On Emptying Small Satellite Networks with In-Network Data Aggregation

Luyao Wang[1] and Kwan-Wu Chin[2] *

Abstract

Recently, small satellites have attracted much attention in both academia and industry. These satellites can form a swarm and communicate via Inter-Satellite Links (ISLs) and help each other download data to one or more Ground Stations (GSs). A key challenge is that each satellite has a short contact period with different GSs over time. Henceforth, this paper considers the following objective: minimize the download time of a given amount of data from each satellite to ground stations. The main challenging issues are that satellites have different energy and buffer constraints. We model the problem as a Mixed Integer Linear Program (MILP). Its objective is to minimize the total number of time slots taken to download a fixed amount of data from each satellite. Its key decision variables relate to (i) routing, (ii) link scheduling, and (iii) data aggregation rate in each time slot. Our results show that in-network data aggregation is able to significantly reduce the total download time by 52% *to* 84% *when the data aggregation rate is between* 10% *to* 50%*.*

1. Introduction

With decreasing manufacturing and launch costs, Low Earth Orbiting (LEO) small satellites are now attractive to both academia and industry [1]. According to the weight of each satellite, LEO small satellites can be categorized as a mini-satellite (500-100 Kg), a micro-satellite (100-10 Kg), a nano-satellite or called a cube-satellite (10-1 Kg) as well as a Femto and Pico-satellite (<1 Kg) [1]. Compared to other satellites operating at higher altitudes, LEO small satellites have the following advantages: small size, light weight, low power consumption, high flexibility, and low communication latency. Also, they can form a constellation or operate in a swarm where communication is facilitated by Inter-Satellite Links (ISLs) [2]. Advantageously, small satellite networks provide low cost solutions for diverse and complex space missions, e.g., weather forecast, environment monitoring, broadband communications and target tracking [3].

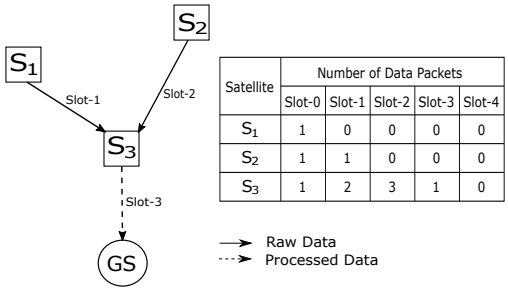
The main objective of any missions is to collect data and download them to Ground Stations (GSs). Here, we assume these GSs are connected by the Internet. This means all GSs can be treated as equal where a satellite can download its data to any GSs within range. However, considering the high orbiting speed of small satellites, each of them only has a very limited contact time with each GS. Consequently, the data stored on satellites cannot be fully downloaded in most cases. The works in [4] and [5] showed that ISLs can help download collected data to a GS. However, the obtained throughput gain is small due to limited download bandwidth. For example, the CubeSat in [6] is only equipped with a radio that has a downlink rate of 2400 bps.

An approach to reduce the amount of data to be downloaded is to employ on-orbit data processing [7] [8]. Specifically, satellites can process/compress/aggregate stored data whilst ensuring a minimum Quality of Information (QoI). This approach not only improves the efficiency of data download, but also reduces the amount of buffered data, which allows for more data to be collected by each satellite. Thus in one time slot, a satellite has the following options: 1) it downloads unprocessed data to a GS directly if there is sufficient download bandwidth; 2) it first aggregates collected data to reduce its size before downloading; 3) it transmits unprocessed data to a neighbor satellite for the purpose of compression or downloading to a GS; 4) it first compresses its collected data before transmitting it to a neighbor for further aggregation or downloading to a GS; 5) it first receives data from neighbors, aggregates the received data and downloads the processed data to a GS. It is worth noting that in-network data processing costs a non-negligible amount of energy [8]. Indeed, as small satellites have limited buffer size and battery capacity, there is a trade-off between in-network data processing and data download.

Figure 1 shows an example of data downloading with in-network data aggregation in a small satellite net-

*Author Wang is with Beijing Advanced Innovation Center for Future Internet Technology Beijing University of Technology, Beijing, China (email: luyaowang1990@163.com[1]). Author Chin is with the School of Electrical, Computer, and Telecommunications Engineering, University of Wollongong, Australia (email: kwanwu@uow.edu.au[2]).

work. The topology has three satellites, namely S_1, S_2 and S_3. In this example, only satellite S_3 is able to establish a download link to ground station *GS*. Initially, each satellite has one data packet to download, and we assume these three packets can be compressed into one packet. The table in Figure 1 shows the number of data packets at each satellite in the optimal downloading scheme over five time slots. Satellite S_1 and S_2 transmit their data packet to S_3 in time Slot-1 and Slot-2 respectively. Then S_3 compresses three data packets into one packet and downloads it to *GS* in Slot-3. We see that this example requires a total of three time slots to download all packets.



Satellite	Number of Data Packets				
	Slot-0	Slot-1	Slot-2	Slot-3	Slot-4
S_1	1	0	0	0	0
S_2	1	1	0	0	0
S_3	1	2	3	1	0

Figure 1. An example of data downloading with in-network data aggregation. The column labeled Slot-0 indicates each satellite has one packet. In Slot-1, satellite S_1 transmits its packet to satellite S_3; this means satellites S_1 and S_3 have zero and two packets, respectively. Satellite S_2 transmits its data to S_3 in Slot-2. Consequently, satellite S_3 has total three packets. In Slot-3, satellite S_3 compresses three packets into one packet and downloads it to the ground station *GS*. The data download process ends in Slot-4.

When downloading data, we have to consider the following issues: 1) *time-varying topology*. This is due to the high orbiting speed of small satellites. In the foregone example, in Slot-3, if S_3 loses its connection with *GS* and satellite S_2 establishes communication with *GS* instead, then the compressed data will be transmitted to S_2 which incurs another time slot, 2) *limited battery capacity*. Considering the size of small satellites, the battery capacity of each is very limited. Referring to Figure 1, after receiving data from S_1 and S_2, if satellite S_3 does not have sufficient energy, it cannot compress or download data until it has harvested sufficient energy in the following time slots. Thus, the total download time will be much longer than the optimal result, 3) *limited buffer size*. In Figure 1, even when satellite S_3 has sufficient energy and a download connection with *GS*, it cannot receive data from its neighbors if it has insufficient buffer space. As we will see later, any of the aforementioned issues will cost more time slots.

In this paper, we consider the joint problem of routing, link scheduling and data aggregation subject to buffer and energy constraints. Our objective is to download a given amount of data from each satellite to ground stations within the shortest time period. As explained in Section 2, this work is the first that considers in-network data aggregation in order to speed up downloads in satellite networks. We formulate the aforementioned problem as a Mixed Integer Linear Programming (MILP). Its objective is to minimize the total number of time slots required to drain data from a satellite network; see Section 3 and 4. Numerical results in Section 5 show that introducing in-network data aggregation is able to significantly reduce the total download time by 52% to 84% as compared to experiments without data aggregation. In addition, we also consider minimizing the total compressed data and consumed energy given a limited download period. The simulation results and analysis in Section 5 show that the total compressed data and consumed energy decrease with increasing download time.

2. Related Works

The authors in [4] consider joint routing and link scheduling to achieve the maximum data download throughput to one GS by using ISLs to offload data among LEO satellites. Based on [4], the authors proposed another iterative data offloading algorithm in [5]. This algorithm constructs a bipartite graph and finds the maximal matching in each time slot to offload data from nodes with more data to nodes with spare time. The work in [9] focuses on energy-efficient data download from satellite swarms to ground stations. The authors formulate and solve the problem using Lyapunov optimization techniques [10]. However, none of the foregone works consider data aggregation nor aim to empty a given amount of data from satellites.

Data aggregation has been considered in Wireless Sensor Networks (WSNs) or wireless ad-hoc networks. In [11], the authors aim to minimize the total amount of uploaded sensory data by using data aggregation among mobile devices in crowd-sensing systems. The work in [12] aims to minimize the total energy cost for both communication and in-network computation in wireless ad-hoc networks while satisfying QoI requirement. In order to determine the optimal data aggregation rate, the authors formulated the problem as a Non-Linear Program (NLP) and solves its dual. Nevertheless, both works in [11] and [12] do not consider the energy harvesting capability of nodes.

The authors in [13] address the joint problem of sensing, link scheduling, routing and in-network data processing for Energy Harvesting Wireless Sensor Networks (EH-WSNs). They solve the problem using Lyapunov optimization [10]. In [14], the authors propose a data aggregation scheme that considers the residual energy of nodes. However, these two works assume a fixed network topology. We, on the other hand, considers nodes with neighbors that change over time. Moreover, the work in [13] and [14] do not aim to empty a given amount of data within the shortest time period.

We emphasize that data aggregation has never been considered in small satellite networks. Conventional data aggregation and transmission approaches that are developed for stationary networks cannot be applied to dynamic satellite networks directly. This is because they required a fixed topology. Thus, our work is the first that introduces in-network data aggregation to small satellite networks in order to empty the network in the shortest possible time. Critically, our aim is to empty a given amount of data from each satellite to one or more GSs. We believe this aim is novel.

3. Network Model

We assume time is divided into time slots and indexed by $t \in \mathbb{N}$. The network topology varies over time but predictable; this is reasonable as an operator knows the orbit and speed of deployed satellites [2]. Let S^t be the set of satellites and G^t is the set of ground stations at slot t. The network topology is fixed in each time slot but may vary over different slots. Let V^t be the set of nodes at time t; here, *nodes* refer to both satellites and ground stations. We write $N_u^t \subseteq V^t$ as the set of neighbors of node u in time slot t, where $u \in V^t$. The set of links between nodes in V^t is denoted as E^t; this set contains both ISLs and links between satellites and ground stations. Hence, the network in slot t can be modeled as a graph $G(V^t, E^t)$. In the sequel, we will use $L^t \subseteq E^t$ to denote the set of ISLs at time t and $D^t \subseteq E^t$ the set of downlinks between satellites and ground stations.

All satellites have identical hardware components and operate in a swarm. Without loss of generality, we assume each satellite is a CubeSat [1]. Satellites are equipped with an adaptive array. This allows a satellite to beamform as well as null interference to/from neighboring satellites via appropriate antenna weights [15]. In each time slot, each satellite can only transmit to or receive from at most one neighboring satellite.

Each satellite has a radio that operates over the Very High Frequency (VHF) band (30 MHz) and is used to download data to GSs. Satellites also have a radio that operates over the S-band (2.45 GHz) and this radio is used for communication with other satellites [1]. Satellites can only pair with one GS at a time to ensure there is no collision between downlinks. Note that given the different operating band of ISLs and downlinks, a node can communicate with a neighbor satellite whilst downloading data to a stationary GS. We assume GSs have unlimited energy and storage, and can communicate with each other via the Internet.

3.1. Communication Model

Active links must not interfere. Let $\mathbf{A^t}$ be a $|E^t| \times M$ matrix that contains a collection of transmission sets, where each column represents a set of non-conflicting links. Each column of $\mathbf{A^t}$ forms an independent set. Specifically, we write the j-th column as $\mathbf{A_j^t}$. Each element of column j is denoted as $a^t_{(u,v),j} \in \{0,1\}$, where a value of one indicates link (u,v) is active in transmission set $\mathbf{A_j^t}$. For example, if we have a column $[1010]^T$ corresponding to four links in a topology, then the first and third links can be activated together.

We generate transmission sets or columns of matrix $\mathbf{A^t}$ by employing the following heuristic: (1) define and initialize two node sets: Set1 and Set2; (2) all nodes are included in Set1, and Set2 is initially empty; (3) for each node in Set1, move it into Set2 if an edge can be established from Set1 to Set2 while ensuring no two ISLs or no two download links share an endpoint; (4) extract paired nodes or edges between Set1 and Set2 and construct a transmission set; (5) remove the edges and endpoints in step (4) and repeat from step (3) to (5) to generate the next transmission set until Set1 is empty. Note, to generate a different sequence of transmission sets, we can use a different node ordering and repeat the above steps. The total number of transmission sets generated for each slot is denoted as M. Notice that we do not generate all possible transmission sets due to computational complexity.

Let $x_j^t \in [0,1]$ denote the active time of the transmission set $\mathbf{A_j^t}$. A TDMA-based link schedule in time slot t consists one or more transmission sets with non-zero active time. This can then be represented as non-zero elements of the column vector $\mathbf{X^t} = [x_1^t, x_2^t, \ldots, x_M^t]^T$. For example, if transmission sets A_1^t and A_2^t are activated in the first and second half of slot t, then we have $\mathbf{X^t} = [0.5, 0.5, 0, \ldots, 0]^T$. This means all links in A_1^t (or A_2^t) will be activated for 0.5 seconds. The schedule length in every slot is constrained by,

$$\sum_{j=1}^{M} x_j^t = 1 \tag{1}$$

We use $c_{(u,v)}$ to represent the capacity of link (u,v).

The corresponding link rate at time t can be calculated by $c_{(u,v)} \sum_{j=1}^{M} a^t_{(u,v),j} \times x^t_j$. Let $f^t_{(u,v)}$ denote the amount of data transmitted through link (u,v) at slot t. The value of $f^t_{(u,v)}$ is constrained by,

$$0 \leq f^t_{(u,v)} \leq c_{(u,v)} \sum_{j=1}^{M} a^t_{(u,v),j} \times x^t_j, \quad \forall (u,v) \in E^t \quad (2)$$

Recall that N^t_u denotes the neighbors of satellite u at slot t. Thus, the total outgoing and incoming data of node u at slot t can be calculated as follows,

$$f^{out,t}_u = \sum_{v \in N^t_u} f^t_{(u,v)}, \quad \forall u \in S^t, \forall v \in V^t, u \neq v \quad (3)$$

$$f^{in,t}_u = \sum_{v \in N^t_u} f^t_{(v,u)}, \quad \forall u \in V^t, \forall v \in S^t, u \neq v \quad (4)$$

3.2. Storage and On-Board Processing

Each satellite has storage to store raw and processed data; e.g., images. We use $Q^t_u \geq 0$ to denote the amount of data stored at node $u \in S^t$ at time slot t. Each satellite has a storage capacity of Q_{max}. We assume at time $t = 1$, each satellite has identified $Q^1_u \in [0, Q_{max}]$ amount of data to be sent to a ground station. Our goal is to download $\sum_{u \in S^t} Q^1_u$ amount of data to GSs. Consequently, we only consider forwarding the data in Q^1_u to a ground station directly or via other satellites. If new data is collected after time $t = 1$, we assume they are scheduled for downloading after all data in Q^1_u of each satellite $u \in S$ has been downloaded to a ground station. This is reasonable as an operator may want to schedule data downloads after satellites have collected the required data. Thus, in the sequel, we ignore new data arrivals.

Each satellite node is equipped with On-Board Processing (OBP) capability [7]. Hence, a node can carry out data aggregation and compression. Let $\delta \in [\delta_{min}, 1]$ represent the aggregation rate, which is defined as the ratio between the the amount of stored data before and after OBP. If the value of δ is equal to one, the data in storage is not compressed. The value of $\delta_{min} > 0$ indicates the compression limit which is varied according to different type of data. To model the portion of aggregated and compressed data that arrives at ground stations, we associate each node u with a virtual sink u'. Let the set of virtual sinks be S', and the amount of aggregated or compressed data by node u is denoted by $f^t_{u'}$. Specifically, the value of $f^t_{u'}$ equals $f^t_{u'} = (1-\delta)Q^t_u \geq 0$. To ensure a node u does not aggregate or compress more data than its compression limit, we set the capacity of its virtual link (u,u') as follows,

$$0 \leq f^t_{u'} \leq (1-\delta_{min})Q^t_u, \quad \forall u \in S^t \quad (5)$$

Notice that the value of $f^t_{u'}$ is always less than Q^t_u because the compression limit δ_{min} cannot be equal to zero. We do not consider the time used to process data thanks to the application of FPGA boards in CubeSats such that the processing time is negligible compared with the length of each time slot [8].

Considering the total amount of incoming, outgoing and processed data at a node in each slot, we have the following constraints,

$$Q^{t+1}_u = Q^t_u - f^{out,t}_u - f^t_{u'} + f^{in,t}_u, \quad \forall u \in S^t \quad (6)$$

$$0 \leq Q^t_u \leq Q_{max}, \quad \forall u \in S^t \quad (7)$$

Constraint (6) indicates the evolution of a node's queue. Constraint (7) provides the feasible range of $Q_u(t)$.

3.3. Energy Model

Let λ^t_u denote the total energy expended by node u at slot t. The total energy expenditure includes energy incurred by data transmission/reception among satellites, in-network processing and data downloading. Formally, let E_t and E_r represent the energy cost of transmitting and receiving one unit of data via ISLs, respectively. Let E_d denote the energy cost of downloading one unit of data to GSs. It is worthwhile noting that the energy consumption of on-orbit data processing can be a linear or a non-linear function of $f^t_{u'}$ [13]. We consider linear function in this work and use E_a to denote the energy cost of processing one unit of data. We will consider non-linear function in a future work. Hence, for each satellite $u \in S^t$ and each ground station $g \in G^t$, the total energy consumption of satellite u at slot t is,

$$\lambda^t_u = f^{out,t}_u E_t + f^{in,t}_u E_r + f^t_{u'} E_a + f^{in,t}_g E_d \quad (8)$$

Each satellite is equipped with a solar panel. Let h^t_u be the harvested energy by node u at slot t, and its value will be drawn from the range $[0, h_{max}]$ according to the normal distribution. Note that h^t_u is fixed for time t but has a different value for different time slot t. We note that predicting solar energy arrivals based on historical records can be carried out accurately; see [16] and [17].

Let E^t_u represent the stored energy at node u at the beginning of slot t. We consider all satellites have the same finite battery capacity E_{max}. At $t = 1$, each satellite has an initial energy storage $E^1_u \in (0, E_{max}]$. The stored energy is constrained as follows:

$$E^{t+1}_u = \text{MIN}\{E^t_u - \lambda^t_u + h^t_u, E_{max}\}, \quad \forall u \in S^t \quad (9)$$

$$\lambda^t_u \leq E^t_u, \quad \forall u \in S^t \quad (10)$$

$$0 < E^t_u \leq E_{max}, \quad \forall u \in S^t \quad (11)$$

Constraint (9) ensures energy storage will never exceed E_{max}. Constraint (10) ensures each satellite only spends

its available energy in each time slot. Constraint (11) ensures no satellites deplete their battery.

4. Optimization Models

Given data Q_u^1 at satellite $u \in S^1$, our *aim* is to empty or download Q_u^1 worth of data from each satellite u to a ground station. In other words, we are interested in *emptying* Q_u^1 from a swarm of satellites subject to capacity and energy constraints. The problem at hand is then to determine (i) the activation time of each transmission set x_j^t, which governs the link capacity in each time slot t, (ii) the amount of data to be forwarded over each link in each time slot, i.e., $f_{(u,v)}^t$, and (iii) the amount of aggregated and compressed data at each node u, i.e., $f_{u'}^t$, in each time slot t.

4.1. Shortest Download Time

We first consider the following objective: minimize the number of time slots taken to download a given amount of data at satellites. Let $\phi^t \in \{0,1\}$ indicate whether the downloading process is complete at slot t. That is, if $\phi^t = 1$ is true, then the data queue of all satellites is not equal to zero at slot t; otherwise, we have $\phi^t = 0$. Let T be a large integer number. The value of T can be set as follows. Let T_u denote the total number of time slots required by satellite u to download all its data to one or more ground stations without using ISLs. Then the value of T can be set to the sum of T_u over all satellites, where $u \in S^t$. Mathematically, we have the following Mixed Integer Linear Program (MILP),

$$\text{MIN} \sum_{t=1}^{T} \phi^t \tag{12}$$

$$\text{s.t. } K\phi^t \geq \sum_{u \in S^t} Q_u^t, \forall t = \{1, \ldots, T\} \tag{13}$$

$$\sum_{u \in S^t} Q_u^1 = \sum_{t=1}^{T} \left(\sum_{g \in G^t} f_g^{in,t} + \sum_{u' \in S'} f_{u'}^t \right) \tag{14}$$

(1) - (11)

Constraint (13) forces the value of ϕ^t to be one whenever any satellite's data storage is non-empty, where $K \gg Q_{max}$. Otherwise, if all satellites have downloaded their data, then $\phi^t = 0$ for some t. Constraints (14) ensures that all data stored on satellites have been processed or downloaded to a ground station.

4.2. Minimum Compressed Data

Data that can be highly compressed will have a low QoI when the objective is to minimize the total download time. In order to satisfy the QoI requirements of different applications, it is necessary to minimize compressed data during a download process. Henceforth, we consider the objective of minimizing the total compressed data while ensuring the downloading process finished within a given period, say T time slots. The new MILP is shown below,

$$\text{MIN} \sum_{t=1}^{T} \sum_{u' \in S'} f_{u'}^t \tag{15}$$

s.t. (1) - (11), (14)

4.3. Minimum Energy Consumption

The last aim is to minimize the total energy consumed in a given downloading period. In practice, energy arrives randomly. Thus minimizing energy cost is necessary to prolong the lifetime of each satellite and to optimize the sustainability of the whole system. We have the following modified MILP,

$$\text{MIN} \sum_{t=1}^{T} \sum_{u \in S^t} \lambda_u^t \tag{16}$$

s.t. (1) - (11), (14)

5. Evaluation

We solve our MILPs using the commercial solver Gurobi [18]. In each time slot, we randomly generate a topology with ten satellites and four GSs. To model time varying topology, the probability that an edge exists between two satellites is 0.5. We assume the buffer size of satellites is 10 Mbits, and the battery capacity is 50 Joules. All links have the same capacity 1 Mbps. Based on [6] and [8], the energy used to transmit, receive, process or download 1 Mbits data is 1J, 0.75J, 1.6J and 1J, respectively. In each time slot, each satellite receives energy in the range $[0,1]$ (Joules). Initially, we have $\delta_{min} = 0.8$. The large integer K is 100, and T is 50. We compare against the case without data aggregation. This is represented as the case with $\delta = 1$ Lastly, every point in our figures is an average of five simulation runs.

First, we study different amounts of initial buffered data and the data reduction rate δ. The initial data increases from 1 to 10 Mbits with an interval of 1 Mbits. The value of δ increases from 0.5 to 1.0 with an interval of 0.1. From Figure 2, we can see the number of slots taken to download increases with the initial buffered data for all δ values. When $\delta = 1$ and the initial data is 10 Mbits, meaning all data needs to be downloaded without any aggregation/compression, the total download time is the longest at 25 slots. The download time

can be reduced by 52% when the value of δ is 0.9. Moreover, when $\delta = 0.5$, downloading 10 Mbits data per satellite only requires four slots, which is 16% of the download time used in the case with no in-network data aggregation.

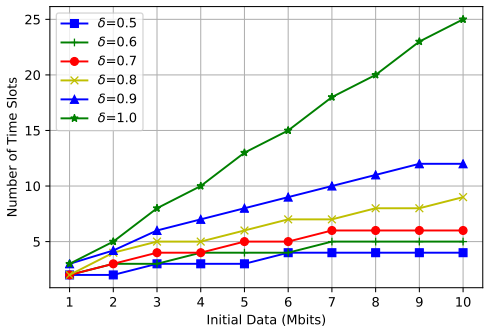


Figure 2. Minimum number of slots with different amount of initial buffered data and data reduction rate δ.

We then study increasing number of GSs, e.g., two to six, while the initial data increases from 1 to 10 Mbits. From Figure 3, we can see that the number of slots taken to download decreases with more GSs. This is reasonable because more GSs provide more download opportunities at the same time. When the initial data is 10 Mbits, download time reduces from 11 to 7 slots while the number of GSs increases from 2 to 6. The third experiment studies the effect of the number

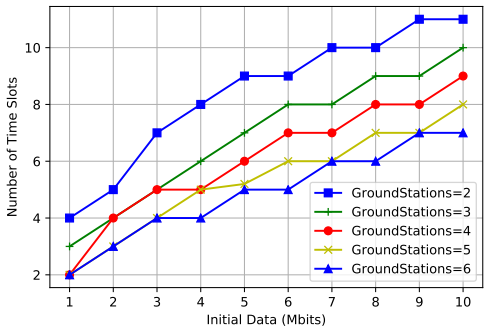


Figure 3. Minimum number of time slots with different amount of initial buffered data and number of GSs.

of edges. From Figure 4, we can see the number of time slots for data download increases with higher initial data. However, with different number of edges, the download time is remains roughly the same. This indicates that the number of edges has no effect on download time. This is because when there are a small number of ISLs, nodes can choose to process its own data before downloading, which also helps reduce download time as we discussed in Section 1.

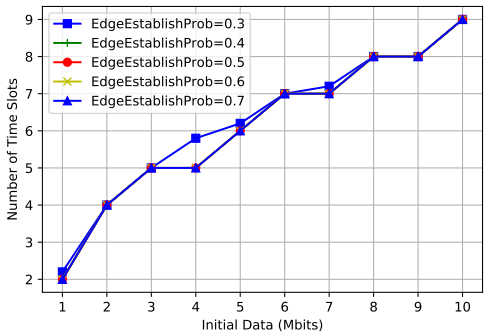


Figure 4. Number of time slots versus varying initial buffered data and edges.

The first three experiments are concerned with the MILP model in Section 4.1. In this fourth experiment, we now study the next model, see Section 4.2. Specifically, its aim is to minimize compressed data to ensure QoI requirements. The total number of download slots T increases from 5 to 14 with an interval of 3. From Figure 5, we can see the total compressed data increases with more initial data. This also translates to a decrease in total download time. When the available download time is short, e.g., 5 slots, satellites have to compress 80 Mbits data and sacrifice QoI to finish downloading. On the other hand, when the download time is sufficient, satellites choose to download data directly and only compress 44 Mbits of data to ensure a high QoI.

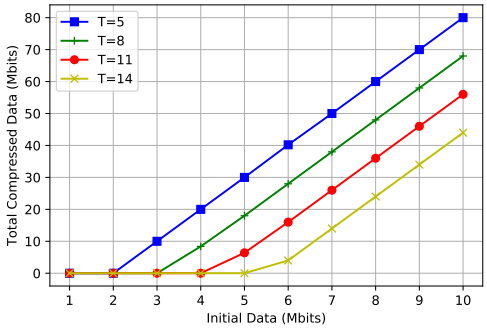


Figure 5. Minimum compressed data versus varying initial data and total download time.

The last experiment studies the minimum consumed energy with different amount of initial buffered data and download time as described in Section 4.3. With the same setting of T as in the fourth experiment, we changed the objective to minimizing the total consumed energy considering the limited resources

of small satellites. From Figure 6, we can see that energy cost increases when satellites have higher amounts of initial data and shorter download time. However, the difference in energy cost between the cases when the download time is 5 and 14 slots is only 21.6 Joules when the initial data is 10 Mbits. The reason is when download time is short, satellites will spend more energy on data compression to reduce the amount of data to download. On the other hand, when the download time is sufficient, more energy will be spent on downloading. As we mentioned at the beginning of this section, data compression costs more energy than data downloading. Thus, these different selections of satellites lead to a difference in the energy cost shown in Figure 6.

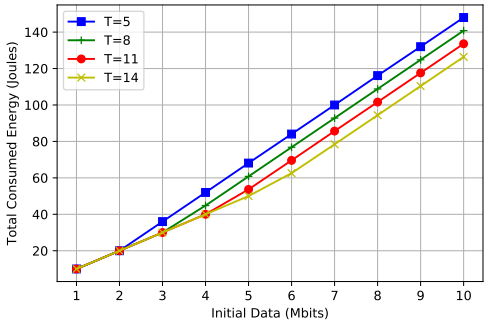


Figure 6. Minimum consumed energy with different amount of initial buffered data and download time.

6. Conclusion

This paper studies for the first time in-network data aggregation in small satellite networks. Our aim is to minimize the number of time slots used to download a given amount of time while satisfying constraints related to capacity and energy. This problem is formulated as an MILP. We also proposed an extension to minimize the total compressed data and consumed energy. Our results indicate that in-network data aggregation is able to significantly reduce download time. In future works, we plan to consider random channel gain and energy arrivals.

References

[1] R. Radhakrishnan, W. W. Edmonson, F. Afghah, R. M. Rodriguez-Osorio, F. Pinto, and S. C. Burleigh, "Survey of inter-satellite communication for small satellite systems: Physical layer to network layer view," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2442–2473, 2016.

[2] T. D. Cola *et al.*, "Network and protocol architectures for future satellite systems," *Foundations & Trends in Networking*, vol. 12, no. 1-2, pp. 1–161, 2017.

[3] A. Toorian, K. Diaz, and S. Lee, "The cubesat approach to space access," in *Aerospace Conference*, 2008, pp. 1–14.

[4] T. Lv, W. Liu, H. Huang, and X. Jia, "Optimal data downloading by using inter-satellite offloading in leo satellite networks," in *IEEE GLobecom*, Washington, DC, USA, Dec. 2017, pp. 1–6.

[5] X. Jia, T. Lv, F. He, and H. Huang, "Collaborative data downloading by using inter-satellite links in leo satellite networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1523–1532, 2017.

[6] O. Popescu, "Power budgets for cubesat radios to support ground communications and inter-satellite links," *IEEE Access*, vol. 5, no. 99, pp. 12 618–12 625, 2017.

[7] P. Hershey, B. Wolpe, J. Klein, and C. Dekeyrel, "System for small satellite onboard processing," in *IEEE International Systems Conference*, Montreal, QC, Canada, Apr. 2017, pp. 1–6.

[8] S. S. Arnold, R. Nuzzaci, and A. Gordon-Ross, "Energy budgeting for cubesats with an integrated FPGA," in *IEEE Aerospace Conference*, Big Sky, MT, USA, Mar. 2012, pp. 1–14.

[9] Y. An, J. Li, W. Fang, B. Wang, Q. Guo, J. Li, X. Li, and X. Du, "Eese: Energy-efficient communication between satellite swarms and earth stations," in *International Conference on Advanced Communication Technology*, Pyeongchang, South Korea, Mar. 2014, pp. 845–850.

[10] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[11] M. Tamai and A. Hsegawa, "Data aggregation among mobile devices for upload traffic reduction in crowdsensing systems," in *The 20th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, Bali, Indonesia, Dec. 2017.

[12] S. Nazemi, K. K. Leung, and A. Swami, "QoI-aware tradeoff between communication and computation in wireless ad-hoc networks," in *IEEE PIMRC*, Valencia, Spain, Sep. 2016, pp. 1–6.

[13] S. Yang, Y. Tahir, P.-Y. Chen, A. Marshall, and J. McCann, "Distributed optimization in energy harvesting sensor networks with dynamic in-network data processing," in *IEEE INFOCOM*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.

[14] S. Jeong, H. Kim, K. N. Dong, and I. Yoon, "Energy-aware data aggregation scheme for energy-harvesting wireless sensor networks," in *IEEE International Conference on Computer Communication and the Internet*, Waikoloa, HI, USA, Aug. 2016, pp. 140–143.

[15] K. Sundaresan, W. Wang, and S. Eidenbenz, "Algorithmic aspects of communication in ad-hoc networks with smart antennas," in *IEEE MobiHoc*, Florence, Italy, May 2006.

[16] S. Kosunalp, "A new energy prediction algorithm for energy-harvesting wireless sensor networks with q-learning," *IEEE Access*, vol. 4, pp. 5755–5763, 2016.

[17] J. R. Piorno, C. Bergonzini, D. Atienza, and T. S. Rosing, "Prediction and management in energy harvested wireless sensor nodes," in *International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology*, 2009, pp. 6–10.

[18] Gurobi, "Gurobi optimization," 2018, accessed on July 25, 2018. [Online]. Available: http://www.gurobi.com/.