Towards Practical Clock Synchronization in the Solar System Internet

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

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

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

Jihun Hwang
Purdue University
hwang102@purdue.edu

Karuna Petwe
University of Washington
kpetwe@outlook.com

Tobias Timofeyev
University of Vermont
tobias.timofeyev@uvm.edu

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

***Abstract*—Clock synchronization remains a notable gap in the Delay Tolerant Networking (DTN) suite of protocols. However, following the great theoretical advances in the area and a highly successful DTN experiment campaign on the International Space Station (ISS), a strong enough foundation has been laid to support a practical clock synchronization protocol and implementation for the Solar System Internet (SSI). In this paper we work with the theoretical developments along with the lessons-learned from the DTN experiments to drive a clock synchronization protocol and implementation for practical SSI network architectures, complete with code and simulation analyses.**

In addition to the recent technical and feature growth of DTN, network architectures have begun taking center stage. This was particularly true with the ISS experiments which operated over a multitude of DTN network boundaries (as well as project and programmatic boundaries), yielding operational experience with proper DTN network architectures. Taking this a step further, it is expected that future users in space will subscribe to multiple service providers, implying multiple network areas and boundaries. Such an architecture is central to the upcoming LunaNet, which notably has nodes that do not neighbor authoritative clock sources.

After covering the basics of DTN, we recall the most germane aspects of clock synchronization for DTNs. This includes remarks on routing in DTNs, which is often schedule-based; we emphasize that this alone demonstrates how crucial clock synchronization is for scalability. The introduction continues with a discussion of the ISS experiment network architectures and its implications for this work.

The key components have been implemented and applied to clock synchronization for a portion of NASA's DTN experiments network. The implementation is openly available and will be integrated into the High-rate DTN (HDTN) implementation. We then cover parameter optimization and the ramifications of choosing underlying equation solvers for convergence. Observations based on network architectures are used to explain the multiple implementation paths available in DTN and which one was chosen. We conclude with the analysis of a simulation based on the ISS network architecture and the future work thereby inspired.

TABLE OF CONTENTS

1. BACKGROUND

Year after year since 2018, records are being broken in the number of satellite launches and deployments [1]. The traditional communications method—scheduled point-to-point contacts—does not scale well and will eventually become too complex to manage. The solution to scalability in communications is *networking*, and in this case, building a Solar System Internet (SSI). Delay Tolerant Networking (DTN) is a suite of protocols that was introduced as a solution to the SSI problem; in particular, DTN supports both scheduled and non-schedule routing. Any realistic solution to SSI will have timing requirements, as some portions of the network will have scheduled contacts, and satellites have pointing, acquisition, and navigation requirements. This paper builds upon past theoretical work to realize a practical time synchronization algorithm for DTNs, which has been a conspicuously missing ingredient.

From its instantiation, DTN was designed with the idiosyncrasies of space systems in mind. While a full introduction is beyond to DTN is beyond the scope of this paper (see e.g. [2–5]), it is also unnecessary; by highlighting a few key differences, the reader can appreciate how traditional approaches (e.g., Network Time Protocol, NTP) do not apply (see also [6] for a more detailed discussion on this):

- *Disconnection*: Due to mobility for example, general nodes or subnetworks cannot be assumed to be connected to others. For any two nodes, end-to-end paths might never exist at any time slice. The usual routing algorithms and protocols hence do not directly apply.
- *Disruption*: For example, due to temporary weather (cloud) obscuration, established links might not be reliable.
- *Latencies*: Round-trip times (RTTs) can be tens of minutes (say, Earth to Mars) or higher (say, Earth to Pluto). Such RTTs preclude the use of the Transmission Control Protocol (TCP) and other more "garden variety" acknowledgment-based protocols.
- *Variance of latencies*: The same network with Earth-to-Mars links might also have Earth-to-low Earth orbit (LEO) links, meaning that some links will have single-digit millisecond latencies. Different underlying transport protocols will likely be used also.

Continuing on this last point, DTN works as an overlay proto-

col, and DTN nodes can have "any" data paths between them; these could be point-to-point protocols, such as Advanced Orbiting Systems (AOS), or end-to-end protocols, such as TCP/IP. With the heterogeneity of a typical DTN becoming apparent, a natural question is how to perform routing; indeed this is an active area of research [7]. A common thread is that while sub-networks of a DTN might support opportunistic routing, there will always be scheduled components, particularly when round trip times (RTTs) preclude any semblance of feedback or if asset use is contentious. For scheduling to make sense, it is necessary for the clocks in the network to be synchronized *to some degree* [8–12].

This work continues a series of papers that developed the theoretical foundations for a time synchronization methodology designed for DTNs [6, 13]. In these papers, the limitations of synchronization (i.e. consensus) in a disconnected network are discussed, and an approximative algorithm is proposed that allows for the estimation of and control for error. In this paper, we discuss a practical implementation of the algorithm that works with the High-rate DTN (HDTN) implementation of DTN [14]. After a brief introduction, we discuss the implementation details, experiments, and observations made. Particular attention is paid to how this protocol would function within a representative network architecture. The software is open-source, and a link is provided in the paper.

Time Synchronization

While DTN has usually been used to realize small, simple networks, the upcoming lunar network known as LunaNet shows that more involved network architectures will be required [15]. LunaNet will feature a topology with multiple hops and multiple nodes – including the separation of nodes from 'authoritative clock sources' – and the difficulty is compounded by programmatic realities as well: there will be multiple space network service providers for the Lunar assets (across agency and country boundaries). This takes DTN beyond its typical use with a flat topology, which has only recently been accomplished [16]. The work herein attempts to answer the recognized needs for DTN time synchronization both technically [17] and politically [18]. We also note that relativistic effects *do in fact matter*, and otherwise identical clocks set in Earth and Lunar orbits will tick at different rates; we do not correct for these factors here. However, the theoretical paper this present work is based upon was designed from the beginning to generalize to include such factors. This is reserved for future work.

When designing and implementing any clock synchronization algorithm, it is further essential that it can satisfy the varying time precision requirements for all systems present in the network. Define *coarse* accuracy as having all clocks within a network be synchronized with a maximum pairwise relative error between 1 ms to 1 second, *fine* accuracy as between 1 µs to 1 ms, and *precision* accuracy as between 1 ns to 1 ms. For example, space networking implementations such as HDTN can function at the coarse accuracy threshold. In contrast, tasks like command-and-control are recommended to be operated at fine accuracy thresholds. Furthermore, tasks related to navigation or science, especially when measurements are taken from multiple vehicles, likely need to be operated at precision accuracy in order to function.

It is important to note that there is more overhead associated with better clock synchronization accuracy, which should be accounted for when designing and implementing any clock synchronization solution. Certain platforms may support and/or require higher accuracy, but the selection should be based on the task at hand.

Introduction and Paper Organization

This work is a continuation of the effort toward the development, deployment, and standardization of a practical clock synchronization protocol for solar-system-scaled networks. In particular, this paper is a sequel to Moy et al. [13] and Hylton et al. [6].

Hylton et al. [6] formulated the clock synchronization problem over Solar System Internet as a local-to-global approximated asynchronous consensus problem, motivated by the previous line of works [19–22] on modeling space communications with sheaves and the celebrated FLP impossibility theorem [23]. Moy et al. [13] followed up on [6] by numerically modeling the said asynchronous consensus problem using sheaf Laplacian on graphs [24, 25]. More specifically, they modeled distributed consensus as 'diffusion of values' [25] where the solution to initial value problems for the heat equation corresponds to the value that nodes reach consensus on, and proved that Euler's method guarantees convergence within a reasonable range of errors and rate.

The focus of this paper is on the actual implementations and experiments of the mathematical models and algorithms built in the prior work [13]. The technical portion of this paper can be divided into the following three consecutive parts:

(1) Section 3 presents some results of numerical simulation of the sheaf Laplacian heat equation, to establish the experimental foundation and soundness of our protocol.

(2) Section 4 showcases the implementation of the protocol proposed in [13] by highlighting key implementation details such as processes and subprocesses. Our implementation is also open-source – see [26, main branch].

(3) Section 5 discusses the setup and results of experiments of our protocol conducted on the Laser Communications Relay Demonstration (LCRD).

We begin by presenting background materials in Section 2 to aid in understanding this area of research and our contributions. After discussing our technical works and results, we conclude the paper in Section 6 with future directions in mind and for readers.

2. PRELIMINARIES

DTN Architecture

Recall that DTN is an overlay network; it adds an extra abstraction layer (with respect to the OSI model) called the *bundle layer*. The bundle layer is typically depicted as a layer between application and transport layers due to its functionalities, but depending on the level of overlay or the architecture of the underlying network, it can reside within the application layer as an application implemented in end nodes. From DTN's viewpoint, a bundle is the primary unit of data (analogous to packets). It is typical to refer to DTN functions and so forth with the Bundle Protocol, or BP. We call out two particular implications of DTN in the context of time synchronization:

- Bundles operate a high layer, and as the height in the stack increases, timing tolerances decrease.
- Because DTN is an overlay network, it *cannot* see what lies between adjacent bundle hops. This connection might be point-to-point over a tightly controlled field programmable gate array (FPGA), or it could be across the Internet. Timing

tolerances cannot be guaranteed.

Numerical Methods for ODEs

We provide below a brief survey on numerical methods commonly used to solve initial value problems (IVPs) for ordinary differential equations (ODEs). This section is only meant to be a brief review for building intuitions for the later parts of this paper; we invite the readers to refer to well-established textbooks such as [27–29] for more detailed overviews and explanations.

Prior work that this paper sets its foundation on [13] analyzes only Euler's method, but a number of other methods suitable for the setting exist. Euler's method, while the most (time) efficient and easiest to comprehend, is known to suffer inaccuracies in practice, especially for equations that change rapidly near initial values.

Let f be a real-valued function, and define the first order ODE $x'(t) = f(t, x)$. The IVP associated to this ODE and a given initial condition is set as follows:

$$x'(t) = f(t, x(t))$$
$$x(t_0) = x_0$$

for some initial values (t_0, x_0). If f is Lipshitz continuous, the Picard-Lindelof theorem guarantees the existence of a unique solution $x(t)$ to this ODE on a closed interval around the initial time t_0.

Intuitively, the goal would be to calculate $x(t)$ by 'integrating' $x'(t)$, which is $f(t, x)$. We achieve this numerically by evaluating $f(t, x(t))$ at each interval of time recursively, computing the 'area' underneath and above the curve $f(t, x(t))$, and adding all the areas (again, recursively). Notice that directly adopting numerical integration:

$$y(t) = \int_{t_0}^{t} f(\tau, y(\tau))d\tau + y_0$$

is invalid, as the integrand depends on $y(t)$, which is unknown. The idea should rather be, for small Δt,

$$\begin{aligned} y(t + \Delta t) &= \int_{t}^{t+\Delta t} f(\tau, y(\tau))d\tau + y(t) \\ &\approx f(t, y(t))\ \Delta t + y(t) \end{aligned} \tag{1}$$

and iterate this over the entire interval of time.

While the methods that we introduce below are primarily for first-order ODEs, they can be applied to higher-order ODEs that can be expressed as a system of first-order ODEs, such as the heat equation.

Euler's Method—This method views $f(t, x)$ as a function for the slope of the tangent line of $x(t)$, and performs a first-order (linear) approximation of it. Let h be the time step so we have $t_n = t_{n-1} + h = t_0 + nh$ at step n, then x at step $n + 1$, denoted as x_{n+1}, is computed as:

$$x_{n+1} = x_n + hf(t_n, x_n)$$
$$t_{n+1} = t_n + h$$

One can also notice that this is the direct consequence of the observation we made in Equation (1). In addition, the time step technically does not have to be the same throughout the entire integration. Δt is hence the more accurate alternative notation than h, but for simplicity and consistency, we shall write it as h unless necessary.

Trapezoidal Method—Recall the trapezoidal rule for approximating integration of f on a finite interval $[a = t_0, b = t_N]$:

$$\int_a^b f(t)dt \approx \frac{b-a}{2}(f(a) + f(b))$$

We obtain the trapezoidal method by applying this to the IVP settings.

$$x_{n+1} = x_n + \frac{h}{2}(f(t_n, x_n) + f(t_{n+1}, x_{n+1}))$$
$$t_{n+1} = t_n + h$$

Runge-Kutta Method— The approximation formula for the trapezoidal rule can be generalized for better accuracy using the Lagrange interpolation polynomial as follows:

$$\int_a^b f(t)dt \approx \sum_{k=0}^{N} f(t_k) \int_a^b \left(\prod_{j \neq k} \frac{t - t_j}{t_k - t_j} \right) dt$$

where $a = t_0 < \cdots < t_N = b$ here is the partition of $[a, b]$. One can see that the $N = 1$ case is equivalent to the trapezoidal rule discussed above. The case where $N = 2$ is referred to as Simpson's rule.

The 2nd-order Runge-Kutta method (RK2) is essentially a two-step Euler's method where, unlike the previous formula, the 'slope' is computed both at the beginning of the interval (t_n) and in the middle $(t_n + h/2)$ for better accuracy. More precisely, x value at the midpoint $x(t_n + h/2)$ is first 'estimated' using Euler's method, then $x(t_{n+1}) = x((t_n + h/2) + h/2)$ is 'refined' using the estimation of $x(t_n + h/2)$.

$$x_n^* = x_n + \frac{h}{2} f(t_n, x_n)$$
$$x_{n+1} = x_n + hf\left(t_n + \frac{h}{2}, x_n^*\right)$$
$$t_{n+1} = t_n + h$$

This particular version of RK2 is also known as Heun's method. Another widely-used version exists (called implicit RK2), but we will not discuss it here as it was not used in our simulations.

The 4th-order Runge-Kutta method (RK4) is similar except the estimation takes place three times (k_2, k_3, and k_4 below)—four times including the initialization step $k_1 = f(t_n, x_n)$—using Midpoint method:

$$k_1 = f(t_n, x_n)$$
$$k_2 = f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1\right)$$
$$k_3 = f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_2\right)$$
$$k_4 = f(t_n + h, x_n + hk_3)$$

which are then refined using Simpson's rule:

$$x_{n+1} = x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$t_{n+1} = t_n + h$$

Expressions for the general N-th order Runge-Kutta method are provided below for the sake of completeness, but our simulations (Section 3) involved only RK2 and RK4.

$$k_i = f\left(t_n + c_i h, z_n + h\sum_{j=1}^{i} a_{ij}k_j\right)$$

$$x_{n+1} = x_n + h\sum_{i=1}^{N} b_i k_i$$

where $\sum_{i=1}^{N} b_i = 1/h$.

Sheaf Laplacian Heat Equation (SLHE)

A graph G is a pair (V, E), where V is a set of vertices, and $E \subseteq V \times V$ is a set of (un)ordered vertex pairs called edges. This mathematical structure is often used to model networks of interacting entities. Throughout this paper, we shall assume that our communications networks are modeled as graphs and the structure of our network is well modeled by pairwise interactions between vertices.

Sheaves refer to a mathematical data structure that not only associates information to regions or locales of a geometric (topological) object, but also expresses how these regions relate to one another in terms of consistency. For networks, we are interested in *sheaves on graphs*, which attach data to nodes and the links joining them; the reader may refer to [7, 20] for details and examples. The typical tag line of sheaves is that data combined with the notion of consistency allows local observations to either be patched together into global observations, or explain why and where this fails. Taking this a step further, sheaves can be used to find extents of consensus, which in a disconnected network cannot be guaranteed to be global.

Certain theoretical and algorithmic tools from graph theory can be generalized to sheaves on graphs, and a particularly powerful case of this is called the *Sheaf Laplacian* [24]. This tool is what enables the application of (heat) diffusion to approximate time diffusion over our graphs, and in a computational way. Details can be found in [13], the direct and theoretical predecessor to this paper, which brought its motivations from [24, 25]. The nature of this paper is more application-oriented, so we begin with a review of the heat equation and show how practical considerations of solving it numerically lead to an implementation of the time synchronization algorithm, error estimation, and even future directions for refinement.

The *heat equation*

$$\frac{\partial u}{\partial t} = \alpha \Delta u \tag{2}$$

describes how heat flows through a physical material over time. It describes the heat transfer, or the time derivative on the left, as proportional to the squared spatial gradient of the temperature. This means that more heat energy in a unit of space relative to those around it results in faster diffusion of that heat across the space. Moy et al. [13] describes how to adapt the heat equation to model time synchronization in a network. This model was the basis for the implementation described in Section 4.

Fix a graph $G = (V, E)$ as a set of vertices V and a set of edges $E \subseteq V \times V$. If $\vec{u}(t)$ is a $|V|$-dimensional vector of functions of time, then the *sheaf Laplacian heat equation* (SLHE) models how $\vec{u}(t)$ changes over time.

$$\frac{\partial u}{\partial t} = -\alpha \mathcal{L} u \tag{3}$$

For ease of implementation and analysis, we prioritized considering the conditions (graphs) where the sheaf Laplacian is equal to the well-known graph Laplacian, defined as

$$\mathcal{L} = \mathcal{D} - \mathcal{A} \tag{4}$$

where D is a diagonal matrix containing the degree of each vertex and A represents the adjacency matrix of the graph. This form of the Laplacian can be thought of as a generalization of the Laplace operator found in the original heat equation. We may note, however, that the networks this modified heat equation is applied to have a much more challenging connective structure than you would see from a standard discretization of a plane.

3. Numerical Simulation of the SLHE

In this section, we bring the theoretical design of the clock synchronization protocol to life through numerical simulations.

Finite Difference Methods for the SLHE

Finite difference methods have been well studied for their ability to approximate solutions to differential equations when it is infeasible to solve them. We will first discuss the application of multiple numerical methods that were investigated on an example network.

Euler's Method

Recall that the time synchronization algorithm approximates the solution to the heat equation by using Euler's method:

$$x_{n+1} = x_n - h\alpha \mathcal{L} x_n$$

to incrementally update clock times until they reach an explicitly defined threshold for convergence. As discussed in [13], the primary advantage of using Euler's method is that it allows for distributed computations among the different clocks. This means different computers can calculate their own graph Laplacian based on their local understanding of their neighbors and perform updates based on what clock times they receive. Hence, individual nodes in a graph do not need global knowledge of the network in order to calculate the next step in the heat equation.

Theoretical analysis of the heat equation where space is discretized on the number line (and uses the standard Laplacian matrix) have shown that, when relying on the method of lines to represent a system of equations as an ODE and iteratively approximate the solution with respect to time, using alternative numerical approximation methods yields different accuracy and solution stability (and thus the behavior of clock convergence) compared to using Euler's method. Since construction of the more generalized graph or sheaf Laplacian matrices are directly dependent on the network layout (as the coefficients in the matrix are dependent on connections in the graph and which nodes are sharing their clock times with their neighbors) [13], if an alternative numerical method allows for greater flexibility in parameter and network setup (due to having a theoretically faster convergence rate or a

greater stability region) and its calculation can be distributed (or the coefficients used by each node can be sent to different computers in a similar fashion as that described in the current protocol), then we will have found potential areas for design improvements to the clock synchronization algorithm. We discuss three alternative methods for investigation below (Runge-Kutta 2, Runge-Kutta 4, and the trapezoidal method), along with a theoretical example further demonstrating the necessity for further inquiry.

Runge-Kutta Methods

As defined in Section 2, the Runge-Kutta (or RK) methods are a family of multi-step methods for approximating solutions to ODEs. Most popular among them are the second-order RK2 and forth-order RK4 methods, which have been built into numerical solvers such as `ode23` [30] and `ode45` [31] in MATLAB, respectively. Let us revisit the generalized RK2 method, of the form:

$$x_n^* = x_n + \frac{h}{2} f(t, x_n)$$
$$x_{n+1} = x_n + hf\left(t + \frac{h}{2}, x_n^*\right)$$

Let $f(h, x_n)$ represent the calculation of our second derivative $\mathcal{L}x_n$, and assume that h is the smallest unit of time on a computer in [13] (and thus the clock time at t equals the time at $t + \frac{h}{2}$). As applied to the heat equation (Equation (3)), RK2 takes the form of:

$$x_n^* = x_n - \frac{h\alpha}{2}\mathcal{L}x_n$$
$$x_{n+1} = x_n - h\alpha\mathcal{L}x_n^*$$

Similarly, we can represent RK4 as:

$$k_1 = \mathcal{L}x_n$$
$$k_2 = \mathcal{L}(x_n - \frac{h\alpha}{2}k_1)$$
$$k_3 = \mathcal{L}(x_n - \frac{h\alpha}{2}k_2)$$
$$k_4 = \mathcal{L}(x_n - h\alpha k_3)$$
$$x_{n+1} = x_n - \frac{h\alpha}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Trapezoidal Method

When applied to the heat equation (Equation (3)), the trapezoidal method is of the form:

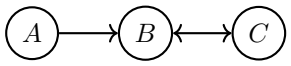
$$x_{n+1} = \left(I + \frac{h\alpha}{2}\mathcal{L}\right)^{-1}\left(I - \frac{h\alpha}{2}\mathcal{L}\right)x_n$$

where I is the identity matrix. Interest in the trapezoidal method derives from previous research and derivation of the Crank-Nicolson method [32], which relies on a second-order central difference scheme for discretizing space and the trapezoidal method for discretizing time. On the standard heat equation (as defined on the number line, as opposed to a graph), the Crank-Nicolson Method has been proven to be unconditionally stable [33, Chapter 3]. While the trapezoidal method is currently limited because it is an implicit method (and thus requires solving a system of equations, which is more difficult of a computation to distribute efficiently among computers), if a computationally fast distributed algorithm is found and similar results are derived for the generalized graph Laplacian matrix, such a method would be ideal for increasing network layout flexibility.

Examples

Example 1: Toy Example

Let us consider a three-node network as depicted below.



Assume that clock A, containing the true time (and thus representing a boundary condition as described in [13]), diffuses its time to clocks B and C while clocks B and C exchange information with each other.

Following the conventions used in Moy et al. [13], the appropriate Laplacian matrix for this graph is as follows:

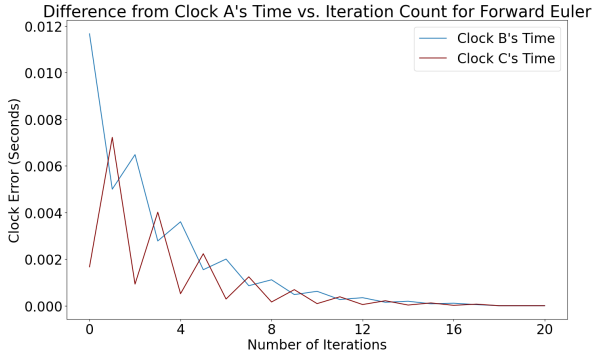
$$\mathcal{L} = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$$

In this matrix, the first row corresponds to clock B's difference in clock value from clock A, and the second corresponds to clock C's difference in clock value from clock A.

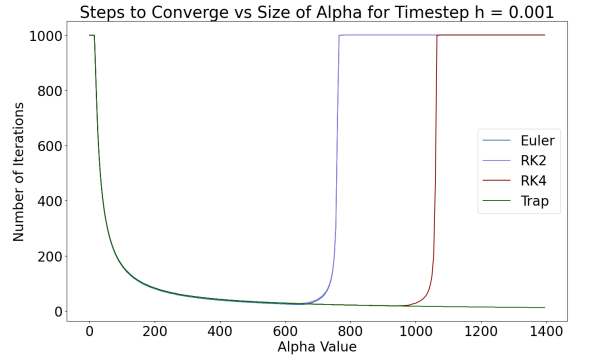
In our example, we let clock A's time be 40023.05400 seconds, clock B's time be 40023.04500, and clock C's time be 40023.06700, and the threshold for convergence be 1×10^{-5}. One can observe in Figure 1a that when ideal coefficients are selected (such as $h\alpha = 2/3$ in this figure), the clock error decays at an overall exponential rate, as theoretically proven already.

When varying the scaling coefficient's α value between 0 and 1400 for a time-step of $h = 0.001$ seconds, we find in Figure 1b that, using a cutoff of 1000 iterations to represent method instability or slowness, the Trapezoidal method appears to outperform Euler's method in both convergence rate and stability, requiring less than or as many iterations as Euler's method for each tested alpha value and having a greater range of alpha values where the number of iterations did not exceed the cutoff.

RK4 appears to have outperformed Euler's method in terms of method stability (and in convergence rate at its most optimal parameter of $\alpha = 920$), while RK2 appears to have approximately the same stability and slightly worse convergence than Euler's (hence the overlap on the Figure 1b), as the minimum number of iterations required for the clocks to reach the true time was found to be 27 for RK2 and 23 for Euler's method at $\alpha = 625$. While more examples or a theoretical stability derivation with respect to the sheaf Laplacian heat equation will provide us with a more confident assessment, efficient distributed algorithms for the trapezoidal and RK4 methods could potentially make them alternative candidates for the clock synchronization design. Furthermore, it is also possible to consider if additional modifications to these tested numerical methods (via equation or implementation) will provide us with improvements to the current protocol in terms of stability and convergence rate, as we are not as concerned with perfectly modeling the heat equation as we are with efficiently reaching clock convergence.



(a) Verification of exponential decay rate



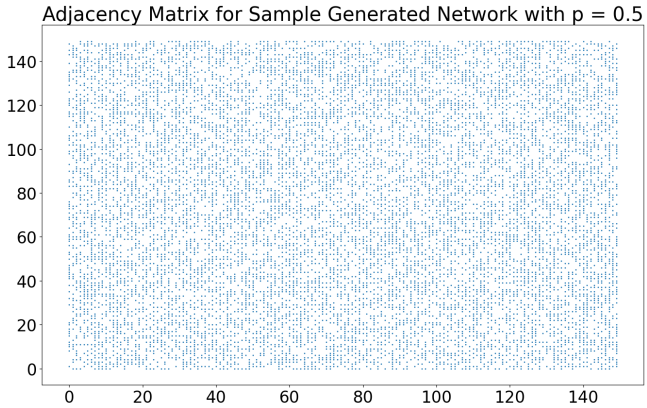
(b) Parameter variation across different methods

Figure 1: Toy Example Results

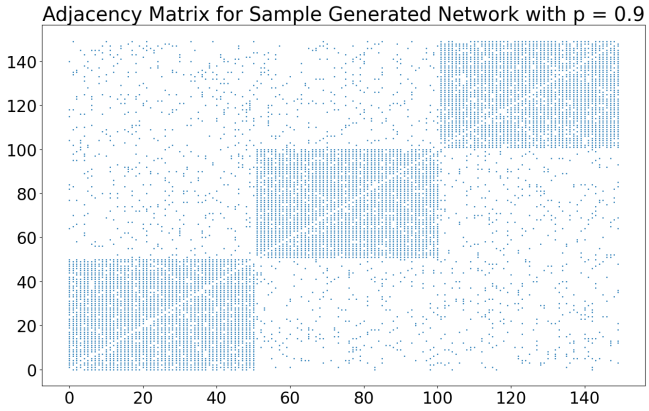
Example 2: 3-Cluster Network

For a larger scale example, we will next consider a network generated using a stochastic block model. Stochastic block models have been studied extensively for their applications in community detection; we encourage readers to read [34] and [35] for further details on their theoretical properties. For our intents and purposes, we are using this model simply because it allows us to efficiently generate a node-adjacency matrix for a sample network, which we can convert into a Laplacian matrix and utilize for our time synchronization example. Various prior works suggest that it is fair to assume the upcoming SSI will contain (or will be able to form) densely connected sub-networks and "gateway" nodes between neighborhoods [16, 22, 36–39]. An example demonstrating how the numerical behavior of the protocol itself may be affected by varying degrees of clustering can provide another viewpoint on how network layouts may affect convergence, beyond number of nodes. In particular, this example is furthermore motivated by the previous work on SDN-based DTN architecture for large-scale space communications [40], which relied on the clustering of nodes to diminish the effects of disruptions within the control channel (i.e., facilitate better connections between the controller and network nodes), and to make better routing decisions by distributing contact plans more reliably and gathering the latest updates on network traffic (within the cluster) more promptly [36, 40].

Define three clusters with each containing $m = 50$ nodes, and let p be the probability that two nodes of the same cluster are connected to each other and $q = 1 - p < p$ be the probability that two nodes from different clusters are connected to each other. Using a uniform sampling distribution, we can randomly assign connections between nodes using these probabilities. In our node-adjacency matrix, we will represent nodes $0, \ldots, m-1$ be those belonging to cluster 1; $m, \ldots, 2m-1$ be those belonging to cluster 2; the remaining nodes belong to cluster 3. We can then calculate the graph Laplacian using Equation (4). A visualization of the connections between nodes in these sample generated networks can be seen in Figure 2a and Figure 2b.



(a) Sample Network with no clustering ($p = 0.5$)



(b) Sample Network with high clustering ($p = 0.9$)

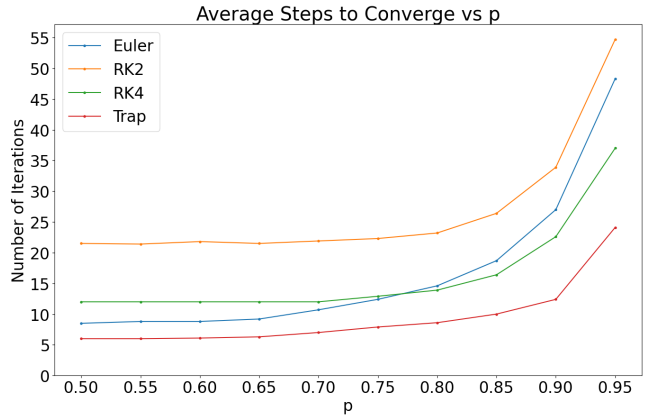
Figure 2: Sample Generated Networks

We will further define our initial time values by adding random noise between $[-0.2, 0.2]$ to 40023.054, and let $h = 0.001$ seconds and our threshold for convergence be 1×10^{-5} seconds as in our first example.

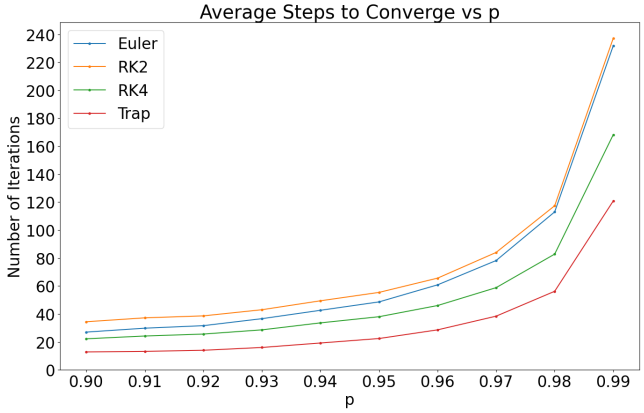
In this example, let us consider how the minimum number of iterations it takes for our sample networks to converge is affected by the degree of clustering of nodes in the network. First, we generate networks by varying p to be between 0.5 (all nodes are equally likely to be connected to each other) and 0.95 (nodes belonging to the same 'community' are most likely to be connected compared to nodes from differing communities), at intervals of 0.05. For each numerical method, we then approximate a 'minimum' number of iterations for our network's convergence by testing the algorithm on a range of α values between 1 and 65. Our results from Figure 3 represent the average number of iterations needed for our clock times to converge across 10 trials (or across 10 randomly generated networks).

From Figure 3a, we notice that for network layouts where

there are significant levels of clustering (i.e. $p = 0.85$ or above), we should expect for clock convergence to take a greater amount of time to be reached regardless of whether we use the original protocol or are considering an alternative method. Observation of the iterative counts for networks generated with values of p between 0.9 and 0.99 in Figure 3b further supports this. It hence becomes likely that we will observe broader time synchronization across our network to experience delays due to clustering between nodes, and thus this will need to be taken into consideration in future implementations.



(a) Convergence behavior for networks with clustering from $p = 0.5$ to $p = 0.95$



(b) Convergence behavior for networks with clustering from $p = 0.9$ to $p = 0.99$

Figure 3: Clustering Example Results

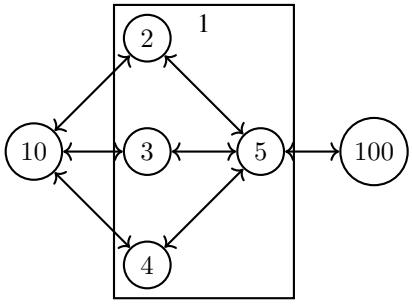
Simulation

We will now discuss work done to create and run a simulation of the time synchronization protocol using a simplified network model based on the recent Laser Communications Relay Demonstration (LCRD) DTN test campaign [16]. As discussed in [41], LCRD is a relay satellite that is connected to three ground stations in California, New Mexico, and Hawaii, but is communicating with only one ground station at any point in time when also communicating to an asset in low Earth orbit (LEO). Furthermore, local understanding of the network topology differs between nodes due to their location/subnetwork in the network as these may be separated by project or programmatic boundaries. The result is that globally the data are incomplete, which algorithmically can result in contradictory setups for the global and local Laplacian matrices. As we expect such characteristics to appear within the larger SSI, running a simulation on a simplified layout will allow us to focus on how the theoretical behavior of the system may be affected without external implementation issues taking precedence.

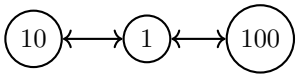
The LCRD DTN LCRD experiment campaign demonstrated the first instance of a space network service provider, a DTN architecture that enables terrestrial-like network structures; this inspired the network architecture used in this paper, and will lead to more comprehensive tests in the future with multiple space network service providers.

Simulation Setup

The network layout for our simulation is represented below: Nodes 2, 3, 4, and 5 are ground stations, which altogether are abstracted into an imaginary node called Node 1.



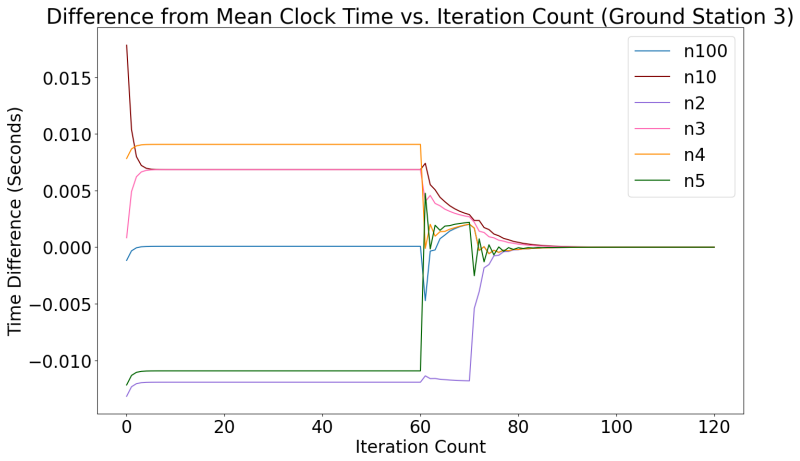
This abstraction was made to represent how nodes outside of the LCRD ground station network (i.e. Nodes 10 and 100 in this example) are forwarding their times to a public address (an imaginary node) "1" because their internal understanding of the network layout prevents them from knowing which ground station the LCRD satellite is connected to. In this simulation, we represented the satellite's connection to an unknown ground station by forwarding the clock time of Node 10 to one of Nodes 2, 3, or 4, which was selected by reading the time at which the simulation was running.



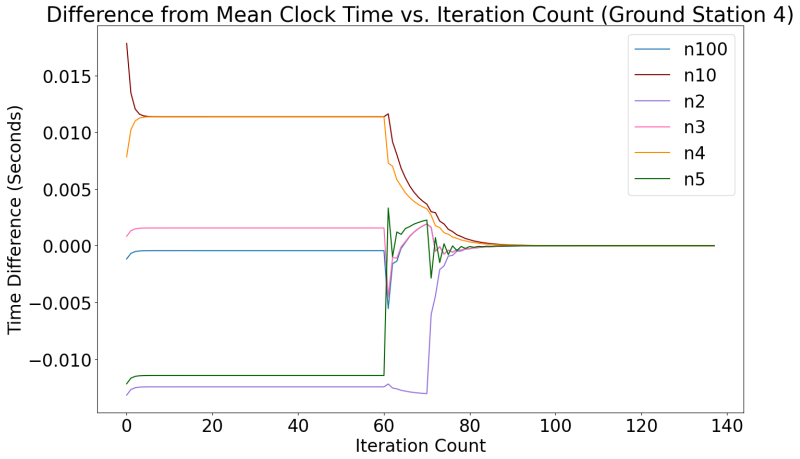
In our first run of the simulation, we primarily focused on observing general convergence behavior without accounting for delays between sending and receiving times. We varied the initial times of the clocks within a range of .02 seconds and set the scheduled send times equal to the scheduled update times. More specifically, our initial conditions were as follows:

Node	Initial Time (seconds):
10	40023.0740
2	40023.0430
3	40023.0570
4	40023.0640
5	40023.0440
100	40023.0550

If the clock value of a node fell within the range of 40023.0500 to 40023.0901 seconds, the clock would send its time to its neighbors and perform Euler's method with the



(a) Convergence behavior while connected to ground station 3



(b) Convergence behavior while connected to ground station 4

Figure 4: Simulation Preliminary Results

clock values it received at that iteration. All clocks were incremented by 0.0001s at the end of each iteration of the simulation. This would allow us to look at an example of what would happen if there was an initial delay in a computer's ability to update its clock due to it being off-sync with its neighbors.

Results and Next Steps

Preliminary results from Figure 4 suggest that the convergence of clock times throughout a network can occur even when local knowledge of network topology is contradictory to the global topology. Furthermore, we notice that changes in which ground station the LCRD is connected to will affect convergence behavior among clock times in the graph, as evidenced by Node 10's immediate convergence with Node 3's time in Figure 4a versus Node 4's time in Figure 4b.

Notice also that the nodes that are not directly connected to Node 10 (with the exception of Node 2, which has a delayed update) tend to converge with each other before converging with the remaining nodes. It is possible that more densely connected nodes (ie. nodes with a higher degree) will have a greater influence on the final clock convergence time in the network compared to more sparsely connected nodes, and thus this behavior will likely need to be accounted for when designing send and routing schedules for updating various nodes.

Combining our results of the preliminary simulation with the behavior observed in the previous example with 3-cluster networks, it is further likely this clustering of clock synchronization among nodes in the graph may affect the global synchronization of clock times, depending on parameters such as initial conditions and send schedules to neighbors, and this can be the premise for further tests or simulation experiments conducted on larger scale networks. Further expansion to the current simulation design would include adding functionality for representing the asymmetric data rate or delays between clock times being sent and received, as well as transitioning towards running simulation tests that record the simulation's running time instead of counting the number of iterations before global time convergence is reached.

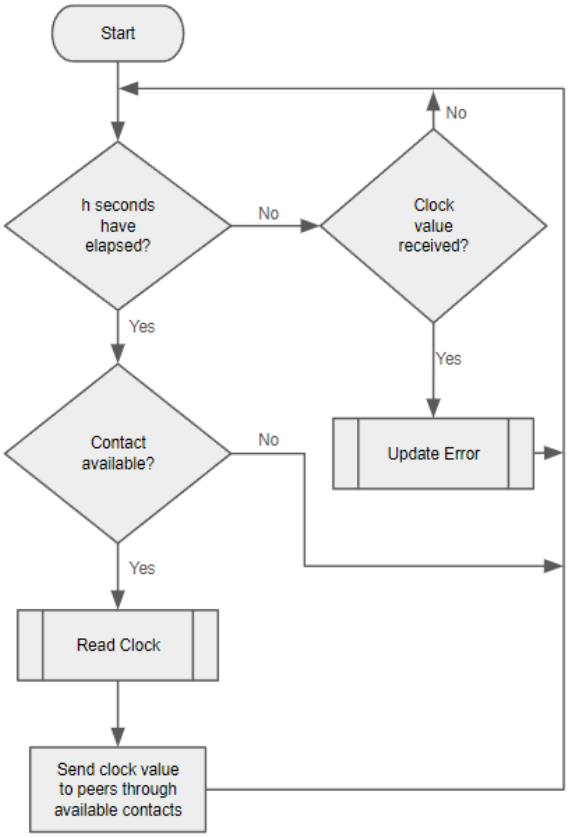


Figure 5: Overall time synchronization process depicted as a flow chart

4. IMPLEMENTATION

In this section, we describe our implementation of the method described in [13]. The code for the time synchronization implementation is available online in [26, main branch] and it does not require a DTN to run. For the DTN experiments, we note that several implementations of DTN are available; the High-rate DTN (HDTN) project was chosen as HDTN was used for the LCRD experiments. Some HDTN-specific comments are necessary; the interested reader can obtain the HDTN code, user's guide, and introductory material if necessary [14].

The implementation approach uses UDP, with code based in C, to transfer clock values between nodes in a network. In order to avoid possible influence by other time synchronization processes running on our experiment nodes, the clock value of a node is read using the C function `clock_gettime` and uses the `CLOCK_MONOTONIC_RAW` clock source as in Figure 6(a).

This clock source represents a monotonic hardware-derived time relative to some unspecified start time and is not modified or updated during normal operation. Rather than update the system clock value directly, a local variable stored the total accumulated error from each time step. Each time a clock value is received, the difference between a node's clock value and the receive value is computed. The result is then multiplied by the coefficient α corresponding to that peer. This is what accomplishes the matrix multiplication in the sheaf Laplacian heat equation but in a distributed manner.

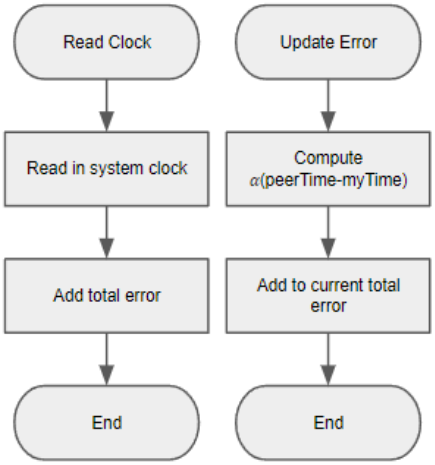


Figure 6: (a) Left: Subprocess for reading clock value using total current error. (b) Right: Subprocess for updating error upon receiving a clock value

Whenever the clock value of a node is read, this accumulated error is added to the clock value read as in Figure 6(b). This corrected clock value is then transmitted and shared among any neighbor nodes as in Figure 5.

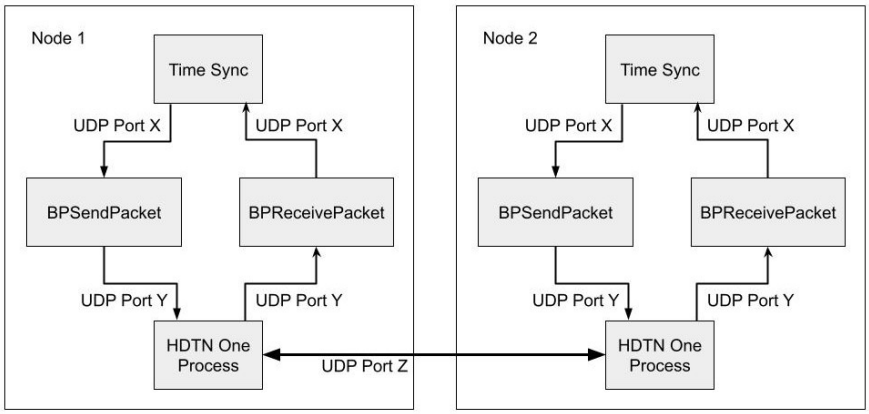


Figure 7: Diagram showing the connection of processes for time synchronization using HDTN

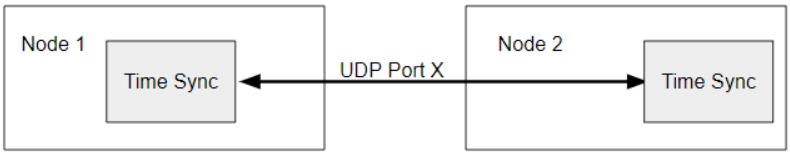


Figure 8: Diagram showing the connection of processes for time synchronization without HDTN

The use of sockets enabled us to easily configure whether HDTN is used or not to transfer the clock values. If HDTN is desired, the internal UDP sockets are set to send clock values to a `BPSendPacket` process and receive clock values from a `BPReceivePacket` process as in Figure 7. If time synchronization without HDTN is desired for a particular test, the UDP sockets are configured to communicate directly with each neighbor of the node in the network as in Figure 8.

5. Experiments

For the following experiments, we consider that 'convergence' has been achieved when the relative error is a significant fraction of the initial relative error. Generally, this fraction is 1%. However, note this is just a rule of thumb, and depending on the situation, certain fractions will not be achieved.

Experiment 1

In the first experiment, we explored the difference between using our time synchronization method with HDTN versus performing synchronization without HDTN. As a proof of concept, only two nodes were included in the network as in Figure 9.

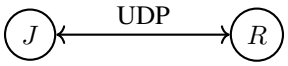


Figure 9: Peer-to-peer network used for Experiment 1

For the first run of Experiment 1, time sync was performed using HDTN as in Figure 7. The result was that synchronization was severely limited by the time `BPSendPacket` and the `HDTNOneProcess` take, as is seen in the second plot of Figure 11. Regardless, synchronization within 1s was achieved. This raises the important question of whether time synchronization should be performed at the bundle layer or perhaps at a lower layer.

For the second run of Experiment 1, time sync was performed without HDTN as in Figure 8. The result was that clock values were able to be directly sent over UDP with minimal latency at the transport layer. As can be seen in the second plot of Figure 12, synchronization within around 2ms was achieved. Further still, depending what synchronization requirements are necessary, the transport layer might have too much latency.

Experiment 2

In the second experiment, we performed time synchronization between two nodes as in Figure 10, this time over a highly variable latency link using TCP.

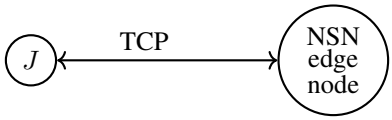


Figure 10: Peer to peer network used for Experiment 2

Moreover, this link included a firewall. Even though this used TCP and had variable latency, time synchronization within 40ms was achieved. Since the Network Time Protocol (NTP) is designed for high latency terrestrial links, this experiment suggests our method is not appropriate for all links. See Figure 13 for the plot.

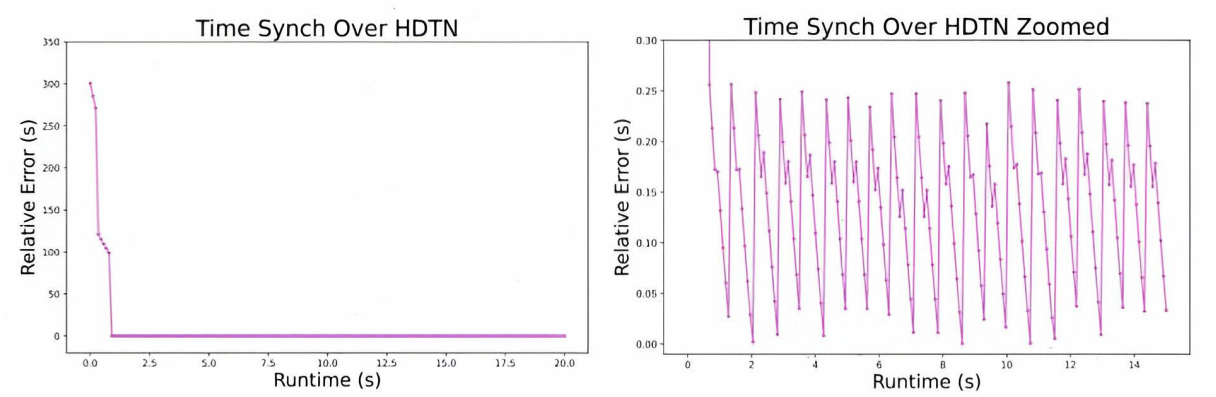


Figure 11: Experiment 1 – Plot of relative errors when performing time sync with HDTN

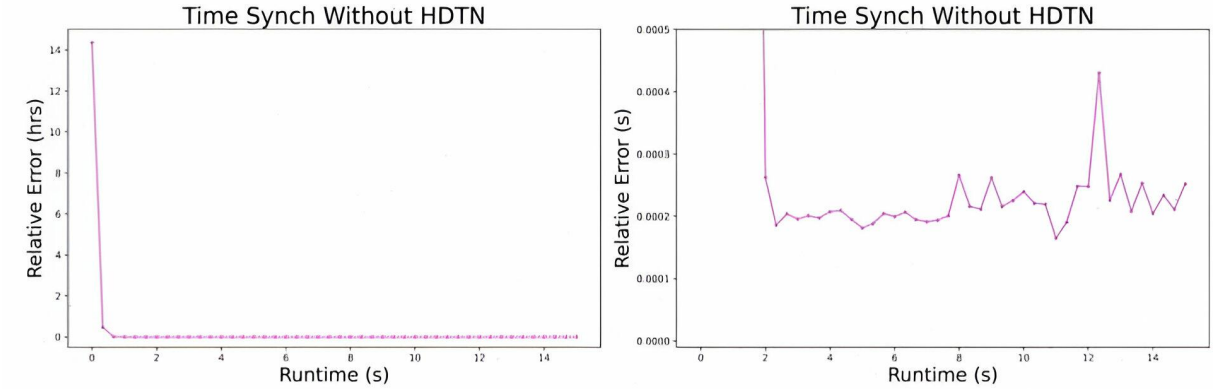


Figure 12: Experiment 1 – Plot of relative errors when performing time sync without HDTN

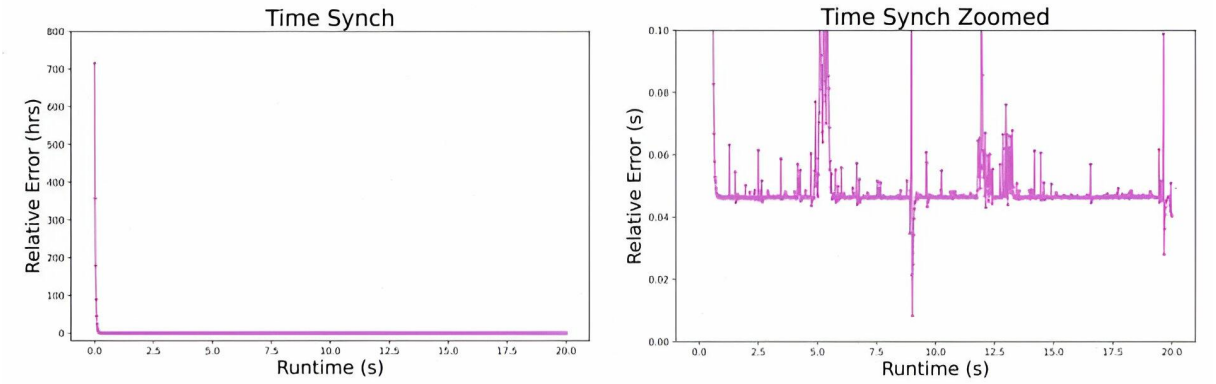


Figure 13: Experiment 2 – Plot of relative errors when performing time sync using TCP

Experiment 3

In the third and final experiment, we performed time synchronization across a network (shown in Figure 14) of several nodes which included a node on board an airplane. The airplane node served as the end user of a hypothetical time synchronization service.

The results seen in Figure 15 show that despite a high initial max error of around 1300s, the network was able to synchronize to roughly within 10ms. In fact, at around 6.8 seconds in, the network experienced a large disruption, but the synchronization was able to recover. This experiment demonstrates that errors are able to diffuse through a network with a more complicated topology.

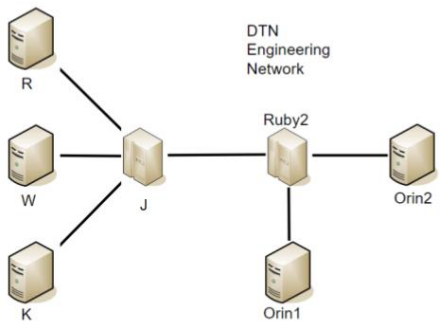


Figure 14: Network used for Experiment 3 showing node connections

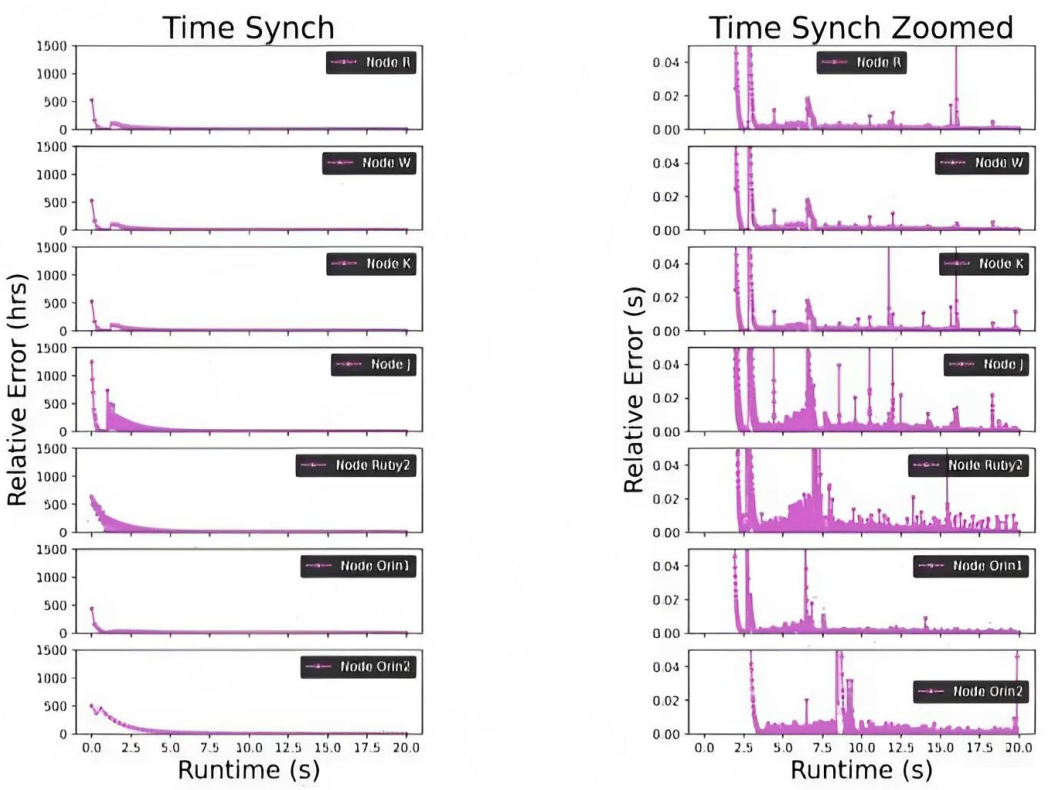


Figure 15: Experiment 3 – Plot of relative errors in the flight test

6. Conclusion

In this paper, we discuss the path taken to realize the algorithm presented in [13] in code and tests conducted using NASA's LCRD as a test bed. Special attention was paid to non-trivial network architectures to demonstrate the real-world practicality, with eyes on LunaNet and the future Solar System Internet. The prototype code is made available [26] and can be used as a starting point for future implementations. The experiments conducted show great promise, and future tests using LCRD are planned to demonstrate the prototype in a space environment. For now, we conclude with suggestions for future research:

Future Works

Integrating the Protocol into SSI/DTN Architecture

It is unclear where in the stack (ISO model) time synchronization should belong in DTN as different layers will have different error characteristics. Greater sensitivity analyses and profiling should be conducted to better understand the ramifications. It might be the case that time synchronization belongs at several layers, depending on use-case. Coarse and fine requirements might also lead to a cross-layer approach. From an implementation standpoint, time synchronization over DTN could use bundle payloads or bundle extension blocks. It is unclear if there is a clear choice. This should be considered carefully, as there are pros and cons of each approach.

Similar questions can be asked in the context of software-defined networking (SDN). SDN essentially is the network architecture where the control planes of the network layer are centralized and abstracted into a single controller running software that makes it serve as the controller for every node. SDN is a promising building block for constructing (time-synchronized) scalable DTN thanks to its centralization; it intuitively is the most canonical network architecture for contact-based routing as the schedule has to be distributed somehow. Many SDN-based DTN architectures or integration of SDN into DTN [40, 42–46] were proposed over the past few years, yet none of them appears to subsume the other(s), especially on the method of handling the disruptions within control channel. For example, [42] uses a Paxos-ish consensus mechanism where the nodes elect the controller, [44] forms a distributed system of controllers where depending on the network status the network (either controllers or the switch) decides which controller talks to the switch, and [40] partitions the network into clusters where each consists of SDN nodes and a controller. However, it is worth noting that their differences are mostly inherited from the differences in their DTN architectures or implementations; to give some examples, in [46] convergence layer adapters (CLAs) are assigned to manage the persistent storage, which is usually the responsibility of the bundle protocol agents (BPAs) in DTN, and [44] models the routing protocol using Markov chain to minimize the storage (for contact plan) and time (for re-calculating the path). Hence, while they may not be compatible directly, one can complement the other, eventually moving towards standardization. However, the entity, process, interface, and/or layer within the SDN that should be responsible for the DTN-level clock synchronization still remains fuzzy.

Security and Policy

The current design of Bundle protocol (BP) and time-variant routing makes the security of DTN inherently rely on timing

profiles and timestamps. Such dependency on timing shows an inherent security risk: one way to effectively disable a node is to give a sufficiently bogus time [47–49]. For instance, different domains (e.g., different space network service providers) may or may not be willing to have their clocks adjusted due to some reasons such as performance issues, and may or may not be willing to take the responsibility of serving as clock references. That is, it is possible that there will be islands of synchronization. The implications of which on network management/administration are unknown; bundle security will also change the timing profiles, and hence, security, policies, recovery mechanisms, and so forth need to be researched.

To elaborate, NTP used today is cryptographically safeguarded with its security extension called Network Time Security (NTS) [50]. NTS ensures the authenticity of NTP packets in a way that the clients can verify that the NTP server they are communicating with is a legitimate server and that the times they are receiving are actually from them and were not tampered with in the middle. However, in a protocol that is rather decentralized with no or only a few actual, dedicated time servers, we might need something more than that: a malicious node can intentionally broadcast the incorrect clock value to, for example, keep all expired bundles as unexpired and/or drop unexpired ones as expired, then use them to flood the network and exhaust persistent storages (denial-of-service type attacks) or authenticate themselves as a legitimate client using expired session keys/nonce (replay attack type attacks) [11, 47–49, 51, 52]. It would be desirable to have a method of generating an unforgeable, easy-to-verify proof showing that the delivered clock values are indeed the values shown and taken directly from the sender's clock without any (unauthorized) post-processing.

Network Congestion

It is generally considered that NTP is not 'chatty,' and our experiments suggest that our protocol is likely not going to be that 'chatty' as well. Still, in the event of network congestion or thrashing (or in the event when they are imminent), pursuing clock synchronization may reversely hinder the performance of the network because it adds additional burden to the network. Compression techniques (e.g., network coding) for our protocol that allow the network to deliver multiple clock values at once can be practical in such scenarios. In addition, noise-processing methods using Kalman filters were shown to be effective in improving various clock synchronization protocols, as they can be used to reduce the effects of common sources of errors such as clock drifts and data rate asymmetry [53–55]. An interesting line of future work would be to see if they can supplement our protocol as well, and reduce the number of total synchronization requests in the network by a conceivable scale.

Relativistic Effects

While relativistic effects may not be significant enough to cause immediate deployment issues in practice, an ability to compensate clock errors caused by relativistic effects such as time dilations and Sagnac interference has been one of the functional requirements for NASA's solar system-scale time distribution protocols [15, 56–59]. Relativistic effects on time are certainly prevalent today (many references exist, with some recent ones being [60–64]), and it will only get aggravated as the scale of the network (and hence our protocol) gets larger. An upcoming need is LunaNet, as pointed out in a White House Briefing [18].

Towards Deployment and Standardization

The experiments conducted and reported in this paper are smaller than what we hope Solar System Internet to be. An example of an Earth network and a Martian network using this protocol should be constructed to uncover subtleties. Currently, neither a Request for Comments (RFC) nor an actual standardized protocol exists. To become a formal protocol, these need to be developed along with the underlying algorithm. This is a crucial next step.

ACKNOWLEDGEMENTS

The authors sincerely thank Dr. Michael Moy for enlightening them by providing numerous valuable inputs which were immensely helpful during the initial stage of this work and simulations, and Mark Sinkiat for his essential assistance in setting up our experiment. The authors are also grateful to the NASA Space Communications and Navigation Program (SCaN) for their generous funding and support (for all authors), and Washington NASA Space Grant Consortium (WASG) Scholarship that greatly benefited one of the authors (K. Petwe).

REFERENCES

[1] United Nations Office for Outer Space Affairs (UNOOSA) - with major processing by Our World in Data, "Annual number of objects launched into space," Jan 4th, 2024, (Original data: "Online Index of Objects Launched into Outer Space" by UNOOSA). [Online]. Available: https://ourworldindata.org/grapher/yearly-number-of-objects-launched-into-outer-space

[2] A. Vasilakos, Y. Zhang, and T. Spyropoulos, *Delay Tolerant Networks: Protocols and Applications*. Taylor & Francis Group, Boca Raton, FL, USA: CRC Press, 2016.

[3] M. J. Demmer, "A Delay Tolerant Networking and System Architecture for Developing Regions," Ph.D. dissertation, Department of Electrical Engineering and Computer Sciences, University of California Berkeley, 09 2008, UCB/EECS-2008-124.

[4] F. Warthman, "Delay and Disruption-Tolerant Networks (DTNs): A Tutorial," Warthman Associates, Tech. Rep., 09 2015. [Online]. Available: https://www.nasa.gov/wp-content/uploads/2023/09/dtn-tutorial-v3.2-0.pdf

[5] 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, 2021.

[6] 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*, 03 2023, pp. 1–20.

[7] A. Hylton, B. Mallery, J. Hwang, M. Ronnenberg, M. Lopez, O. Chiriac, S. Gopalakrishnan, and T. Rask, "Multi-Domain Routing in Delay Tolerant Networks," in *2024 IEEE Aerospace Conference*, 03 2024, pp. 1–20.

[8] O. Mukhtar, "Design and Implementation of Bundle Protocol Stack for Delay-Tolerant Networking," Mas-

ter's thesis, Department of Electrical and Communications Engineering, Aalto University, 08 2006.

[9] I. Cardei, C. Liu, J. Wu, and Q. Yuan, "DTN routing with probabilistic trajectory prediction," in *Wireless Algorithms, Systems, and Applications (WASA 2008)*. Dallas, TX, USA: Springer, 10 2008, pp. 40–51.

[10] W. Ivancic, W. M. Eddy, D. Stewart, L. Wood, J. Northam, and C. Jackson, "Experience with Delay-Tolerant Networking from Orbit," *International Journal of Satellite Communications and Networking*, vol. 28, no. 5-6, pp. 335–351, 2010.

[11] J. A. Fraire, J. M. Finochietto, and S. C. Burleigh, *Delay Tolerant Satellite Networks*. Artech House, 2017.

[12] W. Shi, D. Gao, H. Zhou, B. Feng, H. Li, G. Li, and W. Quan, "Distributed Contact Plan Design for Multi-Layer Satellite-Terrestrial Network," *China Communications*, vol. 15, no. 1, pp. 23–34, 2018.

[13] M. Moy, A. Hylton, R. Kassouf-Short, J. Cleveland, J. Hwang, J. Curry, M. Ronnenberg, M. Lopez, and O. Chiriac, "A Proposed Clock Synchronization Method for the Solar System Internet," in *2024 IEEE Aerospace Conference*, 03 2024, pp. 1–17.

[14] NASA, "High-Rate Delay Tolerant Networking (HDTN)." [Online]. Available: https://www1.grc.nasa.gov/space/scan/acs/tech-studies/dtn/

[15] 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*, 03 2020, pp. 1–14.

[16] A. Hylton, D. Israel, M. Wennersten, R. Chaoua, M. Palsson, M. Sinkiat, G. Menke, D. Raible, R. Dudukovich, N. Kortas, B. Tomko, E. Schweinsberg, P. Choksi, J. Lombay-Gonzalez, T. Basciano, B. Pohlchuck, J. Deaton, J. Sager, M. Kaushik, and T. Kollar, "The International Space Station, Optical Communications, and Delay Tolerant Networking: Towards a Solar System Internet Architecture," *41st International Communications Satellite Systems Conference (ICSSC 2024)*, 2024.

[17] N. Ashby and B. R. Patla, "A Relativistic Framework to Estimate Clock Rates on the Moon," *The Astronomical Journal*, vol. 168, no. 3, p. 112, 2024.

[18] A. Prabhakar, "Policy on Celestial Time Standardization in Support of the National Cislunar Science and Technology (S&T) Strategy," in *Memorandum for Departments and Agencies Participating in the White House Cislunar Technology Strategy Interagency Working Group*, Office of Science and Technology Policy, Executive Office of the President. The White House, 2024. [Online]. Available: https://www.whitehouse.gov/wp-content/uploads/2024/04/Celestial-Time-Standardization-Policy.pdf

[19] A. Hylton, R. Short, R. Green, and M. Toksoz-Exley, "A Mathematical Analysis of an Example Delay Tolerant Network using the Theory of Sheaves," in *2020 IEEE Aerospace Conference*, 03 2020, pp. 1–11.

[20] R. Short, A. Hylton, R. Cardona, R. Green, G. Bainbridge, M. Moy, and J. Cleveland, "Towards Sheaf Theoretic Analyses for Delay Tolerant Networking," in *2021 IEEE Aerospace Conference*, 03 2021, pp. 1–9.

[21] R. Short, A. Hylton, J. Cleveland, M. Moy, R. Cardona, R. Green, J. Curry, B. Mallery, G. Bainbridge, and Z. Memon, "Sheaf Theoretic Models for Routing in Delay Tolerant Networks," in *2022 IEEE Aerospace Conference*, 03 2022, pp. 1–19.

[22] A. Hylton, J. Cleveland, R. Dudukovich, D. Iannicca, N. Kortas, B. LaFuente, J. Nowakowski, D. Raible, R. Short, B. Tomko, and A. Wroblewski, "New Horizons for a Practical and Performance-Optimized Solar System Internet," in *2022 IEEE Aerospace Conference*, 03 2022.

[23] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of Distributed Consensus with One Faulty Process," *Journal of the ACM (JACM)*, vol. 32, no. 2, pp. 374–382, 1985.

[24] 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, 2019.

[25] ——, "Opinion Dynamics on Discourse Sheaves," *SIAM Journal on Applied Mathematics*, vol. 81, no. 5, pp. 2033–2060, 2021.

[26] J. Cleveland and K. Petwe, GitHub Repository. [Online]. Available: https://github.com/jacleveland/timeSync

[27] A. Greenbaum and T. Chartier, *Numerical Methods: Design, Analysis, and Computer Implementation of Algorithms*. Princeton University Press, 2012.

[28] U. M. Ascher and C. Greif, *A First Course in Numerical Methods*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2011.

[29] J. N. Kutz, *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. Oxford University Press, Inc., 2013.

[30] MathWorks, *ode23: Solve nonstiff differential equations – low order method*, MATLAB Documentation. [Online]. Available: https://www.mathworks.com/help/matlab/ref/ode23.html

[31] ——, *ode45: Solve nonstiff differential equations – medium order method*, MATLAB Documentation. [Online]. Available: https://www.mathworks.com/help/matlab/ref/ode45.html

[32] J. Crank and P. Nicolson, "A Practical Method for Numerical Evaluation of Solutions of Partial Differential Equations of the Heat-Conduction Type," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 43, no. 1, p. 50–67, 1947.

[33] J. W. Thomas, *Numerical Partial Differential Equations: Finite Difference Methods*. New York, NY: Springer, 1995.

[34] C. Lee and D. J. Wilkinson, "A review of stochastic block models and extensions for graph clustering," *Applied Network Science*, 2019.

[35] E. Mossel, J. Neeman, and A. Sly, "Reconstruction and Estimation in the Planted Partition Model," *Probability Theory and Related Fields*, vol. 162, 08 2015.

[36] Y. Kirkpatrick, R. Dudukovich, P. Choksi, and D. Ta, "Cooperative Clustering Techniques For Space Network Scalability," in *2023 IEEE Cognitive Communications for Aerospace Applications Workshop (CCAAW)*, 2023, pp. 1–8.

[37] S. Booth, R. Dudukovich, N. Kortas, E. Schweinsberg, B. Tomko, B. LaFuente, E. Danish, T. Recker, and P. Choksi, "High-Rate Delay Tolerant Networking

(HDTN) User Guide," NASA, Glenn Research Center, Cleveland OH 44135, USA, Technical Memorandum NASA/TM-20230000826/REV1, 05 2024.

[38] 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)*. IET, 2019, pp. 1–17.

[39] D. Raible, "HDTN Overview for IEEE Workshop on Optimizing Interplanetary Communication Through Network Autonomy," 08 2024. [Online]. Available: https://ntrs.nasa.gov/citations/20240010738

[40] D. Ta, S. Booth, and R. Dudukovich, "Towards Software-Defined Delay Tolerant Networks," *Network*, vol. 3, no. 1, pp. 15–38, 2022.

[41] D. J. Israel, B. L. Edwards, R. L. Butler, J. D. Moores, S. Piazzolla, N. du Toit, and L. Braatz, "Early Results from NASA's Laser Communications Relay Demonstration (LCRD) Experiment Program," in *Free-Space Laser Communications XXXV*, H. Hemmati and B. S. Robinson, Eds., vol. 12413, International Society for Optics and Photonics. SPIE, 2023.

[42] I. Zacarias, L. P. Gaspary, A. Kohl, R. Q. Fernandes, J. M. Stocchero, and E. P. de Freitas, "Combining Software-Defined and Delay-Tolerant Approaches in Last-mile Tactical Edge Networking," *IEEE Communications Magazine*, vol. 55, no. 10, pp. 22–29, 2017.

[43] T. Yang, L. Kong, N. Zhao, and R. Sun, "Efficient Energy and Delay Tradeoff for Vessel Communications in SDN based Maritime Wireless Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3800–3812, 2021.

[44] S. Babu, "Software Defined Disruption Tolerant Networks," Ph.D. dissertation, Indian Institute of Space Science and Technology, 2021.

[45] S. Booth, A. Hylton, R. Dudukovich, N. Kortas, B. LaFuente, and B. Tomko, "Alleviating Bundle Throughput Constriction for Delay Tolerance Networking (DTN) Bundles with Software-Defined Networking (SDN)," in *The Fourteenth International Conference on Advances in Satellite and Space Communications (SPACOMM 2022)*, 2022.

[46] F. Walter, M. Feldmann, J. A. Fraire, and S. Burleigh, "The Architectural Refinement of µD3TN: Toward a Software-Defined DTN Protocol Stack," in *Space-Terrestrial Internetworking Workshop (STINT 2024) at IEEE International Conference on Space Mission Challenges for Information Technology / Space Computing Conference (SMC-IT/SCC)*, 2024.

[47] 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.

[48] E. J. Birrane, N. Kuhn, Y. Qu, R. Taylor, and L. Zhang, "RFC 9657: Time-Variant Routing (TVR) Use Cases," 10 2024.

[49] D. King, L. M. Contreras, B. Sipos, and L. Zhang, "TVR (Time-Variant Routing) Requirements," IETF Internet-Draft: draft-ietf-tvr-requirements-04, 09 2024. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-tvr-requirements-04

[50] D. F. Franke, D. Sibold, K. Teichel, M. Dansarie, and R. Sundblad, "RFC 8915: Network Time Security for the Network Time Protocol," 09 2020.

[51] G. O. Ansa, "Mitigating Denial of Service (DoS) Attacks in Delay/Disruption Tolerant Networks (DTNs)," Ph.D. dissertation, University of Surrey, 10 2012.

[52] 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.

[53] J. Cano, S. Chidami, and J. Le Ny, "A Kalman Filter-based Algorithm for Simultaneous Time Synchronization and Localization in UWB Networks," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 1431–1437.

[54] B. R. Hamilton, X. Ma, Q. Zhao, and J. Xu, "ACES: Adaptive Clock Estimation and Synchronization Using Kalman Filtering," in *Proceedings of the 14th ACM international conference on Mobile computing and networking*, 2008, pp. 152–162.

[55] A. Bletsas, "Evaluation of Kalman Filtering for Network Time Keeping," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 52, no. 9, pp. 1452–1460, 2005.

[56] J. J. Miller, A. Gifford, B. Brodsky, A. Oria, R. A. Nelson, R. S. Orr, L. Felton, L. Pitts, F. VanLandingham, and B. Welch, "NASA Architecture for Solar System Time Distribution," in *2007 IEEE International Frequency Control Symposium Joint with the 21st European Frequency and Time Forum*, 2007, pp. 1299–1303.

[57] L. Felton, L. Pitts, and F. VanLandingham, "NASA Architecture for Solar System Time Synchronization and Dissemination: Concept of Operations," in *SpaceOps 2008 Conference*, 2008.

[58] L. Wood, W. M. Eddy, and P. Holliday, "A Bundle of Problems," in *2009 IEEE Aerospace conference*, 2009, pp. 1–17.

[59] D. E. Raible and A. G. Hylton, "Integrated RF/Optical Interplanetary Networking Preliminary Explorations and Empirical Results," in *30th AIAA International Communications Satellite System Conference (ICSSC)*, 2012.

[60] R. Angélil, P. Saha, R. Bondarescu, P. Jetzer, A. Schärer, and A. Lundgren, "Spacecraft Clocks and Relativity: Prospects for Future Satellite Missions," *Physical Review D*, vol. 89, no. 6, p. 064067, 2014.

[61] J. Wang, Z. Tian, J. Jing, and H. Fan, "Influence of relativistic effects on satellite-based clock synchronization," *Physical Review D*, vol. 93, no. 6, p. 065008, 2016.

[62] D. G. Messerschmitt, "Relativistic Timekeeping, Motion, and Gravity in Distributed Systems," *Proceedings of the IEEE*, vol. 105, no. 8, pp. 1511–1573, 2017.

[63] J. Kouba, "Relativity Effects of Galileo Passive Hydrogen Maser Satellite Clocks," *GPS Solutions*, vol. 23, no. 4, p. 117, 2019.

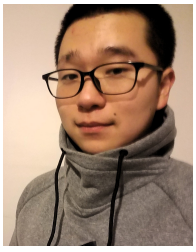
[64] T. Ely, J. Prestage, R. Tjoelker, E. Burt, A. Dorsey, D. Enzer, R. Herrera, D. Kuang, D. Murphy, D. Robison *et al.*, "Deep Space Atomic Clock Technology Demonstration Mission Results," in *2022 IEEE Aerospace Conference*, 2022, pp. 1–20.

BIOGRAPHY

Alan Hylton *would rather work in tube audio, but instead directs Delay Tolerant Networking (DTN) research at NASA GSFC. After studying mathematics at Cleveland State University and Lehigh University, he now advocates for his students and is humbled to work with his powerful and multidisciplinary team by creating venues for mathematicians to work on applied problems.*

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 interested in applying differential geometry and topology to the realms of physics and artificial intelligence. Oliver devotes his free time to soccer, music, and travel.*

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 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, computer networks, and information security. Before Purdue, he studied mathematics and computer science at the University of Massachusetts Amherst.*

Karuna Petwe *is a fourth year undergraduate student majoring in applied and computational math sciences at the University of Washington. She is broadly interested in understanding the role that mathematical structures and models play in solving current issues in computer and computational sciences. She is excited to participate in ongoing investigations into vital networking capabilities for Delay Tolerant Networks.*

Tobias Timofeyev *is a 4th year Ph.D. student in applied mathematics at the University of Vermont (UVM). His interests are varied, but he particularly enjoys to work at the intersection of network science and dynamical systems theory. He finds it often important to ask how the structure of a system and its function interact. Prior to UVM, he received his B.A. in Mathematics from Bard College. In his free time, he enjoys practicing the cello, and playing boardgames with friends.*

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.*