Transmitting, Fast and Slow: Scheduling Satellite Traffic through Space and Time

Bill Tao, Maleeha Masood, Indranil Gupta, Deepak Vasisht
University of Illinois Urbana-Champaign

ABSTRACT

Earth observation Low Earth Orbit (LEO) satellites collect enormous amounts of data that needs to be transferred first to ground stations and then to the cloud, for storage and processing. Satellites today transmit data greedily to ground stations, with full utilization of bandwidth during each contact period. We show that due to the layout of ground stations and orbital characteristics, this approach overloads some ground stations and underloads others, leading to lost throughput and large end-to-end latency for images. We present a new end-to-end scheduler system called *Umbra*, which plans transfers from large satellite constellations through ground stations to the cloud, by accounting for *both spatial and temporal factors*, i.e., orbital dynamics, bandwidth constraints, and queue sizes. At the heart of Umbra is a new class of scheduling algorithms called *withhold scheduling*, wherein the sender (i.e., satellite) selectively under-utilizes some links to ground stations. We show that Umbra's counter-intuitive approach increases throughput by 13-31% & reduces P90 latency by 3-6 ×.

CCS CONCEPTS

• **Networks → Mobile networks**; **Packet scheduling**.

KEYWORDS

Satellite Networks,Satellite Data Scheduling,Earth Observation Satellites

ACM Reference Format:
Bill Tao, Maleeha Masood, Indranil Gupta, Deepak Vasisht. 2023. Transmitting, Fast and Slow: Scheduling Satellite Traffic through Space and Time . In *The 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '23), October 2–6, 2023, Madrid, Spain.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3570361.3592521

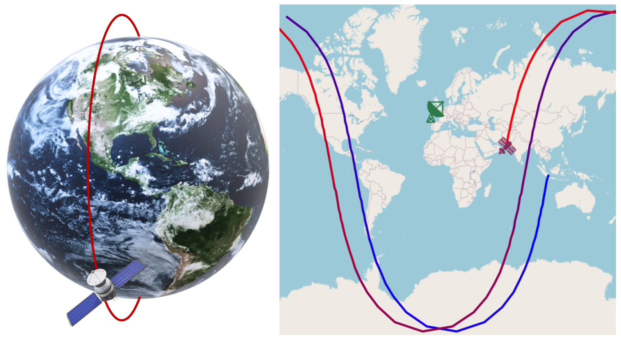


Figure 1: LEO Satellite Constellations. *Earth observation satellites operate in polar low Earth orbits (around 1.5 hours per orbit). As the Earth rotates under them, they scan different parts of the Earth in every orbit. (Right) A LEO satellite motion (Planet Dove [37]) over 3 hours (blue to red) waiting to come in contact with the ground station (green) to transfer data.*

1 INTRODUCTION

The current generation of Low Earth Orbit (LEO) satellite constellations is a unique new class of mobile networking systems characterized by scale and spatio-temporal dynamics. Advancements in technology over the last decade have led to a five-fold increase in the number of LEO satellites in orbit [42]. Many companies [1, 14] launched constellations containing hundreds of satellites to perform frequent high-resolution monitoring of Earth. These satellites rotate around the Earth in low orbits (< 1000 km above Earth) and track planet-scale events. Just during 2021-22, LEO constellations were used to monitor the war in Ukraine [48], the Tonga volcanic eruption [18], and California forest fires [8].

Each satellite generates approximately one Terabyte of data per day. A satellite communicates this massive data to the cloud (for analysis and distribution of Earth imagery) via multiple (fixed) ground stations located thousands of Kilometers away on Earth. This data transfer problem is challenging due to the scale of the imagery and the temporal and spatial challenges, which arise from natural orbital dynamics of LEO satellites (see Fig. 1), and uneven layout of ground stations. The contact between a satellite and a ground station is short-lived: four to six ten-minute windows per day per satellite-ground station pair. In order to reduce interference from ambient signals and blockages and to increase the number of satellite-ground station contacts , ground stations are typically located in remote regions, e.g., closer to the poles, and away from large populations. This limits the backhaul bandwidth from the ground station to the cloud. Due to these factors getting data from satellites to the cloud suffers from day-level delays.

This paper makes the following contributions:

- We discover a new phenomenon that we call *Uneven Queuing Effect* or UQE (pronounced "You-k"), wherein the prevalent strategy of greedy full-utilization data transfer from satellites to ground stations, is creating load imbalance across ground stations and leading to sub-optimal end-to-end throughput and latency, all due to temporal and spatial reasons.
- We propose *withhold scheduling*, a new class of satellite-ground station transfer algorithms, for LEO constellations.
- We build the *Umbra* data transfer system, where we design, implement, and evaluate a new withhold scheduling algorithm based on *time-expanded networks*.
- We perform a large scale trace-driven simulation using data collected from a real 153-satellite constellation.

1.1 Uneven Queuing Effect (UQE)

Due to orbital dynamics, a satellite moves past a ground station receiver in less than ten minutes. Therefore, the conventional wisdom in satellite networks has been to transmit data greedily (or "fast"), i.e., send as much data as possible to a ground station during its contact, using the full available bandwidth. Naturally, a bulk of past work focuses on improving the radio design at the satellite and the ground stations so that they can maximize the amount of data transfer during the short contacts [12, 13, 41]. This line of work has made great progress, and today, even small cubesats in low earth orbits can achieve Gbps links to Earth [13].

With these advances in satellite-ground links, the status quo "fast" transmission style for data from satellites to ground stations leads to long outgoing queues (to the cloud) at some ground stations, and relatively shorter queues and thus idling, at other ground stations. This arises from two reasons. First, ground stations have an uneven (heterogeneous) spatial distribution, due to logistical reasons involving spectrum licensing, country-wise regulations, proximity to poles, etc. This means that the *amount* of *new* data that a satellite has in between its *consecutive* ground station contacts may vary widely, and lead to unbalanced queues at ground stations.

We term this new phenomenon we discovered as the *Uneven Queuing Effect* or UQE[1]. Fig. 2a shows an example of UQE. The shown satellite passes over consecutive ground stations A, B, and C. However the A-B distance is longer than the B-C distance. This means the satellite collects far more data during its A-B segment than its B-C segment. Greedy transfer means B would receive 9 GB from the satellite, while C would receive only 1 GB. Thus, UQE leads to unbalanced queues at B vs. C.

The UQE problem is further exacerbated because different ground stations can have different backhaul bandwidths to the cloud, from 100s of Mbps to a few Gbps. This means that outgoing queue lengths at ground stations can wildly vary across time and space. Therefore, images stuck at high queue, low bandwidth stations experience large delays. This situation is worsening as more compute resources are being added to ground stations for "edge"-style processing, which further exaggerates the problem of load imbalance due to both network delays and computational delays, both of which could be imbalanced. Therefore, even if backhaul bandwidths increase in the future, UQE will continue to back up queues.

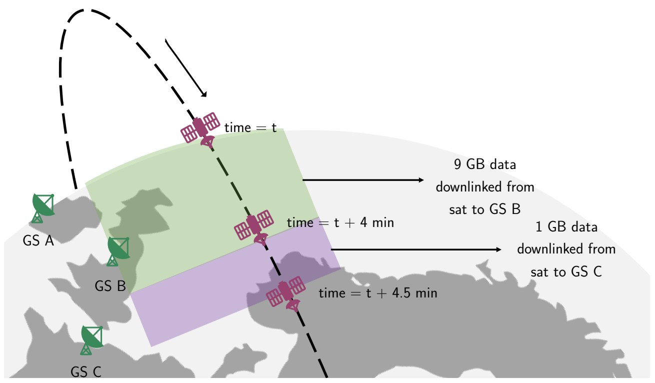
Fig. 3 shows UQE causes idling at some ground stations and uneven egress throughput at the cloud in the greedy approach ("Baseline"), while our system ("Umbra", described soon) offers stable throughput. Section 3.3 formally proves that UQE causes quadratic growth in Greedy's queues.

1.2 Withhold Scheduling

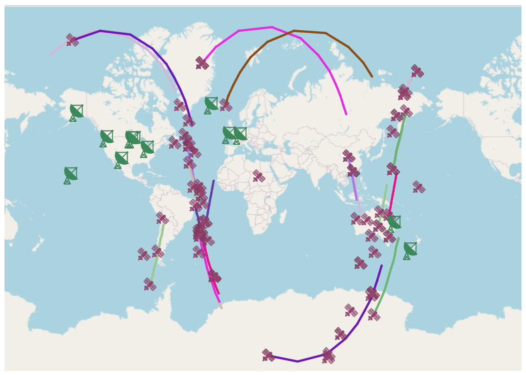
To counter UQE, we define a new scheduling paradigm for satellite data transfers called *withhold scheduling*. The key idea in withhold scheduling is to allow a satellite to selectively *under*-utilize a subset of its ground station contacts and intelligently *withhold* data for subsequent links if it identifies an opportunity for a better end-to-end latency in the future. Withhold scheduling aims to equalize queue sizes across ground stations and leads to higher throughput and lower latency for the transfer of satellite data to the cloud. Returning to Fig. 2a, if the satellite were to intelligently withhold 4 GB of data from B, this would equalize data transferred to B and C. If on the other hand, C had a $1.5\times$ higher backhaul bandwidth (i.e., to the cloud) than B, transferring 4 GB to B and 6 GB to C would be preferable.

Withhold scheduling needs to tell each satellite: *When to withhold*, and *How much to withhold*. This is complex because any decision to withhold data needs to account for both spatial factors and temporal factors. *Spatial factors* include the relative positions of satellites and ground stations.

[1]The UQE effect is analogous to the *Waiting Time Paradox* in networks and public transport [2]. For instance, with uneven bus arrivals, the average waiting time is greater than 50% of the average interarrival gap.


time = t
9 GB data downlinked from sat to GS B
time = t + 4 min
1 GB data downlinked from sat to GS C
time = t + 4.5 min
GS A
GS B
GS C

(a)



(b)

Figure 2: LEO Satellite Orbits and Load Imbalance. *(a) Uneven Spatial layout of ground stations leads to different "new" data sizes abvailable at a satellite during the next ground station contact. (b) Umbra needs to reason about such factors for a large number of satellites moving around the Earth. The figure shows a snapshot of 70 Planet Dove [37] satellites with motion traces for 10 of them over a 10 minute window.*

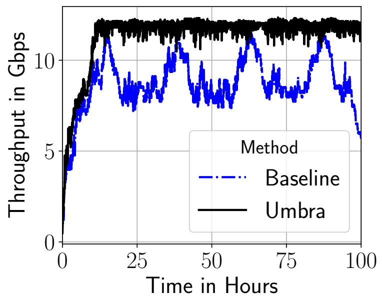


Figure 3: *Cloud Ingress Timeline: Greedy (Baseline) vs. Umbra.*

Temporal factors for withhold scheduling include: (a) the evolution of this link quality and visibility over time due to the orbital motion of the satellite, and (b) the queue size variation at ground stations. Furthermore, any decision made by a satellite (say, X) to withhold data in a time slot has multiple downstream effects: (i) a *different* satellite (say, Y) may choose to use this slot to transfer data to the *same* ground station, (ii) satellite X now needs a slot in the future at a different ground station (and with increased urgency).

Withhold Scheduling via Time Expanded Networks: We formulate the spatial and temporal factors uniquely using a *time expanded network* (or TEN). This allows us to capture spatial factors, like connections between satellites, ground stations and cloud, via a graph representation at each instance of time. Further, we also add *holdover* edges from a vertex to itself in the future, signifying the possibility of the vertex (satellite) withholding outgoing data in spite of available bandwidth. Any withhold scheduling algorithm or heuristic can be captured via this TEN.

Given this TEN network, we design a polynomial-time algorithm that combines bipartite matching, max flow, and binary search. We also compare our approach against simpler heuristics. Time expanded networks have been used to plan flows in the internet [16, 17], and in sneakernets [7]. Our work is the first to adapt them for satellite data transfers.

We build *Umbra*, a new system for scheduling data transfers from satellite constellations to the cloud via ground stations. Umbra accounts for the dynamics of satellite motion, back-end bandwidth constraints of ground stations, and queue sizes at ground stations. We implement the Umbra scheduler, and our trace-driven evaluation uses 6 million images captured by the Planet Dove constellation [37] comprising 153 satellites' trajectories and collected across 15 days. This data is the real set of images collected by the constellation. We simulate the orbital dynamics of the satellites and a ground station layout by using Planet's published and frequently updated orbital information [25, 29]. Our paper is, to the best of our knowledge, the largest evaluation performed using data collected by a real operational satellite constellation. Our evaluation shows that the Umbra scheduler can improve the satellite constellation's throughput of by 13-31% and 90th percentile latency by 3-6× compared to a greedy baseline and a heuristic-based scheduler.

2 SATELLITE NETWORKING PRIMER

There are nearly 5000 satellites in orbit today, up by 5× compared to a decade ago [42]. The increase has been driven by reduced cost of designing and launching hardware for small satellites (e.g., "shoebox-sized" cubesats). A single rocket can launch multiple such satellites using rideshare agreements that amortize cost.

Satellite Orbits: Emerging LEO constellations for earth observation typically operate in polar orbits around 500 km above the Earth. A given satellite may return to the same location above Earth only every six to twelve days. Satellite operators—e.g., Planet Inc. [14], and Spire [1])—deploy large-scale constellations comprising hundreds of satellites to increase imaging frequency to multiple images per day. This is in contrast to traditional earth observation constellations that have only a few satellites, e.g., 2 satellites in the European Space Agency's Sentinel 2 [15].

A satellite's location with respect to Earth is reasonably predictable using Two Line Element (TLE) orbit descriptors published at regular intervals by multiple agencies such as Celestrak [29]. This means that the satellite-ground station contact time points are predictable and we use these as an input for scheduling. Past work [44] has also demonstrated the ability to predict the radio link quality across time.

Imaging Equipment and Data Volume: Earth observation satellites capture images of Earth in different parts of the frequency spectrum, e.g., RGB, Radio Waves, Infrared, etc. The multi-spectral imagery as well as the high resolution are responsible for high volumes of data transfer from satellite to Earth. The Dove constellation captures 120 TB of data per day on average, in our evaluation period.

Ground Station Design: The satellite to ground station link is a high frequency link (e.g., X-band 8-10 GHz) with downlink bandwidths of up to 2 Gbps and uplinks of a few Kbps [13]. Bandwidth varies as a function of distance between satellites and ground stations during a contact period, and across contact periods. A single ground antenna can only talk to one satellite at a time. However, a satellite operator may deploy multiple antennas at the ground station, with each antenna talking to an independent satellite. We assume that satellites cannot communicate amongst each other, i.e., there are no inter-satellite links (this is true for all major LEO constellations today).

The ground station locations are selected using several constraints such as land availability, spectrum licensing requirements, lack of interference, orbital calculations, etc. The ground stations transfer data to the cloud using a backhaul link. The quality of the backhaul connection depends on the location of the ground station and can vary from 100s of Mbps to a few Gbps. This bandwidth is relatively stable across time (as opposed to the satellite-ground station bandwidth which varies due to orbital motion). While there is scant public information about the nature of these links, the range of 100s of Mbps to a few Gbps is consistent with anecdotal evidence based on both our conversations with satellite operators and public statements by ground station operators [38].

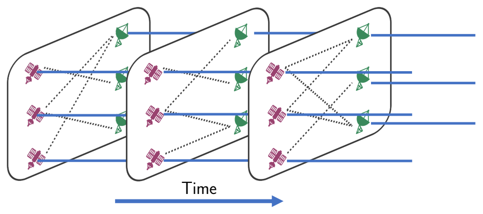
Data Download Process: The images collected by a satellite arrive at the cloud endpoints via two stages: satellite to ground station first, and then ground station to cloud. Each stage incurs hour-level latencies today. For the first step, the access to a ground station is the key bottleneck, i.e. a satellite must wait till its orbit brings it near a ground station, before it can transfer data. For the second step, the backhaul connectivity (to the cloud) is a bottleneck, especially for ground stations that are remote and/or get disproportionately high amount of data from satellites.

Assumptions: In this paper, we assume that (a) satellites cannot send data to each other directly. This is true for all major LEO earth observation satellite constellations today; (b) ground stations cannot send data to each other either, which is because ground station-ground station communication would consume the same bandwidth (to the Internet) that the ground station could use to communicate to the cloud and (c) ground stations are not shared across multiple applications Nevertheless, we believe our withhold scheduling algorithms generalize to any topology that relaxes these assumptions. We also discuss these relaxations at the end of the paper.

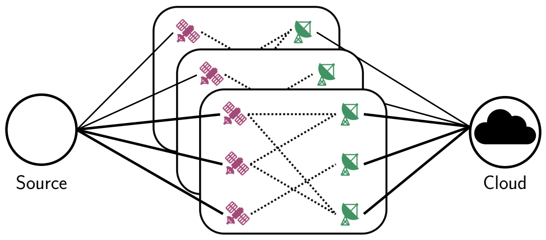
3 WITHHOLD SCHEDULING IN UMBRA

In scheduling the transfer of satellite data to ground stations and then to the cloud, the key decisions that a withhold scheduling algorithm needs to make are: (a) Should a satellite withhold any data during a given contact? and (b) (if yes) How much data should it withhold? The decisions depend on the following factors:

- **Orbital Motion of Satellites:** This defines feasibility, quality, and duration of contact with ground stations in the future. One important factor is the predictable orbital motion of each satellite, i.e., the sequence and timing of ground station contacts are known. However link quality may vary—if it is going to be weak in the future, it may not be optimal to withhold a large amount of data.
- **Contention for Ground Station Time:** Ground stations are typically fewer (10s) than the number of satellites (100s). Thus multiple satellites may contend for the same ground station. Each ground station may have multiple antennas, but only one antenna can talk to a satellite at a time, and vice-versa, i.e., a one-to-one mapping.
- **Traffic Pattern Evolution at Ground Stations:** Queue sizes evolve over time at different ground stations. For instance, if several satellites decide to withhold data at a given ground station, this ground station may become idle, while subsequent ground stations build long queues.



(a) *Time-expanded View*



(b) *Spatial View*

Figure 4: Umbra's Time Expanded Network. *Umbra formulates the withhold scheduling problem as a time expanded network.*

3.1 Time Expanded Network Formulation

We formulate the satellite data transfer problem over space and time, as a *time expanded network* [7, 16, 17], or briefly a TEN. Because of the predictable orbits of each satellite, any transfer strategy can be specified within a TEN. Given snapshots of the network at different time instances, our TEN creates *holdover* edges between time instances of a given node, signifying the node's ability to withhold data.

Fig. 4 shows an example TEN graph evolving in time as satellites move. Each snapshot (or layer, or time) represents a time duration during which those satellite-ground station contacts are possible. The bipartite graph at each time is the set of possible satellite-ground station contacts during that time. The horizontal lines across layers are the holdover edges. To solve the transfer problem, the key needs are thus: i) to select from each (layer's) bipartite graph a *one to one matching* of satellites to ground stations, and ii) at each satellite, to decide how much data to transmit along a downlink vs. to itself via a holdover edge.

Formulation: Formally, denote the set of satellites as $SAT = \{s_1, s_2, \ldots, s_n\}$. Each satellite, s_i captures imagery over time and generates new data $p_i(t)$ at time duration t. Time units are of fixed duration. We include an imaginary source node with an edge of capacity $p_i(t)$ from the source to satellite, s_i to denote this data capture process. The data volume from the source to a satellite varies over time because: (a) earth imagery satellites do not always image over water (e.g., large oceans), and (b) a satellite may use (atmospheric) cloud detection to reject obscured imagery.

Next, the set of ground stations is $GS = \{g_1, g_2, \ldots, g_n\}$. We also add a single sink vertex denoting the cloud. Each g_i has a pre-configured egress bandwidth $b_i(t),\ t = 1, 2, \ldots, T$ to the cloud vertex. We use the predictable, pre-computed orbit of the satellites, and the positions of the ground stations to estimate the bandwidth of each satellite-ground station link. We denote as $b_{s_i,g_j}(t)$ the bandwidth at time t between satellite s_i and ground station g_j. Thus $b_{s_i,g_j}(t)$ is the capacity tagged on the $s_i \rightarrow g_j$ edge of the bipartite graph at time t. We set the capacity of the holdover edges to ∞. This is reasonable because typical satellites have storage of multiple TBs (2 TBs in Planet's Dove [12], sufficient to hold multiple days of data, far exceeding inter-contact durations.

Our goal is to compute a *data transmission plan*, which can be formulated as a matrix $D_{i,j}(t)$, representing the amount of data satellite s_i downlinks to ground station g_j at time t. D is subject to the following constraints:

- A satellite communicates with at most one ground station at a time: $\forall t, i, j_1 \neq j_2,\ D_{i,j_1}(t) = 0 \vee D_{i,j_2}(t) = 0$.
- A ground station communicates with at most one satellite at a time: $\forall t, i_1 \neq i_2, j,\ D_{i_1,j}(t) = 0 \vee D_{i_2,j}(t) = 0$. (Generalizable to multiple antennas per ground station.)
- A satellite's transfer speed cannot exceed downlink bandwidth: $\forall t, i, j,\ D_{i,j}(t) \leqslant b_{s_i,g_j}(t)$.
- A satellite cannot transmit more data than it collects: $\forall t, i,\ \sum_j \sum_{\tau=1}^{t} D_{i,j}(\tau) \leqslant \sum_{\tau=1}^{t} p_i(\tau)$.

Each ground station always utilizes its full bandwidth to upload the images to a cloud service. Therefore, the upload amount $u_j(t)$ for ground station g_j (to the cloud) at time t is:

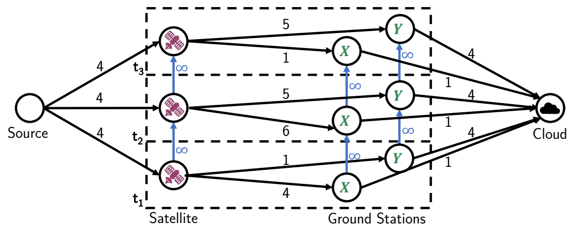
$$u_j(t) = \max\left(\sum_i \sum_{\tau=1}^{t} D_{i,j}(\tau) - \sum_{\tau=1}^{t-1} u_j(\tau), b_j(t)\right)$$

Optimization Objective: Mathematically, our objective is to maximize the end-to-end throughput for the data transfer process from the satellite to the cloud:

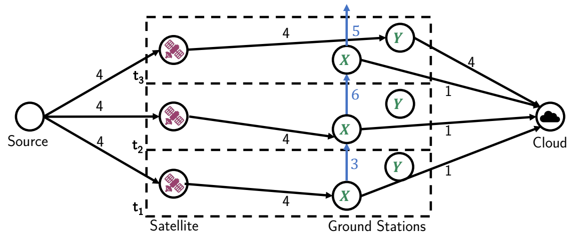
$$D^* = \arg\max_D \sum_i \sum_t u_i(t) \tag{1}$$

Example: Fig. 5a shows a simple example with one satellite and 2 ground stations X and Y. X has a lower backhaul (cloud) bandwidth than Y. For simplicity, we expand the graph to only 3 time steps. The weight on each edge represents its capacity per time unit. Infinite weights on holdover (blue) edges show infinite storage at satellite and ground stations.

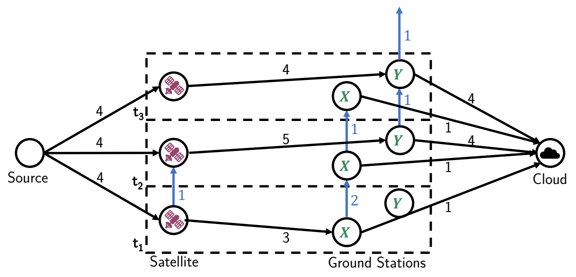
Fig. 5b shows the traditional fast (greedy) approach: at time t_1, the satellite connects to ground station X to maximize its downlink and sends 4 units of data. At time t_2, it maintains that connection and sends an additional 4 units of data. However, note that this leads to queue build up at the ground station since its bandwidth to the cloud is limited. This causes increasing values on the temporal (holdover)



(a) *Example Network*



(b) *Traditional Solution (Greedy)*



(c) *Optimal Withhold Scheduling*

Figure 5: Time Expanded Network (TEN) Example, with Greedy Result, and Optimal Result. *(a) Example time-expanded network with 1 satellite and 2 ground stations, (b) Greedy solution that transfers 7 data units, (c) Optimal (withholding) solution that transfers 11 data units.*

edges for this ground station. Overall, this approach transfers 7 units of data. 5 data units remain at X, un-transferred.

Fig. 5c shows the optimal strategy (by withholding). At time t_1, the satellite decides to withhold 1 unit of data, thus under-utilizing its link to X. At time t_2, the satellite transmits 5 units of data (4 units of new data and 1 unit of withheld data) to Y. Crucially, the satellite picks Y because of its stronger backhaul link, even though the satellite has a stronger link to X. At t_3, the satellite transmits the usual 4 units. *The single unit of withholding from t_1 to t_2 increases the total data transferred in 3 time units to 11 units of data (vs. 7 data units under greedy).* Only 1 data unit remains at Y, un-transferred.

The key insight is that transferring more data to ground stations with better backhaul links may be preferable, in spite of the delay incurred by withholding data.

3.2 Scheduling Algorithm

Our scheduling algorithm needs to make some "hard" selections and some soft selections. Hard selections matches satellites with ground stations to transmit to. This selection is "hard" because the scheduler can only select one satellite or another to transmit to a ground station. Withholding data is a soft decision, i.e., a satellite can choose to withhold all or a fraction of its data. To reduce the complexity of the solution, we take a two-stage approach: (1) matching satellite-ground station pairs, and (2) max flow.

3.2.1 Stage 1: Matching Satellite-Ground Station Pairs. First, we solve the assignment between satellites and ground stations independently for each time step in the TEN graph. For time t, we consider the bipartite graph consisting of satellites and ground stations. Our goal is to select a subset of edges that maximize the sum of weights of selected edges. Each edge (s_i, g_j) has a weight $R_{i,j}(t) = \max\left(b_{s_i,g_j}(t), cache_i(t)\right)$ where $cache_i(t)$ is the amount of data available at satellite s_i at time t. We calculate the maximum matching via the classical Hungarian algorithm [32]. This algorithm runs in $O(n^3)$ time to generate an optimal solution, where n is the number of satellites (or ground stations, whichever is larger). Thereafter, we calculate the data amount each satellite can downlink, and the cache (holdover) size on satellites. We update the TEN to remove links that were not picked by this step. This yields a matching graph for each time step, wherein each satellite is connected to at most one ground station, and vice-versa.

3.2.2 Stage 2: Maximum Flow Across Time. Next, to make withholding decisions, we reason across time. We do so by formulating the optimization problem as a maximum flow problem from source to sink in our entire TEN graph, containing all satellites and ground stations, across multiple time steps (typically a day, but could be shorter). We use the push-relabel algorithms [20], with a complexity $O(V^2\sqrt{E})$, where V is the node set and E is the edge set.

This solution reveals holdover decisions: if the optimal flow passes through any holdover edges, say between time t and $(t+1)$ at satellite s_i, this implies satellite s_i chooses to withhold that amount of data at time t.

3.2.3 Binary Search for Optimal Latency. Our algorithm so far is optimized for throughput. To optimize for latency without affecting throughput, we use the following approach: if the original TEN run was on time interval $[0, T]$ (i.e., was completed in T steps), we find the smallest value of $T'(\leq T)$, so that the throughput of TEN on $[0, T']$ is no lower than 99% of throughput of TEN's solution on $[0, T]$. This reduces latency as it forces data transfers to be completed earlier by T' instead of T. We find T' by performing a binary search on the interval $[0, T]$. The binary search is feasible as throughput increases monotonically with T', since more flow opportunities exist at higher T' values. Once T' is found, we execute the next TEN on time $[T', T' + T]$, and repeat.

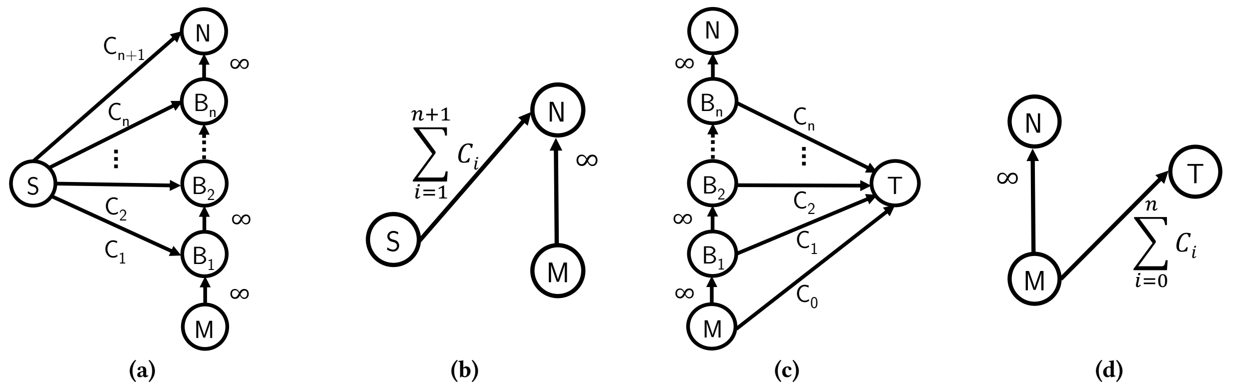


Figure 6: Umbra's Graph Simplification Optimization on Satellite Time Expanded Networks. *(a) Disconnected Satellite: A canonical subgraph at the satellite (B_is), where $B_1, \ldots, B_n$ has no other edges, (b) Corresponding Simplified structure (for max flow). (c) Purely Transmitting Ground Station: A canonical structure at the ground station (B_is), where $B_1, \ldots, B_n$ have no other edges. (d) Corresponding Simplified structure (for max flow).*

3.2.4 Graph Simplification Optimization. The TEN graph can be large. Consider Planet Inc.'s Dove constellation with (over) 150 operational satellites and 12 ground stations. If each time instant is a 1 minute granularity, and we calculate max flow across 1 day, then the resultant time expanded network has 2 Million nodes. Our max flow algorithm may take prohibitively long.

To optimize this, we first observe that the contacts between ground stations and satellites are sparse. During a 5 day period, the number of edges between the satellites and ground stations, in the Planet Dove dataset, are in the ballpark of 10K. This implies that a large number of satellites in the graph only have withholding edges and no connections to ground stations for long durations, as depicted in Fig. 6a. Similarly, ground stations also experience long durations where they are only transmitting to the cloud (Fig. 6c).

We "collapse" such consecutive node sequences into one fused node, i.e., all intermediate nodes inside a fused node have only one in-neighbor and one out-neighbor. This reduces the search space for the max flow algorithm without affecting the correctness of the calculated solution. Fig. 6 shows two canonical scenarios.

3.3 Analysis: How Bad is UQE?

UQE (Section 1.1) leads to quadratically long queueing times.

Theorem 1. *The greedy ("fast") transfer approach (Sections 1.1) causes queuing times to increase proportionally with variance of distances between consecutive ground stations.*

Proof. Consider one satellite S and its orbit. Let N = number of Ground Stations (GSs) that S passes. N is a fixed constant and we only vary the locations of GSs along S's orbit. Let x be the random variable for the distance between consecutive GSes encountered by S. Let the mean of x be μ. Rewrite $x_i = \mu + \delta_i$. Since $\frac{\sum_i x_i}{N} = \mu$, and N is constant, we have: $\sum_i \delta_i = 0$.

Now, first, the probability of a given piece of data being picked up during the ith GS-GS segment is proportional to distance x_i. Therefore (and second), the average additional waiting time for this piece of data at the (second) GS is proportional to its average added queue length, which is $\frac{x_i}{2}$.

Putting these together, the average queueing time for a piece of data is proportional to

$$Y = \sum_i \frac{x_i^2}{2} = \frac{1}{2} \cdot (\sum_i (\mu + \delta_i)^2) = \frac{1}{2} \cdot (\sum_i \mu^2 + \sum_i (2 \cdot \mu \cdot \delta_i) + \sum_i \delta_i^2).$$

The middle term is zero since $\sum_i \delta_i = 0$. So $Y = \frac{1}{2} \cdot (\sum_i \mu^2 + \sum_i \delta_i^2)$. The first term is a constant (given N), and the second term is proportional to the variance of inter-GS distances. □

4 SYSTEM DESIGN

Fig. 7 shows Umbra's control plane architecture. Umbra's scheduler runs on the cloud and communicates the latest schedule to ground stations, which then relay them to satellites upon next contact. The two key components in Umbra are: (a) Simulator, and (b) Scheduler. The former simulates the evolution of the satellite-ground station links using Two Line Element (TLE) orbit descriptors to both perform orbit calculations [24] and to compute link capacities using a link quality model [26–28]. Profilers running on ground stations continuously relay queue sizes and cloud bandwidth data as input to the Umbra Simulator. Umbra's second component, the Scheduler, interacts with the Simulator in an interactive way. The Scheduler constructs the time expanded network (TEN) and computes the optimal data transfer plan.

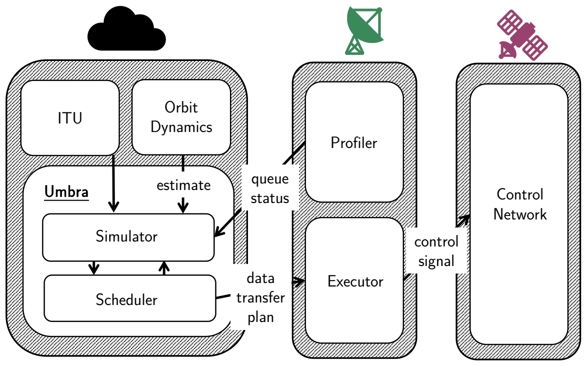


Figure 7: Architecture of Umbra (Control layer).

Updating the data transfer plan: Typically, the TLE orbit descriptors are updated periodically (on the order of a day to few days) to maintain accuracy. Therefore Umbra pulls new orbital data and calculates a new plan every five days, and relays it to the ground stations and satellites. We later evaluate the effect of an outdated plan. Umbra can be forced to generate a new schedule upon events like crashes, new satellite deployment, ground station upgrades, etc.

Handling component failure: Satellite failure is common in large constellations. For example, solar flares recently caused 40 of 49 SpaceX satellites to fail after a recent launch [4]. Ground stations may also fail, due to power outage, machine reboot, extreme weather or local events. We assume the cloud has redundancy and is always available.

Whenever a satellite fails, Umbra computes a new schedule. While a new schedule is being calculated, all satellites and ground stations continue using the old (latest) schedule. Newly joining satellites wait to receive a new plan before transmitting anything and store data locally.

The failure of a ground station has a greater impact, since satellites have to route their data through the ground station. After such a failure, while Umbra is generating a new data transfer plan, a satellite which encounters a failed ground station will detect the lack of acknowledgments, and merely withhold all its planned data until it encounters the next non-faulty ground station.

5 EXPERIMENTAL SETUP

We implemented Umbra in a simulator using about 500 lines of Python code. We plan to release our simulator code in the public domain. In our experiments, we inject traces derived from Planet Inc.'s Dove constellation [37] into our simulator.

5.1 Satellite Constellation

Our satellite dataset from Planet Inc. contains orbital data collected from 153 satellites in orbit as part of the Dove satellite constellation [37]. These satellites orbit around the Earth in one of two polar orbits, as shown in Fig. 8, and collect RGB and NIR (near infrared) imagery.

Imagery: We run each of our data transmission plans on imagery collected from these satellites for 15 days spread across three months (the first five days in June, July, and August 2021). We access the metadata of the imagery collected on these satellites using the Planet Developer API[2]. Each image is approximately 300 MB in size, and spans at least 24 km by 8 km distance on Earth depending on the satellite hardware and the variability in its altitude. In total, we collect data for nearly 6 million images. To the best of our knowledge, this is the largest evaluation involving any satellite dataset. Table 1 shows a summary of these statistics.

Ground Station: We model Planet's ground station architecture using publicly released information. Specifically, we simulate 12 ground stations carrying a total of 48 antennas [9, 12]. Fig. 8 shows the ground station locations.

Network Properties: We follow the radio architecture reported in [13] for simulating the satellite-ground station communication, achieving a bandwidth of up to 2 Gbps. Public information on available bandwidth for ground station-cloud backhaul links is scant. We have been in communication with multiple satellite operators. We use a combination of the anecdotal information we collected via these conversations, along with public domain information [38], to derive typical ground station-cloud bandwidths. We estimate the backhaul bandwidth values to be generally around 1 Gbps, but varying from 100Mbps to a few Gbps depending on the location. In our experiments, we vary ground station-cloud bandwidths.

5.2 Trace-driven Simulator

We evaluate Umbra in a discrete-event simulation. The simulator keeps track of both: i) internal status of the ground station, i.e., time, image queue, upload bandwidth, etc., and ii) satellites, i.e., time, position, captured images, etc. It simulates the system at a time granularity of 1 minute periods. This simulator is decoupled from Umbra's control plane in Section 4, allowing us to explore stale plans, failures, etc.

Simulator execution is fine-grained. During each time step, the simulator computes the bandwidth between each ground station-satellite pair, and then simulates the scheduling algorithm's execution plan on the waiting data at each node.

Simulating Orbital Motion and Bandwidths: We use TLEs obtained from Celestrak [29] to calculate the position, and velocity of satellites in orbit using the PyOrbital library[3]. The TLEs are periodically updated for accuracy—depending on the timestamps of the image being used, we retrieve the

[2]See https://developers.planet.com/quickstart/apis/

[3]https://pyorbital.readthedocs.io/en/latest/

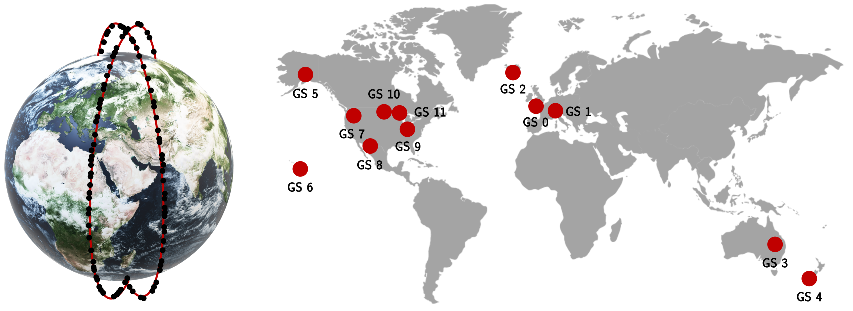


Figure 8: LEO Path & Ground Station Locations *(Left) The 2 orbits with 153 satellites for the Dove constellation. (Right) The position of 12 ground stations on Earth.*

# Satellites	153
# Ground Stations	12
Total Images	5,993,212
Image Size (mean)	300 MB
Total Data Volume	1798 TB
# Days	15

Table 1: Satellite Trace Details.

most up to date TLE information from Celestrak for our simulation. We predict satellite radio bandwidth by using the International Telecommunication Union (ITU) model [26–28] which takes as input satellite distance, elevation, azimuth, and local precipitation at the ground station. Weather information is pulled via the DarkSky weather API [10].

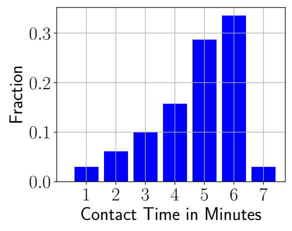
Hardware: We run our simulator and optimization framework on a SuperMicro SYS-4028GR-TR[4] server. To benchmark the time taken by our scheduler, we run the scheduler on a single core. It takes approximately 25 minutes to schedule satellite traffic for a 5-day run of the entire constellation. This time could be optimized further by leveraging parallelization, especially for computing the Hungarian matching (Section 3). However we do not explore this because we expect plans to be infrequently updated.

5.3 Baselines vs. Umbra

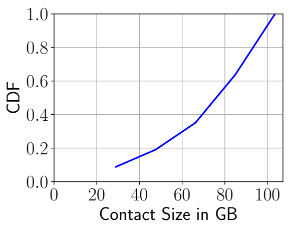
We compare Umbra against three baselines:

1. **Greedy:** This is the status quo: fast "greedy" transfers (Section 1) that fully utilize satellite-ground links. We use the ground station-satellite matching using past work [44].
2. **Withhold – Naive:** Inspired by public documentation [13] where satellites skip over-subscribed ground stations, we designed a simple withhold scheduling strategy. In this strategy, the satellite compares the current queue sizes at its current ground station contact and its next (expected) ground station contact. If the next ground station contact currently has a smaller queue than the current ground station, the satellite decides to withhold all of its data, and instead transmits it to the next ground station.
3. **Withhold – Smart:** We also design a more complex heuristic-based withholding approach in which a satellite transfers an amount of data inversely proportional to its current queue size at the ground stations. Namely, denote q_1, q_2 as the queue size for the current gs and the next contact gs, and denote V as the total volume of the cached data on satellite. Then, the satellite will transmit $V_0 = V\frac{q_2}{q_1+q_2}$ volume of data during the current contact.

[4]https://www.supermicro.com/en/products/system/4U/4028/SYS-4028GR-TR.cfm



(a) *Contact length*



(b) *Data Volume per Contact*

Figure 9: Satellite-ground station contact statistics.

Our key metrics are: (a) Throughput: How much data can a scheme transmit per day? and (b) End-to-end Latency: How long does it take for an image from the time of its capture to get to the cloud?

6 EXPERIMENTAL EVALUATION

We present our evaluation of Umbra below.

6.1 Satellite System Characteristics

Satellite-Ground Station Link Characteristics: Fig. 9a shows the distribution of the contact duration between satellite and ground stations. The contact time varies between 1 minutes and 7 minutes, with the mode at 6 minutes. Fig 9b plots the distribution of (maximum) data volume that can be downloaded during each contact. This value ranges from 10.37 GB to 103.48 GB, with a median 74.98 GB. These numbers are consistent with Planet's public data [13]. The large data volume leads to queue build up at the ground station.

Does increasing satellite bandwidth help? We simulate improved satellite radio hardware by doubling the bandwidths in our simulator for all satellite-ground station links and simulate the data download process using the greedy baseline. Fig. 10 shows the CDF of the latency. Counter-intuitively, the 90th percentile latency *increased by 22%* when

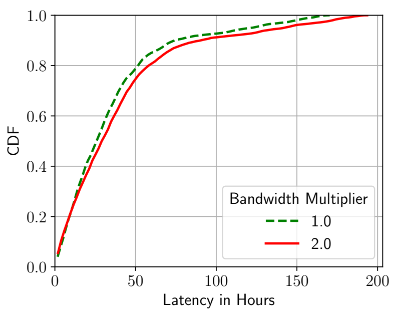


Figure 10: Increasing satellite bandwidth increases latency. *The CDF of end-to-end latency for images when the satellite bandwidth is doubled.*

satellite downlink bandwidth was doubled! This is due to the uneven queue buildups at a subset of ground stations, and points to the need for withhold scheduling.

6.2 End-to-end Performance

Throughput: We evaluate the throughput achieved on three traces spanning five days each across three different months. We plot results from only Day 2 onwards, to measure performance in steady state. We perform this experiment with different values of backhaul (ground station to cloud) bandwidth (1.2 Gbps, 1.5 Gbps, 1.8 Gbps).

Table 2 shows throughput values of Umbra, and the three baselines (Section 5.3). First, we observe that Umbra consistently reaches higher throughput than the alternatives. For the 1.2 Gbps backhaul, Umbra outperforms the greedy baseline by 13%, naive withhold by 31%, and smart withhold scheduling by 13% respectively. This is because Umbra equalizes queue sizes across different ground stations by moving traffic from over-subscribed ground stations to under-utilized ground stations and improves net utilization (and hence throughput). The naive withhold strategy achieves lower performance than Umbra because the former is more aggressive about withhold decisions (i.e., just looks at current queues and decides to skip). On the other hand, the smart heuristic-based withhold scheduling performs better than the other baselines, especially as bandwidth improves. Finally, as backhaul bandwidth improves from 1.2 Gbps to 1.8 Gbps, gains for Umbra over baselines decrease. This is expected because the queue sizes become smaller as the backhaul bandwidth improves.

Takeaways: We summarize two key takeaways from this result – (a) Our proposal for witthold scheduling is essential for efficient network utilization in satellite networks. Even a heuristic-based withhold scheduling outperforms greedy scheduling approaches used today. (b) However, withhold scheduling needs to be done intelligently to maximally realize potential gains. Therefore, Umbra's realization of withhold scheduling using time-expaneded networks outperforms other baselines.

Latency: Fig. 11 shows end-to-end latency for an image from its satellite capture to arrival at the cloud. For 1.2 Gbps backhaul, median latency of greedy (8.8 hours) is 42% higher than Umbra (6.2 hours). The latency of naive withholding is much higher at 13.7 hours, while that of smart withholding strategy is 8.9 hours. Tail latency improvement is even larger: at 1.2 Gbps backhaul, 90-th percentile latency of Umbra is 11.0 hours, vs. 38.7 hours (3.5 × worse) for greedy baseline, 66.5 hours (6 × worse) for naive withholding, and 37.98 (3.5 × worse) hours for smart withholding. The gain continues to hold at higher backhaul bandwidths. At 1.5 Gbps backhaul, 90-th percentile latency of greedy baseline is 19.3 hours, naive withholding is 60.9 hours, smart withholding is 20 hours , while Umbra's is only 8.3 hours.

Tail latency is critical for latency-sensitive applications, e.g., analyzing conflicts, natural disasters, etc. Furthermore, service providers generally have service level objectives (SLO) concerning P90 or even further tailed performance to give guarantee on the worst case of their service, in which case improving P90 is essential in increasing the efficiency of the whole system. Umbra's ability to avoid some data being stuck in long queues, and Umbra's consequent 3.5-6 × improvement in P90 latency, would dramatically reduce the time to act on insights from this data.

6.3 Inside Umbra

First, we look at withholding decisions made by Umbra. We evaluate how often Umbra withholds data and by how much? For each scheduled link between a satellite and ground station, we measure the fraction of data withheld and plot the CDF in Fig. 12(a). The fraction ranges from 0 to 1, with 0 corresponding to no data being withheld. We observe a majority of decisions are binary withholding decisions, i.e., either all the data is withheld or none is. Backhaul bandwidth does not significantly affect these. Nevertheless, compared to our withhold scheduling heuristics(Section 5.3), the *selection* of which links to withhold on, is more intelligent in Umbra, causing it to have better performance.

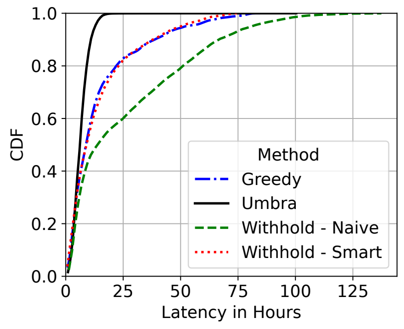
Finally, Figs. 3 and 12(b) show queue size (sampled) at two typical ground stations. With the greedy baseline, queue sizes vary widely (and wildly)—while one ground station sees ever-expanding queues, others stay idle, thus wasting resources. Umbra's intelligent withholding scheme balances loads across ground stations and stabilizes queue sizes.

6.4 Heterogeneity in Ground Stations

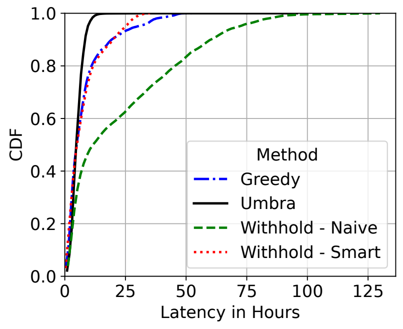
We relax our homogeneity assumption on ground stations and afford a subset of them higher backhaul bandwidth to the cloud. We select a random 50% of the ground stations

Trace	Throughput (TB of data)								
	1.2Gbps			1.5Gbps			1.8Gbps		
	June	July	August	June	July	August	June	July	August
Greedy	392.3 (12.35%)	393.2 (12.02%)	381.8 (10.84%)	428.1 (6.06%)	427.4 (5.92%)	414.3 (4.93%)	443.9 (2.33%)	444.1 (1.81%)	430.1 (1.06%)
Withhold – Naive	335.6 (25.02%)	341.3 (23.63%)	327.7 (23.47%)	343.5 (24.62%)	347.5 (23.51%)	334.3 (23.29%)	351.7 (22.62%)	357.8 (20.89%)	344.9 (20.66%)
Withhold – Smart	395.7 (11.60%)	394.8 (11.66%)	384.5 (10.21%)	436.1 (4.30%)	434.1 (4.45%)	420.8 (3.44%)	454.1 (0.09%)	**455.9** (-0.80%)	**435.5** (-0.18%)
Umbra	**447.6** (0.00%)	**446.9** (0.00%)	**428.2** (0.00%)	**455.7** (0.00%)	**454.3** (0.00%)	**435.8** (0.00%)	**454.5** (0.00%)	452.3 (0.00%)	434.7 (0.00%)

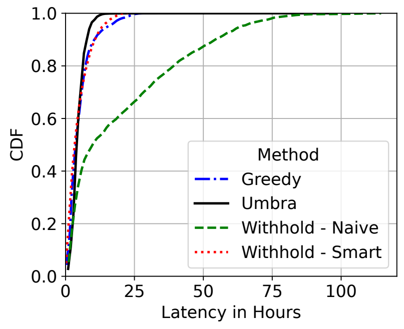
Table 2: Data throughput. *Each row is measured over a 3-day period in the month. Parentheses show % worse than Umbra.*



(a) *1.2 Gbps Backhaul Bandwidth*

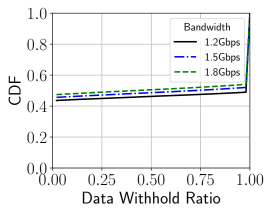


(b) *1.5 Gbps Backhaul*

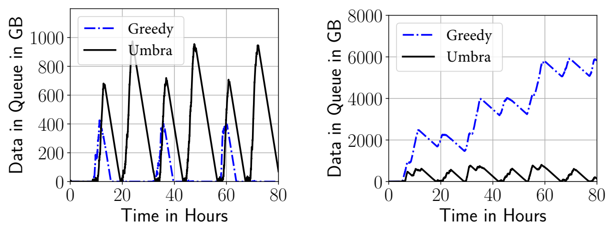


(c) *1.8 Gbps Backhaul*

Figure 11: End-to-end Latency. *For different backhaul bandwidths (ground station to cloud).*



(a) *Data Withhold Rate*



(b) *Queue Size for 2 typical Ground Stations*

Figure 12: Analyzing Umbra's Performance.

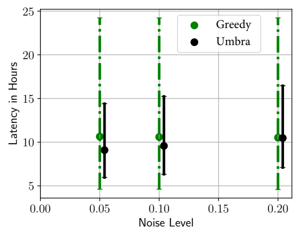
	Mean Throughput (Std dev)
Greedy	435.99 (8.36)
Umbra	571.80 (0.10)

Table 4: Throughput (in TB) for heterogenous backhaul.

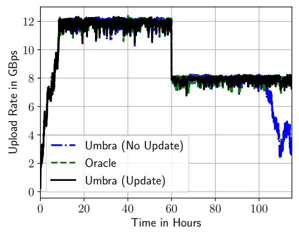
to have a 2 Gbps backhaul bandwidth.Table. 4 shows results over 3 independent trials. We observe that Umbra achieves a throughput of 571.8 TB (std dev 0.10) across 5 days, with a 31% improvement over the greedy baseline. The standard deviation is much lower for Umbra, showing it can load balance even across heterogeneous ground stations.

6.5 Many Distributed Ground Stations

Recent work [43, 44] has proposed distributed ground station architectures where hundreds of tiny low-complexity ground stations outperform the efficient multi-million ground stations deployed today. We tested Umbra in this setting, using the same image trace data as before. We sampled 200 ground stations from the Satnogs database [33], which is an open-source network of amateur ground stations operated by independent enthusiasts. This methodology is similar to [43, 44].



(a) *Bandwidth Estimation Error*



(b) *Handling Crashes*

Figure 13: Robustness Analysis. *In (a): points perturbed horizontally for clarity, and error bars are 25-75 percentile.*

We set their backhaul bandwidth via a Poisson distribution with λ = 75 MBps . The total backhaul capacity is 15GBps, which is 8X the sufficient amount to fully transmit all data

	Mean Throughput (Std dev)
Greedy	425.54 (5.53)
Withhold – Naive	443.94 (8.38)
Withhold – Smart	430.97 (5.56)
Umbra	445.01 (8.80)

Table 5: Throughput (in TB) for distributed ground stations.

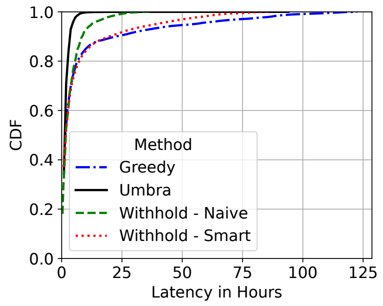


Figure 14: Latency cdf for distributed ground stations

generated by our satellite constellation. We sampled 3 sets of different bandwidth configurations for the ground stations.

Table 5 summarizes throughput during Days 2-4 for all approaches. The improvement on average throughput by Umbra is not significant, which is expected when we provision 8X the required total bandwidth for the ground stations. However, from Fig 14, we see that Umbra reduces the P90 latency greatly (2.5X against naive withholding heuristic, the second-best performing algorithm) even with abundant backhaul bandwidth. This shows that the UQE problem is even harder to solve by just over-provisioning infrastructure, when the EO satellite systems evolve into the new distributed ground station scenario in the future.

6.6 Robustness to Errors & Failures

Bandwidth Estimation Errors: Umbra may receive inaccurate bandwidth estimates due to sudden change in weather, interference at the ground station, etc. We evaluate the robustness of Umbra's solution. We add random noise to the downlink bandwidth in the simulator at run time, i.e. *after* the scheduler has calculated the data transmission plan. Fig. 13a shows that noise moderately degrades median and tail latency for Umbra. A noise level of x means each link's actual bandwidth is chosen uniformly in the $\pm(100 \cdot x)\%$ range from the predicted bandwidth. Concretely, increasing noise factor from 0.05 to 0.2 degrades Umbra's median latency by 15% and P90 latency by 8.4%. Past work has shown the ability to accurately predict data rates within 6% error [44], so we expect Umbra to perform well in real-world settings.

Hardware Failures: Ground stations fail occasionally due to power or network outages. To evaluate Umbra, we compare three scenarios: 1) Umbra (No Update), which never updates the plan even after failure(s), 2) Umbra (Update), which calculates a new plan after failure, and disseminates it, and 3) Oracle, which is prescient about the failure and generates a plan ahead of the failure point, and initiates the new plan right at the failure point.

We fail 33% of ground stations instantly at $t = 60$ hours. Fig. 13b shows that right after the failure, both versions of Umbra (Update and No Update) keep throughput as high as the Oracle. That is, the failure does not cause a massive drop in throughput beyond what is expected. Because one-third of the stations fail, total throughput drops to 66% of the original throughput. The stable throughput lasts for only about 40 hours in Umbra (No Update) and then starts to degrade rapidly ($t = 100$ hours and onwards), showing that plans become stale after less than 2 days. Umbra (Update version) has a plan that stays stable as long as the Oracle, giving ground station crews a longer time to repair the failure(s).

7 RELATED WORK

Satellite Networking: Our work follows recent work [5, 6, 11, 21–23, 43, 44] in the satellite networking domain, including: edge computing on the satellites [6, 11], ground station architectures [22, 43, 44], security of satellite networks [19], inter-satellite links [21, 23], network benchmarking [21], etc. For satellite-ground station traffic, past work [11, 43, 44] treats the satellite-ground station contact as the bottleneck and schedules traffic greedily, i.e., transfers as much data as possible in every contact. Unlike past work, Umbra takes a withhold scheduling approach, where all or part of the data can be withheld for subsequent contacts between satellite and Earth. We are also the first ones to focus on the ground station-cloud bandwidth as an emerging bottleneck given the rapid advances in satellite-ground station radio speeds [13].

Time Expanded Networks: Scheduling dynamic network flows is well-studied [16, 17, 40]. Flow scheduling using time expanded networks has been explored in the context of scheduling traffic in the internet [16] and sneakernets [7]. Recently, some research has looked to formulate time expanded networks in the satellite context [39, 45, 47]. This work focuses on the task of relaying traffic through a network of interconnected satellites and models it from an energy [39], compute [45], and network perspective [47]. Our modeling of this problem is unique because we are the first to model the end-to-end data transfer from *large scale* satellite constellations to the cloud as a time expanded network. This modelling is challenging in its scale – hundreds of satellites, tens of ground station antennas, and time varying links. In addition, we are the first to leverage time expanded networks for load balancing. Finally, our work reveals new insights like how ground stations can suffer from load imbalance and how we can frame a new withhold scheduling approach by performing analysis on this time expanded graph.

Traffic Scheduling Algorithms: There has been a large body of work on scheduling algorithms in other contexts such as data centers. Our observations and results in satellite traffic are analogous to the delay scheduling work for cluster scheduling [50]. Delay scheduling observed that instead of scheduling jobs to the first available node on a cluster, it is advantageous to wait for a small amount of time and find a node that has favorable features (e.g., data locality). This improves overall system performance. In spite of similarity, Umbra's setting and techniques are different— network traffic instead of cluster scheduling. Furthermore, the scale of our problem is enormous and requires network flow formulations & optimizations.

Disruption Tolerant Networks (DTN): DTN is a class of networks experiencing intermittent connections between endpoints. There is a rich literature in routing and traffic engineering in DTN [30], and in applications of TEN in DTN such as running shortest path algorithm for routing [35] or maximum flow algorithm for optimizing throughput [46]. Traffic in DTNs flow in a "Store, Carry and Forward" manner [34] similar to LEO satellite systems, but Umbra is different from the works on DTN in terms of the scale of the system and the objective being optimized. First, Umbra deals with earth imaging satellites which transmit massive images rather than short messages, where the bottleneck is not only intermittent connections but also limited networking bandwidth in the system. Second, while previous work on DTN mostly deals with a multi-agent network and focuses on optimizing for one device in the network, Umbra works in a centralized network where all the ground stations and satellites are owned by the same entity and they can collaborate to optimize for a global objective. Finally, while the DTN works try to optimize either latency or throughput using TENs, Umbra is, to our best knowledge, the first algorithm that can optimize the 2 objectives simultaneously.

8 CONCLUDING DISCUSSION

Delivering data from multiple constellations: Recent years have seen emergence of the Ground-station-as-a-service (GSaaS) model by many commercial entities [3, 31, 36]. These companies allow constellation operators to rent ground station time by the minute to schedule data download. We expect withhold scheduling to be effective in such contexts. However, the measurement of network queue size needs to be indirect, as the queue size at the ground station may not be visible to satellite constellation operators. We note that UQE will get worse if the backhaul bandwidth of ground stations decrease, as shown in our evaluation. Therefore, we expect Umbra to be more efficient under the GSaaS scenario.

Inter-station and inter-satellite links: Our evaluation assumed the absence of these links because they are not common in today's deployments. However, both these kinds of links can be added to our graph and our TEN-based solution (Section 3) would still generate a solution. For instance, a satellite could route data through another satellite using an inter-satellite link, especially when the latter satellite is connected to a low-queue station.

What did not work? To make withholding decisions, we experimented with iterating between: (a) identifying the best matching between a satellite and ground station at a given time instance, and (b) computing the max flow in the time expanded network. In principle, this was reasonable because if a satellite withholds data from a ground station, a different satellite may want to use this ground station (this decision can be made in the next iteration). However, we noticed that the scheduling objective (e.g. throughput) showed little improvement beyond more than one iteration, and only increased computation cost. We believe that this is because most Dove satellites follow each other in one of two orbits and come in contact with the same ground station sequences. So, withholding decisions are similar for different satellites in proximity with a crowded ground station.

Ground Station Backhaul Bandwidths: Our work explores the ground station-cloud backhaul link as the bottleneck in satellite data transfers. Over the next few years, we expect three factors to increase the demand for backhaul bandwidth even more: (i) satellite-ground bandwidths continue to improve to 5 Gbps and beyond ([41, 49]), (ii) operators will amortize the cost of site acquisition and licensing by deploying more antennas at the same ground station site, and (iii) increased computation demands at the ground stations (for pre-processing), will cause queuing delays towards the cloud to continue increasing, even if backhaul bandwidths keep improving.

Umbra and Resource Utilization Efficiency While provisioning higher backhaul bandwidth can always reduce queues on the ground stations and reduce the latency, it will not solve the problem of the low resources utilization. We note that the constellation sizes continue to grow over time. Umbra can support more satellite with the current infrastructure, as well as use less bandwidth to support emerging constellations. Moreover, in the GSaaS setting, the ground station capacity and bandwidth needs to be rented from the provider, and increasing its efficiency has economic benefits.

Acknowledgments – We thank the reviewers and our anonymous shepherd for their insightful feedback. This work was funded partially by Cisco, Microsoft, and National Science Foundation grant CNS 1908888. We are grateful to Kiruthika Devaraj from Planet Inc. for many insightful discussions. This work utilizes resources offered by the HAL cluster at UIUC, supported by NSF MRI grant #1725729.

REFERENCES

[1] Spire Global Inc. https://spire.com/.
[2] Waiting Time Paradox. https://en.wikipedia.org/wiki/Residual_time.
[3] Amazon Inc. AWS Ground Station . https://aws.amazon.com/ground-station/.
[4] Robin George Andrews. Solar storm destroys 40 new SpaceX satellites in orbit. New York Times. https://www.nytimes.com/2022/02/09/science/spacex-satellites-storm.html, 2021.
[5] Debopam Bhattacherjee, Waqar Aqeel, Ilker Nadi Bozkurt, Anthony Aguirre, Balakrishnan Chandrasekaran, P. Brighten Godfrey, Gregory Laughlin, Bruce Maggs, and Ankit Singla. Gearing up for the 21st century space race. In *ACM Workshop on Hot Topics in Networks*, ACM HotNets, 2018.
[6] Debopam Bhattacherjee, Simon Kassing, Melissa Licciardello, and Ankit Singla. In-orbit computing: An outlandish thought experiment? In *ACM Workshop on Hot Topics in Networks*, ACM HotNets, 2020.
[7] Brian Cho and Indranil Gupta. New algorithms for planning bulk transfer via internet and shipping networks. In *International Conference on Distributed Computing Systems*. IEEE Computer Society, 2010.
[8] Christine Lunsford. California's caldor fire seen from space in harrowing satellite images. https://www.space.com/caldor-fire-satellite-images-gallery, 2021.
[9] Kyle Colton, Joseph Breu, Bryan Klofas, Sydney Marler, Chad Norgan, and Matthew Waldram. Merging Diverse Architectures for Multi-Mission Support. In *Small Satellite Conference*, 2020.
[10] Dark Sky. Dark Sky Weather API. https://darksky.net/dev.
[11] Bradley Denby and Brandon Lucia. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *ACM ASPLOS*, 2020.
[12] Kiruthika Devaraj, Ryan Kingsbury, Matt Ligon, Joseph Breu, Vivek Vittaldev, Bryan Klofas, Patrick Yeon, and Kyle Colton. Dove High Speed Downlink System. In *Small Satellite Conference*, 2017.
[13] Kiruthika Devaraj, Matt Ligon, Eric Blossom, Joseph Breu, Bryan Klofas, Kyle Colton, and Ryan Kingsbury. Planet High Speed Radio: Crossing Gbps from a 3U Cubesat. In *Small Satellite Conference*, 2019.
[14] Anna Escher. Inside Planet Labs' new satellite manufacturing site. TechCrunch. https://techcrunch.com/2018/09/14/inside-planet-labs-new-satellite-manufacturing-site/, 2018.
[15] European Space Agency. Sentinel 2: Core Ground Segment Design . https://sentinel.esa.int/web/sentinel/missions/sentinel-2/ground-segment/core-ground-segment.
[16] Lisa Fleischer and Martin Skutella. Quickest flows over time. *SIAM Journal on Computing*, 2007.
[17] L. R. Ford and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 1958.
[18] Jonathan Franklin. Satellite images show the aftermath of tonga volcano's eruption. https://www.npr.org/sections/pictureshow/2022/01/26/1075621526/satellite-images-show-the-aftermath-of-tonga-volcanos-eruption, 2022.
[19] Giacomo Giuliari, Tommaso Ciussani, Adrian Perrig, and Ankit Singla. ICARUS: Attacking low earth orbit satellite networks. In *USENIX Annual Technical Conference (USENIX ATC 21)*, 2021.
[20] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, 1988.
[21] Mark Handley. Delay is not an option: Low latency routing in space. In *ACM Workshop on Hot Topics in Networks*, ACM HotNets, 2018.
[22] Mark Handley. Using ground relays for low-latency wide-area routing in megaconstellations. In *ACM Workshop on Hot Topics in Networks*, ACM HotNets, 2019.
[23] Yannick Hauri, Debopam Bhattacherjee, Manuel Grossmann, and Ankit Singla. "Internet from Space" without inter-satellite links. In *ACM Workshop on Hot Topics in Networks*, ACM HotNets, 2020.
[24] Felix R. Hoots and Ronald L. Roehrich. Models for Propagation of NORAD Element Sets. Technical report, Aerospace Defense Command, United States Airforce, 1980.
[25] Planet Inc. Planet labs public orbital ephemerides. https://ephemerides.planet-labs.com/.
[26] International Telecommunications Union. ITU P.838: Specific attenuation model for rain for use in prediction methods. Technical report, 2019.
[27] International Telecommunications Union. ITU P.839 : Rain height model for prediction methods . Technical report, 2019.
[28] International Telecommunications Union. ITU P.840: Attenuation due to clouds and fog. Technical report, 2019.
[29] T. S. Kelso. Celestrak. https://celestrak.com/.
[30] Maurice J Khabbaz, Chadi M Assi, and Wissam F Fawaz. Disruption-tolerant networking: A comprehensive survey on recent developments and persisting challenges. *IEEE Communications Surveys & Tutorials*, 14(2):607–640, 2011.
[31] Kongsberg Satellite Services. Ground Station Services. https://www.ksat.no/services/ground-station-services/.
[32] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.
[33] Libre Space Foundation. SatNOGS: An Open Source Ground Station and Network. https://satnogs.org/, 2014.
[34] Alex McMahon and Stephen Farrell. Delay-and disruption-tolerant networking. *IEEE Internet Computing*, 13(6):82–87, 2009.
[35] Shashidhar Merugu, Mostafa Hamed Ammar, and Ellen W Zegura. Routing in space and time in networks with predictable mobility. Technical report, Georgia Institute of Technology, 2004.
[36] Microsoft. Azure Orbital. https://azure.microsoft.com/en-us/services/orbital/.
[37] Planet Inc. Dove Satellite Constellation. https://www.planet.com/our-constellations/.
[38] Konsberg Satellite Services. Ksat has installed antenna number 100 at svalbard ground station. https://www.kongsberg.com/newsandmedia/news-archive/2021/ksat-has-installed-antenna-number-100-at-svalbard-ground-station/.
[39] Keyi Shi, Hongyan Li, and Long Suo. Temporal graph based energy-limited max-flow routing over satellite networks. In *2021 IFIP Networking Conference (IFIP Networking)*, 2021.
[40] Martin Skutella. *An Introduction to Network Flows over Time*. Springer Berlin Heidelberg, 2009.
[41] Mark Storm, He Cao, Slava Litvinovitch, Kent Puffenberger, Jeremy Young, Dave Pachowicz, Timothy Deely, Shantanu Gupta, and Michael Albert. Cubesat laser communications transceiver for multi-gbps downlink. In *Small Satellite Conference*, 2017.
[42] Union of Concerned Scientists. UCS Satellite Database. https://www.ucsusa.org/resources/satellite-database, 2020.
[43] Deepak Vasisht and Ranveer Chandra. A distributed and hybrid ground station network for low earth orbit satellites. In *ACM HotNets*, 2020.
[44] Deepak Vasisht, Jayanth Shenoy, and Ranveer Chandra. L2d2: Low latency distributed downlink for low earth orbit satellites. In *ACM SIGCOMM*, 2021.
[45] Chen Wang, Zhiyuan Ren, Wenchi Cheng, Shuya Zheng, and Hailin Zhang. Time-expanded graph-based dispersed computing policy for leo space satellite computing. In *IEEE Wireless Communications and Networking Conference (WCNC)*, 2021.
[46] Peng Wang, Xiushe Zhang, Shun Zhang, Hongyan Li, and Tao Zhang. Time-expanded graph-based resource allocation over the satellite networks. *IEEE Wireless Communications Letters*, 8(2):360–363, 2018.

[47] Peng Wang, Xiushe Zhang, Shun Zhang, Hongyan Li, and Tao Zhang. Time-expanded graph-based resource allocation over the satellite networks. *IEEE Wireless Communications Letters*, 2019.

[48] Haley Willis. Newly released satellite images show the effects of war on ukraine's civilians. https://www.nytimes.com/live/2022/03/03/world/russia-ukraine?smid=url-share#newly-released-satellite-images-show-the-effects-of-war-on-ukraines-civilians, 2022.

[49] Yen Wong, Scott Schaire, Steve Bundick, Peter Fetterer, Trish Perrotto, and Peter Celeste. NASA NEN DVB-S2 demonstration testing for enhancing higher data rates for CubeSat/small satellite missions at X-band and Ka-band. In *Small Satellite Conference*, 2020.

[50] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, and Ion Stoica. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In *Proceedings of the 5th European Conference on Computer Systems*, EuroSys '10, 2010.