INCREASING CUBESAT DOWNLINK CAPACITY WITH STORE-AND-FORWARD ROUTING AND DATA MULES

Trevor Koritza
Research done while student at:
California Polytechnic State Univ.
San Luis Obispo, CA, USA
email: tkoritza@comcast.net

John M. Bellardo
Computer Science Department
California Polytechnic State Univ.
San Luis Obispo, CA, USA
email: bellardo@calpoly.edu

ABSTRACT
Pico-satellites have recently gained substantial traction in research and educational communities due to their relatively low cost. The largest factor in keeping the cost down, their small size, also poses their biggest engineering challenge. The tiny, low power radios used to communicate with earth have extremely slow data rates. A typical pico-satellite is within communication range of the ground station for approximately 40 minutes per day with a theoretical maximum data rate of 1200 bps. At this speed a high-resolution digital photograph can take weeks to download.

This paper presents a novel communication protocol that allows a sparse network of pico-satellites to transfer data directly between one another. This capability is used to get the data to a "data mule". The data mule is a specialized satellite capable of relaying traffic back to earth at higher rates than the current satellites.

This work includes an implementation of the communication protocol and a simulator used to evaluate the protocol. Simulation results show that, regardless of varying satellite topologies and traffic workloads, the protocol has a significant increase in both the quantity of data transferred to earth and a reduction in the total time required to transfer all the data.

KEY WORDS
Wireless Communications Protocols and Standards, Performance Evaluation, Sensor Networks, CubeSat

1 Introduction

Satellites are becoming commonplace for applications including connecting the most remote regions of the world to digital communications networks, positioning systems, military applications and scientific research. For the most part the satellites used for these applications are very large and expensive to design, build, and launch. These economic factors prevent satellites' use as a platform for learning and important small-scale science, inspiring a pico-satellite revolution.

Pico-satellite projects, lead by CubeSat [12], have begun to explore cheaper solutions to flying satellites in very low earth orbits. These satellites are very small, 10 centimeters cubed, and have extremely constrained operating budgets for weight, power, and communications. The low orbits of these satellites permit the use of lower power, shorter range radios, but also creates additional communication problems. Due to the low orbits there is a very limited window of radio communication exposure to ground stations that operate the satellite. Each satellite is typically in range of the ground station for approximately 40 minutes. This, along with the maximum possible data rate of 1200 bps, yields roughly 350Kb during a single pass, ignoring communication protocol overhead. Goodput is substantially less once overhead and interference is taken into account. While this data rate is sufficient to operate the basic satellite systems, it is rapidly becoming a liability for performing more sophisticated scientific experiments.

This work proposes a store and forward network architecture and related protocols that increases the overall downlink capacity for a small cluster of CubeSat satellites. The cluster of satellites is heterogeneous, containing at least one "data mule". The data mule is equipped with a more powerful radio capable of faster communication with the ground station. This radio comes at the expense of any scientific payload in the satellite. Satellites in the cluster can communicate directly with each other. They cooperate to forward data, potentially over multiple hops within the cluster, to the data mule. The data mule finally transfers the data to earth.

This work also evaluates the proposed protocol using an accurate orbital simulator and actual CubeSat orbits as observed from prior launches based on their TLEs. The paper shows this communication architecture is able to send more data over a given amount of time with less end-to-end delay. Depending on the number of satellites and traffic characteristics the overall capacity of the network increases between three and five times and the end-to-end delay is reduced between 50 and 75 percent.

The remainder of the paper is organized into 5 additional sections. Section 2 reviews other work related to this problem. Section 3 proposes the network protocol. Section 4 discusses the simulator, the experiments that were run, and their results. Sections 5 and 6 present future directions for this work and conclude the paper.

2 Related Work

Solutions to the unique set of challenges facing pico-satellite constellations are not readily apparent in related work. These challenges are created by the low earth orbits the satellites fly in. The network is rarely, if every, fully connected. This precludes discovering a single path from a source satellite to the data mule and using it for the duration of the network. The network is also extremely sparse. It is expected that two satellites may wait a few months before flying within communication range of each other. Due to the nature of orbits, it is also the case that these communication opportunities are predictable with a high degree of accuracy.

2.1 Satellite Constellation Protocols

Most related work in the area of store and forward satellite networks is focused specifically on fully connected constellations and satellites with much higher orbits [8]. Parts of these works can be used in a sparse network of LEO satellites, but most of the protocols described are impractical solutions to the problem.

The most common protocol being used in satellite communications is adapted from asynchronous transfer mode (ATM). ATM based protocols such as [7] use Geosynchronous satellites to connect several ground stations. This protocol focuses on a network with GEO satellites, which are always connected and have a static network topology, making this protocol impractical for pico-satellite constellations.

Other routing algorithms have been proposed to create more purpose built protocols that specifically take advantage of satellite networks, but most of these assume that the satellite network is fully connected. CRT is an adaptive routing protocol that aims at providing routing to large constellations of LEO satellites. The main focus of this algorithm is to provide congestion control and load balancing over the constellation. CRT is specialized because it requires hardware that can receive from all neighboring satellites simultaneously [6]. Because of these constraints it is not directly applicable to the pico-satellite application.

Precomputed routes in satellite constellations were proposed by [5]. This routing protocol is not used for inter-satellite links (ISLs), but aimed at communicating from ground stations through satellites to other ground stations. Precomputed routes works well on static networks, such as GEO satellites connecting to a ground station, but it is not well suited to the dynamic network found in pico-satellites.

All of these projects deal with either inter-satellite communications focused on fully connected large meshes of satellites or with communicating directly to ground stations only. This paper, however, focuses on satellites that are very sparse, and will rarely be in range of one another. Because of the sparseness and discontinuity of the network, store and forward techniques need to be used not only when communicating with the ground stations, but also when communicating between satellites.

2.2 Sensor Networks

The existing research literature on sensor networks is more closely related to pico-satellite constellations than the research focused on GEO satellites and satellite constellations. The size and sparseness of the pico-satellites along with the low power budget and difficulty of getting data back to a single end-point make this problem closely related to a sensor network. In sensor networks there are several small nodes collecting data and transmitting that data back to a single place, the sink. In many cases the nodes, including the sink, can be moving. It is also possible to have multiple sinks in the same network. The common approach for improving the throughput and decreasing power consumption of a sensor network is the use of a data mule [14]. In [15] a submarine robot was used as a data mule to retrieve observation data from mobile sensors on the ocean floor that were collecting information on marine microorganisms. A conceptually similar data mule would be ideal for collecting data from various pico-satellites and relaying the data to the sink (ground station).

2.3 CubeSat

The GENSO Project [13] is building a cooperative network of ground stations that all contribute to the CubeSat project. This approach is valuable because the communication window between a single ground station and a single satellite in orbit is small. Increasing the number and geographic diversity of ground stations may allow nearly constant contact with any of the CubeSat satellites. The GENSO project compliments this work by also increasing the communication window with the orbiting data mule. Using the proposed protocol in conjunction with GENSO could further increase the network throughput.

3 Communication Protocol

The communication protocol is based on a simple premise – orbits are predictable. This assumption is appropriate because current CubeSat satellites lack attitude and directional control. Once the individual satellites learn the relative orbits of each other, they can schedule when and how much data to be sent back to earth either directly through the data mule, indirectly through the data mule and other intermediary satellites, or directly to the ground stations. This section describes the details of the communication protocol, including forwarding, routing / packet scheduling, MAC protocol, and the transport protocol. A more detailed explanation of this protocol can be found in Mr. Koritza's thesis [9].

ID1	ID2	Type1	Type2	Last Connection	Duration	Bit Rate	Time Between
S	1	SNSR	SNSR	9/18/2008	2s	5 bytes/s	34 days
S	D	SNSR	HBM	10/03/2008	3s	5 bytes/s	60 days
1	D	SNSR	HBM	8/30/2008	4s	5 bytes/s	76 days

Table 1. Example of what a link state table looks like (SNSR is a Sensor)

3.1 Routing

Terrestrial routing protocols typically operate on a next-link basis. The routing to any destination either implicitly selects the next-hop link using a distance-vector protocol, or explicitly selects the next-hop link using a link-state protocol. Due to the unique characteristics of low earth orbit satellite networks, there is no single persistent link to use as the basis for routing. Satellites are within communication range for a relatively short period of time and then can not directly communicate for a much longer period of time. This work uses the concept of a *connection* to capture this fundamental difference.

A connection is single, contiguous [1] period of time where two satellites are within communication range of each other. Over the course of a year a satellite may be involved in five unique connections with the same peer satellite. The protocol is similar to a link state protocol because every node in the network has a complete view of the global topology. It differs because the topology is composed of connections and not links.

Tracking individual connections requires the addition of three pieces of information to each entry in the routing table. It is necessary to store the anticipated *duration* of a connection. This information, used in conjunction with the expected data rate, gives the routing logic a close approximation of the maximum amount of data that can be transmitted during the connection. This is an important part of scheduling a single, larger data transfer across multiple connections.

The two other new pieces of information are the last time the connection with the same peer was established and the period of connections. Using these two pieces of information it is possible to accurately predict the start of the next n connections to the same peer. This information is necessary for sending a large file across multiple connections.

3.2 Learning Connection Parameters

CubeSats are launched as secondary payloads on much larger missions. As a result there is almost no control over the actual orbit of the satellites, making it extremely difficult to pre-program a table of connection parameters. To work around this problem a newly launched satellite enters the network in an "active learning" state.

[1] Due to orbits and rotational velocity it is possible there are short communication "grey periods" that interrupt an otherwise contiguous connectivity. The protocol accounts for these periods by decreasing the goodput estimate for the connection, but the periods do not cause one connection to terminate and the next one begin.

In the active learning state a satellite broadcasts a *HELLO* packet once every 120 seconds. All satellites that hear the broadcast respond, initiating a new connection. The initialization time is one of the connection parameters that gets recorded in the routing table. Once the connection is initiated routing and payload data is transferred as described below.

After there is no more data to transmit a satellite sends one "you there?" request per minute to its peer satellite. The peer will always acknowledge the "you there?" request. Once the satellites are out of range all responses will stop. The timestamp of the last response is used to determine the duration of the entire connection.

The final piece of information needed is the period between connections. To learn this parameter it is necessary to observe a second connection to the *same* peer satellite. The difference in start timestamps between the connections is the period.

Due to the nature of orbits realistic period values are anywhere from a few days to many months. If a satellite cluster has a set of orbits with a period measured in months, it will take that long before there is enough information to begin routing data through the entire network. As an optimization, it is possible for the earth station to seed the connection states based on the observed *two line elements* (TLEs). These accurate orbit descriptions are provided by NORAD based on radar tracking data within a few days of launch. Note that seeding the network in this way is an optimization, not an operational requirement.

There are two requirements for leaving the "active learning" state and entering the "normal operations" state. First, the connection parameters for all nodes in the cluster must be learned. This requires directly observing two connections, as described above. The second requirement is identifying which satellite or satellites are data mules.

In the "normal operations" mode a satellite stops periodically broadcasting the *HELLO* packet. Instead, the node schedules the next *HELLO* packet based on the information in the connection state table. It determines the anticipated start time of the next earliest connection, and begins broadcasting the *HELLO* packet 5 minutes before the anticipated start of the connection. This small buffer compensates for small errors in the start time estimation. Not transmitting the *HELLO* packets at other times reduces the power consumption of the satellite. The other main difference in "normal operations" mode is that data, and not just routing information, is exchanged during a connection.

Regardless of the operating mode, the properties of each subsequent connection are used to refine and update the connection state table.

3.2.1 Duration Calculations

"Grey periods" complicate the connection duration calculation. The three-dimensional orbit of two satellites cause the straight-line distance between the two to vary. The distance is not strictly decreasing as the satellites approach each other, and it isn't strictly increasing as they fly apart. This interesting characteristic prevents duration calculations from simply subtracting the connection end time from the start time. Instead the algorithm keeps track of the last time a packet was received from the other satellite, including acknowledgments. If the time between packets is greater than five minutes the algorithm assumes a grey period occurred and adds the interval to a running total of time taken by grey periods. Every two hours the satellites check to see when the last time a packet was received, if no packet has been received in the past two hours then the connection is ended and the duration is calculated, subtracting the running grey period total from the duration. Two hours was chosen based on experience with and observation of existing CubeSat orbits.

3.2.2 Best Path

Finding the shortest path from one satellite to another is more complicated in sparse satellite networks than in most terrestrial and dense satellite networks. In most dense satellite networks the topology is static and fully connected. This enables a routing protocol to use either a classic distance-vector algorithm, or to initiate a single route discovery process every time it needs to communicate with a new endpoint. Even in a dynamic, fully connected network protocols such as AODV, DSDV, or DSR can re-run the route determination logic to compensate for a broken link [2]. However, in a sparse satellite network the shortest path changes as a function of the physical location and relative speeds of every other satellite in the network.

The metric optimized by this routing protocol is latency. Given the positions of other satellites and other in-transit packets, it always delivers data with the smallest possible latency. To accomplish this it is necessary to look at all combinations of connections between all satellites, even subsequent connections between the same satellite. The information contained in the link state table enables the protocol to predict connection capacities into the indefinite future. The routing path algorithm must also take into account the preexisting load on each of the connections. The protocol uses a modified version of the Ford Fulkerson [4] flow algorithm to find the available flow in combination with the lowest latency path to the destination.

The Ford Fulkerson Algorithm is an iterative method that is initialized by setting all flow in the network to zero. Each iteration finds an "augmenting path" from the source to the sink, increasing the flow along that path. The maximum capacity of each edge in the graph minus the current assigned flow is an edge's residual flow (the remaining available flow for that edge). An augmenting path is a path from source to sink in the residual network. The iterations are repeated until there are no remaining augmenting paths in the residual network.

3.2.3 Topology Graph

The protocol uses a standard internal graph representation for the satellite network topology. The node originating the data is the one and only source node in the network. Except for connections involving the source node, each connection results in a new edge and new destination node in the network. A single satellite typically results in many nodes in the topology graph. To capture the ability of a single satellite to store and forward data across multiple future connections, an edge with infinite capacity is added between nodes representing the same satellite. This edge is directed so that data can only flow forward in time, not backwards.

The maximum capacity of each edge is computed based on the anticipated bitrate and duration of a connection. This maximum is reduced by a small percentage to allow for protocol overhead and other unforeseen factors that reduce connection goodput. The residual capacity for each edge is initialized to zero. The residual capacity is only changed as the result of augmenting flows and based on actual transferred data. The residual is *never* reset to zero during the life of the satellite. Each edge also includes a timestamp indicating when the edge will become available. This timestamp is used to decide between two otherwise equivalent edges.

The topology is always evolving. Nodes are actively pruned from the graph once a connection has occurred. Maximum connection capacities, connection durations, and timestamps are continually updated as better estimates become available. Also, residual edge weights get updated to reflect the quantity of data actually transmitted during a connection, not just the anticipated amount of data.

The topology graph is extended on-demand by adding nodes and edges for future, predicted connections. The graph is only expanded when there is more data that needs routing and no additional augmenting flow in the residual network.

3.2.4 Routing Algorithm

All data received from application(s) running on the satellite and received from other satellites in the network is routed the same way. The next best path through the residual network is determined using a DFS algorithm based on time of delivery. This algorithm returns the lowest latency path that still has non-zero residual flow available. If no such path exists the topology graph is expanded as previous described. This augmenting path includes the maximum number of bytes that can be transmitted over the path. If the path can accommodate all the data, just that one path is used. If the data is too big for the path, the data is split into two pieces. The first piece is the exact size of the path, and

gets scheduled for transmission over the path. The routing algorithm then iterates with the remaining data until a path has been selected for all pieces of the original data. After using some of a path's capacity the algorithm always updates the residual network graph.

Once the path for each piece of data has been determined, the data itself it queued in the output queue corresponding to the next-hop satellite. Typically the final destination for the data will be the data mule, which will forward it on to earth. However this is not a requirement. The same protocol works to enable multi-hop satellite to satellite communication simply by finding an augmenting flow to a different destination.

3.3 Data Link Layer

The data link layer is responsible for establishing connections, estimating the connection parameters between two adjacent satellites, and detecting and recovering from collisions, dropped packets, and timeouts.

The data link layer is currently implemented using a basic Aloha protocol. Aloha is a naive mac protocol that simply sends data as soon as it becomes available. If a collision occurs it exponentially backs off to minimize the probability of future collisions. Even though Aloha has a maximum capacity of 18% [1] it is acceptable in this network because the vast majority of data is generally flowing towards the data mule. Other than acknowledgements, there isn't much bi-directional communication. If it was desired to be able to upload large amounts of data to the satellites then a more sophisticated protocol would be needed. Such an improvement is left for future work.

The data link layer uses explicit acknowledgements to detect data loss. The recipient of a data packet is required to immediately acknowledge as much to the sender. If the sender doesn't hear the acknowledgement within a short period of time it retransmits the packet. The timing of the Aloha protocol is such that the sender anticipates the acknowledgement and will not begin transmission of the next packet until it arrives. Therefore the acknowledgement and subsequent data packet will not collide.

3.4 Network Layer

The network layer is responsible for routing and uses the algorithm previously described. In addition, it also controls the exchange of information across each connection. The data link layer notifies the network layer when a new connection becomes available. The first few packets in the connection exchange the link state table between the satellites. This enables each satellite to learn about all other satellites in the cluster, whether or not the two will every have a direct connection. Once the entire link state table has been exchanged, the two satellites transfer any pending data.

Pending data is kept in a separate queue for each satellite. Once a piece of data has been routed and the next-hop satellite is determined, the data is added to the outbound queue for that satellite. The network layer begins transmitting the data from this queue after a connection is established and the link state tables have been exchanged. Depending on the size of the queue, the capacity of the connection, and the number of transmission errors, the queue may or may not be completely drained in a single connection. If there is data left in the queue that was *supposed* to be transmitted during the connection, it gets removed from the queue and rerouted. If there is data remaining in the queue that wasn't scheduled for transfer until a future connection, it remains in the output queue.

4 Protocol Evaluation

Due to the expense of building, flying, operating satellites, and the uncertainties of testing a new communication protocol, we decided to prototype the protocol in a simulator. For the simulator to be useful it must be able to simulate accurate three dimensional orbits of CubeSat pico-satellites, detect collisions, and determine whether or not two satellites are within communication range. It is important to simulate actual CubeSat orbits rather than idealized orbits or orbits of much higher satellites.

After considering various commercial alternatives, we decided to build our own simulator around the open source PREDICT [10] orbit tracking software. The PREDICT code provided the logic to accurately determine 3D location of a satellite at any point in its orbit. To further increase realism all orbits used in the simulator are actual orbits of previous CubeSat satellites as reported by NORAD's satellite tracking facility. A discrete event simulator was added to the PREDICT code. This enabled precise simulation of fine-grained network events.

The radio reception model was based on two parameters, bitrate and communication range. Events were scheduled for the beginning end of a transmission, and the beginning and end of a reception. These events were used to detect collisions at a receiver. A collision happens when another packet arrives at the receiver while it is actively receiving a different packet. In order for a transmission to be successful there can not be any collisions and the satellites must have remained within communication range the entire time.

The simulator is designed in such a way that the protocol implementation code can not access the internal simulator data. For example, it is impossible for the protocol implementation to "cheat" and obtain a list of all satellites and their positions from the simulation engine. This forces the protocol implementation to not cut corners, and makes it much easier to port the implementation to a CubeSat platform in the future.

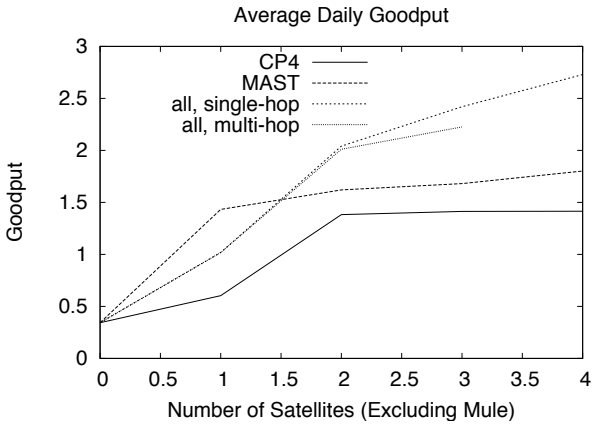

Figure 1. Maximum amount of data through network for a varying number of hops and transmission sources.

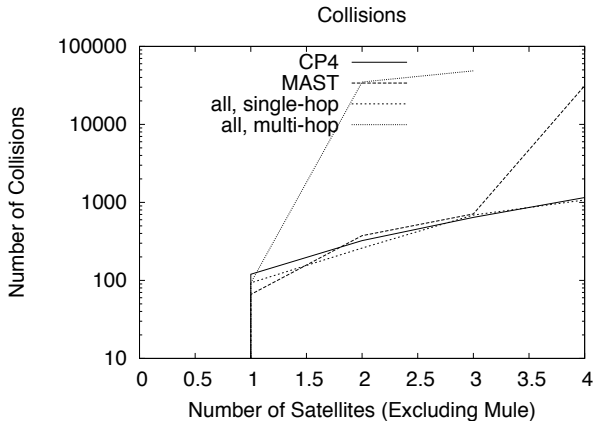

Figure 2. Absolution number of collisions observed while performing the goodput experiments in Figure 1. Note the Y-axis is log-scale.

4.1 PolySat Communication Protocol

The PolySat program at Cal Poly operates two CubeSat satellites, CP3 and CP4. The only communication that takes place is done at 1200 bps directly between the ground station and either the CP3 or CP4 satellites. Due to the orbits of the satellites there are less than 40 minutes per day where communication actually takes place. The satellites lack the capability to communicate directly with each other. The "PolySat" direct communication protocol is used as the baseline performance data point to evaluate the effectiveness of the data mule protocol.

4.2 Results

The results show that this protocol has substantial improvement over the PolySat protocol in both goodput and end-to-end delay. The satellites used in this evaluation were CP3 and CP4 from the PolySat project [3], SAUDICOMSAT 4 and 6 commercial satellites from Saudi Arabia, and MAST from Stanford University [11].

4.3 Goodput

Goodput incorporates all protocol overhead, such as packet headers and retransmissions, to determine the actual network performance as seen by the applications. We ran several 365 day simulations to gather goodput data. The results presented in this paper are average daily goodput values, ignoring the 138 days required to establish the link state table. Ignoring these days is justified because the orbit mode seeding optimization described previously is expected to almost eliminate this startup time.

Figure 1 shows how the goodput changes as a function of the number of satellites in the network. This data is shown for three network configurations. The common assumptions across the configurations are the presence of just one data mule and a satellite-to-satellite radio with 150km range capable of communication at 100kbps. These parameters were chosen to be as realistic as possible given existing radio technology. The configurations vary in the number of transmitting satellites and the utilization of multi-hop routes. In the simplest case there is just one source satellite talking directly to the data mule. This is shown in the "CP4" and "MAST" lines in the graph. This data provides a reasonable upper bound on the amount of data that can be communicated between two satellites using the radios in the simulation. It also confirms the expectation that the actual amount of data transferred is dependent on the relative orbits of the two communicating satellites.

The second configuration tested was all satellites sending as much data as possible directly to the mule ("all, single-hop"). This simulates how the network would behave if there was no multi-hop inter-satellite communication. This isolates the benefit of a data mule from the benefit of a multi-hop path to the data mule. The last configuration was all satellites sending as much data as possible via multi-hop paths to the mule ("all, multi-hop").

The zero value on the X-axis in figure 1 is the maximum goodput in the current PolySat satellites. This is a theoretical maximum calculated on the properties of the actual connection and orbits. Each connection has a data rate of 1.2kbps and a maximum of 40 minutes per day exposure.

Figure 1 shows the PolySat communication only achieves 0.34 MB per day of data transfer. Both CP4 and MAST alone communicating to a data mule are a big improvement. CP4 is approximately double with 0.6 MB per day and MAST is almost five times better with 1.43 MB per day.

4.4 Collisions

Collisions are undesirable. One source of collisions is two separate radio transmissions arriving overlapped at the same receiver. In this case both transmissions are corrupted

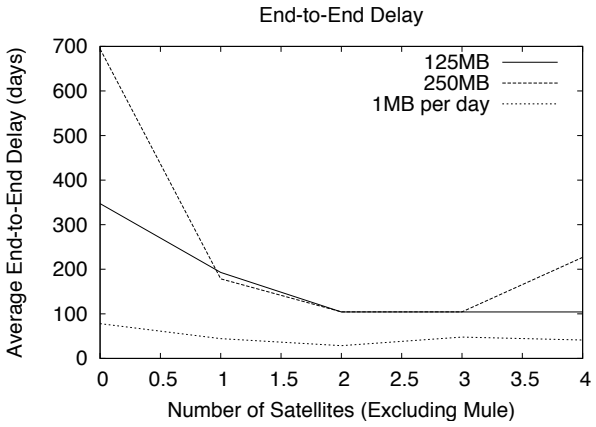

Figure 3. End-to-end Delay of large files transferred back to the ground station. Each line represents a different series of simulations.

and must be retransmitted, resulting in overall lower goodput from the network. Collisions are also one measure of the effectiveness of a MAC protocol. As the number of collisions created by the MAC protocol increases, so does the importance of fixing that protocol to improve overall performance. Figure 2 shows the absolute number of collisions seen in each corresponding experiment from figure 1.

Figure 2 shows that collisions remain relatively infrequent with one, two, or three satellites in the network. 800 collisions is a small number when compared to the several hundred thousand or millions of packets that are transmitted. However once the 4th satellite is added to the network the collisions increase dramatically to over 31000 in the case of "MAST". In addition, the number of collisions in the "all, multi-hop" case is high. The network load is much higher in this case as well, because every node is transmitting as much data as possible.

There are two reasons for this. First, with 4 satellites in the network all transferring large amounts of data, it is more probable for two satellites to have slightly different forwarding schedules. That is, if the source satellite S is expecting the packet to traverse the path $S \Rightarrow T \Rightarrow U$, but intermediate satellite T decides that $T \Rightarrow V$ is more efficient, other packets that S has scheduled on the path $S \Rightarrow V$ will cause a large increase in collisions. One possible solution to this problem is to introduce a source routing scheme where the original sender can specify the entire route the packet traverses.

The second step that can be taken to reduce the number of collisions is to implement a more intelligent mac protocol that does a more accurate job of scheduling radio transmissions.

4.5 End-to-End Delay

End-to-end delay of large transfers is another benefit of this protocol. Figure 3 shows that using this protocol even with a naive mac protocol, there is still significant improvement in end to end delays for larger files. The results also show the delay times achieved, as well as the amount of data that can be sent in those times, differs greatly between different satellites due to the differences in orbits.

As in previous simulations, each transmitter/receiver pair was tested with a varying number of other satellites in orbit. When adding the extra satellites the do not originate any data. These additional satellites will forward the data but do not have any of their own data to send.

The number of satellites at zero on the X-axis signifies the current PolySat method of sending data between a satellite and Earth directly. This is the same theoretical maximum calculation discussed earlier. Including this data point makes it obvious that all configurations of the multi-hop protocol outperform what is currently being used.

One interesting result seen in figure 3 is that the amount of time to transfer the 250MB file actually goes up when three additional satellites are used in the network. This is caused by a large number of collisions occurring during the data transfer. Once the third extra satellite is added the number of collisions goes from about 700 to over 5000. The solutions to this problem were discussed in section 4.4. It is worth noting that despite this behavior, the CubeSat network is able to transfer data back to earth quicker than what is currently being used.

4.6 Data Mule Capacities

The graphs in the previous section assume the data mules that are being used are not acting as a bottleneck in the system. Even though this data mule needs to transmit a longer distance to the ground station it should be able to get much higher speeds than the satellites are currently able to achieve. This is due to the larger communication budget afforded by not including any scientific payload on the data mule. Table 2 shows the minimum bit rates needed by the mule compared to the bit rate of the satellite to satellite radio. Table 2 assumes two satellites in the network always send the maximum amount of data and the data mule has only 40 minutes per day exposure time to the ground station. As the table shows, when there are two satellites in the network the data mule needs to send data to the ground station at less than half the speed of the satellites. As the number of satellites increases the bit rate required by the data mule will also need to increase, but because these networks are sparse with very few satellites this is a realistic constraint. When the data needs of a constellation grow too large for the downlink capacity of a single data mule, it is always possible to add additional data mules. Assuming there are equal bit rates in both the inter-satellite radios and the data mule, there should be approximately one data mule per four satellites.

5 Future Work

This work can be extended in many different directions.

Sensor (bps)	Mule (bps)
1.2k	0.47k
10k	4.7k
100k	47k
1M	470k
10M	4.7M
50M	21.4M

Table 2. MULE data rates required to not be a bottleneck, using 40 minutes per day exposure time

The simulator currently only supports the satellite to satellite communication links. The satellite to ground link capacities are a theoretical evaluation. Simulating these links will further increase the accuracy of the results.

The simulations identified collisions and medium contention as a problem in larger networks. This should be addressed by using a more sophisticated MAC protocol.

Satellites in the network currently assign flow with complete disregard to other flows from other satellites that may have been already be assigned to the same connection. One approach to alleviating this problem is using the low capacity satellite to ground station link to exchange flow assignment information and incorporate this into the scheduling algorithm.

Further improvements to the connection prediction algorithm, perhaps by incorporating first and second order derivatives, has the potential to reduce power consumption.

Another intriguing direction is evaluating the applicability of this protocol to satellites in different orbits, including medium earth orbit satellites using GEO stationary data mules.

A final improvement to this protocol is utilizing network of multiple ground stations, as is being developed by the GENSO [13] project. This is complimentary work that will only increase the overall capacity of the network.

6 Conclusion

The protocol proposed in this paper for connecting a sparse satellite network with the use of data mules shows improvement over sending data directly from each satellite down to the ground. Both maximum throughput and end-to-end delay are improved. The hardware assumed in the simulations was well within a reasonable range, only 150km range and 100kbps data rate. The maximum goodput was increased by three times when all satellites have large amounts of data to send, and almost five times when only one satellite is sending large amounts of data. The end-to-end delay was almost cut in half when sending 1MB per day per satellite and is decreased by a factor of at least three when sending large amount of data from only one satellite. Increasing the hardware of the satellites or connecting more ground stations into the network as well as adding some of the recommended future improvements this protocol will outperform the current one by even more.

7 Acknowledgements

This work was sponsored by the Department of the Navy, Office of Naval Research, under Award # N00014-07-1-1152.

References

[1] N. Abramson. THE ALOHA SYSTEM: another alternative for computer communications. In *Proceedings of the November 17-19, 1970, fall joint computer conference*, pages 281–285, Houston, Texas, 1970. ACM.

[2] S. Bansal, R. Shorey, and A. Misra. Comparing the routing energy overheads of ad-hoc routing protocols. In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, volume 2, pages 1155–1161 vol.2, 2003.

[3] C. Cal Poly San Luis Obispo. PolySat. http://polysat.calpoly.edu/index.php, 2009.

[4] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. pages 651–663. McGraw-Hill Book Company, 2nd edition edition, 2001.

[5] H. Cruz-Sanchez, L. Franck, and A. Beylot. Precomputed routing in a store and forward satellite constellation. In *Vehicular Technology Conference, 2007. VTC-2007 Fall. 2007 IEEE 66th*, pages 240–243, 2007.

[6] Y. He and S. Pelagatti. CRT: an adaptive routing protocol for LEO satellite networks. In *Information and Communication Technologies, 2006. ICTTA '06. 2nd*, volume 2, pages 2496–2501, 2006.

[7] A. Hung, M. Montpetit, and G. Kesidis. ATM via satellite: a framework and implementation. *Wirel. Netw.*, 4(2):141–153, 1998.

[8] O. Korcak and F. Alagoz. Analysis of priority-based adaptive routing in satellite networks. In *Wireless Communication Systems, 2005. 2nd International Symposium on*, pages 629–633, 2005.

[9] T. Koritza. Store and forward routing for sparse pico-satellite sensor networks with data-mules. *http://digitalcommons.calpoly.edu/theses/104*, 2009.

[10] J. Magliacane. PREDICT - a satellite Tracking/Orbital prediction program. http://www.qsl.net/kd2bd/predict.html, 2009.

[11] N2YO.com. LIVE REAL TIME SATELLITE AND SPACE SHUTTLE TRACKING. http://www.n2yo.com/, 2009.

[12] C. S. L. Obispo. CubeSat community website - home. http://cubesat.calpoly.edu/, Feb. 2009.

[13] G. Project. GENSO. http://www.genso.org/, 2009.

[14] S. Shapiro. Generalizations of random store and forward communication networks. *Proceedings of the IEEE*, 55(12):2191–2192, 1967.

[15] B. Zhang, G. Sukhatme, and A. Requicha. Adaptive sampling for marine microorganism monitoring. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 2, pages 1115–1122 vol.2, 2004.