Towards Sheaf Theoretic Analyses for Delay Tolerant Networking

Robert Short and Alan Hylton
NASA Glenn Research Center

Robert Cardona and Robert Green
University at Albany - State University of New York

Gabriel Bainbridge
The Ohio State University

Michael Moy
Colorado State University

Jacob Cleveland
University of Nebraska at Omaha

The goal of Delay Tolerant Networking (DTN) is to take a collection of heterogeneous, disparate connections between satellites, space assets, ground stations, and ground infrastructure and bring it together into a cohesive, functioning overlay network. Depending on the systems being considered, one can find links with a one-way light time exceeding minutes (and hours), periodic links which can sometimes be predicted by orbital mechanics, and restrictions based on the variety of capabilities built into these systems. These characteristics preclude traditional network models and routing techniques and have classically led to either rigid routing tables or purely probabilistic models. As the deeper underlying structures remain unknown, development of more DTN-optimized algorithms has lacked the necessary foundation. In a continuation of previous work, the goal of this paper is to identify and study these fundamental structures that exist in delay tolerant networks (DTN), with a focus on space networks.

The current routing methodology has been to use contact graph routing (CGR) algorithms. CGR models a series of known contacts as a static graph. For CGR to work, this graph must be globally consistent and must have an accurate picture of the network. Because this is a globally controlled structure, there is little room for flexibility in the event of changes to the network which would naturally occur as the network grows. As a response to the desire for flexibility as the network changes, we introduced the mathematical structure known as sheaves to DTNs last year. The tag-line for sheaves is that they are a mathematically precise way of gluing local data together into unique global data. Thus, sheaves lend extra power to traditional models (and routing algorithms) by taking additional information and merging it, in as consistent a manner as possible, with the representation itself.

The clearest example of how Earth-bound networks exhibit behavior that is "sheafy" is link state routers, which build a local-to-global picture of their network by gluing local information together into a global network, exactly as a sheaf would do. For routing within delay tolerant networks to truly exploit this structure, a deeper structure than a graph is required. In this paper, we develop sheaves that can work over directed graphs such as temporal flow networks, we construct a sheaf representation for Dijkstra's algorithm, and we outline a construction for routing sheaves capable of modeling multicast scenarios. Finally, there is a section of future work suggesting follow-on research.

I. Introduction

Terrestrial telecommunication networks as we known them provide many capabilities seen as necessary by today's standards, including returns to scale with the number of participants, standardization so most devices can participate,

and security to protect those involved (and more). The need for similar capability for the so-called Solar System Internet (SSI) have become evident, as increases are seen in the number of nodes, the link capacities involved, and the number of countries involved [1].

The NASA approach uses Delay Tolerant Networking, which is defined in RFC 4838 [2]. While introductions to DTN are many - see [3][4][5] - this paper is not intended to serve as a general introduction; rather we will focus on the components most pertinent to our modeling goals. There have been several papers that work towards developing models for DTNs, and are often influenced by approaches taken in traditional networking. We will provide a brief overview of DTN, and return to some of the existing literature. The thing to keep in mind is that despite strong, useful efforts in this arena, DTN remains to be placed upon firm footing, which we believe to be necessary for DTN to achieve its long-term goals.

One basic assumption of typical networking is the persistent existence of end-to-end paths between any given source and any given destination. Hence from a mathematical perspective, the building-block to model terrestrial connectivity is a connected graph. Then various routing (path-finding) algorithms are employed to find paths between different sources to enable data flow. This assumption is, for example, key to Open Shortest Path First (OSPF) routing. However, such an assumption cannot be reasonably made in mobile networks, and in particular, space networks. Additionally, terrestrial networks are necessarily "small" in a distance sense - the cables, fibers, and wireless channels are bound to the Earth, allowing another basic assumption: the latency is low - typical protocols might cease working if round trip times (RTT) exceed a few seconds. The average distance from the Earth to Mars is 14 minutes, as light travels in a vacuum. On the other end of the scale, the International Space Station (ISS) is less than 2 milliseconds to the Earth. The ISS typically communicates "upwards," towards relay satellites in geosynchronous orbit, which is roughly 120 milliseconds to the Earth. Depending on a given link, one may or may not be able to have a feedback loop in their communication system. These straightforward observations illustrate that not only do key assumptions of terrestrial networking not hold in space, but space networks tend to be very heterogeneous [4].

Another example of heterogeneity comes from mobility. Indeed, Earth assets are mostly stationary (at least from the perspective of the network). This enables strict, hierarchical addressing schemes, such as Internet Protocol (IP) addressing, which allows the division of networks into sub-networks, thereby enabling routing (and hence scaling). There is currently no standardized approach towards mobile addressing for the SSI.

To overcome the rather vast generalizations, a DTN will use the basic actions of storing, carrying, and forwarding data as appropriate. In other words, a DTN functions as a type of abstract architecture: it describes a pattern of implementing networks that address challenges inherent to operating in such environments [2].

Returning to modeling, examples include [6], which develops queuing delay estimates for a particular star topology. In [7], another queuing approach is taken for simple opportunistic networks, particularly to model end-to-end delay. This paper assumes the typical Poisson process for data arrival. There are also others: [8] [9]. A common thread is smaller, manageable networks. While it is true that early and contemporary space networks comprise relatively few nodes, it is not difficult to see that with ride sharing programs and modern commercial launch services, the number of communicating assets in space will increase at unprecedented levels. There are also competing ideas to the NASA definition, given in RFC 4838 [2]. This includes, but is not limited to, Information Centric Networking (ICN), which tries to move the model from host-centric to, as the name implies, information-centric [10].

The common ground underneath all of the methods above is repurposing existing and traditional techniques and theories to the more general problem. While this has been very fruitful, it suffers limitations, not the least of which is limited complexity of any system analyzed. Considering past successes, we have begun looking for ways to work from the bottom up, rather than the top down, by developing the underlying foundations upon with DTN rests. This paper then builds upon the introductory paper [11], which suggested the various mathematical theories that naturally describe DTN phenomena.

II. Graphs for Delay-Tolerant Networking

Before we consider the mathematical formalisms, we will motivate them with a brief discussion on DTN routing.

If a given network is small enough, it can be described by a *contact graph*, which is a globally distributed and consistent contact plan, that includes information such as when a contact between any two nodes will be established, for how long, at what average bit rate, and what the average distance is in seconds. Using this information, a traditional DTN routing algorithm is Contact Graph Routing [12]. By basing the graph structure on the earliest possible time of arrival, CGR was optimized (the so-called enhanced CGR), which prevents routing loops, oscillation, and perhaps most importantly enables the usage of Dijkstra's algorithm for path-finding (routing)[13].

If changes to the network must be made known to DTN, the CGR table must be updated globally - such a propagation might not cover the whole network before additional changes are made. Also, if there are "many" nodes, at some point the table will become unwieldy. This comes from the desire to capture as much information as possible about a system that is expected to increase in scale. Another part of this appears to be a tacit expectation of global synchronization. If these global data are consistent, then local forwarding decisions can be made according to Dijkstra's algorithm.

These assumptions for CGR do make sense in certain situations. These include smaller networks, but also networks with either very predictable links - the latter of which might feature enough distance to preclude discovery or other real-time controls. However, for mobile ad hoc networks, additional mechanisms would be needed to add discovery and insertion into the tables.

Other traditional routing algorithms based on Dijkstra's algorithm include link state routers, which operate on global knowledge of the network - the information includes the topology as well as cost functions to transmit data over a given link. To learn the topology, an LSR will learn about its neighbors; this involves link status, capacity, IDs, and so forth. This information is put into a Link State Packet (LSP). Periodically, Link State Updates, comprised of LSPs, are flooded across the network, meaning they are sent to all routers say they may build local databases about the network topology. While a traditional LSR would not allow a route to be computed when there is end-to-end connectivity, link state routers can be generalized for DTNs. One such approach takes advantage of the history of links to build probablities of a link re-appearing, and allows Link State Announcements (LSAs) to persist even when the temporal cross-section gives a graph with the endpoints in different connected components [14].

Another approach to DTN routing is the Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET) [15]. PRoPHET calculates delivery predictability based on history, near (or exact) periodicity of links, and network transitivity - this is the probability that if a node x is connected to y which is in turn connected to node z, that x is connected to z. The delivery predictabilities are calculated and updated between a node and all other nodes that it is aware of, based on transitivity. As nodes meet, these tables are exchanged. This way, with some confidence, a global picture of the network will emerge as it interacts with itself. This algorithm has been tweaked and updated over time [16].

The first step towards finding candidate structures to model DTN involves making sure they capture they time-variance, mobility, and other features described. The second step is ensuring that these structures support that routing action that we have seen so far - we are always taking local information, patching it together into global information, and using this picture to make decisions on whether or not to forward, store, or drop data, and if to forward, how to do it.

Graphs have enjoyed a great deal of success in networks. It turns out that graphs can also be generalized naturally to include time variance, as well as other types of structure. It is worth noting that from the network perspective, changes cannot happen *too* rapidly, or otherwise communications cannot occur at all. This could happen, for example, if changes occurred quicker than a bundle could be transmitted (for the expert, we ignore fragmentation for the time being). Therefore, it is fair to begin with discretized time-varying graphs.

In [11], Temporal Flow Networks were considered; these are defined in [17]. We will recall the definition below, as well as the necessary building blocks, starting with temporal graphs. In the following, given a graph G, we let V be the set of vertices and E be its set of (possibly directed) edges.

Definition II.1 (Temporal Graph)**.** We call a graph $G = (V, E)$ a temporal graph if:

- G is a directed graph; and
- for each edge $e \in E$, we assign a finite set $L_e \subset \mathbb{N}$ of labels identifying the integer time instances when e is available.

This is denoted $G = (V, E, L)$ where $L = \bigsqcup_{e \in E} L_e$ to summarize the necessary information.

Once we have established a temporal graph, we can extend it to model flow with additional structure. First, we recall that as in a contact graph, link durations and average bit rates are known, and hence link capacities are known. This is taken to the next level by equipping individual nodes with known storage (buffering) limitations.

Definition II.2 (Temporal Flow Network)**.** A temporal flow network G is a temporal graph equipped with:

- an identified source vertex v_S and sink (target) vertex v_T;
- a capacity function $c : E \to \mathbb{R}$; and
- a buffer function $B : V \to \mathbb{R}$, where $B(v_S)$ and $B(v_T)$ are assumed to be infinite.

This is denoted $G = (V, E, L, v_S, v_T, c, B)$ to summarize the necessary information.

One can think of a temporal flow network as a sequence of graphs at each time instance connected by edges mapping each vertex to its copy in the next time instance with buffer capacities representing the storage capacities over the edges between time instances.

Definition II.3 (Time-Extended Graph)**.** Let $G = (V, E, L, v_S, v_T, c, B)$ be a temporal flow network. Then its corresponding time-extended graph $G^{TE} = (V^{TE}, E^{TE})$ is a weighted directed graph where:

- for each vertex $v \in V$, there are $\ell^{TE} + 1$ copies in V^{TE} denoted $v_0, v_1, \ldots, v_{\ell^{TE}}$ where ℓ^{TE} is the largest integer in $\bigcup_{e \in E} L_e$
- for each edge $e = (v, w) \in E$ and each $i \in L_e$, there is an edge $e^{TE}_{vw,i} = (v_{i-1}, w_i) \in E^{TE}$
- for each vertex $v \in V$, there is an edge $e^{TE}_{v,i} = (v_{i-1}, v_i) \in E^{TE}$
- there is a capacity function $c^{TE} : E^{TE} \to \mathbb{R}$ defined by:

$$c^{TE}(e^{TE}_{x,i}) = \begin{cases} c(e) & \text{if } x \in E \\ B(v) & \text{if } x \in V \end{cases}$$

As alluded to above, these structures can formally capture the information required by CGR, but do not have to. They could also be generalized in natural ways to include more types of information. Parameters may be known, unknown, or perhaps even ignored; indeed, right now there are no algorithms present. However, we aim to change that.

These graphs and their temporal generalizations give us a notion of connectivity of a network. This includes time-variance, which makes not only links a function of time, but also the number of connected components (think Euler characteristic). The means we can consider various functions over the entire graph, or a time-slice of it, or more elementarily functions local to a node, which will be a way to consider routing. Algorithms that could build global functions/ideas from local ones actually sound familiar at this point. In fact, this local-to-global action is the common theme in all routing approaches discussed. This motivates the introduction of the mathematical structure known as *sheaves*, whose tagline is that they are the mathematically precise way of gluing local data together into unique, global data. In the light of routing, we see that networking is "sheafy."

It turns out that sheaves are very geometric objects, and in fact just about everything considered geometric in mathematics can be defined in terms of sheaves [18]. Despite their general success, sheaves are often either unfamiliar or foreboding. Fortunately, when applied to graphs and graph-like structures, they become significantly more straightforward. More accurately, we work will cellular sheaves, which are sheaves defined over cellular complexes; see [19] [20] [21].

In the following sections, we will consider how sheaves applied to graphs can yield familiar results, while simultaneously offering generalizations, making them more broadly applicable. While a price is to be paid to gain generality, we are fortunate that the results remain algorithmic in nature, and hence retain practicality and implementability.

III. Sheaves as Networking Models

This idea of moving from local information to painting a global picture occurs all over the place in networking. As a basic example, we return to how link state routing works. Loosely, each link state router knows its local connectivity and is able to share that information with the rest of the network. That connectivity information is sufficient to generate a picture of the entire network if we glue it together appropriately. Locally, the information is insufficient, but once glued together, we have a complete and consistent view of the network.

To model this, what do we need? We need to know the physical connections that allow nodes to share information with each other. Then, we need to represent the data being shared over the connection. Each individual representation of the data is considered local information, so to make the information global, we need a means of relating local information together. Prescribing some relationship between the local data, and a means of ensuring the information is consistent, is the key to turning local data into global data. All of this can be done using a sheaf over a graph; after going over some definitions, examples will be given.

Definition III.1 (Sheaf over a Graph)**.** Let $G = (V, E)$ be a graph. We say $\mathcal{F}$ is a (set-valued) sheaf over G if:

- For each vertex $v \in V$ and edge $e \in E$, $\mathcal{F}$ assigns a set, represented as $\mathcal{F}(v)$ and $\mathcal{F}(e)$ respectively.
- If v is incident to e, then there is a function $\mathcal{F}(v \rightsquigarrow e) : \mathcal{F}(v) \to \mathcal{F}(e)$ (called a restriction map).

The sets $\mathcal{F}(v)$ represent the possible data over each vertex (or $\mathcal{F}(e)$ over each edge respectively), but what matters is how each individual piece of data relates through the restriction maps. This is given by sections of a sheaf in the following way:

Definition III.2 (Sections of a Sheaf)**.** Given a sheaf $\mathcal{F}$ over a graph $G = (V, E)$, a (global) section s of $\mathcal{F}$ is a choice of elements in each set $\mathcal{F}(v)$ and $\mathcal{F}(e)$ for every $v \in V$ and $e \in E$ such that if v is incident to e, $\mathcal{F}(v \rightsquigarrow e)(s(v)) = s(e)$.

For any $H \subset V \cup E$, we say a local section s (over H of $\mathcal{F}$) is a choice of elements in each $\mathcal{F}(v)$ and $\mathcal{F}(e)$ where $v, e \in H$ such that if v is incident to e, $\mathcal{F}(v \rightsquigarrow e)(s(v)) = s(e)$.

That is to say, a section represents data that respects the restriction maps consistently across the network. Local sections play an important role in moving from local to global within a sheaf. Restriction maps get their name because as local sections involve more and more restriction maps, the number of possible local sections decreases. Local sections allow us to expand our graph, either evolving through time or considering data transmissions across a network, while respecting the sheaf structure.

A good way to think about sheaves is as a way of organizing and relating data moving across a network, such as in [11]. For a sheaf $\mathcal{F}$ over a graph G, the sets over the vertices can represent what information can be stored in each node while the edges can represent what information can be transmitted over each edge. Then, restriction maps describe the relationship between data transmitted over edges and the data stored in the vertices. Building on that, a section prescribes information to each vertex and edge in the graph that is consistent with the relationships we expect to see in the data. The real power of sheaves comes from defining restriction maps well so that the sections provide actual solutions to different problems.

Sheaves have also appeared in [22] as a means of modeling wireless routing protocols over networks. The basic sheaf constructed has sections which represent which nodes are able to broadcast simultaneously. It is then demonstrated how to combine multiple copies of this sheaf into a queuing sheaf upon which protocols can be developed.

Our general program seeks to cast networking ideas in the language of sheaves to strengthen the foundations of networking. Our hope is that sheaves will provide a sufficiently strong foundation to allow us to construct networks with more relaxed assumptions, such as delay-tolerant networks.

IV. A Sheaf for Dijkstra

Dijkstra's algorithm [23] is a classic tool for networking, so in [24] we use sheaves to give a structure for determining the shortest path from a source to a sink as sections of a sheaf. While we go over basic definitions, constructions, and examples below, the main details are in the paper above. The assumptions for Dijkstra's algorithm are as follows:

- $G = (V, E)$, a directed graph.
- $v_S, v_T \in V$, the dedicated source and sink of G respectively.
- $w : E \to \mathbb{R}^+$, a weight function defined on each edge of G.

Notice that if we restrict the capacity function c of a time-extended graph to have codomain $\mathbb{R}^+$, and establish a source and sink vertex, we already have the structures needed to run Dijkstra's algorithm. The interpretation may be slightly different than the original intention of a time-extended graph, but the structure is sufficient for use here.

The core of Dijkstra's algorithm is determining the minimum weight path from v_S to v_T. So, one approach to constructing a sheaf for Dijkstra's algorithm might be to come up with a sheaf that gives paths from v_s to v_t as sections and then compute the minimum weight across those sections. This is the approach taken in subsection IV.A. Another approach might be to bake the weight into the sheaf so that the sections track the weight at v_t. This is the approach taken in subsection IV.B. It is this approach that manifests a more local-to-global representation of Dijkstra's algorithm. Both of these are presented in more detail in [24].

A. Paths First, then Weighting

Let $\mathrm{In}(v)$ denote the set of edges pointing at a vertex v and $\mathrm{Out}(v)$ denote the set of edges pointing away from a vertex v, and let the symbols $\perp$ denote an inactive edge or vertex and $\top$ denote an active edge. We define the path sheaf $\mathcal{P}$ on G to map vertices by the rule

$$\mathcal{P}(v) = \begin{cases} \mathrm{Out}(v) & \text{if } v = v_S \\ \mathrm{In}(v) & \text{if } v = v_T \\ (\mathrm{In}(v) \times \mathrm{Out}(v)) \cup \{\perp\} & \text{otherwise} \end{cases},$$

and edges by the rule

$$\mathcal{P}(e) = \{\bot, \top\}.$$

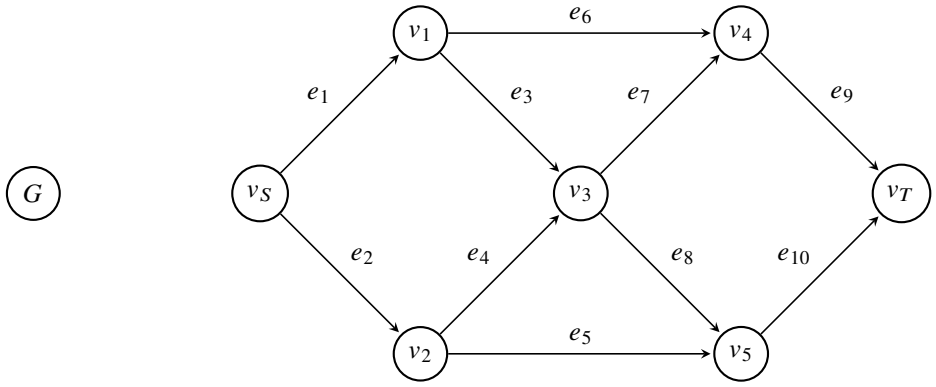
If $v = v_S, v_T$, define the restriction map

$$\mathcal{P}(v \rightsquigarrow e)(e_i) = \begin{cases} \top & \text{if } e = e_i \\ \bot & \text{otherwise} \end{cases}.$$

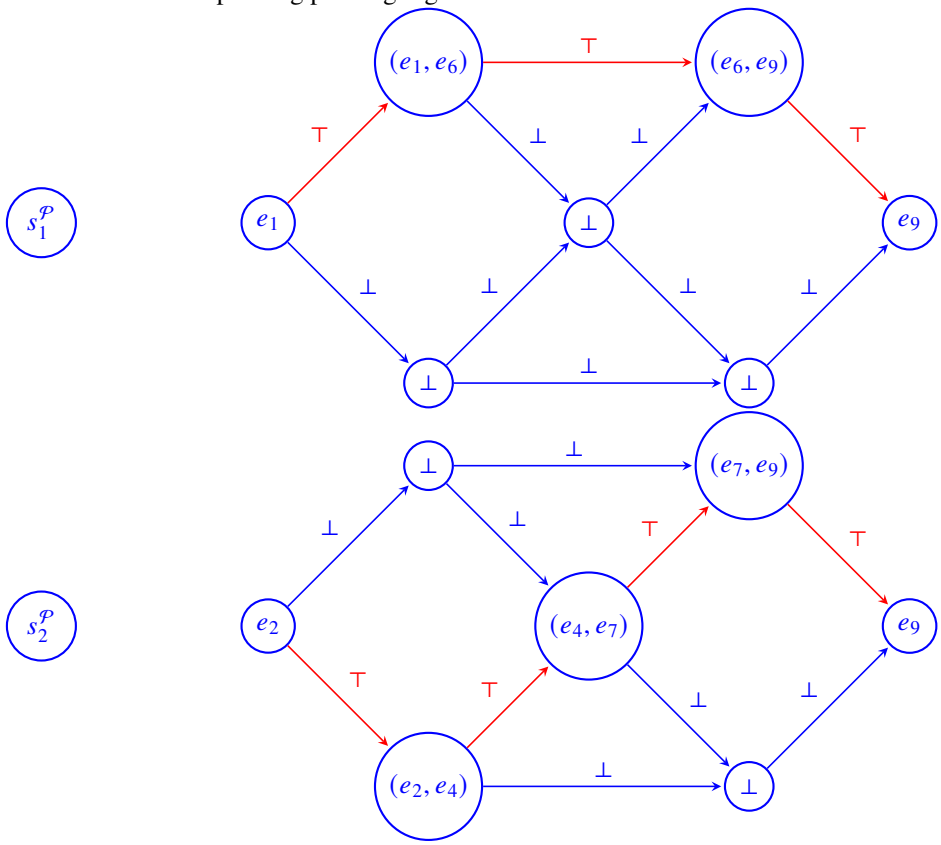
If $v \neq v_S, v_T$, define the restriction map by

$$\mathcal{P}(v \rightsquigarrow e)(x) = \begin{cases} \top & \text{if } x = (e_{in}, e_{out}) \text{ and } e = e_{in}, \text{ or } e_{out} \\ \bot & \text{otherwise} \end{cases}.$$

This sheaf captures the paths from v_S to v_T via the sections. As an example, consider the graph G depicted below.



We can construct the sets defining $\mathcal{P}$ over G, but what is most interesting are the sections of $\mathcal{P}$. There are precisely six global sections of $\mathcal{P}$ which correspond to the six possible paths from v_S to v_T. Below are diagrams for two of the sections with the corresponding path highlighted in red.



Once we have these sections, we need to use some weight function to compute the cost over each path. Then, the output of Dijkstra's algorithm would be whatever the minimum cost path would be. Admittedly, this approach is unsatisfying as a representation of Dijkstra's algorithm since there is no obvious use to the weight function in the sections. If we begin incorporating weights into the sheaf itself, we seemingly get a more complicated sheaf, but it will better represent Dijkstra's algorithm as a sheaf.

B. Baking Weights into Sheaves

Since we will be incorporating the weight function, we begin here by assuming G is a weighted graph with weight function $w : E \to \mathbb{R}^+$. We define the distance path sheaf $\mathcal{DP}$ on G to map vertices by the rule

$$\mathcal{DP}(v) = \begin{cases} \mathrm{Out}(v) \times \{0\} & \text{if } v = v_S \\ \mathrm{In}(v) \times \mathbb{R}^+ & \text{if } v = v_T \\ (\mathrm{In}(v) \times \mathrm{Out}(v) \times \mathbb{R}^+) \cup \{\perp\} & \text{otherwise} \end{cases}$$

and edges by

$$\mathcal{DP}(e) = \mathbb{R}^+ \cup \{\perp\}$$

accordingly. For the source, define the restriction map

$$\mathcal{DP}(v_S \rightsquigarrow e)(e_i, 0) = \begin{cases} w(e) & \text{if } e = e_i \\ \perp & \text{otherwise} \end{cases}.$$

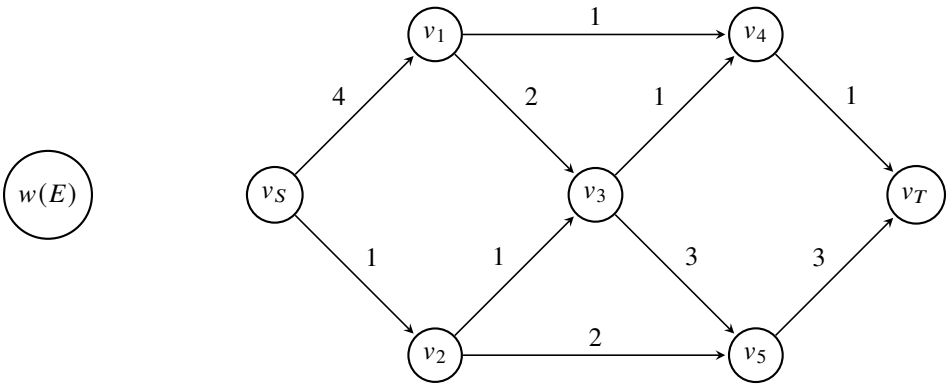
But, for the sink, define the restriction map

$$\mathcal{DP}(v_T \rightsquigarrow e)(e_i, x) = \begin{cases} x & \text{if } e = e_i \\ \perp & \text{otherwise} \end{cases}.$$

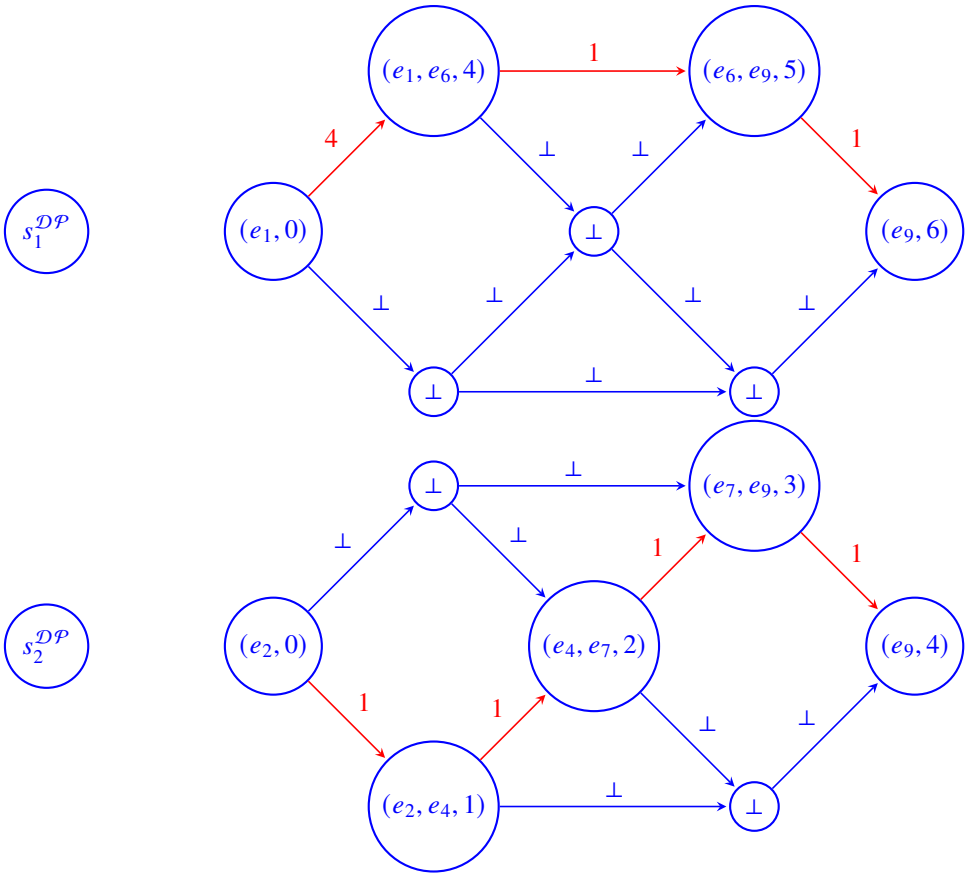
If v is a non-sink and a non-source node, define the restriction maps by

$$\mathcal{DP}(v \rightsquigarrow e)(\alpha) = \begin{cases} x & \text{if } \alpha = (e_{in}, e_{out}, x) \text{ and } e = e_{in} \\ x + w(e) & \text{if } \alpha = (e_{in}, e_{out}, x) \text{ and } e = e_{out} \\ \perp & \text{otherwise} \end{cases}.$$

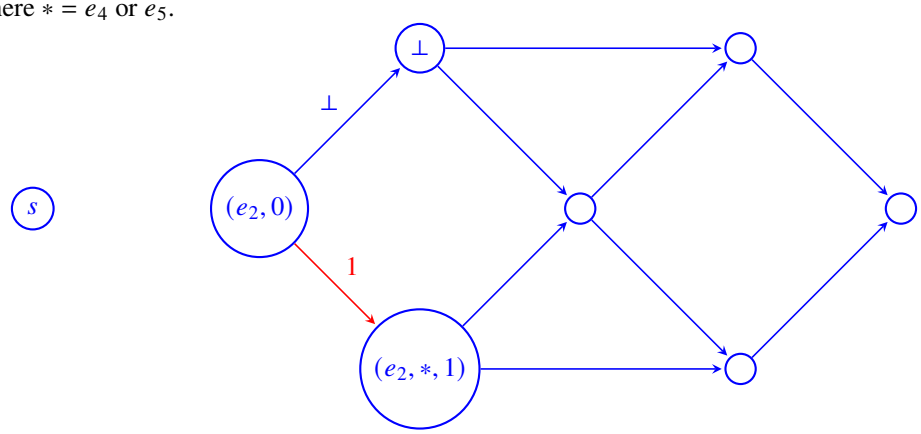
The key difference here is that the weights are included in the sheaf valuations. In addition, this sheaf tracks the sum of the weights as a path is traversed, so in any given section s, $s(v_T)$ will contain the cost of traversing the path. We return to our prior example to demonstrate this, but first we apply the following weight function to the edges of G:



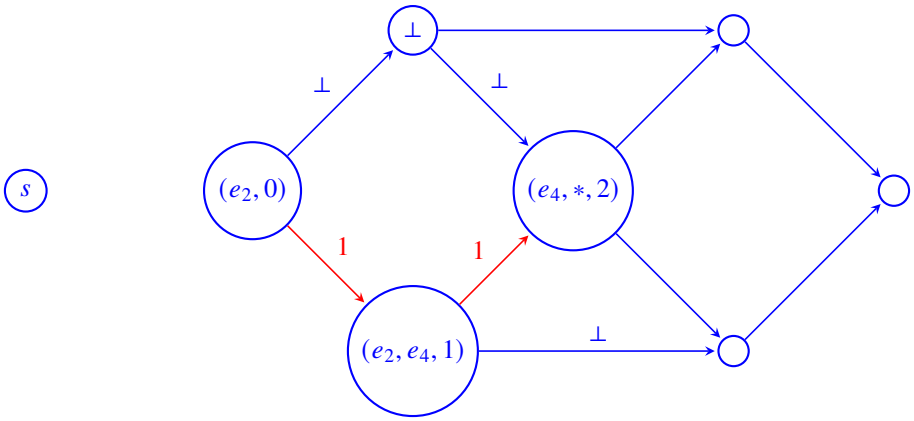
With this weight function in hand, we note that there are still exactly six global sections that correspond to the six paths from v_S to v_T. Below are the two sections corresponding to $s_1^{\mathcal{P}}$ and $s_2^{\mathcal{P}}$ from before:



The beauty of these sections is that the weight updates live, so as a local section is extended, the weight updates with it. This allows us to recognize Dijkstra's algorithm as an algorithmic extension of local sections to global sections of $\mathcal{DP}$. Consider extending the local section $s(v_S) = (e_2, 0)$. The restriction maps dictate that $s(e_2) = 1$ and $s(e_1) = \perp$ is the only valid extension over e_1 and e_2. Then, the only valid extension to v_1 and v_2 are $s(v_1) = \perp$ and $s(v_2) = (e_2, *, 1)$, where $* = e_4$ or e_5.



From here we have a choice. We can choose $s(v_2) = (e_2, e_4, 1)$ which passes to $s(e_4) = 1$ and $s(v_3) = (e_4, *, 2)$, where $* = e_7$ or e_8. Alternatively, we can choose $s(v_2) = (e_2, e_5, 1)$ which passes to $s(e_5) = 2$ and $s(v_5) = (e_5, e_{10}, 3)$. Since $3 > 2$, we will take the first option, leading us to make $s(e_5) = \perp$.



Continuing this line of argumentation, we can get s to represent the minimum weight path (which is equivalent to $s_2^{\mathcal{DP}}$ from before). The only piece left to the interested reader is the comparison of local section options with different weights to achieve the shortest possible path. This is trivial, however, due to the sums being incorporated into the sections. Details of this construction can be found in [24].

We are confident that other algorithms can be realized as sheaves and local sections extending to global sections. Other routing algorithms, such as Bellman-Ford or Floyd-Warshall, would be of particular interest to us as fairly straightforward generalizations of $\mathcal{P}$ or $\mathcal{DP}$.

V. Casting Sheaf Structures

One assumption of [22] is that each node communicates by broadcasting to all possible nodes in the network. However, we wanted a framework in which we could describe more realistic schemes, such as unicast communications. As antennas are largely directional, and the focus of optical communications is greater still, we wish to not only consider non-broadcast methods, but situations where choosing one target necessarily precludes choosing others. In [25], we construct novel data types for sheaves to give a stronger means of modeling communication over networks. This construction is sufficiently general to describe unicast, multicast, and broadcast scenarios. In this paper, we do cover definitions and constructions, but the purpose is to cover intuition and the connection to DTNs. The intrepid reader is encouraged to read [25].

Recall that broadcast communications are when each node transmits to every node that is willing and able to listen. In this setup, it makes sense for the sheaf to hold all neighbors to the node. In a directed graph, especially a time-extended graph, we will want $\text{Nbd}(v) = \{u \in V^{TE} \mid (u, v) \text{ or } (v, u) \in E^{TE}\}$ to represent the neighborhood of the vertex v. By contrast, unicast communications are when each node transmits to only one node at a time. Here, we want to restrict the information in the sheaf so that it only tracks the nodes that are actively involved in communications. Multicast communications encompass the full middle ground, and hence is the most desirable type to model. However, with this generality comes complexity. In the following section, we construct a simplified version of a universal multicast sheaf which can then be restricted and expanded to achieve different multicast scenarios, including both broadcast and unicast scenarios.

A. Universal Multicast Sheaf

Let G^{TE} be the time-extended graph for temporal flow network $G = (V, E, L, v_S, v_T, c, B)$. We define the universal multicast sheaf $\mathcal{M}$ on G^{TE} to map vertices by the rule

$$\mathcal{M}(v_i) = \{A \subset V^{TE} \mid v_i \in A \text{ and } A \setminus \{v\} \subseteq \text{Nbd}(v)\}$$

and edges, allowing that $e^{TE}_{v,i} = e^{TE}_{v,v,i}$, by the rule

$$\mathcal{M}(e^{TE}_{v,w,i}) = \{\perp, \top\}.$$

Then, the restriction maps are defined by

$$\mathcal{M}(v_{i-1} \rightsquigarrow e^{TE}_{v,w,i})(A) = \begin{cases} \top & \text{if } v_{i-1} \in A \\ \perp & \text{otherwise} \end{cases},$$

and

$$\mathcal{M}(w_i \rightsquigarrow e^{TE}_{v,w,i})(A) = \begin{cases} \top & \text{if } w_i \in A \\ \bot & \text{otherwise} \end{cases}.$$

where the case where $w = v$ is handled in this same general case. Sections of this sheaf correspond to every possible multicast scenario on G^{TE}. If we insist that one active edge coming out of a node means that every edge coming out of a node is active, we get broadcast. If we insist that only one edge coming out of a node can be active at any given time, we get unicast. Then, anything in between is covered somewhere as a section of this sheaf.

Much like in [22], it is possible to combine multiple multicast sheaves to construct queuing sheaves for multicast routing. Following this, we can then design and compare multicast routing protocols and discuss other sheaf-theoretic approaches. The mathematics behind this gets more complicated because this type of sheaf is more complicated, however it has been done. By considering directional graphs, *directional cellular sheaves* have been constructed, which give rise to unicast sheaf, the broadcast sheaf, and the multicast sheaf seen above. For more details, see [25], which includes examples and a discussion on the cohomology of the sheaves in question, including the queuing sheaf.

VI. Future Work

The general program of this work is to cast networking ideas using the language of sheaves in order to strengthen the foundations of networking in a naturally unifying way. Towards that goal, there are several directions of future work. The most obvious is to continue by identifying networking ideas and transforming them into sheaves. In fact, it is possible to layer sheaves on top of each other, so a grand plan might be to find a way to represent (and then improve) the OSI model layers as sheaves related appropriately.

That being said, to use sheaves as a foundation for networking requires us to be able to demonstrate capability. This can happen in several ways:

- Studying routing in general and particular DTNs. The dynamic and heterogeneous nature of DTNs make their structure difficult to harness. Understanding how to achieve scale of a large network, which includes securing and managing it, remain necessary-yet-uncleared hurdles for massive DTN deployment. While the times have not caught up with this need at the time of publication, we believe that these challenges need to be addressed before they become emergencies.
- Along the same lines of going beyond OSI, and a little farther out, one might also consider cross-layer protocols. By sharing information across layers, optimizations have been made with respect to energy consumption in wireless networks [26]. Putting this on rigorous grounds could open doors to other forms of optimization. DTNs are often realized as overlay networks, which turn disparate collections of links into cohesive networks, but without integrating much knowledge of lower layers.
- Sheaves are sufficiently complex to model network coding well. In [27], Ghrist and Hiraoka give a construction for network coding as a sheaf over a network, and they use it prove a max-flow, min-cut result. This complexity can be leveraged to construct more efficient flows through a network. Taking this a step further, several applications of network coding have been made to DTN [28][29]. These include improving the efficiency of DTN routing and reducing delivery times. The network coding structures can be more accurately captured in the framework of sheaves.
- Another means of demonstrating capability comes from being able to construct sheaves in code and compare against other routing algorithms for DTN.
- Finally, using these tools, we can analyze and better networking as a whole.

VII. Conclusion

Developing the theory of DTNs is only one part of the goal, as it is not sufficient if it cannot be applied to DTN for its benefit. Fortunately, there are existing tools, such as PySheaf [30], that give a means of constructing sheaves in code. In the future, we would like to see the sheaves mentioned in this paper, and sheaves yet to be constructed, coded up so that a stronger comparison to known algorithms can be made. While PySheaf may not represent the final form of these implementations, it offers a strong starting position.

By means of PhSheaf and occasionally more direct methods, we have used sheaves to generalize existing algorithms, such as Dijkstra's, and even to develop novel star tracking algorithms [31] - this is not surprising due to their geometrical nature.

We have demonstrated deep connections between tools in modeling networks, network algorithms, protocols, and sheaves, following in the footsteps of [22] and [11]. The results have only bolstered our confidence that this is the right track for bringing the structure of DTNs to light. It is our hope that sheaves will inspire stronger algorithms and deeper theory in the same way that graphs have in the past.

References

[1] Israel, D., Edwards, B., Hayes, J., Knopf, W., Robles, A., and Braatz, L., "The Benefits of Delay/Disruption Tolerant Networking for Future NASA Science Missions," *70th International Astronautical Congress (IAC)*, 2019, pp. 1–12.

[2] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., and Weiss, H., "RFC 4838, Delay-Tolerant Networking Architecture," *IETF Network Working Group*, 2007. URL `https://tools.ietf.org/html/rfc4838`.

[3] Araniti, G., Bezirgiannidis, N., Birrane, E., Bisio, I., Burleigh, S., Caini, C., Feldmann, M., Marchese, M., Segui, J., and Suzuki, K., "Contact graph routing in DTN space networks: overview, enhancements and performance," *IEEE Communications Magazine*, Vol. 53, No. 3, 2015, pp. 38–46. https://doi.org/10.1109/MCOM.2015.7060480.

[4] Hylton, A., Raible, D., and Clark, G., "A Delay Tolerant Networking-Based Approach to a High Data Rate Architecture for Spacecraft," *2019 IEEE Aerospace Conference*, 2019, pp. 1–10. https://doi.org/10.1109/AERO.2019.8742135.

[5] Scott, K., and Burleigh, S., "RFC 5050, Bundle Protocol Specification," *IETF Network Working Group*, 2007. URL https://tools.ietf.org/html/rfc5050.

[6] Bezirgiannidis, N., and Tsaoussidis, V., "Predicting Queueing Delays in Delay Tolerant Networks with Application in Space," *Wired/Wireless Internet Communications*, edited by A. Mellouk, S. Fowler, S. Hoceini, and B. Daachi, Springer International Publishing, Cham, 2014, pp. 228–242.

[7] Al Hanbali, A., de Haan, R., Boucherie, R. J., and van Ommeren, J.-K., "A Tandem Queueing Model for Delay Analysis in Disconnected Ad Hoc Networks," *Analytical and Stochastic Modeling Techniques and Applications*, edited by K. Al-Begain, A. Heindl, and M. Telek, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 189–205.

[8] Cabacas, R., and Ra, I., "Evaluating Mobility Models in Delay Tolerant Network," *2013 International Conference on IT Convergence and Security (ICITCS)*, 2013, pp. 1–4.

[9] Sehgal, R., and Peyravi, H., "Delay tolerant networks modeling and analysis," *Proceedings of the 30th International Conference on Computers and Their Applications, CATA 2015*, 2015, pp. 231–236.

[10] Arabi, S., Sabir, E., and Elbiaze, H., "Information-centric networking meets delay tolerant networking: Beyond edge caching," *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pp. 1–6.

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

[12] Fraire, J. A., Madoery, P., Burleigh, S., Feldmann, M., Finochietto, J., Charif, A., Zergainoh, N., and Velazco, R., "Assessing Contact Graph Routing Performance and Reliability in Distributed Satellite Constellations," , Jul. 2017. https://doi.org/https://doi.org/10.1155/2017/2830542, URL https://www.hindawi.com/journals/jcnc/2017/2830542/.

[13] Segui, J., Jennings, E., and Burleigh, S., "Enhancing Contact Graph Routing for Delay Tolerant Space Networking," *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, 2011, pp. 1–6.

[14] Demmer, M., and Fall, K., "DTLSR: Delay Tolerant Routing for Developing Regions," *Proceedings of the 2007 workshop on Networked systems for developing regions*, Association for Computing Machinery, New York, NY, USA, 2007. https://doi.org/10.1145/1326571.1326579, URL https://doi.org/10.1145/1326571.1326579.

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

[16] Grasic, S., Davies, E., Lindgren, A., and Doria, A., "The Evolution of a DTN Routing Protocol - PRoPHETv2," *Proceedings of the 6th ACM Workshop on Challenged Networks*, Association for Computing Machinery, New York, NY, USA, 2011, p. 27–30. https://doi.org/10.1145/2030652.2030661, URL https://doi.org/10.1145/2030652.2030661.

[17] Akrida, E. C., Czyzowicz, J., Gasieniec, L., Kuszner, L., and Spirakis, P. G., "Flows in Temporal networks," *CoRR*, Vol. abs/1606.01091, 2016. URL http://arxiv.org/abs/1606.01091.

[18] Tennison, B. R., *Sheaf Theory*, London Mathematical Society Lecture Note Series, Cambridge University Press, 1975. https://doi.org/10.1017/CBO9780511661761.

[19] Ghrist, R., *Elementary Applied Topology*, CreateSpace Independent Publishing Platform, 2014.

[20] Curry, J., "Sheaves, Cosheaves and Applications," 2013.

[21] Robinson, M., *Topological Signal Processing*, Mathematical Engineering, Springer Berlin Heidelberg, 2014.

[22] Robinson, M., "Modeling wireless network routing using sheaves," *arXiv*, 2016.

[23] Dijkstra, E. W., et al., "A note on two problems in connexion with graphs," *Numerische mathematik*, Vol. 1, No. 1, 1959, pp. 269–271.

[24] Moy, M., Cardona, R., Green, R., Cleveland, J., Hylton, A., and Short, R., "Path Optimization Sheaves," , N.D. Unpublished.

[25] Bainbridge, G., Hylton, A., and Short, R., "Directional Cellular Sheaves for Multicast Network Routing," , N.D. Unpublished.

[26] Messaoudi, A., Elkamel, R., Helali, A., and Bouallegue, R., "Cross-layer based routing protocol for Wireless Sensor Networks using a fuzzy logic module," *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2017, pp. 764–769.

[27] Ghrist, R., and Hiraoka, Y., "Applications of sheaf cohomology and exact sequences to network coding," *Proc. NOLTA*, 2011.

[28] Zhang, Q., Jin, Z., Zhang, Z., and Shu, Y., "Network Coding for Applications in the Delay Tolerant Network (DTN)," 2009, pp. 376–380.

[29] Vazintari, A., Vlachou, C., and Cottis, P. G., "Network Coding for Overhead Reduction in Delay Tolerant Networks," *Wireless Personal Communications*, Vol. 72, No. 4, 2013, pp. 2653–2671. https://doi.org/10.1007/s11277-013-1172-2, URL https://doi.org/10.1007/s11277-013-1172-2.

[30] Robinson, M., Capraro, C., and Praggastis, B., "The pysheaf library," , 2016. URL https://github.com/kb1dds/pysheaf.

[31] Green, R., Cardona, R., Cleveland, J., Ozbolt, J., Hylton, A., Short, R., and Robinson, M., "Dude Where's My Stars: A Novel Topologically Justified Approach to Star Tracking," *In Proceedings*, 2020, pp. 1–15.