Cooperative Clustering Techniques For Space Network Scalability

Yael Kirkpatrick
Mathematics Department
Massachusetts Institute of Technology
Cambridge, MA
yaelkirk@mit.edu

Rachel Dudukovich
Cognitive Signal Processing Branch
NASA Glenn Research Center
Cleveland, OH
rachel.m.dudukovich@nasa.gov

Prash Choksi
Computer Science
University of Houston-Clear Lake
Houston, TX
prash.choksi01@gmail.com

Dominick Ta
Computer Science
University of Washington
Seattle, WA
domta@cs.washington.edu

***Abstract*—Routing in the space internet must face many unique challenges - from unplanned disconnections and interruptions to predictable intermittent connectivity due to high network mobility and long propagation delays. NASA's current approach to such routing is Contact Graph Routing (CGR), using a graph formed of prescheduled communication contacts to compute routes through the network. While this approach manages to tackle issues of connectivity and propagation delays, it is a global approach that requires continuous knowledge of the entire network. In a potential future Solar Space Internet (SSI) such an approach on its own cannot scale to large networks with thousands of members. In this paper we propose clustering as a solution to CGR scalability. Clustering has been used in many networking problems as a way to subdivide the network and allow for localized routing and better scalability. Using techniques from graph theory and game theory, we explore various existing clustering algorithms and adapt them to the Contact Graph Routing setting. We propose a way to combine multiple algorithms to create a Delay Tolerant Clustering Protocol (DTCP). In addition, we explore the underlying networking mechanisms such as multicast, neighbor discovery, and software defined networking that may be used to enable DTCP.**

***Index Terms*—Delay Tolerant Networking, Cooperative Networks, Game Theory, Clustering, Contact Graph Routing, Software Defined Networking, Controller Placement**

I. Introduction

As NASA sets its sights on the Moon once again, a robust space internet is needed to accommodate the increasing need for bandwidth. This network has to tackle many challenges imposed by the space environment. End-to-end connectivity in the network may never exist, connections can vary over time and frequent delays and interruptions can occur. As a result, many algorithms that function well in terrestrial internet are not suitable for the space environment, but rather Delay-Tolerant Networking (DTN) protocols and algorithms are needed [1].

One of the main challenges faced by DTN algorithms is that of routing [2]. Paths in a time-varying space network may change over time, connections can form and disappear and sometimes no path exists between two endpoints. These unique challenges mean than in DTN it is not possible to use common terrestrial algorithms and protocols that assume a stable, connected network. However, in space networks most connections are planned and known well in advance [3]. This allows us to form contact plans detailing the exact contacts expected between nodes of the network in the nearby future. Using these contacts, we can construct a contact graph and adapt various routing algorithms to this setting. This approach is known as Contact Graph Routing (CGR) [4].

In Contact Graph Routing we assume that every node in the network possesses the knowledge of the entire network topology with all the relevant contacts planned to take place in the near future. The node maintains these contacts in a data structure known as a Contact Graph and runs pathfinding algorithms on it. As networks become larger, these pathfinding algorithms become slow and inefficient, imposing a heavy computational burden on the nodes of the network. Furthermore, maintaining an up-to-date contact plan at every node in the network requires an expensive communication overhead as nodes join or leave the network and connections change over time. To allow CGR to function for the future of space internet, with thousands of nodes or more, we need to address its scalability. This issue of DTN scalability was explored in [5], however this work is not immediately applicable to CGR as it does not consider contact based routing solutions.

In [6], the authors first suggest the idea of using clustering as a possible way to address scalability in DTN with CGR in mind. Their work introduces the notion of dividing the network into regions and applying different routing algorithms for inter- and intra-region routing. Clustering can offer additional benefits to space communication, allowing the integration of commercial networks and various methods of routing within and between different clusters.

In this paper we aim to build upon the work in [6] by proposing a clustering scheme fit for CGR. In our approach the network is divided into clusters, each run by a cluster head. The cluster head is responsible for maintaining the cluster and holds more information about the global network. Cluster members can then query the cluster head for the additional information it holds. We consider both global and local knowledge available to nodes in the network and provide a protocol for creating and maintaining cluster structures: forming clusters, changing cluster membership, appointing cluster heads and merging or splitting clusters.

To do so we consider various approaches to clustering problems. As we aim to tackle scalability, we require an algorithm that needs little knowledge of the global network and runs on a small, local scale. We use tools from both cooperative and non-cooperative game theory to model the decisions around forming clusters and electing cluster heads. This allows for a distributed, local algorithm with global consequences.

Organization

In section II we introduce some necessary background on Contact Graph Routing and Game Theory. Then, in section III, we introduce three different existing clustering algorithms that are helpful for the problem of clustering in CGR. We adapt these algorithms to the CGR setting in section IV and then propose a clustering protocol that combines the three approaches in section V. In section VI we show some preliminary simulated results as to the performance of the proposed protocol. Finally, in section VII we discuss several considerations towards an implementation approach.

II. Background

A. Contact Graph and Multigraph Routing

Contact Graph Routing (CGR) refers to a collection of routing algorithms for DTN running on a data structure called a contact graph[4]. A *contact* is a period of time when two nodes in the network are able to communicate and data can be transmitted. We denote a contact as $C_{A,B}^{t_0,t_1}$, where A is the source node, B is the destination node, t_0 is the time at which the contact starts and t_1 the time at which it ends. These contacts are precomputed and distributed throughout the network in the form of Contact Plans.

Pathfinding in CGR unifies these contacts into a data structure known as a Contact Graph. In this graph the vertices represent the contacts, not the nodes in the original network, and edges represent the fact that data can be transferred between two contacts. More formally, a directed edge is drawn between $C_{A,B}^{t_0,t_1}$ and $C_{D,E}^{t_2,t_3}$ if $B = D$ and $t_3 > t_0$. This represents that data can be transferred to B using $C_{A,B}^{t_0,t_1}$ and then forwarded to E using $C_{D,E}^{t_2,t_3}$. To find a path between node A to node D starting at a particular time t_0, a root and contact $C_{A,A}^{t_0,\infty}$ and a terminal contact $C_{D,D}$ are added to the contact graph [4].

Another way to represent these contacts, introduced in [7], is through a Multigraph, a graph in which two vertices can have multiple edges between them. In this representation, the nodes of the graph correspond to the nodes of the network and edges between two nodes represent a contact. These time varying intervals of the contacts can be considered as labels on the edges. As the vertices now represent the original network nodes, this model is more intuitive and more similar to traditional network models.

As an example for the two representations, consider a simple network with three nodes A, B, D, shown below in both Contact Graph and Multigraph form.

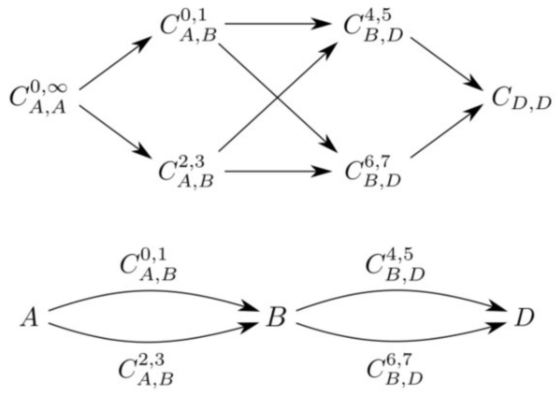

Fig. 1. A Contact Graph (above) and a Multigraph (below) of the same network [7].

It was shown in [7] that the same pathfinding algorithms used in CGR can be applied to the multigraph representation and are in fact more efficient. Therefore, for the remainder of this paper we will consider the problem of routing on a Contact Multigraph.

B. Game Theory

Game theory is a field of mathematics that studies models of conflict and cooperation between intelligent rational decision makers. Game theory provides general mathematical techniques for analyzing situations in which two or more individuals make decisions that will influence one another's welfare [8]. We can differentiate between two types of games - cooperative, where the players share a common goal, and non-cooperative, where the players aim to maximize their own payoff without regarding that of their contenders. Both of these types of games can be of use when considering delay tolerant networks. While we might assume nodes would want to cooperate in order to maximize the well being of the entire network, sometimes such cooperation might be implausible due to long distances and disconnections. It is also beneficial to consider non-cooperative games as they require less communication overhead and allow nodes to make local, individual decisions.

1) Cooperative Games: First we consider cooperative games, specifically *coalition games*. Informally, we think of such a game as players trying to form a coalition, where every subset of the players (or coalition) has some given value. The players would like to divide this value among the members of the subset, to determine how much each player is "worth" to the coalition. More formally:

Definition 1 (Cooperative Games [9]). *A cooperative game on n players $S = \{s_1, \ldots, s_n\}$ is defined by a* ***characteristic function*** *v. $v : 2^S \rightarrow \mathbb{R}$ is defined on subsets of the players and measures the value, or payoff, that a subset of players can achieve on their own, regardless of what the remaining players do. This value can then be split among the players in any way that they agree on. The characteristic function satisfies the following properties:*

- *$v(\emptyset) = 0$.*
- *Monotonicity: For any two subsets $T_1, T_2 \subseteq S$, if $T_1 \subseteq T_2$, then $v(T_1) \leq v(T_2)$.*

Given a characteristic function v, an outcome of the game is an allocation vector $\psi(v) \in \mathbb{R}^n$ where $\psi_i(v)$ is the share of the payoff allocated to player i. The *Shapley Value* is a solution concept that allocates each player the payoff that matches its marginal contribution to the coalition. It is uniquely determined by Shapley's four desired properties of a solution: symmetry, or that identical players receive the same payoff; dummy, or that a player whose addition to the coalition does not change it value receives no payoff; efficiency, or that the payoff of all individual players adds to the value of the coalition of all of them together; additivity, or that the payoff of a player in the sum of two games is equal to the sum of its payoffs in each game. The Shapley Value is defined as follows.

Definition 2 (Shapley Value [9]). *The Shapley Value of a player i is defined as:*

$$\psi_i(v) = \frac{1}{n!} \sum_{\pi \in S_n} \varphi_i(v, \pi).$$

Where $\varphi_i(v, \pi)$ is defined as the marginal contribution of player i at the time of its arrival, assuming players arrive in the order of the permutation π,

$$\varphi_i(v, \pi) = v\left(\pi\left\{1, \ldots, \pi^{-1}(i)\right\}\right) - v\left(\pi\left\{1, \ldots, \pi^{-1}(i) - 1\right\}\right).$$

We will use the Shapley Value in future algorithms to measure the relative 'importance' of nodes in the network. This will help identify nodes that are most central in their environment and can function as good cluster heads.

2) Non-Cooperative Games: Next, we consider non-cooperative games, where players compete to optimize their individual payoff based on their own and other payers' strategies.

Definition 3 (Non-cooperative Games [10]). *A non-cooperative game is defined as a set of players, each with a set of strategies, and a payoff function. Formally, we write $(N, (\Delta_0)_{i \in N}, (P_i)_{i \in N}$ where*

- *N is the set of players.*
- *Δ_i is the strategy set of player i.*
- *$P_i : \Delta_1 \times \ldots \times \Delta_n \rightarrow \mathbb{R}$ is the payoff of player i given the choice of strategies by all players.*

In non-cooperative games we consider a point of equilibrium, a set of strategies for each players at which none would benefit by switching strategies. We call such a point a *Nash Equilibrium*.

Definition 4 (Nash Equilibrium [10]). *A strategy profile $\delta^* = (\delta_1^*, \ldots, \delta_n^*)$ is said to be a Nash Equilibrium of a game if for every $i \in [n]$,*

$$P_i(\delta^*) \geq P_i(\delta_i, \delta_{-1}^*).$$

Where $\delta_i \in \Delta_i$. In other words, a Nash Equilibrium is a state at which no player has any incentive to change their chosen strategy.

When considering non-cooperative games we compute their *Nash Equilibria*, as the players in the games will continuously change their strategies to improve their individual payoffs until such a point is reached. It is therefore usually the case that players' strategies correspond to a point of Equilibrium and we can analyze the overall payoff at these points to evaluate the outcome of the game.

III. Existing Approaches to Clustering

Clustering is a well studied problem in networking, applied in various setting to graph routing problems. Clustering has been studied as a way to reduce the size of routing tables, subdivide a network and improve the scalability of routing algorithms, all while impairing path efficiency by at most a constant factor [11].

In this section we will introduce three recently published clustering algorithms that are useful for our problem of clustering in Contact Graph Routing. In the following sections we will explore how to adapt these algorithms to our Contact Graph setting and how to use them in a delay tolerant clustering protocol.

A. First Approach - Cooperative Game Based Clustering

This algorithm is based on the one introduced in *Game Theory Based Network Partitioning Approaches for Controller Placement in SDN* [10]. It divides the network by computing each node's *Shapley Value* based on a specific value function. At every iteration the algorithm chooses the node with highest Shapley value, assigning a controller at that location and assigning any node close enough to it (depending on a preset threshold) to its cluster. This is done iteratively until all nodes have been assigned to clusters.

Denote the set of nodes in the graph by $S = \{s_1, \ldots, s_n\}$ The value function used in this approach is

$$v(F) = \sum_{s_1 \neq s_2 \in F} U(d(s_1, s_2)),$$

where U is defined using the diameter of the graph $diam$, by $U(x) = 1 - \frac{x}{1+diam}$.

While the Shapley value is usually very costly to compute, in the case of this algorithm we are using a *convex game*, a game where the marginal contribution of a player increases in larger coalitions. In such cases the Shapley value comes out to a simple sum,

$$\Phi_i = \frac{1}{2} \sum_{s_j \in S, j \neq i} U(d(s_i, s_j)).$$

Using this Shapley Value, the algorithm is as follows:

Algorithm 1 Cooperative Game Theory Initialization[10]

```
Input: Set of nodes $S$, all pair shortest path matrix $D$,
       threshold $\alpha \in (0, 1]$.
Output: Set of cluster heads $C_0$.
1: $Q = S,\ C_0 = \emptyset$.
2: for $i = 1, \dots, n$ do
3:     $\Phi_i = \frac{1}{2} \sum_{s_j \in S, j \neq i} U(d(s_i, s_j))$.
4: while $Q \neq \emptyset$ do
5:     $m = \arg\max_{i \in Q} \Phi_i$.
6:     $C_0 = C_0 \cup \{m\}$.
7:     $T_m = \{i \in Q : U(d(s_i, s_m)) \geq \alpha\}$.
8:     $Q = Q \setminus T_m$.
```

This approach provides a balanced partition of the network with centralized, distributed cluster heads. However, it requires the knowledge of the entire network in order to compute the all pair shortest path matrix and in turn the Shapley Values of all the nodes. Therefore, for the cases in which we do not have access to the entire network we need to consider more local clustering approaches.

B. Second Approach - Dynamic Local Clustering

The second approach we consider is based on the algorithm introduced in *Dynamic Local Clustering for Hierarchical Ad Hoc Networks* [12]. In this paper the authors introduce a measure of 'graph fitness' that by optimizing it we create clusters that are both dense and introvert. For a subset of nodes $C \subseteq V$, define $\text{deg}_{\text{int}}(C)$ to be the number of edges with both endpoints in C and $\text{deg}_{\text{ext}}(C)$ to be the number of edges with exactly one endpoint in C. We then define the fitness of a cluster to be

$$f(C) = \frac{2\text{deg}_{\text{ext}}(C)^2}{|C|(|C|-1)(\text{deg}_{\text{int}}(C) + \text{deg}_{\text{ext}}(C))}.$$

In the algorithm proposed in [12], when a node joins the network it queries all its neighbors. Each neighbor responds with the cluster it belongs to, along with the parameters $|C|, \text{deg}_{\text{int}}(C), \text{deg}_{\text{ext}}(C)$. Using this information, the node can then compute the cluster for which its joining would create the highest increase in fitness (or smallest decrease). A node can later decide to move from cluster C_1 to C_2 if by doing so the value of $f(C_1) + f(C_2)$ is increased.

In this approach, cluster heads are selected as the first node to join the cluster. This happens when a node is disconnected from the graph and receives no responses to its cluster queries, in which case it creates its own cluster and appoints itself as the cluster head. In the last approach we consider we tackle the problem of smart cluster head selection.

C. Third Approach - Non-Cooperative Game Based Cluster Head Selection

The third and final approach we consider is based on the clustering algorithm proposed in *Game Theory Based Distributed Clustering Approach to Maximize Wireless Sensors Network Lifetime* [13]. This paper introduces a clustering algorithm centered around the selection of cluster heads. The clusters are then formed by each node joining the cluster of the cluster head closest to it. These cluster heads are selected through a non-cooperative game played by the nodes in the network. In the game, players can choose whether to participate (thus volunteering to possibly become a cluster head) or stay out of the game, while attempting to optimize their individual residual energy. This quantity of residual energy is formalized in an energy model introduced in [13].

Formally, the algorithm introduced in [13] is called the Profitable Energy Market Game (PEMG). In this game each player can choose between two strategies: enter the game or stay out of the game. Denote the players by $1, \dots, N$. The decision of the player j is denote by $x(j)$, where

$$x(j) = \begin{cases} 1 & \text{if } S(j) \text{ enters the game.} \\ 0 & \text{if } S(j) \text{ stays out of the game.} \end{cases}$$

If no nodes decide to enter the market, no cluster head is selected and all players get a payoff of 0. If a player decides not to enter the market while some other player has decided to enter, the first player's residual energy is its original residual energy at the beginning of the round, plus the energy it gains over the course of one round charging its battery. If a player decides to enter the market, its residual energy is the energy it had at the beginning of the round minus the energy consumed by sending a message.

Using the residual energy of players as a payoff function we obtain the following utility function

$$U(j) = \begin{cases} g(j) + f(j) & \text{if } x(j) = 0 \text{ and } \exists k \quad x(k) = 1 \\ 0 & \text{if } x(k) = 0 \text{ for all } k \in [N] \\ g(j) - C(j) & \text{if } x(j) = 1 \end{cases},$$

where $C(j)$, the cost function, is the total energy consumed by j to send a message, the gain function $g(j)$ is the node j's residual energy and $f(j)$ is the energy harvested to recharge the sensor's battery in the case it doesn't participate.

Using this framework, we can analyze the Nash Equilibrium of this game to see the probability of each member to enter the game and become a cluster head. This analysis, as presented in [13], shows the arrival at a stable state where one node volunteers to become cluster head.

IV. Adapting Clustering to Contact Graph Routing

In this section we adapt the three clustering algorithms introduced in section III to the case of Contact Graphs and show how to use them together. The first issue we face with doing so is that all three of the algorithms run on unweighted, undirected graphs. We need different approaches to address these discrepancies for each algorithm.

A. First Approach - Cooperative Game Based Clustering

In approach III-A the algorithms calls for shortest path distances between all pairs of vertices in the graph. In the Contact Multigraph we can instead compute the earliest arrival time. The earliest arrival time from node A to node B, $t_{ARR}(A,B)$, is defined to be the earliest time at which data can reach the node B when starting at node A at the current time. This value needs to consider changes in link availability over time and propagation delays. In [7] a variation of Dijkstra's algorithm is presented which allows an efficient computation of this value.

Using the earliest arrival time we can define a notion of distance between two nodes. Define the *arrival distance* between nodes A, B as,

$$d_{ARR}(A,B) = t_{ARR}(A,B) - t_0.$$

Where t_0 denotes the current time.

Using these distances we can define an all-pair-shortest-path matrix D,

$$D[A,B] = d_{ARR}(A,B).$$

We can now use Algorithm 1 as presented in section III-A. However, this algorithm requires the knowledge of the entire network to allow for the computation of the matrix D. It can therefore only be used when such global knowledge exists, for instance during the construction of the contact plan.

B. Second Approach - Dynamic Local Clustering

In approach III-B the algorithm uses internal and external degrees of clusters. To adapt these parameters to the Contact Multigraph we define a weight function on each pair of nodes. First we define a similar notion of arrival time which only considers paths of one step, namely the earliest arrival time along a single edge between two nodes.

$$\tilde{d}_{ARR}(A,B) = \begin{cases} d_{ARR}(A,B) & \text{if } \exists C_{A,B}^{t_1,t_2} \;\; s.t. \;\; t_2 > t_0 \\ \infty & \text{else} \end{cases}.$$

Where t_0 once again denotes the current time.

This value can be computed faster than the general d_{ARR} as it only requires scanning the edges between A and B. Using this distance, we define a weight function:

$$w(A,B) := \frac{1}{\tilde{d}_{ARR}(A,B)}.$$

Using the weight function we can define a notion of internal and external degree. Given a multigraph with nodes V and a cluster $C \subseteq V$, define its internal and external degrees as:

$$\deg_{\text{int}}(C) := \sum_{(u,v)\in C\times C} w(u,v).$$

$$\deg_{\text{ext}}(C) := \sum_{(u,v)\in C\times V\setminus C \cup V\setminus C\times C} w(u,v).$$

Note that this definition extends the definition of internal and external degrees introduced in section III-B, by defining the weight function as $\frac{1}{2}$ in the unweighted undirected case.

C. Third Approach - Non-Cooperative Game Based Cluster Head Selection

Finally, approach III-C can be used as presented in [13] by considering the residual energy of all nodes of the network. Formalizing the measurement of residual energy along with the cost of participating in the network in CGR is left to future work.

D. Combining All Three Approaches

We can now propose a clustering algorithm combining the three approaches. In our contact multigraph a cluster is a set of nodes consisting of a cluster head and cluster members. We use the various approaches during the different stages of clustering: initialization, new nodes joining a cluster, cluster maintenance (merging or splitting clusters, reassigning a node from one cluster to another) and election of new cluster heads.

1) Initialization: we can use approach III-A to initialize clusters and cluster heads. Since this requires knowledge of the entire network it can be done while computing contact plans and the assigned cluster heads and members can be distributed along with the contact plan.
2) Joining clusters: we can use approach III-B to allow nodes to choose a cluster to belong to when they first join the network.
3) Maintaining clusters: cluster heads will be responsible for maintaining clusters. Since edge weights change over time, the contribution of every node to the internal and external degree changes. The cluster head will ask for periodic updates and can use these to make decisions about splitting/merging clusters. Nodes can also choose to switch clusters as the network changes. We use approach III-B to make individual cluster membership decisions.
4) Electing a Cluster Head: as clusters change (merge, lose cluster head, periodically change cluster head to conserve energy) we can use approach III-C to reelect a new cluster head.
5) Dividing Big Clusters: as clusters get too big we can use approach III-A to split them up into smaller clusters, using the knowledge of the entire cluster.

In the following section we propose a clustering protocol that uses this algorithm on the contact multigraph. This protocol can be implemented and integrated with CGR and other DTN protocols.

V. Proposed Protocol: Delay Tolerant Clustering Protocol (DTCP)

After considering three approaches to clustering, their adaptation to CGR and their combination together, we can now propose a clustering protocol for CGR. In this section we outline the Delay Tolerant Clustering Protocol (DTCP), the data each member of the network must maintain and the packets communicated between different nodes at various stages of the clustering process.

DTCP packets maintain and adapt clusters throughout the network. A node in the network can be either a cluster head or a cluster member. When clusters change, cluster heads update each other on membership. The information is therefore available when a member of the cluster wants to send a packet outside the cluster. They query the cluster head to learn which cluster the destination belongs to and route to it. While the inter- and intra- cluster routing algorithms are beyond the scope of this work, a simple way to do this is by maintaining a partial Contact Multigraph. For a given node belonging to some cluster C, this partial graph contains only the edges within the cluster C or between clusters. It abstracts every cluster other than C into a single node in the graph, and knowing the cluster of its destination, routes its data to the cluster. When the packet leaves the cluster C, the path is recomputed in the next cluster in a similar way.

In order to participate in DTCP, all nodes must maintain some information regarding the cluster. Cluster members need to maintain the ID of the cluster they belong to and its cluster head, to allow them to query the head for any information they do not hold themselves. They also maintain three parameters regarding the structure: size, internal degree and external degree. This allows them to provide this information to new nodes joining the network while they are deciding which cluster to join. Cluster heads maintain the additional information of global cluster membership and a list of cluster members and their contribution to the total cluster internal and external degree.

Cluster heads send out a periodic *CLUSTER INFO* packet containing the cluster ID, cluster head, cluster size and cluster internal and external degree. These information packets are broadcast throughout the cluster and to neighboring clusters, in which they are sent only to the cluster head. Cluster members can also send such packets when prompted, containing the same information. When a new node joins the network it can send a *CLUSTER QUERY* packet, prompting its neighbors to send a *CLUSTER INFO* packet. Using the information from whichever *CLUSTER INFO* packets reach it before a certain timeout period, the node computes the cluster it wants to join. After doing so, it sends the appropriate cluster head a *CLUSTER JOIN* packet, announcing its request to join the cluster and a *CLUSTER UPDATE* packet, reporting its contribution to the total internal and external degree. The *CLUSTER JOIN* packet is broadcast through the cluster to announce the new member to all existing members.

To maintain the cluster structure, nodes in the cluster send out periodic *CLUSTER UPDATE* packets, reporting to the cluster head that node's contribution to the total internal and external degree (as this contribution changes over time). The cluster head uses this information to maintain and update its list of nodes and their contributions. When a node doesn't send an update packet for long enough, it is removed from the cluster. In this case the cluster head send a *CLUSTER LEAVE* packet, announcing the change to the cluster members.

Periodically, a cluster head can decide to merge with a neighboring cluster - clusters C_1 and C_2 will join if $f(C_1) + f(C_2) < f(C_1 \cup C_2)$. In this case we use approach III-C to select the new cluster head, as detailed in section IV. On the other hand, if a cluster becomes bigger than a set threshold, it splits up into smaller clusters using approach III-A. An individual node v can also decide to move from cluster C_1 to C_2 if

$$f(C_1 \setminus \{v\}) + f(C_2 \cup \{v\}) > f(C_1) + f(C_2).$$

In this case it will send a *CLUSTER LEAVE* packet to the head of C_1 and a *CLUSTER JOIN* packet to the head of C_2.

We can allow cluster head status to be either static (in the case of a relay node for instance) or dynamic. In the case of a cluster with no static cluster head, this role can be passed between different members. As this role requires spending more energy on communication, it might be transferred from one node to another after some time has elapsed. When a cluster head changes, the original cluster head sends a *CLUSTER INFO* packet with the new cluster head. To confirm, the new cluster head sends the same packet, and from this point on it acts as the cluster head. Using the periodic *CLUSTER UPDATE* packets, the new head learns the contribution of all members of the network.

In total, there are five types of DTCP packets:

- *CLUSTER INFO*: Sent periodically by a cluster head, or by a cluster member when prompted. Contains the cluster ID, cluster head, cluster size, cluster internal degree and external degree.
- *CLUSTER QUERY*: Sent by a node not associated to a cluster. Doesn't contain any information, prompts any node that receives it to send an info packet.
- *CLUSTER JOIN*: Sent by a node to announce joining a cluster. Contains the ID of the node and the cluster it joined.
- *CLUSTER LEAVE*: Sent by a node/cluster head to announce leaving a cluster. Contains the ID of the leaving node and the cluster it left.
- *CLUSTER UPDATE*: Sent by a cluster member periodically, containing its ID, the cluster it belongs to and its contribution to the internal and external degree.

VI. Simulation

To evaluate the proposed clustering algorithm, we implemented the logic behind approach III-B on a multigraph of satellite communication links. The graph was constructed using the python library *skyfield* [14], which provides the

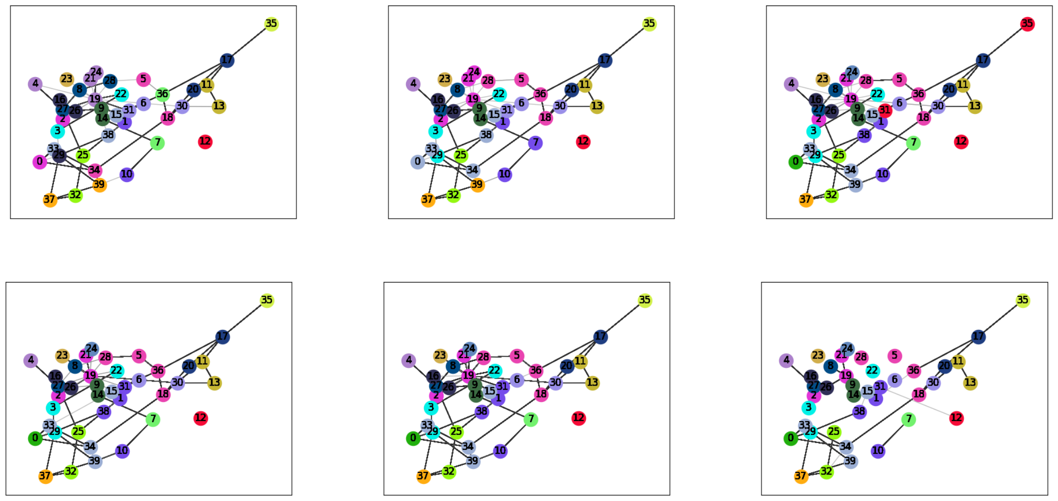
Fig. 2. Time Varying Clusters in a Satellite Line-of-Sight Network.

location of satellites orbiting the Earth. The edges in the graph were computed as the times in which two satellites have a line-of-sight connection to one another.

Preliminary results show intuitive cluster formations which remain relatively stable over time. The clusters are presented in Fig. 2. The six graphs represent the network at six different points in time. The black edges in each graph represent the connections available at that exact time and the grey edges represent connections that are expected to appear in the very near future.

VII. Implementation Considerations

Delay Tolerant Clustering Protocol, as the name implies is intended for delay tolerant networks. As, such it will require a DTN bundle agent to perform much of its base functionality. Future work will focus on developing a DTCP implementation for the High-rate Delay Tolerant Networking (HDTN) bundle agent [15]. HDTN supports store-and-forward capability, Licklider Transmission Protocol, Contact Graph Routing, and Contact Multigraph Routing. In addition to flight missions [16], HDTN also has a focus on several research areas relevant to DTCP, such SDN [17] and cognitive networks [18], [19].

A successful implementation of DTCP not only reduces the computational burden required by nodes to find paths within a contact graphed network, but it also enables DTN networks to have a reliable solution for scalable network management. DTCP's ability to determine cluster heads within a DTN helps solve the controller placement problem in software defined delay tolerant networks (SDDTN). SDDTN controllers would not only manage the membership of clusters with DTCP, but could be used to monitor network traffic and make network policy changes.

An essential mechanism for DTCP that HDTN currently lacks is a multicast capability. The DTCP protocol requires a multicast method to allow cluster heads to disseminate the *CLUSTER INFO* packet. New nodes joining the network will also require this capability in order to send *CLUSTER QUERY* packets. Interplanetary Multicast (IMC) [20] is a promising approach to multicast for delay tolerant networks and is planned for implementation in HDTN.

In addition to multicast, a standardized neighbor discovery method may be required or may provide part of the functionality of the *CLUSTER QUERY* packet. There are a proposed solutions for neighbor discovery, such as DTN IP Neighbor Discovery (IPND) [21], however it is still an open research area. Future work for this project is planned to evaluate similarities between the proposed cluster messaging packets discussed in this work and other discovery methods. Both IMC and neighbor discovery are planned for future versions of HDTN, which will facilitate the development of the DTCP implementation.

VIII. Conclusion

Space networking has come a long way in recent decades. However, to achieve the goal of a robust Solar Space Internet new tools are needed to build upon existing methods. In this work we propose a clustering protocol for Contact Graph Routing which can allow for greater scalability and better performance on large scale networks. We explore various clustering algorithms and adapt them to the context of Contact Graph and Contact Multigraph routing. We hope this will lead to further work into improving the performance of CGR for Delay Tolerant Networking and its scalability in particular.

Acknowledgements

The authors would like to thank Dr. Robert Short for helpful discussions on this project. We would also like to thank Matt Piekenbrock for help on running simulations.

References

[1] Kevin Fall. “A Delay-Tolerant Network Architecture for Challenged Internets”. In: *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. SIGCOMM ’03. Karlsruhe, Germany: Association for Computing Machinery, 2003, pp. 27–34. ISBN: 1581137354. DOI: 10.1145/863955.863960. URL: https://doi.org/10.1145/863955.863960.

[2] Sushant Jain, Kevin Fall, and Rabin Patra. “Routing in a Delay Tolerant Network”. In: *SIGCOMM Comput. Commun. Rev.* 34.4 (Aug. 2004), pp. 145–158. ISSN: 0146-4833. DOI: 10.1145/1030194.1015484. URL: https://doi.org/10.1145/1030194.1015484.

[3] Rémi Diana et al. “A DTN Routing Scheme for Quasi-Deterministic Networks with Application to LEO Satellites Topology”. In: Sept. 2012, pp. 1–5. DOI: 10.1109/VTCFall.2012.6399087.

[4] Juan A. Fraire, Olivier De Jonckère, and Scott C. Burleigh. “Routing in the Space Internet: A contact graph routing tutorial”. In: *Journal of Network and Computer Applications* 174 (2021), p. 102884. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2020.102884. URL: https://www.sciencedirect.com/science/article/pii/S1084804520303489.

[5] Cong Liu and Jie Wu. “Scalable Routing in Delay Tolerant Networks”. In: *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. MobiHoc ’07. Montreal, Quebec, Canada: Association for Computing Machinery, 2007, pp. 51–60. ISBN: 9781595936844. DOI: 10.1145/1288107.1288115. URL: https://doi.org/10.1145/1288107.1288115.

[6] P.G. Madoery et al. “Managing Routing Scalability in Space DTNs”. In: Dec. 2018, pp. 177–182. DOI: 10.1109/WiSEE.2018.8637324.

[7] Alan Hylton et al. “A Survey of Mathematical Structures for Lunar Networks”. In: *IEEE Aerospace Conference*. Institute of Electrical and Electronics Engineers. 2022.

[8] Jonathan Hamilton. “Game theory: Analysis of conflict, by Myerson, R. B., Cambridge: Harvard University Press”. In: *Managerial and Decision Economics* 13.4 (1992), pp. 369–369. DOI: https://doi.org/10.1002/mde.4090130412. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mde.4090130412. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mde.4090130412.

[9] Anna R. Karlin and Yuval Peres. “Game Theory, Alive”. In: (2016).

[10] Bala Prakasa Rao Killi, Ellore Akhil Reddy, and Seela Veerabhadreswara Rao. “Game Theory Based Network Partitioning Approaches for Controller Placement in SDN”. In: *Communication Systems and Networks*. Ed. by Subir Biswas et al. Cham: Springer International Publishing, 2019, pp. 245–267. ISBN: 978-3-030-10659-1.

[11] Leonard Kleinrock and Farouk Kamoun. “Hierarchical routing for large networks Performance evaluation and optimization”. In: *Computer Networks (1976)* 1.3 (1977), pp. 155–174. ISSN: 0376-5075. DOI: https://doi.org/10.1016/0376-5075(77)90002-2. URL: https://www.sciencedirect.com/science/article/pii/0376507577900022.

[12] P. Nikander et al. “Dynamic Local Clustering for Hierarchical Ad Hoc Networks”. In: *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*. Vol. 2. 2006, pp. 667–672. DOI: 10.1109/SAHCN.2006.288528.

[13] Sara Kassan, Jaafar Gaber, and Pascal Lorenz. “Game theory based distributed clustering approach to maximize wireless sensors network lifetime”. In: *Journal of Network and Computer Applications* 123 (2018), pp. 80–88. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2018.09.004. URL: https://www.sciencedirect.com/science/article/pii/S1084804518302911.

[14] Brandon Rhodes. *Skyfield: High precision research-grade positions for planets and Earth satellites generator*. Astrophysics Source Code Library, record ascl:1907.024. July 2019. ascl: 1907.024.

[15] NASA Glenn Research Center. *High-Rate Delay Tolerant Network*. URL: https://github.com/nasa/HDTN.

[16] D. Raible et al. *Developing High Performance Space Networking Capabilities for the International Space Station and Beyond*. Technical Memorandum. NASA, 2022. URL: https://ntrs.nasa.gov/api/citations/20220011407/downloads/TM-20220011407.pdf.

[17] Dominick Ta, Stephanie Booth, and Rachel Dudukovich. “Towards Software-Defined Delay Tolerant Networks”. In: *Network* 3.1 (2023), pp. 15–38. ISSN: 2673-8732. DOI: 10.3390/network3010002. URL: https://www.mdpi.com/2673-8732/3/1/2.

[18] Rachel Dudukovich et al. “A Distributed Approach to High-Rate Delay Tolerant Networking Within a Virtualized Environment”. In: *2021 IEEE Cognitive Communications for Aerospace Applications Workshop (CCAAW)*. 2021, pp. 1–5. DOI: 10.1109/CCAAW50069.2021.9527297.

[19] Rachel Dudukovich et al. “Toward the Development of a Multi-Agent Cognitive Networking System for the Lunar Environment”. In: *IEEE Journal of Radio Frequency Identification* 6 (2022), pp. 269–283. DOI: 10.1109/JRFID.2022.3162952.

[20] S. Burleigh. *CBHE-Compatible Bundle Multicast*. URL: https://datatracker.ietf.org/doc/html/draft-burleigh-dtnrg-imc-00.

[21] D. Ellard, R. Altmann, and A. Glad. *DTN IP Neighbor Discovery (IPND)*. URL: https://datatracker.ietf.org/doc/draft-irtf-dtnrg-ipnd/.