Article

Scheduling LEO Satellite Transmissions for Remote Water Level Monitoring

Garrett Kinman [1,†], Željko Žilić [†*] and David Purnell [†,‡]

[1] Octasic Inc., Montréal, QC, Canada
† McGill University, 3480 University, Montréal, QC, Canada
‡ Département de génie civil and génie des eaux, Laval University, Québec City, QC, Canada

Abstract: This paper deals with the use of low earth orbit (LEO) satellite links in long-term monitoring of water, ice and snow levels across wide geographic areas, where no other communication links can be applied. Unlike geostationary satellites, LEO satellites do not maintain the same position relative to the ground station, and transmissions need to be scheduled for satellite overfly periods. For our application, the energy consumption optimization is critical, and we develop a learning approach for scheduling the transmission times from the sensors. Our approach is based on online learning, and is applicable to any LEO satellite transmissions.

Keywords: Internet of Remote Things; LEO satellite transmission; water-level monitoring.

1. Introduction

The Internet of Things (IoT) has made huge advances in smart homes, industrial and other settings with numerous networking options already present. To achieve progress in the Internet of Remote Things (IoRT) for environmental monitoring in wilderness, connectivity solutions are needed that are widespread, energy-efficient and cost-efficient. This paper presents the exploration of satellite-based connectivity in the context of environmental water-level monitoring.

1.1. Water-Level Monitoring and its Role in Climate

Global water-level monitoring is critical in hydrology and climate change tracking. The polar regions are arguably at the center of the climate crisis, because these regions are experiencing the most rapid changes and the largest current and future contribution to sea level rise is predicted to be from ice sheets losing mass to the ocean [1]. Predicting how the polar regions will change in the future requires field measurements, for example from sensors that monitor changes in the atmosphere (weather stations), coastal water-level sensors, ocean moorings, or from Global Navigation Satellite System (GNSS) stations (for monitoring solid earth deformation) [1]. Despite the ever-expanding capabilities of remote sensing satellites, such measurements cannot yet be obtained from space with the same accuracy or temporal resolution as from ground-based sensors.

Climate model predictions become more reliable with an increased density of sensors, hence low-cost environmental sensor networks are emerging as a powerful tool for climate monitoring [2]. One recent innovation repurposes mass-market GNSS technology for water-level monitoring, using a technique called GNSS Interferometric Reflectometry (GNSS-IR), and has the potential to be used to increase the density of coastal water-level stations [3,4]. In remote regions such as Greenland or Antarctica, where sea level information is critical for climate monitoring [1], field campaigns are expensive and it may be prohibitively expensive to maintain a dense network of sensors. Wireless connectivity should reduce the maintenance cost of remote sensor networks by reducing the frequency of expensive site visits to collect data or check the status of instruments. This paper focuses on a low-cost and

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2023 doi:10.20944/preprints202305.0527.v1

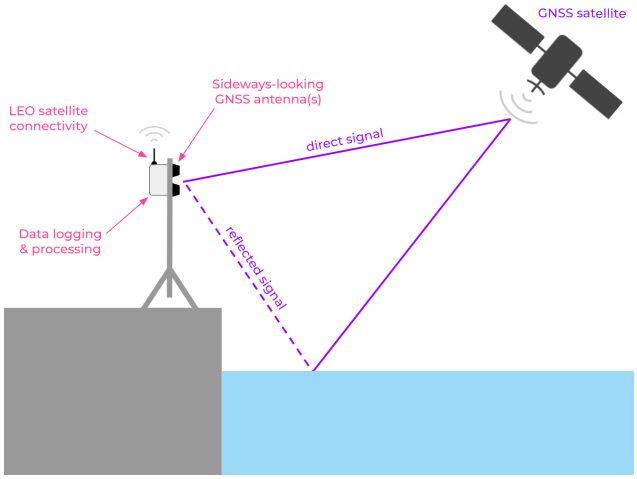

Figure 1. Schematic showing a GNSS-IR water-level sensor with integrated LEO satellite connectivity.

energy-efficient wireless communication technique using low earth orbit (LEO) satellites that is suitable for remote water-level sensor networks, notably GNSS-IR ones, Figure 1.

1.2. Connectivity for Internet of Remote Things

This paper addresses the problem of providing inexpensive and energy-efficient satellite IoT links in the context of GNSS-IR monitoring. Such a water-level sensor must be affordable and widely deployable. Wide geographic reach imposes the challenge of data uplink from remote locations to where that data is needed [5]. This limitation of IoT has spawned a subdomain dedicated to solving the issues of bringing IoT to the remote corners of the globe, the IoRT [6].

There is an abundance of connectivity options for IoT around populated areas. Several standard IoT connectivity options range from cellular technologies to LoRa. Connectivity options for IoRT range from low-power wide area networks (LPWANs) to low-power cellular network standards to geostationary and LEO satellites [7]. Additionally, there have been efforts into unmanned aerial vehicles supporting IoRT [7,8]. Satellite options are the only proven connectivity options for truly global coverage [7,9,10], but only if the cost and energy consumption are kept low enough. Traditional geostationary satellites are always overhead for a fixed earth location, but they are costly, require higher transmission power, and incur around 70 times longer latency than LEO satellites [6,11].

1.3. LEO Satellite Communications for Internet of Remote Things

For IoT applications, LEO satellites are practical for the most remote regions where terrestrial infrastructure is out of reach [6,11]. LEO communications are categorized by:

- communication directness,
- LEO orbit configuration, and
- by satellite service type.

Regarding communication directness, individual devices or sensors can communicate directly to a satellite (known fittingly as "direct-to-satellite") or indirectly via a local network (often an LPWAN) centered around a satellite gateway [7,9].

Among satellite services, there are those provided by companies from the pre-IoT era (who often offer satellite internet and phone coverage as well), and there are those provided by independent LEO satellite companies [7,9], which are more suited for IoT. The independent LEO satellite services use CubeSats, which are small and modular picosatellites [7]. Since IoT can tolerate intermittent connectivity better than satellite phones, their satellites can use polar orbits to provide global, but intermittent, coverage [9]. In contrast, traditional LEO providers deliver continuous or near-continuous coverage using a combination of polar and non-polar orbits [9].

 doi:10.20944/preprints202305.0527.v1

Using intermittent LEO constellations for IoRT has the primary benefit of requiring fewer satellites and less cost [9], but requires data buffering until satellite passes overhead [9]. To achieve low-energy LEO networking, a suitable algorithm must be devised to schedule sensing and transmissions at appropriate times such that the data is transmitted at (near-)minimal cost in energy [9].

1.4. Relation to Previous Work

To the best our knowledge, one previous paper [10] has examined this problem and proposed an online learning algorithm, which can learn sample-by-sample in the field, as opposed to offline in batched datasets. This work examines however indirect-to-satellite communications, where it is unknown when new data will be received by the gateway, and thus the authors pose it as a queue scheduling problem [10]. They then propose an online learning algorithm based on Lyupanov optimization, which is a common approach for similar problems [10,12].

In contrast, our paper deals with direct-to-satellite communications with a known data production rate. The algorithm presented in this paper is derived from reinforcement learning, specifically Monte Carlo learning and the k-armed bandit problem. Because of the relative youth of the LEO satellite service (provided commercially by Swarm Technologies), an integrative approach is taken to the design from requirements, to communications technology selection, to hardware, and finally to software and algorithm design. In doing this, this paper aims to highlight key design considerations for creating a low-cost, low-power communications scheme for an IoRT device. The key contribution of this paper is the online learning-based direct-to-satellite scheduling, and associated energy model.

2. Materials and Methods

2.1. Satellite Modem Operating Specifications

For remote GNSS-IR sensors, we use the independent LEO satellite provider Swarm Technologies. The scheduling algorithm will depend on the characteristics of Swarm's service and its M138 modem. The Swarm modem has four operating states: 1) Sleep Mode, 2) GPS Acquisition Mode, 3) Receive Mode, and 4) Transmit Mode, Figure 2.

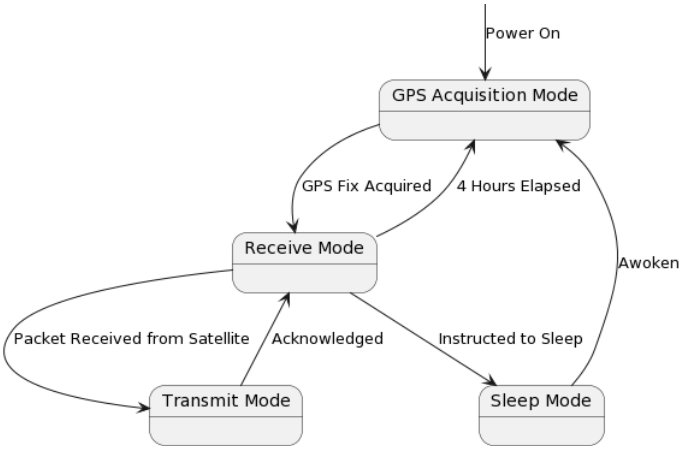

Figure 2. State machine of the Swarm M138 operating modes

When the modem powers on, it enters GPS acquisition mode to determine the time and location. The modem will also re-enter GPS Acquisition Mode every 4 hours or when awoken. Once a GPS fix has been acquired (30 s typical duration), the modem enters the Receive Mode, wherein it listens for a packet from any satellites passing overhead. This mode lasts until either a packet is received from a satellite (at which point it enters Transmit Mode), the modem is instructed to enter Sleep Mode, or enough time elapses that the modem automatically re-enters GPS Acquisition Mode.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2023 doi:10.20944/preprints202305.0527.v1

Mode	**Typical Current**	**Typical Power**
Transmit	$550mA$	$2.8W$
GPS Acquisition	$45mA$	$230mW$
Receive	$26mA$	$130mW$
Sleep	$<110\mu A$	$<550\mu W$

Table 1. DC power characteristics in the four modes of operation for 5V power supply.

Background Noise RSSI (dBm)	**Quality (for Transmission)**
-90 and higher	Bad (unlikely to work)
-93 and lower	Marginal
-97 and lower	OK
-100 and lower	Good
-105 and lower	Great

Table 2. Background noise intensity required for likelihood of transmission.

If a packet is received from a satellite, the modem enters Transmit Mode, attempting to transmit queued packets and receive an acknowledgement. If successful, it will return to Receive Mode, unless put into Sleep Mode. Sleep Mode uses 2 to 3 orders of magnitude less energy, as per Table 1. Swarm reports that a maximum-length 192-byte packet takes $\Delta T = 3.7$s and $E_{total} = 12.24$J.0

For Swarm modem's operating modes, the energy-saving strategy includes:

1. Keep the modem in Sleep Mode as much as possible. When not in Sleep Mode, its default state is Receive Mode, which uses much more power.
2. Being awake dominates energy usage, either from the actual transmission energy or the GPS Acquisition and Receive Modes.
3. Failure to transmit will waste considerable energy. Thus, one should schedule transmission to when there is a high probability of successful communication.

2.2. Swarm Satellite Transmission

To minimize transmission power consumption requires understanding how the satellites, transmission, and data plans work. There is not always a satellite overhead, nor are the elevation angle and environmental conditions (e.g., background RF noise) always suitable. Data rate is limited, and frequent transmissions consume energy. These factors critically impact how we orchestrate transmissions.

The nature of Swarm satellite passages is disclosed by their Web-based tool that lists upcoming satellite passes, their times, durations and max elevation angles for a given location. Elevation angles observed in Montreal, Quebec, Canada range between 15 and 85 degrees, and pass durations typically range between 10 minutes and an hour. In reality, even with a satellite pass, the modem might not always be able to transmit. There are many factors impacting this: satellite pass "quality", RF background noise, environmental conditions, antenna setup, and many others.

The first factor, satellite pass quality, is due to the pass duration and maximum elevation angle. Swarm gives no guidance on what factors impair successful transmission, and one objective of this paper is for each sensor to construct an empirical model for quantifying the likelihood of a pass leading to successful transmission. The second factor, RF background noise, does have guidance provided by Swarm, Table 2, by which noise intensity of -93 dBm or lower is expected for successful transmission.

There are also the constraints imposed by Swarm data plans, priced at USD 60 per year per data plan, with up to four data plans stackable onto a single modem. Each data plan permits up to 750 packets per month, or about 25 packets per day, or about one per hour. These constraints imply that for finer temporal resolution (e.g., every 15 minutes), one must either bundle measurements, or pay to stack multiple data plans. The later, costly

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2023 doi:10.20944/preprints202305.0527.v1

option also reduces the battery life, while bundling reduces the number of packets and possibly the cost.

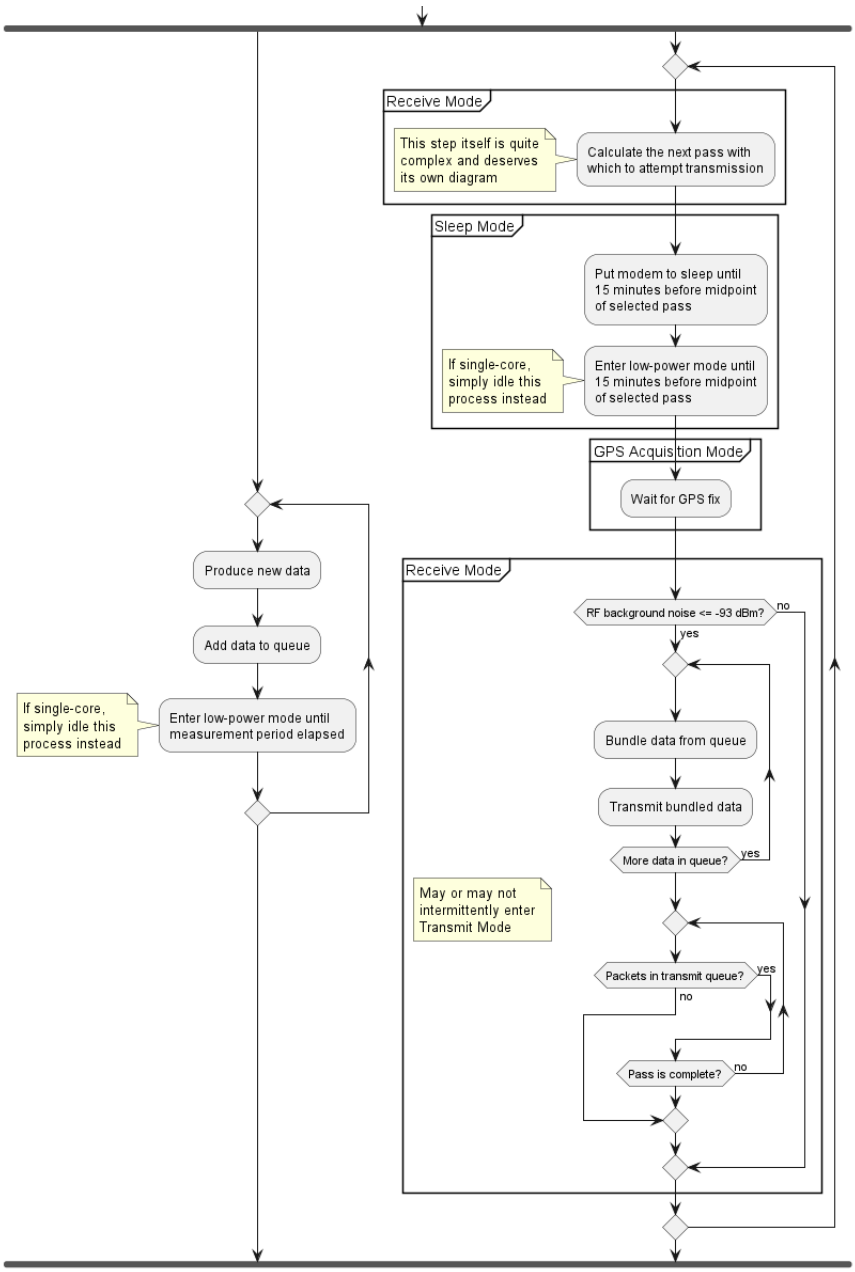

Figure 3. Activity diagram for the host device with two processes

The high-level view of the two main processes is shown in Figure 3. The process on the left produces and inserts the data into a circular queue. Since the Swarm modem's internal queue can drop packets after 48 hours, the circular queue needs to contain 48 hours of data. Each cycle of waking from sleep, acquiring GPS, listening for a satellite, and transmitting uses a lot of extra energy. In addition, due to environmental variables, there is inherent uncertainty as to how long one can expect the modem to be awake before transmitting successfully. This precise question is examined in the rest of the paper.

2.3. Efficient Packet Data Bundling

Note that there are a few important functions in Figure 3, such as the data bundling, as the transmission duration directly causes energy consumption. Table 3 shows the format of data. Each datum includes a timestamp, expressed in minutes since January 1st, 1970.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2023 doi:10.20944/preprints202305.0527.v1

Name	Type	Bits
Water level	Floating-point	32 bits (4 bytes)
Error	Floating-point	32 bits (4 bytes)
Roughness	Floating-point	32 bits (4 bytes)
Minutes since Jan. 1st, 1970	Positive integer	28 bits (<4 bytes)
Status	Positive integer	4 bits (<1 byte)
Total		**128 bits (16 bytes)**

Table 3. Format for each datum within the software.

Due to the nature of the GNSS-IR, we omit seconds, which allows the re-purposing of 4 bits for 16 status codes. For completeness, using 28 bits allows timekeeping as:

$$28 \text{ bits} -> 2^{28} - 1 \text{ minutes} = 510 \text{ years}$$

This format allows the whole datum to fit in 16 bytes, which divides evenly into 192 bytes per packet, such that each packet will be maximally utilized with 12 data points per packet.

Second implicit function within the high-level processes shown in Figure 3 is that of good satellite pass selection. While the algorithm for actually predicting satellite passes—at least from the user perspective—is made fairly simple with the help of an open-source SGP4 satellite pass prediction Arduino library, quantifying what satellite passes are "good" depends much on environmental conditions, setup details, and empirical observations, as described next.

2.4. Online Learning Direct-to-Satellite Packet Scheduling

Transmitting to the LEO satellites can be unreliable due to minute changes in equipment setup or environmental factors. For example, severely cloudy days lead to too high RF background noise (i.e., higher than -93 dBm). Further, unshielded microcontroller within 10 to 20 cm of the antenna could increase measured RF background noise by as much as 5 to 10 dB. Further, slightly angling the antenna towards or away from a cell tower a few kilometers away could vary the RF background noise by several dB. With all these factors, creating a generalizable pass model is intractable.

Previous work with indirect-to-satellite scheduling shows that online learning is a successful strategy [10]. Thus, each individual sensor should learn for itself and for its exact site conditions and hardware setup via online learning. Previous work in indirect-to-satellite scheduling uses a Lyupanov optimization problem for network queuing to avoid making assumptions about when new data would become available [10]. However, in this direct-to-satellite application in which the production rate at which new data is presented, it is not necessary to treat it as an optimization of network queuing problem. Thus, a novel approach will be used.

2.4.1. Algorithmic Problem Statement

For a sensor placed in a remote location, a simple and interpretable model is preferable [13]. To achieve this, a relatively simple algorithm inspired from reinforcement learning has been devised. The goal is to learn the probability of successful transmission, given three input variables: 1) the satellite pass duration (in minutes), 2) the maximum elevation angle of the satellite pass (in degrees), and 3) the RF background noise (in dBm).

Borrowing the notation from reinforcement learning, the state space S is the set of all possible input variable combinations, and the action space A is the set of all possible actions [14]. In this case, A consists of the actions to transmit or not to transmit for each satellite pass with pass characteristics $s \in S$. Let the function v be the mapping of S to a probability of successful transmission, $v : S \mapsto [0, 1]$, and let the policy π represent the conditional probability of choosing a particular action $a \in A$ given a state $s \in S$. Hence, the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2023 doi:10.20944/preprints202305.0527.v1

Bucket Number	Max Elevation Angle (°)	Pass Duration (minutes)	RF Background Noise (dBm)
1	15 to 30	10 to 20	-93 to -95
2	31 to 45	21 to 30	-96 to -98
3	46 to 60	31 to 40	-99 to -101
4	61 to 75	41 to 50	-102 to -104
5	76 to 90	51 and higher	-105 and lower

Table 4. State space bucketing for each state variable.

policy is the mechanism for choosing which satellite pass to select, given a set of passes and their characteristics.

$$\pi(a|s) = P(A_t = a|S_t = s) \quad (1)$$

In Equation 1, A_t represents the action at time step t, and S_t represents the state at time step t. Regarding the probability success mapping V, a natural objective is thus to approximate it with collected experience: as the system runs and has successes and failures transmitting with different states $s \in S$, it will converge to true probabilities of successful transmission for a given state, i.e., the value function v [14,15].

2.4.2. Modified Monte Carlo Learning

Monte Carlo learning methods approximate a value function in a simple and interpretable way by taking the value of a state to be the average return at the end of a training episode [14,15]. In the direct-to-satellite packet scheduling, the episodes are of length one, i.e., there is no sequential decision-making, simplifying the problem. If the reward is taken to be 1 for a successful transmission and 0 for an unsuccessful transmission, then the value function can be taken to be the average rate of successful transmission from a given state. If for a given state of satellite pass characteristics and RF noise, transmission is successful 50% of the time, then the value function is 0.5.

However, Monte Carlo learning requires a discrete state space, whereas the state space for this problem is continuous, so we discretize the state space. Using the Swarm pass checker, it is known that all satellite passes shown are between 15 and 90 degrees and almost all between 10 and 60 minutes. Additionally, while RF noise is technically continuous, the modems only report whole numbers, e.g., -95 dBm. If only integers within the range -93 dBm (the highest noise Swarm reports success transmitting with) to -106 dBm (the lowest noise measured in this project) are considered, this is naturally discretized. Table 4 shows how the state space has been chosen to be discretized. With 5 buckets for each state variable, some simple combinatorics gives 125 unique combinations, where the total set of 125 combinations represents the discretized state space.

The remaining question is that of the policy π. Clearly, once a good approximation of the true value function is made, the policy π should exploit that knowledge to select the most promising satellite passes. At the beginning, the system will not know about a good satellite pass, and it will thus have to explore with passes of different characteristics. This is an example of the *exploration–exploitation* problem in reinforcement learning [14,15]. A common approach is to explore early on and gradually exploit more with time.

2.4.3. Modified k-Armed Bandit

Regarding the policy for packet scheduling, there is a similarity to the k-armed bandit problem, whereby an agent repeatedly plays the same one-step episode. In each game, the agent has a selection of options, which may give varying stochastic rewards. The goal is to learn over time which actions give the greatest expected reward [14,15]. A common approach to this problem involves softmax (Boltzmann) exploration, which derives a set of probabilities corresponding to each possible action [16]. The action with the highest expected reward has the highest probability of selection, plus all the choices are guaranteed to sum to 1 by the design.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2023 doi:10.20944/preprints202305.0527.v1

Our problem is slightly different from the k-armed bandit problem in two important ways: 1) the set of actions available to the agent in each episode is different, and 2) expected reward is not only the probability of successful transmission, but its utility in the given application, most notably the timeliness. Regarding the first point, the agent is faced with a different selection of satellite passes each episode, each with their own set of pass characteristics and times at which they occur. This problem is solvable, as the modified Monte Carlo learning methods will allow keeping track of the estimated reward of each action.

2.4.4. Temporal Bounds for Packet Scheduling

Addressing the reward modeling, we apply the following reasoning. A good pass in an hour is not the same as an equally good pass occurring after 24 hours because: 1) data needs to be transmitted regularly, 2) the circular queue holding data has a finite capacity, and 3) the Swarm modem will drop packets from its transmission queue after a timeout. Thus, to model he preference for more prompt transmissions, a *discount factor* λ is applied to reduce the value of later passes.

Under the data plan, each modem can transmit at most one packet per hour to remain within the budget. Since there are many satellite passes to consider, the rules are needed for the interval of packet scheduling. Such rules are shown in Figure 4, in which t_{min} and t_{max} are the minimum and maximum amount of time (in hours) for a satellite pass, respectively, **a** is a vector of satellite passes between t_{min} and t_{max} (a_i is the i-th element of **a**). Further, **s** is a vector of states (i.e., pass characteristics and RF background noise) of satellite passes of **a**. Let **t** be a vector of midpoint times of satellite passes of **a**.

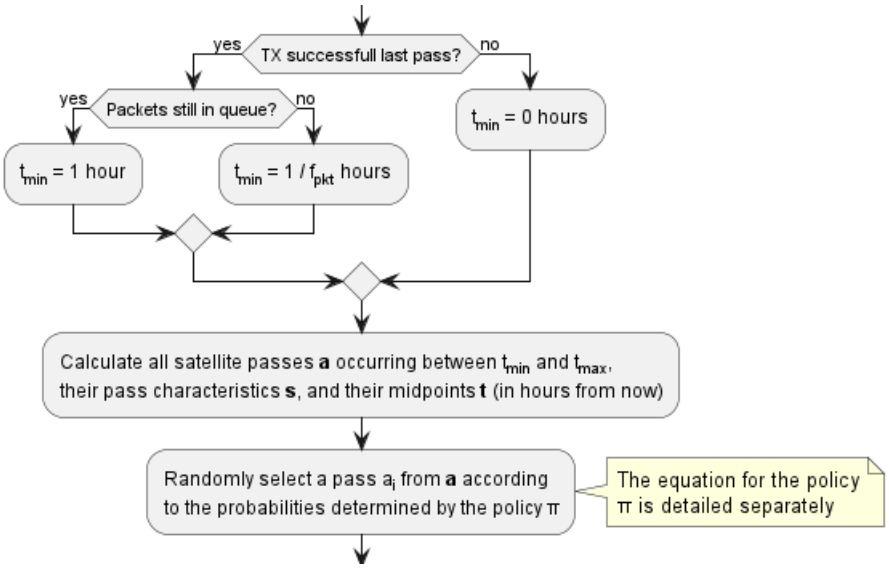

Figure 4. Routine for selecting the next satellite pass to attempt transmission.

2.4.5. Algorithmic Formulation

Since we are developing a learning approach to the LEO transmission scheduling, we will rely on the activation function for classification/learning, softmax. For the set of values $x_j, \{j, 1, N\}$, it is for each value x_i from as:

$$softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j))}.$$

Since softmax adds up to 1 across all inputs, it effectively creates a probability distribution function that disproportionally favors larger values of x_i.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2023 doi:10.20944/preprints202305.0527.v1

Let r_{data} be the data point generation rate (in data points per hour), and *bundlesize* be the number of data points that comprise a full bundle. Then, the rate of full packet bundling r_{pkt} is: $r_{pkt} = \frac{r_{data}}{bundlesize}$. Let softmax($\mathbf{z}$) be the vectorized softmax function where softmax($\mathbf{z}$)$_i$ is the softmax of the i-th element of $\mathbf{z}$, and let $v(\mathbf{s})$ the vectorized value function. We express the policy π as:

$$\pi(a_i|s_i) = \text{softmax}(\lambda^{\mathbf{t} \ominus t_{min}} \odot v(\mathbf{s}))_i \tag{2}$$

where the $\ominus$ and $\odot$ symbols operating on vectors $\mathbf{t}$ and $\mathbf{s}$ are the element-wise subtraction and multiplication, respectively. Equation 2 expresses the probability of selecting a satellite pass a_i from interval $[t_{min}, t_{max}]$ as the softmax of the estimated transmission success probability for the pass, multiplied by a discount factor for future passes. Pass quality and promptness will be prioritized, while still giving a chance for exploration of passes currently predicted to be worse. This preference allows Monte Carlo learning to improve the value function estimates with time.

2.5. Uplink Transmission Energy Model

To determine the average power consumption, we introduce the unified uplink transmission energy model. Since the stochastic nature of transmission success prohibits the derivation of a deterministic model, an model is created to give an estimate of average power consumption. There are two key causes of transmission non-determinism: 1) whether a transmission will succeed for a given pass, and 2) if it does succeed, how long the modem will be in Receive Mode before it is able to transmit.

To build the model, let t_{SL} be the mean time that the modem is in Sleep Mode, t_{GPS} be the mean time the modem is in GPS Acquisition Mode, and t_{RX} be the mean time the modem is in Receive Mode before transmission is successful. With typical modem power consumption values P_{SL}, P_{GPS}, and P_{RX}, the total energy usage in these modes over a single transmit attempt cycle, $E_{attempt}$ is:

$$E_{attempt} = P_{SL}t_{SL} + P_{GPS}t_{GPS} + P_{RX}t_{RX} + E_{TX}N_{pkt} \tag{3}$$

where N_{pkt} is the number of packets transmitted in a given pass. Depending on satellite pass selection and/or previous transmission attempt successes, N_{pkt} may be 1 or larger. In the case of an unsuccessful attempt, N_{pkt} is 0. An expression for non-zero N_{pkt} is:

$$N_{pkt} = \frac{r_{pkt}}{p_{success}r_{attempt}} \tag{4}$$

where $p_{success}$ is the transmission success probability, $r_{attempt}$ is the mean transmission attempt rate, and r_{pkt} is the rate at which fully bundled packets are generated. Since $r_{attempt}$ is smaller or equal to r_{pkt}, N_{pkt} is guaranteed to be 1 or greater because successful transmission of one packet entails successful transmission of all queued packets.

In Equation 3, also note that, while P_{SL}, P_{GPS}, t_{GPS}, P_{RX}, and E_{TX} (at least for full packets) are constant, t_{SL} and t_{RX} are variable. Here, t_{SL} represents the mean time in the Sleep Mode before making a transmission attempt, approximated as: $t_{SL} = \frac{1}{r_{attempt}}$.

The value of t_{RX} depends on how long the modem waits until it receives a packet and begins the transmission, or the pass is over. For a successful transmission, the quickest case is to transmit immediately after exiting GPS Acquisition Mode. The slowest success case is to transmit at the very end of the satellite pass. The worst failure case is the modem reaching the end of a given satellite pass in Receive Mode, with no transmission. In terms of t_{RX}, this case and successful transmission at the very end of the pass would be approximately equal. All three cases depend on the mean pass duration, denoted as t_{pass}.

$E_{attempt}$ can take two forms, depending on the transmission attempt success. A success is expressed in Equation 5, where ϵ_{pass} represents the proportion of a satellite pass spent in Receive Mode before receiving a packet from the satellite and is able to transmit. For

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2023 doi:10.20944/preprints202305.0527.v1

pessimistic and optimistic models, ϵ_{pass} can be treated as either 1 or 0, as these serve as the upper and lower bounds of the time in Receive Mode for a given satellite pass.

$$E_{success} = P_{SL}\frac{1}{r_{attempt}} + P_{GPS}t_{GPS} + \epsilon_{pass}P_{RX}t_{pass} + E_{TX}\frac{r_{pkt}}{p_{success}r_{attempt}} \tag{5}$$

If the attempt is a failure, the model is represented by Equation 6. Note that there is no ϵ_{pass} value and no N_{pkt}, as the system will wait out a full pass without transmissions.

$$E_{fail} = P_{SL}\frac{1}{r_{attempt}} + P_{GPS}t_{GPS} + P_{RX}t_{pass} \tag{6}$$

Since the relative probabilities of transmission successes and failure are related by $p_{success}$, the above two cases can be combined for a more complete model.

$$E_{attempt} = p_{success}E_{success} + (1 - p_{success})E_{fail} \tag{7}$$

which expands into the following expression:

$$E_{attempt} = \frac{P_{SL}}{r_{attempt}} + P_{GPS}t_{GPS} + p_{success}(\epsilon_{pass}P_{RX}t_{pass} + \frac{E_{TX}r_{pkt}}{p_{success}r_{attempt}}) + \\ +(1 - p_{success})P_{RX}t_{pass} \tag{8}$$

where P_{SL}, P_{GPS}, and P_{RX} are all constants and given by Swarm. Similarly, location fix time t_{GPS} is rather constant, reported to be about 30 seconds by Swarm. Also note that $r_{attempt}$ depends on site conditions, project requirements, and packet scheduling. Similarly, $p_{success}$ and t_{pass} depend heavily on site conditions and packet scheduling.

2.6. Simulation Model of Online Learning Algorithm

Setting up a number of sensors in the representative environment is expensive in time and money. Instead, the algorithm is tested with a simulated environment, similar to the methodology chosen in previous work on indirect-to-satellite scheduling [10]. Using simulations first can demonstrate the ability of the algorithm to learn underlying unknown patterns about satellite pass qualityand tune the discount factor λ parameter. To simulate the algorithm, two key components are needed: 1) virtual transmitters with an underlying probability model for which pass qualities are likely to result in transmission, and 2) randomly generated satellite pass characteristics and RF noise data. For virtual transmitters, three conceptual preference models were created, Table 5, to see how different transmitting obstacles would affect the algorithm. Note that, in Table 5, the preferences refer to the conditions required for a high likelihood of success. For example, the first preference model requires high angles, long durations, and low noise in order to have a high likelihood of success.

Model	Elevation Angle	Pass Duration	RF Background Noise
1	High angles	Long time	Low noise
2	Mid to high angles	Mid to long time	Low to mid noise
3	Low to high angles	Short to long time	Low to high noise

Table 5. Conceptual preference models for the virtual transmitters.

To create above virtual transmitter models, a function is constructed that outputs a transmission success probability for the three variables above, by multiplying three stretched-and-shifted sigmoid curves, one for each variable preference. For example, the sigmoid to represent a preference for high angles would produce a value close to 1 for high

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2023 doi:10.20944/preprints202305.0527.v1

angles (e.g., 70 degrees and higher) but a value close to 0 for low angles (e.g., 30 degrees and lower). The general form of the preference models is shown in Equation 9.

$$P(success) = \sigma(k_\theta(\theta - \theta_0)) \times \sigma(k_d(d - d_0)) \times \sigma(k_\gamma(\gamma - \gamma_0)) \quad (9)$$

where $\sigma(x)$ represents the sigmoid function, θ represents the max elevation angle, d represents the pass duration, and γ represents the RF background noise. Note that k_θ, k_d, k_γ, θ_0, d_0, and γ_0 represent configurable stretching and shifting constants to represent the different conceptual preference models. The values of these constants used to create the three preference models are shown in Table 6.

Preference Model	k_θ	θ_0	k_d	d_0	k_γ	γ_0
1	0.5	70	0.5	35	-1	-102
2	0.5	50	0.5	20	-1	-99
3	0.5	30	0.5	10	-1	-96

Table 6. Constants for the three preference models.

Simulated satellite passes are presented to virtual transmitters by agents imbued with a preference model and a value function approximator. The generated pass characteristics are randomly generated: each agent is exposed to random RF background noise, a vector **a** of satellite passes with corresponding random midpoint times **t**, and random pass characteristics **s** (except each $s_i \in \mathbf{s}$ also includes the RF noise value). The randomly generated pass characteristics are drawn from a uniform distribution, and the RF background noise values are drawn from two differing distributions for two different experiments:

1. Uniform across all buckets (-107 to -93 dBm).
2. Uniform within one bucket (-107 to -105 dBm).

It is possible that a given sensor may experience the full range of possible RF noise intensities, but it is also possible a given sensor in a remote location will only often see a narrow sub-range of possible RF noise intensities. Thus, these three noise configurations were selected to observe how the algorithm performs under these differing RF noise distributions.

Each agent then calculates the probabilities of selecting each satellite pass from the discretized states, the agents' value function approximators, and the pass midpoints. These probabilities are calculated according to the policy π. Then, satellite passes are chosen by the calculated probabilities. For chosen passes, the satellite pass characteristics and the RF background noise values are fed into the agents' respective preference models to produce probabilities of transmission success. Finally, transmission successes are determined according to the agent preference model outputs. The agent value function approximators are updated, and the process repeats.

3. Experimental Results

For evaluation of the transmission scheduling, the goal is to 1) demonstrate the ability of the algorithm to learn patterns behind transmission success probabilities, and 2) to make apparent how the performance gets affected by the main variables, such as the overpass duration, angle and the RF noise.

Simulations were conducted for the proposed algorithm under all three preference models in Table 5. The power data published by Swarm, as per Table 1 forms the basis of the transmitter model. The effects of random noise were expressed in two ways: noise contained in a single bucket, or across all buckets from Table 4.

Figures 5 to 10 show the transmission success for three preference models as a function of the epochs (number of runs), and parameterized by the discount factor λ. Most notable is the response to overall transmission difficulty. When it is relatively hard or relatively easy to transmit, there is less room for the algorithm to make huge improvements in success rate. For example, when there are few good passes, the algorithm often has to make a

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2023 doi:10.20944/preprints202305.0527.v1

choice between a mediocre and a bad pass. This is seen in the lower (but still significant) improvement for the first preference model.

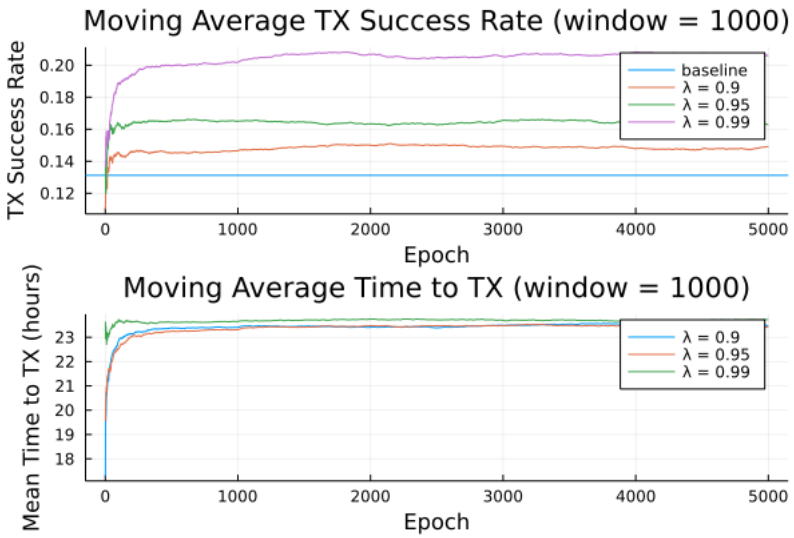

Figure 5. Simulated results for preference model 1 and random noise within 1 bucket.

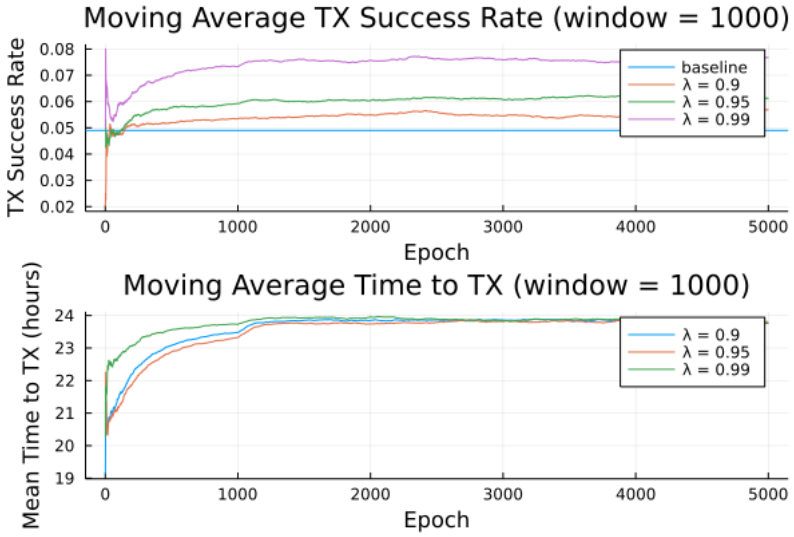

Figure 6. Simulated results for preference model 1 and random noise across all buckets.

Second to note is the difference in response to discount factors. In the first preference model, less likely to succeed, differing discount factors made little difference to $r_{attempt}$, as the penalty for long wait for a decent pass outweigh the cost of other poorer options. For the other two preference models, however, the discount factors closer to 1 did see significantly higher average times to attempt transmission. For certain applications, an average time to transmit of 24 hours may be unacceptable. If it is necessary to keep $r_{attempt}$ higher, using a lower value of t_{max} or a value of λ closer to 0 would lower the average time to transmit. A lower value of t_{max} in particular forces the algorithm to only consider satellite passes within a more constrained time frame.

Regarding noise distribution, Figures 5 to 10 also show results compared for two noise models, one where the RF noise values were drawn uniformly from all noise buckets, and one where the RF noise values were drawn uniformly from one bucket only. This second model was simulated, as remote sites have largely consistent RF background noise levels. The primary impact is that the algorithm consistently learned faster and converged to higher success rates when limited to the same bucket of RF noise values. For example, the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2023 doi:10.20944/preprints202305.0527.v1

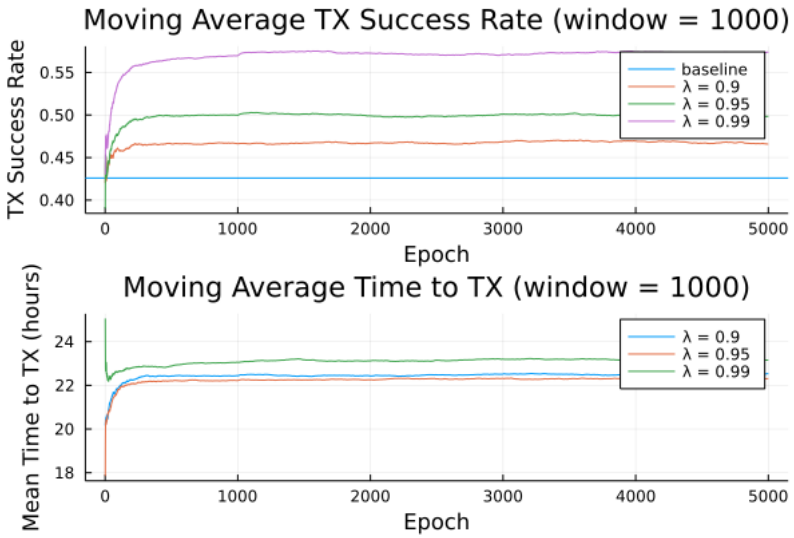

Figure 7. Simulated results for preference model 2 and random noise within 1 bucket.

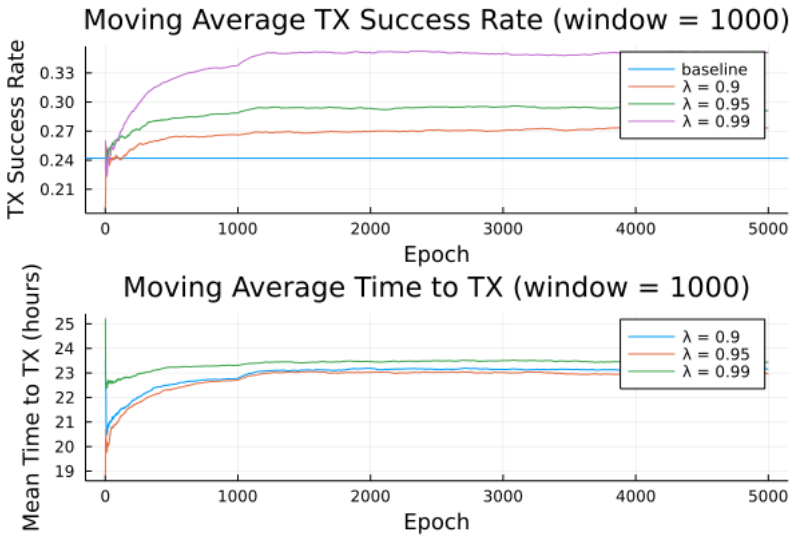

Figure 8. Simulated results for preference model 2 and random noise across all buckets.

algorithm learned in 1000 epochs for preference model 2 and fully random noise what it learned in under 300 epochs for same-bucket noise.

3.1. Power and Energy Consumption

Since the purpose of the packet scheduling is to reduce average power, it is valuable to evaluate energy savings. Using Equation 8 for $E_{attempt}$, values for parameters are shown in Table 7. While these values may depend on exact hardware setup and application, they are all constant, as opposed to terms that are variable, as detailed next.

In addition to the above constants, the ranges of the variables in the equation above are shown in Table 8. Note that "pessimistic" refers to the boundary of the interval that results in higher average power consumption, per the equation for $E_{attempt}$, and "optimistic" refers to the boundary of the interval that results in lower average power consumption. A value denoted as optimistic or pessimistic does not necessarily mean a value judgement of how good it is for system operation. For example, a lower $r_{attempt}$ means less frequent transmission attempts, which is good for energy consumption, but causes high average time to transmit and lost data. Also note the difference between average power consumption and the value for $E_{attempt}$ given by the equation above; while a low $r_{attempt}$ will result in a

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2023 doi:10.20944/preprints202305.0527.v1

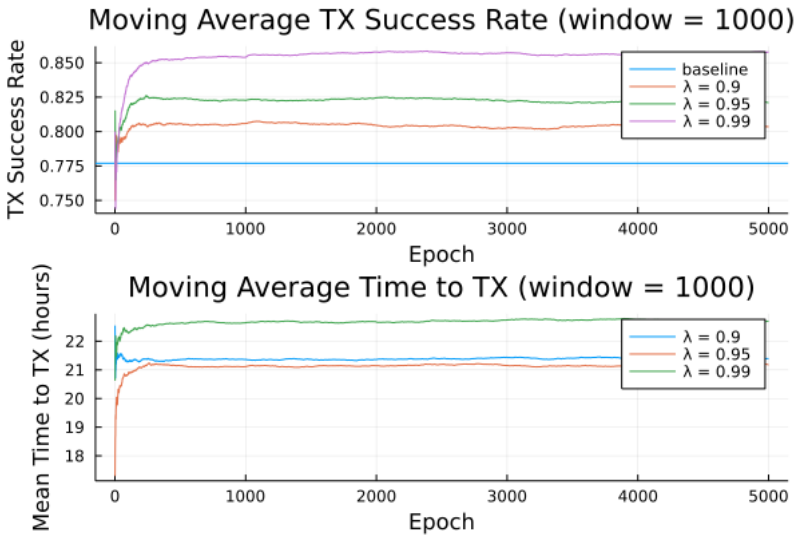

Figure 9. Simulated results for preference model 3 and random noise within 1 bucket.

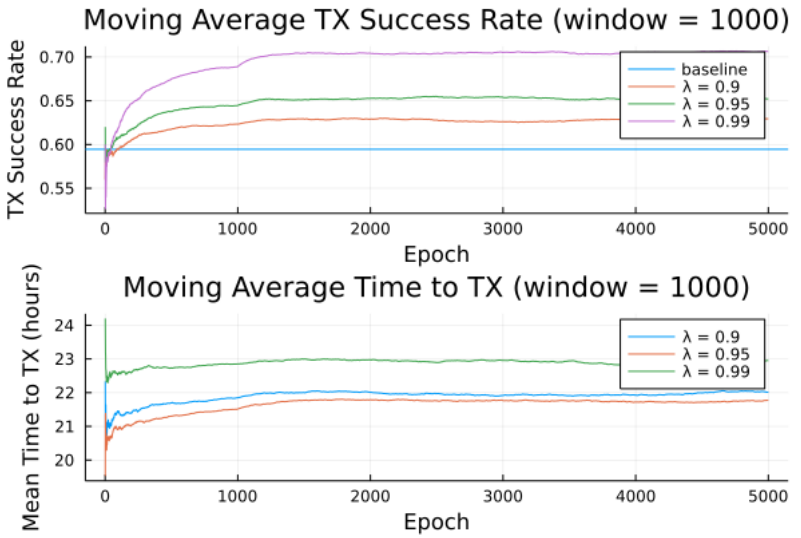

Figure 10. Simulated results for preference model 3 and random noise across all buckets.

higher $E_{attempt}$, it will result in lower average power, as shown by Equation 10 below. The term in the denominator is the average time elapsed during a complete cycle.

$$P_{avg} = \frac{E_{attempt}}{\frac{1}{r_{attempt}} + t_{GPS} + p_{success}\epsilon_{pass}t_{pass} + (1 - p_{success})t_{pass}} \tag{10}$$

The results of these four variables on average modem power consumption are shown in Figure 11. Most important to observe is that the two dominant factors in determining average power consumption are average success rate and average time between attempts. These two variables are not independent; higher success rates lead to lower attempt rates.

For perspective on the potential impact of improved success rates, we refer to Table 9. Note that the $p_{success}$ and $r_{attempt}$ values are taken from the simulations, and ϵ_{pass} and t_{pass} are taken as 0.5 and 25 minutes, respectively. For each preference model, three results are shown: 1) a baseline based on taking the earliest available passes and no scheduling, 2) another baseline based on the same average $r_{attempt}$ as the simulated results but no scheduling, and 3) the test case with scheduling and simulated average $r_{attempt}$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2023 doi:10.20944/preprints202305.0527.v1

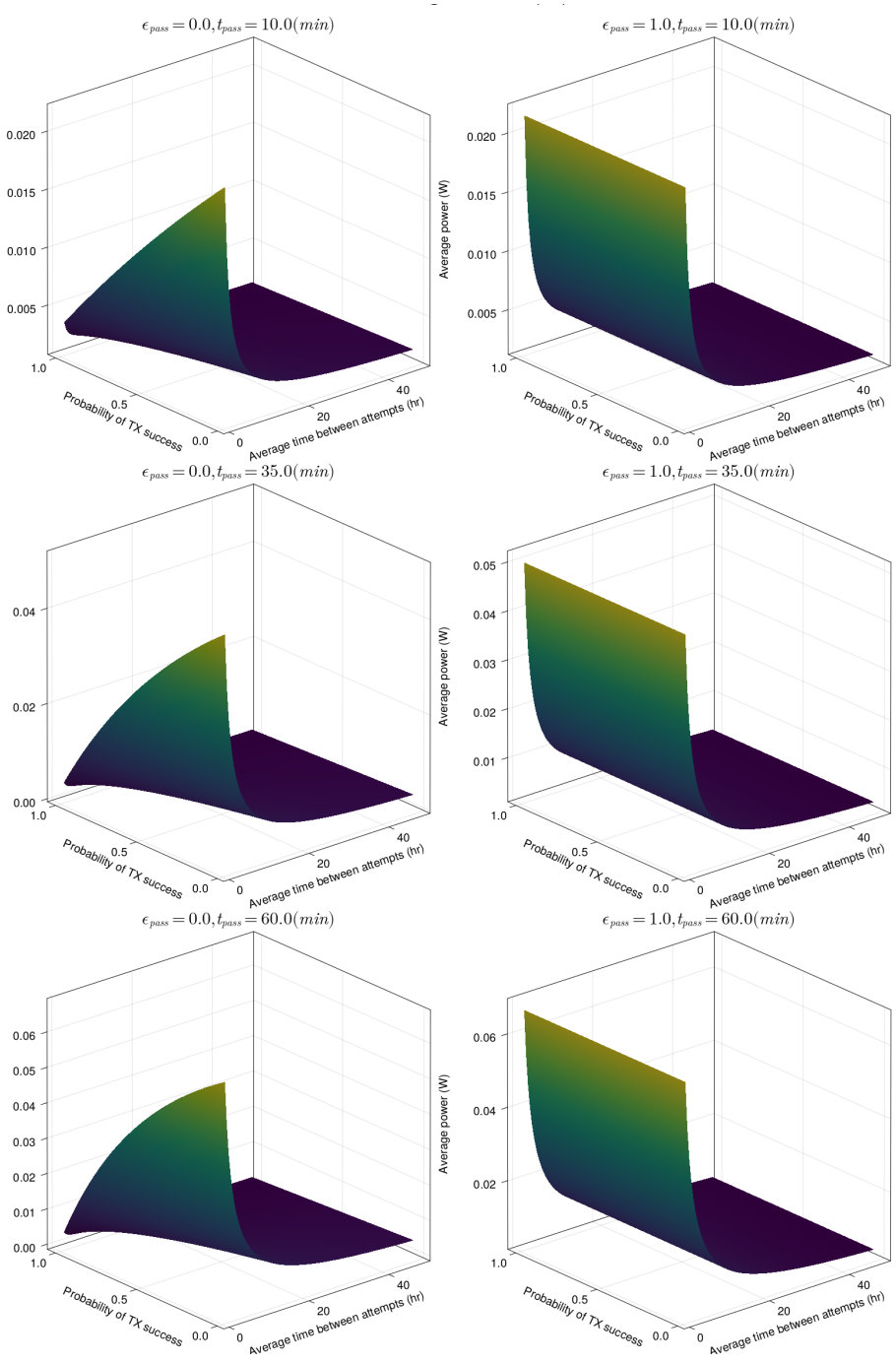

Figure 11. Average energy consumption of the Swarm modem under various variable values.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2023 doi:10.20944/preprints202305.0527.v1

Constant	Value
P_{SL}	$550\mu W$
P_{GPS}	$230mW$
t_{GPS}	$30s$
P_{RX}	$130mW$
E_{TX}	$12.24J$
r_{pkt}	$\frac{1}{3}hr^{-1}$

Table 7. Constants used for transmission attempt energy model.

Variable	Pessimistic Value	Optimistic Value
$r_{attempt}$	$1hr^{-1}$	$\frac{1}{48}hr^{-1}$
$p_{success}$	0.0	1.0
ϵ_{pass}	1.0	0.0
t_{pass}	$60min$	$10min$

Table 8. Sample variable ranges for transmission attempt energy model.

For simulated baseline values, the average time to attempt is represented as the first available satellite pass, modeled by

$$t_{SL,baseline} = \frac{p_{success}}{r_{pkt}} \Rightarrow r_{attempt,baseline} = \frac{r_{pkt}}{p_{success}},$$

which represents the algorithm for determining t_{min}, where successful transmission lead to waiting a minimum of $\frac{1}{r_{pkt}}$ hours, while unsuccessful attempts lead to no minimum wait.

The results demonstrate that online learning direct-to-satellite packet scheduling is capable of reducing average power consumption and battery requirements. Note that the dominant power saving comes from reducing average attempt frequency, although the improved success rate of the scheduling algorithm introduce significant power saving. Additionally, a direct-to-satellite packet scheduling algorithm enables lowering the attempt frequency, as it provides a built-in mechanism for selecting future passes.

4. Conclusions and Future Work

We have presented and evaluated the LEO satellite transmission scheduling for IoRT. A detailed simulation model was used to assess the suitability of the proposal for the water-level monitoring in remote locations, and the results demonstrate the ability of the proposal to facilitate energy-efficient data collection over prolonged periods of time.

The system was implemented on a printed circuit board (PCB), Figure 12, following Swarm recommendations for RF shielding by ground plane design and decoupling capacitors. The PCB includes from right to left: a Swarm M138 LEO Modem, and a Raspberry Pi

Preference Model	Success Rate	Attempt Frequency	Average Power	Required Battery Capacity
1	0.13	$2.564hr^{-1}$	$67.54mW$	$592.1Wh$
	0.13	$\frac{1}{24}hr^{-1}$	$3.810mW$	$33.40Wh$
	0.20	$\frac{1}{24}hr^{-1}$	$3.735mW$	$32.74Wh$
2	0.42	$0.7937hr^{-1}$	$29.31mW$	$256.9Wh$
	0.42	$\frac{1}{23}hr^{-1}$	$3.575mW$	$31.34Wh$
	0.57	$\frac{1}{23}hr^{-1}$	$3.405mW$	$29.85Wh$
3	0.78	$0.4274hr^{-1}$	$14.95mW$	$131.1Wh$
	0.78	$\frac{1}{22}hr^{-1}$	$3.234mW$	$28.35Wh$
	0.85	$\frac{1}{22}hr^{-1}$	$3.151mW$	$27.62Wh$

Table 9. Sample energy savings from simulated packet scheduling for a year of operation

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 8 May 2023 doi:10.20944/preprints202305.0527.v1

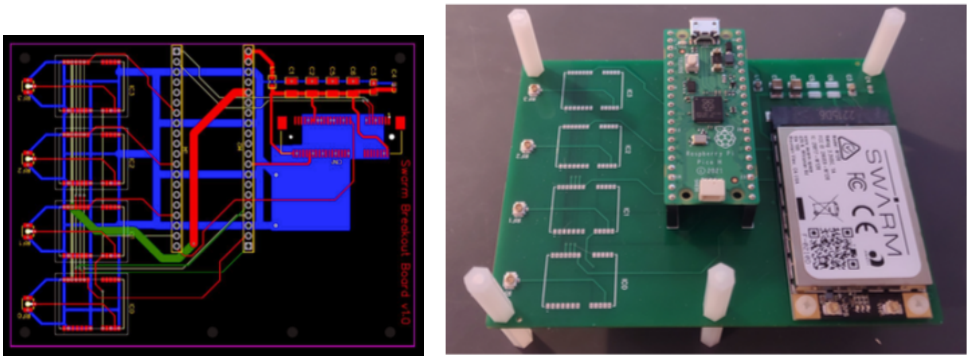

Figure 12. Board design: PCB layers and board populated with Raspberry and Swarm modules.

Pico, while four GNSS modules for GNSS-IR water-level measuring (together with GNSS antennas) are unpopulated. The proposed scheduling algorithm was implemented on a dual-core ARM Cortex M processor of the Raspberry Pi Pico, where each processor core executes one process of the code in Figure 3. The board is sending one message per hour packed as per Table 3, to fit within a single Swarm data plan for USD 60/year.

The whole analysis was done by simulations to reduce the burden in cost, time and also to combat location-dependence, since RF noise in the city differs from the intended remote locations. While we demonstrate the ability of our proposed scheme to adapt appropriately, obtain interpretable learning and explore the impact of variables on the performance of the algorithm, they are nonetheless simulated results. There are two natural steps that could be taken to improve in future: 1) the simulations could be based on extracted real-world data for a given application, or 2) multiple sensors could be placed in the field for weeks or months. The first option is natural — as the risk of needing to adjust the algorithm is high, making tuning in simulations and testing on hardware only when confident in good results is the best route.

4.1. Potential Simulation Improvements

The most apparent way to improve the simulations is to generate a more representative RF noise distribution and to use actual Swarm satellite passes. For the latter, here are the versions of the satellite pass prediction library as used on open-source SGP4 Arduino library or in Python [17,18].

4.1.1. Tradeoff Between Low Power and Learning Rate

Satellite pass algorithm in its current state exhibits low average $r_{attempt}$ attempt rate values (still sufficiently high for the intended application). This parameter could be improved, as it leads to a real sensor taking shorter time to learn. A downside to this is in smaller expected power savings, as more frequent transmissions consume significantly more average power. There is a tradeoff between having a high learning rate and achieving low average power consumption. Further research could examine the possibility of adaptive learning rates. One possibility for polar regions would be increasing the transmission attempt frequency (e.g., by decreasing the discount factor λ) during the summer when solar power is abundant, and lowering the transmission attempt frequency during the polar winters.

4.1.2. RF Noise Impact on Prototype System

One factor discovered in verification testing of a prototype is the sensitivity of the antenna to noise. For example, minor changes in the exact positioning of the PCB and microcontroller under the ground plane could vary the measured noise by as much as 3 dB. Clearly, the antenna receives RF interference from an unshielded device in the immediate vicinity (e.g., 10 to 20 cm). In fact, it was precisely this interference that played a role in the design of a online learning direct-to-satellite packet scheduling algorithm; since no method

could possibly account for the range of possible housing, ground plane, and even power supply designs, as well as varied site conditions and lines of sight, there could be no single ultimate "good" pass model, so a learning-based approach is used.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: The data used in this study was generated by a simulation program placed in the repository: https://github.com/garrettkinman/Self-Learning-Satellite-Pass-Selection.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pörtner, H.O.; Roberts, D.; Masson-Delmotte, V.; Zhai, P.; Tignor, M.; Poloczanska, E.; Mintenbeck, K.; Alegría, A.; Nicolai, M.; Okem, A.; et al. Summary for policymakers. *IPCC special report on the ocean and cryosphere in a changing climate* **2019**, *7*.
2. Carotenuto, F.; Brilli, L.; Gioli, B.; Gualtieri, G.; Vagnoli, C.; Mazzola, M.; Viola, A.P.; Vitale, V.; Severi, M.; Traversi, R.; et al. Long-Term Performance Assessment of Low-Cost Atmospheric Sensors in the Arctic Environment. *Sensors* **2020**, *20*. https://doi.org/10.3390/s20071919.
3. Purnell, D.J.; Gomez, N.; Minarik, W.; Porter, D.; Langston, G. Precise water level measurements using low-cost GNSS antenna arrays. *Earth Surface Dynamics* **2021**, *9*, 673–685. https://doi.org/10.5194/esurf-9-673-2021.
4. Karegar, M.A.; Kusche, J.; Geremia-Nievinski, F.; Larson, K.M. Raspberry Pi Reflector (RPR): A Low-Cost Water-Level Monitoring System Based on GNSS Interferometric Reflectometry. *Water Resources Research* **2022**, *58*, e2021WR031713. https://doi.org/https://doi.org/10.1029/2021WR031713.
5. Purnell, D.J.; Gomez, N.; Minarik, W.; Porter, D.; Langston, G. Precise water level measurements using low-cost GNSS antenna arrays. *Earth Surface Dynamics* **2021**, *9*, 673–685. https://doi.org/10.5194/esurf-9-673-2021.
6. Sanctis, M.D.; Cianca, E.; Araniti, G.; Bisio, I.; Prasad, R. Satellite Communications Supporting Internet of Remote Things. *IEEE Internet of Things Journal* **2016**, *3*, 113–123.
7. Centenaro, M.; Costa, C.E.; Granelli, F.; Sacchi, C.; Vangelista, L. A Survey on Technologies, Standards and Open Challenges in Satellite IoT. *IEEE Communications Surveys & Tutorials* **2021**, *23*, 1693–1720.
8. Jia, Z.; Sheng, M.; Li, J.; Niyato, D.T.; Han, Z. LEO-Satellite-Assisted UAV: Joint Trajectory and Data Collection for Internet of Remote Things in 6G Aerial Access Networks. *IEEE Internet of Things Journal* **2021**, *8*, 9814–9826.
9. Fraire, J.A.; Umaña, S.C.; Accettura, N. Direct-To-Satellite IoT - A Survey of the State of the Art and Future Research Perspectives - Backhauling the IoT Through LEO Satellites. In Proceedings of the ADHOC-NOW, 2019.
10. Huang, H.; Guo, S.; Liang, W.; Wang, K. Online Green Data Gathering from Geo-Distributed IoT Networks via LEO Satellites. *2018 IEEE International Conference on Communications (ICC)* **2018**, pp. 1–6.
11. Palma, D.; Birkeland, R. Enabling the Internet of Arctic Things With Freely-Drifting Small-Satellite Swarms. *IEEE Access* **2018**, *6*, 71435–71443.
12. Qi, Y.; Pan, L.; Liu, S. A Lyapunov optimization-based online scheduling algorithm for service provisioning in cloud computing. *Future Generation Computer Systems* **2022**, *134*, 40–52. https://doi.org/https://doi.org/10.1016/j.future.2022.03.037.
13. Marcinkevics, R.; Vogt, J.E. Interpretability and Explainability: A Machine Learning Zoo Mini-tour. *ArXiv* **2020**, *abs/2012.01805*.
14. Silver, D. Lectures on Reinforcement Learning. URL: https://www.davidsilver.uk/teaching/, 2015.
15. Sutton, R.; Barto, A. *Reinforcement Learning, second edition: An Introduction*; Adaptive Computation and Machine Learning series, MIT Press, 2018.
16. Kuleshov, V.; Precup, D. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028* **2014**.
17. Hopperpop. Library for calculating satellites positions and predicting overpasses.
18. Rhodes, B. Python version of the SGP4 Satellite Position Library.