Ad Hoc Networks 80 (2018) 31–40

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

On route table computation strategies in Delay-Tolerant Satellite Networks

Juan A. Fraire[a,*], Pablo G. Madoery[a], Amir Charif[b], Jorge M. Finochietto[a]

[a] IDIT, Universidad Nacional de Córdoba, CONICET, Córdoba, Argentina
[b] Computing and Design Environment Laboratory, CEA LIST, Gif-sur-Yvette, F-91191, France

ARTICLE INFO

Article history:
Received 1 December 2017
Revised 7 May 2018
Accepted 2 July 2018
Available online 3 July 2018

Keywords:
Delay-Tolerant Satellite Networks
Contact Graph Routing
Schedule Aware Bundle Routing

ABSTRACT

Delay-Tolerant Networking (DTN) has been proposed for satellite networks with no expectation of continuous or instantaneous end-to-end connectivity, which are known as Delay-Tolerant Satellite Networks (DTSNs). Path computation over large and highly-dynamic yet predictable topologies of such networks requires complex algorithms such as Contact Graph Routing (CGR) to calculate route tables, which can become extremely large and limit forwarding performance if all possible routes are considered. In this work, we discuss these issues in the context of CGR and propose alternatives to the existing route computation scheme: *first-ending, first-depleted, one-route*, and *per-neighbor* strategies. Simulation results over realistic DTSN constellation scenarios show that network flow metrics and overall calculation effort can be significantly improved by adopting these novel route table computation strategies.

1. Introduction

There is an increasing interest of the space community (commercial, civilian and military) in deploying large-scale satellite networks with the purpose of providing high quality imagery, video and communication services [1,2]. This trend has pushed for more efficient space-terrestrial communication techniques and technologies capable of successfully moving large volumes of data between space and ground networks. In this context, Delay- Tolerant Networking (DTN) has been identified as a disruptive approach which can meet this goal in a cost-effective way by means of loose communication requirements.

Inspired in deep-space applications, the DTN architecture [3] assumes no upper bound on the signal propagation delay nor an expectation of a continuous nor bidirectional end-to-end connectivity through the network. This certainly differs from traditional Internet-based networks where end-to-end connectivity is generally assumed stable enough to pass data from source to destination. Since no instantaneous end-to-end feedback can be assumed, data might be temporarily stored (i.e., delayed) at intermediate nodes [4,5]. To determine when to store or when to forward data to a given neighbor, existing DTN forwarding schemes have sought to acquire the best possible knowledge of the network [6]. When applied in space, episodes of communications in DTN can be precisely computed in advance based on orbital elements. Also, power-conserving spacecrafts may communicate on infrequent, fixed intervals established by configuration. In any case, forthcoming episodes of communications (a.k.a. *contacts*) are typically scheduled weeks or months before they occur and can be imprinted in a *contact plan*. The resulting contact plan can be either distributed in advance to DTN nodes, or used by a centralized node (i.e., mission control) to execute route determination procedures.

A DTN paradigm can indeed be used to forward data on near-Earth satellite networks with sporadic satellite-to-satellite and satellite-to-ground communication opportunities. If so, we define them as Delay-Tolerant Satellite Networks (DTSNs). DTSNs differ from other space DTNs in the size of the topology and the speed at which it changes. In particular, interplanetary networks are rather scarce in terms of spacecrafts as a few rovers on a remote planet plus some orbiters are typically assumed in the literature [7]. While this density of deep-space nodes is unlikely to change in the near future, DTSNs topologies are expected to be promptly based on dozens or even hundreds of satellites [1]. Furthermore, while in interplanetary DTNs the topological changes are dictated by planetary dynamics, communication opportunities in DTSNs typically occur much more frequently between satellites in Low-Earth Orbit (LEO). As a result, the scalability limits of current DTN protocols and algorithms are likely to be met sooner in DTSN than in deep-space applications. Thus, DTSNs become an immediate object of study for evaluating efficient routing strategies which will also, in the long term, be valuable in the interplanetary domain.

* Corresponding author.
E-mail addresses: juanfraire@unc.edu.ar, jfraire@sti-tech.com.ar (J.A. Fraire).

https://doi.org/10.1016/j.adhoc.2018.07.002
1570-8705/© 2018 Elsevier B.V. All rights reserved.

In order to run traditional network algorithms in DTNs, previous studies sought to derive suitable graph structures out of contact plans. Initial approaches translated the contact plan to *time-expanded graphs* [8]. In a time-expanded graph, topological changes are modeled by a succession of graphs each representing the connectivity of the whole network during an interval where it is considered stable. Although convenient to calculate all-to-all paths, algorithms based on such data structure scale poorly with time (require more graphs) and nodes quantity (require more nodes in all graphs). A not so intuitive, yet more efficient modeling was proposed under the name of *contact graph*, a suitable structure that facilitates the distributed execution of adapted Dijkstra's searches [9]. The convenience of contact graph models motivated the study [7], implementation [10] and flight-validation [11] of early versions of distributed Contact Graph Routing (CGR) algorithms. Also, a reference version of CGR has been seamlessly integrated with Bundle Protocol [12] (DTN standard protocol) in the Interplanetary Overlay Network (ION) stack developed by JPL (NASA) [10]. ION is currently on version 3.6 and is operative in the International Space Station [13], and its CGR statement is being considered for the Schedule Aware Bundle Routing (SABR) recommendation book at the Consultative Committee for Space Data Systems (CCSDS) [14].

In spite of the increasing interest of the space community in CGR route determination algorithms [15], only recently there has been attention to the computation strategies that are used in CGR to populate its route table [16]. Indeed, since the route table computation method is responsible for triggering and setting subsequent CGR calls parameters (i.e., current network status), it has a direct impact on the final network flow and overall route processing effort. Previous works have sought to construct static route tables (left unchanged throughout the contact plan duration) similarly to those used in Internet [10]. As a first contribution, this work analyzes existing techniques, discusses their weaknesses and introduces *first-ending* and *first-depleted* which improve the construction of complete and accurate route tables. Next, we discuss that different approaches can be proposed to cope and to better adapt to the large and rapidly changing topologies of DTSNs. To validate this hypothesis, we also introduce *one-route* and *per-neighbor* methods which are presented as novel alternatives to dynamically update route tables on-demand, minimizing calculation effort without sacrificing data-forwarding efficiency. Finally, the resulting performance of each of these techniques are compared with the reference CGR implementation in ION v3.6 and analyzed by means of simulations over two appealing Low-Earth-Orbit (LEO) DTSN constellations. At the time of writing, the solutions explored in this paper are being rolled out in ION v3.7 and included in CCSDS's SABR specification.

This paper is structured as follows. In Section 2 relevant CGR concepts are defined and overviewed. Section 3 describes current route table calculation limitations and propose novel methodologies. Performance analysis by means of simulations are provided in Section 4 and discussed in Section 5. Final conclusions are drawn in Section 6.

2. Contact Graph Routing

Routing in DTNs is typically divided in three stages: planning (performed by mission control on ground), routing (typically distributed but can be centralized on ground as well) and forwarding (executed locally by each DTN node). In the planning stage, contact plans are determined based on the estimation of future episodes of communications. This task involves the physical disposition and orientation of nodes as well as their communication system configuration (antenna, modulation, transmission power, etc.). As a result, orbital propagators and communication models are combined to determine the final contact plan which can be further tuned to reduce energy consumption or remove conflicting contacts [17]. Whether kept in a centralized planning node or distributed to all nodes, the contact plan is then used by algorithms such as CGR to derive efficient routes. The forwarding process is then responsible for selecting the best route, out of many available on the route table, when local or in-transit data need to be queued for transmission. In DTN, data units are known as bundles. Then, a bundle can be either instantaneously transmitted or enqueued until a contact with the next hop node takes place.

CGR takes as input a contact plan, a data structure comprised of a series of contacts. A contact $C_{A,B}^{t_1,t_2}$ is defined as a time interval $(t_1; t_2)$ during which it is expected that data will be transmitted by node A (the contact's sending node) and received by node B (the contact's receiving node)[1]. In Fig. 1(a), each contact is identified by a number (#1 ... 16) and characterized by its start time, its end time, the identities of the sending and receiving nodes, and the rate at which data is expected to be transmitted by the sending node throughout the indicated time period. Furthermore, each contact is characterized by an approximate range value between nodes A and B expressed in light seconds. Since contacts are unidirectional, a pair is needed to describe a bidirectional link. In Fig. 1, contacts $C_{A,B}^{0,60}$, $C_{B,C}^{0,60}$ and $C_{A,C}^{0,60}$ represent permanent links (e.g., between ground stations connected through Internet). Contacts $C_{C,D}^{0,30}$ and $C_{A,E}^{10,20}$ might stand for sporadic Ground to Space Links (GSLs) while $C_{D,E}^{0,10}$, $C_{D,E}^{30,40}$ and $C_{D,E}^{50,60}$ for opportunistic Inter-Satellite Links (ISLs). The data volume of a contact is given by the product of its duration and its data transmission rate. Thus, a contact plan captures the time-evolving nature of a dynamic topology which can also be presented in a static graph as in Fig. 1(b) or in a time line view as in Fig. 1(c).

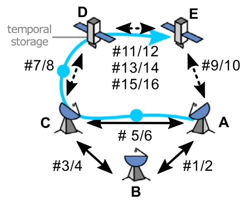
To compute data paths, CGR creates a contact graph model based on the contact plan. Specifically, a contact graph for destination node D at source node S is a conceptual directed acyclic graph $CG_S^D = (V, E)$ where vertices V correspond to contacts $C_{A,B}^{t_1,t_2}$ in the contact plan while edges E can be seen as episodes of data retention at a node i. Fig. 2 illustrates the CG_A^E based on the contact plan example of Fig. 1. Indeed, the structure of the contact graph may seem somewhat counterintuitive as it bears almost no relation to the topology of the network as illustrated in Fig. 1(b). In return, this static graph representation facilitates the execution of network algorithms over time-evolving networks. Specifically, a contact graph is formed by one vertex for each contact in the contact plan that signifies transmission either directly or indirectly (i.e., through other contacts) from A to node E. Edges are then added between contacts where destination and source nodes correspond (i.e., the receiving node of a contact matches the source node of the next contact in the path). In the example of Fig. 2, the receiving node of contact 1 ($C_{A,B}^{0,60}$) is the same as the transmission node of contact 3 ($C_{B,C}^{0,60}$). An edge between them represents a temporal storage in the connecting node which could be 0 when contacts are overlapped in time, meaning a direct transmission is possible. Finally, notional contacts from node A to itself and from node E to itself (a.k.a. root and terminal contacts) are also included as part of the contact graph. As discussed in [7] and detailed in [9], adapted Dijkstra's searches can be used to determine optimal routes over contact graphs.

Reference CGR implementation in ION 3.6 stores resulting routes in *route tables*. A routing table is a list of *route lists*, one route list for every other node in the network that is cited in any contact in the contact plan. A route list can be of size 0 (no routes

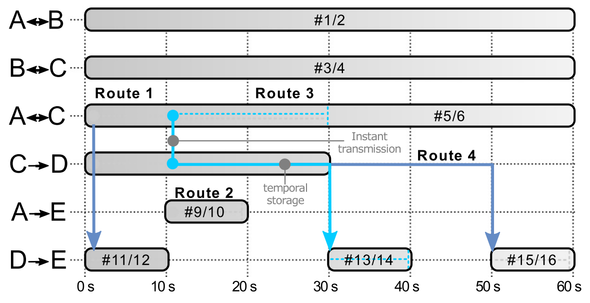
[1] A contact is specifically not an episode of activity on a link. Episodes of activity on different links (e.g., different radio transponders operating on the same spacecraft) may well overlap, but contacts by definition cannot. Therefore, all concurrent links should be considered together in a single contact.

#	frm/to	to/frm	ini	end	rate	range
1/2	A	B	0	60	1	0
3/4	B	C	0	60	1	0
5/6	A	C	0	60	1	0
7/8	C	D	0	30	1	0
9/10	A	E	10	20	1	0
11/12	D	E	0	10	1	0
13/14	D	E	30	40	1	0
15/16	D	E	50	60	1	0

a)



b)



c)

Fig. 1. (a) Contact plan example, (b) topology, and (c) time line view.

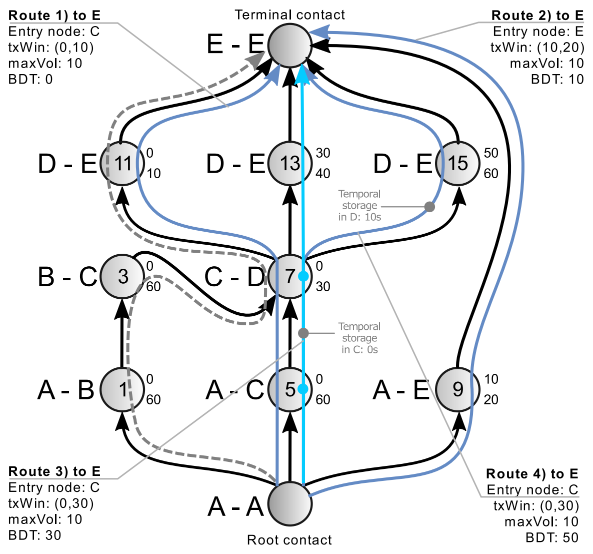


Fig. 2. Contact graph CG_A^E.

for destination) up to all possible routes. A route R_A^E for a bundle whose current location is node A and whose destination is node E is defined as a sequence of contacts such that (a) the sending node for the first contact is A, (b) the receiving node for the last contact is E, (c) the receiving node for contact i is the sending node for contact $i+1$, and (d) the time at which contact $i+1$ ends is no earlier than the time at which contact i begins. In the example of Figs. 1 and 2, a possible route 3, $R_A^E = \{C_{A,C}^{0,60}, C_{C,D}^{0,30}, C_{D,E}^{30,40}\}$, is highlighted. Since DTN assumes no persistent end-to-end connectivity, a route might require of temporal storage at intermediate nodes (e.g., node D), which can also be bounded and considered during the contact graph exploration. The receiving node for the first contact of a route is termed the route's entry node (e.g., node C), and each route is characterized by a *best-case delivery time* (BDT), a *maximum volume* (maxVol), and a valid *transmission window* (txWin) defined by the earliest and latest transmission time. These parameters can be consulted on forwarding time to determine the appropriate outbound queue for local or in-transit bundles.

3. Route table computation

Although the utilization of route tables for CGR is considered mandatory to avoid repeating Dijkstra calculations for each bundle, there has been no comprehensive discussion on how they should be populated, utilized and updated as the DTSN topology evolves through time. This section discusses this non-evident matter with significant impact on the final network performance and calculation effort. The only related previous work is the CGR reference implementation [10] in ION 3.6 (henceforth refereed as *current CGR*), which executes several Dijkstra's searches in order to populate a route list with all paths in a contact graph [18]. Indeed, such table is considered *static* throughout the duration of the contact plan. By suppressing certain contacts in the construction phase, each search for destination D on local node i's CG_i^D returns a new valid route R_i^D and is added to the route list to D. As a result, static routing table in current CGR is populated from scratch each time the contact plan is updated. However, this approach has a series of limitations as discussed next.

3.1. Current limitations

Route table construction. In order to avoid finding the same route in successive searches, current CGR suppresses the initial (i.e., first) contact ($C_{i,x}$) of the previously found route, ending the routine when no further paths can be found. For example, route 1 in Fig. 2 (the best route with BDT=0 and 3 hops), would derive in the suppression of contact 5 ($C_{A,C}^{0,60}$), excluding it from further searches. However, the suppression of the initial contacts incurs in a first evident issue when in presence of long lasting contacts with many possible outgoing paths. This situation is frequently seen when a satellite can be sporadically reached via a ground station connected to Internet (e.g., B reaching D through a long contact 3, $C_{B,C}^{0,60}$, in Fig. 1(b)). Indeed, if suppressing $C_{B,C}^{0,60}$, only one route between B and D through C will be discovered, although many might exist. An anchoring mechanism (implemented in current CGR and discussed in [18]) allowed to partially overcome this limitation. However, anchoring (a) only solves the problem for the first contact in the path and (b) it can end prematurely when a better path is found through a different initial contact. In the example of Fig. 2, the discovery of the first best route 1 $R_A^E = \{C_{A,C}^{0,60}, C_{C,D}^{0,30}, C_{D,E}^{0,10}\}$ with BDT=0, would trigger an anchor search on initial contact $C_{A,C}^{0,60}$. This is correct since other paths (routes 3 and 4) using $C_{D,E}^{30,40}$ and $C_{D,E}^{40,50}$ through this same initial contact are also valid. However, since the second best route $R_A^E = \{C_{A,E}^{10,20}\}$, with a BDT=10, has a different initial contact, it would prematurely end the anchoring[2], suppressing $C_{A,C}^{0,60}$ and hindering the discovery of valid routes 3 and 4. As the txWin of route 1, 3 and 4 are (0,10), (0,30) and (0,30) respectively, missing the last two would leave route list to E in node A without valid routes from 20 onwards. Although route 4 has the same txWin than route 3, its discovery would allow to consider 10 extra volume units to destination E. Indeed, the maximum volume

[2] In fact, the second route that current CGR (distributed with ION v3.6) will find after ending the first anchor is $R_A^E = \{C_{A,B}^{0,60}, C_{B,C}^{10,60}, C_{C,D}^{0,30}, C_{D,E}^{0,10}\}$ (illustrated with dashed lines in Fig. 2). However, this path overlaps and conflicts with route 1 volume resources, particularly with the volume-limiting contact $C_{D,E}^{0,10}$, which volume can either be used for the first or second route, but never for both. Volume sharing mechanisms or fail-safe calculations might make use of these redundant paths, but they are not in the scope of the present analysis. Strictly speaking, route 1 is considered a better path given its lower hop count. In the next step, CGR would anchor again in $C_{A,B}^{0,60}$ to derive in a second premature ending of the anchor search to finally find route 2.

that node A can allocate for E is 40 (10 units per each route 1 – 4), 30 units can be delivered to node C via $C^{0,60}_{A,C}$, and 10 can directly reach node E via $C^{10,20}_{A,E}$. Remarkably, all this information required to execute a successful route discovery is already in the contact graph, but its misuse has gone unnoticed in current CGR statement.

Route table size. In order to avoid unnecessary calculations, static route lists in current CGR are computed at once when a bundle is dispatched to the corresponding destination D. This construction takes place once per contact plan update (even if it is a minimum update respect the existing version). However, the series of Dijkstra's searches are concentrated in time and their associated computational effort will evidently depend on the contact graph connectivity and size. Previous works [19–21] have already characterized and verified the exponential growth of the computational effort involved by this CGR approach. In particular, simulation experiences detected that calculation time can rise to several hours in modern processors (significantly more powerful than those on a spacecraft), becoming intractable for contact graphs representing more than 3 h of propagation of a typical 16 satellite DTSN [20]. On the other hand, the bigger the obtained route list, the more costly the list exploration required at forwarding time [22]. Indeed, each route in the list must be filtered by txWin (those not matching with current time shall be disregarded) and compared based on BDT and maxVol properties. The latter is a critical point given that route lists are constructed on a contact plan update basis, but route list exploration happens a lot more frequently, once for each forwarded bundle. Furthermore, all these intractability effects would become even more dramatic if the route discovery issues previously identified were solved (i.e., more routes would be added to the resulting route list). Remarkably, by using the current route list population approach, a route list would be built with all possible routes to D, even though only a few of them might finally be effectively used to assist traffic forwarding.

3.2. Static route table calculation

In this section, we propose new strategies to improve the calculation and operation of route tables whose size remains constant throughout the contact plan period. Specifically, *first-ending* and *first-depleted* are introduced as static route table calculations alternatives to current CGR implemented in ION 3.6 described in the previous section.

First-ending. It is still based on a series of Dijkstra's searches, but instead of suppressing the initial contact, it suppresses the first ending contact of the last path found. In the example of Fig. 2, route 1 $R^E_A = \{C^{0,60}_{A,C}, C^{0,30}_{C,D}, C^{0,10}_{D,E}\}$ would trigger the suppression of $C^{0,10}_{D,E}$ instead of the initial $C^{0,60}_{A,C}$ from CG^E_A. The next search would return route 2 and suppress $C^{10,20}_{A,E}$, and the final search would provide route 3 suppressing $C^{0,30}_{C,D}$. Thus, route 3, overlooked by initial+anchor, would be discovered by first-ending approach. Indeed, first-ending emulates the evolution of time and for each instant, it provides the best route if any. In other words, the topological state kept throughout the route calculation in this approach is *time-based*. However, route 4 would still be undiscovered as it has the same txWin than route 3. To address this, we propose a second alternative based on traffic volume.

First-depleted. It suppresses the first contact whose volume would get fully booked if data were to flow through the path. In this case, after discovering route 1, contact $C^{0,10}_{D,E}$ would be the first contact to become depleted with 10 units of traffic and thus, removed

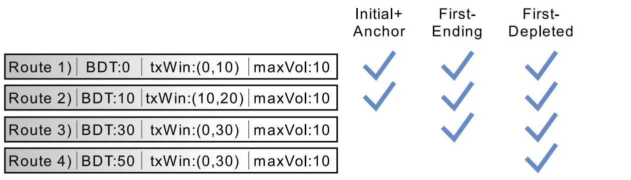

Initial+ Anchor
First- Ending
First- Depleted
Route 1) | BDT:0 | txWin:(0,10) | maxVol:10
Route 2) | BDT:10 | txWin:(10,20) | maxVol:10
Route 3) | BDT:30 | txWin:(0,30) | maxVol:10
Route 4) | BDT:50 | txWin:(0,30) | maxVol:10
✓ ✓ ✓
✓ ✓ ✓
✓ ✓
✓

Fig. 3. Route tables for initial+anchor (current CGR in ION 3.6), first-ending and first-depleted route construction approaches.

from CG^E_A. As with first-ending, route 2 would provoke the suppression of $C^{10,20}_{A,E}$, but route 3 would trigger the suppression of $C^{30,40}_{D,E}$. Therefore, route 4 would be provided as the last Dijkstra search result in the route list to node E. It is important to notice that the volume booking status in first-depleted must be stored along the successive searches. Indeed, the first-depleted route list construction approach emulates the volume utilization that a node would make in presence of continuous traffic to the destination. In this case, the topological state kept throughout the route calculation is *volume-based*. Fig. 3 summarizes the route list to E for each of these construction approaches[3].

3.3. On-demand route table calculation

Although the proposed static route table approaches seem to deliver a more comprehensive route list construction than initial+anchor, the route list size intractability is still not solved, in fact, the problem worsens. In this paper, we argue that the reason behind this is that route tables in DTSNs shall not be handled statically as in Internet, but in a dynamic fashion in order to keep up with the highly dynamic nature of time-evolving topology of DTSNs. To better adapt route tables to the DTSN paradigm, we propose to manage them in such a way that successive Dijkstra calls are executed only if required and on-demand. In other words, instead of populating a static and complete route list with all possible paths to a given destination D, we control and bound how many entries are calculated and update them as necessary. Although different route table sizes can be chosen, we focus on two approaches: *one-route* and *per-neighbor-route*. These strategies not only solves the problem of the calculation intractability for very large contact graphs but also bounds the route list exploration effort in forwarding time.

One-route. A first evident proposal is to consider a *one-route* approach allowing to have a single entry route list permanently holding the best route to D at any given time. Then, each time a bundle must be forwarded, the entry validity is verified and updated if (a) there is no route in the entry (initial condition of the table), (b) its txWin is due or (c) its maxVol was reached; otherwise, it is left unchanged. It is interesting to notice that conditions (b) and (c) resemble the network status kept during first-ending and first-depleted computation strategies (i.e., time and volume). In contrast with a complete route list population, the one-route approach drastically increases the computation efficiency as the outcome of every Dijkstra calculation is effectively used to assist in the forwarding of bundles.

[3] For the sake of truth, if all possible routes in a graph are to be discovered, a full search would be the most efficient approach; or if a fixed *k* number of best paths is required, existing K-Shortest Paths (KSP) algorithms such as Yen's [23] can provide efficient solutions. However, neither of the latter qualifies as valid CGR candidates because they keep no track of time nor consumed volume, providing conflicting paths as discussed in footnote [2] and illustrated in the route in dashed lines of Fig. 2. Nonetheless, studying KSP adaptations to be used over contact graphs might be a promising research line.

Per-neighbor-route. Of course, intermediate approaches are possible. Among them, a *per-neighbor-route* approach would allow to have N possible entries, each with a different entry node (first-hop) corresponding to each neighbor node *n* in the contact plan. In the example of Fig. 2, three different entry nodes are found in CG_A^E: B, C and E; each one with a corresponding best path towards E. In this approach, first contacts leading to different nodes than the corresponding *n* entry node shall be suppressed from the search when computing the *n* position in the route table. As a result, only *n* Dijkstra calls will be made to populate the table the first time, and a maximum of *n* would be calculated if updates are needed. Although route validation would have to be executed for every *n* entries in the list for each forwarded bundle, this procedure facilitates the rapid determination of alternative paths to a destination (i.e., for critical data which is expected to be forwarded through all possible paths to a destination).

3.4. Volume awareness

Although the improvements of first-ending, first-depleted, one-route, and per-neighbor approaches seem obvious in simple examples such as the one shown in Fig. 2, important considerations have to be taken regarding the volume booking assumptions when using distributed CGR in large-scale DTSNs.

All-contacts. In previous examples, we had assume that the CG_A^E model, used for successive CGR calculations, is node A's local view of the status of the forthcoming topology and its volume occupation. Since the volume of all contacts in CG_A^E are considered and updated, we will refer to this type of local volume awareness as *all-contacts*. Even though all nodes can share the same contact plan (timely and correctly provisioned in advance), there is no guarantee that the local volume booking (nor the order in which it was made) will be synchronized with the rest of the DTSN nodes. In fact, given that DTN protocols assumes no upper bound on the signal delay and no expectation of continuous connectivity, the opposite is more likely. However, this scheme might result convenient if there is a single node that injects most of the traffic [24].

1st-contact. To avoid making false assumptions, current CGR implementation in ION (initial+anchor) only tracks the volume consumed in the first contact of each path. Indeed, it is the only safe assumption to make since the real occupation can be directly measured from nodes' local buffer [25]. Henceforth, we will refer to this conservative and local volume awareness type as *1st-contact*. Nonetheless, such approach ignores maximum volume allocation bounds already encoded in the contact plan (i.e., even if the first contact is not fully-booked, others in the path could become overbooked by local traffic) [26].

Source-routing. On the other hand, more proactive schemes have sought to synchronize the local volume occupation view by means of informative feedback messages [27]. In general, such strategies were conveniently combined with source routing approaches (a.k.a. *extension-block* [28]) to accelerate information distribution and mitigate forwarding conflicts. However, these approaches have been validated in continuously connected networks as only marginal improvements are expected in highly disrupted DTSNs.

Whichever the volume awareness type, routing loops will still be a hazard between nodes with different or out of sync topological views. As a result, a *not-return-to-sender* policy has been implemented in ION to avoid unwanted routing loops[4]. In general, whether by a 1st-contact or all-contact with or without source-routing, analyzing the correctness and congestion-free delivery of data in DTSNs requires comprehensive and complex studies that consider several variables. Indeed, all these phenomena directly interact and articulate with the chosen route table computation methodology. For example, a correct utilization of a volume-based strategy such as first-depleted would depend on an accurate volume-awareness (i.e., all-contacts) during forwarding. Table 1 summarizes the route table calculation strategies and volume awareness types discussed in this section.

[4] This policy forced the one-route method to also include a backup route entry in case the best next-hop happens to be the sender of the bundle.

4. Simulation analysis

In this section we analyze the combined performance of each route table calculation with different volume awareness types by means of simulation. Obtained numerical results are compared with the performance of current CGR included in ION v3.6, classified as initial+anchor and 1st-contact in this paper. Thus, the results here plotted under the initial+anchor and 1st-contact labels are used as benchmarking for analyzing the advantages of the proposed CGR alternatives. It is worth mentioning that the benchmarking version of CGR includes its many enhancements described throughout the literature [15]. To understand the resulting network flow efficiency and calculation effort, we focus on the *delivery ratio* and *route table entries created* metrics respectively. Simulations are run for each of the methods in Table 1 under increasing traffic injection (more bundles of the same size are generated). DtnSim[5] is the chosen event-driven simulation platform as it eases the study of several runs [29].

The example scenario in Figs. 1 and 2, with a single traffic flow from node A to E, will be analyzed and contrasted with the claims made in Section 3. Next, two larger and realistic scenarios, with an all-to-all traffic pattern are proposed to study the behavior of suggested methodologies under challenging volume conditions discussed in Section 3.4. A walker-delta formation and a sun-synchronous along-track composed by 16 cross-linked LEO satellites (max. link range of 1000 Km at 500 Km height), 25 ground target points (e.g., user terminals), and 6 ground stations have previously been presented in [18] and reconsidered in this analysis[6]. In this case, 24 h of orbital propagation were encoded in contact plans with 3184 and 1512 contacts respectively.

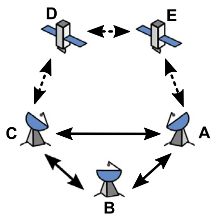
It is important to notice that the best-route criteria adopted in the route table exploration in these simulations are (1) best BDT, (2) minimum hop count and (3) higher residual volume. The Dijkstra search implemented in DtnSim is based in the one in ION which honors (1) and (2). However, DtnSim search keeps track of visited nodes while looping through the contact graph. If a visited contact leads to an already visited node it is ignored. Indeed, different contacts can connect to a same node, implying that a loop-free path in a contact graph does not necessarily is so in the original network topology. Route calculation accuracy in current CGR is expected to be improved by this consideration. Since this is not currently considered in CGR reference implementations, this Dijkstra modification will be suggested for future ION releases. Furthermore, to mimic the default ION stack configuration, DtnSim implementation does not affect BDT by local or remote buffer occupancy as discussed in [24]. Also, not-return-to-sender restriction is en-

[5] DtnSim is a publicly available simulator developed at the UNC, based in Omnet++ (C++) and specialized in DTSNs. The interested reader will find the documented implementation of all the algorithms and scenarios under analysis in the CgrModelRev17 class in the DtnSim public repository: https://bitbucket.org/lcd-unc-ar/dtnsim/.

[6] An intuitive visualization of the scenarios as well as the obtained contact plans can be found at https://sites.google.com/unc.edu.ar/dtsn-scenarios. Furthermore, the interested reader is referred to [18] for specifics on orbital parameters and ground locations.

Table 1
Route table calculation and volume awareness types.

Static route table calculation	
All-routes	Populates with all found routes for a destination.
Initial+anchor	Suppress initial+anchor on each search (benchmark CGR).
All-routes	Populates with all found routes for a destination.
First-ending	Suppress first-ending on each search.
All-routes	Populates with all found routes for a destination.
First-depleted	Suppress first-depleted on each search.
On-demand route table calculation	
One-route	Calculates 1 route for a destination.
	Replaces the entry when due or depleted.
Per-neighbor	Calculates N routes for a destination.
Route	Replaces the entries when due or depleted.
Volume awareness	
None	Volume is not considered nor annotated.
1st-contact	Decreases the volume of the 1st contact in path.
	Updates all affected routes in route table (benchmark CGR).
All-contacts	Decreases all contacts volume in the path.
	Updates all affected routes in route table.
All-contacts	All-contacts + extension-block source routing.
Source-routing-eb	Based on a local contact graph in each node.



Simulation parameters

Bundle size	2 Bytes
GSL data-rate	1 Byte/s
ISL data-rate	1 Byte/s
Source nodes	[A]
Sink nodes	[E]

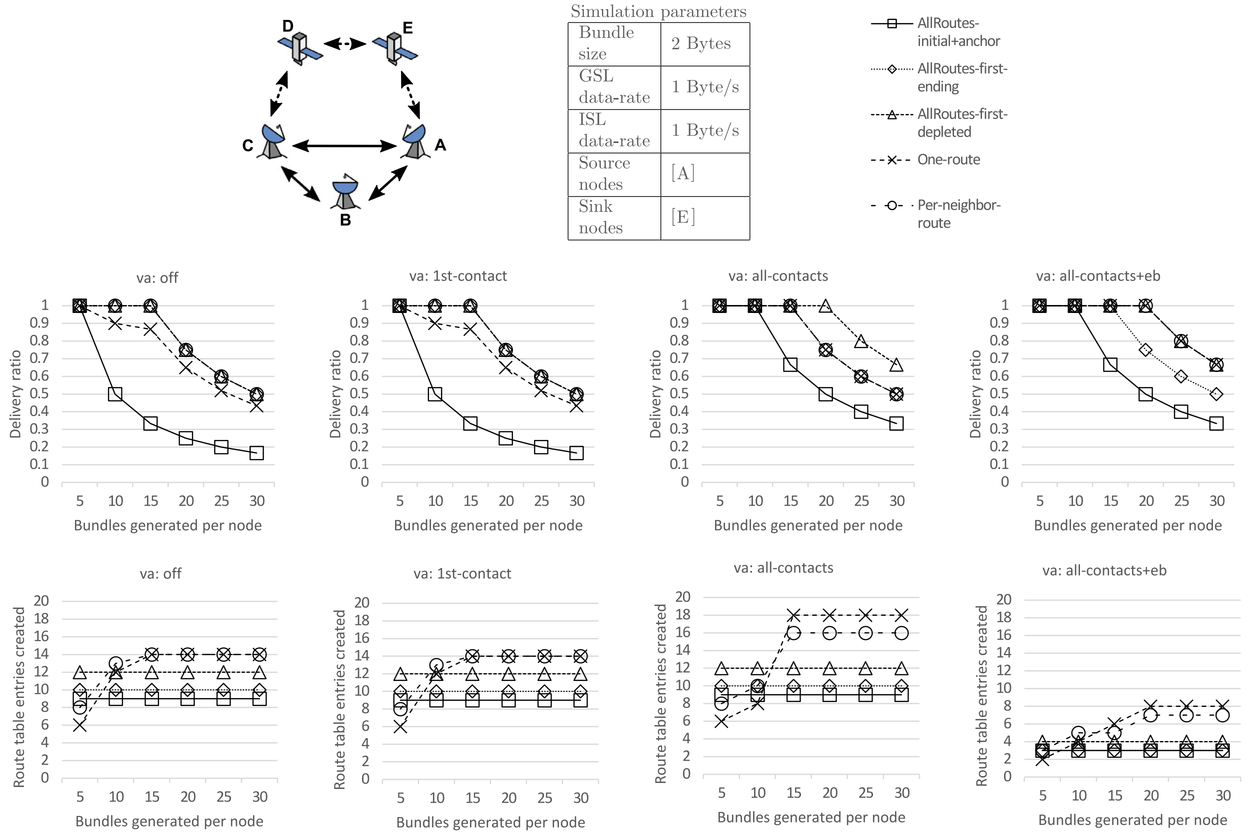


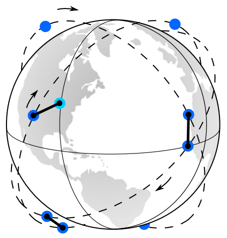
Fig. 4. Example scenario (in Figs. 1 and 2): delivery ratio and route entries created in route table.

abled as in ION, and all generated traffic is of the same bulk-type (i.e., non-prioritized).

4.1. Results and analysis

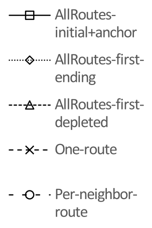
Delivery ratio and route table utilization metrics for the example scenario, walker and along-track formations are plotted in Figs. 4, 5 and 6 respectively. For each case, a different chart is offered for each of the four volume awareness types comparing how proposed route table calculation strategies perform.

Simulation results for the example topology in Fig. 4 corroborates the discussion in Section 3. In general, the delivery ratio improves with more complex volume awareness approaches. As previously argued, such schemes tend to operate correctly where a single node generates most of the traffic. Specifically, the first-depleted approach is able to deliver the maximum throughput (20 delivered bundles) by relying on an all-contacts volume knowledge. Other route table methods achieve such performance when using a source-routing approach. It is interesting to note that initial+anchor performance in benchmark CGR is always below the proposed alternatives. Regarding the total route table entries cre-



Simulation parameters

Bundle size	125 KBytes
GSL data-rate	12.5 KByte/s
ISL data-rate	12.5 KByte/s
Source nodes	[7..31] [48]
Sink nodes	[48] [7..31]



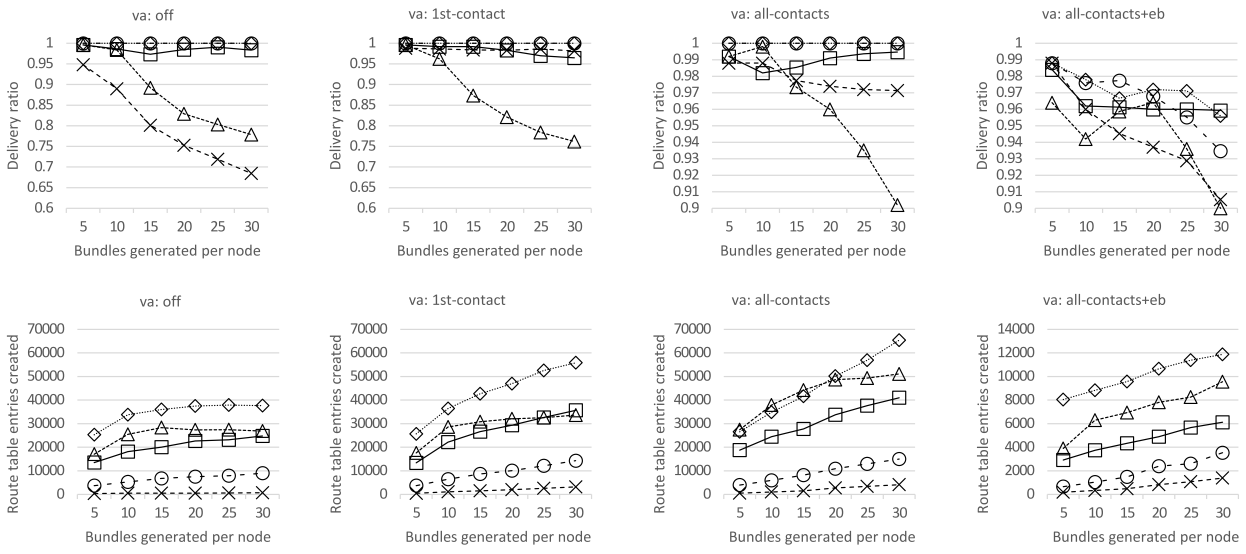
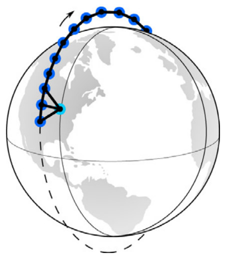
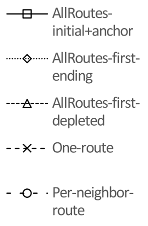


Fig. 5. Walker-formation scenario: delivery ratio and route entries created in route table.



Simulation parameters

Bundle size	125 KBytes
GSL data-rate	12.5 KByte/s
ISL data-rate	12.5 KByte/s
Source nodes	[7..31] [48]
Sink nodes	[48] [7..31]



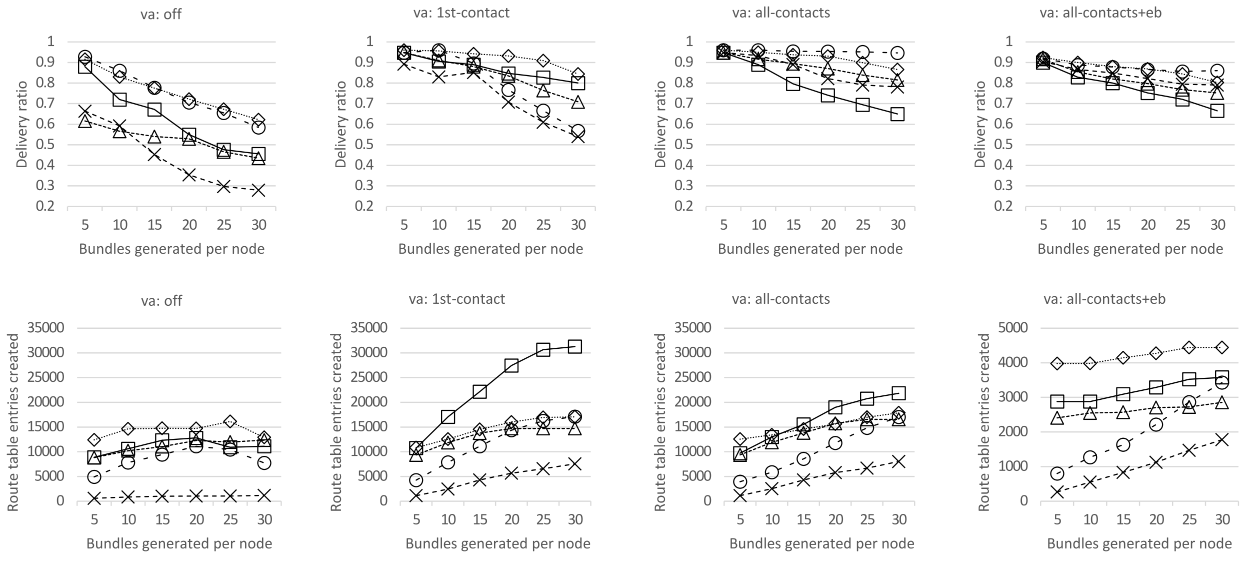


Fig. 6. Along-track scenario: delivery ratio and route entries created in route table.

ated (sum of all nodes), it was expected that one-route and per-neighbor strategies would increase their calculation effort with higher traffic demands. On the other hand, full table population strategies remain independent of traffic. Among them, and as anticipated in Section 3.2, first-depleted is able to find more routes, followed by first-ending and then initial+anchor. Evidently, source routing approaches significantly diminish route calculations.

Figs. 5 and 6 provide the same metrics but for realistic DTSNs topologies with stressing all-to-all traffic patterns. They also illustrate their orbital disposition and simulation parameters for further clarity. Two general conclusions can be drawn at a first glance. On the one hand, a delivery ratio of 1 is not always guaranteed even for low network loads for certain combination of route table calculation and dummyTXdummy- volume awareness. Different reasons, explained below, are behind this effect for the walker and more noticeably for the along-track formation. On the other hand, the quantity of routes found in a relatively moderate contact plan of a one day duration can be in the order of several tens of thousands. These measurements confirm and highlight the need of exploring on-demand route table computation approaches.

To analyze the particular case of the walker formation, we need to recall that such topology is highly disrupted as ISLs and GSLs always occur sporadically during bounded period of times (typically around 3 minutes for the configured orbits and communication ranges). As previously introduced in Section 3.4, the correct synchronization of each DTSN node local view of network-wide volume utilization is impossible to achieve in such scenarios even when using source routing extensions. This principle is clearly evidenced in the delivery ratio plotted in Fig. 5, where volume-sensitive first-depleted route table calculation falls behind other volume-agnostic strategies such as first-ending. On the other hand, one-route and per-neighbor on-demand methods perform well for all traffic injection rates. Furthermore, they significantly minimize calculated routes for all types of volume-awareness, which proves dynamic route table calculation techniques are very suitable for this kind of DTSN constellations. Static route table approaches such as first-depleted and first-ending generally double the quantity of discovered routes with respect to initial+anchor, improving forwarding decision quality at the expense of heavier calculation effort.

While walker formation is a highly disconnected constellation, the along-track stands in the opposite side as every satellite is permanently reachable by the immediate front and back neighbors. As a result, this system can be seen as a linear topology that sporadically connects with ground stations (also continuously connected among them) or target nodes. Moreover, when a ground node is reachable, several satellites might simultaneously establish GSLs connections forming temporal mesh topologies. Therefore, highly connected contact plans (less yet longer contacts) are the rule and not the exception in this kind of constellations. In spite of a high connectivity, curves in Fig. 6 suggest that achieving a complete data delivery is challenging regardless of the route table method and dummyTXdummy- volume knowledge assumption. In particular, first-ending approaches, followed by initial+anchor, provide the best results for limited volume information (1st-contact). One-route slightly outperforms other procedures for all-contacts and extension block strategies with ratios between 0.8 and 1. After a rigorous analysis of the simulation traces, it was found that the not-return-to-sender policy played a crucial role in hindering the correct flow of traffic. Naturally, in a linear topology, data might flow in one direction; but if a contact ahead in the path is congested, bundles might have to be sent back through the original path. Such restriction forbids a reaction towards fully-booked contacts, limiting data delivery. However, careful studies have to be made before disabling this policy, since two directly connected nodes with different views of the topology can easily run into harmful routing loops. Regarding route table entries, similar conclusions as with the walker constellation can be drawn. Yet, in this case, the initial+anchor approach provides higher number of routes than the theoretically better performing first-ending and first-depleted, particularly in the 1st-contact volume awareness. Nonetheless, we found that many of these extra routes evidenced the same problem discussed in footnote[2] of Section 3. They were redundant expressions of a same communication resource.

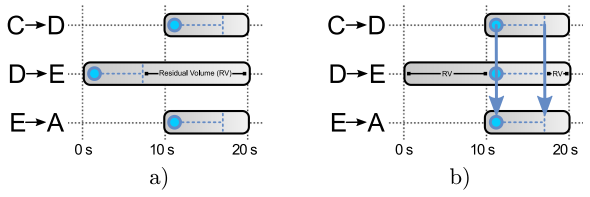


Fig. 7. (a) Simplistic volume consumption modeling and (b) real volume utilization for traffic flowing from C to A.

To wrap up the presented simulation analysis, we leave the reader with the following relevant highlights. Presented route table computation strategies have a direct and positive impact on the resulting network flow metrics when compared with current CGR statement in ION 3.6. However, the nature of the DTSN topology (strongly or loosely connected in orbit) is even more relevant and dictates how different DTN algorithms and strategies perform, suggesting that there is not a one-size-fits-all solution. For example, volume-aware route table calculation (e.g., first-depleted) relies on the correct annotation and synchronization of volume consumption, which efficiency in turn depends on the type of the underlying topology. Last but not least, extension-block source-routing has a predominant role in minimizing the route calculation effort; but it does not necessarily contribute to a better data delivery[7].

5. Discussion

Volume consumption. We have assumed that the updated residual volume of a contact is equal to the previous residual volume minus the bundle volume. However, this volume consumption model is rather simplistic as it assumes that the volume utilization of the contact always begins at the contact start time. Clearly, this is not the general case, as networks such as the along-track make uses of very long (continuous) contacts which volume might be booked in different time intervals within contact's txWin. Fig. 7 illustrates the difference between an erroneous estimation and the real utilization of volume for traffic flowing from C to A. As a result, volume calculation for these cases needs to contemplate a more realistic volume estimation. Studying the impact of this simplification, how they might have affected the results here discussed and study alternatives are left as future research.

Volume annotation. We have assumed that the queuing of a bundle implies an update of the residual capacity of the involved contacts (the initial contact in 1st-contact, and all contacts in the all-contacts case). However, all routes in the route table which happen to use any of the annotated contacts (all route lists to all possible destinations), would also need to be updated regarding its associated residual volume metric. While this can be optimized from an implementation perspective (i.e., pointer arithmetics), it might result a tremendous effort to be tackled in forwarding time (once

[7] It should be noticed that plain source routing techniques were simulated. More complex protocols such as the one presented in [27] consider the dissemination and fusion of information in feedback messages which might improve data delivery, but are out of scope of this analysis.

for every transmitted bundle). This fact certainly favors the selection of bounded-size route table calculation strategies such as one-route or per-neighbor-route.

Contact plan update. Throughout this paper we have assumed that a contact plan is a fixed data structure, while in fact, it is a dynamic model that can and needs to be updated and modified via configuration commands. In this regard, ION stack currently erases the complete route table by the minimum change in the contact plan. Naturally, previous calculations can no longer be considered accurate even if a single contact duration was modified. Since this likely circumstance might dissipate the present attempt of minimizing route entries calculations, the research on mechanisms that could efficiently update route tables (e.g., intelligently detect affected entries and perform a selective update) is highly encouraged.

Storage. Due to space limitations, the effect of memory utilization was not addressed in this paper. In particular, DTSNs nodes' storage was not measured and considered infinite in the simulations. Since complete data delivery was not always assured, the delivery ratio metric was kept in the main focus throughout the presented analysis. However, if not considered properly, the lack of local storage could become another source of congestion further degrading the DTSN performance. An extended analysis to study the memory utilization in congested DTSNs is left as future work.

Station keeping. We have assumed complex contact graphs obtained from LEO DTSNs with in-orbit communication opportunities. Maintaining a flight formation as those studied here will probably require some type of station keeping. This might be a weak point for DTSNs systems build upon Cubesats which tend to be rudimentary (i.e., lacking a propulsion system). However, this will likely change in the future with the development of small-form-factor electric propulsion. In addition, since CubeSats are generally launched as secondary payload from existing launch opportunities, the final orbital parameters might derive from a heterogeneous mix that would require a different topological analysis such as the one presented in [30].

Wireless links. It has been assumed that all wireless links can operate simultaneously and in a bidirectional configuration (i.e., data can be uploaded by ground nodes and transmitted via ISLs at the same time). This might not be the general case given the limited power and constrained hardware of satellite platforms, an issue which can be tackled as a contact plan design problem and is out of the scope of this article [17]. On the other hand, the Doppler frequency shift between satellites can be as large as 50 KHz when using the 2.4 GHz radio band. The ISL subsystem would need to be able to track received frequencies in this range.

On-board software. ION reference implementation is designed to operate on flight processors configured with real-time operating systems and substantial storage and processing resources; not all satellites are so equipped, particularly Cubesats. However, the Micro Planetary Communication Network (μPCN) software has been proposed as a lightweight Bundle Protocol implementation specifically designed to operate over small processors and microcontrollers [31]. The implementation of dynamic routing table calculation strategies here discussed could bridge the gap between these DTSN flight software. At the time of writing, the one-route and all-contact techniques are being specified as part of the SABR book at CCSDS [14] and are being officially rolled out as part of future ION v3.7. However, the reader will notice that SABR and ION also consider traffic priorities. Indeed, the solutions discussed in this paper have been further adapted to route prioritized traffic. This is currently achieved by including a per-traffic type volume modeling on each route entry and potentially adding as many entries as traffic types present. However, a formal analysis and evaluation on optimal approaches to prioritized route table management techniques is left as further work.

6. Conclusions

In this paper, we have explored, for the first time, efficient route table computation techniques for CGR-based DTSNs. Based on the perspective that the time-evolving nature of DTSNs requires of different route table calculation paradigms, several new strategies were introduced. By applying them in realistic DTSN scenarios, they were thoroughly evaluated in terms of delivery, performance and size efficiency under varying operating conditions and using different volume allocation awareness policies.

Results validated the starting hypothesis which stated that route tables completeness could be improved and that their entries could be dynamically managed on-demand to reduce the overall calculation effort. Although proposed methods improved delivery ratio metrics while significantly minimizing route table sizes, it was shown that the nature of the DTSN topology mandates which of them results a better strategy for each case. In other words, there does not seem to be a one-size-fits-all routing table computation procedure for DTSNs.

Acknowledgments

We want to thanks Scott C. Burleigh (Jet Propulsion Laboratory, NASA), ION chief architect, from his insightful contributions to this paper. Marius Feldmann and Felix Walter (Technische Universität Dresden) also supported this effort by validating the presented ideas from the μPCN implementation perspective. Part of this research was carried out in the frame of the "Calcul Paralléle pour Applications Critiques en Temps et Sûreté" (CAPACITES) project. This work was partially funded bygrant PICT 2015-1227 of the Science and Technology Ministry of Argentina.

References

[1] J. Alvarez, B. Walls, Constellations, clusters, and communication technology: Expanding small satellite access to space, in: 2016 IEEE Aerospace Conference, 2016, pp. 1–11, doi:10.1109/AERO.2016.7500896.
[2] M. Berioli, A. Molinaro, S. Morosi, S. Scalise, Aerospace communications for emergency applications, Proc. IEEE 99 (11) (2011) 1922–1938, doi:10.1109/JPROC.2011.2161737.
[3] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, H. Weiss, Delay-Tolerant Networking Architecture, RFC, RFC Editor 4838, 2007.
[4] K. Fall, A delay-tolerant network architecture for challenged internets, in: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, in: SIGCOMM '03, ACM, New York, NY, USA, 2003, pp. 27–34, doi:10.1145/863955.863960.
[5] A. McMahon, S. Farrell, Delay- and disruption-tolerant networking, IEEE Internet Comput. 13 (6) (2009) 82–87, doi:10.1109/MIC.2009.127.
[6] S. CC, V. Raychoudhury, G. Marfia, A. Singla, A survey of routing and data dissemination in delay tolerant networks, J. Netw. Comput. Appl. 67 (2016) 128–146.
[7] J. Segui, E. Jennings, S. Burleigh, Enhancing contact graph routing for delay tolerant space networking, in: Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE, 2011, pp. 1–6, doi:10.1109/GLOCOM.2011.6134460.
[8] S. Merugu, M. Ammar, Z. E., Routing in Space and Time in Networks with Predictable Mobility, Technical Report, Georgia Institute of Technology, 2004.
[9] E. Birrane, S. Burleigh, N. Kasch, Analysis of the contact graph routing algorithm: bounding interplanetary paths, Acta Astronaut. 75 (2012) 108–119. https://doi.org/10.1016/j.actaastro.2012.02.004.
[10] S. Burleigh, Interplanetary overlay network: An implementation of the dtn bundle protocol, in: 2007 4th IEEE Consumer Communications and Networking Conference, 2007, pp. 222–226, doi:10.1109/CCNC.2007.51.
[11] J. Wyatt, S. Burleigh, R. Jones, L. Torgerson, S. Wissler, Disruption tolerant networking flight validation experiment on nasa's epoxi mission, in: Advances in Satellite and Space Coms., 2009. SPACOMM 2009. First International Conference on, 2009, pp. 187–196, doi:10.1109/SPACOMM.2009.39.
[12] K. Scott, S. Burleigh, Bundle Protocol Specification, RFC, RFC Editor 5050, 2007.

[13] A. Jenkins, S. Kuzminsky, K. Gifford, R. Pitts, K. Nichols, Delay/ disruption-tolerant networking: Flight test results from the international space station, in: 2010 IEEE Aerospace Conf., 2010, pp. 1–8.
[14] Consultative Committee for Space Data Systems (CCSDS), Schedule-Aware bundle routing (SABR), white book, CCSDS 232.0-B-2, 2016.
[15] G. Araniti, N. Bezirgiannidis, E. Birrane, I. Bisio, S. Burleigh, C. Caini, M. Feldmann, M. Marchese, J. Segui, K. Suzuki, Contact graph routing in DTN space networks: overview, enhancements and performance, IEEE Coms. Mag. 53 (3) (2015) 38–46, doi:10.1109/MCOM.2015.7060480.
[16] S.E. Alaoui, B. Ramamurthy, Routing optimization for DTN-based space networks using a temporal graph model, in: 2016 IEEE International Conference on Communications (ICC), 2016, pp. 1–6, doi:10.1109/ICC.2016.7510733.
[17] J.A. Fraire, J.M. Finochietto, Design challenges in contact plans for disruption–tolerant satellite networks, IEEE Commun. Mag. 53 (5) (2015) 163–169.
[18] J.A. Fraire, P. Madoery, S. Burleigh, M. Feldmann, J. Finochietto, A. Charif, N. Zergainoh, R. Velazco, Assessing contact graph routing performance and reliability in distributed satellite constellations, Hindawi J. Comput. Netw. Commun. In Press.
[19] J.A. Fraire, P. Madoery, J.M. Finochietto, Leveraging routing performance and congestion avoidance in predictable delay tolerant networks, in: Wireless for Space and Extreme Environments (WiSEE), 2014 IEEE International Conference on, 2014, pp. 1–7, doi:10.1109/WiSEE.2014.6973079.
[20] J.A. Fraire, M. Feldmann, S. Burleigh, Benefits and challenges of cross-linked ring road satellite networks: A case study, 2017 IEEE International Conference on Communications Workshops (ICC), 2017. In Press.
[21] G. Wang, S.C. Burleigh, R. Wang, L. Shi, Y. Qian, Scoping contact graph-routing scalability: investigating the system's usability in space-vehicle communication networks, IEEE Veh. Technol. Mag. 11 (4) (2016) 46–52, doi:10.1109/MVT.2016.2594796.
[22] P. Madoery, P. Ferreyra, J. Fraire, F. Gomez, J. Barrientos, R. Velazco, Enhancing contact graph routing forwarding performance for segmented satellites architectures, in: 1st IAA Latin American Symposium on Small Satellites, Argentina, 2017.
[23] J.Y. Yen, Finding the k shortest loopless paths in a network, Manage. Sci. 17 (11) (1971) 712–716.
[24] N. Bezirgiannidis, C. Caini, V. Tsaoussidis, Analysis of contact graph routing enhancements for dtn space communications, Int. J. Satell. Coms. Netw. 34 (5) (2016) 695–709.
[25] N. Bezirgiannidis, F. Tsapeli, S. Diamantopoulos, V. Tsaoussidis, Towards flexibility and accuracy in space DTNcommunications, in: 8th ACM MobiCom Workshop on Challenged Networks, in: CHANTS '13, ACM, New York, NY, USA, 2013, pp. 43–48.
[26] P.G. Madoery, J.A. Fraire, J.M. Finochietto, Congestion management techniques for disruption-tolerant satellite networks, Int. J. Satell. Commun. Netw. n/a–n/a. Sat.1210. doi:10.1002/sat.1210.
[27] E. Birrane, Congestion modeling in graph-routed delay tolerant networks with predictive capacity consumption, in: 2013 IEEE Global Communications Conference (GLOBECOM), IEEE, Atlanta, GA, 2013, pp. 3016–3022.
[28] E. Birrane, Contact Graph Routing Extension Block, Internet Draft, 2013.
[29] J.A. Fraire, P.G. Madoery, F. Raverta, J.M. Finochietto, R. Velazco, Dtnsim: bridging the gap between simulation and implementation of space-terrestrial dtns, in: Space Mission Challenges for Information Technology (SMC-IT), 2017 IEEE Int. Conference on, 2017.
[30] A. Marinan, A. Nicholas, K. Cahoy, Ad hoc cubesat constellations: Secondary launch coverage and distribution, in: Aerospace Conference, 2013 IEEE, 2013, pp. 1–15, doi:10.1109/AERO.2013.6497174.
[31] M. Feldmann, F. Walter, Micro PCN; a bundle protocol implementation for microcontrollers, in: Wireless Communications Signal Processing (WCSP), 2015 Int. Conf. on, 2015, pp. 1–5, doi:10.1109/WCSP.2015.7341252.

Juan A. Fraire is an associate researcher at the National Research Council of Argentina (CONICET) and a professor at FAMAF University. He received the Telecommunications Engineering degree at the Instituto Universitario Aeronáutico and his PhD in the National University of Cćrdoba (UNC). His research focuses on DTN networking for space-oriented applications such as earth-observation LEO constellation projects defined in the Argentinian Space Agency (CONAE) National Space Plan. He has co-authored over 35 papers published in international journals and presented in leading international conferences.

Pablo G. Madoery received the Telecommunications Engineering degree from the Instituto Universitario Aeronáutico, Argentina, in 2012. He has worked at the Digital Communication Laboratory of the National University of Cćrdoba (UNC) in the field of satellite communications and currently he is pursuing a PhD in Engineering Sciences. His research interest is focused in models, algorithms and protocols for delay and disruption tolerant networks.

Amir Charif

Jorge M. Finochietto is Full Professor at Universidad Nacional de Córdoba (FCEFyN), Argentina. He is also Independent Reseracher at the National Research Council of Argentina (CONICET). Since November 2016, he is Visiting Professor at Politecnico di Torino. He holds a MS and PhD in Electronics Engineer from Universidad Nacional de Mar del Plata, Argentina, and from Politecnico di Torino, Turin, Italy, respectively. His research interests are in the field of performance evaluation, high-speed networking and switching, and optical and wireless networks. He has co-authored over 60 papers published in international journals and presented in leading international conferences.