Hindawi
Journal of Computer Networks and Communications
Volume 2017, Article ID 2830542, 18 pages
https://doi.org/10.1155/2017/2830542

Research Article

Assessing Contact Graph Routing Performance and Reliability in Distributed Satellite Constellations

J. A. Fraire,[1,2] P. Madoery,[2] S. Burleigh,[3] M. Feldmann,[4] J. Finochietto,[2] A. Charif,[1] N. Zergainoh,[1] and R. Velazco[1]

[1] *Université Grenoble-Alpes, INPG, TIMA Laboratoires, Grenoble, France*
[2] *Universidad Nacional de Córdoba-CONICET, Laboratorios LCD, Córdoba, Argentina*
[3] *Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, USA*
[4] *Faculty of Computer Science, Technische Universität Dresden, Dresden, Germany*

Correspondence should be addressed to J. A. Fraire; juanfraire@gmail.com

Received 3 February 2017; Accepted 5 April 2017; Published 4 July 2017

Academic Editor: Sandra Céspedes

Existing Internet protocols assume persistent end-to-end connectivity, which cannot be guaranteed in disruptive and high-latency space environments. To operate over these challenging networks, a store-carry-and-forward communication architecture called Delay/Disruption Tolerant Networking (DTN) has been proposed. This work provides the first examination of the performance and robustness of Contact Graph Routing (CGR) algorithm, the state-of-the-art routing scheme for space-based DTNs. To this end, after a thorough description of CGR, two appealing satellite constellations are proposed and evaluated by means of simulations. Indeed, the DtnSim simulator is introduced as another relevant contribution of this work. Results enabled the authors to identify existing CGR weaknesses and enhancement opportunities.

1. Introduction

The autonomous transmission of information resources and services through Internet has changed the lifestyle on Earth. Moreover, the potential benefits of extending Internet into space have been analyzed by the community [1–4]. Nonetheless, the consideration of Internet for space missions has been limited due to fundamental environmental differences. In particular, in a space flight mission, the highly varying communication ranges, the effect of planet rotation, and on-board power restrictions compels communication systems to face several disruptive situations nonexistent on Internet systems. Furthermore, the propagation delay of signals on Deep Space environments is generally in the order of minutes or even hours. These delay and disruption conditions contraindicate traditional Internet protocol operations as they are largely based on instant flow of information among sending and receiver nodes.

As a result, Delay/Disruption Tolerant Networks (DTNs) have recently been considered as an alternative to extending Internet boundaries into space [5]. In particular, recent studies have considered their applicability in Low-Earth Orbit (LEO) satellite constellations [6–11]. To overcome link disruptions, DTN nodes temporarily *store* and *carry* in-transit data until a suitable next-hop link becomes available [12]. To overcome delays, end-to-end feedback messages are no longer assumed continuous nor instantaneous. This distinctive characteristic allows DTN to operate in environments where communications can be challenged by latency, bandwidth, data integrity, and stability issues [13].

During the last decade, the DTN Bundle protocol, along with different adaptation layers, has been proposed [14–19], several routing strategies were studied [20–30], and diverse software stacks were publicly released [31–35]. Furthermore, some of the latter approaches were successfully validated both on LEO [36] and Deep Space missions [37] driven by the UK Space Agency and NASA, respectively. Also, DTN has been in pilot studies in the International Space Station (ISS) since 2009 [38] and has been operational on ISS since May of 2016. Presently, the Internet research community [39] along with several space organizations [40, 41] has joined the industry in the standardization of DTN protocols [42].

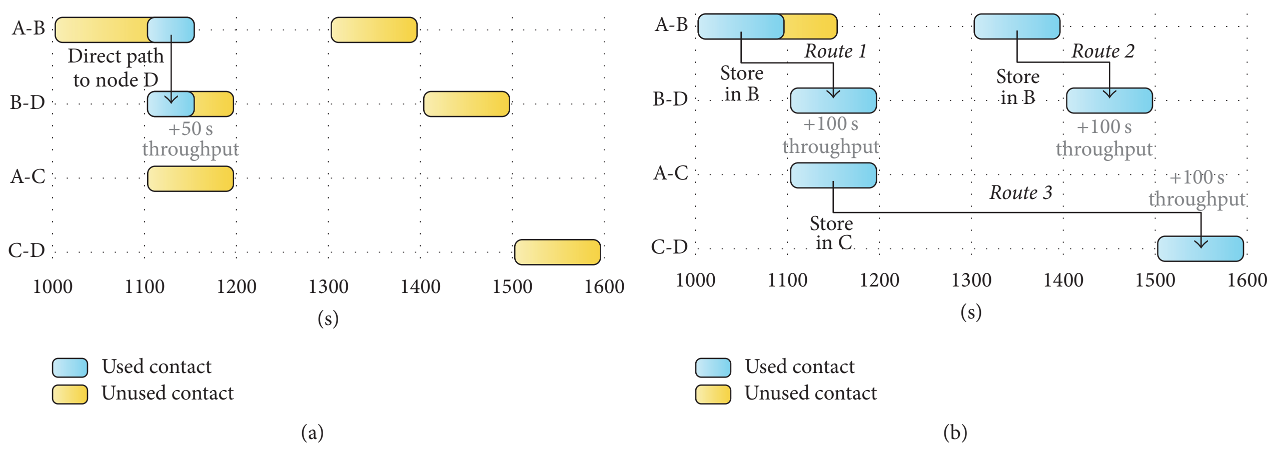

Figure 1: DTN topology example. Effective throughput with (a) Internet protocols and (b) DTN store-and-forward.

In spite of the recent advances in the area, the analysis of the fault-tolerance of existing DTN solutions remains an open research topic. Studying reliability of DTN is mandatory before seriously considering its applicability in the harsh space environment where radiation effects, vibrations, collisions, and outgassing, among others, pose significant challenges for man-made spacecraft. In [43], the reliability of opportunistic DTNs was studied, but their results do not apply to the space domain where communications are deterministic. More recently, authors have presented preliminary results on a reliability assessment of DTN for space applications [44]. However, the provided simulation analysis was based on simplistic satellite networks without a comprehensive analysis on the behavior of the underlying routing algorithm.

In this work, we tackle the weaknesses of [44] by providing an extensive fault injection analysis based on two appealing and realistic case studies of delay-tolerant satellite constellations previously presented in [10]. An initial performance comparison of these topologies is one of the contributions of this work. For the reliability analysis, the state-of-the-art version of Contact Graph Routing (CGR) algorithm was considered [30]. However, to the best of the authors' knowledge, there is a lack of an in-depth description of the latest CGR algorithm which is only available as part of an open-source DTN implementation [32]. This results in a fuzzy interpretation of the algorithm which is specifically an issue because several further approaches claim to leverage CGR as a basis (such as [45, 46]). Consequently, providing an accurate and thorough description of how CGR is currently implemented is another contribution of this paper. The CGR algorithm was implemented in DtnSim, a new simulator specifically designed to evaluate space DTNs. DtnSim is also introduced in this article for the first time and is expected to be available via open-source licensing. Results obtained from DtnSim are the final contribution of this article: an assessment of the fault-tolerance capability of the CGR algorithm in DTN constellations. This work can thus be considered an improved, extended, and archival quality version of [44].

The present paper is structured as follows. In Section 2 an overview of DTN, a detailed description of the CGR algorithm, and an appealing failure model are provided. Two realistic satellite constellation scenarios are described and analyzed by means of a new simulation framework in Section 3. In Section 4 open research aspects are summarized and discussed before concluding in Section 5.

2. DTN Overview and System Model

A simple disrupted network of 4 nodes is illustrated in Figure 1, where node A sends data to node D. Similar examples can be found in [47, 48] describing the behavior of satellite constellations with sporadic connectivity as well as Deep Space networks. The time-evolving topology is represented by a timeline of 1600 seconds, where different communication opportunities (also known as *contacts*) exist among nodes A, B, C, and D. Formally, a contact is defined in [47] as an interval during which it is expected that data will be transmitted by one DTN node to another. For example, node A has two direct contacts with B (A-B) from 1000 s to 1150 s and from 1300 s to 1400 s. However, the effective utilization of these communication episodes depends on the implemented protocols.

Figure 1(a) shows the expected performance of Internet protocols which require a persistent connectivity with the final destination (also known as end-to-end path). Due to the disruptive nature of the network, node A is only able to directly reach nodes D through B from 1100 s to 1150 s (cyan-colored contacts), which allows for an effective throughput of 50 s multiplied by the node's data-rate. Other contacts will remain unutilized by Internet protocols (yellow-colored contacts). By exploiting a local storage within each node, DTN is able to make a better utilization of the communication resources as shown in Figure 1(b). For example, in order to better use the B to D contact at 1100 s, node A can transmit in advance the data to node B starting from 1000 s. Furthermore, two other delay-tolerant paths can be considered to node D, one via node C at 1100 s and another via node B at 1300 s. Thus, the effective throughput of DTN in this example is of 300 s (600% higher than traditional Internet protocols). For a more general overview, Table 1 compares different aspects between Internet and DTN Protocols.

Table 1: IP and DTN protocols comparison.

	Internet protocols	DTN protocols
Connectivity	Continuous end-to-end transaction.	Sporadic or none end-to-end transaction and high link latency.
Storage	Small buffers to mitigate link congestion.	Large memories to overcome link disruption.
Underlying protocols	Ethernet, WiFi, fiber optic, and so forth.	Internet protocols, ethernet, WiFi, fiber optic, and so forth.
Applications	Internet, mobile networks without disruptions, and wireless sensor networks.	Deep Space systems, satellite networks, underwater networks, and mobile networks with disruptions.

Table 2: Contact plan example.

Contact #	From	To	Ini	End	Rate
1	A	B	1000	1150	1000
2	B	A	1000	1150	1000
3	B	D	1100	1200	1000
4	D	B	1100	1200	1000
5	A	C	1100	1200	1000
6	C	A	1100	1200	1000
7	A	B	1300	1400	1000
8	B	A	1300	1400	1000
9	B	D	1400	1500	1000
10	D	B	1400	1500	1000
11	C	D	1500	1600	1000
12	D	C	1500	1600	1000

In spite of the data delivery improvement, different challenges and optimization opportunities exist in DTN. For example, the first contact of nodes A to B in Figure 1(b) is not fully utilized. This is because node A was able to make a good "guess" on node B's connectivity and its residual capacity with final destination D. However, it can be quite difficult to accurately make such estimations without a stable and permanent connection with neighbor nodes. Without this local knowledge assumed by node A, more data could have been transmitted provoking congestion at node B. Congestion is, indeed, a popular and open research topic in DTN [49–51]. In general, and in contrast with Internet protocols, nodes' reliance on a realistic local understanding of the current connectivity in the network is not always feasible and depends on the type of DTN they run on.

As a result, existing routing and forwarding schemes for DTN have sought to acquire the most complete and precise network state information. In *opportunistic* DTNs, no assumptions can be made as encounters between nodes occur unexpectedly [28]. For these networks, epidemic strategies based on message replication driven by different criteria have been applied [27, 29]. Nevertheless, in realistic situations, contacts are rarely totally random but obey nodes' movement with greater probability of meeting certain neighbors than others. Protocols such as [20, 25, 26] are popular solutions that propose to infer the encounter probability to improve data delivery metrics. On the other hand, connectivity in certain DTNs such as space networks can be precisely anticipated based on accurate mathematical models describing object trajectories in space. In the literature, these networks are known as *scheduled* or *deterministic* DTNs [12] and are the appropriate model to study satellite constellations.

In general, spacecraft trajectories and orientation can be accurately predicted by means of appropriate mathematical models [52]. Also, mission operations generally account for precise models of the communication systems both on-board and on-ground. As a result, the forthcoming spacecraft to spacecraft or spacecraft to ground contacts can be determined or even controlled in advance. This unique characteristic has made routing in scheduled DTN a distinct research area of increasing interest during the last decade.

First analysis on routing in scheduled DTNs dates back to 2003 where Xuan et al. proposed time-evolving graph to represent changes in network topologies to then study shortest, foremost, and fastest journey metrics [21]. Later, a specific routing framework was introduced in [22] to derive a space-time routing table comprising next hops for each time interval. Similar schemes were reported in [23]. However, these static route calculation approaches relied on a complete precalculation of routes on ground and a timely distribution to the network nodes. Due to the precalculation, these approaches lacked responsiveness to varying traffic conditions and topology changes resulting from dynamically added or removed contacts. An alternative approach addressing this shortcoming was later introduced under the name of *Contact Graph Routing* (CGR).

2.1. Contact Graph Routing. Instead of a centralized route calculation, CGR, proposed by S. Burleigh (NASA JPL), follows a distributed approach: the next hop is determined by each DTN node on the path by recomputing the best route to destination, as soon as a bundle (i.e., bundle protocol data unit) is received. This routing procedure assumes that a global *contact plan*, comprising all forthcoming contacts, is timely distributed in advance in order to enable each node to have an accurate understanding of the network [47]. Table 2 shows the contact plan for the sample network from Figure 1. Routes can thus be calculated by each node on demand, based on extensive topological knowledge of the network combined with the assumed traffic status. This workflow is illustrated in Figure 2 where a contact plan is initially determined by means of orbital propagators and communication models, then distributed to the network, and finally used by the DTN nodes to calculate efficient routes to the required destinations. Indeed, by combining the contact plan with local information such as outbound queue backlog and excluded neighbors (e.g., unresponsive neighbors), CGR is able to dynamically respond to changes in network topology and traffic demands. An early version of CGR was flight-validated in Deep Space

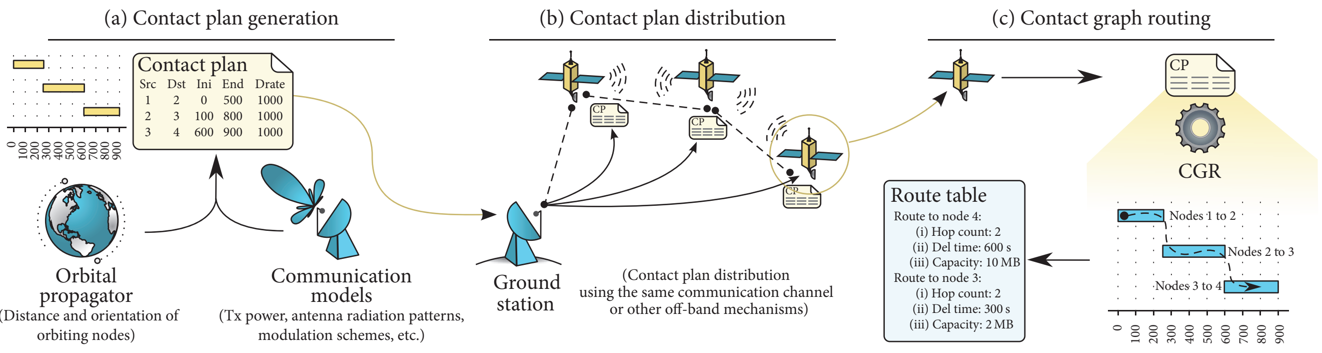

Figure 2: Contact plan generation, distribution, and utilization by CGR.

by NASA in 2008 [37] and CGR has been one of the most studied routing solutions for space networking since then.

CGR was initially documented in 2009 and later updated in 2010 as an experimental IETF Internet Draft [53]. By the end of the same year, Segui et al. [54] proposed earliest-arrival-time as a convenient monotonically decreasing optimization metric that avoids routing loops and enables the use of standard Dijkstra's algorithm for path selection [55]. This enhancement was then introduced in the official version of CGR, included in the Interplanetary Overlay Network (ION) DTN Stack developed by the Jet Propulsion Laboratory [32]. In 2012, Birrane et al. proposed source routing to reduce CGR computations in intermediate nodes at the expense of packet header overhead [56]. Later, in 2014, authors studied the implementation of temporal route lists as a mean to minimize CGR executions [57]. In the same year, Bezirgiannidis et al. [58] suggested to monitor transmission queues within CGR as they increase the earliest transmission opportunity (CGR-ETO). On the same paper, a complementary *overbooking management* innovation enables proactive reforwarding of bundles whose place in the outbound queue was taken by subsequent higher priority traffic. Both ETO and overbooking management are now included in the official CGR version. Regarding congestion management, further extensions were proposed as well [49–51]. Most recently, in 2016, Burleigh et al. introduced an opportunistic extension as a means of enlarging CGR applicability from deterministic space networks to opportunistic terrestrial networks [48]. The latter is not a trivial contribution since it could, if successful, pave the way towards implementing space DTN advances on ground-based networks. At the time of this writing, CGR procedure is being formally standardized as part of the Schedule-Aware Bundle Routing (SABR) procedure in a CCSDS Blue Book [47]. All these modifications have been implemented in ION software. As a result, this software becomes an important point of reference for the latest routing and forwarding mechanisms for space DTNs. The current version 3.5.0 of ION was released in September 2016 and is available as free software [33].

Even though the latest version of the CGR algorithm is implemented as part of the ION 3.5.0 open-source code (in fact, the CGR algorithm implemented in ION 3.5.0 includes a few parameters and procedures related to opportunistic CGR (O-CGR) [48]; O-CGR is an experimental CGR extension that also considers discovered contacts in addition to those in the contact plan; since in this work all contacts are scheduled, the probabilistic calculations are not discussed nor described in this section), there is no detailed and formal description of the algorithm available yet (the CCSDS documentation is still under development [47]). As a result, in this section, an in-depth explanation of the CGR scheme is provided. For the sake of clarity, the discussed algorithms are structured following their implementation in ION; however, they can be translated to more compact and elegant expressions without *continue*, *break*, and *return* statements.

The CGR Forward routine depicted in Algorithm 1 is called at any network node every time a new bundle B is to be forwarded. Initially, the algorithm checks if the local view of the topology expressed in the contact plan Cp was modified or updated since the last call (Algorithm 1, lines (2) and (3)). If modified, a route list Rl structure, holding all valid routes to each known destination, is cleared in order to force an update of the route table. Indeed, the route list Rl is derived from the local contact plan Cp. It is interesting to note that, in contrast with Internet routes, DTN routes are expressed in function of time. Therefore, they are only valid for a given period of time and need to be revisited by CGR for every new bundle. Next, the procedure populates a proximate nodes list Pn comprising all possible nodes that, according to the route list Rl, have a valid path towards the destination (Algorithm 1, lines (6)). This step is executed by the *identifyProxNodes* routine which is detailed in Algorithm 2 and discussed below. An excluded nodes list En is used in this step to avoid the consideration of administratively forbidden neighbors (e.g., unresponsive nodes) or the previous bundle sender to minimize route loops (Algorithm 1, lines (4) and (5)).

Once populated, the Pn list can be used to forward the bundle to the appropriate neighbors. If the bundle is critical (a special type of bundle), the bundle is cloned and enqueued to all possible neighbors in the Pn list (Algorithm 1, lines (7) to (9)). If the bundle is of normal type, a single candidate node is chosen from the proximate nodes list Pn. Neighbors with best arrival times to the destination are the top priority, then those with least hop count (in CGR terminology, one contact is one hop), and finally those with a smaller node id (Algorithm 1, lines (11) to (23)). Then, if a suitable proximate node is found, the bundle B is inserted in the corresponding outbound queue before executing the overbooking management

```
     input: bundle to forward $B$, contact plan $Cp$, route list $Rl$, excluded nodes $En$, proximate nodes $Pn$
     output: bundle $B$ enqueued in the corresponding queue
(1)  $En \leftarrow \emptyset$;
(2)  if $Cp$ changed since last $Rl$ calculation then
(3)      $Rl \leftarrow \emptyset$;
(4)  if $B$ forbids return to sender then
(5)      $En \leftarrow B$ sender node;
(6)  $Pn \leftarrow$ identifyProxNodes ($B$, $Cp$, $Rl$, $En$);
(7)  if $B$ is critical then
(8)      enqueue a copy of $B$ to each node in $Pn$;
(9)      return
(10) set $nextHop$ to empty;
(11) for $pn \in Pn$ do
(12)     if $nextHop$ is empty then
(13)         $nextHop = pn$
(14)     else if $pn.arrivalTime < nextHop.arrivalTime$ then
(15)         $nextHop = pn$
(16)     else if $pn.arrivalTime > nextHop.arrivalTime$ then
(17)         continue
(18)     else if $pn.hops < nextHop.hops$ then
(19)         $nextHop = pn$
(20)     else if $pn.hops > nextHop.hops$ then
(21)         continue
(22)     else if $pn.id < nextHop.id$ then
(23)         $nextHop = pn$
(24) if $nextHop$ is not empty then
(25)     enqueue $B$ to $nextHop$;
(26)     manageOverbook ($B$, $nextHop$)
(27) else
(28)     enqueue $B$ to limbo
(29) return
```

Algorithm 1: CGR.

procedure (the overbooking management procedure defined in [58] aims at reordering the local outbound queues when bundles with higher priorities replace less urgent bundles; since in this work we assume all bundles have the same priority, this procedure is not described) (Algorithm 1, lines (25) to (26)). However, if the CGR Forward fails to find a suitable neighbor, the bundle is stored in special memory space called *limbo* waiting for a higher level process to either erase it or retry a new forwarding later (e.g., after a contact plan update). As a result, after the CGR routine is completed, one or more bundles might be stored in the local memory waiting for the contact with the corresponding proximate node. As discussed later, if, for one reason or another (i.e., congestion or failure), the bundle is not transmitted during the expected contact, it will be removed from the queue and rerouted by this CGR routine.

The *identifyProxNodes* routine, depicted in Algorithm 2, explores existing routes in order to derive a proximate node list *Pn*. The *Pn* list is used by the main CGR routine and is formed by a set of nonrepeating neighbor nodes that are capable of reaching the bundle destination. If the route list *Rl* is empty (i.e., first time the routing routine is executed after a contact plan update), the load route list function is called in order to find all routes towards the destination of *B* (Algorithm 2, lines (1) and (2)). At this stage, *Rl* accounts for all possible routes to the destination. Each of them needs to be evaluated in order to populate the proximate node list *Pn*. Initially, those routes that do not satisfy specific selection criteria are discarded (Algorithm 2, lines (3) to (14)). In particular, routes with the latest transmission time (*toTime*) in the past, an arrival time later than the bundle deadline, a capacity less than the bundle size, and a proximate node within the excluded nodes, or that require a local outbound queue that is depleted, are ignored. Remaining routes are then considered suitable, and the corresponding proximate node in the *Pn* list is either replaced by a better route (Algorithm 2, lines (15) to (24)) or directly added to the list (Algorithm 2, lines (26) and (27)). The replacement criteria is coherent with Algorithm 1: best arrival time is considered first and then route hop count. During this process, necessary route metrics such as arrival time and hop count of the best route are also stored in each proximate node data structure contained in *Pn*.

The routines in Algorithms 1 and 2 are executed on a per-bundle basis. The reason behind this is that the parameters of each bundle (destination, deadline, and size), the local outbound queue status, and the excluded nodes list need to be revised on every new forwarding in order to base the decision on an up-to-date version of the proximate nodes list *Pn*. In general, these routines are considered part of a *forwarding* process of CGR. On the other hand, the route list (*Rl*) will

```
      input: bundle to forward $B$, contact plan $Cp$, route list $Rl$, excluded nodes $En$,
      output: proximate nodes list $Pn$
(1) if $Rl$ is empty then
(2)     $Rl \leftarrow$ loadRouteList $(B, Cp)$;
(3) $Pn \leftarrow \emptyset$;
(4) for $route \in Rl$ do
(5)     if $route.toTime \leq currentTime$ then
(6)         continue (ignore past route)
(7)     if $route.arrivalTime \geq B.deadline$ then
(8)         continue (route arrives late)
(9)     if $route.capacity < B.bitLenght$ then
(10)        continue (not enough capacity)
(11)    if $route.nextHop \in En$ then
(12)        continue (next hop is excluded)
(13)    if $localQueue(route.nextHop) < B.bitLength$ then
(14)        continue (outbound queue depleted)
(15)    for $pn \in Pn$ do
(16)        if $pn = route.nextHop$ then
(17)            if $pn.arrTime > route.arrTime$ then
(18)                replace $pn$ with $route$.nextHop
(19)            else if $pn.arrTime < route.arrTime$ then
(20)                continue (previous route was better)
(21)            else if $pn.hops > route.hops$ then
(22)                replace $pn$ with $route$.nextHop
(23)            else if $pn.hops < route.hops$ then
(24)                continue (previous route was better)
(25)            break
(26)    if $route.nextHop \notin Pn$ then
(27)        $pn \leftarrow route$.nextHop;
(28)        $Pn \leftarrow pn$;
(29) return $Pn$
```

Algorithm 2: CGR: Identify proximate node list.

need to be updated whenever the local contact plan *Cl* is modified. The determination of the routes is considered part of a *routing* process of CGR and is described below.

In general, to find all possible routes from a source to a destination, CGR uses a *contact graph* expression of the contact plan. A contact graph is a conceptual directed acyclic graph whose vertices correspond to contacts while the edges represent episodes of data retention (i.e., storage) at a node [47]. Also, two notional vertices are added: the root vertex, which is a contact from the sender to itself, and a terminal vertex, which is a contact from the destination to itself. Even though the resulting contact graph structure may seem counterintuitive, it is a convenient static representation of a time-evolving topology that can be used to run traditional graph algorithms such as Dijkstra's searches. For example, Figure 3 illustrates the contact graph corresponding to the network shown in Figure 1. The three discussed routes with their corresponding metrics are also included in the illustration (note that another feasible path from A to D exists through contacts 1 and 9).

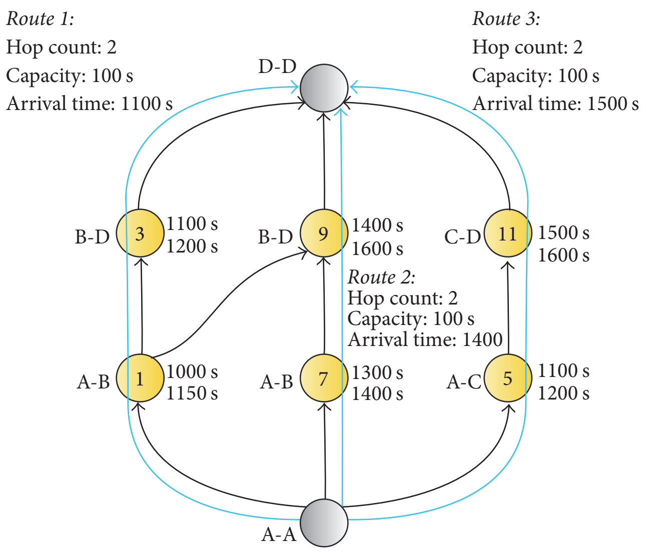

Figure 3: Contact graph example.

In order to find all possible routes in the contact plan, the load route list routine performs a series of Dijkstra's searches over a contact graph derived from the contact plan. The algorithm is listed in Algorithm 3 and is described as follows. A *work* area is reserved for each contact in the contact plan in order to run Dijkstra's searches. Initially, the algorithm clears all the required parameters in each working area of the contacts (Algorithm 3, lines (2) to (7)). Then, the routine

```
     input: bundle to forward $B$, contact plan $Cp$
     output: route list $Rl$
(1)  $Rl[B.\text{destination}] \leftarrow \emptyset$;
(2)  for $contact \in Cp$ do
(3)      $contact$.work.arrivalTime ← inf;
(4)      $contact$.work.capacity ← 0;
(5)      $contact$.work.predecessor ← 0;
(6)      $contact$.work.visited ← $false$;
(7)      $contact$.work.suppressed ← $false$;
(8)  set $anchorContact$ to empty;
(9)  while 1 do
(10)     $route$ ← Dijkstra ($B$.dst, $Cp$, $rootContact$);
(11)     if $route$ is empty then
(12)         break (no more routes in contact graph)
(13)     $firstContact \leftarrow route$.firstHop;
(14)     if $anchorContact$ not empty then
(15)         if $anchorContact \neq firstContact$ then
(16)             for $contact \in Cp$ do
(17)                 $contact$.work.arrivalTime ← inf;
(18)                 $contact$.work.predecessor ← 0;
(19)                 $contact$.work.visited ← $false$;
(20)                 if $contact.source \neq local\ node$ then
(21)                     $contact$.work.suppressed ← $false$;
(22)             $anchorContact$.work.suppressed ← $true$;
(23)             set $anchorContact$ to empty;
(24)             continue (go to next Dijkstra's search)
(25)     $Rl[B.\text{destination}] \leftarrow route$;
(26)     if $route.toTime = firstContact.endTime$ then
(27)         $limitContact \leftarrow firstContact$;
(28)     else
(29)         $anchorContact \leftarrow firstContact$;
(30)         for $contact \in route.hops$ do
(31)             if $contact.toTime = route.toTime$ then
(32)                 $limitContact \leftarrow contact$;
(33)                 break (limit contact found)
(34)     $limitContact$.work.suppressed ← $true$;
(35)     for $contact \in Cp$ do
(36)         $contact$.work.arrivalTime ← inf;
(37)         $contact$.work.predecessor ← 0;
(38)         $contact$.work.visited ← $false$;
(39) return
```

Algorithm 3: CGR: Load route list.

loops to find different routes using Dijkstra's algorithm (Algorithm 3, line (10)). The metric driving the shortest path search is the arrival time, which must be calculated from the starting time and expected delay of each contact in the explored path. In order to guarantee that each Dijkstra execution provides a distinct route, the load route list process removes the *limiting contact* of the last route found from the following search. The limiting contact is defined as the earliest ending contact in the route path [47]. In general, the limiting contact often happens to be the first contact of the route (e.g., see contacts 1, 7, and 5 in Figure 3). Therefore, removing the limiting contact forces the shortest path search to provide the next best route towards the destination (Algorithm 3, line (13)). However, in some special cases, the transmitter node is behind a very long contact (e.g., an Internet contact). For these cases, an *anchoring* mechanism allows the algorithm to find several routes through longer contacts. The anchor search begins as soon as the algorithm detects that the first contact is not the limiting contact (Algorithm 3, lines (26) to (28)). In this stage, the anchor contact is stored and the limiting contact in the path is found and suppressed for further calculations (Algorithm 3, lines (29) to (34)). After clearing the working area (Algorithm 3, lines (35) to (38)), a new search is executed, now with the limiting contact detached from the contact graph. As soon as the first route without the anchor contact as the first contact is found, the anchored search ends suppressing the anchor contact and the normal search continues (Algorithm 3, lines (15) to (24)). The search ends when no more routes can be found in the contact graph.

Although a complex and extensive algorithm, CGR is considered to be among the most mature strategies towards forwarding and routing in space DTNs [30]. To the best of authors' knowledge, this section constitutes one of the most detailed overviews of the algorithm in the literature as of today. The correctness of such description is supported by the implementation of CGR in a simulator platform where the satellite network is submitted to faults as described below.

2.2. Fault Model. Over the last years, the semiconductor industry has been particularly concerned by the effects of radiation on integrated circuits and embedded systems in general [59]. The rationale behind this motivation lies not only in the use of these systems in radioactive environments but also in the increasing degree of integration of devices embedded in the same chip. Recent studies have shown that the smaller the feature sizes, the greater the sensitivity to radiation-induced errors [60]. As a consequence, modern embedded systems may be susceptible to low-energy particles including those observed within the Earth's atmosphere.

This effect is even more dramatic in space missions, which require systems that can operate reliably for long periods of time with little or no maintenance. This is the case in satellite constellations under study in this work. Among the possible errors, transient errors occur in the system temporarily and are usually caused by radiation interference, also known as *single event upsets* (SEUs). In particular, any circuit comprising memory elements (registers, flip-flops, internal memory, etc.) can at any moment undergo the modification of one or several bits of information due to ionizing particles. Other transient outages might be provoked by overheating, radiofrequency interference, or a processor reboot following a software exception. Traditionally, missions are designed to tolerate these failures by detecting the erroneous behavior and then recovering the system, typically by means of a full restart [59].

The random occurrence in time and space of such failure phenomenon, the probability of the error to happen, and the probability of effectively detecting an unwanted behavior, can be modeled by means of an exponential (Poisson) distribution [61]. The exponential model makes emphasis on the number of failures, the time interval over which they occur, and the environmental factors that may have affected the outcomes. As a result, it provides two feasible outputs at every moment: normal operation or failure. The model does not assume wear-out or depletion (as in batteries), implying that it only accounts for failures occurring at random intervals but with fixed long-term average frequency. Indeed, the outcome of this kind of failure model is also known as *memoryless* distribution [62]. The exponential distribution model is the most commonly adopted fault model, mainly due to its simplicity and effectiveness. It takes a single known average failure rate parameter and can be conveniently described by means of

$$F(t) = \int_0^t \lambda e^{-\lambda\tau} = 1 - e^{-\lambda\tau}. \qquad (1)$$

In (1) λ is known as the *failure rate* and τ is the *time*. The failure rate reduces to the constant λ for any time: this proves that the exponential model is indeed memoryless. One

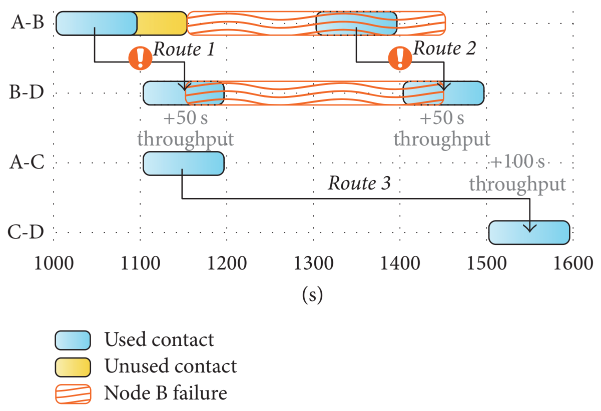

FIGURE 4: Failures in a DTN network.

of the input parameters for obtaining the modules outages is the Mean Time To Failure (MTTF), which is MTTF $= 1/\lambda$. Moreover, while the time to failure is determined by the exponential model with the MTTF, the Mean Time To Repair (MTTR) defines the time to recover after the failure. Both the error detection and system recovery mechanism are presented as part of the MTTR interval. Accurately determining both MTTF and MTTR for a real spacecraft component was investigated in [63].

By means of the MTTF, MTTR, and the fault model, a fault injection system was designed to study the resulting traffic flow of a DTN based on CGR algorithm under failure conditions. In particular, MTTF and MTTR are used as parameters to randomly select fault position (i.e., node to fail), starting point, and end point in the DTN network. This phenomenon is illustrated by an example in Figure 4, where a failure in node B at 1150 s and a subsequent recovery at 1450 s forces making a partial usage of route 1 and forbids sending any data through route 2. In this particular example, 50% of route 1 data is kept stored in node B memory after the failure is found (i.e., a reliable storage is assumed in each node). Once recovered, the node finds it has bundles stored for a previous contact that need to be reforwarded. As per the description of Section 2.1, CGR will find that the ongoing direct contact with D (from 1400 s to 1500 s) is a valid path. However, all data that node A was supposed to forward to B via route 2 will need to be reforwarded to an alternative path. In CGR, this will happen after the contact ends with bundles in the outbound queue. As a result, the overall throughput of the DTN system as well as the delivery time of the data might be degraded by transient errors that block the calculated route path.

Although this example may be intuitively obvious, the complexity of the analysis drastically increases as more nodes, more contacts, more traffic flows, and more failures are injected in the network. Furthermore, the system degradation is expected to depend on the network topology and the derived contact plan. In consequence, the CGR algorithm and the exponential failure model were integrated into a single simulation framework designed to provide accurate measurements of the degradation of the metrics of DTNs in presence of transient failures.

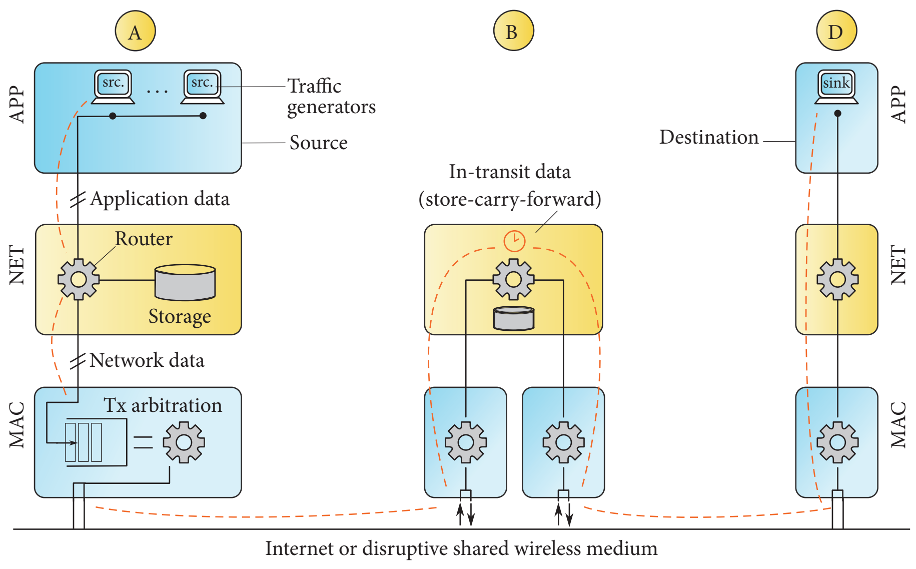

Figure 5: DTN node architecture in DtnSim.

3. Simulation Analysis

3.1. Simulation Platform. There exist several tools to evaluate DTNs. Among them, the ONE simulator [64] has been extensively used for DTN studies; however, the platform is specifically designed to model opportunistic DTNs (social networks, vehicular networks, etc.). On the other hand, emulation environments such as CORE [65] are available to directly test existing DTN implementations; but CORE has to be executed in real time, which hinders its application in extensive analysis. To tackle these limitations, a new simulator called DtnSim (DtnSim is still under development; however, the code will be made publicly available under an open-source license at https://bitbucket.org/lcd-unc-ar/dtnsim) was implemented in Omnet++ [66], a discrete event network simulator platform. Using an event driven framework allows DtnSim to efficiently simulate scenarios at accelerated speeds. This is crucial for space environments where analysis over orbital periods spanning several days or weeks of duration is required. Indeed, the architecture of DtnSim is specifically designed to evaluate scheduled DTNs such as satellite and Deep Space systems.

An indefinite number of DTN nodes can be spanned and configured by a single file in DtnSim. Each of these nodes are based on the layered architecture illustrated in Figure 5. This architecture is a simplification of the original DTN architecture [12] that has been adapted for DTN-based satellite systems in [10] and is comprised of an application (APP), network (NET), and Medium Access Control (MAC) layer. Physical layer effects such as bit error rate can be modeled within the MAC layer if required.

The APP layer is the element that generates and consumes user data. In general, in the case of Earth observation missions, a large amount of information is produced in on-board remote sensing instruments and is required to be delivered to a centralized node on Earth [10]. On the other hand, in communication or data relay systems, the data is generated or demanded by end users on ground and is generally sent via satellites to another node on ground (i.e., Internet gateway or mission control center). Typically, the traffic exchange in communication-oriented DTN systems is of the publish/subscribe type, instead of a client/server as traditionally seen on Internet. A publish/subscribe system allows nodes to autonomously generate and send data to those nodes which can potentially be interested without relying on instantaneous feedback messages, a key principle in DTN. As a result, in DtnSim, the APP layer allows generating either a large amount of punctual traffic or periodic amounts of data.

The NET layer is the element in charge of providing delay-tolerant multihop transmission (i.e., routes) and is probably the most mature module of DtnSim at the moment. In this layer, each DTN node includes a local storage unit (non-volatile memory model) in order to store in transit bundles. Also, the CGR routing algorithm described in Section 2.1 was implemented as an exchangeable submodule of the network layer; thus, other algorithms can easily be supplied via a clearly defined interface. Independently from the routing submodule, the NET layer can take a contact plan as an input which defines the forthcoming contact opportunities among the nodes. The contact plan format is based on the format used in the ION software stack: a text file comprised of a list of contacts in the following form:

```
contact <start> <end> <source> <destination> <rate>
```

Table 3: Satellite parameters.

		Common parameters		
Orbit epoch		1 Jan 2017 10:00:00.000 UTCG		
Semimajor axis [km]		6878.14		
Eccentricity		0		
Arg. perigee [deg]		0		
EID Satellite 1, 5, 9, 13	32	36	40	44
EID Satellite 2, 6, 10, 14	33	37	41	45
EID Satellite 3, 7, 11, 15	34	38	42	46
EID Satellite 4, 8, 12, 16	35	39	43	47
		Walker formation		
	Plane A	Plane B	Plane C	Plane D
Inclination [deg]		50		
RAAN [deg]	0	90	180	270
True Anomaly Sat 1	0	22.626	45.245	67.619
True Anomaly Sat 2	90	112.626	135.245	157.619
True Anomaly Sat 3	180	202.626	225.245	247.619
True Anomaly Sat 4	270	292.626	315.245	337.619
		Along-track formation		
Inclination [deg]		97.03		
RAAN [deg]	0	0	0	0
True Anomaly Sat 1	0	20	40	60
True Anomaly Sat 2	5	25	45	65
True Anomaly Sat 3	10	30	50	70
True Anomaly Sat 4	15	35	55	75

The MAC layer of the DtnSim node is designed to provide a reliable wireless link and to multiplex the shared medium among nodes with wireless interfaces. Although each DtnSim node is based on a single APP and NET layer, several MAC modules can be attached to the NET module in order to mimic various transceivers on a single node. Thus, this layer sets the real bundle transmission rate which will depend on the transmitter/receiver module bit-rate as well as any possible medium arbitration mechanism (e.g., contention). An example analysis based on this layer has been presented in [10], where dynamic and static channel negotiations are compared. For the sake of simplicity, the MAC layer used for the present analysis transparently sends and receives bundles without further intervention. Thus, the same module is used to communicate nodes connected through Internet on ground.

Finally, DtnSim was extended with a fault-injector module based on the exponential model described in Section 2.2. In particular, the MTTF and MTTR are provisioned to each node so as to determine when the node is in a failure or normal operation state. When a node is in a failure state, the node is not able to send nor receive bundles by any of its MAC interfaces. However, if it has bundles stored in the local storage, they are not modified. Such a model mimics existing nonvolatile spacecraft data recorder systems typically on board of medium and high-end spacecraft. Nonetheless, the fault model can be easily extended to delete or corrupt stored data.

3.2. Simulation Scenarios. In order to assess CGR behavior under transient failures in realistic delay-tolerant satellite constellations, two appealing and realistic configurations are proposed and discussed below: a sun-synchronous along-track and a Walker-delta formation. Both constellations are based on 16 cross-linked LEO satellites (max. link range of 1000 Km at 500 km height), 25 ground target points, and 6 ground stations. The specific orbital parameters of the satellites are summarized in Table 3 while the ground stations and target locations are in Table 4 (the resulting contact plans used for the simulations as well as the STK [67] scenario of the Walker formation can be found in https://upcn.eu/icc2017.html). Systems Tool Kit (STK) [67] software was used to propagate these parameters for an analysis period of 24 hs. An intuitive illustration of the node's location on a world map is provided in Figure 6. The left side of the picture plots the ground tracks of the Walker formation while the along-track is on the right.

3.3. Simulation Results. In the along-track formation, all 16 satellites are equally spaced and follow a very similar orbital trajectory. In this analysis, each satellite is able to reach the next neighbor in the front and in the back along the trajectory. Among the many benefits of such formation, satellites do not require complex transfer maneuvers if launched from the same vector. Also, since satellites perceive similar gravitational perturbations, significant savings in propellant for formation-keeping can be made [68]. From a

Table 4: Ground station and ground target parameters.

Station	Lat [deg]	Lon [deg]	EID
Córdoba	−31.524	−64.463	1
A. Springs	−23.758	133.883	2
Leolut	39.908	116.42	3
Neustrelitz	53.329	13.07	4
Hartebeesthoek	−25.886	27.712	5
JPL (USA)	34.203	−118.173	6
Target	Lat [deg]	Lon [deg]	EID
Afghanistan	33.833	66.025	7
Argentina	−40.843	−68.090	8
Australia	−23.274	129.385	9
Bangladesh	23.730	90.306	10
Brazil	−10.813	−55.460	11
Chile	−53.132	−70.915	12
Congo	−0.842	15.230	13
Egypt	26.756	29.860	14
Iraq	33.045	43.775	15
Kazakhstan	44.161	79.766	16
Kenya	0.577	37.839	17
Mali	17.357	−3.527	18
Mongolia	46.836	103.067	19
Namibia	−22.150	17.177	20
Nepal	28.259	83.944	21
Nigeria	9.560	8.077	22
Papua NG	−6.889	146.214	23
Peru	−3.732	−73.240	24
Russia	50.339	127.553	25
Algeria	25.808	10.021	26
Sudan	16.085	30.087	27
Suriname	4.217	−55.889	28
Bolivia	−21.521	−64.738	29
Vietnam	16.940	106.816	30
Zambia	12.407	9.254	31

communications perspective, the topological stability of this formation also favors the simplicity of fixed antennas against complex gimbal mounts or electronically steered antennas for ISLs. Similar topologies have been used in previous satellite DTN studies [10].

On the other hand, a Walker constellation pattern is defined by 4 orbital planes each with 4 satellites. In comparison with along-track, this constellation provides wider coverage at the expense of reduced ISLs communication time. Indeed, as shown in Figure 6, contacts among satellites are only feasible when orbital planes cross. This setup is known to provide efficient communication opportunities [69] and can also be used for DTN studies.

To the best of the authors' knowledge, this is the first time these topologies have been compared within the DTN communication paradigm.

The chosen constellations are suitable for Earth observation missions [10], data-collection, or high-latency communication systems [11]. If used for Earth observation, the ground target locations would represent points of interest from which on-board instrumentation can acquire optical or radar images, or other remote sensing data. If used for data-collection or high-latency communication systems, ground targets would stand for ground-based equipment relaying either science or local user data (a.k.a. cold spots). In both cases, data sent from ground targets would be addressed via orbiting satellites to a centralized Mission Operations and Control (MOC) reachable through Internet (i.e., any of the 6 ground stations). Indeed, a MOC could act as an Internet gateway to deliver traffic to otherwise inaccessible ground targets. As a result, the publish/subscribe traffic pattern analyzed in the simulation is bidirectional: from all ground targets to the MOC and in reverse. In particular, each ground target will generate a bundle of 125000 Bytes per hour to be delivered to the MOC. In turn, the MOC will send one bundle of equal characteristics to each ground target also every hour. In both cases, traffic generation will only occur during the first 10 hs of the 24 hs of simulation. Therefore,

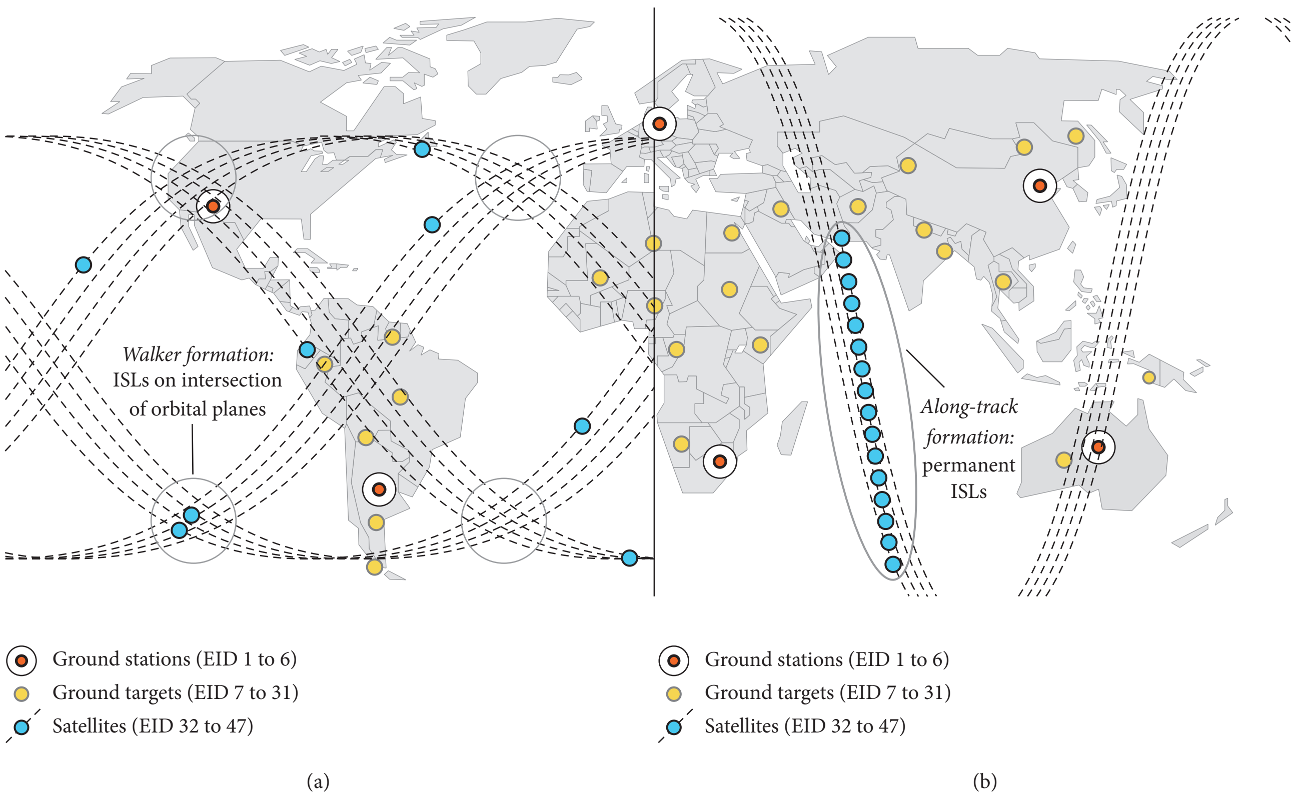
Walker formation: ISLs on intersection of orbital planes
Along-track formation: permanent ISLs
Ground stations (EID 1 to 6)
Ground targets (EID 7 to 31)
Satellites (EID 32 to 47)
(a)
(b)

Figure 6: Simulation case studies: (a) Walker formation and (b) along-track formation.

250 bundles will flow to the MOC and 250 to the ground targets in the return path. Also, the transmission data-rates for both intersatellite and Earth-to-satellite links were set to 100 Kbps, which can be obtained from a state-of-the-art CubeSat transponder (a CubeSat is a type of miniaturized satellite that is made up of multiples of $10 \times 10 \times 11.35$ cm cubic units [70]; CubeSats are gaining increased popularity as they leverage existing Commercial Off-The-Shelf (COTS) components providing a cost-efficient alternative to building distributed satellite constellations) [71].

Simulation results are plotted in Figure 7. The abscissa axis shows the variation in MTTF including an infinite (INF) value standing for a single simulation execution without the occurrence of failures. Indeed, measurements at this last point in the horizontal axis stand for a reference performance for each of the proposed constellations. For the rest of MTTF values (MTTF from 100 s up to 2200 s were considered with a step of 300 s), resulting metrics are averaged over 160 simulation runs and then averaged over bundles or nodes accordingly. On the other hand, the mean recovery time (MTTR) was set to 5 minutes mimicking a full system reboot. Failures were enabled only in orbiting satellites; ground stations and ground targets were not assumed to fail in this analysis.

The effective fail time curve on the left shows the accumulated amount of time that a given node in the network refrained from transmitting a bundle either because of a local or a remote (i.e., next-hop node) fault condition. In general, the along-track formation exhibits a higher effective fail rate which is consistent with the higher connectivity of a permanently connected constellation. In other words, the probability of finding a failure during a contact is higher in the along-track than in the Walker system.

A not so intuitive result is shown in the total bundles received curve. This metric measures the quantity of bundles that arrived at the final destination (application layer). The Walker constellation is able to deliver the full traffic load of 500 bundles for MTTF of 700 s and higher. However, the along-track formation is unable to deliver such load even without any failures injected in the nodes (453 bundles reach the destination in this case). After a thorough analysis of the simulation traces, it was found that a pathological routing behavior impeded CGR to find all feasible routes leaving certain bundles without valid routes (i.e., in the limbo). These miscalculations are magnified as the failure rate increases. The issue found in CGR statement is related to the anchoring concept and is discussed in detail in Section 4.

The mean bundle delay curve is shown with the absolute mean delay value measured in minutes as well as relative to the metric observed without faults in the network (i.e., the delay for infinite MTTF). These results confirm that the Walker formation provides a better overall bundle delivery time. However, the along-track system results significantly more stable with the variation of the MTTF. In other words, the along-track formation is less sensitive (i.e., more robust) to faults than the Walker constellation. This effect is clearly observed in the relative expression of the mean bundle delay. It is interesting to note that given the delay-tolerant nature

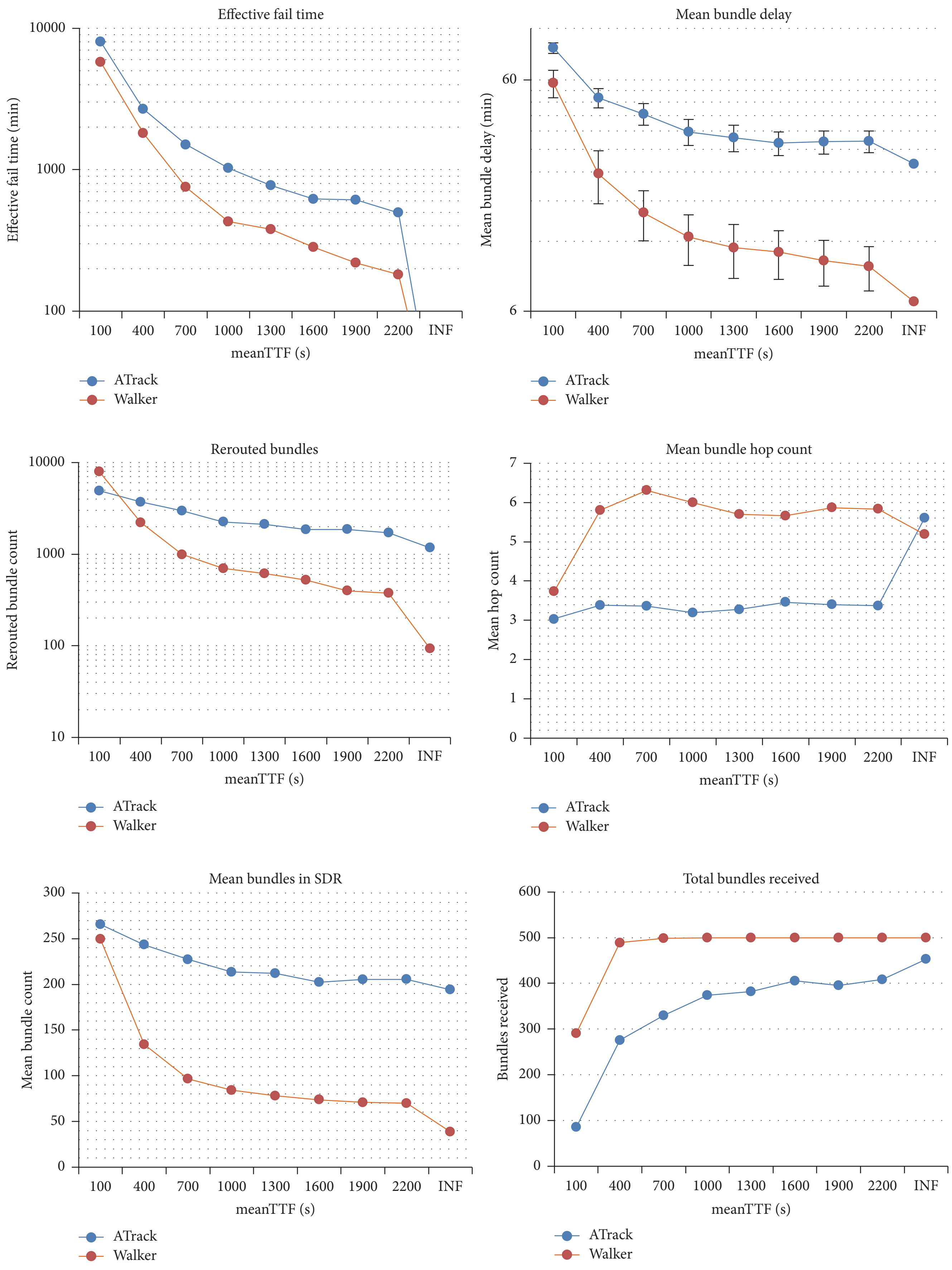

Figure 7: Continued.

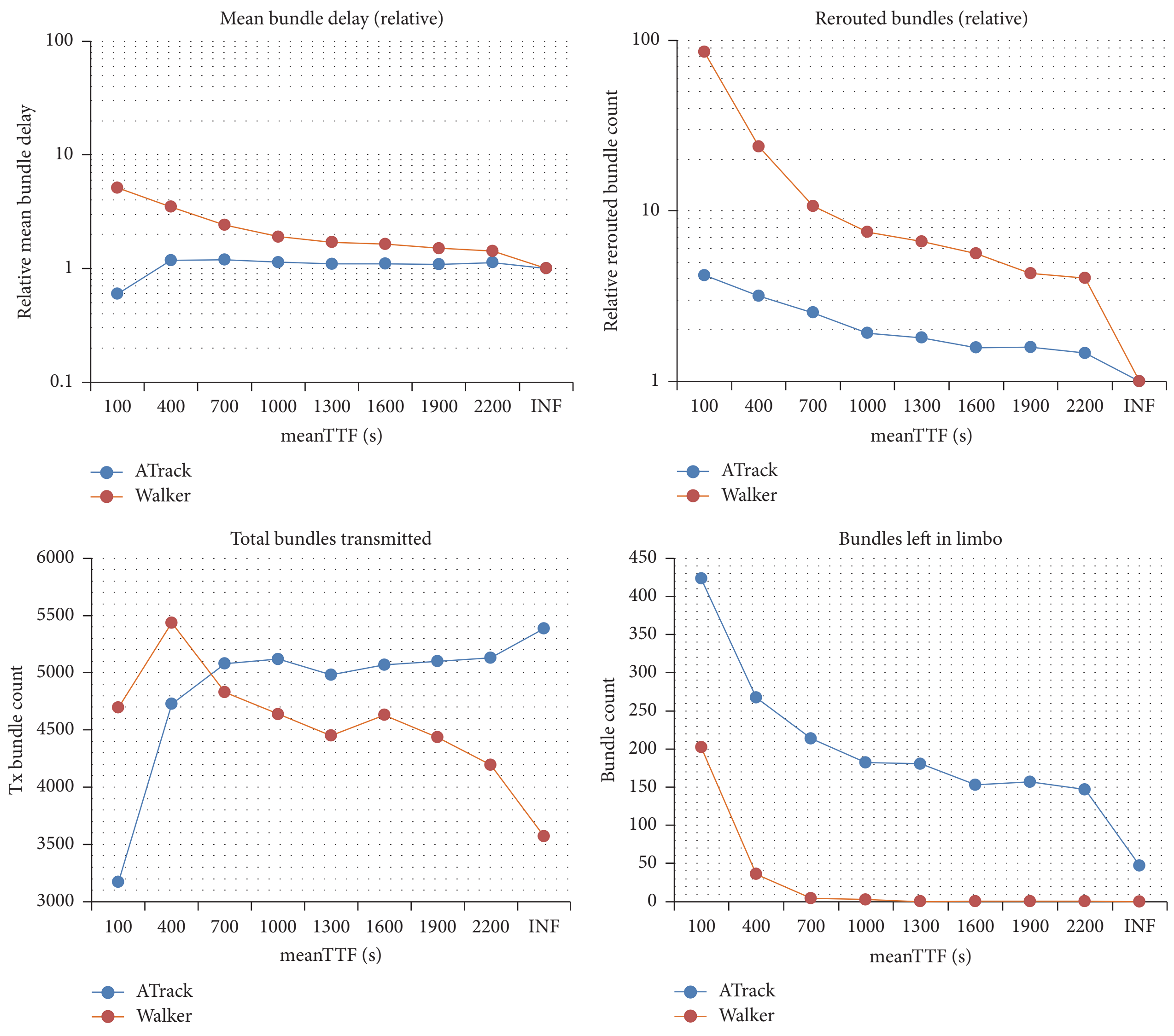

FIGURE 7: Simulation results.

of both systems, the fact that latency remains in the order of minutes is completely reasonable.

The rerouted bundles curves are also expressed in absolute and relative format. These results show the number of bundles that had to be routed while in transit because a contact ended with a nonempty outbound queue. As previously discussed, this happens when the capacity allocated by the first CGR execution did not match the real capacity in the system. Successive CGR executions are necessary either because of failures or congestion problems [49–51]. Indeed, the result without failures (infinite MTTF) shows the quantity of rerouting required due to congestion in both constellations. The relative curve thus evidences that rerouting in Walker constellation increases more dramatically than the along-track in the presence of failures. Such an increase can also be justified by the higher delivery rate of this formation. This metric then confirms that the along-track formation results more insensible towards higher fault rates.

The mean bundle hop count curve shows that the Walker constellation makes a more extensive usage of multihop paths. Such a feature is only observed in the along-track system in the absence of failures. For all other cases, the along-track constellation uses 3 hop paths (MOC to ground station, ground station to satellite, and satellite to ground target) meaning that several ISLs opportunities are underutilized in the presence of failures. Moreover, the along-track system evidences higher total transmitted bundles in most cases. This curve measures the total quantity of bundles transmitted by all nodes in the network (either for local or remote traffic). This metric can thus be directly correlated with energy usage in the node transmitter. In this case, the along-track not only uses lower hop counts but also evidences a higher energy usage due to bundle transmissions.

The last two plots on the right provide a metric on the storage used by the nodes. The upper curve depicts the overall spacecraft data recorder (SDR) memory used (outbound queue buffers occupancy) while the lower one shows the

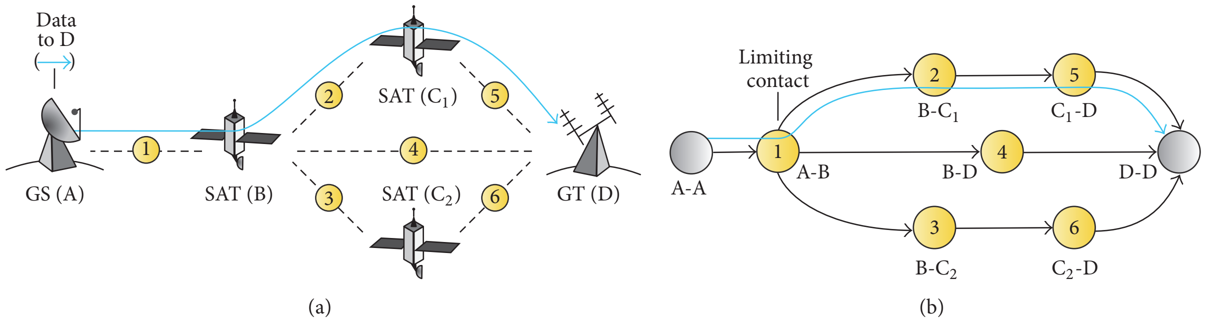

Figure 8: Routes overlooked by CGR.

quantity of bundles that remains in the limbo at the end of the simulations because of the absence of feasible routes. In general, a significant number of bundles remains in the limbo for the along-track formation. Also, the Walker constellation evidences a lower memory utilization. This is coherent with previous analysis and confirms the inability of CGR of finding all feasible routes in the along-track topology. Such a problem is not observed in the Walker system which also features a significantly lower storage utilization as shown in the bundles in SDR plot.

4. Discussion

In order to describe the CGR inability to find all feasible routes in the along-track constellation, a simplified along-track topology and its corresponding contact graph are presented in Figures 8(a) and 8(b), respectively. In contrast to the Walker formation, the along-track formation exhibits a remarkable link redundancy since a single point on Earth is able to simultaneously reach several orbiting nodes while each satellite has simultaneous access to the front and back neighbor. In this example, a ground station (GS A) can reach a satellite (SAT B) which in turn can reach the front and back satellites in the constellation (SAT C_1 and SAT C_2). Eventually, the three satellites will be able to reach a ground target (GT D) when the constellation passes over that ground area (notice that contacts 4, 5, and 6 might occur later in time requiring a temporal storage of data in the satellites).

Even though the existence of several paths towards D is desirable from a reliability perspective, it requires a routing algorithm that is able to discover and manage several parallel paths towards a given destination. However, the CGR specification discussed in Section 2.1 was found to overlook several valid paths in this kind of scenarios. In this example, the first Dijkstra search executed in node A will find a valid route towards D, let us say via contacts 1, 2, and 5 (cyan arrow). The limiting contact in this path is contact 1, which will end before the ISLs which are permanent in an along-track formation. As previously discussed in Algorithm 3, this contact will then be removed from the contact graph in order to begin the next search. But removing contact 1 will hinder the discovery of other two feasible paths via contacts 1 and 4, and contacts 1, 3, and 6. Therefore, node A will refrain from sending more bundles than those that can fit in the discovered route (as discussed in Algorithm 2, route residual capacity must be enough to accommodate the forwarded bundle). The capacity in the remaining routes will remain underutilized favoring a partial delivery of bundles, increased delay, congestion, and an increased storage utilization as evidenced in Figure 7. It is worth noticing that this issue is not an implementation matter but part of the inner core of current CGR definition.

At the moment, the authors are investigating alternative DTN routing algorithms leveraging a contact graph. As part of these efforts, an alternative definition of the CGR algorithm is under development that can enhance current route discovery capabilities. A generalization of the anchoring concept might lead to a complete route discovery. Also, relying on K-Shortest Path (KSP) algorithms is also being considered [72]. Another future work involves the investigation of novel mechanisms that could exploit MTTF and MTTR parameters (generally known or estimated in advance) to support proactive fault-avoidance forwarding measurements. This research line might be based on similar approaches to the ones used in opportunistic CGR [48]. Finally, the presented analysis can be further extended by modeling high channel delays and more complex custody transfer protocols. In real networks, the lack of a custody acceptance is the means by which a DTN node can realize that a neighbor is unresponsive. Even though disregarded in presented simulations, timeout configurations of custody messaging and significant channel delays can play a significant role in a correct and timely reaction towards network faults.

5. Conclusion

In this work, an extensive analysis on the reliability of satellite-based Delay/Disruption Tolerant Networks (DTN) was presented. Two appealing and realistic Low-Earth Orbit constellations using state-of-the art routing algorithms were considered and compared by means of simulations. To this end, a unique overview of Contact Graph Routing (CGR) was provided and implemented in DtnSim, a novel space DTN simulator provided to the DTN community.

Results include the first evidence of the performance of Walker and along-track formations under different failure rates. As expected, the higher the failure rate, the more significant the performance degradation. The Walker formation proved to provide better delivery and resource utilization metrics, while the along-track was found more insensitive (i.e., robust) towards faults. Even though the intrinsic redundancy present in the along-track topology favors an improved fault-tolerance, analysis showed that current version of CGR was unable to make an optimal utilization of the communication resources. A proper identification of the algorithm weakness was presented and discussed based on the detailed overview of CGR.

The presented analysis becomes a solid starting point towards an improved CGR statement, which is currently under study by the authors. Also, future work includes the exploration of further CGR enhancements that could improve its robustness when implemented in fault-prone DTN systems.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

Part of this research was carried out in the frame of the "Calcul Parallèle pour Applications Critiques en Temps et Sûreté" (CAPACITES) project. Part of this research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Government sponsorship is acknowledged.

References

[1] M. D'errico, *Distributed Space Missions for Earth System Monitoring*, vol. 31, Springer, New York, NY, USA, 2013.

[2] H. F. Rashvand, A. Abedi, J. M. Alcaraz-Calero, P. D. Mitchell, and S. C. Mukhopadhyay, "Wireless sensor systems for space and extreme environments: a review," *IEEE Sensors Journal*, vol. 14, no. 11, pp. 3955–3970, 2014.

[3] O. Brown and P. Eremenko, "The value proposition for fractionated space architectures," in *Proceedings of the AIAA-2006-7506*, AIAA Space, San Jose, Calif, USA, 2006.

[4] M. Mosleh, K. Dalili, and B. Heydari, "Optimal modularity for fractionated spacecraft: the case of system F6^a," *Procedia Computer Science*, vol. 28, pp. 164–170, 2014.

[5] S. Burleigh, A. Hooke, L. Torgerson et al., "Delay-tolerant networking: an approach to interplanetary internet," *IEEE Communications Magazine*, vol. 41, no. 6, pp. 128–136, 2003.

[6] C. Caini, H. Cruickshank, S. Farrell, and M. Marchese, "Delay- and disruption-tolerant networking (DTN): an alternative solution for future satellite networking applications," *Proceedings of the IEEE*, vol. 99, no. 11, pp. 1980–1997, 2011.

[7] C. Caini and R. Firrincieli, "DTN for LEO satellite communications," in *Proceedings of the International Conference on Personal Satellite Service*, pp. 186–198, Springer, Berlin, Germany, 2011.

[8] C. Caini and R. Firrincieli, "Application of contact graph routing to LEO satellite DTN communications," in *Proceedings of the IEEE International Conference on Communications, ICC*, pp. 3301–3305, IEEE, Ottawa, Canada, June 2012.

[9] P. G. Madoery, J. A. Fraire, and J. M. Finochietto, "Analysis of communication strategies for earth observation satellite constellations," *IEEE Latin America Transactions*, vol. 14, no. 6, pp. 2777–2782, 2016.

[10] J. A. Fraire, P. G. Madoery, J. M. Finochietto, P. A. Ferreyra, and R. Velazco, "Internetworking approaches towards along-track segmented satellite architectures," in *Proceedings of the International Conference on Wireless for Space and Extreme Environments, WiSEE*, pp. 123–128, IEEE, Aachen, Germany, September 2016.

[11] S. C. Burleigh and E. J. Birrane, "Toward a communications satellite network for humanitarian relief," in *Proceedings of the 1st International Conference on Wireless Technologies for Humanitarian Relief, ACWR*, pp. 219–224, ACM, Kerala, India, December 2011.

[12] V. Cerf, S. Burleigh, A. Hooke et al., "Delay-tolerant networking architecture," Internet Requests for Comments, RFC Editor 4838, 2007.

[13] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM*, pp. 27–34, ACM, New York, NY, USA, August 2003.

[14] K. Scott and S. Burleigh, "Bundle protocol specification," Internet Requests for Comments, RFC Editor 5050, 2007.

[15] M. Demmer, J. Ott, and S. Perreault, "Delay-tolerant networking tcp convergence-layer protocol," Internet Requests for Comments, RFC Editor 7242, 2014.

[16] H. Kruse, S. Jero, and S. Ostermann, "Datagram convergence layers for the delay- and disruption-tolerant networking (DTN) bundle protocol and licklider transmission protocol (LTP)," Internet Requests for Comments, RFC Editor 7122, 2014.

[17] S. Burleigh, "Compressed bundle header encoding (CBHE)," Internet Requests for Comments, RFC Editor 6260, 2011.

[18] N. Bezirgiannidis, F. Tsapeli, S. Diamantopoulos, and V. Tsaoussidis, "Towards flexibility and accuracy in space DTN communications," in *Proceedings of the 8th ACM MobiCom Workshop on Challenged Networks, CHANTS*, pp. 43–48, New York, NY, USA, September 2013.

[19] M. A. Alfonzo, J. A. Fraire, E. Kocian, and N. Alvarez, "Development of a DTN bundle protocol convergence layer for spacewire," in *Proceedings of the 2nd IEEE Biennial Congress of Argentina, ARGENCON 2014*, pp. 770–775, IEEE, Bariloche, Argentina, June 2014.

[20] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 3, pp. 19-20, 2003.

[21] B. B. Xuan, A. Ferreira, and A. Jarry, "Computing shortest, fastest, and foremost journeys in dynamic networks," *International Journal of Foundations of Computer Science*, vol. 14, no. 2, pp. 267–285, 2003.

[22] S. Merugu, M. Ammar, and E. Zegura, "Routing in space and time in networks with predictable mobility," Tech. Rep., Georgia Institute of Technology, 2004.

[23] M. Sheng, G. Xu, and X. Fang, "The routing of interplanetary internet," *China Communications*, vol. 3, no. 6, pp. 63–73, 2006.

[24] A. Lindgren, A. Doria, E. Davies, and S. Grasic, "Probabilistic routing protocol for intermittently connected networks," Internet Requests for Comments, RFC Editor 6693, 2012.

[25] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: routing for vehicle-based disruption-tolerant networks," in *Proceedings of the 25th IEEE International Conference on Computer Communications, INFOCOM*, pp. 1–11, IEEE, Barcelona, Spain, April 2006.

[26] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the ACM SIGCOMM Workshop on Delay-Tolerant Networking, WDTN*, pp. 252–259, ACM, Philadelphia, Pa, USA, August 2005.

[27] Z. Feng and K.-W. Chin, "A unified study of epidemic routing protocols and their enhancements," in *Proceedings of the 26th IEEE International Parallel and Distributed Processing Symposium Workshops & PhD Forum, IPDPSW*, pp. 1484–1493, IEEE, Shanghai, China, May 2012.

[28] S. Palazzo, A. T. Campbell, and M. Dias De Amorim, "Opportunistic and delay-tolerant networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, Article ID 164370, 2011.

[29] Y. Wu, S. Deng, and H. Huang, "Performance analysis of epidemic routing in DTNs with limited forwarding times and selfish nodes," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 13, no. 3-4, pp. 254–263, 2013.

[30] G. Araniti, N. Bezirgiannidis, E. Birrane et al., "Contact graph routing in DTN space networks: overview, enhancements and performance," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 38–46, 2015.

[31] "DTN2: a DTN reference implementation," https://sourceforge.net/projects/dtn/.

[32] S. Burleigh, "Interplanetary overlay network an implementation of the DTN bundle protocol," in *Proceedings of the 4th Annual IEEE Consumer Communications and Networking Conference, CCNC*, pp. 222–226, IEEE, Las Vegas, Nev, USA, January 2007.

[33] "Interplanetary Overlay Network (ION) software home page," https://sourceforge.net/projects/ion-dtn/.

[34] S. Schildt, J. Morgenroth, W.-B. Pöttner, and L. Wolf, "Performance comparison of DTN Bundle Protocol implementations," in *Proceedings of the 6th ACM workshop on Challenged networks*, pp. 61–64, ACM, Las Vegas, Nev, USA, September 2011.

[35] M. Feldmann and F. Walter, "Micro pcn; a bundle protocol implementation for microcontrollers," in *Proceedings of the International Conference on Wireless Communications and Signal Processing, WCSP*, pp. 1–5, IEEE, Nanjing, China, October 2015.

[36] W. Ivancic, W. M. Eddy, D. Stewart et al., "Experience with delay-tolerant networking from orbit," in *Proceedings of the 4th Advanced Satellite Mobile Systems, ASMS*, pp. 173–178, IEEE, Bologna, Italy, August 2008.

[37] J. Wyatt, S. Burleigh, R. Jones, L. Torgerson, and S. Wissler, "Disruption tolerant networking flight validation experiment on NASA's EPOXI mission," in *Proceedings of the 1st International Conference on Advances in Satellite and Space Communications, SPACOMM*, pp. 187–196, IEEE, Colmar, France, July 2009.

[38] "NASA and ESA use experimental interplanetary internet to test robot from international space station," https://www.nasa.gov/home/hqnews/2012/nov/HQ_12-391_DTN.html.

[39] Internet Engineering Task Force (IETF), "Delay tolerant network research group (DTNRG)," http://www.dtnrg.org/wiki/Home.

[40] Internet Society (ISOC), "InterPlanetary Networking Special Interest Group (IPNSIG) Chapter," http://ipnsig.org.

[41] Consultative Committee for Space Data Systems (CCSDS), "Delay tolerant networking working group (SIS-DTN)," http://www.ccsds.org.

[42] Internet Engineering Task Force (IETF), "Delay tolerant networking working group (DTNWG)," https://datatracker.ietf.org/wg/dtnwg/charter/.

[43] M. Tanase and V. Cristea, "Quality of service in large scale mobile distributed systems based on opportunistic networks," in *Proceedings of the IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA*, pp. 849–854, IEEE, Singapore, March 2011.

[44] J. A. Fraire and P. A. Ferreyra, "Assessing DTN architecture reliability for distributed satellite constellations: preliminary results from a case study," in *Proceedings of the IEEE Biennial Congress of Argentina, ARGENCON*, pp. 564–569, IEEE, Bariloche, Argentina, June 2014.

[45] Y. Wu, Z. Yang, and Q. Zhang, "A Novel DTN routing algorithm in the geo-relaying satellite network," in *Proceedings of the 11th International Conference on Mobile Ad-Hoc and Sensor Networks, MSN*, pp. 264–269, IEEE, Shenzhen, China, December 2015.

[46] S. E. Alaoui and B. Ramamurthy, "Routing optimization for DTN-based space networks using a temporal graph model," in *Proceedings of the IEEE International Conference on Communications, ICC*, pp. 1–6, IEEE, Kuala Lumpur, Malaysia, May 2016.

[47] C. Secretariat and CCSDS, "Schedule-Aware Bundle Routing (SABR), White Book," ccsds 232.0-b-2ed, Recommendation for Space Data System Standards, 2016.

[48] S. Burleigh, C. Caini, J. J. Messina, and M. Rodolfi, "Toward a unified routing framework for delay-tolerant networking," in *Proceedings of the IEEE International Conference on Wireless for Space and Extreme Environments, WiSEE*, pp. 82–86, IEEE, Aachen, Germany, September 2016.

[49] E. J. Birrane, "Congestion modeling in graph-routed delay tolerant networks with predictive capacity consumption," in *Proceedings of the IEEE Global Communications Conference, GLOBECOM*, pp. 3016–3022, IEEE, Atlanta, Ga, USA, December 2013.

[50] H. Yan, Q. Zhang, and Y. Sun, "Local information-based congestion control scheme for space delay/disruption tolerant networks," *Wireless Networks*, vol. 21, no. 6, pp. 2087–2099, 2015.

[51] J. A. Fraire, P. Madoery, J. M. Finochietto, and E. J. Birrane, "Congestion modeling and management techniques for predictable disruption tolerant networks," in *Proceedings of the IEEE 40th Conference on Local Computer Networks, LCN*, pp. 544–551, IEEE, Clearwater Beach, Fla, USA, October 2015.

[52] J. Fraire, P. Ferreyra, and C. Marques, "Opencl-accelerated simplified general perturbations 4 algorithm," in *Proceedings of the 14th Argentine Symposium on Technology, AST*, vol. 2013.

[53] S. Burleigh, "Contact graph routing, IETF-Draft draft-burleigh-dtnrg-cgr-01," 2009.

[54] J. Segui, E. Jennings, and S. Burleigh, "Enhancing contact graph routing for delay tolerant space networking," in *Proceedings of the IEEE Global Telecommunications Conference, GLOBECOM*, pp. 1–6, IEEE, Kathmandu, Nepal, December 2011.

[55] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[56] E. Birrane, S. Burleigh, and N. Kasch, "Analysis of the contact graph routing algorithm: bounding interplanetary paths," *Acta Astronautica*, vol. 75, pp. 108–119, 2012.

[57] J. A. Fraire, P. Madoery, and J. M. Finochietto, "Leveraging routing performance and congestion avoidance in predictable delay tolerant networks," in *Proceedings of the International IEEE Conference on Wireless for Space and Extreme Environments, WiSEE*, pp. 1–7, IEEE, Noordwijk, Netherlands, October 2014.

[58] N. Bezirgiannidis, C. Caini, D. D. P. Montenero, M. Ruggieri, and V. Tsaoussidis, "Contact graph routing enhancements for delay tolerant space communications," in *Proceedings of the 7th Advanced Satellite Multimedia Systems Conference, ASMS and the 13th Signal Processing for Space Communications Workshop, SPSC*, pp. 17–23, IEEE, Livorno, Italy, September 2014.

[59] R. R. Velazco, P. Fouillat, and R. Reis, *Radiation Effects on Embedded Systems*, Springer, Secaucus, NJ, USA, 2007.

[60] E. Ibe, H. Taniguchi, Y. Yahagi, K.-I. Shimbo, and T. Toba, "Impact of scaling on neutron-induced soft error in SRAMs from a 250 nm to a 22 nm design rule," *IEEE Transactions on Electron Devices*, vol. 57, no. 7, pp. 1527–1538, 2010.

[61] P. A. Ferreyra, C. A. Marqués, R. T. Ferreyra, and J. P. Gaspar, "Failure map functions and accelerated mean time to failure tests: new approaches for improving the reliability estimation in systems Exposed to single event upsets," *IEEE Transactions on Nuclear Science*, vol. 52, no. 1, pp. 494–500, 2005.

[62] H. Hecht, *Systems Reliability and Failure Prevention*, Artech House Inc, Norwood, Mass, USA, 2004.

[63] P. A. Ferreyra, G. Viganotti, C. A. Marqués, R. Velazco, and R. T. Ferreyra, "Failure and coverage factors based Markoff models: a new approach for improving the dependability estimation in complex fault tolerant systems exposed to SEUs," *IEEE Transactions on Nuclear Science*, vol. 54, no. 4, pp. 912–919, 2007.

[64] A. Keranen, J. Andott, and T. Karkkainen, "The ONE simulator for DTN protocol evaluation," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques, Simutools*, pp. 1–10, ICST, Rome, Italy, March 2009.

[65] J. Ahrenholz, "Comparison of CORE network emulation platforms," in *Proceedings of the Military Communications Conference, MILCOM*, pp. 166–171, IEEE, San Jose, Calif, USA, November 2010.

[66] A. Varga, "Using the OMNeT++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference, ESM*, Pargue, Czech, 2013.

[67] "AGI Systems Tool Kit (STK)," http://www.agi.com/STK.

[68] Z. Yoon, W. Frese, A. Bukmaier, and K. Brieß, "System design of an S-band network of distributed nanosatellites," *CEAS Space Journal*, vol. 6, no. 1, pp. 61–71, 2014.

[69] C. Wang, "Structural properties of a low Earth orbit satellite constellation—the Walker delta network," in *Proceedings of the Military Communications Conference, MILCOM*, vol. 3, pp. 968–972, IEEE, Boston, Mass, USA, October 1993.

[70] The CubeSat Program Cal Poly SLO, "CubeSat Design Specification rev. 13," 2014.

[71] "GomSpace NanoCom S100 Communication Module (NanoCom U482C)," http://gomspace.com/?p=products-u482c.

[72] J. Y. Yen, "Finding the *K* shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.