study, but adaptable for consideration on a range of other potential architectures.

Our research in energy-efficient, software-based cryptography marks an initial exploration in applying intermittent computation to aid the security of space-based missions and the scope of our efforts was limited to a selection of common schemes and implementations. Many alternative or variant schemes remain untested, and different implementations of those already explored could produce notably different results. Moreover, as space exploration evolves, both projected processor architectures and the nature of spacecraft workloads may diverge from the initial assumptions we've made, requiring reevaluation.

Going forward, we plan to explore the benefit of using IC for cryptographic operations on embedded microcontrollers, given that specific missions will likely continue to favor low-power hardware despite reduced flexibility. This expanded work would need to examine specialized libraries (e.g., WolfSSL) and state-of-the-art JIT embedded checkpointing methods appropriate for such constrained environments.

7.1.3 Realistic Evaluation and Testing

Our efforts on developing and advancing realistic space network simulators will continue to provide benefits to the research community beyond this immediate body of work. The nature of space networks is qualitatively different from those of terrestrial origin, with significant work becoming possible only when researchers fully understand the operating constraints facing computing platforms in space, as well as the unique aspects and opportunities afforded by them.

LEO mega-constellations can still utilize terrestrial protocols due to short light-speed lag, but possess a time-dependent topology whose scope and scale necessitate novel solutions. Deep-space networks have much less 'churn' and are far sparser than LEO mega-constellations, but they are precluded from using many 'interactive' terrestrial protocols due to extreme latency and frequent non-availability of end-to-end connections while facing more stringent energy and computational limitations than near-Earth spacecraft. Aiding the development of valid and robust space network simulators that effectively simulate both ends of the space-network spectrum represent significant contributions to the field of study that will benefit the entire space network community.

Numerous opportunities for future capability and fidelity advancement exist with the open-source SpaceNet simulator. Specifically, we intend to continue contributing to dynamic routing capabilities, with the goal of implementing our TriCoordinate logical topology and Priority Axis routing algorithm to generate increased performance analysis. Additionally, we intend to bring Bundle Protocol-like capabilities to SpaceNet that allows cislunar and deep-space network researchers benefit from the system's flexibility and versatility; whether this is done by porting a BPv7 implementation from ION-DTN or simply replicating its primary behaviors remains to be seen.

7.2 Conclusion

As human and commercial interests expand toward increasingly distant and challenging space environments, the assurances of secure, energy-efficient, and resilient spacecraft and networks become paramount. This dissertation contributes significantly to these efforts by proposing innovative solutions and tailored methodologies that effectively bridge critical gaps left by the terrestrial-focused approaches of the previous decades. Through comprehensive emulation, rigorous testing, and an emphasis on real-world applicability, the research we presented lays

a solid and robust foundation for advancing the security, efficiency, and resilience of next-generation space missions.

Bibliography

[1] Mohammed Y Abdelsadek, Aizaz U Chaudhry, Tasneem Darwish, Eylem Erdogan, Gunes Karabulut-Kurt, Pablo G Madoery, Olfa Ben Yahia, and Halim Yanikomeroglu. Future space networks: Toward the next giant leap for humankind. *IEEE Transactions on Communications*, 71(2):949–1007, 2022.

[2] Inc Advanced Micro Devices. Versal adaptive SoC technical reference manual (AM011), ver 1.7, March 2025. URL https://docs.amd.com/r/en-US/am011-versal-acap-trm/Overview?tocId=608aaepLE460v_G35ldpLg. [Online; accessed 25-March-2025].

[3] National Aeronautics and Space Administration. NASA's high performance spaceflight computer, 2021. URL https://www.nasa.gov/wp-content/uploads/2024/07/hpsc-white-paper-tmg-23jul2024.pdf.

[4] European Space Agency. Optical ground station (OGS), 2025. URL https://www.esa.int/Enabling_Support/Space_Engineering_Technology/Space_Optoelectronics/Optical_Ground_Station_OGS. [Online, accessed 7-April-2025].

[5] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Daniel Smith-Tone, and Yi-Kai Liu. Status report on the third round of the nist post-quantum cryptography standardization process. Technical Report NIST Internal Report (IR) 8413, Update 1, Includes updates as of September 26, 2022, National Institute of Standards and Technology, Gaithersburg, MD, 2020.

[6] alastair and opalmer. netifaces - PyPI, May 2021. URL https://pypi.org/project/netifaces/. [Online; accessed 15-May-2024].

[7] Rafael Alexandre, Rodrigo Bruno, João Barreto, and Rodrigo Rodrigues. Evicting

for the greater good: The case for reactive checkpointing in serverless computing. In *Proceedings of the 4th Workshop on Resource Disaggregation and Serverless*, pages 44–50, 2023.

[8] Omar Alkhazragi, Abderrahmen Trichili, Chun Hong Kang, Mohammed Sait, Islam Ashry, Tien Khee Ng, Mohamed-Slim Alouini, and Boon S Ooi. Toward wide-field-of-view and large area optical detectors for high-speed optical wireless communication. *IEEE Communications Magazine*, 61(12):162–167, 2023.

[9] Thomas Aulbach, Soundes Marzougui, Jean-Pierre Seifert, and Vincent Quentin Ulitzsch. MAYo or MAY-not: Exploring implementation security of the post-quantum signature scheme MAYO against physical attacks. In *2024 Workshop on Fault Detection and Tolerance in Cryptography (FDTC)*, pages 28–33. IEEE, 2024.

[10] Brandon Bailey. Cybersecurity protections for spacecraft: A threat based approach. *The Aerospace Corporation*, 2021.

[11] Domenico Balsamo, Alex S. Weddell, Geoff V. Merrett, Bashir M. Al-Hashimi, Davide Brunelli, and Luca Benini. Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems. *IEEE Embedded Systems Letters*, 7(1):15–18, 2015. doi: 10.1109/LES.2014.2371494.

[12] Bruce Barbour, Richard Gibbons, Samantha Kenyon, James McClure, Devin Ridge, and Jonathan Black. Network testbed for small satellites (netsat) - distributed space adaptive communications and security for multi-constellation networks. In *AIAA SCITECH 2023 Forum*, 2023.

[13] Elaine Barker. Recommendation for key management: Part 1 - general. Technical Report NIST Special Publication (SP) 800-57, Part 1, Rev. 5, Includes updates as of May 4, 2020, National Institute of Standards and Technology, Gaithersburg, MD, 2020.

[14] Debopam Bhattacherjee and Ankit Singla. Network topology design at 27,000 km/hour. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, pages 341–354, 2019.

[15] William Lloyd Bircher and Lizy K John. Complete system power estimation using processor performance events. *IEEE Transactions on Computers*, 61(4):563–577, 2011.

[16] En III Birrane and K McKeever. RFC9172, bundle protocol security (BPSec), 2023. URL https://www.rfc-editor.org/rfc/rfc9172.pdf.

[17] Nathaniel Bleier, Muhammad Husnain Mubarik, Gary R Swenson, and Rakesh Kumar. Space microdatacenters. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '23, page 900–915, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400703294. doi: 10.1145/3613424.3614271. URL https://doi.org/10.1145/3613424.3614271.

[18] Jonas Boner. Latency numbers every programmer should know, 2025. URL https://gist.github.com/jboner/2841832.

[19] Nicolò Boschetti, Nathaniel G Gordon, and Gregory Falco. Space cybersecurity lessons learned from the Viasat cyberattack. In *ASCEND 2022*, page 4380. 2022.

[20] Scott Burleigh, Kevin Fall, and Edward J. Birrane. RFC 9171 - bundle protocol version 7, December 2022. URL https://datatracker.ietf.org/doc/rfc9171/. [Online; accessed 13-February-2023].

[21] Scott Burleigh, Alberto Montilla, and Joshua Deaton. draft-ietf-dtn-bibect-04 - bundle-in-bundle encapsulation, July 2024. URL https://datatracker.ietf.org/doc/draft-ietf-dtn-bibect/. [Online; accessed 29-August-2024].

[22] Scott C. Burleigh, Stephen Farrel, and Manikantan Ramadas. RFC 5326 - licklider

transmission protocol - specification, December 2022. URL https://datatracker.ietf.org/doc/rfc5326/. [Online; accessed 13-February-2023].

[23] Scott C. Burleigh, Stephen Farrel, and Manikantan Ramadas. RFC 9172 - bundle protocol security (BPSec), October 2023. URL https://datatracker.ietf.org/doc/rfc9172/. [Online; accessed 2-September-2024].

[24] Robert Cataldo. *Outer Solar System, Prospective Energy and Material Resources*, chapter 16, pages 767–790. Springer International Publishing AG, Cham, Switzerland, 2018.

[25] CelesTrak. Starlink-1071. https://celestrak.org/NORAD/elements/gp.php?GROUP=starlink&FORMAT=tle, 2023. [Online; accessed 15-May-2023].

[26] German Aerospace Center. New optical ground station inaugurated at DLR's site in Oberpfaffenhofen, October 2022. URL https://www.dlr.de/en/latest/news/2022/04/new-optical-ground-station-inaugurated-at-dlr-site-oberpfaffenhofen. [Online; accessed 7-April-2025].

[27] Wireless Networks Research Center. ETS-9 satellite communications project, 2019. URL https://www2.nict.go.jp/spacelab/en/pj_ets9.html. [Online; accessed 7-April-2025].

[28] Jin-Hee Cho, Shouhuai Xu, Patrick M Hurley, Matthew Mackay, Trevor Benjamin, and Mark Beaumont. Stram: Measuring the trustworthiness of computer-based systems. *ACM Computing Surveys (CSUR)*, 51(6):1–47, 2019.

[29] Andrea Cigliano and Francesco Zampognaro. A machine learning approach for routing in satellite mega-constellations. In *2020 international symposium on advanced electrical and communication technologies (ISAECT)*, pages 1–6. IEEE, 2020.

[30] Alexei Colin and Brandon Lucia. Chain: tasks and channels for reliable intermittent programs. *SIGPLAN Not.*, 51(10):514–530, October 2016. ISSN 0362-1340. doi: 10.1145/3022671.2983995. URL https://doi.org/10.1145/3022671.2983995.

[31] Federal Communications Commission. FCC-21-48: Space exploration holdings, LLC request for modification of the authorization for the SpaceX NGSO satellite system, 2021. URL https://www.fcc.gov/document/fcc-grants-spacexs-satellite-broadband-modification-application. Granted April 27, 2021.

[32] Xiphos Systems Corporation. Products, 2025. URL https://xiphos.com/products. [Online; accessed 21-April-2024].

[33] Datacenters.com. Data center locations: Top cities, states, countries and regions, 2024. URL https://www.datacenters.com/locations. [Online; accessed 15-August-2024].

[34] Bradley Denby and Brandon Lucia. Orbital edge computing: Machine inference in space. *IEEE Computer Architecture Letters*, 18(1):59–62, 2019.

[35] Asside Christian Djedouboum, Ado Adamou Abba Ari, Abdelhak Mourad Gueroui, Alidou Mohamadou, and Zibouda Aliouat. Big data collection in large-scale wireless sensor networks. *Sensors*, 18(12):4474, 2018.

[36] PE Dodd, MR Shaneyfelt, JR Schwank, and JA Felix. Current and future challenges in radiation effects on cmos electronics. *IEEE Transactions on Nuclear Science*, 57(4): 1747–1763, 2010.

[37] John S Downs, Bruce Barbour, Alexander Kedrowitsch, Deven Mhadgut, Suryansh Aryan, and Samantha P Kenyon. Space network (SpaceNet) testbed-development of a multi-functional testbed for simulating space communication networks. In *AIAA SCITECH 2025 Forum*, page 2716, 2025.

[38] New Space Economy. What is considered a deep space mission?, 2024. URL https://newspaceeconomy.ca/2024/01/15/what-is-considered-a-deep-space-mission/.

[39] Mahmoud MA Eid, Ahmed Nabih Zaki Rashed, and Ehab Salah El-Din. Simulation performance signature evolution of optical inter satellite links based booster EDFA and receiver preamplifiers. *Journal of Optical Communications*, (0):000010151520200190, 2020.

[40] Samsung Electronics. 288pin registered DIMM based on 4GB D-die | 78FBGA with lead-free and halogen-free, August 2016. URL https://download.semiconductor.samsung.com/resources/data-sheet/DDR4_4Gb_D_die_Registered_DIMM_Rev1_8_Aug_16-0.pdf. [Online; access April 1, 2025].

[41] Gregory Falco. The vacuum of space cyber security. In *2018 AIAA SPACE and Astronautics Forum and Exposition*, page 5275, 2018.

[42] Efat Fathalla and Mohamed Azab. Beyond classical cryptography: A systematic review of post-quantum hash-based signature schemes, security, and optimizations. *IEEE Access*, 2024.

[43] Florian Fischer. [patchset v4 next 0/3] perf stat: add user_time and system_time tool events, 2022. URL https://lore.kernel.org/lkml/20220420102354.468173-1-florian.fischer@muhq.space/.

[44] NASA Space Flight. Starlink v1.0 L28 mission completes first "shell" of satellites for worldwide coverage. https://www.nasaspaceflight.com/2021/05/starlink-complete-first-shell/, 2021. [Online; accessed 23-May-2023].

[45] The Consultative Committee for Space Data Systems. CCSDS 734.5-r-2, CCSDS bundle protocol security specification–draft recommended standard, 2023. URL https://public.ccsds.org/review/CCSDS%20734.5-R-2/734x5r2.pdf.

[46] The Consultative Committee for Space Data Systems. CCSDS 734.2-p-1.1, CCSDS bundle protocol specification–draft recommended standard, 2023. URL https://public.ccsds.org/review/CCSDS%20734.2-P-1.1/734x2p11e1.pdf.

[47] World Economic Forum. Space economy set to triple to $1.8 trillion by 2035, new research reveals, April 2024. URL https://www.weforum.org/press/2024/04/space-economy-set-to-triple-to-1-8-trillion-by-2035-new-research-reveals/. [Online; accessed 2-September-2024].

[48] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over NTRU. Specification v1.2, 2020. Available at: https://falcon-sign.info/falcon.pdf.

[49] Yue Gao, Kun Qiu, Zhe Chen, Wenjun Zhu, Qi Zhang, Handong Luo, Quanwei Lin, Ziheng Yang, and Wenhao Liu. Plotinus: A satellite internet digital twin system. *Journal of Communications and Information Networks*, 9(1):24–33, 2024.

[50] Gerard Garcia. GerardGarcia/ns3-bundle-protocol: Completion of the bundle-module created at the GSOC2013, 2015. URL https://github.com/GerardGarcia/ns3-bundle-protocol. [Online; accessed 15-April-2023].

[51] Rong Ge, Xizhou Feng, Shuaiwen Song, Hung-Ching Chang, Dong Li, and Kirk W Cameron. Powerpack: Energy profiling and analysis of high-performance systems and applications. *IEEE Transactions on Parallel and Distributed Systems*, 21(5):658–671, 2009.

[52] Antoine Gelain. Opinion: Will SpaceX spur another wave of smallsat innovation?, February 2021. URL https://aviationweek.com/space/commercial-space/opinion-will-spacex-spur-another-wave-smallsat-innovation. [Online; accessed 11-November-2022].

[53] Taha Gharaibeh, Steven Seiden, Mohamed Abouelsaoud, Elias Bou-Harb, and Ibrahim Baggili. Don't, stop, drop, pause: Forensics of container checkpoints (conpoint). In *Proceedings of the 19th International Conference on Availability, Reliability and Security*, pages 1–11, 2024.

[54] Alexandre Augusto Giron, Ricardo Custódio, and Francisco Rodríguez-Henríquez. Post-quantum hybrid key exchange: a systematic mapping study. *Journal of Cryptographic Engineering*, 13(1):71–88, 2023.

[55] Giacomo Giuliari, Tommaso Ciussani, Adrian Perrig, and Ankit Singla. ICARUS: Attacking low earth orbit satellite networks. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 317–331, 2021.

[56] Paul Grislain, Nicolas Pelissier, François Lamothe, Oana Hotescu, Jérôme Lacan, Emmanuel Lochin, and José Radzik. Rethinking LEO constellations routing with the unsplittable multi-commodity flows problem. In *2022 11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, pages 1–8. IEEE, 2022.

[57] Isispace Group. 12U/16U cubesat bus, 2024. URL https://www.isispace.nl/product/12u-16u-cubesat-bus/. [Online; accessed 29-November-2023].

[58] Yacov Y Haimes. On the definition of resilience in systems. *Risk Analysis: An International Journal*, 29(4), 2009.

[59] Thomas R Henderson, Mathieu Lacage, George F Riley, Craig Dowell, and Joseph Kopena. Network simulations with the NS-3 simulator. *SIGCOMM demonstration*, 14 (14):527, 2008.

[60] Dongxu Hou, Kanglian Zhao, Wenfeng Li, and Sidan Du. A realistic, flexible and extendible network emulation platform for space networks. *Electronics*, 11(8):1236, 2022.

[61] Honeywell International Inc. Hxnv06400 64mb non-volatile mram, October 2022. URL https://aerospace.honeywell.com/content/dam/aerobt/en/documents/learn/products/microelectronics/datasheet/HXNV06400-C.pdf. [Online; accessed April 1, 2025].

[62] Texas Instruments Incorporated. FRAM - a new generation of non-volatile memory, 2009. URL https://www.ti.com/lit/po/szzt014a/szzt014a.pdf.

[63] Canturk Isci and Margaret Martonosi. Runtime power monitoring in high-end processors: Methodology and empirical data. In *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, pages 93–104. IEEE, 2003.

[64] Saad Islam, Koksal Mus, Richa Singh, Patrick Schaumont, and Berk Sunar. A signature correction attack on the post-quantum scheme dilithium. In *Proceedings of the IEEE European Workshop on Security and Privacy*, 2022.

[65] Steve Jolly. Is software broken?, March 2009. URL https://appel.nasa.gov/2009/03/01/is-software-broken/. [Online; accessed 3-September-2024].

[66] Rebecca Jones. Advanced materials for missions to the moon - and beyond, March 2022. URL https://news.vcu.edu/article/2022/03/advanced-materials-for-missions-to-the-moon--and-beyond. [Online; accessed November 27th, 2023].

[67] Simon Kassing, Debopam Bhattacherjee, André Baptista Águas, Jens Eirik Saethre, and Ankit Singla. Exploring the "Internet from space" with Hypatia. In *Proceedings of the ACM Internet Measurement conference*, pages 214–229, 2020.

[68] Alexander Kedrowitsch. An energy efficient framework for secure cislunar communication. Poster presented at the 2023 Cislunar Security Conference, December 2023.

[69] Alexander Kedrowitsch. alexk1/orbital_routing_basic_simulator, 2024. URL https://github.com/alexk1vt/orbital_routing_basic_simulator. [Online; accessed 24-April-2025].

[70] Alexander Kedrowitsch. alexk1vt/ic_cryptography, 2025. URL https://github.com/alexk1vt/IC_cryptography. [Online; accessed 21-March-2025].

[71] Alexander Kedrowitsch. alexk1vt/ns3-bundle-protocol-v7, 2025. URL https://github.com/alexk1vt/ns-3-bundle-protocol-v7. [Online; accessed 21-March-2025].

[72] Alexander Kedrowitsch. alexk1vt/ns3-ltp, 2025. URL https://github.com/alexk1vt/ns-3-ltp. [Online; accessed 21-March-2025].

[73] Alexander Kedrowitsch, Jonathan Black, and Danfeng Daphne Yao. Resilient routing for low Earth orbit mega-constellation networks. In *2nd Workshop on the Security of Space and Satellite Systems (SpaceSec)*, 2024.

[74] Benjamin Kempton and Anton Riedl. Network simulator for large low Earth orbit satellite networks. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE, 2021.

[75] Marc A Koerschner, Kavya Navaneetha Krishnan, Alexia P Payan, and Dimitri N Mavris. Decision-making and optimization framework for the design of emerging satellite constellations. In *AIAA SCITECH 2023 Forum*, page 2549, 2023.

[76] Rajalakshmi Krishnamurthi, Adarsh Kumar, Dhanalekshmi Gopinathan, Anand Nayyar, and Basit Qureshi. An overview of IoT sensor data processing, fusion, and analysis techniques. *Sensors*, 20(21):6076, 2020.

[77] Erik Kulu. Broadband internet constellations, December 2024. URL https://www.

newspace.im/assets/fig/Newspace_constellations_internet_2024-12-31.pdf. [Online; accessed April 23, 2025].

[78] Zeqi Lai, Hewu Li, Yangtao Deng, Qian Wu, Jun Liu, Yuanjie Li, Jihao Li, Lixin Liu, Weisen Liu, and Jianping Wu. {StarryNet}: empowering researchers to evaluate futuristic integrated space and terrestrial networks. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 1309–1324, 2023.

[79] Lucas Laursen. Satellite signal jamming reaches new lows. https://spectrum.ieee.org/satellite-jamming/, May 2023.

[80] Qing Li, Shangguang Wang, Xiao Ma, Ao Zhou, and Fangchun Yang. Towards sustainable satellite edge computing. In *2021 IEEE International Conference on Edge Computing (EDGE)*, pages 1–8. IEEE, 2021.

[81] Qing Li, Shangguang Wang, Xiao Ma, Ao Zhou, Yue Wang, Gang Huang, and Xuanzhe Liu. Battery-aware energy optimization for satellite edge computing. *IEEE Transactions on Services Computing*, 17(2):437–451, 2024.

[82] Mengjie Liu, Yongqiang Gui, Jian Li, and Hancheng Lu. Large-scale small satellite network simulator: Design and evaluation. In *2020 3rd International Conference on Hot Information-Centric Networking (HotICN)*, pages 194–199. IEEE, 2020.

[83] Brandon Lucia and Benjamin Ransford. A simpler, safer programming and execution model for intermittent systems. *SIGPLAN Not.*, 50(6):575–585, June 2015. ISSN 0362-1340. doi: 10.1145/2813885.2737978. URL https://doi.org/10.1145/2813885.2737978.

[84] Kiwan Maeng and Brandon Lucia. Supporting peripherals in intermittent systems with just-in-time checkpoints. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, page 1101–1116, New

York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367127. doi: 10.1145/3314221.3314613. URL https://doi.org/10.1145/3314221.3314613.

[85] Kiwan Maeng, Alexei Colin, and Brandon Lucia. Alpaca: intermittent execution without checkpoints. *Proc. ACM Program. Lang.*, 1(OOPSLA), October 2017. doi: 10.1145/3133920. URL https://doi.org/10.1145/3133920.

[86] Cubesat Market. Titan-1: 350whr cubesat compact battery pack, 2024. URL https://www.cubesat.market/titan1-battery-matrix. [Online; accessed 29-November-2023].

[87] Ruben Martinez. GSOC2014LTP - nsnam, 2015. URL https://www.nsnam.org/wiki/GSOC2014LTP. [Online; accessed 15-June-2023].

[88] Soundes Marzougui, Vincent Ulitzsch, Mehdi Tibouchi, and Jean-Pierre Seifert. Profiling side-channel attacks on Dilithium: A small bit-fiddling leak breaks it all. *Cryptology ePrint Archive*, 2022.

[89] mattfox and oremanj. netfilterqueue - PyPI, March 2023. URL https://pypi.org/project/NetfilterQueue/. [Online; accessed 15-May-2024].

[90] Microsoft. Data residency in Azure | Microsoft Azure, 2024. URL https://azure.microsoft.com/en-us/explore/global-infrastructure/data-residency. [Online; accessed 15-August-2024].

[91] Microsoft. Azure network round-trip latency statistics | Microsoft learn, August 2024. URL https://learn.microsoft.com/en-us/azure/networking/azure-network-latency. [Online; accessed 15-August-2024].

[92] David Miller. NASA makes RISC-V the go-to ecosystem for future space missions, September 2022. URL https://www.sifive.com/press/nasa-selects-sifive-and-makes-risc-v-the-go-to-ecosystem. [Online; accessed 21-July-2023].

[93] Dustin Moody, Ray Perlner, Andrew Regenscheid, Angela Robinson, and David Cooper. Transition to post-quantum cryptography standards. Technical Report NIST Internal Report (IR) 8547, Initial Public Draft, National Institute of Standards and Technology, Gaithersburg, MD, 2024.

[94] Kendall Murphy. What's next: The future of NASA's laser communications, August 2022. URL https://www.nasa.gov/directorates/somd/space-communications-navigation-program/whats-next-the-future-of-nasas-laser-communications/. [Online, accessed 7-April-2025].

[95] Koksal Mus, Saad Islam, and Berk Sunar. Quantumhammer: a practical hybrid attack on the luov signature scheme. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1071–1084, 2020.

[96] NASA. NASA-STD-1006A: Space system protection standard, 2022. URL https://standards.nasa.gov/sites/default/files/standards/NASA/A/0/2022-07-15-NASA-STD-1006A-Approved.pdf.

[97] NASA. LunaNet interoperability specification, February 2023. URL https://www.nasa.gov/directorates/somd/space-communications-navigation-program/lunanet-interoperability-specification/. [Online; accessed 29-November-2023].

[98] NASA. LunaNet interoperability specification, version 5, January 2025. URL https://www.nasa.gov/wp-content/uploads/2025/02/lunanet-interoperability-specification-v5-baseline.pdf. [Online; accessed 3-May-2025].

[99] National Institute of Standards and Technology. Digital signature standard. Technical Report Federal Information Processing Standards Publications (FIPS PUBS) 186-5, U.S. Department of Commerce, Washington, D.C., 2023.

[100] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard. Technical Report Federal Information Processing

Standards Publications (FIPS PUBS) 203, U.S. Department of Commerce, Washington, D.C., 2024.

[101] National Institute of Standards and Technology. Module-lattice-based digital signature standard. Technical Report Federal Information Processing Standards Publications (FIPS PUBS) 204, U.S. Department of Commerce, Washington, D.C., 2024.

[102] National Institute of Standards and Technology. Stateless hash-based digital signature standard. Technical Report Federal Information Processing Standards Publications (FIPS PUBS) 205, U.S. Department of Commerce, Washington, D.C., 2024.

[103] National Institute of Standards and Technology. Announcing issuance of federal information processing standards (fips) fips 203, module-lattice-based key-encapsulation mechanism standard, fips 204, module-lattice-based digital signature standard, and fips 205, stateless hash-based digital signature standard. NIST Docket Number 240719-0201, 2024. URL https://www.federalregister.gov/documents/2024/08/14/2024-17956/announcing-issuance-of-federal-information-processing-standards-fips-fips-203-module-lattice-based.

[104] Committee on National Security Systems. National information assurance policy for space systems used to support national security missions, November 2012. URL https://www.hsdl.org/?view&did=726945. [Online; access 4-March-2025].

[105] Sung Wook Paek, Sangtae Kim, and Olivier de Weck. Optimization of reconfigurable satellite constellations using simulated annealing and genetic algorithm. *Sensors*, 19 (4):765, 2019.

[106] Carrie Parecki. Where are the azure data center locations?, November 2023. URL https://blog.purestorage.com/purely-educational/where-are-the-azure-data-center-locations/. [Online; accessed 15-August-2024].

[107] Andrew Paul. Hackers broadcast movies over a dead satellite | Popular Science, August 2022. URL https://www.popsci.com/technology/hackers-dead-satellite/. [Online; accessed 1-September-2024].

[108] James Pavur and Ivan Martinovic. Building a launchpad for satellite cyber-security research: Lessons from 60 years of spaceflight. *Journal of Cybersecurity*, 8(1):tyac008, 2022.

[109] James Pearson. Russia downed satellite internet in Ukraine - western officials. https://www.reuters.com/world/europe/russia-behind-cyberattack-against-satellite-internet-modems-ukraine-eu-2022-05-10/, May 2022.

[110] James Pethokoukis. Moore's law meet Musk's law: The underappreciated story of SpaceX and the stunning decline in launch costs, March 2024. URL https://www.aei.org/articles/moores-law-meet-musks-law-the-underappreciated-story-of-spacex-and-the-stunning-decline-in-launch-costs/. [Online; accessed 1-September-2024].

[111] Tobias Pfandzelter and David Bermbach. Celestial: Virtual software system testbeds for the LEO edge. In *Proceedings of the 23rd ACM/IFIP International Middleware Conference*, pages 69–81, 2022.

[112] Scott Poretsky, Shobha Erramilli, Jerry Perser, and Sumit Khurana. Terminology for Benchmarking Network-layer Traffic Control Mechanisms. RFC 4689, October 2006. URL https://www.rfc-editor.org/info/rfc4689.

[113] Xiaoxin Qi, Bing Zhang, and Zhiliang Qiu. A distributed survivable routing algorithm for mega-constellations with inclined orbits. *IEEE Access*, 8:219199–219213, 2020.

[114] Jon Rask, Wenonah Vercoutere, Barbara Navarro, and Al Krouse. Space faring: The radiation challenge, April 2017. URL https://www.nasa.gov/wp-content/uploads/2017/04/radiationchallenge.pdf. [Online; accessed 2-September-2024].

[115] Kaushik Ray and William Selvamurthy. Starlink's role in Ukraine. *Journal of Defence Studies*, 17(1):25–44, 2023.

[116] Paul Reinheimer. A day in the life of the Internet - WonderProxy blog, October 2020. URL https://wonderproxy.com/blog/a-day-in-the-life-of-the-internet/. [Online; accessed 15-August-2024].

[117] Bingyin Ren, Hailong Ge, Guangfei Xu, and Yongxin Zhang. Anti-jamming analysis and application of Starlink system. In *2023 International Conference on Networking, Informatics and Computing (ICNETIC)*, pages 149–151. IEEE, 2023.

[118] Grand View Research. Broadband services market size | industry report, 2030, 2025. URL https://www.grandviewresearch.com/industry-analysis/broadband-services-market. [Online; accessed April 23, 2025].

[119] Brandon Rhodes. Skyfield - documentation. https://rhodesmill.org/skyfield/, 2023.

[120] Kathleen Riesing, Hyosang Yoon, and Kerri Cahoy. A portable optical ground station for low-Earth orbit satellite communications. In *2017 IEEE International Conference on Space Optical Systems and Applications (ICSOS)*, pages 108–114. IEEE, 2017.

[121] Thomas G. Roberts. Space launch to low Earth orbit: How much does it cost?, September 2022. URL https://aerospace.csis.org/data/space-launch-to-low-Earth-orbit-how-much-does-it-cost/. [Online; accessed 2-September-2024].

[122] Werner Rosenkranz and Semjon Schaefer. Receiver design for optical inter-satellite links based on digital signal processing. In *2016 18th International Conference on Transparent Optical Networks (ICTON)*, pages 1–4. IEEE, 2016.

[123] Donald H. Rumsfeld, Duane P. Andrews, Charles A. Homer, Robert V. Davis, David E. Jeremiah, Howell M. Estes, Thomas S. Moorman, Ronald R. Fogleman, Douglas H.

Necessary, Jay M. Garner, Glenn K. Otis, William R. Graham, and Malcolm Wallop. Report to the commission to assess United States national security space management and organization. Report, Commission to Assess United States National Security Space Management and Organization, Washington D.C., January 2001. URL https://aerospace.csis.org/wp-content/uploads/2018/09/RumsfeldCommission.pdf.

[124] Archanaa S. Krishnan and Patrick Schaumont. Exploiting security vulnerabilities in intermittent computing. In *Security, Privacy, and Applied Cryptography Engineering: 8th International Conference, SPACE 2018, Kanpur, India, December 15-19, 2018, Proceedings 8*, pages 104–124. Springer, 2018.

[125] Nishanth Sastry, Mohamed Kassem, Vinod Khandkar, Abdullahi Abubakar, Roger Zhang, Saeed Fadaei, James Cocker, Aravindh Raman, Debopam Bhattacherjee, Aryan Tanega, Shubham Tiwari, and Saksham Bushan. LEOScope: A measurement testbed for low-earth orbit (LEO) satellite networks. URL https://leoscope.surrey.ac.uk/.

[126] Parth Satam. Russia's TOBOL EW system 'cuts off' Starlink from it's ground terminals; how did Moscow delink the starlink. https://www.eurasiantimes.com/russias-tobol-ew-system-cuts-off-starlink-from-its-ground-terminals/, April 2023.

[127] Keith Scott and Scott C. Burleigh. RFC 5050 - bundle protocol specification, January 2020. URL https://datatracker.ietf.org/doc/rfc5050/. [Online; accessed 13-February-2023].

[128] Crowd Sourced. Unofficial Starlink global gateways and pops - Google My Maps, April 2025. URL https://www.google.com/maps/d/u/0/viewer?mid=1805q6rlePY4WZd8QMOaNe2BqAgFkYBY. [Online; accessed 15-August-2024].

[129] VT SpaceNet. VT SpaceNet simulation phase, 2025. URL https://github.com/VTSpaceNetLab/VTSpaceNetPhase1. [Online; accessed 28-February-2025].

[130] VT SpaceNet. VT SpaceNet emulation phase, 2025. URL https://github.com/VTSpaceNetLab/VTSpaceNetPhase2. [Online; accessed 5-March-2025].

[131] Gregory Stock, Juan A Fraire, and Holger Hermanns. Distributed on-demand routing for LEO mega-constellations: A Starlink case study. In *2022 11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, pages 1–8. IEEE, 2022.

[132] Radostin Stoyanov, Adrian Reber, Daiki Ueno, Michał Clapiński, Andrei Vagin, and Rodrigo Bruno. Towards efficient end-to-end encryption for container checkpointing systems. In *Proceedings of the 15th ACM SIGOPS Asia-Pacific Workshop on Systems*, pages 60–66, 2024.

[133] Daniel Suárez, Francisco Almeida, and Vicente Blanco. Comprehensive analysis of energy efficiency and performance of ARM and RISC-V SoCs. *The Journal of Supercomputing*, pages 1–19, 2024.

[134] Charles Suslowicz, Archanaa S Krishnan, and Patrick Schaumont. Optimizing cryptography in energy harvesting applications. In *Proceedings of the 2017 Workshop on Attacks and Solutions in Hardware Security*, pages 17–26, 2017.

[135] Clayton Swope, Kari A Bingen, Makena Young, Madeleine Chang, Stephanie Songer, and Jeremy Tammelleo. Space threat assessment 2024, 2024.

[136] Yuehao Tang. An efficient LTP transport protocol implementation and testing. In *Proceedings of the 2023 8th International Conference on Systems, Control and Communications*, pages 52–56, 2023.

[137] IR Team. Network jitter - common causes and best solutions. https://www.ir.com/guides/what-is-network-jitter, 2023.

[138] Frontgrade Technologies. Frontgrade SpaceVPX reconfigurable processing module advanced datasheet, ver 1.0.5, October 2024. URL https://www.frontgrade.com/sites/default/files/documents/rpm-ps_advanced_datasheet_v1.0.5.pdf. [Online; accessed 25-March-2025].

[139] Vorago Technologies. Edge computing microprocessor, 2025. URL https://www.voragotech.com/va7230-edge-computing-microprocessor. [Online; accessed 21-April-2024].

[140] O.C.E Technology. Hisaor rad-tolerant AI SOC, October 2021. URL https://ocetechnology.com/hisaor-rad-tolerant-ai-soc/. [Online; accessed 21-April-2024].

[141] Shubham Tiwari, Saksham Bhushan, Aryan Taneja, Mohamed Kassem, Cheng Luo, Cong Zhou, Zhiyuan He, Aravindh Raman, Nishanth Sastry, Lili Qiu, et al. T3p: Demystifying low-Earth orbit satellite broadband. *arXiv preprint arXiv:2310.11835*, 2023.

[142] Vincent Quentin Ulitzsch, Soundes Marzougui, Alexis Bagia, Mehdi Tibouchi, and Jean-Pierre Seifert. Loop aborts strike back: Defeating fault countermeasures in lattice signatures with ILP. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(4):367–392, 2023.

[143] Aiden Valentine and George Parisis. Developing and experimenting with LEO satellite constellations in OMNeT++. *arXiv preprint arXiv:2109.12046*, 2021.

[144] Robert Vamosi. Hacking satellites, satellites today lack basic security..., August 2023. URL https://medium.com/@robvamosi/hacking-satellites-43c3135f18fd/. [Online; accessed 1-December-2023].

[145] Mike Wall. Space mining startup astroforge aims to launch historic asteroid-landing mission in 2025, August 2024. URL https://www.space.com/asteroid-mining-astroforge-docking-mission-2025. [Online; accessed 2-September-2024].

[146] Harrison Williams, Xun Jian, and Matthew Hicks. Forget failure: Exploiting SRAM data remanence for low-overhead intermittent computation. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 69–84, 2020.

[147] Harrison Williams, Saim Ahmad, and Matthew Hicks. A difference world: High-performance, NVM-invariant, software-only intermittent computation. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 1223–1238, Santa Clara, CA, July 2024. USENIX Association. ISBN 978-1-939133-41-0. URL https://www.usenix.org/conference/atc24/presentation/williams.

[148] Joel Van Der Woude and Matthew Hicks. Intermittent computation without hardware support or programmer intervention. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 17–32, Savannah, GA, November 2016. USENIX Association. ISBN 978-1-931971-33-1. URL https://www.usenix.org/conference/osdi16/technical-sessions/presentation/vanderwoude.

[149] Zengyin Yang, Hewu Li, Qian Wu, and Jianping Wu. NPVT: Network protocol validation testbed for integrated space-terrestrial network. *IEEE Access*, 7:46831–46845, 2019.

[150] Kasım Sinan Yıldırım, Amjad Yousef Majid, Dimitris Patoukas, Koen Schaper, Przemyslaw Pawelczak, and Josiah Hester. InK: Reactive kernel for tiny batteryless sensors. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, SenSys '18, page 41–53, New York, NY, USA, 2018. Association for

Computing Machinery. ISBN 9781450359528. doi: 10.1145/3274783.3274837. URL https://doi.org/10.1145/3274783.3274837.

[151] Eren Yıldız, Lijun Chen, and Kasim Sinan Yıldırım. Immortal threads: Multithreaded event-driven intermittent computing on Ultra-Low-Power microcontrollers. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 339–355, Carlsbad, CA, July 2022. USENIX Association. ISBN 978-1-939133-28-1. URL https://www.usenix.org/conference/osdi22/presentation/yildiz.

[152] Lingjing Yu, Jingli Hao, Jun Ma, Yong Sun, Yijun Zhao, and Bo Luo. A comprehensive analysis of security vulnerabilities and attacks in satellite modems. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 3287–3301, 2024.

[153] Florian Zaruba and Luca Benini. The cost of application-class processing: Energy and performance analysis of a linux-ready 1.7-ghz 64-bit RISC-V core in 22-nm FDSOI technology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27 (11):2629–2640, 2019.

[154] Arkady Zaslavsky, Charith Perera, and Dimitrios Georgakopoulos. Sensing as a service and big data. *arXiv preprint arXiv:1301.0159*, 2013.

[155] Shengyu Zhang, Xiaoqian Li, and Kwan Lawrence Yeung. Segment routing for traffic engineering and effective recovery in low-earth orbit satellite constellations. *Digital Communications and Networks*, 2022.

[156] Peng Zhao, Jiang Liu, Ran Zhang, and Tao Huang. Self-healing motif-based distributed routing algorithm for mega-constellation. In *2022 5th International Conference on Hot Information-Centric Networking (HotICN)*, pages 90–98. IEEE, 2022.

Appendices

Appendix A

Resilient Routing for Low Earth Orbit Mega-Constellations Supplementary Content

A.1 Constellation Characteristic Definitions

Common satellite constellation definitions are:

$$P : \text{Number of orbital planes} \tag{A.1}$$

$$O_n : \text{An orbital plane's number within the constellation, with values ranging from } O_0, O_1, ..., O_{P-1} \tag{A.2}$$

$$S : \text{Number of satellites in each orbital plane} \tag{A.3}$$

$$i : \text{The index value of a satellite within its orbital plane, with values ranging from } 0, 1, ..., S-1 \tag{A.4}$$

$$s_n : \text{A satellite's number within the constellation, with values ranging from } s_0, s_1, ..., s_{T-1} \tag{A.5}$$

$$\nu : \text{Number of ISL interfaces on each satellite} \tag{A.6}$$

A.2 Definitions and Assumptions

A.2.1 Supplemental Definitions

$$\phi_a \rightarrow \phi_b : \begin{array}{l}\text{Satellite latitudinal state change, such} \\ \text{as non-latitude extreme to latitude ex-} \\ \text{treme and vice versa}\end{array} \tag{A.7}$$

A.2.2 Assumed Constellation Properties

We assume a 'standard' employment for Walker Delta constellations in LEO with the intent for maximal ground coverage will possess the following properties:

P1: $S < P \Rightarrow$ The number of orbital planes will outnumber the number of satellites within each orbital plane.

P2: $P \neq S * k$ where k is an arbitrary integer $\Rightarrow$ No algebraic relationship exists between the number of orbital planes in a constellation and the number of satellites in each orbital plane.

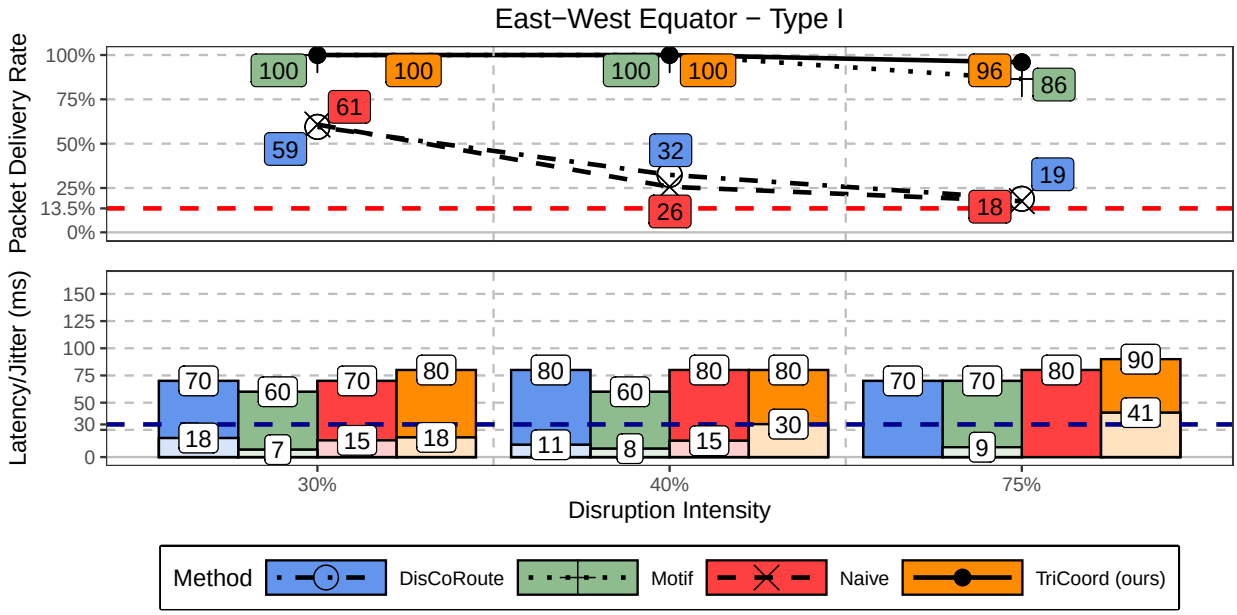


Figure A.1: Comparative performance of four dynamic routing algorithms in the East-West Equator scenario under Type I adversarial disruptions. Packet delivery rate is shown in upper point graph, with packet latency and latency jitter represented in lower bar graph and bar fill pattern, respectively. Red dashed line (top) marks packet delivery threshold below which routing is ineffectual during adversarial disruption. Blue dashed line (bottom) marks jitter threshold above which degrades user experience. Results are shown for 30%, 40%, and 75% disruption intensities.

A.3 Additional Evaluation Results

A.3.1 East–West Equatorial Scenario

A.3.2 East–West High Latitude Scenario

A.3.3 North–South Americas Scenario

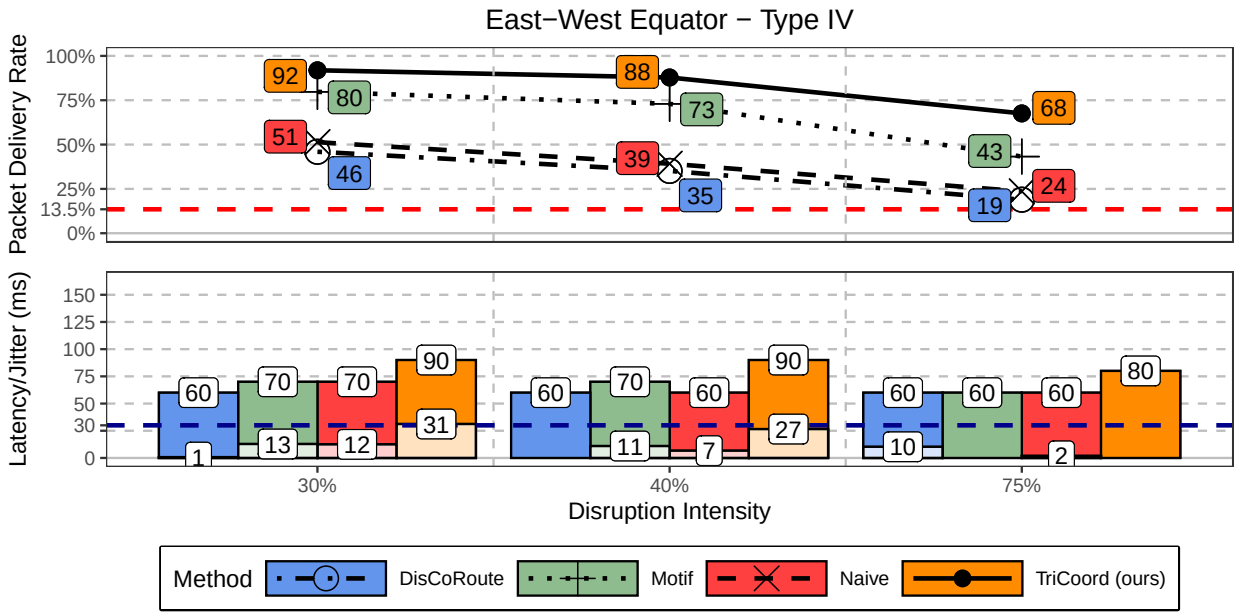


Figure A.2: Comparative performance of four dynamic routing algorithms in the East-West Equator scenario under Type IV adversarial disruptions. Packet delivery rate is shown in upper point graph, with packet latency and latency jitter represented in lower bar graph and bar fill pattern, respectively. Red dashed line (top) marks packet delivery threshold below which routing is ineffectual during adversarial disruption. Blue dashed line (bottom) marks jitter threshold above which degrades user experience. Results are shown for 30%, 40%, and 75% disruption intensities.

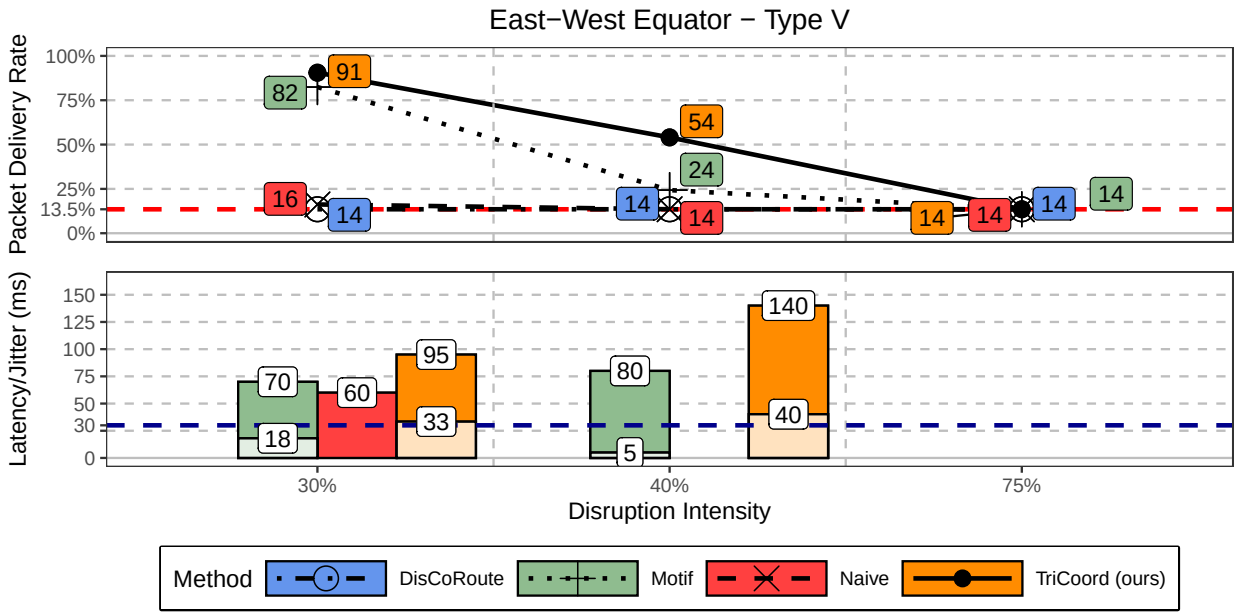


Figure A.3: Comparative performance of four dynamic routing algorithms in the East-West Equator scenario under Type V adversarial disruptions. Packet delivery rate is shown in upper point graph, with packet latency and latency jitter represented in lower bar graph and bar fill pattern, respectively. Red dashed line (top) marks packet delivery threshold below which routing is ineffectual during adversarial disruption. Blue dashed line (bottom) marks jitter threshold above which degrades user experience. Results are shown for 30%, 40%, and 75% disruption intensities.

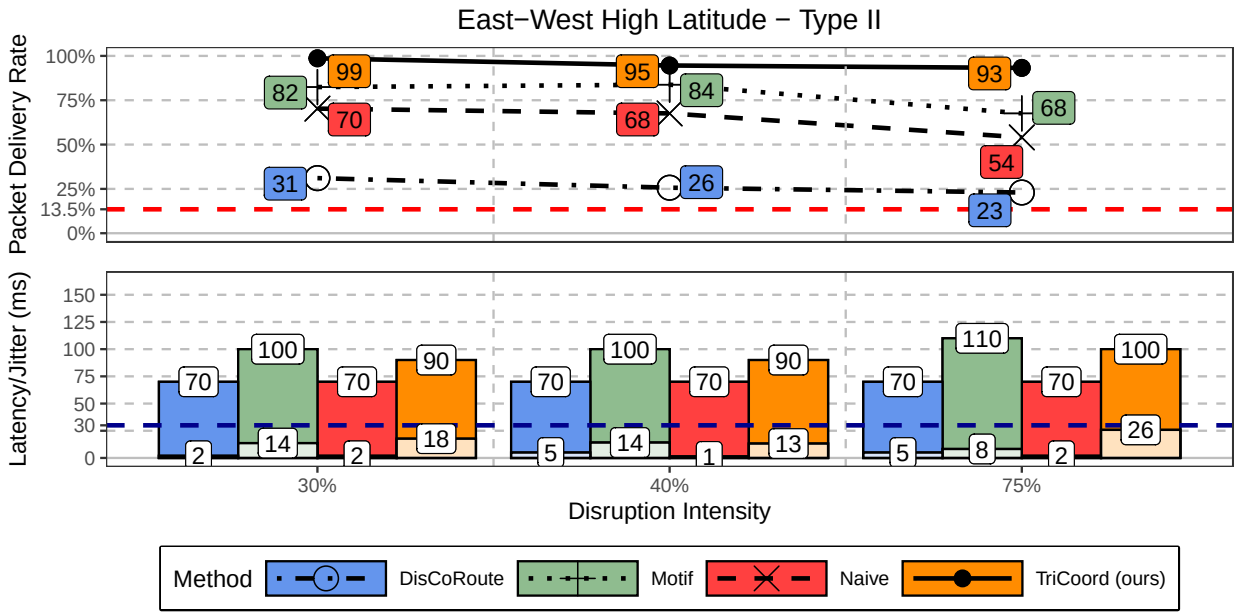


Figure A.4: Comparative performance of four dynamic routing algorithms in the East-West High Latitude scenario under Type II adversarial disruptions. Packet delivery rate is shown in upper point graph, with packet latency and latency jitter represented in lower bar graph and bar fill pattern, respectively. Red dashed line (top) marks packet delivery threshold below which routing is ineffectual during adversarial disruption. Blue dashed line (bottom) marks jitter threshold above which degrades user experience. Results are shown for 30%, 40%, and 75% disruption intensities.

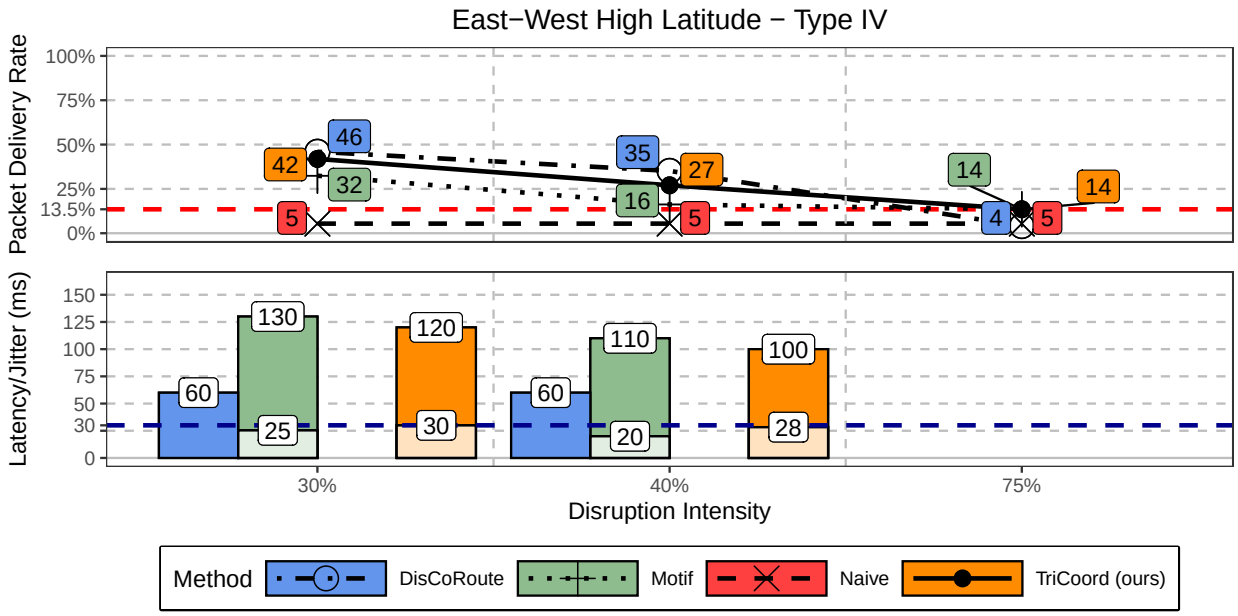


Figure A.5: Comparative performance of four dynamic routing algorithms in the East-West High Latitude scenario under Type IV adversarial disruptions. Packet delivery rate is shown in upper point graph, with packet latency and latency jitter represented in lower bar graph and bar fill pattern, respectively. Red dashed line (top) marks packet delivery threshold below which routing is ineffectual during adversarial disruption. Blue dashed line (bottom) marks jitter threshold above which degrades user experience. Results are shown for 30%, 40%, and 75% disruption intensities.

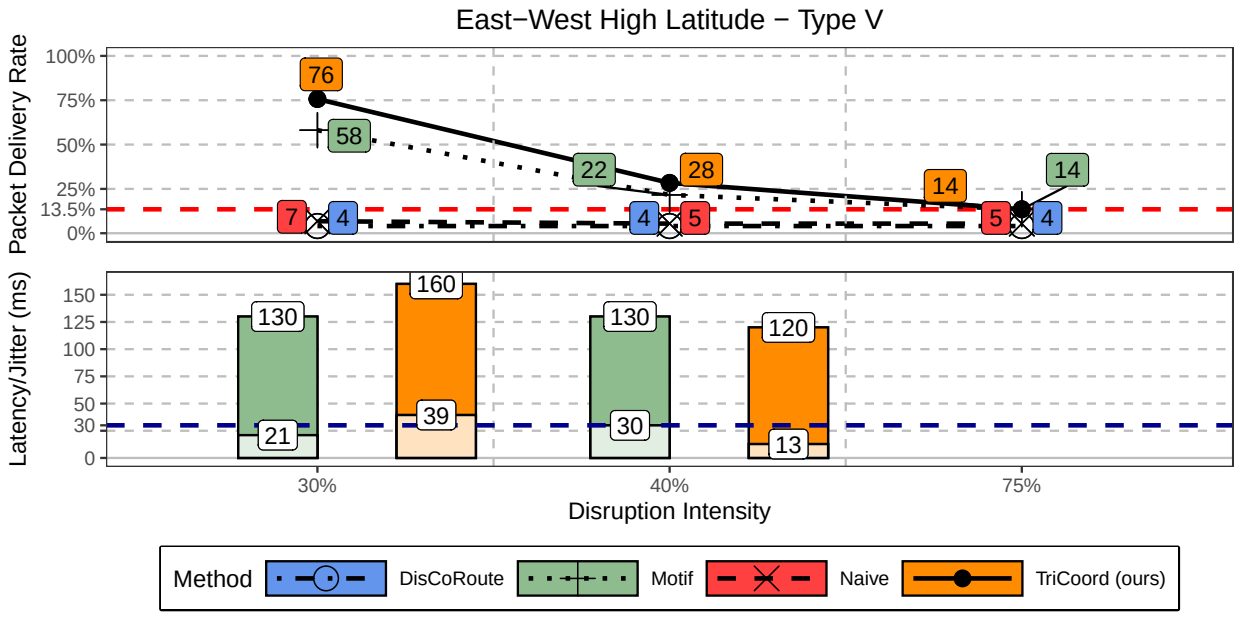


Figure A.6: Comparative performance of four dynamic routing algorithms in the East-West High Latitude scenario under Type V adversarial disruptions. Packet delivery rate is shown in upper point graph, with packet latency and latency jitter represented in lower bar graph and bar fill pattern, respectively. Red dashed line (top) marks packet delivery threshold below which routing is ineffectual during adversarial disruption. Blue dashed line (bottom) marks jitter threshold above which degrades user experience. Results are shown for 30%, 40%, and 75% disruption intensities.

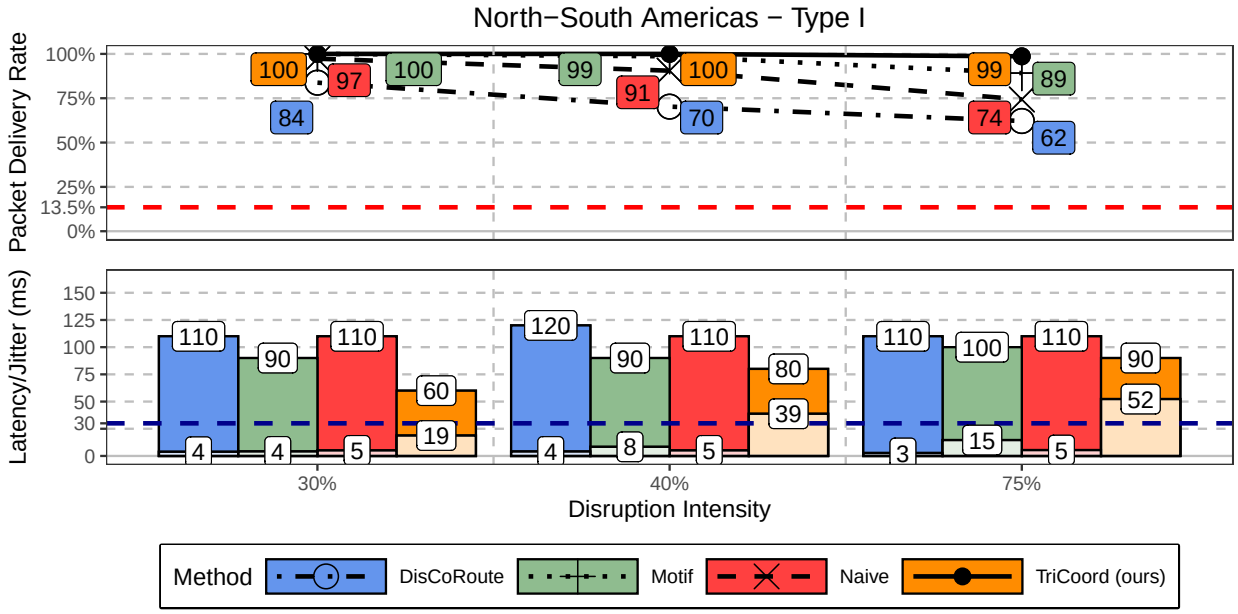


Figure A.7: Comparative performance of four dynamic routing algorithms in the North-South Americas scenario under Type I adversarial disruptions. Packet delivery rate is shown in upper point graph, with packet latency and latency jitter represented in lower bar graph and bar fill pattern, respectively. Red dashed line (top) marks packet delivery threshold below which routing is ineffectual during adversarial disruption. Blue dashed line (bottom) marks jitter threshold above which degrades user experience. Results are shown for 30%, 40%, and 75% disruption intensities.

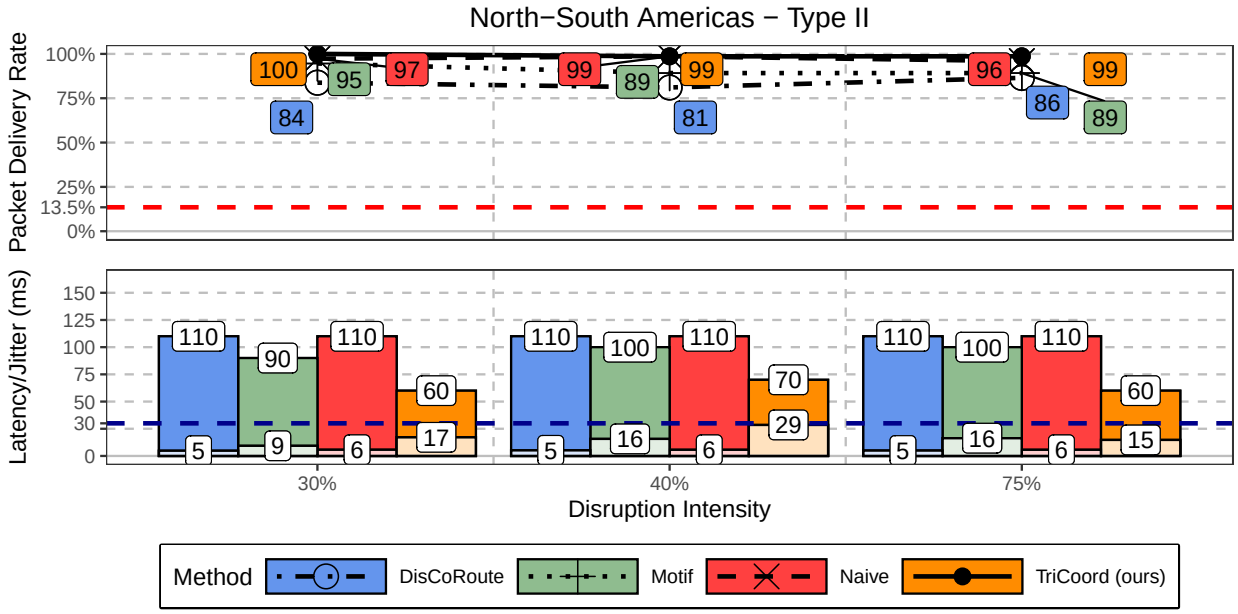


Figure A.8: Comparative performance of four dynamic routing algorithms in the North-South Americas scenario under Type II adversarial disruptions. Packet delivery rate is shown in upper point graph, with packet latency and latency jitter represented in lower bar graph and bar fill pattern, respectively. Red dashed line (top) marks packet delivery threshold below which routing is ineffectual during adversarial disruption. Blue dashed line (bottom) marks jitter threshold above which degrades user experience. Results are shown for 30%, 40%, and 75% disruption intensities.

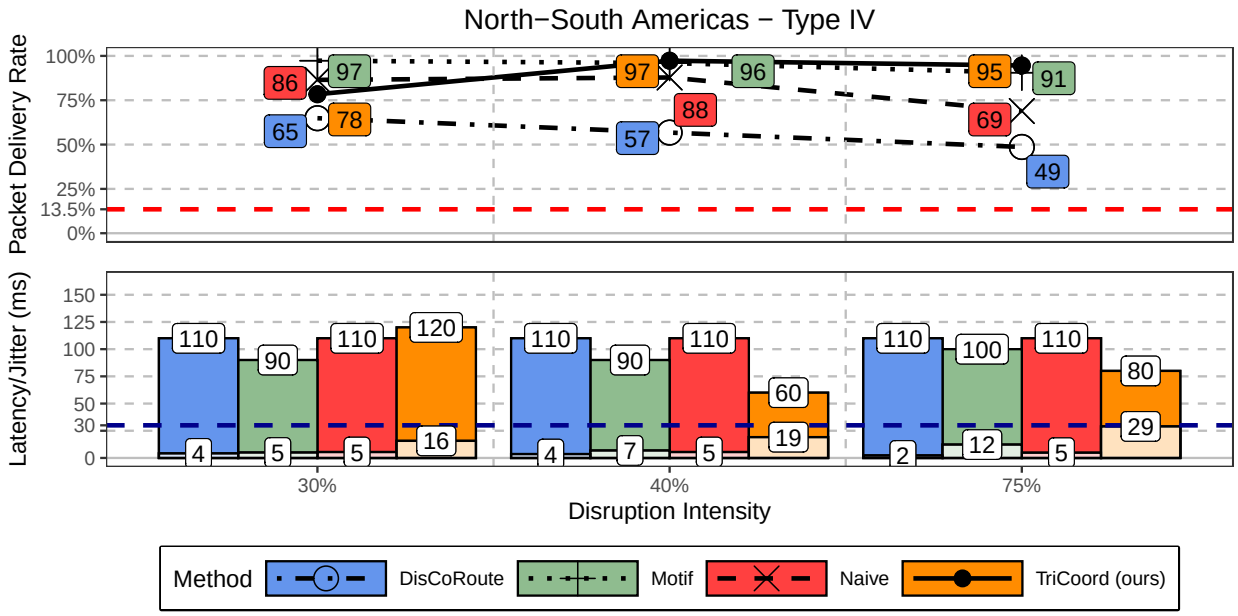


Figure A.9: Comparative performance of four dynamic routing algorithms in the North-South Americas scenario under Type IV adversarial disruptions. Packet delivery rate is shown in upper point graph, with packet latency and latency jitter represented in lower bar graph and bar fill pattern, respectively. Red dashed line (top) marks packet delivery threshold below which routing is ineffectual during adversarial disruption. Blue dashed line (bottom) marks jitter threshold above which degrades user experience. Results are shown for 30%, 40%, and 75% disruption intensities.

Appendix B

Secure and Energy-Efficient Cryptography for Cislunar and Deep Space Platforms Supplementary Content

B.1 Software Versions

Software:	**Version Used**:
OpenSSL	3.4.0-dev
Open Quantum Safe Library (libOQS)	0.11.1-dev
OpenSSL oqs-provider	0.7.1-dev
CRIU	3.18
perf	6.8.12

Table B.1: List of software and version numbers used in evaluation.

Appendix C

Bundle Protocol version 7 Validation Supplement

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation;
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 */

/*
Updates made by: Alexander Kedrowitsch <alexk1@vt.edu>

Aggregate changes for commits in range: ca769ae..f12268c

Modified/Added Function: Receive_char_array
  - Related commit message: Completed implementation of receive bundle callbacks.  Nodes can register a
  ↪  callback function that will be called whenever they successfully process a bundle that is addressed to
  ↪  them

Modified/Added Function: print_Ipv4InterfaceAddress
  - Related commit message: Corrected issue where updates to newly created bundles were not reflected in CBOR
  ↪  conversion under certain conditions
```

```
27
28  */
29
30  // Network topology
31  //             link1           link2
32  //       n0 ----------- n1 ----------- n2
33  //            500 Kbps       500 Kbps
34  //             20 ms          20 ms
35  //
36  // - Flow from n0 to n2 using bundle protocol.
37  // - link2 is set to 'down' until second '1' when it is set to 'up'
38  // - Tracing of queues and packet receptions to file "bundle-protocol-multihop-lsc-ltp.tr"
39  //   and pcap tracing available when tracing is turned on.
40
41  #include <string>
42  #include <fstream>
43  #include "ns3/core-module.h"
44  #include "ns3/point-to-point-module.h"
45  #include "ns3/internet-module.h"
46  #include "ns3/network-module.h"
47  #include "ns3/bp-endpoint-id.h"
48  #include "ns3/bundle-protocol.h"
49  #include "ns3/bp-static-routing-protocol.h"
50  #include "ns3/bundle-protocol-helper.h"
51  #include "ns3/bundle-protocol-container.h"
52  #include "ns3/bp-ltp-cla-protocol.h"
53
54  #include "ns3/ltp-protocol-helper.h"
55  #include "ns3/ltp-protocol.h"
56
57  using namespace ns3;
58
59  NS_LOG_COMPONENT_DEFINE ("BundleProtocolMultihopLinkStatusChangeLtp");
60
61  void Send_char_array (Ptr<BundleProtocol> sender, char* data, BpEndpointId src, BpEndpointId dst)
62  {
63    NS_LOG_INFO ("Sendpacket(...) called.");
64    uint32_t size = strlen(data);
65    std::cout << Simulator::Now ().GetMilliSeconds () << " Send a PDU with size " << size << ", containing:" <<
      ↪  std::endl << data << std::endl;
66    sender->Send_data (reinterpret_cast<const uint8_t*>(data), size, src, dst);
```

```
67  }
68
69  void Receive_char_array (Ptr<BundleProtocol> receiver, BpEndpointId eid)
70  {
71
72    std::vector<uint8_t> data = receiver->Receive_data (eid);
73    NS_LOG_INFO ("Receive(..) called.");
74    while (!data.empty())
75      {
76        uint32_t size = data.size();
77        std::cout << Simulator::Now ().GetMilliSeconds () << " Receive bundle size " << size << std::endl;
78        char* buffer = new char[size+1];
79        std::copy(data.begin(), data.end(), buffer);
80        buffer[size] = '\0'; // Null terminating char_array to ensure cout doesn't overrun when printing
81        std::cout << "Data received: " << std::endl << buffer << std::endl;
82
83        delete [] buffer;
84        // Try to get another packet
85        data = receiver->Receive_data (eid);
86      }
87  }
88
89  void RecvCallback (Ptr<BundleProtocol> receiver)
90  {
91    NS_LOG_FUNCTION ("RecvCallback called for " << receiver);
92    BpEndpointId eid = receiver->GetBpEndpointId ();
93    Receive_char_array (receiver, eid);
94  }
95
96  void SetRecvCallback (Ptr<BundleProtocol> receiver)
97  {
98    Callback<void, Ptr<BundleProtocol> > cb = MakeCallback (&RecvCallback);
99    receiver->SetRecvCallback (cb);
100 }
101
102 void Register (Ptr<BundleProtocol> node, BpEndpointId eid, uint64_t l4Address)
103 {
104     std::cout << Simulator::Now ().GetMilliSeconds () << " Registering external node " << eid.Uri () <<
        ↪  std::endl;
105     node->ExternalRegisterLtp (eid, 0, true, l4Address);
106 }
```

```
107
108 void print_Ipv4InterfaceAddress (Ptr<Ipv4Interface> iface_node)
109 {
110   Ipv4InterfaceAddress iface_node_i_address = iface_node->GetAddress (0);
111   Ipv4Address iface_node_address = iface_node_i_address.GetLocal ();
112
113   std::cout << "Node 1, interface 2 address is: ";
114   iface_node_address.Print(std::cout);
115   std::cout << std::endl;
116 }
117
118 void
119 SetRedMode (Ptr<BundleProtocol> bpNode, uint8_t redMode)
120 {
121   std::cout << Simulator::Now ().GetMilliSeconds () << " Setting Node " << bpNode->GetBpEndpointId ().Uri ()
    ↪  << " red mode to " << (uint32_t) redMode << std::endl;
122   DynamicCast<ns3::BpLtpClaProtocol> (bpNode->GetCla ())->SetRedDataMode (redMode);
123 }
124
125 int
126 main (int argc, char *argv[])
127 {
128
129   //bool tracing = true;
130   bool tracing = false;
131
132   ns3::PacketMetadata::Enable ();
133
134   NS_LOG_INFO ("Create bundle nodes.");
135   NodeContainer nodes, link1_nodes, link2_nodes;
136   nodes.Create (3);
137
138   link1_nodes.Add(nodes.Get(0));
139   link1_nodes.Add(nodes.Get(1));
140
141   link2_nodes.Add(nodes.Get(1));
142   link2_nodes.Add(nodes.Get(2));
143
144
145   NS_LOG_INFO ("Create channels.");
146
```

```
147     InternetStackHelper internet;
148     Ipv4ListRoutingHelper routingList;
149     Ipv4StaticRoutingHelper staticRoutingHelper;
150     routingList.Add(staticRoutingHelper, 0);
151
152     internet.SetRoutingHelper(routingList);
153     internet.Install (nodes);
154
155     std::ostringstream channelDelay;
156     channelDelay << "20ms";
157     PointToPointHelper pointToPoint;
158     pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("500Kbps"));
159     pointToPoint.SetChannelAttribute ("Delay", StringValue (channelDelay.str ()));
160
161     NetDeviceContainer link1_devices, link2_devices;
162     link1_devices = pointToPoint.Install (link1_nodes);
163     link2_devices = pointToPoint.Install (link2_nodes);
164
165
166     NS_LOG_INFO ("Assign IP Addresses.");
167     Ipv4AddressHelper ipv4;
168
169     ipv4.SetBase ("10.1.1.0", "255.255.255.0");
170     Ipv4InterfaceContainer link1_i = ipv4.Assign (link1_devices);
171
172     ipv4.SetBase ("10.1.2.0", "255.255.255.0");
173     Ipv4InterfaceContainer link2_i = ipv4.Assign (link2_devices);
174
175     Ptr<Ipv4> ipv4_node0 = link1_nodes.Get(0)->GetObject<Ipv4> ();
176     Ptr<Ipv4StaticRouting> staticRouting_node0 = staticRoutingHelper.GetStaticRouting (ipv4_node0);
177
178     staticRouting_node0->AddNetworkRouteTo (Ipv4Address ("0.0.0.0"), Ipv4Mask ("0.0.0.0"), 1); // setting node0
        ↪  interface 1 as default route (interface 0 is loopback)
179
180     Ptr<Ipv4> ipv4_node2 = link2_nodes.Get(1)->GetObject<Ipv4> ();
181     Ptr<Ipv4StaticRouting> staticRouting_node2 = staticRoutingHelper.GetStaticRouting (ipv4_node2);
182
183     staticRouting_node2->AddNetworkRouteTo (Ipv4Address ("0.0.0.0"), Ipv4Mask ("0.0.0.0"), 1);  // setting node2
        ↪  interface 1 as default route (interface 0 is loopback)
184
185     NS_LOG_INFO ("Create bundle applications.");
```

```
186     // Now install LTP on the nodes
187
188     // Defines the ClientService ID code that will be using the Ltp protocol in both sides of the link
189     // Bundle protocol code as defined by IANA: "LTP Client Service Identifiers" referenced in RFC 7116
190     uint64_t ClientServiceId = 1; // 1 -- bundle protocol service id
191
192     // Create LtpIpResolution tables to perform mappings between Ipv4 adresses and LtpEngineIDs
193     // Receving node requires its own LtpIpResolutionTable because it maps the Forwarder's LtpEngineId to a
        ↪  different IP address than the Sender does
194     Ptr<ns3::ltp::LtpIpResolutionTable> ltpRouting =  CreateObjectWithAttributes<ns3::ltp::LtpIpResolutionTable>
        ↪  ("Addressing", StringValue ("Ipv4"));
195     Ptr<ns3::ltp::LtpIpResolutionTable> ltpReceiverRouting =
        ↪  CreateObjectWithAttributes<ns3::ltp::LtpIpResolutionTable> ("Addressing", StringValue ("Ipv4"));
196
197
198     // Use a helper to create and install Ltp Protocol instances in the nodes.
199     uint64_t baseLtpEngineId = 0;
200     ns3::ltp::LtpProtocolHelper ltpHelper;
201     ltpHelper.SetAttributes (''CheckPointRtxLimit",  UintegerValue (20),
202                              "ReportSegmentRtxLimit", UintegerValue (20),
203                              "RetransCyclelimit",  UintegerValue (20),
204                              "OneWayLightTime", StringValue (channelDelay.str ()));
205     ltpHelper.SetLtpIpResolutionTable (ltpRouting);
206     ltpHelper.SetBaseLtpEngineId (baseLtpEngineId);
207
208     ltpHelper.InstallAndLink (nodes); // This installs Ltp and sets linking for one-way loop (node0 -> node1 ->
        ↪  node2 -> node0 -> ...)
209
210     // get node L4 addresses
211     uint64_t senderL4addr = link1_nodes.Get (0)->GetObject<ns3::ltp::LtpProtocol> ()->GetLocalEngineId ();
212     uint64_t forwarderL4addr = link1_nodes.Get (1)->GetObject<ns3::ltp::LtpProtocol> ()->GetLocalEngineId ();
213     uint64_t recvL4addr = link2_nodes.Get (1)->GetObject<ns3::ltp::LtpProtocol> ()->GetLocalEngineId ();
214
215     // Build reverse order links
216     ltpHelper.Link(link1_nodes.Get (1), senderL4addr);  // Link forwarder to Sender (node0 <- node1)
217     ltpHelper.Link(link2_nodes.Get (1), forwarderL4addr); // Link Receiver to Forwarder (node1 <- node2)
218
219     // Create binding of Forwarder's link2 IP address to its LtpEngineId for the Receiver to use
220     ltpReceiverRouting->AddBinding(forwarderL4addr, link2_i.GetAddress (0), 1113); // Add binding between
        ↪  LtpEngineId and Ipv4 address; 1113 is default server port
221     // Set Receiver to use the LtpIpResolutionTable
```

```
222 nodes.Get (2)->GetObject<ns3::ltp::LtpProtocol> ()->SetIpResolutionTable (ltpReceiverRouting); // Set the
    ↪ LtpIpResolutionTable for the Receiver node
223
224 // Configure the BP nodes for LTP
225 std::ostringstream l4type;
226 l4type << "Ltp";
227 Config::SetDefault ("ns3::BundleProtocol::L4Type", StringValue (l4type.str ()));
228 Config::SetDefault ("ns3::BundleProtocol::BundleSize", UintegerValue (400));
229
230 // build endpoint ids
231 BpEndpointId eidSender ("dtn", "node0");
232 BpEndpointId eidForwarder ("dtn", "node1");
233 BpEndpointId eidRecv ("dtn", "node2");
234
235 // set bundle static routing for sender
236 Ptr<BpStaticRoutingProtocol> route_sender = CreateObject<BpStaticRoutingProtocol> (); // static routes for
    ↪ sender
237 route_sender->AddRoute (eidRecv, eidForwarder); // dest: recv; next_hop: forwarder
238
239 // set bundle static routing for forwarder
240 Ptr<BpStaticRoutingProtocol> route_forwarder = CreateObject<BpStaticRoutingProtocol> (); // static routes
    ↪ for forwarder
241
242 // set bundle static routing for recv
243 Ptr<BpStaticRoutingProtocol> route_recv = CreateObject<BpStaticRoutingProtocol> (); // static routes for
    ↪ recv
244 route_recv->AddRoute (eidSender, eidForwarder); // dest: sender; next_hop: forwarder
245
246 // sender
247 BundleProtocolHelper bpSenderHelper;
248 bpSenderHelper.SetRoutingProtocol (route_sender);
249 bpSenderHelper.SetBpEndpointId (eidSender);
250 BundleProtocolContainer bpSenders = bpSenderHelper.Install (link1_nodes.Get (0));
251 bpSenders.Start (Seconds (0.2));
252 bpSenders.Stop (Seconds (10.0));
253
254 // forwarder
255 BundleProtocolHelper bpForwarderHelper;
256 bpForwarderHelper.SetRoutingProtocol (route_forwarder);
257 bpForwarderHelper.SetBpEndpointId (eidForwarder);
258 BundleProtocolContainer bpForwarders = bpForwarderHelper.Install (link1_nodes.Get (1));
```

```
259     bpForwarders.Start (Seconds (0.1));
260     bpForwarders.Stop (Seconds (10.0));
261
262     // receiver
263     BundleProtocolHelper bpReceiverHelper;
264     bpReceiverHelper.SetRoutingProtocol (route_recv);
265     bpReceiverHelper.SetBpEndpointId (eidRecv);
266     BundleProtocolContainer bpReceivers = bpReceiverHelper.Install (link2_nodes.Get (1));
267     bpReceivers.Start (Seconds (0.0));
268     bpReceivers.Stop (Seconds (10.0));
269
270     // register external nodes with each node
271     Simulator::Schedule (Seconds (0.2), &Register, bpSenders.Get (0), eidForwarder, forwarderL4addr);
272     Simulator::Schedule (Seconds (0.2), &Register, bpSenders.Get (0), eidRecv, recvL4addr);
273
274     Simulator::Schedule (Seconds (0.1), &Register, bpForwarders.Get (0), eidSender, senderL4addr);
275     Simulator::Schedule (Seconds (0.1), &Register, bpForwarders.Get (0), eidRecv, recvL4addr);
276
277     Simulator::Schedule (Seconds (0.0), &Register, bpReceivers.Get (0), eidForwarder, forwarderL4addr);
278     Simulator::Schedule (Seconds (0.0), &Register, bpReceivers.Get (0), eidSender, senderL4addr);
279
280     // Get forwarder's link2 interface and set to down
281     Ptr<Ipv4L3Protocol> ipv4l3_node1 = link2_nodes.Get(0)->GetObject<Ipv4L3Protocol> ();
282     Ptr<Ipv4Interface> iface2_node1 = ipv4l3_node1->GetInterface(2);
283
284     // Shutting node1 interface 2 down
285     Simulator::Schedule (Seconds (0.211), &Ipv4Interface::SetDown,iface2_node1);
286
287     char data[] = "The Senate of the United States shall be composed of two Senators from each State, "
288                   "chosen by the Legislature thereof, for six Years; and each Senator shall have one Vote."
289                   "Immediately after they shall be assembled in Consequence of the first Election, they shall"
290                   " be divided as equally as may be into three Classes. The Seats of the Senators of the"
291                   " first Class shall be vacated at the Expiration of the second Year, of the second Class at"
292                   " the Expiration of the fourth Year, and of the third Class at the Expiration of the sixth"
293                   " Year, so that one third may be chosen every second Year; and if Vacancies happen by "
294                   "Resignation, or otherwise, during the Recess of the Legislature of any State, the Executive"
295                   " thereof may make temporary Appointments until the next Meeting of the Legislature, which"
296                   " shall then fill such Vacancies. No Person shall be a Senator who shall not have attained"
297                   " to the Age of thirty Years, and been nine Years a Citizen of the United States, and who"
298                   " shall not, when elected, be an Inhabitant of that State for which he shall be chosen."
299                   "The Vice President of the United States shall be President of the Senate, but shall have "
```

```
300                 "no Vote, unless they be equally divided.  The Senate shall chuse their other Officers, "
301                 "and also a President pro tempore, in the Absence of the Vice President, or when he shall"
302                 " exercise the Office of President of the United States. The Senate shall have the sole "
303                 "Power to try all Impeachments. When sitting for that Purpose, they shall be on Oath or"
304                 " Affirmation. When the President of the United States is tried, the Chief Justice shall"
305                 " preside: And no Person shall be convicted without the Concurrence of two thirds of the "
306                 "Members present. Judgment in Cases of Impeachment shall not extend further than to removal"
307                 " from Office, and disqualification to hold and enjoy any Office of honor, Trust or Profit"
308                 " under the United States: but the Party convicted shall nevertheless be liable and subject"
309                 " to Indictment, Trial, Judgment and Punishment, according to Law.";
310
311   // setting LTP red mode
312   Simulator::Schedule (Seconds (0.1), &SetRedMode, bpSenders.Get (0), 1); // 0 = no red data; 1 = slim red
      ↪  data; 2 = robust red data, 3 = all red data
313
314   // sending data bundle
315   NS_LOG_INFO ("Sending data of size: " << strlen(data) << std::endl);
316   Simulator::Schedule (Seconds (0.3), &Send_char_array, bpSenders.Get (0), data, eidSender, eidRecv);
317
318   // set forwarder's link2 interface to up
319   Simulator::Schedule (Seconds (1), &Ipv4Interface::SetUp,iface2_node1);
320
321   // receive function
322   Simulator::Schedule (Seconds (1.5), &Receive_char_array, bpReceivers.Get (0), eidRecv);
323
324   // receive function
325   //Simulator::Schedule (Seconds (3), &Receive_char_array, bpReceivers.Get (0), eidRecv);
326   Simulator::Schedule (Seconds (0), &SetRecvCallback, bpReceivers.Get (0));
327
328   if (tracing)
329     {
330       AsciiTraceHelper ascii;
331       pointToPoint.EnableAsciiAll (ascii.CreateFileStream ("bundle-protocol-multihop-lsc-ltp.tr"));
332       pointToPoint.EnablePcapAll ("bundle-protocol-multihop-lsc-ltp", false);
333     }
334
335   NS_LOG_INFO ("Run Simulation.");
336   Simulator::Stop (Seconds (4.0));
337   Simulator::Run ();
338   Simulator::Destroy ();
339   NS_LOG_INFO ("Done.");
```

```
340
341 }
```