A Proposed Clock Synchronization Method for the Solar System Internet

Michael Moy
Colorado State University
michael.moy@colostate.edu

Alan Hylton
NASA Goddard
alan.g.hylton@nasa.gov

Robert Kassouf-Short
NASA Glenn
robert.s.short@nasa.gov

Jacob Cleveland
NASA Glenn
jacob.a.cleveland@nasa.gov

Jihun Hwang
Purdue University
hwang102@purdue.edu

Justin Curry
University at Albany, SUNY
jmcurry@albany.edu

Mark Ronnenberg
Indiana University
maronnen@iu.edu

Miguel Lopez
University of Pennsylvania
mlopez3@sas.upenn.edu

Oliver Chiriac
University of Oxford
oliver.chiriac@trinity.ox.ac.uk

***Abstract*—Networked communications in space are necessary to achieve scalability in terms of the number of communicating nodes but also in terms of the overall system complexity. A key component to such a system is the ability to synchronize clocks, which is the focus of this paper. The so-called Solar System Internet (SSI) will be built upon Delay Tolerant Networking (DTN), which, in analogy to the Internet Protocol (IP), can be considered a suite of protocols necessary for networking in the space domain. Therefore, our goal is to extend this suite to include a DTN clock synchronization capability, analogous to the Network Time Protocol (NTP) used in the Internet. A motivating example of a network in space is NASA's LunaNet, a vision for a multi-hop multi-path network extending to the moon wherein not all nodes will have direct connections to an authoritative reference clock. In this paper, we propose a general clock synchronization methodology and algorithm that could be used for LunaNet as well as more elaborate time-varying networks.**

In recent years, DTN has benefited from modeling efforts founded on the mathematical tool of *sheaves*. Here we continue this work to provide an approach to clock synchronization. Due to the time-varying nature of space networks, absolute consensus is not possible. However, the *sheaf Laplacian* provides a practical, distributed approach to approximating consensus by allowing data to diffuse through the network. In particular, the sheaf Laplacian is readily computable, lending our approach to implementation.

Our approach is well suited to handle the difficulties of space networks. For instance, differences in clock accuracy mean certain nodes are more authoritative than others; we can account for these differences through hierarchies in the network, generalizing the strata in NTP. Furthermore, just as error estimation is an integral part of NTP, we are able to give concrete error bounds for our approach. Indeed, different applications (e.g., communications schedules, pointing, navigation, distributed science) will have different requirements, hence it is necessary to maintain clocks within a given tolerance. We outline some of the necessary steps to turn our approach into a practical network protocol that could be used in DTN, and we conclude the paper with suggestions for future research.

TABLE OF CONTENTS

1. INTRODUCTION

The large number of spacecraft expected over the next several decades has motivated research into networked communications in space. A computer network in space must be able to handle the delays and disruptions inherent in space communications, and as such, the network protocols used on Earth must be adapted before being applied in space. Recent research addressing these problems has produced *Delay Tolerant Networking* (DTN), a collection of protocols and techniques that aim to help realize a Solar System Internet (SSI) [1–4]. NASA is actively researching DTN for use in future missions; to this end, the High-Rate Delay Tolerant Networking (HDTN) project at NASA Glenn Research Center is one implementation of DTN currently being developed and tested [5–10]. DTN operates as an overlay network over an extremely heterogeneous setting, where one-way light times can vary from milliseconds to hours; a consequence is that some links (and hence network segments) can feature reactivity and others must be proactive. In particular, there may be portions of the overall network that can utilize discovery-based routing, but schedule-based routing will always be a component. Therefore, it is essential for routing (and antenna pointing, attitude control, etc.) that the nodes agree on what time it is.

In this paper, we propose a method for automated clock synchronization in a space network (Solar System Internet, in particular) that fits into the current framework of DTN and could be incorporated into implementations such as HDTN in the near future.

Certain aspects of clock synchronization in space are, of course, well understood. Indeed, the Messenger and New Horizon missions were able to synchronize their clocks to UTC within sub-millisecond and sub-ten-milliseconds, respectively [11]. Satellite tracking and ranging techniques allow us to determine the distance between a satellite and a ground station with a high degree of precision, which in turn allows for a precise exchange of clock data that takes into account the light speed delay. Relativity predicts the differing speeds at which clocks in orbit run compared to those on

Earth. And the increasing accuracy of clocks in space is exemplified by the Deep Space Atomic Clock (DSAC) [12–14], launched in 2019. The clock synchronization problem expands its presence beyond the "classic" world; the quantum space network—including the deep space network—is expected to provide a clock synchronization service with the error only within a degree of picoseconds [15–17]. This paper does not aim to improve upon any of these aspects of timekeeping in space, but rather addresses the mathematical theory needed to make these existing techniques extensible across networked communications. Specifically, our goal is to automate clock synchronization in space networks in a way that builds on existing technology and can be incorporated into SSI-scaled DTNs.

These goals are motivated in part by clock synchronization in the Internet, governed by the *Network Time Protocol* (NTP) [18,19]. Based on the ability of computers to exchange clock data over the Internet, NTP provides a systematic way for a computer to update its clock based on clock values and error estimates obtained from multiple sources. As with other Internet protocols, NTP cannot be applied directly to space networks, as it relies on properties of Internet communications that do not hold in space. In particular, it assumes some amount of consistency in the time it takes a packet to travel along a given path in the Internet, whereas in space networks, an end-to-end path does not always exist and travel times can be more variable. Additionally, a node using NTP will typically obtain and process an ample amount of clock data from other nodes in the network, whereas in space, links are intermittent and less frequent, and updates will likely need to be made using limited clock data; put another way, communications is expensive, and despite the importance of clock synchronization it cannot afford to be *too* chatty. Our method accounts for these differences by only requiring clock data to be exchanged across direct links, i.e. between adjacent nodes in the network, and by updating a node's clock using only a single clock value from each of its neighbors. It also requires minimal computation by the network nodes, an important feature in space networks, where computational power follows a stringent budget.

Clock synchronization methods for DTN and space communications based on NTP have been actively studied and proposed. The Double-pairwise Time Protocol (DTP) by Ye and Cheng [20] is one of them; however, DTP appears to require the presence of the time server within a certain distance, which may not be possible for SSI-scaled DTNs. Thanks to its robustness and fault tolerance, even direct applications of NTP have shown successful results, although in limited-scaled environments such as the Proximity Link [21–23]. The technical introduction to the previous deployment of NTP for space missions is available in [24, Chapter 17-18], and see [25] for further explanations on why NTP or protocols that rely on the existence of time servers may not be sufficient for upcoming space missions.

In spite of the aforementioned restrictions, which we believe are realistic and are necessary constraints on a clock synchronization protocol for space, we obtain theoretical guarantees of the performance of our method. In particular, we show that we can maintain synchronized clocks to within certain error tolerances, which can be computed given knowledge of the errors introduced into the network (which most importantly include those resulting from inaccuracies of clocks in the network and the exchange of clock data between nodes). Our method is based on a linear model of diffusion, which lends itself to a simple distributed approach and has the benefit of requiring only minimal computations by spacecraft computers.

Such a method was suggested but not thoroughly developed in [25] by many of the same authors. [25] also provides a summary of other clock synchronization approaches proposed in various settings. These past approaches, being designed for different types of networks, generally were based on assumptions that would not hold for delay-tolerant networks in space. For instance, the protocol given by Choi et al. in [26] and [27] addresses clock synchronization in terrestrial DTNs, which behave differently from space networks, especially the large-scaled ones such as SSI. Our method is somewhat related to the simple approach proposed by Sasabe and Takine [28] or the asynchronous diffusion (AD) method such as the one by Li and Rus [29], in which nodes average their clock values when they communicate, similar to the traditional methods of approximating consensus in asynchronous and/or distributed settings [30–32]. Indeed, in our method, nodes update by taking a carefully designed weighted average. Our method stands in contrast to these previous approaches because of our focused application—delay tolerant networks in space—and because of the particular mathematical approach we take.

We give the motivation for our method and establish some assumptions on networks in the following section (Section 2), before describing the theory in detail in Section 3. While this paper is dedicated to developing the method and not to the intricacies of designing a network protocol, Section 4 outlines future steps needed to turn this method into a network protocol that could be incorporated into DTN. We then conclude with additional thoughts on future work in Section 5.

2. PRELIMINARIES AND CONVENTIONS

Assumptions about the network

Before discussing clock synchronization techniques, we will need to make certain assumptions about the networks we consider and the other networking protocols involved. Our goal will be to base our assumptions on the requirements of space networks in the near future, primarily those operated by a single entity such as NASA and designed using current approaches to DTN. NASA's LunaNet [33], the envisioned communications network between the moon and Earth, serves as a motivating example of a network, and NASA Glenn Research Center's HDTN [10] serves as a current implementation of DTN.

We assume throughout that our network follows the basic principles of DTN and use this to guide our assumptions about the network's capabilities. With the current schedule-based approach to routing used in DTN [8, 34], it is assumed that each node has been sent a schedule of when links will be available, which has been created on Earth. Our method does not rely on the details of the routing algorithm, so it could be applied alongside future schedule-based routing approaches as well. We will assume that in addition to routing data, parameters of a clock synchronization protocol can also be sent to each node; our protocol will require only a minimal amount of data to be sent, so this should not place a large burden on the existing process. Some parameters will be introduced in Section 3, and in Section 4, we include a more thorough list of parameters we expect to be included in a future protocol.

As mentioned above, we assume our network will have more limited exchange of clock data than in the Internet: we

assume clock data is only exchanged across direct links, with a node receiving data from each neighbor on a regular basis, on the order of once per day. The requirement of direct links could be adjusted in the future to allow for cases of predictable multi-hop routes, including the use of "bent pipes" like TDRS, but for now we will assume only adjacent nodes exchange clock data. For assets near Earth, including ground stations and satellites on both the Earth and the moon, we assume standard techniques for ranging can be used to determine the distance between a pair of nodes, and thus to exchange clock data between them with a precise measurement of the light speed delay. Such communications near Earth will form the majority of applications of DTN in the immediate future. To handle the automated exchange of clock data over larger distances, we have the option of using an estimated light speed delay between nodes that is computed on Earth and included as part of the routing information distributed to network nodes – this comes at the cost of possibly introducing larger errors. A protocol implementing our method will need to allow for multiple such methods of exchanging clock data, as we discuss in Section 4.

We must keep in mind that a space network may contain clocks with widely varying accuracies. These may include devices using a typical crystal oscillator with a frequency error on the order of 100 parts per million (ppm) or 10^{-4}, an oven controlled oscillator (OCXO) with an error on the order of one part per billion (ppb) or 10^{-9}, or an atomic clock with an error on the order of 10^{-12}. These drastically different accuracies mean we need to take care to update a clock's value according to clocks with comparable or better accuracy. As a first approach, we suggest organizing the network into different regions or "strata," to borrow a term from NTP, in which clocks have comparable accuracies – this is described in Section 4. For finer control, our work on error tolerances may provide some insight on how to choose parameters that handle different levels of accuracy, which we leave as a problem for future work (see Section 5).

Model of the Network and Clock Data

Here we give some conventions for our modeling of networks and the clock data exchanged by adjacent nodes. Networks will be modeled by simple, undirected graphs. An edge between a pair of nodes will indicate that there are regular but possibly intermittent periods of time during which the two nodes can communicate.

To model the clock data kept by each node, let t be a variable representing true time (in practice, UTC time), and for each vertex v, let $C_v(t)$ be the clock value of v. Since $C_v(t)$ will inherently fluctuate from the true time t, the job of a clock synchronization protocol is to periodically correct this clock values. Let $x_v = x_v(t) = C_v(t) - t$ represent the clock error at vertex v at time t, and let $x = x(t)$ be the vector with a component x_v for each vertex v. Our goal is a protocol for updating the clock values C_v such that x remains close to the zero vector.

Of course, the errors in time x_v are never actually known in the network, but with the right approach, we can still adjust them toward zero. The key observation is that the difference in errors between two vertices is the same as the difference in their clock values:

$$\begin{aligned} x_v(t) - x_w(t) &= (C_v(t) - t) - (C_w(t) - t) \\ &= C_v(t) - C_w(t). \end{aligned}$$

This shows that if updates to clock values are computed entirely in terms of the difference in clock values, we are justified in mathematically modeling the system in terms of the vector x, without mention of the clock values $C_v(t)$. These differences are in fact all that are needed to model diffusion in a network, and our method will be based on the idea of allowing clock errors to diffuse through the network so that clocks converge to consensus. In the following section, we explain some of the initial mathematical motivation, which will be further generalized shortly.

Motivation: Diffusion and the sheaf Laplacian

In physical settings, diffusion is typically modeled by the heat equation $\frac{\partial u}{\partial t} = \alpha \Delta u$, where $\alpha > 0$ is the "diffusivity constant." Previous works by Hansen and Ghrist [35, 36] have considered a discrete analog of this equation to model diffusion on a graph: the function u is replaced by a vector x of values assigned to the vertices and the Laplacian Δ is replaced by a matrix L to produce an ordinary differential equation[2]

$$\frac{dx}{dt} = -\alpha L x.$$

In the simplest case, the matrix L is the *graph Laplacian*, which is a well-studied object in spectral graph theory. The heat equation with the graph Laplacian can be visualized as allowing heat to flow along the edges of the graph, so that each vertex changes in temperature according to the relative temperature of its neighbors. For more general modeling, L may be a *sheaf Laplacian*, which allows different edges to conduct heat differently. The sheaf Laplacian is also flexible enough to allow each vertex to store a vector of values rather than a single value, thus modeling the evolution of multiple interdependent quantities rather than a single temperature. See Appendix A for a brief introduction to the sheaf Laplacian. Similarly to how the Laplacian Δ in the original heat equation is a differential operator, the graph Laplacian and sheaf Laplacian are computed locally, with each vertex using only the differences of the components of x at the vertex and its neighbors. For our purposes, this allows them to be computed in a distributed fashion in a computer network; each node needs only knowledge of its neighbors' information.

To apply these ideas to clock synchronization, we aim to let the clock errors x_v diffuse through the network. Recall that we are assuming a network with intermittent links; as adjacent vertices can only exchange clock values during the periods during which they are in contact, we are forced to use a discrete-time approximation of the heat equation. Euler's method for the heat equation with the sheaf Laplacian is defined by $x_{n+1} = x_n - h\alpha L x_n$, where h is a chosen length of a time step. It can be shown that as long as $h\alpha$ is sufficiently small, the sequence $\{x_n\}$ converges to a steady state for any initial value x_0. Moreover, it tolerates errors well; error introduced at each step produces a predictable fluctuation around the steady state. Adjusting this method slightly allows us to introduce nodes whose values do not change, modeling reference clocks that are maintained by an outside method. These reference clocks are analogous to boundary conditions for a differential equation. In the following section, we present a generalization of this approach: we replace the matrix L by a more general type of matrix that retains the properties important to our setting. Throughout, we will keep

[2]Because of the common choices for the matrix L, the negative sign is needed to produce the correct discrete analog of traditional heat equation. See the discussion in Appendix A.

the sheaf Laplacian in mind as an important special case of this more general matrix, describing how particularly useful results apply when using a sheaf Laplacian.

3. THE METHOD AND ANALYSIS

Definitions

We begin with a linear model of clock synchronization in which adjacent nodes compare and adjust their clock values. Mathematically, we will work exclusively with finite graphs and finite-dimensional vector spaces (with real or complex coefficients). Let G be a graph, let B be a subset of vertices, referred to as "boundary" vertices, and let Y consist of all vertices of G not in B. Let $C^0(G)$ be the real vector space spanned by the vertices of G. We will write elements as column vectors with components indexed by vertices: if $x \in C^0(G)$ and v is a vertex of G, we will write the v component of x as x_v. The component x_v is interpreted as the difference between the clock value of v and the true time, and we will refer to this as the value of x over v. Later we will consider sequences $\{x_n\}$ in $C^0(G)$, where terms are also written with subscripts; the meaning of subscripts should be clear from context, and when both meanings are needed, the v component of and element x_n of a sequence will be written $(x_n)_v$. Linear maps from $C^0(G)$ to itself will be represented by matrices, with entries indexed by pairs of vertices. We also let $C^0(B)$ and $C^0(Y)$ be the subspaces spanned by the vertices in B and Y respectively.

For each edge $e = \{v, w\}$, suppose we have a pair of "weights" $a_{v,w}, a_{w,v} \in \mathbb{R}$, not necessarily equal. We define a matrix M that will specify how clocks are adjusted. For each $v \in Y$, define row v of M by setting

$$M_{v,w} = -a_{v,w}$$

for each w adjacent to v, and

$$M_{v,v} = \sum_{w \text{ adjacent to } v} a_{v,w}.$$

Let all other entries of M be zero.

We can use this matrix to model diffusion in a network: letting $x = x(t) \in C^0(G)$, we have the differential equation

$$\frac{dx}{dt} = -\alpha M x \tag{1}$$

similar to the heat equation[3], where the constant[4] $\alpha > 0$ mimics the diffusivity constant in the usual heat equation. For any vertex v in B, the v component of $\frac{dx}{dt}$ is always zero because the corresponding row of M is zero. This means the value of x over v does not change, although it may influence the values of its neighbors that lie in Y. Thus, the components of x lying over B are analogous to a boundary condition in a partial differential equation.

[3]This differential equation can be most reasonably viewed as modeling diffusion or heat flow when all weights $a_{v,w}$ are positive. In this case, the definition of M shows that a difference in values of x over two adjacent vertices yields "heat flow" from the higher valued vertex to the lower.

[4]Note that α is redundant, as it simply scales all the weights $a_{v,w}$. However, it is helpful to keep it for our upcoming analysis. For instance, it provides precise language for Theorem 2 on convergence, and in certain cases, once weights $a_{v,w}$ are chosen, we can show there is an optimal choice of α.

To apply this idea of diffusion to our setting of clock synchronization in a network with intermittent connections, we need a discrete-time analog of Equation (1). Suppose we can divide time into steps of length $h > 0$ such that for each edge of G, its two nodes are able to exchange clock data every time step[5]. We apply Euler's method with these time steps of length h, obtaining

$$x_{n+1} = x_n - h\alpha M x_n. \tag{2}$$

An initial condition x_0 determines the values of all x_n, and the collection of values of x_0 over B can still be interpreted as a boundary condition. Note that we are free to choose any $\alpha > 0$; we will discuss how to choose α later on.

Importantly, the structure of the matrix M allows for distributed computation: each node only needs data from its neighbors to compute its component of x_{n+1}. This follows from the definition of M, since for each $v \in Y$, we have

$$(Mx_n)_v = \sum_{w \text{ adjacent to } v} a_{v,w}((x_n)_v - (x_n)_w). \tag{3}$$

This further shows that $(Mx_n)_v$ can be computed with just the knowledge of the differences $(x_n)_v - (x_n)_w$, as we required in Section 2. By our assumption that each edge is active at least once per time step, we will require that during the n^{th} time step, each node v gathers clock values from its neighbors, uses them to calculate the differences $((x_n)_v - (x_n)_w)$, and uses them to calculate $(x_{n+1})_v$ and update its clock accordingly at the end of the time step.

We can simplify Equation (2) by setting $A = I - h\alpha M$, so that

$$x_{n+1} = Ax_n. \tag{4}$$

This gives the explicit expression $x_n = A^n x_0$. It can also be helpful to express the update just for the values of x over Y, since the values over B do not change. Split into block matrices corresponding to the decomposition $C^0(G) = C^0(Y) \oplus C^0(B)$, the equation above becomes

$$\begin{bmatrix} y_{n+1} \\ b_{n+1} \end{bmatrix} = \begin{bmatrix} A_{Y,Y} & A_{Y,B} \\ 0 & I \end{bmatrix} \begin{bmatrix} y_n \\ b_n \end{bmatrix}, \tag{5}$$

where $x_n = \begin{bmatrix} y_n \\ b_n \end{bmatrix}$ and we have used the fact that rows of M indexed by vertices in B consist of zeros. Thus, $b_n = b_0$ for all n, so we obtain

$$y_{n+1} = A_{Y,Y} y_n + A_{Y,B} b_0. \tag{6}$$

Here, the y_n are given explicitly by

$$y_n = A_{Y,Y}^n y_0 + (A_{Y,Y}^{n-1} + \cdots + A_{Y,Y} + I) A_{Y,B} b_0.$$

Example 1. For a simple example, let G be the complete graph on three vertices u, v, and w, and let B consist of the

[5]In practice, the choice of h may influence the decision of which intermittent links to include as edges in the graph. See Section 4 for further discussion.

single vertex w. Then,

$$M = \begin{bmatrix} a_{u,v} + a_{u,w} & -a_{u,v} & -a_{u,w} \\ -a_{v,u} & a_{v,u} + a_{v,w} & -a_{v,w} \\ 0 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 - h\alpha(a_{u,v} + a_{u,w}) & h\alpha a_{u,v} & h\alpha a_{u,w} \\ h\alpha a_{v,u} & 1 - h\alpha(a_{v,u} + a_{v,w}) & h\alpha a_{v,w} \\ 0 & 0 & 1 \end{bmatrix}$$

Given a boundary condition $(x_0)_w = b$, Equation (6) becomes

$$y_{n+1} = \begin{bmatrix} 1 - h\alpha(a_{u,v} + a_{u,w}) & h\alpha a_{u,v} \\ h\alpha a_{v,u} & 1 - h\alpha(a_{v,u} + a_{v,w}) \end{bmatrix} y_n + b \begin{bmatrix} h\alpha a_{u,w} \\ h\alpha a_{v,w} \end{bmatrix}$$

Choosing $\alpha = 1/h$, $a_{u,v} = 0$, $a_{v,u} = 0$, $a_{u,w} = 1$, $a_{v,w} = 1$ models the obvious process in which vertices u and v do not exchange any information and simply update their clocks to match w. The equation above becomes

$$y_{n+1} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} y_n + b \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

so that $y_n = \begin{bmatrix} b \\ b \end{bmatrix}$ for all $n \geq 1$. Thus, in this case, we can achieve perfect synchronization after just one time step – this is a result of every vertex being adjacent to the single boundary vertex.

Choice of Weights

Notice that since we have the freedom to choose the weights $a_{v,w}$ in the definition of M, Equation (2) and its reformulations do not serve to model behavior in the network, but rather give a prescriptive choice of how our clock synchronization method will operate. This raises the possibility of optimizing the choice of weights to produce the optimal behavior. We list some ways of formulating this optimization problem later, after developing the necessary framework in the analysis below.

For now, we comment that by placing certain restrictions on these weights, we obtain important special cases; we will see later on that certain special cases can lead to better theoretical properties and in particular may be more practical for large networks. In the most restrictive case, if the boundary B is empty and if $a_{v,w} = 1$ for all adjacent v and w, the matrix M is the graph Laplacian; if instead we only require symmetric weights $a_{v,w} = a_{w,v} > 0$, then it is the more general sheaf Laplacian, arising from a certain sheaf on the graph (see Appendix A). In these cases, M is symmetric and positive semi-definite. This implies it is diagonalizable by orthogonal matrices and has all real, nonnegative eigenvalues. If we allow B to be nonempty, these properties are still true of the square block of M corresponding to the vertices in Y. These properties lead to particularly simple behavior. We will frequently mention how our results apply in the case that $a_{v,w} = a_{w,v} > 0$, which we will refer to as the "sheaf Laplacian case" (even when the boundary B is nonempty).

Some less restrictive requirements on the weights are also useful. For instance, we can require that $a_{v,w} = a_{w,v}$ but allow negative values. Then the block of M corresponding to the vertices of Y is symmetric and thus is diagonalizable by orthogonal matrices and has real eigenvalues. In some cases, these matrices may still have all nonnegative eigenvalues, in which case many of our techniques will still apply – we mention this as a possibility for future work, but we will not address this case much here. Finally, for the case we will use most often, we instead allow for $a_{v,w}$ and $a_{w,v}$ to be different, but require that they be positive. We will prove our main results with these assumptions, as they are reasonably general but retain some of the useful properties of M.

Convergence to Steady States

Here we turn to the theoretical analysis of our method. Readers who are willing to trust the results can skim the mathematical details, and those wanting to move directly to understanding a protocol based on these ideas can glance ahead to Section 4. However, many of the benefits of our method follow from this analysis, and future work on optimizing performance will likely depend on the work below.

Referring to Equations (2) and (4), we will call any vector x such that $x = Ax = x - h\alpha Mx$ a *steady state* or a *steady-state solution*. These are of course the eigenvectors of A with eigenvalue 1, which can equivalently be defined as vectors in the kernel of M. They represent consensus in the network, in the sense that the value over each vertex agrees with a weighted average of its neighbor's values, defined by the weights $a_{v,w}$. Note that $(1, \ldots, 1)^T$ is always a steady-state solution, representing the exact agreement between all vertices. Different steady-state solutions arise if the values over the boundary vertices disagree. However, the following result shows that all steady-state solutions can be viewed as consensus to within the level of accuracy of the boundary nodes.

Theorem 1. *Suppose that G is connected and all weights $a_{v,w}$ are positive. If x is a steady-state solution of Equation* (4)*, then the component of x with maximal absolute value occurs over a vertex in B.*

Proof. If x is a steady-state solution, then $Mx = 0$. Suppose $v \in Y$ and x_v is a component of x with maximal absolute value; we must show this value is also attained by a vertex in B. We have

$$0 = (Mx)_v = \sum_{w \text{ adjacent to } v} a_{v,w}(x_v - x_w),$$

and since x_v is the component with maximal absolute value, we have either $x_v - x_w \geq 0$ for all neighbors w or $x_v - x_w \leq 0$ for all neighbors w. Since $a_{v,w} > 0$, we must in fact have $x_v - x_w = 0$ for all w, so v shares its value with all of its neighbors. Since G is connected, applying this fact repeatedly along a path from v to a node in B shows the maximal absolute value will also be attained by a node in B. □

Next, we establish that the sequence $\{x_n\}$ defined by Equation (4) converges under certain assumptions on the weights $a_{v,w}$ and the constant α. We will assume that nonnegative weights $a_{v,w}$ have been fixed and show that choosing any small enough $\alpha < 0$ ensures convergence. Convergence depends on the eigenvalues of A; to begin, note that if λ

is an eigenvalue of M, then $1 - h\alpha\lambda$ is an eigenvalue of A. More specifically, if x is a generalized eigenvector of M with eigenvalue λ and rank k, then it is also a generalized eigenvector of A with eigenvalue $1 - h\alpha\lambda$ and rank k. We thus have a shared canonical basis that can be used to put M and A into Jordan canonical form, which we use in the following theorem.

Theorem 2. *In the notation above, suppose each $a_{v,w}$ is nonnegative and we have chosen $\alpha \in \left(0, \frac{1}{hd}\right)$, where d is the largest diagonal entry of M. Then given any x_0, the sequence $\{x_n\}$ converges to a vector in $\ker M$ at an exponential rate. Specifically, the limit x_* is given by projecting x_0 onto $\ker M$ in the canonical basis of M, and $\|x_n - x_*\| = O(n^c \mu_{\max}^{n-c})$, where $\mu_{\max}$ is the magnitude of the largest eigenvalue of A not equal to 1 and $c \leq \operatorname{rank}(M) - 1$.*

The statement of the theorem holds for any norm, since all norms on a finite-dimensional real vector space are equivalent. In the sheaf Laplacian case, we can strengthen the rate of convergence to $\|x_n - x_*\| = O(\mu_{\max}^n)$.

Proof. By the Gershgorin circle theorem, all eigenvalues of M lie in a closed disk of radius d centered at d in the complex plane. Thus, for any $\alpha < \frac{1}{hd}$, the eigenvalues of $A = I - h\alpha M$ are either 1 or have magnitude strictly less than 1. Since $x_n = A^n x_0$, we may write A in Jordan canonical form to observe the limiting behavior of x_n: in the direction of any eigenvector or generalized eigenvector associated to an eigenvalue with magnitude less than 1, the components of x_n approach 0. Specifically, by taking powers of Jordan blocks, we find that the magnitude of each of these components is $O\left(n^{b-1}\mu_{\max}^{n-(b-1)}\right)$, where $\mu_{\max}$ is the magnitude of the largest eigenvalue of A not equal to 1 and b is the size of the largest Jordan block corresponding to $\mu_{\max}$.

To complete the proof, we must examine the generalized eigenspace of A corresponding to the eigenvalue 1. In the Jordan canonical form of A, if there were a Jordan block for eigenvalue 1 of size $m > 1$, then powers would be given by

$$\begin{bmatrix} 1 & 1 & & & \\ & 1 & 1 & & \\ & & \ddots & & \\ & & & 1 & 1 \\ & & & & 1 \end{bmatrix}^n = \begin{bmatrix} 1 & n & \binom{n}{2} & \dots & \binom{n}{m} \\ & 1 & n & \dots & \binom{n}{m-1} \\ & & \ddots & & \vdots \\ & & & 1 & n \\ & & & & 1 \end{bmatrix}$$

(where the rest of the entries are all zeroes) and hence x_n would grow arbitrarily large for certain choices of x_0. To show this cannot be the case, given an arbitrary x_0, we show that the norm of x_n remains less than a fixed bound. Letting C be an upper bound on the absolute values of the entries of x_n (in the original basis), for any $v \in Y$, the v component of $x_{n+1} = Ax_n$ is computed as

$$(Ax_n)_v = \left(1 - h\alpha \sum_w a_{v,w}\right)(x_n)_v + h\alpha \sum_w a_{v,w}(x_n)_w,$$

where the sums are taken over all w adjacent to v. By our choice of α, we have $0 \leq h\alpha \sum_w a_{v,w} < 1$, so

$$|(Ax_n)_v| \leq \left(1 - h\alpha \sum_w a_{v,w}\right) C + h\alpha \sum_w a_{v,w} C = C.$$

Therefore, the ∞-norm of x_n does not increase as n increases, so there cannot be a Jordan block for the eigenvalue 1 of size greater than 1.

This shows that in the canonical basis, A^n approaches a block matrix of the form

$$\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix},$$

with the identity block corresponding to the eigenspace of A associated to eigenvalue 1. This is exactly the kernel of M, so x_n converges to the projection of x_0 onto $\ker M$ in the canonical basis of M. The size b of the largest Jordan block corresponding to $\mu_{\max}$ is at most $\operatorname{rank}(M)$, giving the rate of convergence. □

From here on, we will assume α has been chosen so that each eigenvalue of A is either equal to 1 or has magnitude strictly less than 1. Any α meeting the requirements of Theorem 2 is sufficient, for instance. Note that once weights $a_{v,w}$ are chosen, α can be chosen to minimize $\mu_{\max}$. There is in fact a unique optimal α, since $\mu_{\max}$ can be shown to be a strictly convex function of α.

Finally, we remark that the techniques used in the proof of Theorem 2 can give us insight into the effect of time steps with "faulty edges," that is, time steps during which clock data was not able to be exchanged across some edge as expected. This scenario is modeled by an alternate matrix $\widetilde{M}$, defined to be the same as M except with a certain weight $a_{v,w}$ changed to 0. This has the effect of shrinking the corresponding Gershgorin disk, so all eigenvalues of $\widetilde{M}$ remain in the largest Gershgorin disk of M. Choosing α according to Theorem 2 thus gives a convergent sequence x_n using either M or $\widetilde{M}$, although the rate of convergence for $\widetilde{M}$ may be less than optimal if α was chosen to optimize for M. This suggests that a faulty edge at a single step will not result in drastically different behavior, but will simply slow the rate of convergence temporarily. The same reasoning holds if there are multiple faulty edges in a single step. Possible future work could examine the effect of faulty edges in more detail (see Section 5).

Error Analysis

We now consider a more realistic scenario in which error is added at every step. We provide various results showing that bounds on the errors at each step yield bounds on the distance between x_n and the nearest steady state as $n \to \infty$. These results are more practical than the convergence results above, as they imply that in a realistic network, we can maintain synchronization within a certain, predictable error tolerance. We carry out our error analysis in the general case of the matrix M above with positive weights, but we find that often the cleanest error bounds apply to the case of symmetric weights, i.e. the sheaf Laplacian case. These results are just examples of what can be said in general settings, and future work could consider how they can be specialized or adapted to specific networks.

Absolute Error Bounds—We first demonstrate a method for bounding the error tolerance of our method using the standard Euclidean norm. To begin, we introduce error terms e_n into Equation (4) to produce

$$x_{n+1} = Ax_n + e_{n+1}. \tag{7}$$

This gives an explicit solution for x_n:

$$x_n = A^n x_0 + A^{n-1} e_1 + \cdots + A e_{n-1} + e_n. \tag{8}$$

In some cases, it may be useful to add an error term to x_n: this reduces to the above by rewriting $A(x_n + e'_n) + e_{n+1} = Ax_n + (Ae'_n + e_{n+1})$. In particular, the inclusion of error terms e_{n+1} and e'_n provide flexible modeling of errors inherent in clocks, errors in the values over boundary vertices that are maintained by an outside method, and errors arising from the asynchronous observation of neighbors' clock values.

For an initial error analysis, we will suppose a uniform bound $\|e_n\| < \varepsilon$ for all n. We have seen that $A^n x_0$ converges to a steady state, so we wish to bound the accumulated error represented by the remaining terms. Since any vector in the eigenspace W associated to eigenvalue 1 is a steady state, we wish to bound the distance between W and $x_n - A^n x_0 = A^{n-1} e_1 + \cdots + A e_{n-1} + e_n$, which we find by projecting to the orthogonal complement to W. We first check that by applying an appropriate orthogonal change of basis, we get a matrix $\widetilde{A} = UAU^T$ that can be written as a block matrix

$$\widetilde{A} = \begin{bmatrix} I & \widetilde{A}_{1,2} \\ 0 & \widetilde{A}_{2,2} \end{bmatrix},$$

where the upper left I represents the map on the eigenspace W. Indeed, in the proof of Theorem 2, we saw that the dimension of W is equal to the multiplicity of the eigenvalue 1, so we may choose any orthonormal basis for W and extend arbitrarily to an orthonormal basis for all of $C^0(G)$. Letting U be the matrix with columns consisting of these basis vectors, we get $\widetilde{A}$ of the form above. The lower right block $\widetilde{A}_{2,2}$ describes how the map behaves after projecting onto the orthogonal complement $W^\perp$, and thus its eigenvalues consist of the eigenvalues of A with magnitude less than 1. The difference between $x_n - A^n x_0 = A^{n-1} e_1 + \cdots + A e_{n-1} + e_n$ and the nearest point of W is thus equal to $\widetilde{A}_{2,2}^{n-1}\widetilde{e}_1 + \cdots + \widetilde{A}_{2,2}\widetilde{e}_{n-1} + \widetilde{e}_n$, where each $\widetilde{e}_i$ is the projection of e_i onto $W^\perp$, written in the basis of $\widetilde{A}$. In detail, if U is written in blocks corresponding to those of $\widetilde{A}$ as $U = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}$, then $\widetilde{e}_i = U_2 e_i$. This further implies $\|\widetilde{e}_i\| < \varepsilon$ for all i, since U is unitary. The norm of this difference is the distance to W, or equivalently the distance to the nearest steady state, and we call this the *steady-state error* at step n:

$$\begin{aligned} \mathrm{sse}_n &= \|\widetilde{A}_{2,2}^{n-1}\widetilde{e}_1 + \cdots + \widetilde{A}_{2,2}\widetilde{e}_{n-1} + \widetilde{e}_n\| \\ &= \left\| \sum_{i=1}^{n} \widetilde{A}_{2,2}^{n-i}\widetilde{e}_i \right\|. \end{aligned}$$

We can find bounds on the steady-state error in terms of the matrix $\widetilde{A}_{2,2}$. To begin, using the matrix operator norm corresponding to the Euclidean norm, we have

$$\begin{aligned} \mathrm{sse}_n &\le \|\widetilde{A}_{2,2}^{n-1}\|\|\widetilde{e}_1\| + \cdots + \|\widetilde{A}_{2,2}\|\|\widetilde{e}_{n-1}\| + \|\widetilde{e}_n\| \\ &\le \sum_{i=0}^{\infty} \|\widetilde{A}_{2,2}^i\|\varepsilon. \end{aligned} \tag{9}$$

The sum converges because the eigenvalues of $\widetilde{A}_{2,2}$ are those of A that have magnitude less than 1.

The bound given in Equation (9) is computable and can serve as a useful bound if $\widetilde{A}_{2,2}$ is built from a graph that is not too large. However, in most cases, it is too computationally expensive to be computed many times, and we would like a more computationally practical bound to use in optimization problems discussed below. The norm $\|\widetilde{A}_{2,2}\|$ is equal to the largest singular value $\sigma_{\max}$ of $\widetilde{A}_{2,2}$. In cases where $\sigma_{\max} < 1$, we further have an upper bound of

$$\mathrm{sse}_n \le \sum_{i=0}^{\infty} \|\widetilde{A}_{2,2}\|^i \varepsilon \le \sum_{i=0}^{\infty} \sigma_{\max}^i \varepsilon,$$

so we get

$$\mathrm{sse}_n \le \frac{\varepsilon}{1 - \sigma_{\max}}. \tag{10}$$

In the sheaf Laplacian case, we can do a similar error analysis beginning with Equation (6) and using the fact that the matrix $A_{Y,Y}$ in the equation is symmetric and positive semi-definite. In this case, a variation on the analysis above gives the bound $\mathrm{sse}_n \le \frac{\varepsilon}{1-\mu_{\max}}$, where $\mu_{\max}$ is the magnitude of the largest eigenvalue of A not equal to 1. This is a convenient error bound and leads to a practical optimization problem, described below.

In general, $\|\sigma_{\max}\|$ can be greater than 1 in spite of the eigenvalues having magnitude less than 1 and it can also take a value close to 1 when the eigenvalues are comparatively small, so in some cases Equation (10) may not be an effective bound. If we assume $\widetilde{A}_{2,2}$ is diagonalizable, we can produce a bound in terms of the eigenvalues instead, at the cost of introducing new terms: if $\widetilde{A}_{2,2}$ is diagonalized as $\widetilde{A}_{2,2} = SDS^{-1}$, then the operator norm of $\widetilde{A}_{2,2}^i$ is bounded by $\|\widetilde{A}_{2,2}^i\| \le \|S\|\|S^{-1}\|\mu_{\max}^i$. Note that this depends on the choice of the matrix S used in the diagonalization. The term $\|S\|\|S^{-1}\|$ is familiar from computational linear algebra: it is the *condition number* of the matrix S, denoted $\kappa(S)$. We can bound our error as follows:

$$\begin{aligned} \mathrm{sse}_n &\le \|\widetilde{A}_{2,2}^{n-1}\|\|\widetilde{e}_1\| + \cdots + \|\widetilde{A}_{2,2}\|\|\widetilde{e}_{n-1}\| + \|\widetilde{e}_n\| \\ &\le \|S\|\|S^{-1}\|\left(\varepsilon\mu_{\max}^{n-1} + \cdots + \varepsilon\mu_{\max} + \varepsilon\right) \\ &\le \kappa(S) \sum_{i=0}^{\infty} \varepsilon \mu_{\max}^i. \end{aligned}$$

This gives a final bound of

$$\mathrm{sse}_n \le \frac{\kappa(S)}{1 - \mu_{\max}} \varepsilon. \tag{11}$$

Statistical error analysis—Next we take a statistical approach to understanding the errors: suppose that the errors e_n are now independent identically distributed random vectors. Then the projections $\widetilde{e}_n$ are as well. Let $\mathrm{var}(x)$ denote the variance/covariance matrix of a vector x, defined as usual by $\mathrm{var}(x) = \mathbb{E}\big((x - \mathbb{E}(x))(x - \mathbb{E}(x))^T\big)$, where $\mathbb{E}$ denotes the expected value of random vectors. Variance/covariance matrices are symmetric and positive semi-definite. Set

$V = \mathrm{var}(e_n)$ and $\widetilde{V} = \mathrm{var}(\widetilde{e}_n) = U_2 V U_2^T$.

Again, we examine the sum $\widetilde{A}_{2,2}^{n-1}\widetilde{e}_1 + \cdots + \widetilde{A}_{2,2}\widetilde{e}_{n-1} + \widetilde{e}_n$. The expected value is simply

$$\widetilde{A}_{2,2}^{n-1}\mathbb{E}(\widetilde{e}_1) + \cdots + \widetilde{A}_{2,2}\mathbb{E}(\widetilde{e}_{n-1}) + \mathbb{E}(\widetilde{e}_n);$$

in particular, the expected value is zero if $\mathbb{E}(\widetilde{e}_i) = 0$ for all i. The variance/covariance matrix is more interesting: it is given by

$$\begin{aligned}
&\mathrm{var}(\widetilde{A}_{2,2}^{n-1}\widetilde{e}_1 + \cdots + \widetilde{A}_{2,2}\widetilde{e}_{n-1} + \widetilde{e}_n) \\
&= \widetilde{A}_{2,2}^{n-1}\mathrm{var}(\widetilde{e}_1)(\widetilde{A}_{2,2}^{n-1})^T + \ldots \\
&\qquad\qquad + \widetilde{A}_{2,2}\mathrm{var}(\widetilde{e}_{n-1})\widetilde{A}_{2,2}^T + \mathrm{var}(\widetilde{e}_n) \\
&= \widetilde{A}_{2,2}^{n-1}\widetilde{V}(\widetilde{A}_{2,2}^{n-1})^T + \cdots + \widetilde{A}_{2,2}\widetilde{V}\widetilde{A}_{2,2}^T + \widetilde{V}.
\end{aligned}$$

As n approaches infinity, this variance/covariance matrix converges since the eigenvalues of $\widetilde{A}_{2,2}$ have a magnitude less than 1. The limit is the matrix below:

$$\widetilde{\mathrm{ssv}} = \sum_{i=0}^{\infty} \widetilde{A}_{2,2}^i \widetilde{V} (\widetilde{A}_{2,2}^i)^T.$$

This is expressed in the basis of $\widetilde{A}_{2,2}$. Recalling that we made an orthogonal change of basis setting $\widetilde{A} = UAU^T$, we can convert back to the original basis, and we call the resulting matrix the *steady-state variance*:

$$\mathrm{ssv} = U^T \begin{bmatrix} 0 & 0 \\ 0 & \widetilde{\mathrm{ssv}} \end{bmatrix} U.$$

Thus, we find that the variance/covariance matrix for the difference between x_n and the nearest steady-state solution approaches ssv.

Fortunately, ssv can be computed explicitly as long as $\widetilde{A}_{2,2}$ can be diagonalized. Suppose $\widetilde{A}_{2,2} = SDS^{-1}$ as above and let $\mu_1, \ldots, \mu_r$ be the eigenvalues of $\widetilde{A}_{2,2}$ (the diagonal entries of D). Then we have

$$\begin{aligned}
&S^{-1}\widetilde{\mathrm{ssv}}(S^{-1})^T \\
&= \sum_{i=0}^{\infty} S^{-1}\widetilde{A}_{2,2}^i S S^{-1}\widetilde{V}(S^{-1})^T S^T (\widetilde{A}_{2,2}^i)^T (S^{-1})^T \\
&= \sum_{i=0}^{\infty} D^i \Big(S^{-1}\widetilde{V}(S^{-1})^T \Big) D^i,
\end{aligned}$$

so the entries are given by

$$\begin{aligned}
\big(S^{-1}\widetilde{\mathrm{ssv}}(S^{-1})^T\big)_{j,k} &= \sum_{i=0}^{\infty} \mu_j^i \mu_k^i \Big(S^{-1}\widetilde{V}(S^{-1})^T \Big)_{j,k} \\
&= \frac{1}{1-\mu_j\mu_k} \Big(S^{-1}\widetilde{V}(S^{-1})^T \Big)_{j,k}.
\end{aligned} \tag{12}$$

This provides a practical way to compute $\widetilde{\mathrm{ssv}}$, requiring just the diagonalization of $\widetilde{A}_{2,2}$, and from this ssv can be computed.

In addition to finding this explicit means of computing ssv, we can find bounds on its norm. Below we use the operator norm on matrices corresponding to the Euclidean norm and again let $\sigma_{\max}$ be the largest singular value of $\widetilde{A}_{2,2}$. As long as $\sigma_{\max} < 1$, we have

$$\begin{aligned}
\|\mathrm{ssv}\| &= \|\widetilde{\mathrm{ssv}}\| \\
&\le \sum_{i=0}^{\infty} \|\widetilde{A}_{2,2}^i\| \|\widetilde{V}\| \|(\widetilde{A}_{2,2}^i)^T\| \\
&\le \sum_{n=0}^{\infty} \sigma_{\max}^i \|\widetilde{V}\| \sigma_{\max}^i.
\end{aligned}$$

Since $\widetilde{V}$ is a block of UVU^T, which has the same norm as V, we have $\|\widetilde{V}\| \le \|V\|$ and thus obtain the following bound:

$$\|\mathrm{ssv}\| \le \frac{1}{1-\sigma_{\max}^2}\|V\|. \tag{13}$$

As above, we can find a bound in terms of the eigenvalues if we assume that $\widetilde{A}_{2,2}$ is diagonalizable: if $\widetilde{A}_{2,2} = SDS^{-1}$ as above, then we have

$$\begin{aligned}
\|\mathrm{ssv}\| &= \|\widetilde{\mathrm{ssv}}\| \\
&\le \sum_{i=0}^{\infty} \|\widetilde{A}_{2,2}^i\| \|\widetilde{V}\| \|(\widetilde{A}_{2,2}^i)^T\| \\
&\le \sum_{n=0}^{\infty} \big(\|S\|\|S^{-1}\|\mu_{\max}^i\big) \|\widetilde{V}\| \big(\|S\|\|S^{-1}\|\mu_{\max}^i\big) \\
&\le \frac{\kappa(S)^2}{1-\mu_{\max}^2}\|V\|.
\end{aligned}$$

If we choose to ignore the covariances and focus entirely on the variances, the trace provides a convenient summary of ssv. Since $\widetilde{V}$ is symmetric and positive semi-definite, we have a Cholesky factorization $\widetilde{V} = LL^T$. Letting $\|\cdot\|_F$ be the Frobenius norm,

$$\begin{aligned}
\mathrm{tr}(\mathrm{ssv}) &= \mathrm{tr}(\widetilde{\mathrm{ssv}}) \\
&= \sum_{i=0}^{\infty} \mathrm{tr}\Big(\widetilde{A}_{2,2}^i \widetilde{V} (\widetilde{A}_{2,2}^i)^T \Big) \\
&= \sum_{i=0}^{\infty} \mathrm{tr}\Big(\widetilde{A}_{2,2}^i L L^T (\widetilde{A}_{2,2}^i)^T \Big) \\
&= \sum_{i=0}^{\infty} \|\widetilde{A}_{2,2}^i L\|_F^2 \\
&\le \|L\|_F^2 \sum_{i=0}^{\infty} \|\widetilde{A}_{2,2}^i\|_F^2 \\
&= \mathrm{tr}(\widetilde{V}) \sum_{i=0}^{\infty} \|\widetilde{A}_{2,2}^i\|_F^2.
\end{aligned}$$

To bound $\mathrm{tr}(\widetilde{V})$ in this expression, again write $U = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}$, so that

$$\mathrm{tr}(V) = \mathrm{tr}\big(UVU^T\big) = \mathrm{tr}\left(\begin{bmatrix} U_1VU_1^T & U_1VU_2^T \\ U_2VU_1^T & U_2VU_2^T \end{bmatrix}\right).$$

Then $\mathrm{tr}(\widetilde{V}) = \mathrm{tr}(U_2VU_2^T)$ and $\mathrm{tr}(U_1VU_1^T) \geq 0$ as it is the trace of the variance/covariance matrix $\mathrm{var}(U_1e_n)$, so $\mathrm{tr}(\widetilde{V}) = \mathrm{tr}(V) - \mathrm{tr}(U_1VU_1^T) \leq \mathrm{tr}(V)$.

Combining with the above gives a bound on the trace of ssv:

$$\mathrm{tr}(\mathrm{ssv}) \leq \mathrm{tr}(V) \sum_{i=0}^{\infty} \|\widetilde{A}_{2,2}^i\|_F^2. \tag{14}$$

This is analogous to Equation (9).

Alternately, we can work with the explicit expression of Equation (12), beginning as follows:

$$\begin{aligned} \mathrm{tr}\big(S^{-1}\widetilde{\mathrm{ssv}}(S^{-1})^T\big) &= \sum_j \frac{1}{1-\mu_j^2}\Big(S^{-1}\widetilde{V}(S^{-1})^T\Big)_{j,j} \\ &\leq \frac{1}{1-\mu_{\max}^2}\mathrm{tr}\Big(S^{-1}\widetilde{V}(S^{-1})^T\Big). \end{aligned}$$

Again using the Cholesky factorization $\widetilde{V} = LL^T$, we get a bound

$$\begin{aligned} \mathrm{tr}(S^{-1}\widetilde{V}(S^{-1})^T) &= \|S^{-1}L\|_F^2 \\ &\leq \|S^{-1}\|_F^2\|L\|_F^2 \\ &= \|S^{-1}\|_F^2\mathrm{tr}(\widetilde{V}). \end{aligned}$$

Similarly, we also have

$$\mathrm{tr}(\widetilde{\mathrm{ssv}}) \leq \|S\|_F^2\ \mathrm{tr}\big(S^{-1}\widetilde{\mathrm{ssv}}(S^{-1})^T\big).$$

Combining with the previous step, we have

$$\begin{aligned} \mathrm{tr}(\mathrm{ssv}) &= \mathrm{tr}(\widetilde{\mathrm{ssv}}) \\ &\leq \|S\|_F^2\ \mathrm{tr}\big(S^{-1}\widetilde{\mathrm{ssv}}(S^{-1})^T\big) \\ &\leq \frac{\|S\|_F^2}{1-\mu_{\max}^2}\ \mathrm{tr}\Big(S^{-1}\widetilde{V}(S^{-1})^T\Big) \\ &\leq \frac{\|S\|_F^2\|S^{-1}\|_F^2}{1-\mu_{\max}^2}\ \mathrm{tr}(\widetilde{V}). \end{aligned}$$

As before, $\mathrm{tr}(\widetilde{V}) \leq \mathrm{tr}(V)$, so we get a final bound on the trace of ssv:

$$\mathrm{tr}(\mathrm{ssv}) \leq \frac{\|S\|_F^2\|S^{-1}\|_F^2}{1-\mu_{\max}^2}\ \mathrm{tr}(V). \tag{15}$$

As with the absolute error bounds, we get simpler bounds on ssv in the sheaf Laplacian case, again resulting from the matrix $A_{Y,Y}$ in Equation (6) being diagonalizable by orthogonal matrices. In the sheaf Laplacian case, we get bounds $\|\mathrm{ssv}\| \leq \frac{1}{1-\mu_{\max}^2}\|V\|$ and $\mathrm{tr}(\mathrm{ssv}) \leq \frac{1}{1-\mu_{\max}^2}\mathrm{tr}(V)$ analogous to the above, and it is also possible to show $\|\mathrm{ssv}\|_F \leq \frac{1}{1-\mu_{\max}^2}\|V\|_F$.

Optimization

The results above have suggested the quantities appearing in our error bounds could be optimized when choosing the weights $a_{v,w}$ and the constant α. Mathematically, we can formulate a variety of optimization problems from choices of objective functions and constraints on the weights $a_{v,w}$. With these problems, there is a trade-off between how accurately an objective function represents errors in the network and how easily it can be computed. For small networks (where we note that some space networks in the immediate future may indeed be rather small), a computationally expensive objective function may be usable. For larger networks, we may have to settle for a less accurate but more computationally practical objective function. We list some of the options of optimization problems here, roughly in order from the most computationally expensive objective function to the least.

Problem 1. Choose weights $a_{v,w} \geq 0$ and the constant $\alpha > 0$ to minimize the steady-state error bound $\sum_{i=0}^{\infty}\|\widetilde{A}_{2,2}^i\|\varepsilon$ given in Equation (9).

Problem 2. Choose weights $a_{v,w} \geq 0$ and the constant $\alpha > 0$ to minimize one of the following expressions appearing in the previous section: $\frac{\kappa(S)}{1-\mu_{\max}}$, $\frac{\kappa(S)^2}{1-\mu_{\max}^2}$, or $\frac{\|S\|_F^2\|S^{-1}\|_F^2}{1-\mu_{\max}^2}$.

Problem 2 is perhaps not very practical because there is a choice involved in the matrix S. Additionally, since the second and third expressions in this problem are bounds on the norm and trace of ssv, these cases are subsumed by the following option, which is likely more computable.

Problem 3. Choose weights $a_{v,w} \geq 0$ and the constant $\alpha > 0$ to minimize a matrix norm or the trace of ssv, computed as described in Equation (12).

The remaining problems are the most practical, being based on singular values and eigenvalues.

Problem 4. Choose weights $a_{v,w} \geq 0$ and the constant $\alpha > 0$ to minimize $\sigma_{\max}$, the largest singular value of $\widetilde{A}_{2,2}$, which appears in the bounds in Equations (10) and (13).

We can also choose to optimize the eigenvalues of $\widetilde{A}_{2,2}$ instead of the singular values. The eigenvalues of $\widetilde{A}_{2,2}$ are those of A that are not equal to 1. Thus $\mu_{\max}$, the magnitude of the largest eigenvalue of $\widetilde{A}_{2,2}$, is equivalently the magnitude of the largest eigenvalue of A not equal to 1. While minimizing $\mu_{\max}$ does not directly minimize any of the error bounds above, we include the following problem as a possibly more practical option.

Problem 5. Choose weights $a_{v,w} \geq 0$ and the constant $\alpha > 0$ to minimize $\mu_{\max}$, the largest magnitude of an eigenvalue of A not equal to 1.

Finally, in the sheaf Laplacian case, minimizing the error bounds does in fact reduce to minimizing $\mu_{\max}$. Furthermore, once constants $a_{v,w}$ are chosen, we have seen that there is a unique α that minimizes $\mu_{\max}$, and we can assume that this α is always chosen. We formulate a problem for this case as follows:

Problem 6. Choose symmetric weights $a_{v,w} = a_{w,v} \geq 0$ to minimize $\mu_{\max}$, the largest magnitude of an eigenvalue of A not equal to 1. This is equivalent to minimizing $\frac{\lambda_{\max}}{\lambda_{\min}}$,

where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest nonzero eigenvalues of M.

The ratio of eigenvalues $\frac{\lambda_{\max}}{\lambda_{\min}}$ in Problem 6 is in fact the condition number of the restriction of the Y block of M to the orthogonal complement of its kernel. The fact that minimizing $\mu_{\max}$ is equivalent to minimizing this condition number follows from finding $\mu_{\max}$ in terms of $\lambda_{\max}$ and $\lambda_{\min}$, which are necessarily real in the sheaf Laplacian case. Note that the minimum and maximum nonzero eigenvalues of A are $1 - h\alpha\lambda_{\max}$ and $1 - h\alpha\lambda_{\min}$. We have assumed the choice of α that minimizes the larger absolute value of these two: this places them at equal distances from 0, giving $h\alpha\lambda_{\max} - 1 = 1 - h\alpha\lambda_{\min}$ and thus $\alpha = \frac{2}{h(\lambda_{\max}+\lambda_{\min})}$. Substituting to find the maximal and minimal eigenvalues of A gives

$$\mu_{\max} = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} = \frac{\frac{\lambda_{\max}}{\lambda_{\min}} - 1}{\frac{\lambda_{\max}}{\lambda_{\min}} + 1},$$

which is minimized when the condition number $\frac{\lambda_{\max}}{\lambda_{\min}}$ is minimized.

Many of the objective functions in the problems above are generally not differentiable due to the terms $\mu_{\max}$ and $\sigma_{\max}$. Thus, the common technique of gradient descent is not fully justified for these problems, although it could still be attempted. A more principled approach would be to use a subgradient method – we suggest this as a fruitful area for future work. We will conclude this section by emphasizing that the problems posed here are simply a first suggestion of how to optimize the weights, as more useful error estimations and approaches tailored to specific networks could be found in the future.

4. TOWARDS A PROTOCOL

The previous section considered the mathematical theory behind our proposed clock synchronization method. In this section, we outline some steps necessary to turn this method into a practical protocol that could be incorporated into DTN. This section is divided into several subsections addressing different aspects of implementing our method, and we conclude each with a list of what will need to be addressed by a protocol.

Predetermined parameters sent to nodes

We have assumed throughout that certain data, which we call the *parameters* of our method, are determined on Earth and distributed to the network nodes. Here we outline what decisions are made on Earth and list the parameters needed by the network nodes. To begin, the structure of a network needs to be determined up front, based on knowledge of the nodes and the periods during which they can communicate, which are assumed to be scheduled in current approaches to DTN. The structure of the network includes the choice of which edges to include – as we have described above, an edge should only be included between two vertices when they have sufficiently regular periods of communication. The parameter h is chosen as a length of the time steps so that each pair of vertices connected by an edge can exchange clock data once per time step (where we may allow some exceptions if we are willing to accept occasional faulty edges, as discussed in Section 3). Thus, the choice of edges is intertwined with the choice of h, and they provide competing goals: we would like to minimize h, as more frequent updates should improve performance, but we would also like to maximize the number of edges, as more communication between nodes should also improve performance. These choices are highly dependent on the specific network and the scheduled periods of communication between nodes.

Additionally, the structure of the network should take into account the relative accuracies of clocks in the network, since in our method, each clock influences its neighbors. To handle a large variety of accuracies of clocks, possibly multiple orders of magnitude, we would like to prevent significantly less accurate clocks from providing updates to more accurate ones. The notion of boundary conditions that we have investigated above provides an approach. Above, we have thought of the boundary B within the graph G as being synchronized to a high degree of accuracy by an outside method. Lower accuracy nodes outside B did not influence the nodes of B, but the higher accuracy nodes within B were allowed to influence nodes outside B. We can iterate this approach as needed, organizing our network G into nested subgraphs $G_0 \subseteq G_1 \subseteq \cdots \subseteq G_n$, with more accurate clocks placed in lower G_i. Then G_0 can consist of true boundary nodes, synchronized by an outside method, while nodes in $G \setminus G_i$ treat G_i as the boundary. Borrowing a term from NTP, we refer to nodes in G_0 as *stratum* 0, and nodes in $G_i \setminus G_{i-1}$ as *stratum* i, where roughly speaking, lower strata are more accurate. We thus have a stratum number assigned to every node, where nodes only receive updates from neighbors that have lesser or equal stratum numbers. For instance, our simple case of a graph G with boundary vertices in B has two strata, where vertices in B have stratum number 0 and vertices in $G \setminus B$ have stratum number 1.

After determining the structure of the network, the weights $a_{v,w}$ and the constant α must be determined upfront as well. Note though that in practice, α is superfluous, since it simply scales all of the weights. Additionally, the parameter h used above does not need to be explicitly recorded, as it just serves to specify the times at which an update in clocks takes place. In practice, a schedule of times at which clocks are updated should be sent to each node, with consecutive times differing by h. Our method best describes a case where all nodes update their clocks at the same time, although it would certainly be possible to experiment with less rigid scheduling. In addition to the times at which clocks will be updated, nodes also need a schedule of when to exchange clock data with the appropriate neighbors; this is dependent on the scheduled periods of communication between nodes used for routing in DTN. One simple approach to creating a schedule for exchanging clock data is to have nodes that are connected by an edge exchange clock data during their earliest period of communication in each time step.

Once the structure of the network and the parameters above have been determined, the information a node needs is simply its row in the matrix A of Equation (4). Referring back to Equations (2) and (3) and accounting for our partitioning of the network into strata, the update for vertex v is given by

$$(x_{n+1})_v = (x_n)_v - h\alpha \sum_w a_{v,w}((x_n)_v - (x_n)_w),$$

where the sum is taken over all vertices w adjacent to v that have a stratum number less than or equal to that of v. As described in Section 2, even though each $(x_n)_v$ represents the difference between the clock value of v and the true time,

the sum in the equation above can be computed from the clock values, as it only requires knowledge of the differences. To describe the update at the end of a given time step, let $C_{v,w} = C_v(t) - C_w(t)$ be the difference in clock values observed at some time t during the time step. Then according to the above, v updates its clock by subtracting

$$\sum_w h\alpha \; a_{v,w} \; C_{v,w}. \tag{16}$$

It is the coefficients $h\alpha a_{v,w}$ that need to be communicated to node v, since from these, the update to the clock can be computed.

To summarize, the parameters that must be computed on Earth and sent to a node v consist of:

- A list of nodes w adjacent to v that have a stratum number less than or equal to that of v, along with the method of receiving clock data from w. This list of nodes will need to be compatible with the schedule used by DTN for routing.
- A schedule containing regularly spaced times at which the node v will update its clock and the times at which v will exchange clock values with its neighbors. This schedule will also necessarily depend on the schedule used by DTN for routing.
- The coefficients $h\alpha a_{v,w}$ used to compute updates, as given in Equation (16).
- Additional parameters described in the subsections below: tolerance levels for reality checks, other parameters needed for future fault tolerance methods, expiration times for the $C_{v,w}$ if used, and any parameters required for initializing a clock value based on neighbors' data.

Data managed by nodes

Here we examine the types of data nodes will need to store and use in the update process. We will not attempt to specify formats for the data, as we leave this for a time in the future when the needs of the protocol are better understood. The data managed by a node of course includes the parameters sent to the node as described above. This data will be accessed as needed when adjusting the clock and updated whenever a new set of parameters is received by the node.

The other data managed by a node include the clock data received from its neighbors and, of course, its own clock value. For the clock data node v receives from a neighbor w, only the difference in clock values, written as $C_{v,w}$, needs to be recorded. So node v will maintain a list of values of $C_{v,w}$, one for each neighbor w that it is assigned to receive clock data from (those with lesser or equal stratum number, as above). The value of $C_{v,w}$ should be initialized to 0 and updated when a clock value is received from w. We have assumed that w sends its clock value once per time step; however, in practice, there may be time steps during which w cannot send to v (we described such a scenario as a "faulty edge" in Section 3). The simplest solution is to have $C_{v,w}$ reset to 0 if no clock value is received from w during a time step. A more flexible solution involves a choice of "expiration time" of the $C_{v,w}$ values, that is, a number of time steps without a clock value from w after which $C_{v,w}$ is reset to 0. This would allow a node v to use an old value of $C_{v,w}$ until it expires, if no new value has been received. More nuanced variations of this approach in which old $C_{v,w}$ values decay over time are also possible.

The updating of a $C_{v,w}$ value is a convenient place to include a reality check. Before updating $C_{v,w}$, a node should check that the new value is reasonable, i.e. has absolute value less than some specified tolerance computed based on reasonable errors to expect in the network. This tolerance becomes another predetermined parameter that must be distributed to the nodes.

We have recorded the following data each node v must manage:
- The parameters sent to the node, as described in the previous subsection.
- $C_{v,w}$ for each neighbor w from which clock data is received.
- If the $C_{v,w}$ come with an expiration time or a related approach to faulty edges is taken, then we must include, for each $C_{v,w}$, a counter of how many time steps have passed since $C_{v,w}$ was last updated.

Procedures performed by nodes

The nodes of the network need the ability to perform certain procedures to make use of our method. To begin, we need to be able to initialize a node v without any assumed data. Upon receiving an initial set of parameters, v should store the parameters and set values of all $C_{v,w}$ to 0, and we can allow for multiple options for how v initializes its clock value. It may keep a preexisting clock value or can set a clock value based on the first clock value received from a neighbor or some type of average of the first few clock values received from neighbors. Nodes should only be allowed to keep preexisting clock values that are reasonably accurate, so that they do not introduce large errors as they begin to send their clock values to neighbors. After initialization, nodes must be able to accept updates to parameters as they are received.

Next, nodes need to be able to exchange clock data, which can occur across an edge in the graph when two nodes can communicate. Existing techniques for exchanging clock data across a link in space will need to be incorporated into the protocol. As we have described, we need to allow for multiple scenarios in which clock data is exchanged: this includes cases where ranging techniques can be used to determine the speed of light delay, as well as cases where long distances force us to use a predetermined estimate of the speed of light delay. We must allow the exchange of clock data between nodes v and w to occur in only one direction, e.g. from v to w, or in both directions. In the case of a two-way exchange of clock values, there is the possibility to allow one node to compute the difference and send it to its neighbor, which ensures that $C_{v,w} = -C_{w,v}$ (instead of having both nodes perform the computation separately and introduce different errors). Because of the complexity, the exchange of clock data may be broken down into multiple cases and may require the sending of multiple messages containing clock data.

Finally, each node must have a procedure to update its clock value. This simply requires the computation of the expression given in 16 and access to the node's clock. This provides another opportunity for a reality check: nodes can be limited in how much they are able to adjust their clock in a single step to improve fault tolerance.

In summary, a node must be able to perform the following procedures:
- Initialization, with multiple options of how to set an initial clock value.
- Update of predetermined parameters when new values are received.
- Exchange of clock values with neighbors – this is the most complicated and may be divided into multiple cases.

- Update of the node's clock value.

Types of bundles used

We conclude this section with an outline of how nodes may exchange data relevant to clock synchronization using *bundles*, the packets of data used in DTN. As in our discussion of data managed by nodes, we will not specify the formats of the bundles to be used, as this should be chosen at a later date when the interaction of clock synchronization with other aspects of DTN is more clearly established. Once a general format of clock synchronization bundles is chosen, we can categorize these bundles by their function, and these categories should include at least:

- Bundles for updates to parameters, including the option of initializing a node's data. A bundle initializing a node could contain the choice of how to initialize the clock value.
- Bundles for the exchange of clock data. This should include at least a type of bundle that simply carries a time stamp. Other variations could include data used in a two-way exchange of clock information.

5. Conclusion and Future Work

In our current work, aiming to advance the idea of a DTN clock synchronization method based on diffusion, we have chosen to focus primarily on the mathematical motivation and theoretical guarantees on performance. Having outlined the work necessary to turn this method into a viable network protocol in the previous section, there remain many avenues for future research, both to improve the theoretical understanding of our method and to begin to validate it experimentally. We conclude with some suggestions for such future work.

- **Fault tolerance and security**
  - In Section 3, after the proof of Theorem 2, we gave a heuristic argument for why our method should be able to tolerate occasional faulty edges. Future work could look into making this argument more precise or testing experimentally the effect of faulty edges.
  - In Section 4, we suggested that certain reality checks should be incorporated into a protocol in order to handle faulty data. Future work could look into the best approaches to these reality checks, along with other ways of detecting faulty data. One possible objective for this work would be to make a protocol that is robust to Byzantine faults, along the lines of [37]. Currently, our protocol offers no adversarial fault tolerance; it assumes that a trusted institution (e.g., NASA) acts as a centralized certificate authority, that every user with a certificate is honest, and that the network is secure. The interaction of a clock synchronization protocol and security protocols for DTN will be important, especially in determining how much fault tolerance should be built into a clock synchronization protocol.
  - To our knowledge, the only standardized security protocol designed and aimed for DTN currently is Bundle Protocol Security (BPSec) [38–41]. However, BPSec alone could be insufficient for providing the fault tolerance of our desired level, as BPSec provides only confidentiality and integrity of bundles, and assumes the existence of a key management service. The use of identity-based encryption (IBE) [42] could reduce the necessity of public key infrastructure and key distribution and can be used as a way to combine a public key and certificate (identity) into one. The use of IBE or related schemes for DTN is actively being studied already [43–46]. However, IBE still requires a central authority, known as the private key generator (PKG), and in fact requires more trust in them than traditional public-key encryption schemes as they have the authority to store and generate secret keys [47].
  - Some anonymous credential schemes based on non-interactive zero-knowledge proofs (e.g., zkSNARKs [48]) such as the one presented in [49] hence could be desired as they can be used to verify a user's identity asynchronously while preserving the anonymity. They can also potentially prevent DoS attacks, which are proven to be effective against DTNs due to their store-and-carry architecture [4, 50–53]. In particular, the scheme presented in [49] allows the issuance list to be auditable publicly (while the attributes of credentials remain private) as a form of Merkle trees.
- **Layer of responsibility**
  - In the last few subsections of Section 4, we have mostly assumed that it would be the responsibility of the bundle layer/protocol (BP) to handle the clock synchronization across SSI. It is imperative that which agent within the BP should primarily be responsible for it. For example, if the application element (AE) of the application agent (AA) is the one that handles the clock synchronization, creating a new bundle format for clock synchronization requests may not be necessary, because then the clock synchronization request bundle can be just a bundle that encapsulates an admin record that contains clock information. On the other hand, given that computer clocks and their synchronization protocols (e.g., NTP) usually reside in the application layer of the network, and that DTN is an overlay architecture and hence can be heterogeneous, deploying our protocol in the application layer may offer more efficiency and robustness than in the bundle layer.
- **Automated exchange of clock data**
  - Any automated clock synchronization protocol requires an automated exchange of clock data between network nodes. Future work on a clock synchronization protocol for DTN will need to incorporate existing techniques for exchange of clock data in space as automated capabilities of DTN so that the schedule of these exchanges can be created in advance.
  - The sending of clock data in bundles will need to not interfere with the techniques of exchanging clock data – for precise timing, this may require careful consideration of time stamps and the time required to read and write bundles. Alternately, a protocol could allow certain techniques of exchanging clock data to take place outside of DTN, without formatting data as bundles.
- **Error bounds in specific settings**
  - The error analysis given in Section 3 applies to general networks and general error vectors. The techniques could potentially be improved if we assume more knowledge of the errors or the structure of the network. Future work could examine error tolerance if specific distributions of the error vectors e_n are known or could analyze error bounds for specific simulated networks.
- **Optimization problems**
  - Future work can consider developing techniques to solve the optimization problems we have formulated, including investigating subgradient methods.
  - Related to the future work on error bounds, different optimization problems could be formulated for a given network if we are able to find objective functions that better reflect the error tolerance of the network.
  - An important, but less specific optimization problem is that of choosing the edges to be included in the graph and the length of the time step h given a network and a schedule of when nodes are able to communicate – see the discussion in Section 4.
  - Another interesting problem would be to explore different 'diffusion' equations to diffuse the clock values and see which method converges the fastest and is the most stable. In this paper, only the (variant of) heat equation (Equation (1))

was considered. The use of mean curvature flow (or curve-shortening flow) can be seen as an immediate generalization of our protocol as it is a geometric heat equation. This direction of studies would also initiate a new study of numerical analysis of differential equations on cellular sheaves.

- **Tests in simulated networks**

– Once h, α, and all weights $a_{v,w}$ have been chosen, a straightforward simulation of our method is uninteresting: it is essentially repeated multiplication by the matrix A. A more interesting simulation would build on some of the proposed future work on fault tolerance. This could include simulation of faulty edges as mentioned above and the inclusion of reality checks or other methods to detect faulty data. Simulation of Byzantine faults could be useful once sufficient fault tolerance methods have been introduced.

– We have suggested certain optimization problems as methods for choosing the weights $a_{v,w}$, and future work may suggest other methods as well. Various candidate methods for choosing weights should be tested and compared in simulated networks.

- **An approach without scheduling**

– Our approach has been based on the assumption that exchanges of clock data can be scheduled – this is appropriate with the schedule-based approach to routing in DTN that is currently standard. However, as DTN expands to include more techniques, it may be beneficial to have a method of clock synchronization that does not depend on (or depends less heavily on) scheduled exchanges of clock data. For instance, one of the prominent alternatives to the schedule-based approaches is PRoPHET [54, 55], which takes a probabilistic approach to routing. Certain aspects of our method, especially the linear models of diffusion in a graph and the minimal computations required, could be applied to methods that do not assume regular time steps or scheduled times for updates. While the mathematical analysis of such a method would likely be more difficult, the principles behind it would remain the same, and such a method would be a useful option to have for the future.

REFERENCES

[1] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 27–34. [Online]. Available: https://doi.org/10.1145/863955.863960

[2] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, "Delay-tolerant networking: an approach to interplanetary internet," *IEEE Communications Magazine*, vol. 41, no. 6, pp. 128–136, 2003.

[3] I. F. Akyildiz, O. B. Akan, C. Chen, J. Fang, and W. Su, "Interplanetary internet: state-of-the-art and research challenges," *Computer Networks*, vol. 43, no. 2, pp. 75–112, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128603003451

[4] K. Fall and S. Farrell, "Dtn: an architectural retrospective," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 5, pp. 828–836, 2008.

[5] A. Hylton and D. E. Raible, *High Data Rate Architecture (HiDRA)*. AIAA, 2016. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2016-5756

[6] A. Hylton, D. Raible, and G. Clark, "A delay tolerant networking-based approach to a high data rate architecture for spacecraft," *2019 IEEE Aerospace Conference*, pp. 1–10, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:85511897

[7] A. Hylton, D. Raible, G. Clark, R. Dudukovich, B. Tomko, and L. Burke, "Rising above the cloud: Toward high-rate delay-tolerant networking in low earth orbit," in *Advances in Communications Satellite Systems. Proceedings of the 37th International Communications Satellite Systems Conference (ICSSC-2019)*, 2019, pp. 1–17.

[8] M. Moy, R. Kassouf-Short, N. Kortas, J. Cleveland, B. Tomko, D. Conricode, Y. Kirkpatrick, R. Cardona, B. Heller, and J. Curry, "Contact multigraph routing: Overview and implementation," in *2023 IEEE Aerospace Conference*, 2023, pp. 1–9.

[9] R. Dudukovich, B. LaFuente, A. Hylton, B. Tomko, and J. Follo, "A distributed approach to high-rate delay tolerant networking within a virtualized environment," in *2021 IEEE Cognitive Communications for Aerospace Applications Workshop (CCAAW)*, 2021, pp. 1–5.

[10] NASA Glenn Research Center (B. Tomko, E. Schweinsberg, N. Kotas, B. LaFuente, R. Dudukovich, K. J. Vernyi *et al.*), "HDTN," 2019, GitHub Repository. [Online]. Available: https://github.com/nasa/hdtn

[11] S. B. Cooper, "From mercury to pluto: A common approach to mission timekeeping," *IEEE Aerospace and Electronic Systems Magazine*, vol. 21, no. 10, pp. 18–23, 2006.

[12] T. Ely, J. Seubert, and J. Bell, "Advancing navigation, timing, and science with the deep space atomic clock," in *13th International Conference on Space Operations*, ser. SpaceOps 2014, 05 2014, p. 1856.

[13] T. A. Ely, J. Seubert, J. Prestage, R. Tjoelker, E. Burt, A. Dorsey, D. Enzer, R. Herrera, D. Kuang, D. Murphy *et al.*, "Deep space atomic clock mission overview," in *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, Portland, ME, USA*, 2019, pp. 11–13.

[14] T. A. Ely, E. A. Burt, J. D. Prestage, J. M. Seubert, and R. L. Tjoelker, "Using the deep space atomic clock for navigation and science," *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 65, no. 6, pp. 950–961, 2018.

[15] E. O. Ilo-Okeke, L. Tessler, J. P. Dowling, and T. Byrnes, "Remote quantum clock synchronization without synchronized clocks," *npj Quantum Information*, vol. 4, no. 1, p. 40, 2018.

[16] J. S. Sidhu, S. K. Joshi, M. Gündoğan, T. Brougham, D. Lowndes, L. Mazzarella, M. Krutzik, S. Mohapatra, D. Dequal, G. Vallone *et al.*, "Advances in space quantum communications," *IET Quantum Communication*, vol. 2, no. 4, pp. 182–217, 2021.

[17] J. Troupe, S. Haldar, I. Agullo, and P. Kwiat, "Quantum clock synchronization for future nasa deep space quantum links and fundamental science," *arXiv preprint arXiv:2209.15122*, 2022.

[18] D. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, 1991.

[19] J. Martin, J. Burbank, W. Kasch, and P. D. L. Mills, "Network Time Protocol Version 4: Protocol and Algorithms Specification," RFC 5905, Jun. 2010.

[Online]. Available: https://www.rfc-editor.org/info/rfc5905

[20] Q. Ye and L. Cheng, "Dtp: Double-pairwise time protocol for disruption tolerant networks," in *2008 The 28th International Conference on Distributed Computing Systems*, 2008, pp. 345–352.

[21] J. Rash, R. Parise, K. Hogie, E. Criscuolo, J. Langston, C. Jackson, H. Price, and E. I. Powers, "Internet access to spacecraft," in *14th Annual/USU Conference on Small Satellites*, no. SSC00-IX-1, 2000.

[22] L. Felton, L. Pitts, and F. VanLandingham, "Nasa architecture for solar system time synchronization and dissemination: Concept of operations," in *SpaceOps 2008 Conference*. AIAA, 2008.

[23] S. Woo, J. Gao, and D. Mills, "Space network time distribution and synchronization protocol development for mars proximity link," in *SpaceOps 2010 Conference*, no. 2010-2360. Huntsville, AL: AIAA, 04 2010.

[24] D. L. Mills, *Computer network time synchronization: the network time protocol*. CRC press, 2006.

[25] A. Hylton, N. Tsuei, M. Ronnenberg, J. Hwang, B. Mallery, J. Quartin, C. Levaunt, J. Quail, and J. Curry, "Toward time synchronization in delay tolerant network based solar system internetworking," in *2023 IEEE Aerospace Conference*, 2023, pp. 1–20.

[26] B. J. Choi and X. Shen, "Distributed clock synchronization in delay tolerant networks," in *2010 IEEE International Conference on Communications*, 2010, pp. 1–6.

[27] B. J. Choi, H. Liang, X. Shen, and W. Zhuang, "Dcs: Distributed asynchronous clock synchronization in delay tolerant networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 3, pp. 491–504, 2012.

[28] M. Sasabe and T. Takine, "A simple scheme for relative time synchronization in delay tolerant manets," in *2009 International Conference on Intelligent Networking and Collaborative Systems*, 2009, pp. 395–396.

[29] Q. Li and D. Rus, "Global clock synchronization in sensor networks," in *IEEE INFOCOM 2004*, vol. 1, 2004, p. 574.

[30] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl, "Reaching approximate agreement in the presence of faults," in *Third Symposium on Reliability in Distributed Software and Database Systems*, ser. SRDS 1983. Clearwater Beach, FL, USA: IEEE Computer Society, 10 1983, pp. 145–154.

[31] J. Lundelius and N. Lynch, "A new fault-tolerant algorithm for clock synchronization," in *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC '84. New York, NY, USA: Association for Computing Machinery, 1984, p. 75–88. [Online]. Available: https://doi.org/10.1145/800222.806738

[32] L. Lamport and P. M. Melliar-Smith, "Synchronizing clocks in the presence of faults," *Journal of ACM*, vol. 32, no. 1, p. 52–78, 01 1985. [Online]. Available: https://doi.org/10.1145/2455.2457

[33] D. J. Israel, K. D. Mauldin, C. J. Roberts, J. W. Mitchell, A. A. Pulkkinen, L. V. D. Cooper, M. A. Johnson, S. D. Christe, and C. J. Gramling, "Lunanet: a flexible and extensible lunar exploration communications and navigation infrastructure," in *2020 IEEE Aerospace Conference*, 2020, pp. 1–14.

[34] J. A. Fraire, O. De Jonckère, and S. C. Burleigh, "Routing in the space internet: A contact graph routing tutorial," *Journal of Network and Computer Applications*, vol. 174, January 2021.

[35] J. Hansen and R. Ghrist, "Toward a spectral theory of cellular sheaves," *Journal of Applied and Computational Topology*, vol. 3, no. 4, pp. 315–358, Dec 2019. [Online]. Available: https://doi.org/10.1007/s41468-019-00038-7

[36] ——, "Opinion dynamics on discourse sheaves," *SIAM Journal on Applied Mathematics*, vol. 81, no. 5, pp. 2033–2060, 2021. [Online]. Available: https://doi.org/10.1137/20M1341088

[37] M. R. Malekpour, "A byzantine-fault tolerant self-stabilizing protocol for distributed clock synchronization systems," in *Symposium on Self-Stabilizing Systems*. Springer, 2006, pp. 411–427.

[38] S. Farrell, H. Weiss, S. Symington, and P. Lovell, "Bundle Security Protocol Specification," RFC 6257, 05 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6257

[39] E. J. Birrane and K. McKeever, "Bundle Protocol Security (BPSec)," RFC 9172, 01 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9172

[40] E. J. Birrane, A. White, and S. Heiner, "Default Security Contexts for Bundle Protocol Security (BPSec)," RFC 9173, 01 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9173

[41] E. J. Birrane, S. Heiner, and K. McKeever, *Securing Delay-Tolerant Networks with BPSec*. Wiley, 12 2022.

[42] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology*, G. R. Blakley and D. Chaum, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 47–53.

[43] A. Kate, G. M. Zaverucha, and U. Hengartner, "Anonymity and security in delay tolerant networks," in *2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops (SecureComm 2007)*, 2007, pp. 504–513.

[44] N. Asokan, K. Kostiainen, P. Ginzboorg, J. Ott, and C. Luo, "Applicability of identity-based cryptography for disruption-tolerant networking," in *Proceedings of the 1st International MobiSys Workshop on Mobile Opportunistic Networking*, ser. MobiOpp '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 52–56. [Online]. Available: https://doi.org/10.1145/1247694.1247705

[45] R. Patra, S. Surana, and S. Nedevschi, "Hierarchical identity based cryptography for end-to-end security in dtns," in *2008 4th International Conference on Intelligent Computer Communication and Processing*, 2008, pp. 223–230.

[46] S. A. Menesidou, V. Katos, and G. Kambourakis, "Cryptographic key management in delay tolerant networks: A survey," *Future Internet*, vol. 9, no. 3, p. 26, 2017.

[47] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proceedings of the 15th ACM conference on Computer and communications security*, 2008, pp. 417–426.

[48] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von

neumann architecture," in *23rd USENIX Security Symposium (USENIX Security '14)*, 2014, pp. 781–796.

[49] M. Rosenberg, J. White, C. Garman, and I. Miers, "zk-creds: Flexible anonymous credentials from zksnarks and existing identity infrastructure," in *2023 IEEE Symposium on Security and Privacy (S & P)*. IEEE, 2023, pp. 790–808.

[50] S. Farrell and V. Cahill, "Security considerations in space and delay tolerant networks," in *2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06)*, 2006, pp. 8 pp.–38.

[51] W. D. Ivancic, "Security analysis of dtn architecture and bundle protocol specification for space-based networks," in *2010 IEEE Aerospace Conference*, 2010, pp. 1–12.

[52] P. Asuquo, H. Cruickshank, Z. Sun, and G. Chandrasekaran, "Analysis of dos attacks in delay tolerant networks for emergency evacuation," in *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies*, 2015, pp. 228–233.

[53] S. Saha, S. Nandi, R. Verma, S. Sengupta, K. Singh, V. Sinha, and S. K. Das, "Design of efficient lightweight strategies to combat dos attack in delay tolerant network routing," *Wireless Networks*, vol. 24, pp. 173–194, 2018.

[54] S. Grasic, E. Davies, A. Lindgren, and A. Doria, "The evolution of a dtn routing protocol - prophetv2," in *Proceedings of the 6th ACM Workshop on Challenged Networks*, ser. CHANTS '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 27–30. [Online]. Available: https://doi.org/10.1145/2030652.2030661

[55] A. Lindgren, A. Doria, E. Davies, and S. Grasic, "RFC 6693: Probabilistic Routing Protocol for Intermittently Connected Networks," *IETF Network Working Group*, 2012. [Online]. Available: https://tools.ietf.org/html/rfc6693

[56] J. Curry, "Sheaves, cosheaves and applications," Ph.D. dissertation, University of Pennsylvania, 2013.

APPENDIX

1. OVERVIEW OF SHEAF LAPLACIANS

Here we provide a brief overview of sheaf Laplacians and describe how they relate to our techniques. The material on sheaf Laplacians is based on [35,36], and general information on sheaves (cellular sheaves, the approach we use here) can be found in [56]. We will work exclusively with finite graphs and finite-dimensional vector spaces, treated as $\mathbb{R}^n$ with the standard inner product $\langle \cdot, \cdot \rangle$; however, many of the ideas here can be generalized to infinite cases and general inner products.

Let G be a graph and write $v \leq e$ to indicate that vertex v is connected to edge e. For our purposes, a *sheaf* $\mathcal{F}$ on G consists of real vector spaces $\mathcal{F}(v)$ and $\mathcal{F}(e)$ for all vertices v and edges e, as well as a linear map $\mathcal{F}(v \leq e) \colon \mathcal{F}(v) \to \mathcal{F}(e)$ called a *restriction map* whenever $v \leq e$. We treat elements of these spaces as column vectors. Define

$$C^0(G;\mathcal{F}) = \bigoplus_v \mathcal{F}(v)$$
$$C^1(G;\mathcal{F}) = \bigoplus_e \mathcal{F}(e),$$

called the spaces of 0-*cochains* and 1-*cochains* respectively. We indicate components of elements in the direct sums with subscripts: for instance, the v component of a vector $x \in C^0(G;\mathcal{F})$ is denoted x_v.

We will need to assign an arbitrary orientation to all edges: for any edge e connecting vertices v_1 and v_2, we set $[v_1;e] = \pm 1$ and $[v_2;e] = \mp 1$ so that $[v_1;e] = -[v_2;e]$. We define the *coboundary map* $\delta \colon C^0(G;\mathcal{F}) \to C^1(G;\mathcal{F})$ in terms of its components: if e connects v_1 and v_2, then

$$(\delta x)_e = [v_1;e]\mathcal{F}(v_1 \leq e)(x_{v_1}) + [v_2;e]\mathcal{F}(v_2 \leq e)(x_{v_2}).$$

Thus, δ can be viewed as a block matrix, with the nonzero blocks given by the matrices representing the linear maps $[v;e]\mathcal{F}(v \leq e)$. The *sheaf Laplacian* $L \colon C^0(G;\mathcal{F}) \to C^0(G;\mathcal{F})$ for the sheaf $\mathcal{F}$ is defined by $L = \delta^T \delta$. It is given explicitly by

$$(Lx)_v = \sum_{v,w \leq e} \mathcal{F}(v \leq e)^T \Big(\mathcal{F}(v \leq e)(x_v) - \mathcal{F}(w \leq e)(x_w) \Big),$$

where w ranges over all vertices connected to v by some edge e. Note that the terms $[v;e]$ do not appear in this expression, so the sheaf Laplacian does not depend on the choice of orientation of edges. The *graph Laplacian* occurs as a special case of the sheaf Laplacian when all vector spaces $\mathcal{F}(v)$ and $\mathcal{F}(e)$ are $\mathbb{R}$ and all maps $\mathcal{F}(v \leq e)$ are identity maps. The sheaf Laplacians we have worked with in this paper have also had $\mathcal{F}(v)$ and $\mathcal{F}(e)$ be one-dimensional but have allowed $\mathcal{F}(v \leq e)$ to be more general linear maps.

The sheaf Laplacian is indeed related to the usual Laplacian. This is easiest to see using the graph Laplacian: in this case, components of Lx are given by

$$\begin{aligned}(Lx)_v &= \sum_{w \text{ adjacent to } v} (x_v - x_w) \\ &= \deg(v)\ x_v - \sum_{w \text{ adjacent to } v} x_w.\end{aligned}$$

This is reminiscent of numerical approximations of second derivatives; approximating the usual Laplacian on a lattice in $\mathbb{R}^n$ will produce a similar formula, but with a change of sign. As with the regular Laplacian, the graph Laplacian provides a measure of curvature of a function on a graph (technically a 0-cochain). It is positive at local maxima and negative at local minima in the graph – reversed from the usual Laplacian's signs at extrema. While the sheaf Laplacian generalizes the graph Laplacian, it serves a similar purpose as it compares the values over two adjacent vertices. Further connections are explored in *Hodge theory*.

The sheaf Laplacian is used to define an analog of the heat equation on the space $C^0(G;\mathcal{F})$, and in this paper we have

examined a discrete-time version given by

$$x_{n+1} = x_n - h\alpha L x_n. \tag{17}$$

Certain properties of the sheaf Laplacian make analysis of this system particularly simple. Since it is defined by $L = \delta^T \delta$, it is symmetric and positive semi-definite. This implies it is diagonalizable by orthogonal matrices and has all nonnegative real eigenvalues.

As in the main body of the paper, we will decompose $C^0(G, \mathcal{F})$ as $C^0(G, \mathcal{F}) = C^0(B, \mathcal{F}|_B) \oplus C^0(Y, \mathcal{F}_Y)$, where B is a subgraph of G and and $Y = G \setminus B$. The coboundary map δ can then be written as a block matrix:

$$\delta = [D_Y \quad D_B]$$

This allows us to write the sheaf Laplacian $L = \delta^T \delta$ in block matrices as

$$L = \begin{bmatrix} D_Y^T D_Y & D_Y^T D_B \\ D_B^T D_Y & D_B^T D_B \end{bmatrix} = \begin{bmatrix} L_{Y,Y} & L_{Y,B} \\ L_{Y,B}^T & L_{B,B} \end{bmatrix},$$

where we let $L_{Y,Y} = D_Y^T D_Y$, and similarly for $L_{Y,B}$ and $L_{B,B}$. Setting

$$M = \begin{bmatrix} L_{Y,Y} & L_{Y,B} \\ 0 & 0 \end{bmatrix}$$

gives a special case of the matrix M considered in Section 3, which treats values over B as boundary conditions. In this case, the discrete-time version of the heat equation becomes

$$x_{n+1} = x_n - h\alpha M x_n. \tag{18}$$

One reason for interest in the sheaf Laplacian is the fact that $\ker L = \ker \delta$ (which follows from noting that $\langle \delta x, \delta x \rangle = x^T L x$ is equal to 0 if and only if $\delta x = 0$). This kernel is called the *space of global sections* or the 0^{th} *cohomology* of $\mathcal{F}$, denoted $H^0(G, \mathcal{F})$. An element of $H^0(G, \mathcal{F}) = \ker L = \ker \delta$ is called a *global section* or simply a *section* of $\mathcal{F}$. The definition of δ then shows a section is a vector $x \in C^0(G; \mathcal{F})$ such that for any edge e with vertices v and w, we have $\mathcal{F}(v \le e)(x_v) = \mathcal{F}(w \le e)(x_w)$. Sections thus represent collections of information over vertices that agree over edges when viewed through the restriction maps.

Elements of $H^0(G, \mathcal{F})$ are steady-state solutions of Equation (17), as they are exactly the x such that $Lx = 0$. Steady-state solutions of Equation (18) can also be interpreted in similar terms. First, if $x = \begin{bmatrix} y \\ b \end{bmatrix}$ is a steady-state solution, then Equation (18) shows

$$y = y - h\alpha L_{Y,Y} y - h\alpha L_{Y,B} b,$$

and adding an element of $\ker L_{Y,Y}$ to y produces another steady-state solution. The space $\ker L_{Y,Y}$ is a 0^{th} *relative cohomology* space, denoted $H^0(G, B; \mathcal{F})$: it can be interpreted as the space of sections of $\mathcal{F}$ that are zero over B. Thus, as long as a steady-state solution exists (which Theorem 2 guarantees if $h\alpha$ is small enough), then the space of Y components of steady-state solutions is $y + H^0(G, B; \mathcal{F})$, where y is the Y component of any particular steady-state solution.

Convergence

The results of Theorem 2 apply to Equations (17) and (18), but in these cases we can give more explicit descriptions of convergence. Suppose we have an initial value x_0 and a sequence $\{x_n\}$ defined by Equation (17). Since L is orthogonally diagonalizable, Theorem 2 and the description of $H^0(G; \mathcal{F})$ above show that for all sufficiently small α, the sequence $\{x_n\}$ in fact converges to the orthogonal projection of x_0 onto $H^0(G; \mathcal{F})$.

Similarly, now suppose we have an initial value x_0 and a sequence $\{x_n\}$ defined by Equation (18). If y_n is the Y component of x_n, then $y_{n+1} = y_n - h\alpha L_{Y,Y} y_n - h\alpha L_{Y,B} b$, where $b = b_n$ is the unchanging B component of x_n. Setting $A_{Y,Y} = I - h\alpha L_{Y,Y}$ and $z = -h\alpha L_{Y,B} b$, this becomes

$$y_{n+1} = A_{Y,Y} y_n + z, \tag{19}$$

so we have the explicit solution

$$y_n = A_{Y,Y}^n y_0 + (I + A_{Y,Y} + \cdots + A_{Y,Y}^{n-1}) z. \tag{20}$$

Note that $A_{Y,Y}$ has the same orthogonal basis of eigenvectors as $L_{Y,Y}$. Furthermore, since $L_{Y,Y}$ has all real nonnegative eigenvalues, as long as α is sufficiently small, the eigenvalues of $A_{Y,Y}$ all lie in $(-1, 1]$; from here on we will assume such an α has been chosen. We have

$$z \in \operatorname{im} L_{Y,B} \subseteq \operatorname{im} D_Y^T \subseteq (\ker L_{Y,Y})^\perp,$$

which means z has zero component in the eigenspace of $L_{Y,Y}$ associated with eigenvalue 0, which is equivalently the eigenspace of $A_{Y,Y}$ associated with eigenvalue 1. Rewriting Equation (19) in the basis of eigenvectors gives

$$\begin{bmatrix} u_{n+1} \\ v_{n+1} \end{bmatrix} = \begin{bmatrix} \Lambda & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} u_n \\ v_n \end{bmatrix} + \begin{bmatrix} w \\ 0 \end{bmatrix}, \tag{21}$$

where Λ is the diagonal matrix consisting of the eigenvalues of $A_{Y,Y}$ not equal to 1. This gives an explicit solution: for all n, we have $v_n = v_0$ and

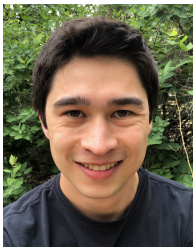
$$u_n = \Lambda^n u_0 + (I + \Lambda + \cdots + \Lambda^{n-1}) w.$$

Since Λ is diagonal with entries in $(-1, 1)$, the first term $\Lambda^n u_0$ approaches 0, and the second term

$$(I + \Lambda + \cdots + \Lambda^{n-1}) w$$

approaches $(I - \Lambda)^{-1} w$. Thus, $\begin{bmatrix} u_n \\ v_n \end{bmatrix}$ converges to $\begin{bmatrix} (I - \Lambda)^{-1} w \\ v_0 \end{bmatrix}$. This describes the convergence of y_n in the basis of the eigenvectors: the top component depends only on the boundary condition, while the bottom component depends only on the initial condition. In fact, $\begin{bmatrix} 0 \\ v_0 \end{bmatrix}$ is the projection of the initial y_0 onto $\ker L_{Y,Y} = H^0(G, B; \mathcal{F})$, written in the basis of eigenvectors.

BIOGRAPHY

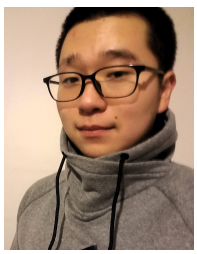


Michael Moy *is pursuing a PhD in mathematics at Colorado State University, having completed his master's there in 2021. His research is focused on applied topology. During the summers of 2020 through 2023, he worked as an intern at NASA through the SCaN Internship Project. His research at NASA has focused on mathematical modeling of networks and approaches to delay tolerant networking in space.*

Alan Hylton *should probably be designing tube audio circuits, but instead directs Delay Tolerant Networking (DTN) research and development at the NASA Goddard Space Flight Center, where he is humbled to work with his powerful and multidisciplinary team. His formal education is in mathematics from Cleveland State University and Lehigh University, and he considers it his mission to advocate for students. Where possible, he creates venues for mathematicians to work on applied problems, who add an essential diversity to the group.*

Robert Kassouf-Short *earned his PhD in mathematics from Lehigh University in 2018. He worked as a Visiting Assistant Professor of Mathematics at John Carroll University until he joined the Secure Networks, System Integration and Test Branch at NASA Glenn Research Center in 2020. His research interests lie in the intersection of abstract mathematics and real world applications. Currently, his focus is on the foundations of networking theory and how to efficiently route data through a network using local information.*

Jacob Cleveland *is a PhD student studying mathematics at Colorado State University. They have a bachelors degree in mathematics from the University of Nebraska at Omaha and a bachelors degree in computer engineering from the University of Nebraska - Lincoln. They joined the Secure Networks, System Integration and Test Branch as a Pathways Intern at NASA Glenn Research Center in 2020. Since joining, they have contributed to several research projects applying pure mathematics to engineering problems in space networking, star tracking, and artificial neural networks.*



Jihun Hwang *(Jimmy) is a third-year Ph.D. student in computer science at Purdue University. He is primarily interested in information-theoretic cryptography and secure (multi-party) computations, but he ultimately likes to talk about any topics in or related to theoretical computer science and computer security. Before Purdue, he studied mathematics and computer science at the University of Massachusetts Amherst.*

Justin Curry *is an Associate Professor of Mathematics and Statistics at the University at Albany, SUNY. Before arriving at Albany in 2017, he was a Visiting Assistant Professor at Duke. Professor Curry earned his PhD in mathematics from the University of Pennsylvania in 2014, under the direction of Robert Ghrist. His research interests include the use of category theory in applied mathematics, with particular emphasis on applied sheaf theory, and inverse problems in topological data analysis (TDA).*

Mark Ronnenberg *earned a Ph.D. in mathematics from Indiana University in 2023, where he was trained in gauge theory and low dimensional topology. He is now an assistant professor of math at Anne Arundel Community College. Outside of math, Mark loves music, books, and video games.*

Miguel Lopez *received his bachelor's degree in mathematics at Boston University and is currently a fourth-year Ph.D. student in applied math at the University of Pennsylvania. Under the supervision of Robert Ghrist, he is studying how algebraic topology can inform network science and machine learning algorithms. Outside of research Miguel is an avid board gamer and rock climber.*

Oliver Chiriac *received his B.Sc. in mathematics from the University of Toronto and is currently a M.Sc. student studying mathematics at the University of Oxford. Prior to this, he has done research in symplectic geometry and quantum field theory. He is a first-time NASA intern and is interested in applying differential geometry and topology to the world of physics, deep learning, and space networks. Outside the realm of mathematics, Oliver devotes his time to soccer, music, and travel.*