Orbital Edge Computing: Nanosatellite Constellations as a New Class of Computer System

Bradley Denby
bdenby@andrew.cmu.edu
Carnegie Mellon University

Brandon Lucia
blucia@andrew.cmu.edu
Carnegie Mellon University

https://abstract.ece.cmu.edu/space

Abstract

Advances in nanosatellite technology and a declining cost of access to space have fostered an emergence of large constellations of sensor-equipped satellites in low-Earth orbit. Many of these satellite systems operate under a "bent-pipe" architecture, in which ground stations send commands to orbit and satellites reply with raw data. In this work, we observe that a bent-pipe architecture for Earth-observing satellites breaks down as constellation population increases. Communication is limited by the physical configuration and constraints of the system over time, such as ground station location, nanosatellite antenna size, and energy harvested on orbit. We show quantitatively that nanosatellite constellation capabilities are determined by physical system constraints.

We propose an Orbital Edge Computing (OEC) architecture to address the limitations of a bent-pipe architecture. OEC supports edge computing at each camera-equipped nanosatellite so that sensed data may be processed locally when downlinking is not possible. In order to address edge processing latencies, OEC systems organize satellite constellations into computational pipelines. These pipelines parallelize both data collection and data processing based on geographic location and without the need for cross-link coordination. OEC satellites explicitly model constraints of the physical environment via a runtime service. This service uses orbit parameters, physical models, and ground station positions to trigger data collection, predict energy availability, and prepare for communication. We show that an OEC architecture can reduce ground infrastructure over 24× compared to a bent-pipe architecture, and we show that pipelines can reduce system edge processing latency over 617×.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ASPLOS '20, March 16–20, 2020, Lausanne, Switzerland

© 2020 Association for Computing Machinery.

ACM ISBN 978-1-4503-7102-5/20/03…$15.00
https://doi.org/10.1145/3373376.3378473

CCS Concepts. • **Hardware → Emerging architectures**; **Emerging simulation**; • **Computer systems organization → Embedded software**; **Embedded hardware**.
Keywords. orbital edge computing, intermittent computing, nanosatellites

ACM Reference Format:
Bradley Denby and Brandon Lucia. 2020. Orbital Edge Computing: Nanosatellite Constellations as a New Class of Computer System. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '20), March 16–20, 2020, Lausanne, Switzerland* . ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3373376.3378473

1 Introduction

A resurgence in the space industry [13, 23, 67, 98], concurrent with standardization of nanosatellite form factors [69] and a declining cost of access to space [31], has stimulated an exponential growth in nanosatellite launches over the past two decades [85]. The largest commercial satellite constellations today consist of hundreds of Earth-observing, camera-equipped nanosatellites [12, 59], each measuring centimeters, massing a few kilograms, and costing only thousands of USD. These emerging systems are a stark contrast to the extremely expensive, monolithic space vehicles of the past. For example, the $192,000,000, 500 kg Earth Observing-1 (EO-1) is a "one-of-a-kind" [70] satellite operating for 16 years under a NASA ground control team. This single-satellite remote sensing mission depended on NASA's extensive ground infrastructure to support its communication model: operators manually schedule communication on ground stations shared among all other space missions. The EO-1 mission terminated when its ground support was de-funded, leaving no one to schedule communication and data management operations. As nanosatellites proliferate, the viability of building and operating a manual, bent-pipe system architecture diminishes. The scale of this challenge is increasing; several commercial ventures have announced plans to deploy satellite constellations consisting of *thousands* of devices over the next ten years [34, 40–42, 76, 77, 86–88].

A trend toward massive constellations of low Earth orbit (LEO) nanosatellites demands a new architecture for space systems. As with large, expensive space vehicles of the past,

nanosatellite constellations today still rely on a communication model that sends remote control commands to orbit and delivers sensed data to Earth [20]; this design is referred to as a "bent pipe" by space system architects [58]. Momentum towards large constellations of nanosatellites requires a reimagining of space systems as distributed, edge-sensing and edge-computing systems. As work on warehouse scale computing [6] did for datacenters-as-computers, we aim to raise awareness of system-level research questions for *orbital edge computer systems* equipped with high-datarate cameras and sensors. This work characterizes and addresses computer hardware and software design challenges for orbital edge computer systems, many of which stem from physical deployment constraints and limitations inherent to ground infrastructure.

Addressing the challenges of the orbital edge is a timely and important problem due to the recent proliferation of nanosatellite systems. Space system architects are eschewing large, costly (e.g. $650,000,000 [28]), "exquisite" [89] satellites for constellations of small, inexpensive (e.g. $65,000 ea.) "CubeSats" [69]. Commercial efforts [12, 59] use this 10,000× lower per-device cost to deploy large, sensor-equipped nanosatellite constellations to LEO and observe the planet with high temporal resolution. Such systems create new capabilities for precision agriculture, environmental and infrastructure monitoring, humanitarian assistance and disaster relief, security, climate, and other commercial uses.

Challenges faced by existing systems under a bent-pipe architecture stem from fundamental physical constraints. The time-varying relationship between the geographic location of ground stations and the orbital position of nanosatellites imposes limitations on link availability and can lead to high downlink latencies. Intermittently available downlinks incur high latency between data collection and data processing in existing systems that simply downlink raw observations. Downlinks can be unreliable; one nanosatellite mission reports 88% packet loss [72]. Commercial ventures require complex, custom downlink solutions [20]. Shared "last mile" infrastructures [2, 99] aid availability but do not address the terrestrial centralization bottleneck. Limits on downlink bitrate prevent bent pipes from scaling to accommodate the extreme data volumes of large constellations and create a need for a new system architecture less reliant on communication.

On Earth, sensor systems increasingly leverage edge computing by performing sensor-local data processing in lieu of communication to a cloud datacenter [83]. While access to the cloud from the edge can accelerate computing [8], any benefits depend on backhaul network availability. High-datarate sensors deployed across large geographic environments face a network bottleneck from the sensor to the datacenter as datarate exceeds bandwidth [44, 83]. Processing data at the edge avoids high-bitrate infrastructure at each sensor and supports a larger population of deployed devices.

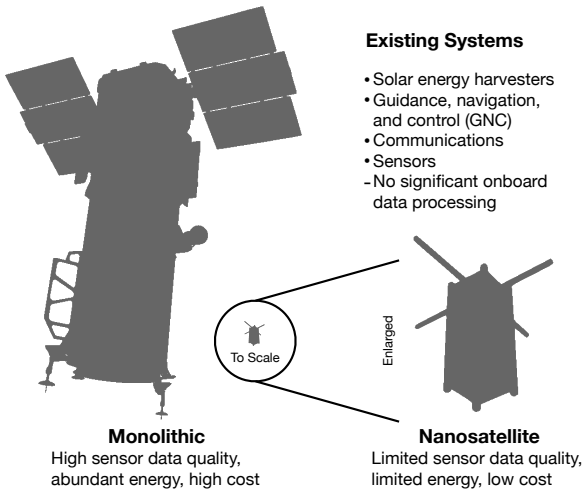

Figure 1. A comparison between a monolithic satellite and a nanosatellite. We propose augmenting existing systems by incorporating onboard visual processing components.

Edge processing avoids the privacy and security risks of multi-tenant infrastructures in shared datacenters [26, 56, 60].

Applying these terrestrial edge computing techniques directly to space systems is appealing, but nanosatellite constellations are subject to a unique set of operating constraints that typically do not affect terrestrial edge systems. In space, unlike on Earth, all energy for computation and communication must be harvested from the environment — plugging into a power grid is not an option. The small size of a nanosatellite, which is dictated by the cubesat standard, limits solar panel surface area and thus limits power. Unlike on Earth, the quality of visual data in space is fundamentally limited not only by onboard sensors, but also by chassis size (which limits focal length) and orbit altitude (which limits optical resolution). Communication bitrate, which is affected by orbit parameters, ground station capability, and ground station location, dictates the amount of data satellites buffer between downlinks. Any viable orbital edge computer system must directly address these unique physical constraints.

We propose *Orbital Edge Computing* (OEC) as an alternative to existing nanosatellite constellation bent-pipe architectures. OEC colocates processing hardware with high-datarate spectral sensors in small, low-cost satellites. We characterize the physically-constrained design space of a computational nanosatellite, revealing fundamental limitations on data quality and computation inherent to state-of-the-art designs. Based on this design space study, we introduce *computational nanosatellite pipelines* (CNPs) as an organizational principle for OEC constellations. A CNP distributes sensing, processing, and communication across a constellation in order to remain within latency and energy envelopes.

We then develop `cote`[1], the first orbital edge computing simulator (`cote-sim`) and runtime service (`cote-lib`). `cote` physically models orbital mechanics and Earth rotation to track ground station and satellite positions over time. `cote` models data collection along each satellite ground track, as well as the energy and latency of sensing, computing, and communication for an entire constellation. `cote` is useful for mission design and simulation (`cote-sim`) and as an online runtime service for each nanosatellite and ground station (`cote-lib`).

We use `cote-sim` to quantitatively demonstrate the limitations of bent pipes, the advantages of OEC, and the benefits of nanosatellite pipelines. `cote-lib` runs on each device and provides continuous access to a physics-based model of the constellation and ground infrastructure in order to enable autonomy. By directly modeling the physics of the system, each satellite determines at runtime when to downlink, when to process locally, and how to distribute responsibilities across a pipeline *without the need for online coordination or cross-link communication.* `cote-lib` enables OEC by eliminating the reliance on remote control through a bent pipe.

In summary, this paper makes the following contributions:

- We demonstrate the limitations of bent pipes using a novel, physics-based simulator that includes orbital dynamics, communication, energy harvesting, and data collection; an OEC architecture can reduce ground station infrastructure over 24× compared to a bent-pipe architecture.
- We characterize the physical design space of an OEC device and identify key limitations that drive constellation design.
- We propose and evaluate computational nanosatellite pipelines, an organizational principle for OEC constellations that distributes work across devices; CNPs can reduce system edge processing latency over 617×.
- We present a runtime service deployed to each nanosatellite and ground station that models the constellation, ground infrastructure, and energy environment in order to autonomously schedule sensing, communication, and computing *without* the need for cross-link coordination.

2 Background on Nanosatellite Constellations

Momentum away from exquisite [89], monolithic satellites towards small, cheap nanosatellites reduces the cost of remote sensing in space by orders of magnitude. A *nanosatellite* has a mass between 1 kg and 10 kg, often adhering to the "CubeSat" standard [69] to enable use of commercial, off-the-shelf (COTS) components and avoid custom deployers [75]. A cubesat is physically constrained to 10 cm×10 cm×10 cm ("1U") volumes, with mass limited to 1.33 kg per 1U. This volume must house all sensors, actuators, and communication subsystems. Computers onboard existing nanosatellites are simple, low-performance systems for command and data handling (C&DH), guidance navigation and control (GNC), buffering sensor data, and communication. Virtually all nanosatellites today rely on a ground control segment to manage data.

A nanosatellite electrical power system (EPS) collects, stores, and distributes energy. Many low-cost cubesats avoid higher-risk, deployable solar arrays and instead rely on surface-mounted solar panels. As a result, the small size of the satellite constrains collected power to a few watts. Batteries must be small due to limited cubesat volume and mass. To prevent damage, batteries are heated in the cold of space (e.g. -40°C), incurring a power cost overhead [27]. Supercapacitor storage is less energy-dense, but has less mass, less volume, and avoids thermal issues; we focus on supercapacitor energy storage.

Figure 1 illustrates the magnitude of the shift from large, monolithic satellites to nanosatellites. Monolithic satellites are meters in size, thousands of kilograms in mass, collect kilowatts of power, and can cost over half a billion USD [21]. A nanosatellite is four orders of magnitude smaller (cubic centimeters), has three orders of magnitude less mass (a few kilograms), collects three orders of magnitude less power (a few watts), and has four orders of magnitude lower cost.

A *nanosatellite constellation* is a collection of nanosatellites that share a purpose. Existing nanosatellite constellations are coordinated from the ground, often to accomplish a remote sensing task (e.g. imaging the Earth). Today, commercial ventures leverage the relatively low per-device cost of nanosatellites to operate large constellations [12, 59]. In the future, public and private organizations expect to launch constellations of thousands of devices, each with high-datarate sensors and the capacity for more capable onboard computers.

A constellation consists of a *ground segment* and a *space segment*. In bent-pipe architectures, the ground segment consists of geographically-distributed, manually-controlled transceivers, and the space segment consists of remote--controlled satellites in one or more orbits. As we show quantitatively in Section 3.3 and Section 7.1, bent pipes break down as the amount of edge-sensed data increases. Further, limited link availability and bitrate bottlenecks can cause reconfiguration of a constellation to take days, weeks, or months [22]. These growing limitations of bent-pipe architectures motivate the OEC techniques presented in this work.

3 Challenges of Computational Nanosatellites

Computational nanosatellite architects face three key challenges. First, physical constraints on nanosatellite design (e.g.

[1] **C**omputing **o**n **t**he **e**dge. A cote is a shelter for carrier pigeons.

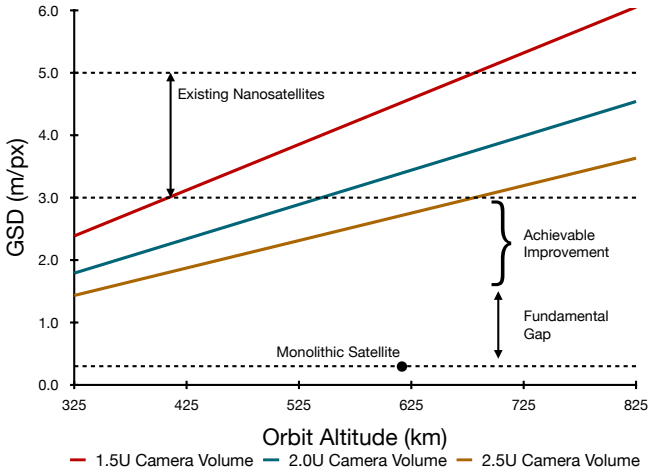

Figure 2. A 3U cubesat camera design space, assuming the smallest commercially available pixel sensor size (1.1 um)

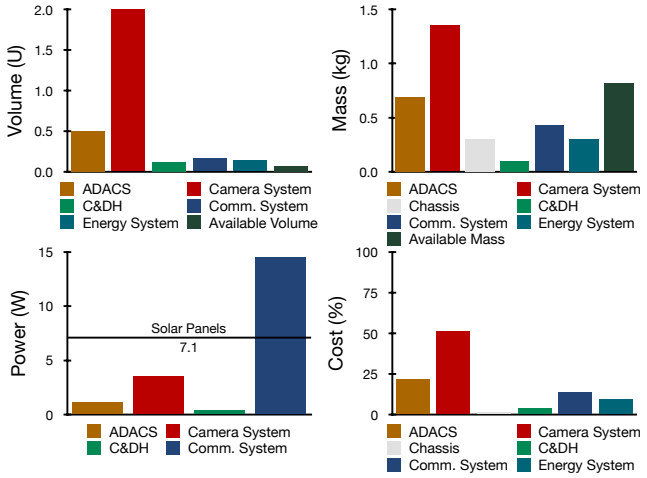

Figure 3. The volume, mass, power, and cost of an Earth-observing, 3U cubesat. These nanosatellites are constrained in volume and power, but not in mass or cost.

orbit altitude or volume limitations imposed by the cubesat standard) bound the achievable fidelity of visual data. Second, orbit characteristics determine data collection rate because satellite position and velocity dictate when and how often to capture data. Third, the relationship between orbit characteristics, Earth rotation, and ground station locations determines downlink availability, duration, and bitrate.

3.1 Data Quality is Physically Constrained

The physical design of a nanosatellite limits visual sensor data quality. Satellite image quality is measured by *ground sample distance* (GSD), i.e. the geographic distance represented between centers of adjacent pixels. As GSD decreases, image quality increases. Commercial, monolithic systems have GSD as low as 0.3 m/px [21], while commercial nanosatellite systems have GSD around 3.0 m/px [74].

Three parameters govern GSD: orbit altitude, camera focal length, and pixel sensor size. Merit is proportional to focal length and inversely proportional to altitude and sensor size.

Sensor size. We assume a COTS image sensor of at least 4096×3072 pixel sensors (i.e. 4K), each 1.1 µm [73] in size.

Orbit altitude. Orbit altitude is often between 325 km and 825 km for LEO. Higher altitudes support longer missions (years instead of weeks), but suffer worse image quality.

Focal length. The cubesat standard bounds camera focal length because components compete for limited volume. Earth-observation cubesats must include radios, energy storage, computing systems, and an attitude determination and control system (ADACS) to point the camera.

Designing for Low GSD. An Earth-observing satellite should optimize for low GSD. Figure 2 plots a cubesat design space, assuming a 3U volume (like existing, commercial systems [20]) and calculating GSDs with the pinhole camera model [35]. Each curve represents a different camera focal length. The data show that a 20 cm focal length provides a GSD of 2.26 m/px, which is 7.5× worse than a monolithic satellite but comparable to existing cubesats. Low GSD requires a long focal length, limiting non-camera components to a 1U volume. Figure 3 charts contributions of state-of-the-art nanosatellite subsystems [1, 4, 25, 27, 46–49, 52, 84] toward volume, mass, power, and cost, revealing that volume and power limit cubesat design while mass and cost do not.

3.2 Data Rate Depends on Orbit Parameters

The astrodynamics of a nanosatellite determine the optimal rate at which to collect images. This rate must be both frequent enough to cover the entire ground track, and infrequent enough to avoid redundant data. A satellite avoids collecting redundant images by making observations at the *ground track frame rate* (GTFR). The GTFR is the rate at which an entirely new geographic scene fills the camera view with no pixel overlapping a previous frame. The *ground track frame period* (GTFP) is the inverse of GTFR, i.e. the time a nanosatellite takes to pass over one ground track frame. To minimize data volume, camera sensors need not capture frames at rates higher than the GTFR. In order to achieve sufficient *coverage* of a ground track, a satellite or constellation must capture images individually or in aggregate at the GTFR. We discuss distributing sensing across a constellation in Section 4.2.

3.3 Bitrate Bottlenecks Constrain Constellations

Bent pipes break as nanosatellite constellation population increases due to limitations on downlink availability and bitrate. Under a bent-pipe architecture, each nanosatellite stores data (generally for minutes or hours) until it nears a ground station. During a downlink session, which lasts between a few seconds and ten minutes, the satellite transmits its unprocessed data. Under this model, existing systems [20]

experience a 5.5 h delay before data reach customers. Reconfiguring a constellation via uplink takes longer; initial configuration lasts months [12].

Bent-pipe architectures require many ground stations to support a large constellation. We evaluate existing, bent-pipe constellations with `cote-sim` in Section 7.1 and provide a simple motivating example here. A satellite in a polar orbit has access to all latitudes and, with a sun-synchronous orbit, ensures consistent pass times. Such a satellite at a 410 km altitude has an orbit period of 92.9 min and revisits the same point on Earth every two days [11]. Existing ground stations with a 200 Mbit/s downlink datarate [20] retrieve up to 15 GB of data per 10 min session. With these parameters, a ground station optimistically and ideally positioned to observe every pass (e.g. a polar station for a polar orbit) supports only 9 satellites per revolution. Supporting a future 1000-satellite constellation requires 112 of these ideally positioned stations.

Provisioning this costly ground station network may be pointless, because a large fraction of downlinked images contain no features of interest. And, as `cote-sim` reveals in Section 7.1, this estimate is overly ideal. It assumes that satellites are positioned in orbit such that all 112 ground stations are constantly in use, and it assumes that all satellites receive a full 10 min of downlink time. `cote-sim` reveals that neither of these assumptions hold true in realistic systems.

OEC Eliminates the Bent-Pipe Bottleneck OEC reduces the need for a large number of ground stations by processing images on orbit, downlinking only interesting images, and discarding or logging the rest. For example, assuming that machine inference identifies 0.75 GB of interesting data out of 15 GB of raw data, *all data downlink in only* 30 s *at* 200 Mbit/s. Instead of servicing just 9 satellites per revolution, each ground station supports 185 satellites, and a constellation of 1000 OEC satellites requires only 6 ground stations.

4 Orbital Edge Computing

OEC is a nanosatellite system design consisting of a set of organizational principles that relies on near-sensor processing in order to avoid the limitations of bent-pipe architectures. We first provide an overview of an individual OEC nanosatellite, i.e. a *computational nanosatellite*. We then describe a *computational nanosatellite pipeline* (CNP), which organizes a constellation into a parallel pipeline to hide processing latency by leveraging formation flying techniques [5, 62].

4.1 Computational Nanosatellites

A computational nanosatellite is a nanosatellite with several key changes to its computing hardware and power system.

Computing Hardware. A computational nanosatellite supplements existing sensing, communication, and control hardware with onboard computing. In this work, we characterize onboard computing with a Jetson TX2 industrial module. The

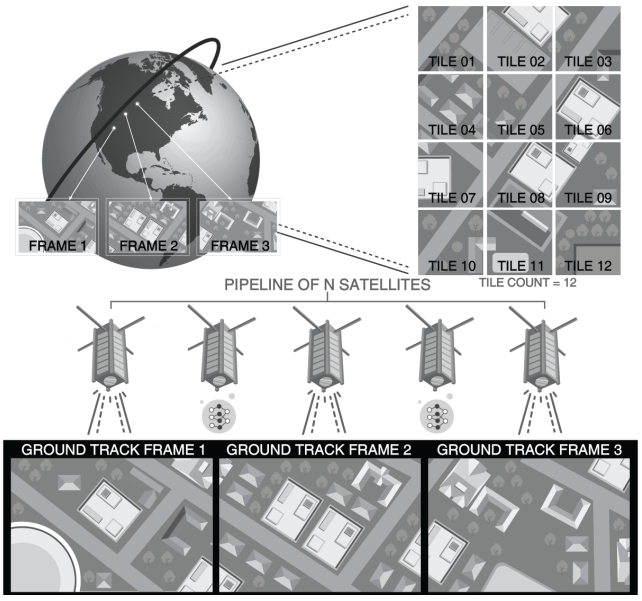

Figure 4. Top: The orbit determines the ground track, i.e. the locations over which a satellite passes. A ground track can be separated into a sequence of ground track frames. Typically, each frame is tiled before processing. Bottom: An illustration of a CNP. A satellite images a ground track frame and performs processing until arriving at the next frame.

Jetson TX2 includes a high-capability, low-power, efficient mobile GPU; the industrial variant is designed for extreme temperature environments. Its small volume allows for integration among all other necessary components within the 1U volume available to a 3U cubesat containing a 2U camera system. A 7.5 W power mode closely matches the 7.1 W input power provided by surface-mounted solar panels. The OEC computing model supports computing systems other than the Jetson by varying input performance and energy parameters.

Energy. A computational nanosatellite harvests and stores energy. In this work, the energy harvester is a low-risk, chassis-mounted solar cell array that avoids the mechanical complexity of deployable panels. A chassis-mounted array limits total solar cell area and, as a result, available power peaks at about 7.1 W. A high-density supercapacitor bank stores energy. Supercapacitors hold less total energy than batteries of the same volume, but offer several advantages. Supercapacitors charge quickly and provide immediate, high current; batteries charge slowly and are current-limited. Supercapacitors operate across the wide range of temperatures in space, while batteries fail in excessive heat or cold. OEC systems operate like intermittent systems [16, 17, 37–39, 61, 65, 93, 100], harvesting energy while sleeping to charge capacitors. When energy is sufficient, it performs its sensing, computing, or communication task.

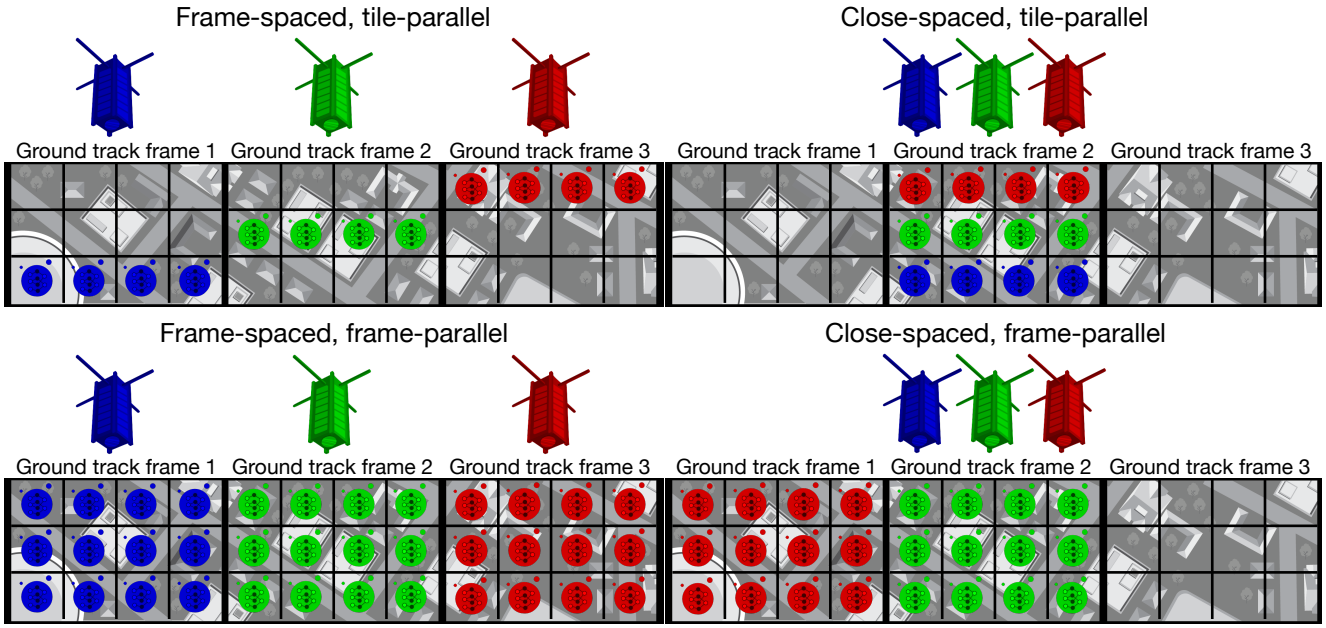

Figure 5. Four modes of operation for computational nanosatellite pipelines. Each of the four graphics depicts a snapshot in time. In the next time step, each satellite will have progressed one ground track frame to the right.

Operating Model. A computational nanosatellite operates by capturing an image and processing it locally instead of transmitting it to Earth through a bent pipe. The application determines the processing method. Examples include CNN-based image classification, object detection, and segmentation, or any other computation; Section 7 evaluates OEC systems with onboard machine inference. A typical OEC processing task identifies features of interest, separating them from raw sensor data. An OEC system discards uninteresting data and sends processed features of interest to Earth, using intelligent early discard as described by prior work [19].

4.2 Computational Nanosatellite Pipelines

An OEC system is energy and latency constrained. While processing a frame, a nanosatellite can capture but not process additional frames. A nanosatellite cannot capture a frame while sleeping. Effective OEC systems leverage the constellation as a whole to overcome energy and latency constraints.

A constellation of OEC nanosatellites overcomes the energy and latency limitations of individual satellites by organizing them into a *computational nanosatellite pipeline* (CNP). A CNP leverages existing formation flying techniques [5, 62] to orbit in a fixed configuration, parallelizing data collection and processing across a constellation. CNPs divide image frames into tiles; in some modes, tile processing is distributed among satellites to reduce system-level frame processing latency. Figure 4 illustrates a CNP operating on ground track frames, which are tiled during processing.

We identify and evaluate several modes of operation for CNPs: frame-spaced, tile-parallel; frame-spaced, frame-parallel; close-spaced, tile-parallel; and close-spaced, frame-parallel. Figure 5 illustrates each of these modes. *Frame-parallel* and *tile-parallel* describe how image processing tasks are distributed across a CNP. Under frame-parallel processing, each nanosatellite processes all tiles in each captured frame. Under tile-parallel processing, each nanosatellite processes a subset of tiles in each captured frame. *Frame-spaced* and *close-spaced* describe the physical configuration of a CNP. A frame-spaced pipeline places each nanosatellite exactly one GTF apart in distance. A close-spaced pipeline places each nanosatellite as close together as is feasible, e.g. meters or tens of meters apart, with a requirement that the end-to-end pipeline distance is less than the length of one GTF.

A frame-spaced, tile-parallel CNP separates devices by one GTF in distance; each device images every GTF (so long as there is sufficient energy) and processes a subset of tiles. A frame-spaced, frame-parallel CNP also separates devices by one GTF in distance; each device images a distinct subset of GTFs and processes all tiles in the frame. A close-spaced, tile-parallel CNP places devices close together in distance; each device images every GTF and processes a subset of tiles. A close-spaced, frame-parallel CNP also places devices close together in distance; each device images a distinct subset of GTFs and processes all tiles in the frame. An orbit-spaced constellation, in which satellites are evenly distributed across an orbit, is a modified version of a frame-spaced constellation

offering improved communication opportunities. Station-keeping, which allows a nanosatellite convoy to maintain consistent distances between adjacent devices (e.g. frame-spaced, close-spaced), requires formation flying techniques to compensate for atmospheric drag and other astrodynamic effects.

The number of devices in a CNP, or *pipeline depth*, increases the aggregate parallel work performed and the aggregate energy collected. Pipeline depth does not affect total data per revolution, because the number of frames per orbit remains constant. When the aggregate energy harvested by a CNP is less than the aggregate energy required to process all data, adding devices increases *coverage* (the fraction of GTFs captured per revolution) by increasing total system energy per revolution. When the aggregate energy harvested by a CNP is enough to process all data, a CNP may achieve full coverage. However, such a pipeline may still fall short of full coverage due to processing latency. If there are too few nanosatellites to complete parallel processing of all frames in one revolution, then coverage remains incomplete. Adding devices to the pipeline increases parallelism and decreases latency, eventually yielding a system capable of full coverage.

A computational nanosatellite pipeline requires propulsion and positioning. Unlike uncontrolled constellation configurations, formation flying requires each nanosatellite to have a propulsion system to correct for drag. One recent survey describes a variety of nanosatellite propulsion systems [91], including cold gas, liquid, ion thrusters, and hall effect propulsion systems. Additionally, 39 deployed and tested propulsion systems were evaluated in a recent survey of nanosatellite formation flying [5]. The wealth of recent research on propulsion makes CNP formation flying feasible.

To avoid the complications and expense of cross-link satellite communication, each device independently triggers camera captures based on position. The predetermined orbit and formation of a CNP allows capture coordinates to be defined before launch. Contemporary navigation constellation receivers track position with milliwatts of power, and they can be unlocked for high-velocity, high-altitude use in space [9]. We anticipate that cubesat pipelining may motivate further research in nanosatellite guidance, navigation, and control.

5 cote: A Model for Design and Control

`cote` is the first full-system model for orbital edge computing. The unique characteristics of OEC systems stem from the astrodynamics that govern them, giving rise to fundamental differences compared to terrestrial edge computing systems.

`cote` provides a detailed, physical simulation of OEC systems through an analytical model of orbital mechanics, the time-evolution of celestial and terrestrial coordinate frames, and physical bounds on communication bitrates, as well as a discrete-time model for harvested and buffered energy, sensing, data storage, and computing. These core components are shared across two applications: `cote-sim` and `cote-lib`. `cote-sim` supports OEC system design, and `cote-lib` supports dynamic, online autonomy at the edge.

5.1 Model Applications

`cote` has two main OEC applications: pre-mission simulation (`cote-sim`) and online, autonomous control (`cote-lib`).

`cote-sim` provides *offline simulation* of OEC systems for mission design, planning, and analysis. No existing space mission planning software (e.g. AGI's STK) supports modeling interactions between energy-constrained, intermittent computing, computational nanosatellite pipeline configurations, data collection at the GTFR, and communication. `cote-sim` fills this gap by integrating computing, communication, and energy as first-class counterparts to space mission dynamics.

`cote-lib` supports *online autonomy*, continuously running on each device in an OEC system. As described in Section 2, existing nanosatellites rely on a bent-pipe architecture for command and control instead of using autonomous control [36]. No existing online nanosatellite software system models the interaction between astrodynamics and intermittent computing in order to autonomously decide when to compute locally and when to communicate. `cote-lib` fills this gap by integrating space mission dynamics into an edge computing runtime system, enabling autonomous control at the orbital edge.

`cote-lib` runs continuously in the background on an OEC device, explicitly modeling ground station availability. `cote-lib` estimates latency and energy collection given input power and computing workload parameters. When an OEC satellite collects an image, it leverages `cote-lib` to determine whether to process an image locally or to transmit raw data to the ground. Thus, `cote-lib` enables an OEC satellite to adapt to changing orbit and power conditions in real-time; such fine-grained adaptation is impossible with high-latency, bent-pipe terrestrial control.

5.2 Model Design

`cote` integrates standard, analytical models of astrodynamics with discrete-time models for edge computing. In the following sections, we discuss major components of `cote`.

Time. Orbital edge computer systems deployed to LEO for Earth observation must be aware of time. While orbits are periodic, typically completing one LEO revolution in about 90 minutes, several factors make each revolution unique. Over the course of one revolution, the Earth rotates more than 22.5°. Due to the oblateness of the Earth, the orbit longitude of the ascending node precesses. Additionally, the Earth advances through its revolution around the Sun. Although a satellite returns to the same true anomaly value after each revolution, system conditions change. Distances to nearby ground stations shift, and the amount of harvestable energy changes with the relative position of the Sun. These changes

can be modeled and predicted by plotting them within a system of time.

`cote` tracks time with Universal Time, or UT1 [3], which measures the rotation of Earth relative to distant astronomical objects [68]. The more familiar Coordinated Universal Time, or UTC, is a civil time system closely aligned with UT1. The precise difference between UT1 and UTC (UT1−UTC) and the approximate difference (DUT1) are published by the International Earth Rotation and Reference Systems Service (IERS) in Bulletins A and D, respectively [50]. When the IERS projects a UT1−UTC value exceeding 0.9 s, it announces a leap second via Bulletin C [50, 68]

`cote` represents dates and times with the ISO 8601 standard [53]. To compare different points in the Gregorian calendar, `cote` converts each date and time to the equivalent Julian date [29]. A Julian date counts the number and fraction of days since the Julian epoch; the J2000 epoch is set to noon on January 1, 2000 with a Julian date of 2451545.0. `cote` represents a date and time with seven values: an integer for the Gregorian year, unsigned integers for the Gregorian month and day, and unsigned integers for the hour, minute, second, and nanosecond. Fractions of a second are rounded to the nearest nanosecond.

Coordinate Frames. In addition to time, orbital edge computer systems deployed to LEO for Earth observation must be aware of position. The position of an Earth-observation satellite determines which data can be collected and whether communication channels are available. `cote` supports three coordinate frames: the Earth-centered, inertial (ECI) frame, the latitude, longitude, and height above the ellipsoid (LLH) frame, and the south, east, z (SEZ) frame.

The origin of the ECI frame is located at Earth's center, i.e. the intersection of the north-south axis and the equatorial plane. The positive x-axis points toward the vernal equinox, the positive z-axis points toward the north pole, and the positive y-axis completes the right-handed Cartesian coordinate frame [92]. Because the Earth revolves around the Sun, the ECI frame is not truly inertial. Nevertheless, the ECI frame is widely used for celestial positioning. `cote` uses the ECI frame as the fundamental, universal coordinate frame.

The LLH frame, which is well-known due to widespread use of latitude and longitude, is tied to the reference ellipsoid defined in the World Geodetic System 1984 (WGS84) [71] (most recently updated in 2014) and, optionally, the World Geodetic System 1972 (WGS72) [81] for backwards compatibility. Modeling the oblate nature of the Earth, rather than approximating its shape as a sphere, is important particularly for establishing communication links via ground stations with narrow, high-gain antenna beams. Satellite sub-point latitude and longitude coordinates (useful for generating ground tracks) are calculated using an exact, non-iterative solution [7].

Ground station operations use the SEZ frame which, like the LLH frame, is non-inertial and rotates with the Earth. Axes point south, east, and up (normal to the local ellipsoid surface) [92]. `cote` uses standard transformations [80] for the azimuth and elevation of satellites in SEZ. `cote` uses the great circle arc [95] on the celestial sphere as the measure of distance between satellites in the ground station frame.

Orbital Mechanics. Given a model of time and position, orbital mechanics provides a means for modeling the state of a satellite relative to a rotating Earth. `cote` uses the *de facto* standard simplified general perturbation model (SGP4) as its orbital mechanics engine. The SGP4 model [43] consists of a collection of analytical equations tailored for Earth orbit. These analytical equations are seeded with a set of empirically determined measurements provided as two-line element sets (TLEs). SGP4 has become a de facto standard in the same way as GPS; the equations and source code for SGP4 are openly available [43], and TLEs for every public object in orbit around Earth are posted freely and regularly. Unlike more general, but less detailed, physics models [18], SGP4 is able to capture the effects of atmospheric drag through the empirical nature of the TLEs.

Communication. As we show in Section 3.3, the utility of OEC stems from the communication bottleneck between the space segment and the ground segment as constellation population increases. `cote` models the maximum achievable bitrate under received signal power for downlink, crosslink, and uplink channels [58]. Received signal power C is given by

$$C = PL_l G_t G_r \left(\frac{\lambda}{4\pi S}\right)^2, \tag{1}$$

where P represents transmit power, L_l represents the line loss factor at the transmitter, G_t represents the transmitter gain parallel to the separation vector, G_r represents the receiver gain parallel to the separation vector, λ is the center frequency of the channel, and S is the magnitude of the separation vector. Maximum bitrate $R_{\max}$ is given by

$$R_{\max} = B \log_2 (1 + C/N), \tag{2}$$

where B is the channel bandwidth, C is the received signal power as defined above, and N is the received noise power. The received noise power $N = kTB$, where k is the Boltzmann constant and T is the system noise temperature. System noise temperature of satellite and ground systems is modeled as described in [30, 51, 58].

Energy. In order to study the impact of data collection and computing on system energy, we develop an analytical model for energy harvesting, storage, and consumption. We use this model in Section 7 to simulate different system designs and evaluate their relative merits. Table 1 lists the input parameters to this model, which have been directly measured from our test hardware or taken from component datasheets. For energy harvesting, we model a simple solar cell at its maximum power point with an I-V curve consisting of the device short-circuit current at open-circuit voltage. For energy storage, we model a capacitor bank and its equivalent

Parameter	Value
Solar panel I_{MP} (A)	1.0034
Solar panel V_{MP} (V)	7.0290
Capacitance per capacitor (F)	1.0
ESR per capacitor (Ω)	0.84
Volume per capacitor (cm^3)	4.224
ADACS power (W)	1.13
Camera power - imaging (W)	3.5
Camera power - readout (W)	2.5
Jetson sleep power (W)	0.5
Jetson avg. power - detect (W)	11.3
Time step (s)	2.0×10^{-5}

Table 1. Energy model parameters. Increasing solar panel surface area by placing additional panels in series increases input power by increasing current. Total energy storage capacity is determined by the number of capacitors in parallel; increasing capacity also decreases effective ESR.

series resistance (ESR). The modeled load includes a Jetson TX2 module, a camera system, and an ADACS, each represented as variable resistors consuming energy over time as determined by the power mode at each time step. The power consumption of the TX2 module varies depending on the computation, while the power consumption of the camera varies depending on whether an image is being captured or read out for analysis. Our system simulations produce a time series of events and measurements, including device current and voltage at the granularity of the simulation time step and power state transition events.

Under the simple solar cell model described previously, an energy-harvesting, storage, and consumption system can be modeled over time with the following node voltage equation:

$$v(t) = \frac{\frac{q(t)}{C} + I_S R_{ESR} + \sqrt{(\frac{q(t)}{C} + I_S R_{ESR})^2 - 4 P_{MODE} R_{ESR}}}{2}. \tag{3}$$

Here, $q(t)$ is the charge held in the capacitor bank at time t; C is the total capacitance of the capacitor bank; I_S is the instantaneous current provided by the solar panel (either I_{MP} or, when $v(t) = V_{MP}$, zero); R_{ESR} is the equivalent series resistance of the capacitor bank; and P_{MODE} is the instantaneous power draw of all energy-consuming devices.

The current flowing into the energy-consuming systems is governed by the following equation.

$$I_D = \frac{P_{MODE}}{v(t)} \tag{4}$$

The current flowing into the capacitor bank is given by

$$I_C = I_S - I_D. \tag{5}$$

These equations hold under the condition that

$$\frac{q(t)}{C} \leq v(t) - I_S R_{ESR}, \tag{6}$$

i.e. so long as the capacitor charging rate is current-limited.

6 Methodology

We evaluate OEC systems running a remote sensing application on nanosatellite constellations.

We evaluate an OEC system in which each nanosatellite includes a Jetson TX2 module. Prior work shows that these systems remain effective in the space radiation environment [96]. Each nanosatellite collects data and either downlinks to a ground station or performs onboard machine inference. We use building footprint detection for the remote sensing application. We train the DetectNet [90] CNN on satellite images and ground-truth labels from the SpaceNet [94] dataset, and we evaluate performance on separate test data. The SpaceNet dataset is a collection of 0.3 m/px satellite images with labeled building footprints; we decimate the images to achieve higher GSDs. To evaluate the energy cost of computing on a satellite, we directly measure average power and latency of the inference application running on a Jetson TX2. We measure power with multimeters, recording current and voltage into the Jetson while workloads run from energy stored in a capacitor bank closely resembling our modeled power system. These operating energy values are an input to `cote` in its model of energy available to a nanosatellite during a deployment.

To quantify the limitations of bent-pipe architectures, we use `cote` to evaluate the performance of existing and future constellations. Space segments consist of polar (97.3°) orbits with 250 or 1000 satellites. This orbit is identical to one occupied by existing, deployed satellites; we use TLEs from nanosatellites operated by Planet. For each of the two space segments, we consider three constellation configurations, as described in Section 4.2: close-spaced, frame-spaced, and orbit-spaced. These configurations are compared to the current practice, bent-pipe configuration. We consider a polar ground segment consisting of two rings of ground stations, one at 87°N and one at 87°S. Each ring contains the same number of stations spaced evenly longitudinally.

The downlink frequency is centered at 8.15 GHz with a bandwidth of 20.0 MHz. We model nanosatellite patch antennas with a peak gain of 6.0 dB, and we model ground station receiving dishes with a peak gain of 44.1 dB.

7 Evaluation

In order to evaluate OEC as an architecture addressing the limitations of bent pipes, we first use `cote` to identify shortcomings in existing practice. We then characterize several benefits of computing onboard each nanosatellite instead of downlinking all data. We demonstrate that computational nanosatellite pipelines effectively hide frame processing latency, enabling persistent Earth observation at the GTFR with a feasible constellation population. We show that latency depends not only on individual device capability and constellation population, but also on the physical configuration of the CNP. We use our energy model to show that under

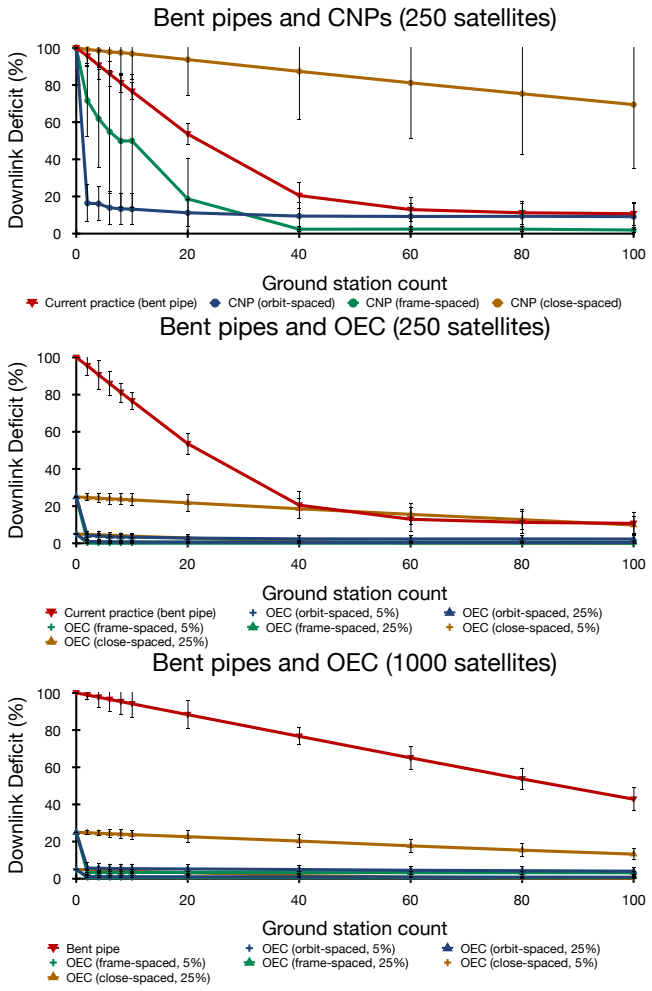

Figure 6. Top: Frame-spaced and orbit-spaced CNPs reduce downlink deficits without OEC by collecting data at the GTFR. Close-spaced CNPs without OEC increase downlink deficits due to communication contention. Middle: The same constellations enhanced with OEC; intelligent early discard leaves 5% to 25% of the data to downlink. Bottom: Larger constellations experience worse performance under current practice, while OEC performance remains consistent.

realistic, limited solar power and supercapacitor energy storage, CNPs achieve high ground track coverage. Finally, we demonstrate the benefits of the `cote-lib` runtime service by quantifying the long reconfiguration times inherent to bent pipes that `cote-lib` eliminates.

7.1 Bent Pipes Break Down

Figure 6, top, shows that bent pipes are fundamentally unscalable using a constellation of 250 nanosatellites in a 97.3° inclination orbit. The figure of merit, the *downlink deficit*, indicates the amount of data remaining on a nanosatellite (averaged across all satellites in the constellation) at the end of the time of interest. The modeled time spans two orbit revolutions over a real ground track. The plot shows that frame-spaced and orbit-spaced CNPs downlink a substantially larger fraction of data than a bent-pipe constellation or a close-spaced CNP. Frame-spaced and orbit-spaced CNPs are superior because they put distance between satellites, reducing *downlink contention*. Ground stations with high-gain antennas must choose which satellite to target for communication; if multiple satellites appear in view simultaneously, a ground station cannot service both for their entire respective passes. The effect of downlink contention is especially clear in a close-spaced CNP that does not use OEC to discard data intelligently before downlinking. In this configuration, ground stations remain idle for much of a revolution because satellites are clustered closely. Section 7.2 shows that the benefits of frame-spaced and orbit-spaced CNPs come at the cost of increased frame processing latency. Close-spaced CNPs suffer a data deficit without OEC but have much lower frame latency. The top plot only evaluates distributed data collection at the GTFR with CNPs; the remaining two plots evaluate the additional benefits of intelligent early discard with OEC.

Figure 6, middle, shows that by enabling intelligent early discard, OEC decreases data deficits and improves constellation scalability. The data show that, using a bent-pipe communication architecture, the downlink deficit plateaus above 0% even as polar ground station count increases. This plateau represents the residual data across a constellation waiting for downlink at the end of the experimental period of interest. OEC intelligently discards data, downlinking only data of interest and reaching a much lower downlink deficit plateau (a few percent) during the period of interest with 24× fewer ground stations than a bent-pipe configuration. To lower the plateau further, ground stations must be placed at lower latitudes. However, such stations can communicate with polar-orbit satellites only when the rotation of the Earth aligns them with a satellite ground track.

OEC makes constellations more scalable. Figure 6, bottom, repeats the experiments in Figure 6, middle, but increases constellation population to 1000 nanosatellites. These data show that the bent-pipe architecture does not scale, suffering a high downlink deficit even with many ground stations. In contrast, CNP configurations using OEC to discard data intelligently exhibit consistently low downlink deficits — even with an increased constellation population. The bent-pipe architecture requires 3.5× more ground stations to service 1000 nanosatellites than to service 250 nanosatellites. OEC CNPs can use the same ground infrastructure for constellations of both 250 and 1000 nanosatellites; OEC enables scaling up constellation population without increasing ground infrastructure.

7.2 Pipeline Configuration Impacts Latency

We use `cote-sim` to simulate the CNP configurations described in Section 4.2 and evaluate OEC *latency per ground*

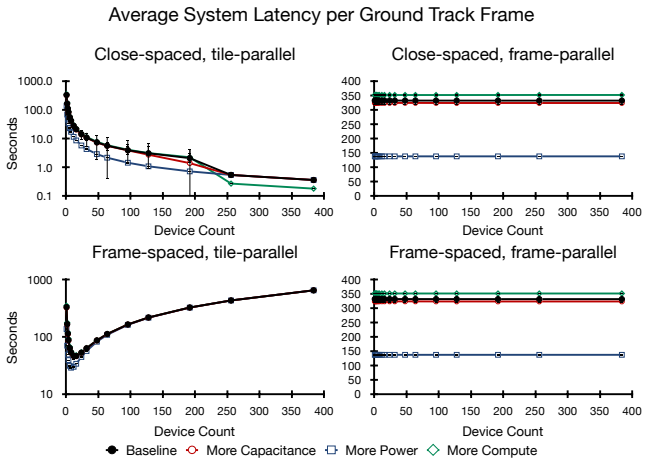

Figure 7. Left: Tile-parallel processing dramatically decreases system latency in close-spaced constellations. Improvements disappear when satellites are frame-spaced. Right: System latency is high with frame-parallel processing, but latency no longer depends on constellation configuration.

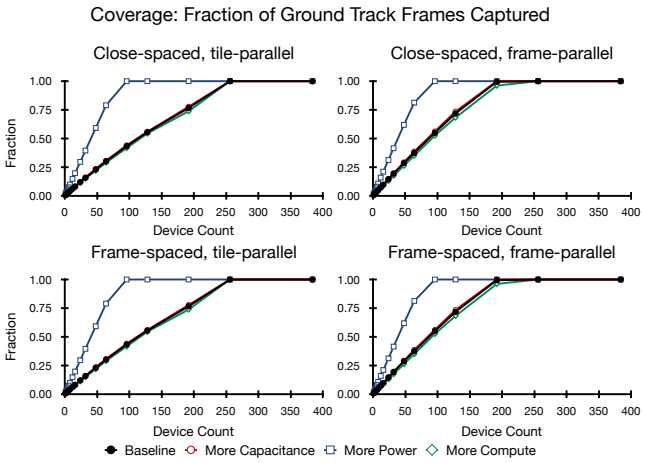

Figure 8. Coverage as a function of device count. Compared to the baseline, increasing overall system energy with deployable solar panels strongly increases coverage.

track frame, i.e. the time between frame capture and completion of frame analysis. This latency varies with parallelization across a constellation and physical distance between nanosatellites in a CNP. System latency determines the geographic location at which frame processing completes — an important factor when satellites can transmit results only when in range of a ground station.

The system characteristics of frame-parallel CNPs are identical regardless of whether satellites are frame-spaced or close-spaced. This fact is visible in Figure 7, right, by comparing the close-spaced, frame-parallel plot with the frame-spaced, frame-parallel plot. In a frame-parallel CNP, the complexity of close-spaced formation flying provides no benefit over an uncoordinated, frame-spaced configuration. Once each satellite has been assigned its GTFs, relative drift between devices does not impact GTF latency. However, the benefit of not requiring formation flying is tempered by uniformly high GTF latencies. Once the CNP pipeline is full, a new frame is completed every GTFP. Nevertheless, processing has long latency and the geographic location at which a particular GTF completes processing is far from the original observation location.

Figure 7, left, shows that tile-parallel CNPs require a close-spaced configuration to maintain low per-GTF latency. When devices are close-spaced, work is distributed among satellites and all tiles are processed shortly after all satellites observe a frame. However, frame-spaced satellites have high per-GTF latency with deeper pipelines. This effect emerges because, in a tile-parallel configuration, the satellite responsible for processing the last subset of tiles captures the frame long after the first satellite captures the frame. This effect is magnified in an orbit-spaced configuration. The data show that a close-spaced configuration reduces latency over 617×.

7.3 Collected Energy Impacts Coverage

We use `cote-sim` to simulate CNP configurations described in Section 4.2 and evaluate OEC *ground track coverage*, i.e. the fraction of GTFs captured per revolution. To examine the impact of energy and computing on coverage, we vary energy buffer capacity, solar panel surface area, and computing hardware. We select a baseline design with a 5.0 F capacitor bank, 7.1 W surface-mounted solar panels, and one Jetson TX2 compute module. We compare this baseline to a CNP of devices with 10.0 F capacitor banks ("more capacitance"), 14.2 W of power due to a deployable panel ("more power"), and two Jetson TX2 modules ("more compute").

Aggregate energy collected across a CNP limits coverage. As device count increases, a pipeline achieves the minimum computing capability needed for full coverage before collecting sufficient aggregate energy to process all frames. Figure 8 plots coverage as a function of pipeline depth. Each plot lists four series: the baseline configuration, and the configurations with increased energy buffering, energy harvesting, and computing. Due to the energy-constrained nature of this design point, increasing solar panel surface area increases coverage at a faster rate than other parameters. While "more power" supports full coverage with a shorter pipeline, this configuration depends on a complex, mechanically-deployed solar array that increases system cost and risks a catastrophic deployer failure.

Under frame-parallel operation, individual satellites collect a GTF and process all tiles, which means that all GTFs are imaged exactly once across the CNP system. As a result, the number of times that any camera is activated across the

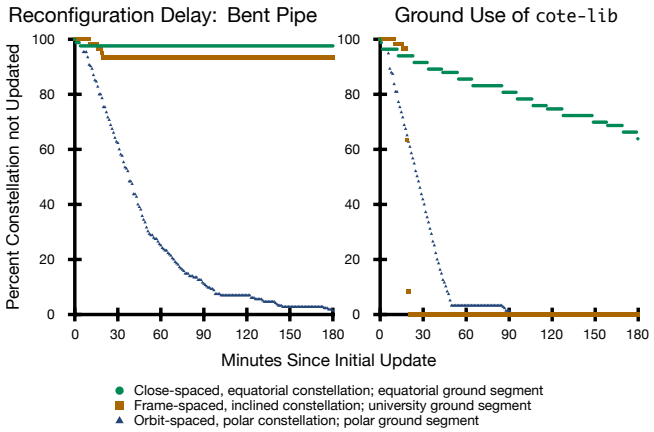

Figure 9. Constellation configuration takes hours or more with bent pipes. Ground use of `cote-lib` enables shorter configuration times for all constellation types, and use of `cote-lib` for OEC in space entirely eliminates configuration delays.

CNP system matches the number of ground track frames as coverage approaches 100%. In tile-parallel operation, every satellite images each GTF. Thus, the number of times that cameras are activated across the CNP system is equal to the pipeline depth times the number of GTFs. However, the overall energy effect of more frequent camera use at the system level is small because camera energy is four orders of magnitude less than compute energy. The effect of more frequent camera activation manifests as a lower slope in the tile-parallel graphs compared to the frame-parallel graphs as coverage approaches 100%. In Figure 8, the slopes in the tile-parallel configurations are less steep than the slopes in the frame-parallel configurations.

The data show that for object detection, full coverage of the ground track is feasible under multiple configurations, requiring around 100 satellites with deployable solar panels and around 250 satellites with surface-mounted panels. Existing nanosatellite constellations contain more than 200 devices, which means that OEC pipeline depths achieving full coverage are feasible. Increasing solar panel surface area dramatically reduces pipeline depth at the expense of greater engineering complexity (deployable panels) and higher per-device cost. Coverage degrades gracefully as pipeline depth decreases.

7.4 OEC Enables Online Autonomy

We use `cote-lib` to avoid the long *reconfiguration time* inherent to bent-pipe architectures. Uplink channels are much lower in bitrate than downlink channels. Maximum downlink channel bitrate increases with receiver gain; on Earth, receiver gain increases with dish diameter. Nanosatellites cannot increase receiver gain arbitrarily due to physical restrictions on device size. Thus, uplinked data volume may be limited to kilobytes per pass. Larger constellations experience longer reconfiguration times because satellites compete for uplink opportunities. Bent-pipe architectures require hours, days, or even months to reconfigure existing constellations.

Satellites with onboard positioning (e.g. GPS) and knowledge of ground station locations are aware of approaching link opportunities. However, under a bent-pipe architecture, remote-controlled satellites cannot predict whether a ground station will establish a link session. Knowledge of upcoming communication events allows satellites to intelligently choose between onboard processing and data transmission. `cote-lib` augments ground segments by modeling satellite states; we show in Figure 9 that `cote-lib` improves existing ground systems by enabling *link-schedule policies* that prioritize communication to nanosatellites with the largest amount of data to communicate, instead of heuristic policies that instead optimize for signal strength. The benefits of `cote-lib` are greatest when used in space for OEC. Rather than a ground station forgoing valuable downlink opportunities to upload link schedules, an OEC nanosatellite uses `cote-lib` to model the state of the entire constellation. With knowledge of the link-schedule policy, each nanosatellite generates upcoming communication events on orbit.

Figure 9 plots reconfiguration times with and without `cote-lib`. At left, ground stations operating under a bent-pipe architecture use a highest-elevation link-schedule policy to maximize signal quality. At right, `cote-lib` augments existing ground segments by enabling a more advanced link-schedule policy prioritizing satellites with the largest amount of buffered data. Ground use of `cote-lib` reduces reconfiguration times, but constellations of tens to hundreds of devices often require multiple revolutions before the entire system reconfigures. During this time, the constellation misses thousands of GTFs. `cote-lib` avoids lengthy reconfiguration times. By running continuously in the background on an OEC satellite, computing and communication decisions can be made autonomously on orbit.

8 Related Work

Several categories of work relate to orbital edge computing. Section 2 provides a brief space systems background. The NASA guide to the state-of-the-art in small satellites [9] describes characteristic aspects of nanosatellites, including technology readiness levels of essential subsystems. Recent surveys [5, 62] study multi-satellite orbital dynamics and provide an overview of propulsion systems. Commercial efforts demonstrate the viability of camera-equipped nanosatellite constellations [20, 74]. The SpaceNet challenge [94] illustrates broad interest in visual computing on space data, and the proposed Amazon ground station network [2] further cements the value of computing on visual and other space sensor data.

Recent edge computing work provides context for our work. Edge computing recognizes that, as high-datarate sensors (e.g. cameras, lidar) proliferate, streaming all data to central cloud systems for processing becomes infeasible [78, 79, 97]. Edge computing is important particularly in cases of complex processing, e.g. video querying [44], search [57], or DNN speech processing [55]. We propose to leverage early discard, which has been studied for search [45], video indexing [44], and drone video processing [97]. Recent work [8, 82] demonstrates the utility of simulation frameworks for edge computing on drones; `cote` is an analogous utility for the orbital edge. Machine inference accelerators [14, 15, 24, 33] could significantly shorten full-coverage CNP pipeline depths, although some that rely on temporal data redundancy [10] may have limited benefit for devices capturing images at the GTFR.

Intermittent computing [61] shares challenges with orbital edge computing in that both types of systems are fully energy-autonomous. A number of recent intermittent systems [16, 39, 54, 64–66] function despite unpredictable power failures using techniques that may be applicable to the orbital edge in future work. Some intermittent computing platforms [17] are similar in that they rely only on supercapacitors for energy storage. Other intermittent systems are similar in that they target DNN workloads [32] and communication minimization [63] at the edge in batteryless devices. While similar in spirit, these efforts differ significantly in scale, deployment environment, and in their inability to rely on processor sleep modes; instead, they power off frequently.

9 Conclusion & Future Work

In this work, we develop orbital edge computing: edge computing on orbit using processing resources colocated with sensors inside small, low-cost satellites. The low cost of a nanosatellite compared to monolithic satellite systems makes large satellite constellations feasible for the first time. Applications of this emerging technology are impeded by existing, bent-pipe architectures. Orbital edge computing provides responsiveness, reliability, and scalability benefits. Future work should study energy collection and storage for orbital edge computing and radiation-hardened, machine-learning (ML) accelerators.

For example, we observe that incomplete ground track coverage stems from the energy-constrained, intermittent nature of these nanosatellites. Once a constellation, in aggregate, collects sufficient energy for full coverage, high processing latency can still limit coverage. Increasing energy availability with deployable solar panels is unsatisfactory, because this solution raises mission cost and mission risk. As an alternative solution, future work could investigate energy-efficient, domain-specific accelerators (DSAs) for orbital edge computing workloads. Future work could evaluate the architectural vulnerability factors (AVFs) of recently-proposed ML accelerators and propose new ML accelerators that operate intermittently in the space environment.

While this work has focused on nanosatellite constellations that share a single orbit and a single workload, future work may investigate heterogeneous systems and heterogeneous workloads. For example, a constellation operator may wish to serve many different clients over time. Clients will be interested in different features at different scales. As a result, different orbit altitudes and different hardware will be better suited to different clients. Supporting a dynamic set of workloads from a dynamic set of clients poses an interesting challenge, especially with regards to constellation reconfiguration. The high overhead of uplinking new ML models could be offset with federated learning techniques.

Looking forward, we expect deployments of satellites that are even smaller than nanosatellites. Chip-scale or gram-scale satellites ("chipsats") can be deployed more numerously and at even lower cost. Such devices are even more attritable than nanosatellites, but the smaller size places even tighter constraints on capability. Successful operation of these emerging devices will require the application of orbital edge computing techniques.

Acknowledgements

We thank the anonymous reviewers for the helpful feedback that improved this paper. We thank members of CALCM and the Abstract research group at Carnegie Mellon University for useful feedback and discussions. We thank Zac Manchester for invaluable technical discussions on satellite actuation and control. This work was generously funded by the Kavčić-Moura Endownment Fund with support from National Science Foundation Award #1629196.

References

[1] Adcole Maryland Aerospace. MAI-400 1/2U CubeSat ADACS datasheet, 2018.
[2] Amazon Web Services. Amazon ground station. https://aws.amazon.com/ground-station/, 2018.
[3] Shinko Aoki, H Kinoshita, B Guinot, GH Kaplan, Dennis Dean McCarthy, and Paul Kenneth Seidelmann. The new definition of universal time. *Astronomy and Astrophysics*, 1982.
[4] AzurSpace. Triple Junctions GaAs Solar Cell Assembly datasheet, 2018.
[5] Saptarshi Bandyopadhyay, Rebecca Foust, Giri P Subramanian, Soon-Jo Chung, and Fred Y Hadaegh. Review of formation flying and constellation missions using nanosatellites. *Journal of Spacecraft and Rockets*, 2016.
[6] Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 2013.
[7] Kazimierz M Borkowski. Accurate algorithms to transform geocentric to geodetic coordinates. *Bulletin géodésique*, 1989.
[8] Behzad Boroujerdian, Hasan Genc, Srivatsan Krishnan, Wenzhi Cui, Aleksandra Faust, and Vijay Reddi. Mavbench: Micro aerial vehicle benchmarking. In *MICRO*. IEEE, 2018.
[9] Bruce Yost. State of the Art of Small Spacecraft Technology. https://sst-soa.arc.nasa.gov/, 2016.

[10] M. Buckler, P. Bedoukian, S. Jayasuriya, and A. Sampson. Eva2: Exploiting temporal redundancy in live computer vision. In *ISCA*. IEEE, 2018.

[11] Michel Capderou. *Satellites: Orbits and missions*. Springer Science & Business Media, 2006.

[12] Jeroen Cappaert. Building deploying and operating a cubesat constellation-exploring the less obvious reasons space is hard. In *Proc. AIAA/USU Conf. Small Satellites*, 2018.

[13] Kristen C Castonguay. Additive manufacture of propulsion systems in low earth orbit. Technical report, Air Command and Staff College, Air University, Maxwell AFB, 2018.

[14] Lukas Cavigelli and Luca Benini. Origami: A 803-gop/s/w convolutional network accelerator. *IEEE TCSVT*, 2017.

[15] Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *Solid-State Circuits*, 2017.

[16] Alexei Colin and Brandon Lucia. Chain: tasks and channels for reliable intermittent programs. In *ACM SIGPLAN Notices*. ACM, 2016.

[17] Alexei Colin, Emily Ruppel, and Brandon Lucia. A reconfigurable energy storage architecture for energy-harvesting devices. ACM, 2018.

[18] Howard D Curtis. *Orbital mechanics for engineering students*. Butterworth-Heinemann, 2013.

[19] Bradley Denby and Brandon Lucia. Orbital edge computing: Machine inference in space. *Computer Architecture Letters*, 2019.

[20] Kiruthika Devaraj, Ryan Kingsbury, Matt Ligon, Joseph Breu, Vivek Vittaldev, Bryan Klofas, Patrick Yeon, and Kyle Colton. Dove high speed downlink system. In *Proc. AIAA/USU Conf. Small Satellites*, 2017.

[21] DigitalGlobe. Worldview-3 data sheet. Technical report, 2017.

[22] Deanna Doan, Robert Zimmerman, Lawrence Leung, James Mason, Nate Parsons, and Kam Shahid. Commissioning the worldâĂŹs largest satellite constellation. In *Proc. AIAA/USU Conf. Small Satellites*, 2017.

[23] Lauren Dreyer. Latest developments on spacex's falcon 1 and falcon 9 launch vehicles and dragon spacecraft. In *2009 IEEE Aerospace conference*. IEEE, 2009.

[24] Zidong Du, Robert Fasthuber, Tianshi Chen, Paolo Ienne, Ling Li, Tao Luo, Xiaobing Feng, Yunji Chen, and Olivier Temam. Shidiannao: Shifting vision processing closer to the sensor. In *ACM SIGARCH*, 2015.

[25] EnduroSat. X-Band Transmitter datasheet, 2018.

[26] Dmitry Evtyushkin, Ryan Riley, Nael CSE Abu-Ghazaleh, Dmitry Ponomarev, et al. Branchscope: A new side-channel attack on directional branch predictor. In *ACM SIGPLAN Notices*. ACM, 2018.

[27] EXA. BA0X High Capacity Battery Arrays datasheet, 2018.

[28] Warren Ferster. Digitalglobe adding infrared capability to worldview-3 satellite. *Space News, https://spacenews.com/digitalglobe-adding-infrared-capability-worldview-3-satellite/*, 2012.

[29] Henry F Fliegel and Thomas C Van Flandern. Letters to the editor: a machine algorithm for processing calendar dates. *Communications of the ACM*, 1968.

[30] Warren L Flock. Propagation effects on satellite systems at frequencies below 10 ghz, a handbook for satellite systems design. 1983.

[31] Warren Frick and Carlos Niederstrasser. Small launch vehicles-a 2018 state of the industry survey. In *Proc. AIAA/USU Conf. Small Satellites*, 2018.

[32] Graham Gobieski, Brandon Lucia, and Nathan Beckmann. Intelligence beyond the edge: Inference on intermittent embedded systems. In *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2019.

[33] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: Efficient inference engine on compressed deep neural network. In *ISCA*. IEEE, 2016.

[34] Mark Harris. Tech giants race to build orbital internet [news]. *IEEE Spectrum*, 2018.

[35] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[36] Alexander Hellemans. Autonomous nanosatellites: Satellites that make up their mind [news]. *IEEE Spectrum*, 2016.

[37] Josiah Hester and Jacob Sorber. Flicker: Rapid prototyping for the batteryless internet-of-things. In *Conference on Embedded Network Sensor Systems*. ACM, 2017.

[38] Josiah Hester, Kevin Storer, and Jacob Sorber. Timely execution on intermittently powered batteryless sensors. In *Conference on Embedded Network Sensor Systems*. ACM, 2017.

[39] Matthew Hicks. Clank: Architectural support for intermittent computation. In *International Symposium on Computer Architecture (ISCA)*, 2017.

[40] SpaceX Space Exploration Holdings. FCC Fixed Satellite Service Filing SAT-LOA-20161115-00118. Federal Communication Commission, 2016.

[41] SpaceX Space Exploration Holdings. FCC Fixed Satellite Service Filing SAT-LOA-20170301-00027. Federal Communication Commission, 2017.

[42] SpaceX Space Exploration Holdings. FCC Fixed Satellite Service Filing SAT-LOA-20170726-00110. Federal Communication Commission, 2017.

[43] Felix R Hoots and Ronald L Roehrich. Models for propagation of norad element sets. Technical report, Aerospace Defense Command, Peterson AFB, Office of Astrodynamics, 1980.

[44] Kevin Hsieh, Ganesh Ananthanarayanan, Peter Bodik, Shivaram Venkataraman, Paramvir Bahl, Matthai Philipose, Phillip B Gibbons, and Onur Mutlu. Focus: Querying large video datasets with low latency and low cost. In *USENIX OSDI*, 2018.

[45] Larry Huston, Rahul Sukthankar, Rajiv Wickremesinghe, Mahadev Satyanarayanan, Gregory R Ganger, Erik Riedel, and Anastassia Ailamaki. Diamond: A storage architecture for early discard in interactive search. In *FAST*, 2004.

[46] Innovative Solutions in Space. ISIS Antenna Systems datasheet, 2018.

[47] Innovative Solutions in Space. ISIS Communication Systems datasheet, 2018.

[48] Innovative Solutions in Space. ISIS CubeSat Structures datasheet, 2018.

[49] Innovative Solutions in Space. ISIS On board computer datasheet, 2018.

[50] International Earth Rotation and Reference Systems Service. Iers bulletins. https://www.iers.org/IERS/EN/Publications/Bulletins/bulletins.html, 2019.

[51] Louis J Ippolito. *Radiowave propagation in satellite communications*. Springer Science & Business Media, 1986.

[52] IQ Spacecom. XLINK X-Band Transceiver datasheet, 2018.

[53] ISO ISO8601. Data elements and interchange formats–information interchange–representation of dates and times. *Geneva: International Organization for Standardization*, 2004.

[54] Hrishikesh Jayakumar, Arnab Raha, Woo Suk Lee, and Vijay Raghunathan. Quickrecall: A hw/sw approach for computing across power cycles in transiently powered computers. *Journal on Emerging Technologies in Computing Systems (JETC)*, 2015.

[55] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. In *ACM SIGARCH Computer Architecture News*. ACM, 2017.

[56] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. *arXiv preprint arXiv:1801.01203*, 2018.

[57] Emmanouil Koukoumidis, Dimitrios Lymberopoulos, Karin Strauss, Jie Liu, and Doug Burger. Pocket cloudlets. In *ACM SIGARCH Computer Architecture News*. ACM, 2011.

[58] Wiley J Larson and James Richard Wertz. *Space mission analysis and design*. Microcosm, 1992.

[59] Lawrence Leung, Vincent Beukelaers, Simone Chesi, Hyosang Yoon, Daniel Walker, and Joshua Egbert. Adcs at scale: Calibrating and monitoring the dove constellation. In *Proc. AIAA/USU Conf. Small Satellites*, 2018.

[60] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, et al. Meltdown: Reading kernel memory from user space. In *USENIX Security Symposium (USENIX Security 18)*, 2018.

[61] Brandon Lucia and Benjamin Ransford. A simpler, safer programming and execution model for intermittent systems. In *ACM SIGPLAN Notices*. ACM, 2015.

[62] Kim Luu, Maurice Martin, Alok Das, Andrew Peffer, Howard Schlossberg, Joe Mitola, Dave Weidow, Richard Blomquist, Mark Campbell, and Christopher Hall. Microsatellite and formation flying technologies on university nanosatellites. In *Space Technology Conference and Exposition*, 1999.

[63] Kaisheng Ma, Xueqing Li, Mahmut Taylan Kandemir, Jack Sampson, Vijaykrishnan Narayanan, Jinyang Li, Tongda Wu, Zhibo Wang, Yongpan Liu, and Yuan Xie. Neofog: Nonvolatility-exploiting optimizations for fog computing. In *ACM SIGPLAN Notices*, 2018.

[64] Kaisheng Ma, Xueqing Li, Jinyang Li, Yongpan Liu, Yuan Xie, Jack Sampson, Mahmut Taylan Kandemir, and Vijaykrishnan Narayanan. Incidental computing on iot nonvolatile processors. In *International Symposium on Microarchitecture (MICRO)*, 2017.

[65] Kiwan Maeng, Alexei Colin, and Brandon Lucia. Alpaca: intermittent execution without checkpoints. *Proceedings of the ACM on Programming Languages*, 2017.

[66] Kiwan Maeng and Brandon Lucia. Adaptive dynamic checkpointing for safe efficient intermittent computing. In *Symposium on Operating Systems Design and Implementation (OSDI)*, 2018.

[67] Henry Martin, Conor Brown, Tristan Prejean, Nathan Daniels, and LLC NanoRacks. Bolstering mission success: Lessons learned for small satellite developers adhering to manned spaceflight requirements. In *Proc. AIAA/USU Conf. Small Satellites*, 2018.

[68] Dennis D McCarthy and P Kenneth Seidelmann. *Time: from Earth rotation to atomic physics*. Cambridge University Press, 2018.

[69] Arash Mehrparvar, D Pignatelli, J Carnahan, R Munakat, W Lan, A Toorian, A Hutputanasin, and S Lee. Cubesat design specification rev. 13. Technical report, California Polytechnic State University, San Luis Obispo, 2014.

[70] Elizabeth M Middleton, Stephen G Ungar, Daniel J Mandl, Lawrence Ong, Stuart W Frye, Petya E Campbell, David R Landis, Joseph P Young, and Nathan H Pollack. The earth observing one (eo-1) satellite mission: Over a decade in space. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2013.

[71] National Geospatial-Intelligence Agency. Department of Defense World Geodetic System 1984 NGA Standard, NGA.STND.0036 1.0.0 WGS84. Technical report, 2014.

[72] Connor Nogales, Braden Grim, Mitch Kamstra, Benjamin Campbell, Aaron Ewing, Robert Hance, Joshua Griffin, and Stephen Parke. Makersat-0: 3d-printed polymer degradation first data from orbit. In *Proc. AIAA/USU Conf. Small Satellites*, 2018.

[73] ON Semiconductor. AR1335 CMOS Image Sensor Datasheet. https://www.mouser.com/datasheet/2/308/R1335-1143316.pdf, 2018.

[74] Planet Labs. Planet imagery product specifications. Technical report, 2018.

[75] Jordi Puig-Suari, Clark Turner, and William Ahlgren. Development of the standard cubesat deployer and a cubesat class picosatellite. In *Aerospace Conference, IEEE Proceedings*, 2001.

[76] OneWeb WorldVu Satellites. FCC Fixed Satellite Service Filing SAT-LOI-20160428-00041. Federal Communication Commission, 2016.

[77] OneWeb WorldVu Satellites. FCC Fixed Satellite Service Filing SAT-MOD-20180319-00022. Federal Communication Commission, 2018.

[78] M. Satyanarayanan et al. Edge analytics in the internet of things. *IEEE Pervasive Computing*, 2015.

[79] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 2017.

[80] P Kenneth Seidelmann. *Explanatory Supplement to the Astronomical Almanac, Completely Revised and Rewritten*. University Science Books, 1992.

[81] Thomas O Seppelin. The department of defense world geodetic system 1972. Technical report, World Geodetic System Committee Washington DC, 1974.

[82] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*. Springer, 2018.

[83] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE IoT*, 2016.

[84] Space Advisory Company. Chameleon Imager datasheet, 2018.

[85] Michael Swartwout. Cubesat database. https://sites.google.com/a/slu.edu/swartwout/home/cubesat-database, 2018.

[86] Amazon Kuiper Systems. ITU Satellite Network Filing USA2019-12905. International Telecommunications Union, 2019.

[87] Amazon Kuiper Systems. ITU Satellite Network Filing USA2019-12909. International Telecommunications Union, 2019.

[88] Amazon Kuiper Systems. ITU Satellite Network Filing USA2019-13020. International Telecommunications Union, 2019.

[89] Tactical Technology Office. Broad Agency Announcement: Blackjack Pit Boss, HR001119S0012. Technical report, DARPA, 2018.

[90] Andrew Tao, Jon Barker, and Sriya Sarathy. Detectnet: Deep neural network for object detection in digits. *Parallel Forall*, 4, 2016.

[91] Akshay Tummala and Atri Dutta. An overview of cube-satellite propulsion technologies and trends. 2017.

[92] David A Vallado and Wayne D McClain. *Fundamentals of astrodynamics and applications*. Microcosm Press, 2013.

[93] Joel Van Der Woude and Matthew Hicks. Intermittent computation without hardware support or programmer intervention. In *Symposium on Operating Systems Design and Implementation (OSDI)*, 2016.

[94] Adam Van Etten, Dave Lindenbaum, and Todd M Bacastow. Spacenet: A remote sensing dataset and challenge series. *arXiv*, 2018.

[95] Thaddeus Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey review*, 1975.

[96] Haibin Wang, Qingyu Chen, Li Chen, David M Hiemstra, and Valeri Kirischian. Single event upset characterization of the tegra k1 mobile processor using proton irradiation. In *Radiation Effects Data Workshop*. IEEE, 2017.

[97] Junjue Wang, Ziqiang Feng, Zhuo Chen, Shilpa George, Mihir Bala, Padmanabhan Pillai, Shao-Wen Yang, and Mahadev Satyanarayanan. Bandwidth-efficient live video analytics for drones via edge computing. In *SEC*. IEEE, 2018.

[98] Matthew Weinzierl and Angela Acocella. Blue origin, nasa, and new space (a). 2016.

[99] Dan White, Corey Shields, Pierros Papadeas, Agisilaos Zisimatos, Manolis Surligas, Matthaios Papamatthaiou, Dimitrios Papadeas, Eleytherios Kosmas, Vasileios Tsiligiannis, Alexandru Csete, et al. Overview of the satellite networked open ground stations (satnogs) project. In *Proc. AIAA/USU Conf. Small Satellites*, 2018.

[100] Kasım Sinan Yıldırım, Amjad Yousef Majid, Dimitris Patoukas, Koen Schaper, Przemyslaw Pawelczak, and Josiah Hester. Ink: Reactive kernel for tiny batteryless sensors. In *Conference on Embedded Networked Sensor Systems*. ACM, 2018.

A Artifact Appendix

A.1 Abstract

This appendix describes software artifacts associated with this work.

A.2 Artifact checklist

- **Compilation:** GCC 8.3.0 (setup script provided)
- **Data set:** Sample configuration files are included
- **Runtime environment:** Ubuntu 18.04 or similar
- **Hardware:** Simulations run concurrently as separate processes; 48 cores or more are recommended for timely results
- **Execution:** Managed by bash scripts
- **Metrics:** Percent data not downlinked per revolution; average system ground track frame latency; fraction of ground track processed per revolution (coverage)
- **Output:** Data plots
- **Experiments:**
  - Close-spaced downlink communication simulations
  - Frame-spaced downlink communication simulations
  - Orbit-spaced downlink communication simulations
  - Close-spaced, frame-parallel energy and computing simulations
  - Close-spaced, tile-parallel energy and computing simulations
  - Frame-spaced, frame-parallel energy and computing simulations
  - Frame-spaced, tile-parallel energy and computing simulations
- **How much disk space required (approximately)?:** 23 GB
- **How much time is needed to prepare workflow (approximately)?:** A few hours
- **How much time is needed to complete experiments (approximately)?:** About a week
- **Publicly available?:** https://github.com/CMUAbstract/oec-asplos20-artifact
- **Code licenses (if publicly available)?:** Apache License, Version 2.0

A.3 Description

A.3.1 How delivered. The Carnegie Mellon University ABSTRACT Research Group GitHub: https://github.com/CMUAbstract/oec-asplos20-artifact

A.3.2 Hardware dependencies. For timely results, 48 cores or more are recommended. A few gigabytes of RAM should be sufficient. Around 25 GB of empty hard drive space may be required.

A.3.3 Software dependencies. This artifact assumes execution with Ubuntu 18.04 or similar. GCC 8.3.0 is used for compilation. CMake is used for building Makefiles. Common command line tools (e.g. `git`, `make`, `wget`) are also assumed. For plot generation, Python 3 with `venv` is required.

A.3.4 Data sets. Sample configuration files are included. Up-to-date satellite TLEs can be accessed from http://celestrak.com/NORAD/elements/.

A.4 Installation

https://github.com/CMUAbstract/oec-asplos20-artifact/blob/1.0.0/README.md

A.5 Experiment workflow

https://github.com/CMUAbstract/oec-asplos20-artifact/blob/1.0.0/README.md

A.6 Evaluation and expected result

After completing the README, the user will have launched seven sets of experiments. The README describes how to use included scripts to parse the resulting log files. The user generates plots with the included plot generation scripts.

A.7 Experiment customization

Scenarios are implemented as C++ programs located in the `artifacts` directory. To customize a scenario, select a directory from `artifacts` and modify the `.cpp` file located in `source`.

The communication scenarios parse configuration files at runtime to set program behavior. Users can modify these configuration files in order to customize experiments without recompiling.