SolarNet: A Proposed Reliable Extraterrestrial Communications System

Jessica Chen[1], Daph Guo[2], and Allen Lin[3]

[1]jesschn@mit.edu
[2]daphgull@mit.edu
[3]allenees@mit.edu

Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA 02139

Recitation Instructor: Olivia Brode-Rogers

WRAP Instructor: Rachel Molko

May 7, 2025

Contents

1. Introduction

The SolarNet project is a proposed update to NASA's previous point-to-point extraterrestrial communication system between antennas on Earth and a network of devices in space. Taking into account specific environment and behavior requirements for delivery of often time-sensitive and high-volume data, our design aims to optimize the reliability and latency of the network. We take into consideration existing limits of transmission time and link bandwidths, storage capacities of individual devices, and necessity of regular system updates.

Our proposed design augments the existing structure to improve reliability and performance by delegating the sending of data across modules, allocating space in memory and storage, and maintaining communication within our network. In this system, we refer to reliability as the system's ability to deliver messages and updates from the source to its intended destination with high probability of success. We also value performance, namely that the system has the ability to send messages and updates in a timely fashion (minimizing latency). Our design choices also take potential expansion of the system in the future into consideration, but scalability is not a prioritized goal.

This paper presents an extensive analysis of our design. First, we provide an overview of the general architecture and the operational framework of SolarNet. Subsequently, we analyze how each module contributes to our system's main goals of achieving reliability and performance. Finally, we conclude by examining and mathematically evaluating various use cases to demonstrate the wide range of applications and far-reaching impacts of our system, ranging from daily communications to disaster warnings for numerous regions on Earth.

2. System Overview

SolarNet comprises the physical module, the Bundle Protocol, and the routing protocol. These modules must not only function reliably across long distances but also perform proficiently even in conditions such as high congestion, strict storage constraints, and limited bandwidth of transmission.

The physical module consists of satellites and other hardware that can be organized into three main components, which are shown :

- **near-Earth devices**, which include Earth's antennas, low-Earth orbit communication satellites (LEOComs), and geosychronous equatorial orbit communication satellites (GEOComs);
- **near-Moon devices**, which include the Moon's antennna and its two relays; and
- **near-Mars devices**, which include Mars' antenna and its two relays.

These components are shown in Figure 1.

In our proposed update of the system, we implement a heartbeat system to detect potential outages and maintain functionality of devices, as well as design a more specific protocol for fragmentation in the bundle protocol to improve performance. We also allocate memory and storage for specific types of data, store backup versions of updates, and distribute updates in such a way that improves the reliability of the system.

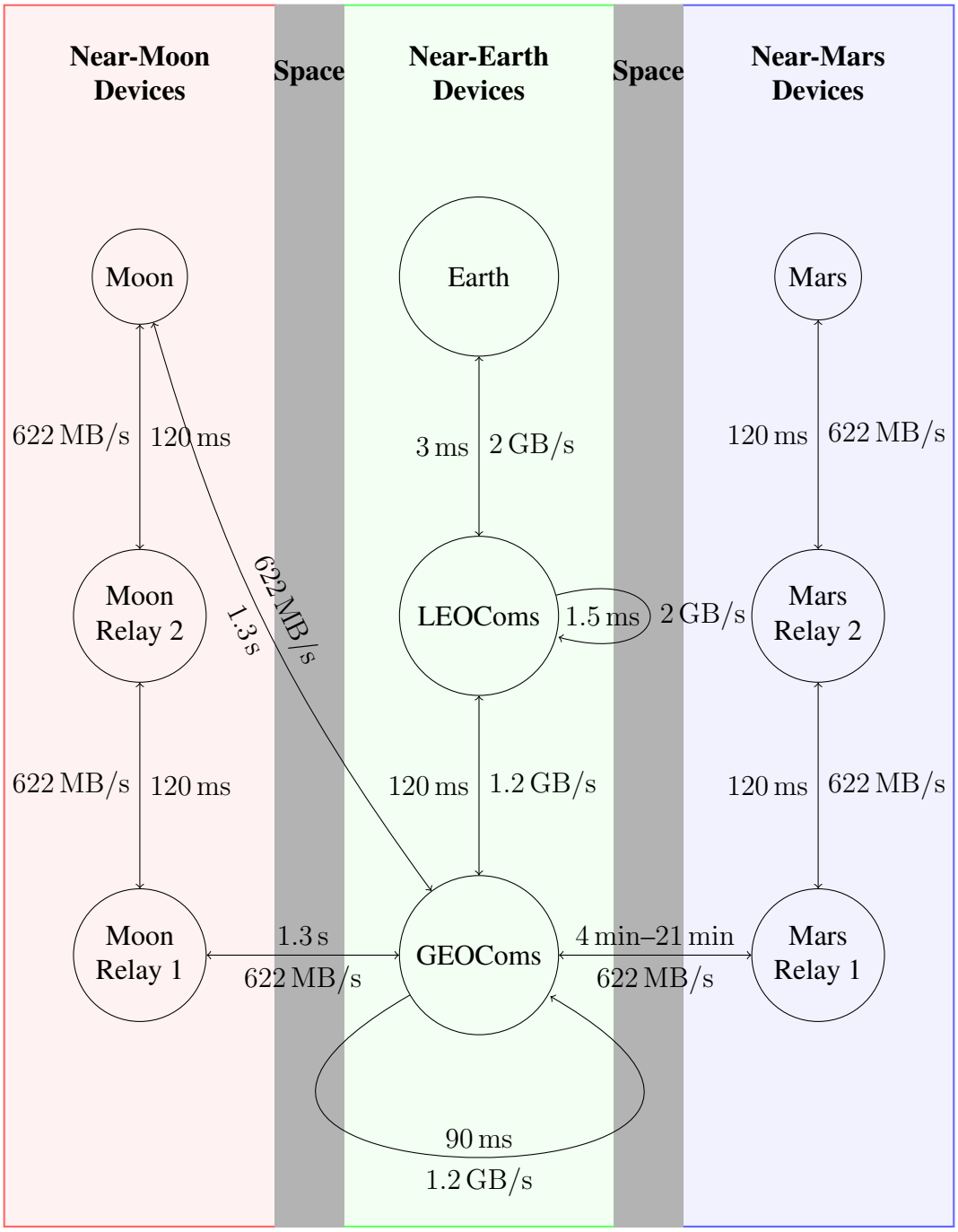

Figure 1. The minimum time required and bandwidth of each point-to-point connection.

Figure 1 shows each point-to-point connection with its bandwidth and the minimum time required for a bundle to traverse it. The latter is calculated by computing the amount of time light takes to traverse the distance between the devices, establishing a lower bound on the time of any transmission.

Note that any transmission from Earth to the moon or Mars must traverse through a series of LEOComs and GEOComs, and vice versa. A transmission to or from the moon either traverses through both moon relays or directly to the moon if the GEOCom is sufficiently

close. On the other hand, any transmission to or from Mars must use the two Mars relays.

3. System Design

3.1. Memory and Storage Allocations

Because each type of device serves a different purpose in the system and has contrasting requirements for storage contents, the organization of long-term storage depends on the device type.

Figure 2 shows the allocation for different types of transmissions. The storage is comprised of two halves: a persistent storage and a volatile storage that intermittently fails.

Our design allocates more storage for higher-priority transmissions, free space for any data type in the case that our allocated space overflows, and storage for a copy of the most recent software update, heartbeat ACKs, and duplicates of custody bundles. Although Figure 2 depicts these as discrete categories within the memory or storage, bundles of the same category might not actually be distributed as discrete chunks. Figure 3 depicts the process of storing a bundle in the node's memory or storage.

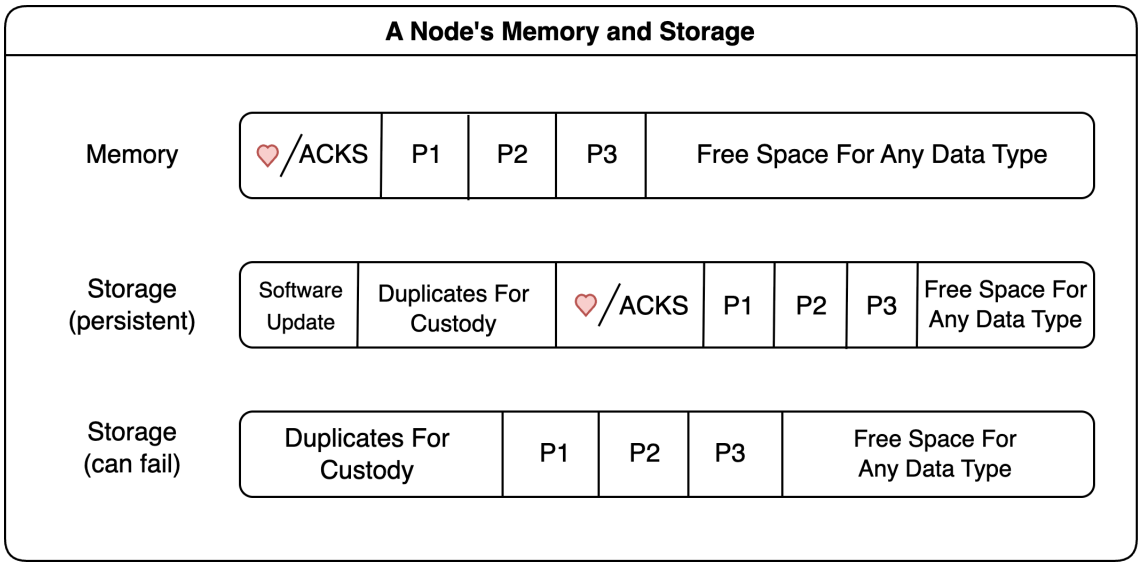

Figure 2. Distribution of memory and storage allocation (not to scale).

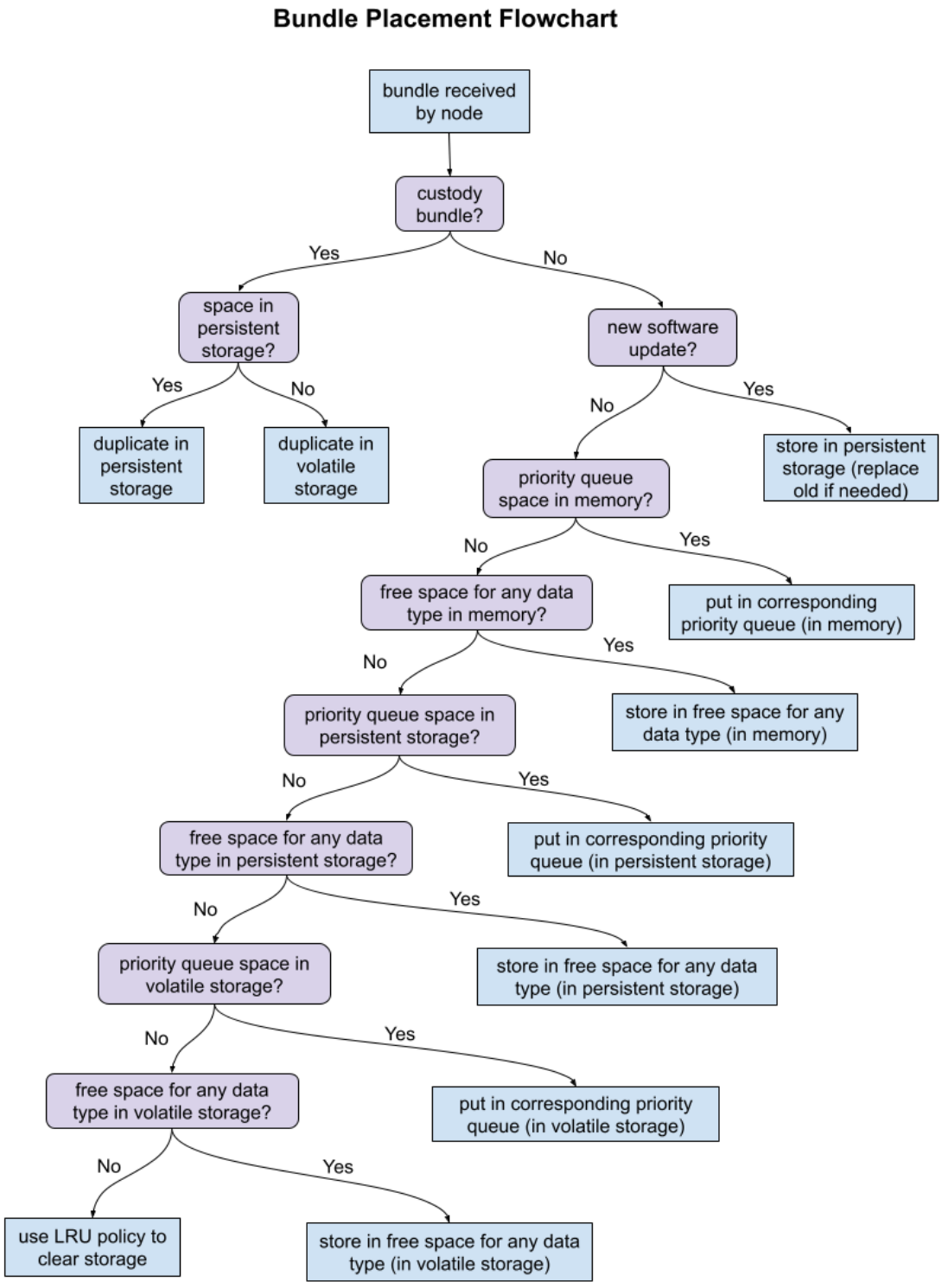

Figure 3. Actions taken by the node upon receipt of a bundle.

We keep counters for how much total storage space is being used by each type of transmission. As such, we can detect types that have overflowed their guaranteed storage space. Counters can be easily incremented and decremented whenever a file is stored or deleted.

Each device stores a copy of the previous version of its own software in its persistent storage (see Table 2), which ranges from $1\,\mathrm{GB}$ to $100\,\mathrm{GB}$ in size depending on the type of device. This redundancy makes the system more robust, as the device remains proficiently functional and is able to reset itself to a working version by pulling from the persistent storage if bugs are detected in the updates, or if the node crashes mid-update.

This decision trades off space in storage for availability of the network. We argue that the second one is more valuable: only a small fraction of storage needs to be allocated as shown in Table 2, and parts of the network being down for long periods of time would affect users who route time-sensitive messages through the network.

Device	**Minimum Space Allocated (in GB)**					
	Priority			**Acks Heartbeats**	**Free Space**	**Total**
	1	**2**	**3**			
LEOCom	8	8	16	0.1	31.9	64
GEOCom	8	8	16	0.1	31.9	64
Relay	8	8	16	0.1	31.9	64
Antenna				0.1	63.9	64
Land Telemetry				0.1	63.9	64
Solar Telemetry				0.1	63.9	64

Table 1. Distribution of memory for telemetry and non-telemetry devices.

In Table 1, we allocate in memory $0.1\,\mathrm{GB} = 100\,000\,\mathrm{kB}$ for ACKs/Heartbeats; this easily holds 100,000 heartbeat ACKs assuming a high upper bound of $1\,\mathrm{kB}$-size ACKs. We also allocate a minimum space for each priority to make sure that every node has the ability to store bundles of all priorities so that one priority never starves out the others for storage space. However, we reflect the importance of priority 3 messages by allocating double the amount of space ($16\,\mathrm{GB}$) given to priority 2 and 1 messages ($8\,\mathrm{GB}$). The rest is free space that can be used by any priority of message, such that the allocations serve as guaranteed minimum available storage levels rather than strict allocations. No transmission storage is allocated in antennas and land/solar telemetry satellites because they are usually just sources and destinations for information.

Device	**Minimum Space Allocated (in GB)**						
	Priority			**Software Update**	**Duplication Custody**	**Free Space**	**Total**
	1	**2**	**3**				
LEOCom	31.25	31.25	62.50	4	62.50	58.50	250
GEOCom	31.25	31.25	62.50	16	62.50	46.50	250
Relay	31.25	31.25	62.50	1	62.50	61.50	250
Antenna				100		4900	5000
Land Telemetry				4		4996	5000
Solar Telemetry				16		4985	5000

Table 2. Distribution of storage on the persistent disk of each device.

In Table 2's persistent storage distribution, we follow a similar reasoning to allocate double the space for priority 3 bundles compared to priority 1 and 2, but here we also account for allocating space for software updates (specific values shown in the table) and custody duplicates. Since at least a third but no more than half of the nodes are custody nodes, we upper bound the space allocated to custody duplicates as half of the total space given to the three priority queues. This is precisely $(31.25 + 31.25 + 62.50)/2 = 62.50\,\mathrm{GB}$. The remaining space is once again allocated to free space for any data type.

Device	**Minimum Space Allocated (in** GB**)**					
	Priority			**Duplication Custody**	**Free Space**	**Total**
	1	**2**	**3**			
LEOCom	31.25	31.25	62.50	62.50	62.50	250
GEOCom	31.25	31.25	62.50	62.50	62.50	250
Relay	31.25	31.25	62.50	62.50	62.50	250
Antenna					5000	5000
Land Telemetry					5000	5000
Solar Telemetry					5000	5000

Table 3. Distribution of storage on the volatile disk of each device.

In Table 3's volatile storage distribution, the amount of space allocated for the priority queues and custody duplication is the same as in Table 2, but we do not keep copies of software updates. This is due to the fact that volatile storage fails intermittently, and we want to store the software update copies reliably to maintain a more functional network. Thus, the extra space gets allocated to the free space for transmission overflows instead.

3.2. Queues and Deletion Policy

The queues are implemented using doubly-linked lists in storage and memory, where each file is stored with pointers to the next and previous files in its queue. Queue operations are then a simple reassignment of pointers, and the space overhead is constant for each stored transmission.

We similarly implement a least-recently-used (LRU) deletion policy; that is, when a node runs out of storage, it deletes stored files starting with the least recently accessed transmissions. The node keeps track of this using another set of doubly-linked-lists implemented as before, one list for each priority level. When a file is added to storage or modified, it is moved to the front of its LRU list. Therefore, we maintain that the entries at the back of the list are the least recently used.

We delete LRU files first in order to give all transmissions a fair amount of time in the queue, since we must delete something to free up storage for new transmissions. It causes the least harm to delete messages that are definitely old and less relevant, rather than newer ones that are likely to still be important.

As detailed in the evaluation section, it is not likely that nodes will run out of storage under normal use cases. Overflowing storage is likely caused by an influx of important

messages, such as emergency telemetry data as detailed in use cases. Our deletion policy must also take into account priority levels, since deleting higher priority messages may have more consequences for their sender and receiver. When a node runs out of storage, we first check which priority levels have overflowed their minimum storage allocations using the counters and delete from the lowest overflowed priority using the LRU data structure. However, no matter the priority level, custody nodes should never delete custody bundles until the next node on the custody path has confirmed custody transfer.

3.3. File System

We implement our nodes' storage systems using the Zettabyte File System (ZFS) to prepare for the possibility of crashes and to automate administration. We found that ZFS best suits our system's needs and priorities compared to UNIX and GFS.

Though UNIX is beneficial for efficiently accessing and enforcing modularity, crashing with unfinished data can prevent nodes from accessing any data at all [1]. Since reliability is one of our top priorities, we want to better prepared for crashes. As well, all administrative commands must originate from Earth and be passed through the network remotely which takes valuable time and resources. Therefore, ZFS's automated administration capabilities are crucial to streamline recovery after a node crashes. ZFS also uses a simpler administration compared to UNIX, which requires manual commands when making changes to data [2].

In addition to the benefit of having simpler, automated administration when using ZFS, this file system enforces data integrity using error detection and corrections, as well as enforcing recoverability [2]. Data integrity ensures that we maintain the accuracy of our data, which is especially important for storing custody bundles. Recoverability is imperative for our system which is not fault-tolerant and safe from crashes on its own.

The other file system we considered was GFS, which is beneficial for systems consisting of large files and in need of recovery mechanisms [3]. However, GFS is optimized for a specific pattern of traffic unique to Google's databases, which does not apply to our network [3].

3.4. Detecting Nonfunctional Devices

With so many physical devices, for the sake of performance and reliability our network needs to adapt when devices unexpectedly become defunct. We implement a heartbeat system: every device periodically sends a heartbeat request to each of its neighbors, who must answer as soon as possible, enabling failure detection if no response is received within the timeout period. Thus, the system can respond to failures in a timely manner by reporting this information back to Earth, improving performance by minimizing the time the network remains disrupted.

We distinguish two types of heartbeats based on physical distance of links: inter- and intra-device group heartbeats. Inter-device group heartbeats verify that connections between different types of devices are functional (e.g. from GEOCom to relay), whereas intra-device heartbeats verify that the connections between the same types of devices are operational (e.g. within all LEOComs).

The exact frequencies of inter-device group heartbeats are described in Table 4. We considered sending more frequent heartbeats, on the magnitude of seconds, but each node

can only send and receive one bundle in parallel at a time. Sending out heartbeats too frequently risks flooding the network and starving out actual communications. However, sending less frequent heartbeats risks affecting communications that are expected by the sender to be delivered within 10 minutes. We consider the two following cases.

The distinct inter-device case here is the transmission of heartbeats between GEOComs and the Mars relay. In the worst case, a GEOCom-to-Mars relay roundtrip takes 42 minutes. We set the heartbeat frequency to $5\,\text{min}$ so that at steady state, the GEOCom and Mars relay are receiving heartbeats from each other every 5 minutes. We design a timeout of $42+5=47$ minutes so that the system allows for a buffer of one heartbeat ACK before detecting a timeout.

For all other inter-device groups, we set the heartbeat frequency to 1 minute and timeout to be 3 minutes. The roundtrip times between inter-device groups are on the order of milliseconds. As such, at steady state, our system incorporates flexibility to allow a buffer of 2 heartbeat ACKs before detecting a timeout. Thus, even if time-critical information is sent and a node realizes a device it was initially forwarded to has failed, the system will discern this in 1 minutes and can forward to another node instead, fitting within the 10 minute time frame and still allowing for the small chance that the next nodes fail.

Device-to-Device	Frequency	Timeout
Earth Antennas to LEOComs	$1\,\text{min}$	$3\,\text{min}$
LEOComs to GEOComs	$1\,\text{min}$	$3\,\text{min}$
GEOComs to Mars Relay	$5\,\text{min}$	$47\,\text{min}$
GEOComs to Moon Relay	$1\,\text{min}$	$3\,\text{min}$
Moon Relay to Moon Antenna	$1\,\text{min}$	$3\,\text{min}$
Mars Relay to Mars Antenna	$1\,\text{min}$	$3\,\text{min}$

Table 4. Inter-device heartbeat frequencies.

The exact frequencies for intra-device group heartbeats are described in Table 5. Similarly to the previous table, we wanted the frequencies to be on the order of minutes to prevent flooding while also being mindful of potential time restrictions for messages. While it is still important to check the connections between devices that are in the same device group, these heartbeats have lower frequency because there are many LEOComs and GEOComs; if one or two fail, their neighbors can usually compensate. Going further with this logic, since there are many more LEOComs than GEOComs, LEOCom heartbeats are half as frequent as GEOCom heartbeats.

Heartbeats between relays keep the same frequency and timeout as the inter-device group heartbeat system. There are only two of them on each of Mars and the Moon, and it is necessary to go through them before reaching their antennas, making it critical to keep track of their statuses.

Device Group	Frequency	Timeout
Relays	1 min	3 min
LEOComs	3 min	6 min
GEOComs	1.5 min	3 min

Table 5. Intra-device group heartbeat frequency and timeouts.

These heartbeat acknowledgment requests are small in size and high-priority. ACKs alone are usually around 150 bytes, and when calling `create_bundle` using the ADU to add a packet header of about 134 bytes, we get a total size of around $150 + 134 = 284$ bytes. This is less than $1\,\mathrm{kB}$ per minute, which is already on the smallest end of our average routine communication traffic, so we know that our SolarNet system can handle the addition of this heartbeat system without flooding the network.

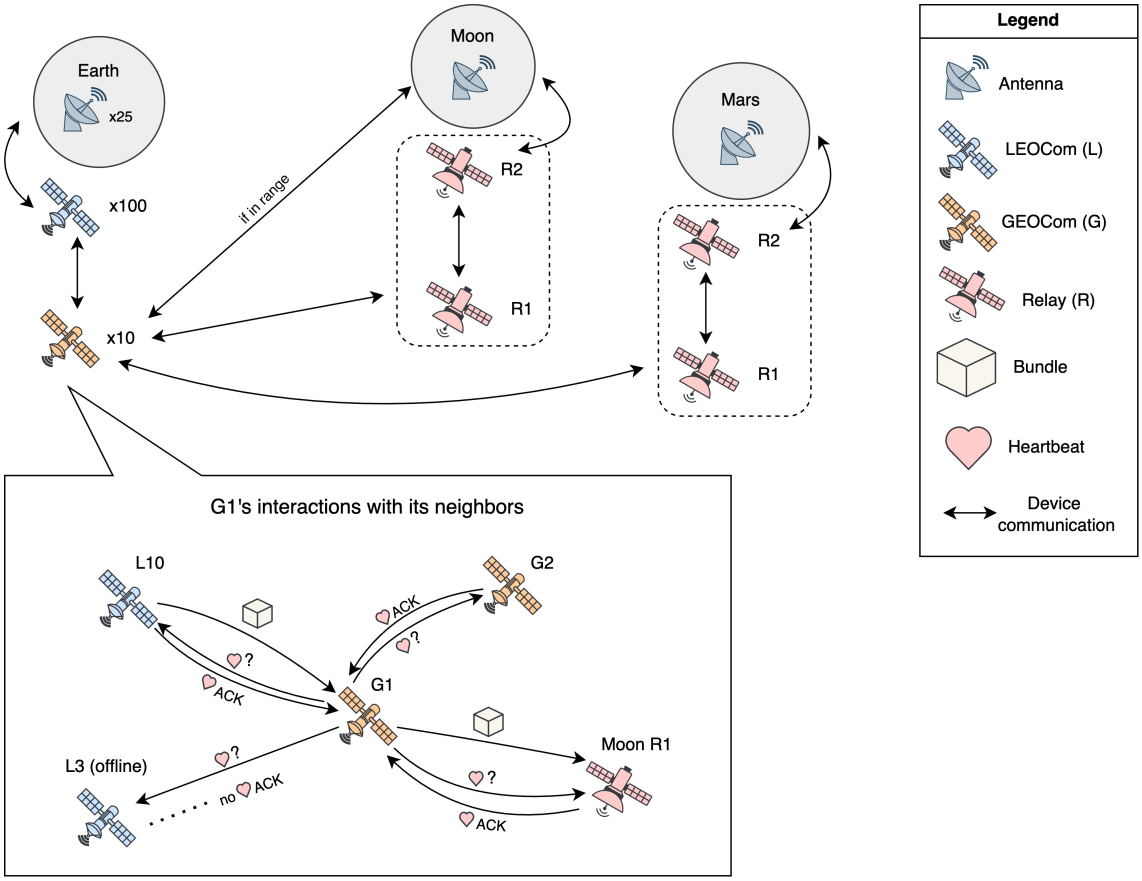

Figure 4. The inter-device group heartbeat system.

Figure 4 illustrates the inter-device group heartbeat system in action for a single GEOCom node and its neighbors in the SolarNet system, showcasing how failed/offline devices can be detected when no heartbeat ACK is sent back when expected within a timeout period.

3.5. Bundle Protocol Fragmentation

Due to the long periods of the devices' orbits and the potential for some nodes to be temporarily unusable, it is possible that the link to the next node accessed by `next_hop(destination)` does not have enough capacity to send the full bundle according to the routing table. The network should take advantage of connections between nodes when they exist by fragmenting the portion of the bundle that falls within the maximum transmissible bytes using the `fragment_bundle` function call. The exception to this is administrative bundles and ACKs, which are small enough that fragmenting would generate unnecessary traffic through multiple status reports and impact performance.

The remaining bundle portion still needs to be sent; one option is to put it back into the corresponding queue with `push_fragment_onto_head_of_queue(bundle)` and wait for a better next hop connection. The remainder is pushed back onto the head instead of the tail of the queue because we want to maintain its original position in the queue.

Our other option is to send a copy of the remainder to a neighbor obtained from `get_neighbors()`, in the hopes that the neighbor has either a better connection to the next node or a viable alternate route. We take this option if the bundle is priority 2 or 3, or if the bundle's queue has become longer than expected based on the calculations in Section 5. Time-sensitive messages would be negatively impacted by simply remaining at the same node waiting for an opportunity to send that might not appear soon.

To minimize the risk of infinite loops caused by nodes simply forwarding bundles back and forth forever, neighbor forwarding only happens with probability p, and otherwise the bundle is put back on the queue as before. The probability p is initialized to 0.5 and dynamically changes based on a running estimate of how much network traffic is currently from rerouting traffic, similar to window sizing in TCP. It increases by 0.1 every minute to a maximum of 1, since as time goes on with no neighbor bundles, it is more likely that nearby nodes have enough resources to enable alternate path-finding. When the node receives a forwarded bundle from a neighbor, p is reduced by a factor of 0.5 to avoid overloading the network; this also helps prevent a bundle from being forwarded back and forth on an infinite loop. We also make the neighbor rerouting asymmetrical by always choosing the neighbor in the same direction along the GEOCom or LEOCom orbit circle (e.g. always the one towards the east). See Figure 5 for a visual of these two options during fragmentation.

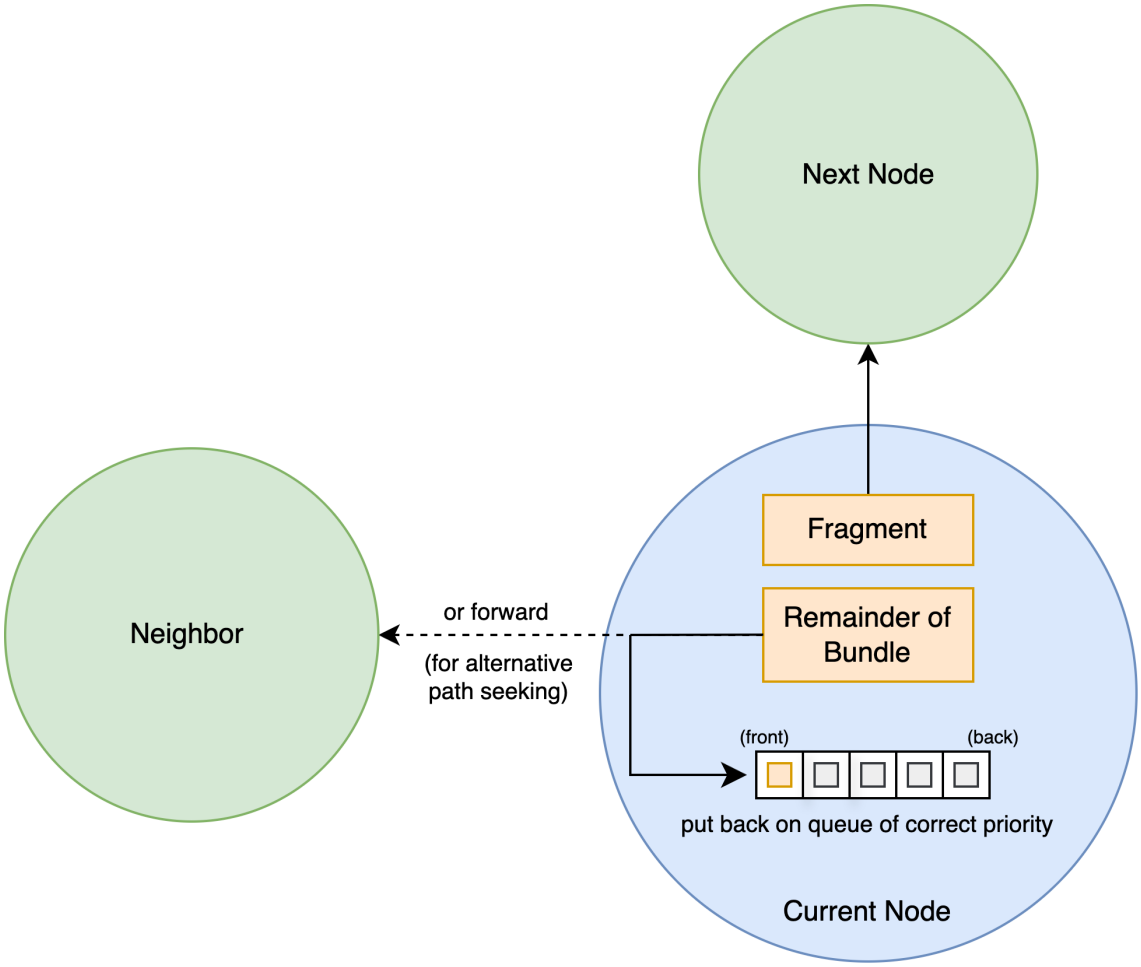

Figure 5. When a node fragments a bundle, it can choose to forward the remainder to a neighbor or return it to the front of the queue.

Duplication leads to redundant packets in our network which could increase congestion, but it would help improve reliability by allowing bundles to travel along multiple possible paths to their destination. Although we incur the risk that duplicated traffic may overwhelm the network or create an infinite loop, implementing this policy probabilistically and adjusting for observed congestion levels makes this less likely.

3.6. Routing Protocol

3.6.1. Basic Forwarding

When a node receives a bundle, it first calls `unpack_bundle` to extract the bundle's ADU and source. Given the ADU's size, the node consults the routing table to determine the next possible hops, specifically presenting the maximum number of bytes that can be forwarded to each hop. The choice of how much to forward to which next hop(s) depends on the properties of the bundle. Specifically, the routing protocol considers

(I) custody bundles,
(II) administrative bundles and bundles with the report requested flag,
(III) bundles containing software updates, and

(IV) other bundles

separately.

To handle the different priority levels, we must have that for all source-destination pairs all messages with higher priority are sent before all messages with lower priority. This is especially critical for edge cases where we get many high priority messages at once.

The system implements this by simply sending all bundles in the higher priority queues before all lower priority bundles, at every node. We concluded that this is our best option in terms of completing the specifications while remaining simple to implement and minimizing overhead. We also considered trying to keep pointers for each source and destination pair so we can implement the requirement individually while not starving out priority 2 messages for pairs that have no priority 3 messages, but ultimately the overhead of a more complicated system is not worth the slight performance improvement.

3.6.2. Software Updates

In order to guarantee that software updates are delivered correctly and maintain the functionality of the network, when updates are sent out they should be prioritized over all other transmissions. Since the management software on Earth must know if updates are completed in order to schedule the next batch of updates, each update bundle requests a status report from its final destination, and these reports must also be prioritized.

Although it is beneficial to the overall health of the system to initiate all updates as soon as possible, this simple solution introduces several adverse effects. The updates will disrupt usual traffic in the network, both because the bundles themselves are more high-priority and may starve out other types of traffic, and since devices undergoing updates are non-operational for several hours while the update completes. Additionally, a single wave of updates means bugs are not detected until the entire system has been updated and needs to be reverted.

Therefore, updates will be sent in multiple waves rather than all at once, servicing the furthest nodes (Mars and Moon antennas) first, then gradually servicing all the Earth satellites. This additionally enables bugs in the more distant nodes to be detected first so that the critical innermost nodes are less likely to receive buggy updates. Moreover, not all nodes are reachable at any given time, so Earth must work out a schedule of which nodes are updated when. It is important that the majority of the GEOComs and LEOComs remain operational at all times to maintain basic functionality of the network, since they are the sole points of contact to the Earth antennas from which updates and other messages originate. Thus, at most 20 LEOComs and 1 GEOCom will undergo updates at any given time.

4. Use Cases

4.1. Main Applications

We give a brief discussion on the main applications pertaining to NASA's usage of the SolarNet system: routine communications, land satellite telemetry, and solar telemetry. We additionally discuss some unintended negative impacts that may result from our system.

Since human communications between terrestrial users can utilize other more direct avenues, telemetry use cases should be prioritized except in the special case of human communication between mission control and manned missions. The telemetry data is necessary to give timely warnings of natural disasters, which impacts even people who are not directly using the network.

4.1.1. Routine Communications

The main purpose of our system is to support routine communications between users on the Earth, the moon, and Mars. Our implementation allows users at two different locations to reliably and efficiently transmit and receive messages. Such messages include personal messages, emails, and videos between family members, as well as some more urgent and mission-critical communications from NASA. When a user sends a message, the SolarNet system arranges the data into a bundle and forwards it to the appropriate location. Users can choose to designate important messages as custody bundles to improve the reliability of the delivery.

These messages are usually low-volume, non-critical, and range from $1\,\mathrm{kB}$–$10\,\mathrm{MB}$ with average traffic $1\,\mathrm{kB/min}$. In Section 5.1, we show that our design delivers these transmissions quickly and completely. This holds true even with intermittent mission-critical transmissions, as our system prioritizes sending messages with higher priority; consequently, users receive these critical messages reliably and within 10 minutes of the expected delivery time.

4.1.2. Land Satellite Telemetry

There are two land telescopes that orbit the Earth solar-synchronously and provide satellite imagery that are crucial in accurately measuring various altitudes. Kyrgyzstan uses the satellite imagery to predict landslides and mudslides in very steep, rocky, and remote regions with at least 10 minutes of advanced warning for the population to evacuate. Normally, these land telescopes send the telemetry data to the ground stations in northern Scotland and eastern Australia. However, in an extreme emergency when the telescopes cannot transmit to the ground stations directly, they instead transmit data through the SolarNet network. Our system provides quick and reliable delivery of the massive telemetry data (twenty $2\,\mathrm{GB}$ files) to the ground stations by prioritizing these transmissions.

4.1.3. Solar Telemetry

In a similar vein, large volumes of data from the Sun-observing satellites will frequently pass through the network to be analyzed for the start of solar storms. Solar storms have far-reaching consequences for the operation of the terrestrial networks most of humanity relies on, as well as on the SolarNet itself. This is because solar storms can disrupt magnetic fields, which can interrupt GPS systems, radio communication, and power grids. To combat this, SolarNet is used to predict solar storms and issue global warnings when necessary; once again, reliability and latency are important concerns.

5. Evaluation

SolarNet aims to provide NASA with reliable and efficient communication for its operations, so it is appropriate to evaluate our design in this context. We analyze the performance of the system by measuring measuring the latency of bundle deliveries in various contexts including average case and worst-case.

Our design provides more reliable delivery and performance to critical and urgent management data. Such data include priority 3 bundles, emergency land telemetry data, solar telemetry data, and system and maintenance updates.

5.1. Average Case Analysis

In a usual setting, the SolarNet system is used to transmit routine communicnations, urgent tasks, solar telemetry data, and software updates. We analyze the performance and reliability of our system in each of those cases.

5.1.1. Latency of Routine Communications

Usually, the routine communications comprise the main traffic on the SolarNet system since telemetry data transmissions and software updates only occur at a per-month frequency. These transmissions range from $1\,\mathrm{kB}$ to $10\,\mathrm{MB}$. These files are so small that each node takes at most $(10\,\mathrm{MB})/(622\,\mathrm{MB\,s^{-1}}) = 0.016\,\mathrm{s}$ to send each bundle. In fact, the average rate of traffic of $1\,\mathrm{kB/min} = 0.0166\,\mathrm{kB/s}$ is 3.732×10^7 times smaller than the smallest bandwidth of $622\,\mathrm{MB/s}$, meaning that such transmissions are immediately transmitted by the node. As such, there is plenty of bandwidth for the four people on the moon and Mars and administrators on Earth to communicate, resulting in no significant buildup of the queue. This holds true even when the traffic is bursty since we do not expect the traffic rate to increase by a factor of more than 3.732×10^7.

The only buildup is caused by the disruption by the LEOCom-GEOCom communication link when the LEOCom are at the poles, which happens 10% of the time. As such, we expect a total delay of $2.4\,\mathrm{h}$ between LEOCom and GEOCom. At the average rate, such delays only generate buildups of $2.4\,\mathrm{h} \cdot 60\,\mathrm{min/h} \cdot 1\,\mathrm{kB/min} = 144\,\mathrm{kB}$, which can be sent in at most $(144\,\mathrm{kB})/(622\,\mathrm{MB\,s^{-1}}) = 0.231\,\mathrm{ms}$ and hence considered negligible.

Moreover, the size of these files are small enough such that the transmit times are bounded by the physical distance between devices. For example, assuming worst-case that the bundles traverse through around 50 LEOComs and 5 GEOComs, the amount of time for

a $10\,\mathrm{MB}$ bundle to reach the moon is

$$\underbrace{\max\left(\frac{10\,\mathrm{MB}}{2\,\mathrm{GB/s}}, 0.003\,\mathrm{s}\right)}_{\text{Earth antenna to LEOCom}} + 50\underbrace{\max\left(\frac{10\,\mathrm{MB}}{2\,\mathrm{GB/s}}, 0.015\,\mathrm{s}\right)}_{\text{LEOCom to LEOCom}} + \underbrace{\max\left(\frac{10\,\mathrm{MB}}{1.2\,\mathrm{GB/s}}, 0.120\,\mathrm{s}\right)}_{\text{LEOCom to GEOCom}}$$
$$+ 5\underbrace{\max\left(\frac{10\,\mathrm{MB}}{1.2\,\mathrm{GB/s}}, 0.090\,\mathrm{s}\right)}_{\text{GEOCom to GEOCom}} + \underbrace{\max\left(\frac{10\,\mathrm{MB}}{622\,\mathrm{MB/s}}, 1.3\,\mathrm{s}\right)}_{\text{GEOCom to Moon Relay 1}} + \underbrace{\max\left(\frac{10\,\mathrm{MB}}{622\,\mathrm{MB/s}}, 0.120\,\mathrm{s}\right)}_{\text{Moon Relay 1 to Moon Relay 2}}$$
$$+ \underbrace{\max\left(\frac{10\,\mathrm{MB}}{622\,\mathrm{MB/s}}, 0.120\,\mathrm{s}\right)}_{\text{Moon Relay 2 to Moon antenna}} = \boxed{2.865\,\mathrm{s}}.$$

Similarly, the amount of time for a $1\,\mathrm{kB}$ bundle to reach the moon is

$$\underbrace{\max\left(\frac{1\,\mathrm{kB}}{2\,\mathrm{GB/s}}, 0.003\,\mathrm{s}\right)}_{\text{Earth antenna to LEOCom}} + 50\underbrace{\max\left(\frac{1\,\mathrm{kB}}{2\,\mathrm{GB/s}}, 0.015\,\mathrm{s}\right)}_{\text{LEOCom to LEOCom}} + \underbrace{\max\left(\frac{1\,\mathrm{kB}}{1.2\,\mathrm{GB/s}}, 0.120\,\mathrm{s}\right)}_{\text{LEOCom to GEOCom}}$$
$$+ 5\underbrace{\max\left(\frac{1\,\mathrm{kB}}{1.2\,\mathrm{GB/s}}, 0.090\,\mathrm{s}\right)}_{\text{GEOCom to GEOCom}} + \underbrace{\max\left(\frac{1\,\mathrm{kB}}{622\,\mathrm{MB/s}}, 1.3\,\mathrm{s}\right)}_{\text{GEOCom to Moon Relay 1}} + \underbrace{\max\left(\frac{1\,\mathrm{kB}}{622\,\mathrm{MB/s}}, 0.120\,\mathrm{s}\right)}_{\text{Moon Relay 1 to Moon Relay 2}}$$
$$+ \underbrace{\max\left(\frac{1\,\mathrm{kB}}{622\,\mathrm{MB/s}}, 0.120\,\mathrm{s}\right)}_{\text{Moon Relay 2 to Moon antenna}} = \boxed{2.863\,\mathrm{s}}.$$

Thus, there is no noticeable difference in sending these bundles from Earth to the moon despite the bundle sizes ranging over 5 orders of magnitude. A similar computation for a Mars-bound bundle (for both sizes) yields a time of $21.026\,\mathrm{min}$ at the worst case. Since there is no queue buildup, we expect all crucial task-related messages to be delivered within $10\,\mathrm{min}$ of their expected time. For the same reason, we do not expect higher priority queues to starve out lower priority queues since the bundles are sent out almost instantaneously.

5.1.2. Solar Telemetry

Prediction of solar storms occur about once a month. The solar telemetry data is stored in $200\,\mathrm{MB}$ files that are transmitted to Earth at a rate of one file every minute. Our system prioritizes the transmission of solar telemetry data if it is determined to be high-priority. Each node takes at most $(200\,\mathrm{MB})/(1.2\,\mathrm{GB\,s^{-1}}) = 0.166\,\mathrm{s}$ to send a $200\,\mathrm{MB}$ file. The $1.2\,\mathrm{GB/s}$ bandwidth is 360 times larger than the rate of $200\,\mathrm{MB/min}$, so the system is not overwhelmed by the introduction of these $200\,\mathrm{MB}$ files. In fact, the $0.166\,\mathrm{s}$ creates a queue buildup of $0.166\,\mathrm{s} \cdot 0.0166\,\mathrm{min/s} \cdot 1\,\mathrm{kB/min} = 2.75\,\mathrm{B}$ of routine messages, a negligible amount.

Assuming worst-case the bundles traverse through 5 GEOComs and 50 LEOComs to

reach the desired Earth antenna, the amount of time that one $200\,\mathrm{MB}$ file takes is

$$\underbrace{\max\left(\frac{200\,\mathrm{MB}}{1.2\,\mathrm{GB/s}}, 0.090\,\mathrm{s}\right)}_{\text{solar GEOCom to GEOCom}} + 5\underbrace{\max\left(\frac{200\,\mathrm{MB}}{1.2\,\mathrm{GB/s}}, 0.090\,\mathrm{s}\right)}_{\text{GEOCom to GEOCom}} + \underbrace{\max\left(\frac{200\,\mathrm{MB}}{1.2\,\mathrm{GB/s}}, 0.120\,\mathrm{s}\right)}_{\text{GEOCom to LEOCom}}$$
$$+ 50\underbrace{\max\left(\frac{200\,\mathrm{MB}}{2\,\mathrm{GB/s}}, 0.015\,\mathrm{s}\right)}_{\text{LEOCom to LEOCom}} + \underbrace{\max\left(\frac{200\,\mathrm{MB}}{2\,\mathrm{GB/s}}, 0.003\,\mathrm{s}\right)}_{\text{LEOCom to Earth antenna}} = \boxed{6.3\,\mathrm{s}}.$$

Since the bandwidth is more than plentiful, our system can send twenty $200\,\mathrm{MB}$ file in $20\,\mathrm{min}$ (since we are constrainted by the rate of $200\,\mathrm{MB/s}$), while simultaneously transmitting routine messages as well. This is well within the required $5\,\mathrm{h}$ warning window.

5.1.3. Software Updates

Software updates are sent fairly frequently to different devices through the SolarNet network. We make the assumption that Earth can schedule these updates to avoid the $2.4\,\mathrm{h}$ delay caused by their orbits. Our system prioritizes the software updates when sent from Earth. We analyze each type of device separately and consider the impact of these updates on queue sizes.

We first consider LEOComs updates. Each LEOCom update has size $4\,\mathrm{GB}$. At a minimum bandwidth of $2\,\mathrm{GB/s}$, each node takes at most $(4\,\mathrm{GB})/(2\,\mathrm{GB\,s^{-1}}) = 2\,\mathrm{s}$ to transmit the update bundle. The queue buildup of routine messages caused by the $2\,\mathrm{s}$ processing is $2\,\mathrm{s} \cdot 0.0166\,\mathrm{min/s} \cdot 1\,\mathrm{kB/min} = 0.033\,\mathrm{kB}$, which is minuscule.

At the worst case, such an update is transmitted through 50 other LEOComs to arrive at the desired LEOCom. This takes about

$$\underbrace{\max\left(\frac{4\,\mathrm{GB}}{2\,\mathrm{GB/s}}, 0.003\,\mathrm{s}\right)}_{\text{Earth antenna to LEOCom}} + 50\underbrace{\max\left(\frac{4\,\mathrm{GB}}{2\,\mathrm{GB/s}}, 0.015\,\mathrm{s}\right)}_{\text{LEOCom to LEOCom}} = \boxed{102\,\mathrm{s}},$$

which is within the $30\,\mathrm{min}$ requirement.

Next, each GEOCom update has size $16\,\mathrm{GB}$. At bandwidth $1.2\,\mathrm{GB/s}$, each node takes at most $(16\,\mathrm{GB})/(1.2\,\mathrm{GB/s}) = 13.3\,\mathrm{s}$. This causes $13\,\mathrm{s} \cdot 0.0166\,\mathrm{min/s} \cdot 1\,\mathrm{kB/min} = 0.215\,\mathrm{kB}$ of queue buildup of routine messages, which is again insignificant.

At the worst case, a GEOCom update is transmitted through 50 LEOComs and 5 GEOComs to arrive at the desired GEOCom. This takes about

$$\underbrace{\max\left(\frac{16\,\mathrm{GB}}{2\,\mathrm{GB/s}}, 0.003\,\mathrm{s}\right)}_{\text{Earth antenna to LEOCom}} + 50\underbrace{\max\left(\frac{16\,\mathrm{GB}}{2\,\mathrm{GB/s}}, 0.015\,\mathrm{s}\right)}_{\text{LEOCom to LEOCom}} + \underbrace{\max\left(\frac{16\,\mathrm{GB}}{1.2\,\mathrm{GB/s}}, 0.120\,\mathrm{s}\right)}_{\text{LEOCom to GEOCom}}$$
$$+ 5\underbrace{\max\left(\frac{16\,\mathrm{GB}}{1.2\,\mathrm{GB/s}}, 0.090\,\mathrm{s}\right)}_{\text{GEOCom to GEOCom}} = \boxed{8.1\,\mathrm{min}},$$

which is again within the $30\,\mathrm{min}$ limit.

We next consider relays. The minimum bandwidth here is $622\,\mathrm{MB/s}$, so a node processes a $1\,\mathrm{GB}$ relay update bundle in at most $(1\,\mathrm{GB})/(622\,\mathrm{MB/s}) = 1.6\,\mathrm{s}$. From the above analysis, this produces a negligble amount of queue buildup.

We compute the estimated time for the second relay to receive the update since the first relay is upper bounded by that time. Again assuming that it is transmitted through 50 LEOComs and 5 GEOComs to arrive at the desired relay, the moon relay update takes

$$\underbrace{\max\left(\frac{1\,\mathrm{GB}}{2\,\mathrm{GB/s}}, 0.003\,\mathrm{s}\right)}_{\text{Earth antenna to LEOCom}} + 50\underbrace{\max\left(\frac{1\,\mathrm{GB}}{2\,\mathrm{GB/s}}, 0.015\,\mathrm{s}\right)}_{\text{LEOCom to LEOCom}}$$
$$+ \underbrace{\max\left(\frac{1\,\mathrm{GB}}{1.2\,\mathrm{GB/s}}, 0.120\,\mathrm{s}\right)}_{\text{LEOCom to GEOCom}} + 5\underbrace{\max\left(\frac{1\,\mathrm{GB}}{1.2\,\mathrm{GB/s}}, 0.090\,\mathrm{s}\right)}_{\text{GEOCom to GEOCom}}$$
$$+ \underbrace{\max\left(\frac{1\,\mathrm{GB}}{622\,\mathrm{MB/s}}, 1.3\,\mathrm{s}\right)}_{\text{GEOCom to Moon Relay 1}} + \underbrace{\max\left(\frac{1\,\mathrm{GB}}{622\,\mathrm{MB/s}}, 0.120\,\mathrm{s}\right)}_{\text{Moon Relay 1 to Moon Relay 2}} = \boxed{35.3\,\mathrm{s}}.$$

Similarly, the second Mars relay takes

$$\underbrace{\max\left(\frac{1\,\mathrm{GB}}{2\,\mathrm{GB/s}}, 0.003\,\mathrm{s}\right)}_{\text{Earth antenna to LEOCom}} + 50\underbrace{\max\left(\frac{1\,\mathrm{GB}}{2\,\mathrm{GB/s}}, 0.015\,\mathrm{s}\right)}_{\text{LEOCom to LEOCom}}$$
$$+ \underbrace{\max\left(\frac{1\,\mathrm{GB}}{1.2\,\mathrm{GB/s}}, 0.120\,\mathrm{s}\right)}_{\text{LEOCom to GEOCom}} + 5\underbrace{\max\left(\frac{1\,\mathrm{GB}}{1.2\,\mathrm{GB/s}}, 0.090\,\mathrm{s}\right)}_{\text{GEOCom to GEOCom}}$$
$$+ \underbrace{\max\left(\frac{1\,\mathrm{GB}}{622\,\mathrm{MB/s}}, 21\,\mathrm{min}\right)}_{\text{GEOCom to Moon Relay 1}} + \underbrace{\max\left(\frac{1\,\mathrm{GB}}{622\,\mathrm{MB/s}}, 0.120\,\mathrm{s}\right)}_{\text{Moon Relay 1 to Moon Relay 2}} = \boxed{21.5\,\mathrm{min}}.$$

Both times are within $30\,\mathrm{min}$.

Finally, we consider antenna equipment on the moon and Mars. The update is of the form of ten $10\,\mathrm{GB}$ files. At the minimum bandwidth of $622\,\mathrm{MB/s}$, each node takes at most $(10\,\mathrm{GB})/(622\,\mathrm{MB/s}) = 16\,\mathrm{s}$. If all ten files were sent to the same node, then this causes $160\,\mathrm{s}$ of queue buildup from routine messages, which is $160\,\mathrm{s} \cdot 0.0166\,\mathrm{min/s} \cdot 1\,\mathrm{kB/min} = 2.656\,\mathrm{kB}$. After the node processes these software update bundles, it can process the $2.656\,\mathrm{kB}$ in at most $0.016\,\mathrm{s}$ by Section 5.1.1.

Each file to the moon takes about

$$\underbrace{\max\left(\frac{10\,\mathrm{GB}}{2\,\mathrm{GB/s}}, 0.003\,\mathrm{s}\right)}_{\text{Earth antenna to LEOCom}} + 50\underbrace{\max\left(\frac{10\,\mathrm{GB}}{2\,\mathrm{GB/s}}, 0.015\,\mathrm{s}\right)}_{\text{LEOCom to LEOCom}} + \underbrace{\max\left(\frac{10\,\mathrm{GB}}{1.2\,\mathrm{GB/s}}, 0.120\,\mathrm{s}\right)}_{\text{LEOCom to GEOCom}}$$
$$+ 5\underbrace{\max\left(\frac{10\,\mathrm{GB}}{1.2\,\mathrm{GB/s}}, 0.090\,\mathrm{s}\right)}_{\text{GEOCom to GEOCom}} + \underbrace{\max\left(\frac{10\,\mathrm{GB}}{622\,\mathrm{MB/s}}, 1.3\,\mathrm{s}\right)}_{\text{GEOCom to Moon Relay 1}}$$
$$+ \underbrace{\max\left(\frac{10\,\mathrm{GB}}{622\,\mathrm{MB/s}}, 0.120\,\mathrm{s}\right)}_{\text{Moon Relay 1 to Moon Relay 2}} + \underbrace{\max\left(\frac{10\,\mathrm{GB}}{622\,\mathrm{MB/s}}, 0.120\,\mathrm{s}\right)}_{\text{Moon Relay 2 to Moon antenna}} = \boxed{5.9\,\mathrm{min}}.$$

On the other hand, each Mars-bound file takes

$$\underbrace{\max\left(\frac{10\,\mathrm{GB}}{2\,\mathrm{GB/s}}, 0.003\,\mathrm{s}\right)}_{\text{Earth antenna to LEOCom}} + 50\underbrace{\max\left(\frac{10\,\mathrm{GB}}{2\,\mathrm{GB/s}}, 0.015\,\mathrm{s}\right)}_{\text{LEOCom to LEOCom}} + \underbrace{\max\left(\frac{10\,\mathrm{GB}}{1.2\,\mathrm{GB/s}}, 0.120\,\mathrm{s}\right)}_{\text{LEOCom to GEOCom}}$$
$$+ 5\underbrace{\max\left(\frac{10\,\mathrm{GB}}{1.2\,\mathrm{GB/s}}, 0.090\,\mathrm{s}\right)}_{\text{GEOCom to GEOCom}} + \underbrace{\max\left(\frac{10\,\mathrm{GB}}{622\,\mathrm{MB/s}}, 21\,\mathrm{min}\right)}_{\text{GEOCom to Mars Relay 1}}$$
$$+ \underbrace{\max\left(\frac{10\,\mathrm{GB}}{622\,\mathrm{MB/s}}, 0.120\,\mathrm{s}\right)}_{\text{Mars Relay 1 to Mars Relay 2}} + \underbrace{\max\left(\frac{10\,\mathrm{GB}}{622\,\mathrm{MB/s}}, 0.120\,\mathrm{s}\right)}_{\text{Mars Relay 2 to Mars antenna}} = \boxed{26.6\,\mathrm{min}}.$$

The ten files can be sent along the same path or along different routes, which adds at most $160\,\mathrm{s}$ from bandwidth. This falls within the 30 minute requirement.

5.2. Worst-Case Analysis

In Section 5.1, we analyze the performance of our system in normal circumstances. Here, we predict some worst-case scenarios and give a treatment on the performance of our system in these cases.

5.2.1. Failure of Relays

The four relays in the SolarNet network are extremely crucial in sending transmissions. In fact, most transmissions to and from the moon are sent through the two moon relays, and all transmissions to and from Mars are sent through the two Mars relays. This is shown in Figure 1.

When one of the Mars relays become nonfunctional, users on Earth and on Mars cannot communicate with each other until the relay is repaired. This can cause massive queue buildups at the GEOComs since they cannot transmit bundles to the relays. Our system is able to handle this critical situation by detecting nonfunctional devices using our heartbeat system. At least one of the other Mars relay, the nearest GEOCom, or the Mars antenna is able to detect that the relay is unresponsive since the relay is not responding to their heartbeat messages. The device then reports back to Earth that the relay is defunct, from which administrators can take appropriate action to return functionality of the relay.

5.2.2. Land Telemetry

In extreme emergencies, land telemetry data from the two land telescopes can be transmitted to Earth using SolarNet. The data consists of twenty $2\,\mathrm{GB}$ files. At the minimum bandwidth of $2\,\mathrm{GB/s}$, each $2\,\mathrm{GB}$ file takes $1\,\mathrm{s}$ for the node to send out. If all twenty files were sent to the same node, then this causes $20\,\mathrm{s}$ of queue buildup from routine messages, which is insignificant from previous analysis.

Assuming it is transmitted through 50 LEOComs before reaching the desired Earth antenna, the file takes

$$\underbrace{\max\left(\frac{2\,\mathrm{GB}}{2\,\mathrm{GB/s}}, 0.015\,\mathrm{s}\right)}_{\text{Land Telemetry to LEOCom}} + 50\underbrace{\max\left(\frac{2\,\mathrm{GB}}{2\,\mathrm{GB/s}}, 0.015\,\mathrm{s}\right)}_{\text{LEOCom to LEOCom}} + \underbrace{\max\left(\frac{2\,\mathrm{GB}}{2\,\mathrm{GB/s}}, 0.003\,\mathrm{s}\right)}_{\text{LEOCom to Earth antenna}} = \boxed{52\,\mathrm{s}}$$

to be sent from the telemetry LEOCom to Earth. The twenty files can be sent along the same path or along different routes. This only contributes about $20\,\mathrm{s}$ from bandwidth, which falls within the $10\,\mathrm{min}$ warning window. Therefore, emergency land telemetry data transmissions do not significantly impact the performance of routine message transmissions.

5.2.3. Simultaneous Telemetry and Updates

Consider the absolute worst case where the system is simultaneously flooded with land telemetry data, solar telemetry data, and software updates. Our system prioritizes transmitting land telemetry data first with the goal of providing the $10\,\mathrm{min}$ warning and saving lives. Then, the system focuses on transmitting solar telemetry data with the goal of providing the $5\,\mathrm{h}$ warning. Finally, the system transmits the necessary updates.

If all the land telemetry bundles, solar telemetry bundle, and software updates were sent to the same node, then the queue buildup of routine messages is $20\,\mathrm{s} + 20 \cdot 0.166\,\mathrm{s} + 160\,\mathrm{s} = 183.3\,\mathrm{s} \approx 3\,\mathrm{min}$. This is $183.3\,\mathrm{s} \cdot 0.0166\,\mathrm{min/s} \cdot 1\,\mathrm{kB/min} = 3.04\,\mathrm{kB}$, which can be processed in at most $0.016\,\mathrm{s}$ by Section 5.1.1. As such, the queue buildup caused by simultaneous mission-critical messages is negligible.

5.3. Scalability

We now consider the scalability of our system, since NASA plans to expand future space operations. The heartbeat system may not work as well in a larger network, since it necessitates frequent messages between neighboring nodes; as the network grows, the number of heartbeats being sent could significantly increase traffic on the network despite their small sizes. However, a sufficient increase in node count without physical expansion of the network to bodies other than the Moon or Mars could potentially reduce the need for the overhead of a more frequent heartbeat system, since there would be more backup nodes to compensate when their neighbors fail.

Due to developers and most users being based on Earth, it is also necessary for Earth to be a central point in SolarNet. Expansion of the network beyond Mars to even more distant solar system bodies with more extreme transit times may be hindered by aspects of our system such as Earth being the origin point of all updates and commands, which have to travel through the network itself to their destinations.

6. Conclusion

This proposed implementation for SolarNet supplements the pre-existing system protocols that prioritizes reliability and performance. We tradeoff the latency of lower-priority transmissions to optimize the performance of high-priority transmissions. At the physical hardware level, we allocate space for different categories of data storage and memory to increase nodes' ability to keep bundles. Our heartbeat system for detecting nonfunctional devices allows the network to gracefully adapt to occasional outages. Moreover, the Bundle Protocol gives specifications on bundle fragmentations and their copies, promoting the reliability of the system. Finally, the routing protocol optimizes the route for each bundle based on its properties, improving the system's performance.

Our system design is also motivated by several use cases that affect many communities and users. However, before implementing this design, we recommend further actions that improve the scalability of the network, such as decentralizing the design such that it is not based on Earth. Nonetheless, our robust design, including enhanced reliability measures and fast performance, ensures that the impacts to these communities are beneficial.

7. Author Contributions

All authors collaborated closely in making design choices for the system and writing this paper. Jessica made most of the figures and wrote the System Overview section as well as the Memory and Storage Allocation, Detecting Nonfunctional Devices, and File Systems subsections. Daph wrote the Queues and Deletion Policy, File Systems, Bundle Protocol Fragmentation, and Routing Protocol. Allen performed the analysis and evaluation of the design, generated the tables and TikZ figure, and wrote the Use Cases and Evaluation sections.

8. Acknowledgements

The authors would like to thank Olivia Brode-Rogers for her in-depth feedback on our design as well as being a wonderful recitation instructor throughout the entire semester. The authors would also like to thank Emeka Echezona for his support as our TA and his willingness to indulge our memes in his inbox. In addition, this paper would not be possible without Rachel Molko's writing guidance and feedback in tutorials. Finally, the authors would like to thank Dr. Katrina LaCurts for her enthusiastic lectures—her passion for teaching this subject shines through and made the class all the more enjoyable.

References

[1] D. M. Ritchie and K. Thompson, "The UNIX Time-Sharing System," *Association for Computing Machinery Inc*, 1974.

[2] J. Bonwick, M. Ahrens, V. Henson, M. Maybee, and M. Shellenbaum, "The Zettabye File System," *Usenix Symposium on File and Storage Technologies*, 2003.

[3] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The Google file system,” *SIGOPS Oper. Syst. Rev.*, 2003.