POWVER

Technical Report 2020-12

Title: **Routing in the Space Internet: A contact graph routing tutorial**

Authors: Juan A.Fraireauthor, Scott C. Burleigh

Report Number: 2020-12

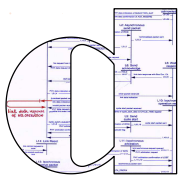
ERC Project: Power to the People. Verified.

ERC Project ID: 695614

Funded Under: H2020-EU.1.1. – EXCELLENT SCIENCE

Host Institution: Universität des Saarlandes, Dependable Systems and Software
Saarland Informatics Campus

Published In: J. Netw. Comput. Appl. 174



POWER TO THE PEOPLE.
VERIFIED.
erc

Routing in the Space Internet: A Contact Graph Routing Tutorial

Juan. A. Fraire*†, Olivier De Jonckère‡, Scott C. Burleigh§
*CONICET - Univerisidad Nacional de Córdoba, Córdoba, Argentina
†Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
‡Technische Universität Dresden, Dresden, Germany
§Jet Propulsion Laboratory (JPL), California Institute of Technology, USA

***Abstract*—A Space Internet is possible, as long as the *delay* and *disruption* challenges imposed by the space environment are properly tackled. Because these conditions are not well addressed by terrestrial Internet, more capable Delay-Tolerant Networking (DTN) protocols and algorithms are being developed. In particular, the principles and techniques for routing among ground elements and spacecraft in near-Earth orbit and deep-space are enacted in the Contact Graph Routing (CGR) framework. CGR blends a set of non-trivial algorithm adaptations, space operations concepts, time-dynamic scheduling, and specific graph models. The complexity of that framework suggests a need for a focused discussion to facilitate its direct and correct apprehension. To this end, we present an in-depth tutorial that collects and organizes first-hand experience on researching, developing, implementing, and standardizing CGR. Content is laid out in a structure that considers the planning, route search and management, and forwarding phases bridging ground and space domains. We rely on intuitive graphical examples, supporting code material, and references to flight-grade CGR implementations details where pertinent. We hope this tutorial will serve as a valuable resource for engineers and that researchers can also apply the insights presented here to topics in DTN research.**

***Index Terms*—Contact Graph Routing; Schedule Aware Bundle Routing; Delay-Tolerant Networks.**

I. INTRODUCTION

The autonomous transmission of information through Internet has changed life on Earth, and bringing these connectivity advantages into space is essential in the context of a thriving *new space* era [1]. During the first decade of the 21st century, the space community put forward enabling technologies such as formation flying [2] and efficient inter-satellite communications [3]. Most of these efforts were supported by government agencies, enabling significant reductions in components' weight, size, price, and acquisition lead time [4]. Since then, many private investors and companies have begun leveraging these advances by deploying networked constellations of small and nano-satellites [5]. While around 2,000 active satellites are currently in orbit, recent studies show that more than 50,000 are on the launch schedule for the next 10 years [6].

This revolution is also affecting missions beyond Earth-orbit. Nano-satellites based on Commercial off-the-shelf (COTS) components have been tested in interplanetary distances [7] and have proven valuable in missions to Mars [8], and others are scheduled for launch to the Moon [9]. Indeed, the new space trend is lowering the cost of access to deep-

Table I
LIST OF ACRONYMS

BDT	Best-case Delivery Time
BFS	Breadth First Search
CBHE	Compressed Bundle Header Encoding
CCSDS	Consultative Committee for Space Data Systems
CG	Contact Graph
CGR	Contact Graph Routing
CLA	Convergence Layer Adapter
CLEO	Cisco Router in LEO
COTS	Commercial off-the Shelf
CP	Contact Plan
CRP	Contact Review Procedure
CSP	Contact Selection Procedure
DINET	Deep Impact Network
DSN	Deep Space Network
DTLSR	Delay-Tolerant Link-State Routing
DTN	Delay Tolerant Network
ESA	European Space Agency
ETO	Earliest Transmission Opportunity
EVC	Estimated Volume Consumption
EVL	Effective Volume Limit
FBTX	First Byte Transmission time
GEO	Geostationary Earth-Orbit
GSL	Ground-to-Space Links
IETF	Internet Engineering Task Force
Inter-RR	Inter Region Routing
Intra-RR	Intra Region Routing
ION	Interplanetary Overlay Network
IP	Internet Protocol
IRTF	Internet Research Task Force
ISL	Inter-Satellite Links
JAXA	Japan Aerospace Exploration Agency
LBRX	Last Byte Reception time
LBTX	Last Byte Transmission time
LEO	Low-Earth Orbit
LTP	Licklider Transmission Protocol
MAV	Maximum Available Volume
MCS	Modulation and Coding Scheme
MDP	Markov Decision Process
MEO	Medium-Earth Orbit
OBC	On-Board Computer
OCGR	Opportunistic CGR
OWLT	One-Way Light Time
PAT	Projected Arrival Time
POSIX	Portable Operating System Interface
SABR	Schedule Aware Bundle Routing
TCP	Transfer Control Protocol
TTL	Time to Live
UDP	User Datagram Protocol
UK-DMC	United Kindom Disaster Monitoring Constellation
μPCN	Micro-Planetary Communication Network
USLP	Unified Space Data Link Protocol
UTCG	Gregorian Coordinated Universal Time

space, thus fostering the progress of larger scale missions such as the Lunar gateway, from which several networked assets will need to be managed [10].

The realization of a Space Internet that bridges ground, near-Earth and deep-space assets is imminent in this context, offering the operational advantages of extended connectivity times, enhanced reliability via multiple communication paths, and in-orbit resource and service sharing. However, fundamental environmental challenges need to be considered. In space flight, the orbital dynamics, the varying and very long-range communication distances, the effect of planet rotation, and on-board power restrictions result in prolonged episodes of disconnection. Furthermore, the propagation delay of signals in deep-space and interplanetary environments is generally in the order of minutes, hours or even days. The resulting *delay* and *disruption* conditions prevent the optimal operation of traditional Internet protocols, which are largely based on an effectively instantaneous flow of information between sending and receiver nodes. Instead, Delay Tolerant Networking (DTN), which assumes no instantaneous feedback but potentially lengthy storage of data in intermediate nodes, has been proposed to overcome these issues [11].

One of the most critical aspects of DTN architecture is the routing of data [12]. In DTN it is not possible to use traditional routing schemes based on stable connections. In particular, it is not enough to simply determine the next hop (next neighbor to send the traffic to) from an analysis of current network topology; it is necessary to decide *when* to send data depending on *when* it is expected to arrive via delayed and disrupted links. The consideration of the time dimension is an aspect that challenges any underlying routing algorithm. Fortunately, in space networks it is possible to know the future connectivity of the assets [13], which may be expressed in a *contact plan* defining the expected resources that the space network will have for transporting data. These conditions inspired the creation of new graph models, time-evolving abstractions, and algorithm adaptations of the Contact Graph Routing (CGR) framework. CGR is unique in the sense that it is the sole approach (of which the authors are aware) that integrates the set of techniques capable of coping with these challenges from a practical perspective. CGR is the most mature autonomous routing fabric for the forthcoming Space Internet. But CGR is complex. Its dynamics deserve particular attention and care.

This document provides an in-depth tutorial overview of CGR principles and processes. This work differs from [14] inasmuch as we survey the modeling and algorithmic aspects of CGR, rather than only the experimental and high-level technological achievements. Moreover, the presented content originates from authors' first-hand experience with CGR in (*i*) its implementation in flight-grade protocol stacks [15], [16], (*ii*) its standardization efforts for Internet and space operations [17]–[19], and (*iii*) extensive simulation and emulation research [20]. However, much of the published research on CGR assumes the reader is already knowledgeable on the subject, while standardization recommendations only provide high-level procedure descriptions and implementations are hard to interpret. These facts motivate this tutorial, which comprises consolidated support material for those interested in learning the specifics of CGR, whether for a correct implementation or research purposes. To aid in understanding the time dynamics, we rely on extensive graphical examples and provide an easy-to-follow Python-based CGR library (pyCGR) that realizes the algorithms presented here[1]. To the best of the authors' knowledge, this constitutes the most detailed instructive material available to date regarding CGR.

This paper is organized as follows. In Section II a *space networks overview* is provided, establishing the basic concepts, terminology and framework. A detailed formalization of the abstractions and *space network models* assumed by CGR is provided in Section III. Section IV describes the routine at the core of CGR: the *route search* algorithm based on Dijkstra's shortest path algorithm. The *route management* procedure is based on Yen's algorithm and is discussed in Section V. Section VI analyzes the *forwarding* phase, where the best route is selected for posterior data enqueuing. The current *trends outlook* for CGR is discussed in Section VII. Finally, *closing remarks* and future research challenges are summarized in Section VIII.

II. Space Networks Overview

Space networking is one of the principal applications of Delay-Tolerant Networking (DTN). The term "DTN" was introduced by a team of researchers led by Vint Cerf in 2003 to designate time-evolving networks lacking continuous and instantaneous end-to-end connectivity [11]. Since then, DTNs have drawn much attention from many researchers due to their applicability in a variety of domains including airborne networks [21], vehicular ad-hoc networks [22], mobile social networks [23], Internet of things [24], underwater networks [25], deep space [11], and near-Earth communication networks [26]. The same concepts and mechanisms devised to deal with the *delays* and *disruptions* of interplanetary communications can readily be applied to other communication domains characterized by long signal propagation time, frequent node occlusion, high node mobility, and/or reduced communication range and resources.

The time-evolving and partitioned nature of DTNs favors the representation of connectivity by means of *contacts*, a contact being an episode of time during which a node is able to transfer data to another node. The literature [27] classifies contacts as: (*i*) opportunistic: no assumptions can be made on future contacts, (*ii*) probabilistic: contact patterns can be inferred from history (e.g., social networks), and (*iii*) scheduled: contacts can be accurately predicted and documented in a *contact plan*. Space networks are characterized by scheduled contacts. In turn, we can distinguish between two classes of space networks:

- **Deep-space**: Networks of assets in an interplanetary [11] domain and beyond where links are *disrupted* by planet

[1]The parameters listed in Tables II, III, IV, as well as algorithms 1, 2, 3, 4, and 5 echo the naming and structure of the pyCGR library. Bundle, contact and route concepts conform to the SABR specification [17], unless noted otherwise. Because of the time dynamics of CGR, we strongly suggest the use of pyCGR along with this document in order to experiment and observe the debug traces and results. Example contact plans used in this paper and others are available in the pyCGR repository: https://bitbucket.org/juanfraire/pycgr.git.

occlusion and *delayed* because of signal propagation at light speed (which might render delays on the order of minutes, hours or days at planetary distances [28]).

- **Near-Earth**: Networks of satellites in low, medium or geostationary Earth orbits where communications are *disrupted* due to orbital dynamics, highly directive antenna orientation, or platform constraints (i.e., power limits). Signal propagation *delay* is not a predominant constraint in near-Earth scenarios [29]–[31].

A. Protocols and Procedures

High-latency links and network disruptions are addressed by the principles of *store-carry-and-forward* and *minimal end-to-end messaging exchange* for control or feedback. As detailed in the DTN architecture [27], these principles are addressed in a protocol at the *bundle layer* that sits above various stacks of protocols (at what is termed the *convergence layer*) suitable for transmission within the various communication environments that data must traverse on the end-to-end path from source to destination (e.g., deep-space protocols [32], proximity inter-satellite links [33], satellite-to-ground links [34], or TCP/IP on Internet). A key feature of the bundle layer protocol is the availability of persistent storage resources that enable DTN nodes to retain data locally while waiting for transmission opportunities to become available, whether that takes seconds, minutes, or days. This is in contrast to traditional Internet switch and router buffering where data are expected to leave the device immediately, on the order of milliseconds after arrival.

The Bundle Protocol, as specified by the Internet Research Task Force (IRTF) [35] and the Consultative Committee for Space Data Systems (CCSDS) [36], implements these mechanisms. Besides the encoding and block format of *bundles* (Bundle Protocol data units), the protocol addresses five unique features that shall be considered when routing data in space DTNs. The bundle data elements supporting these features are listed in Table II and are referenced throughout this paper.

(1) **Convergence layer adapters**: A CLA is the component that sends and receives bundles on behalf of the Bundle Protocol agent utilizing the services of protocol stacks that are well-suited to the local communication environment. When applicable, CLA responsibilities include session establishment, encoding adaptations, transfer of bundles using smaller maximum transmission unit (MTU) sizes, and others depending on the native protocol. CLAs for TCP, UDP, Licklider Transmission Protocol (LTP), Bluetooth, and raw Ethernet enable the integration with existing Internet and space networking infrastructures [14]. To account for CLA overhead, the estimated volume consumption ($B.EVC$) of a bundle B is defined in [17] as the bundle size plus a 3% margin or 100 Bytes, whichever the less.

(2) **Custody transfers**: In order to cope with unreliable lower layer protocols and congestion situations (storage depletion), an intermediate node can *take custody* of a bundle, i.e., assume responsibility for forwarding it. A custody acceptance signal from a remote node relieves the current custodian of the bundle from further responsibility for forwarding it. The custodian thus can release storage resources and use them to receive further data.

Table II
BUNDLE DATA ELEMENTS (RELEVANT TO ROUTING)

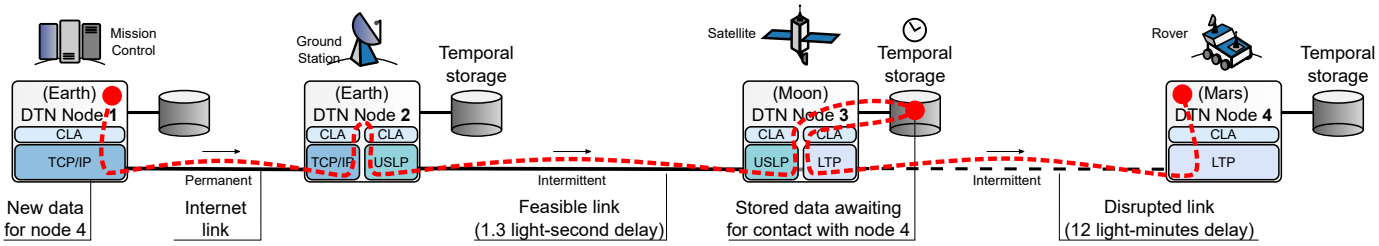
Bundle primary block parameters	
$B.src$ & $B.dst$	Source and destination nodes for the bundle
$B.size$	Bundle size, including both header and payload
$B.p$ or $priority$	Priority class of the bundle (1... p)
$B.critical$	Bundle critical flag
$B.custody$	Custody transfer requested flag
$B.fragment$	Fragmentation authorized flag
$B.deadline$	Expiration time of the bundle
Computed parameters	
$B.EVC$	Estimated volume consumption ($size * 1.03$) [17]
$B.sender$	Previous sender of the bundle, informed by CLA

(3) **Fragmentation**: Since a bundle carries all necessary data for a successful and complete information transfer to the destination, its byte-length can be arbitrarily long. While the largest UDP/IP protocol data unit is 64KB, there is virtually no size limit for a bundle, which can carry MBs or GBs of payload data. In this context, fragmentation can ensure a correct fit to time-bounded contacts. Fragmentation can be proactive (if it occurs prior to bundle transmission) and/or reactive (if the need for fragmentation is identified during bundle transmission).

(4) **Priorities**: The Bundle Protocol specification reserves class of service flags in the bundle's primary block (the first element of the bundle's "header") that can be mapped to unicast traffic priority levels, (1...P), where $P = 3$ is recommended in [17]. Also, an extended class of service mechanism has been proposed that, separately, enables a bundle to be marked as *critical*, indicating that copies of the data shall be forwarded through as many interfaces as possible to maximize the chance it will reach its destination.

(5) **Deadlines**: In order to prevent data from persisting for long periods of time and thus congesting nodes' storage resources, each bundle is configured with a *deadline* or expiration time. The deadline of a bundle is computed as its creation time plus its *lifetime* (also called "Time To Live" (TTL), but note that TTL in DTN is a time limit rather than a hop limit as in the Internet architecture). In particular, when the expiration time of a bundle is reached, the Bundle Protocol agent is authorized to remove that bundle from the network. The responsibility for setting a suitable deadline rests on the application originating the data.

Fig. 1 illustrates an interplanetary network example using Bundle Protocol as overlay for different protocol families via CLAs. DTN node 1, a mission control center located on Earth, sends data to a rover on Mars (DTN node 4). Since there is no direct communication between the source and destination nodes, data needs to go through the ground station (DTN node 2) to an intermediate DTN node 3 relay satellite orbiting the Moon. However, the visibility between nodes 3 and 4 does not allow establishing the link yet (e.g., rover is on the opposite side of Mars). Thus, DTN node 3's routing routine autonomously decides to retain in-transit data in its



	Mercury	Venus	Earth	Mars	Jupiter	Saturn	Uranus	Neptune
Avg. Distance to Sun (AU)	0,38	0.72	1.00	1.52	5.21	9.54	19.18	30.11
Latency	3 min	6 min	8 min	**12 min**	43 min	1h20	2h40	4h10

Figure 1. Store, carry and forward example in a space DTN. A node on Earth transmits data to a satellite orbiting the Moon, which stores the data until a long-range contact with Mars becomes available. Because of a) delays and b) disruptions, reliance on immediate feedback messages must be reduced or eliminated from DTNs. Since a light-second is defined as the distance that electromagnetic radiation travels through free space in one second, it is a unit of distance that corresponds trivially to a unit of time.

local storage until the contact with DTN node 4 becomes available, after the planet rotates. In such a case, DTN node 3 can become a *custodian* of the stored data in order to commit towards its successful delivery, authorizing the bundle removal from DTN node 2 storage. Also, if visibility episodes are insufficient for the transmission of a given bundle using a particular convergence layer protocol, the bundle can be *fragmented* at DTN node 3, node-2, or node 1. Urgent commands to the rover can use higher *priority* classes, important telemetry alarms can be reported to mission control via *critical* flagged bundles, and *TTL* can be set to the period during which the observations obtained from the rover instruments are relevant.

B. Implementations and Experiments

Several Bundle Protocol software stacks have been developed [37]. Interplanetary Overlay Network (ION) [15] and Micro-Planetary Communication Network (μPCN) [16] are two BP implementations that are specifically targeted for the space environment. While μPCN is lightweight software intended to operate in resource-constrained processors and micro-controllers, ION is a full-featured protocol stack including state-of-the-art CGR routing for space DTNs. Both rely on POSIX interfaces, can run on several embedded operating systems, and are available as open-source software [38], [39].

DTN protocols were successfully validated early in 2008 as part of the near-Earth spacecraft operations of the United Kingdom Disaster Monitoring Constellation (UK-DMC) [40]. These satellites' use of IP was enabled by earlier experiments with an on-board Cisco router in low Earth orbit (CLEO). Later, the Japanese space agency (JAXA) tested DTN and CGR in geostationary satellites [41] in 2012. Since 2018, ION has been in continuous operational use on the International Space Station (ISS) to automate the delivery of science payload data to investigators on Earth [42]. Most recently, μPCN was launched as part of ESA's LEO flying laboratory, OPS-SAT.

In the interplanetary domain, in 2008 NASA's DINET (Deep Impact Network) experiment successfully demonstrated the applicability of the ION stack and CGR over long-range deep-space links for 27 days (at distances between 40 and 80 light seconds) [43]. This remarkable experiment confirmed that DTN principles and CGR routing can pave the way toward autonomous networks in the interplanetary domain.

Due to its success, ION is now being integrated into NASA's Deep Space Network (DSN) protocol set; ION incorporates CGR at the core of the routing framework. Further DTN and CGR experiments are planned for the Lunar IceCube mission, targeted to launch in 2021 [9].

However, the study of routing in space networks requires validation over larger scale systems. Although DTN simulators and test-beds exist [44]–[46], most have been focused on opportunistic and probabilistic contacts. For scheduled DTNs, virtualization of the ION protocol stack enabled the emulation of large space network topologies [47]–[49], but those emulations are required to run in real-time. When CGR operation in a large system needs to be analyzed over a long period of time (i.e., days or weeks), accelerated simulation is preferred. For this purpose, DtnSim [20] includes a direct interface to the ION routing library that is adapted to run in simulation time. The CGR variants studied in this paper are also implemented in DtnSim, expressed in C++ language. The pyCGR library accompanying this tutorial is written in Python.

C. Routing Framework

Routing in scheduled space networks can be divided into three clearly differentiated but interrelated stages illustrated in Fig. 2: a) planning, b) routing and c) forwarding.

a) **Planning**: In the planning stage, a centralized entity (e.g., mission control) computes contact plans based on the estimation of future episodes of communications. This task involves orbital propagators that predict the physical disposition and orientation of nodes as well as missions' communication system models and configurations (antenna, modulation, transmission power, etc.). The resulting contact plan comprises the envelope within which network connectivity can occur. That plan can then be post-processed to accommodate operational plans (anticipated episodes of disconnection due to power management, body-fixed instrument pointing, etc.) and to enhance fairness [50], adapt to mandated routing [51],

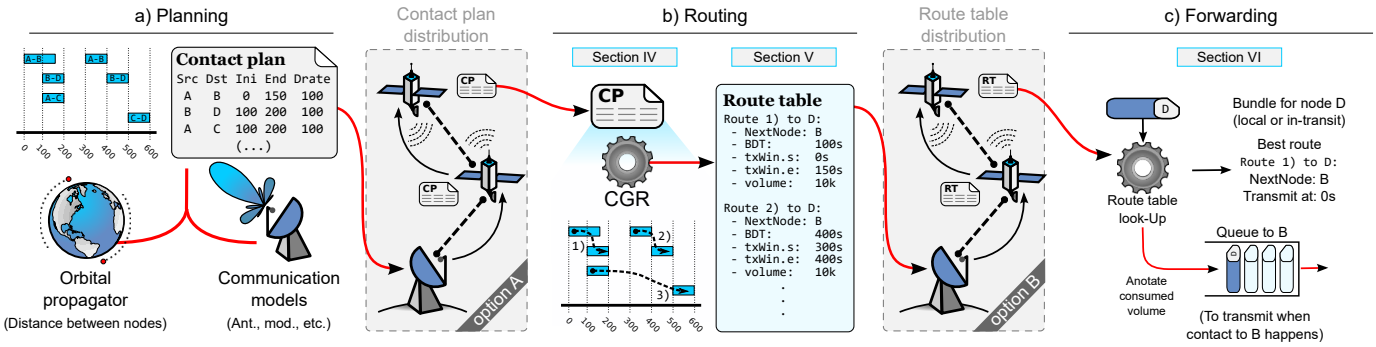


Figure 2. CGR routing framework in scheduled space DTNs. a) Orbital propagators and communication models allow the computation of a contact plan, which may be kept at a central control center or distributed to DTN nodes (option A). b) Routing uses CGR to compute a route table from the contact plan, which may be distributed to DTN nodes if computed in a central control center (option B). c) DTN nodes consult the route table to decide upon the proper outbound queue for forwarding outbound data.

accommodate known traffic flows [52], [53], mitigate congestion [54], reduce energy consumption [55], [56] or fit specific missions [57]. Contact plan design [58] is a distinct research area, out of scope of this tutorial. In the remaining of this paper, and from the routing perspective, the contact plan is assumed to represent simply the data transmission possibilities between network nodes.

At this point, the contact plan can be either distributed to the space DTN nodes (option A in Fig. 2) for a *distributed* routing computation, or kept at mission control for subsequent *centralized* routing calculation.

b) **Routing**: The contact plan serves as input for the routing routine with CGR sitting at its core. Algorithmic approaches for CGR, analyzed in detailed in Section IV, are leveraged to compute the paths to destinations in the network. The resulting routes not only identify which next hop node to forward the bundle to but also indicate the best delivery time (BDT), the route volume limit ($volume$), and the interval on which this route is valid for transmission (tx_win). Among others, these metrics are computed by CGR and stored in route tables. In the case that routes are computed in a centralized node, the resulting tables must be distributed to DTN nodes in timely fashion (option B in Fig. 2). The proper management of CGR route tables is discussed in Section V.

c) **Forwarding**: Finally, the forwarding process is responsible for selecting the best route, out of many on the route table. This selection considers local conditions that are only available at that forwarding moment such as local time, the size and priority of the data to be forwarded ($B.EVC$, $B.priority$), and current queue backlog conditions. The best route is thus expected to provide adequate resources for successful delivery of the bundle. Based on the selected route, the bundle is placed in the outbound queue to the node that is identified as the next hop on the route. Once in the queue, the bundle might be transmitted immediately or kept in storage until the next contact occurs. The specifics of the CGR forwarding procedure are examined in Section VI.

D. Contact Graph Routing

Routing on space networks requires precise determination of *when* and to *which* neighbor a given bundle should be forwarded. Because connectivity episodes and their respective propagation delays can be predicted and made available in advance, routing in space networks can be accurate and efficient although a complicated and challenging problem. The determinism and availability of information in space networks is in contrast with opportunistic and probabilistic DTNs, where routing is less certain but can be based on simpler flooding or probability inference methods [59]–[64]. Instead, the space network connectivity information encoded in the contact plan has to be efficiently processed to determine accurate candidate paths. Theoretical models (e.g., linear programming models) can ensure optimal results at the expense of processing [65], a fact that limits any practical value, especially when considering resource-constrained on-board computers. A further limit on linear programming models is the requirement for accurate information about the state of remote nodes, which is typically unavailable in a delay-tolerant network.

As a result, CGR has received increasing attention in recent years. CGR has been able to demonstrate sufficient accuracy and efficiency to become the *de facto* routing framework for space DTNs.

CGR was first mentioned as an interplanetary routing approach by S. Burleigh in 2008 [66]. IETF drafts were also proposed for CGR in 2009 [18] and 2010 [19]. By then, CGR was being periodically released with ION [15]. The first improvements came in 2011 with the proposal of source routing extensions by Dr. Edward Birrane in [67], [68], documented in an IETF draft [69]. In that same year, the adaptation of Dijkstra in CGR was introduced by Segui *et al.* in [70]. By profiting from a monotonically increasing time-related cost function, this contribution provided CGR with a solid algorithmic framework motivating Section IV of this document. In 2012, Dr. Birrane presented an extended vision of CGR including prevention of routing loops and multiple destination analysis [71]. At the same time, Dr. Carlo Caini *et al.* argued in favor of implementing CGR as a routing solution for near-Earth Low-Earth Orbit (LEO) satellite networks [29]–[31]. Similar studies followed afterwards [72]–[74], especially in the context of "ring-road" networking, an inexpensive data mule approach for relaying data to and from isolated networks [26], [75]–[77]. In 2014, Bezirgiannidis *et al.* made the first steps in modeling the

impact of traffic in bundle delivery time estimation together with overbooking management techniques [78], [79]. These methods are reviewed in detail in Section VI. At about the same time, Fraire *et al*. explored congestion mitigation by proper volume annotations in CGR routes combined with source routing updates [80], [81]. Araniti *et al*. summarized the advances and experimental experiences with CGR up to 2015 in [14], the most cited CGR article at the time of this writing. In 2016, Burleigh *et al*. proposed an opportunistic enhancement to CGR so that unplanned contacts could be correctly reacted upon and included in the routing decisions [82]. Dr. Ruhai Wang *et al*. then presented the first investigations into CGR scalability [83], which motivated later contributions by Madoery *et al*. via efficient forwarding [84] and region-based approaches [85], [86]. Initial CGR reliability studies followed in 2017. Dr. Juan Fraire *et al*. showed how CGR behaved under uncertain contact plans [87], [88], for which reliable CGR variations based on state-of-the-art computer science models were introduced [89]–[91]. As discussed in Section VII, scalability as well as uncertain and opportunistic CGR extensions are among the most active and promising research lines in CGR. In 2018, route table management strategies were also analyzed by Fraire *et al*. in [92]. It was from this contribution that Yen's algorithm became the default routing management approach for CGR in ION as described in Section V. In 2019, a spanning-tree formulation was proposed as a CGR alternative to compute routes to several destinations [93], and a partial queue information sharing was introduced in [94]. In that same year, the Schedule Aware Bundle Routing (SABR) recommended standard (blue book) was released by CCSDS recommending CGR as the routing procedure for the Solar System Internet [17]. As the development of this extensive ecosystem demonstrates, CGR has become something more than a simple algorithm. CGR is a comprehensive process for tackling the operation and management of a scheduled space DTN. As such, it is quite unique when compared with other routing approaches,

E. Comparison with Other Routing Algorithms

CGR operates on networks where local nodes can consult asserted knowledge of scheduled connectivity intervals as noted in a contact plan. CGR is the sole routing algorithm, to the best of the authors' knowledge, that aligns with the planning, routing and forwarding procedures of scheduled space flight communications. For this reason, virtually all deployments of DTN for space flight missions have adopted CGR. As discussed below, CGR is qualitatively superior to other DTN routing approaches that could be utilized in space communication environments.

On the one hand, *flooding* approaches such as epidemic [62] and spray-and-wait [95], [96] assume abundant link capacity and energy, enabling each node of the network to forward multiple copies of each bundle to multiple available neighbors. Although these approaches are popular because they effectively address the unscheduled connectivity of opportunistic terrestrial DTN communications, they are not applicable to space networks due to the limited bandwidth of space links and the severe constraints on satellites' power subsystems. On the other hand, *probabilistic* solutions such as MaxProp [60] and Prophet [64] attempt to infer the connectivity patterns of the network by means of metadata exchanges between the nodes at the beginning of each contact. These exchanges are infeasible over high-latency space links where round trip times can be on the order of seconds or minutes. While assuming Internet-like capacity and latency, flooding approaches and probabilistic routing are unable to take advantage of the time-varying topological information provided in a flight mission's contact plan as CGR does. Contact plan awareness enables CGR to compute well-informed accurate forwarding decisions that avoid dead-ends and unwanted congestion [89]

In contrast, topological information is central to the design of *delay-tolerant link-state* routing (DTLSR) [97]. However, in DTLSR that information is acquired automatically in the course of network operation rather than asserted during mission planning. Link state announcements are flooded throughout the network as it evolves over time. Each node uses these announcements to maintain a graph representing its current view of the topology of the network, and a shortest path computation is employed to find routes for messages. However, the effectiveness of DTLSR depends on the accuracy of that graph. The lengthy signal propagation delays of deep space communication increase the likelihood that the link state information on which routing decisions are based is out of date at the time the decisions are made.

The following sections cover these unique aspects at the core of CGR in detail.

III. Space Networks Model

This section details the model and abstractions on which CGR is based: contacts, contact plan, contact graph, routes, and volume considerations.

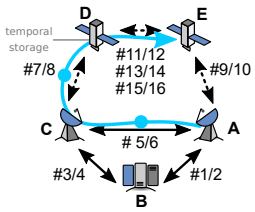
A. Contact

A contact $C_{A,B}^{t_1,t_2}, R$ is defined as a time interval $(t_1; t_2)$ during which it is expected that data will be transmitted by DTN node A (the contact's sending node) at rate R such that data will be received by node B (the contact's receiving node). The time values can be expressed either in absolute units (e.g., Gregorian Coordinated Universal Time, UTCG) or in relative time with respect to a reference epoch. In Fig. 3 a), a table lists each contact identified by a number ($\#1...16$). Contacts $C_{A,B}^{0,60}$, $C_{B,C}^{0,60}$ and $C_{A,C}^{0,60}$ represent permanent links (e.g., between mission control and ground stations connected through Internet). Contacts $C_{C,D}^{0,30}$ and $C_{A,E}^{10,20}$ stand for episodic Ground to Space Links (GSLs) while $C_{D,E}^{0,10}$, $C_{D,E}^{30,40}$ and $C_{D,E}^{50,60}$ identify episodic Inter-Satellite Links (ISLs).

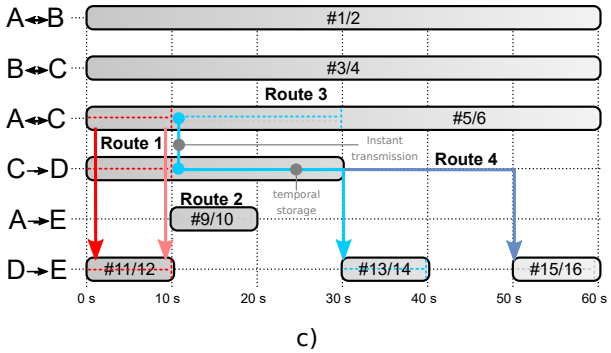
Each contact is characterized by a $start$ time, an end time, a data $rate$, and the identifiers of the sender (snd) and receiver (rcv) nodes. Nodes must be uniquely identified, nominally by unique node numbers as discussed in the specification for Compressed Bundle Header Encoding (CBHE) [98]. A contact's data $rate$ is the mean rate at which data is expected to be transmitted by the sending node throughout the indicated time period. In other words, the data $rate$ for a contact can

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

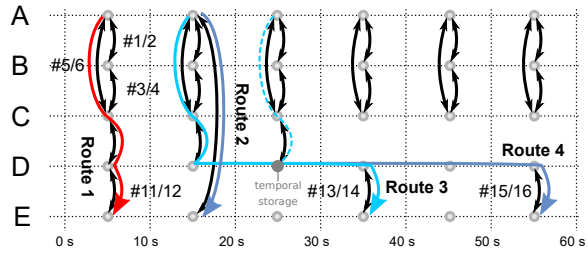
a)



b)



c)



d)

Figure 3. An example space network represented by a) Contact plan table, b) static graph of the topology, c) time line view, and d) time-evolving graph. Example routes from A to E are highlighted and ordered by best delivery time (BDT) (others are possible, see Section V)

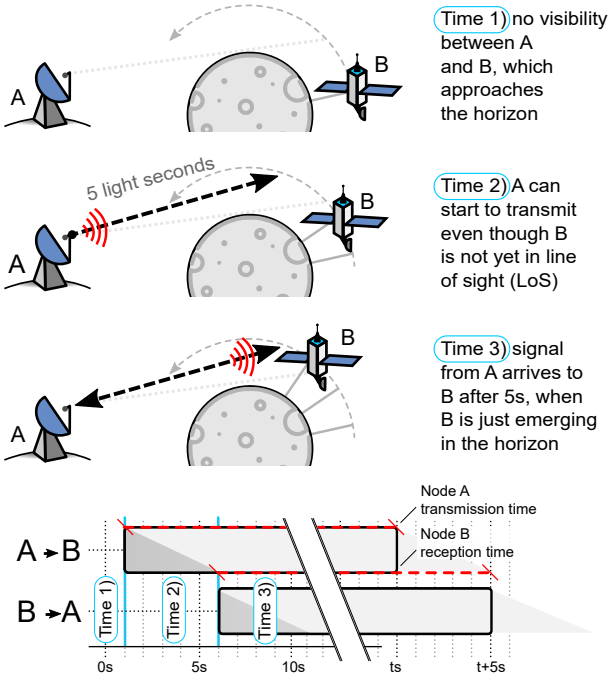


Figure 4. The propagation delay over long-range bidirectional communications requires two different contacts shifted in time for a proper modeling. In interplanetary occlusions, transmission shall start from A before the target node emerges in the horizon (5 seconds in the example). Once over the horizon, node B can start transmitting. The inverse effect is observed when the remote satellite hides on the planet horizon.

be computed by dividing the total volume of data that can be transferred during the contact by the duration of the contact. Furthermore, the approximate distance (also known as One-Way Light Time ($owlt$), measured in light-seconds) between nodes A and B during a contact must be known in order for routes to be computed. It is theoretically possible for this "range" value to change between the start and end of a contact, but for simplicity we will here assume that each contact is associated with a single range value [17].

Since unidirectional transmission is not uncommon in space communications, contacts are by definition unidirectional; bidirectional communication is represented in a contact plan by a pair of unidirectional contacts. Moreover, because of $owlt$, the start time of a contact in one direction is typically not the same in the reverse channel of a bidirectional link. In particular, if the time interval (t_1, t_2) is the transmission time for node A, then the reception time interval for B is $(t_1 + owlt, t_2 + owlt)$. This asymmetry has some non-intuitive consequences as illustrated in Fig. 4. A transmission to a spacecraft which is out of line of sight must start before it rises in the horizon of e.g., a remote planet. After a period equal to the $owlt$ between sender and receiver, both forward and return contacts can exist simultaneously, but not before. The same effect, but opposite, is seen when the satellite hides in the horizon at the end of the contact. Finally, space communication contacts also typically exhibit different forward and return data rates. For these reasons we use unidirectional contacts to properly model space networking connectivity.

Contact *volume* is a function of data transmission rate and contact duration. If the rate is constant, the product $(end - start) * rate$ is enough to model the connectivity throughout the contact. Variable bit rates can otherwise be approximated via averaging as illustrated in Fig. 5 a). Also, adaptive modulation and coding schemes can be modeled via multiple consecutive contacts with different rates as depicted in Fig. 5 b). It is important to note at this point that a contact is specifically **not** an episode of activity on a link between two nodes. 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 must be considered together in a single contact as shown in Fig. 5 c). The correct balancing of data over different link technologies is a convergence layer adapter matter and should be tackled on that level; it is beyond the scope of this tutorial.

So far, the parameters of route computation have been fixed, meaning that they do not change after the contact has been predicted in the planning phase. A contact can, however, include variables to store valuable information for the CGR routing and forwarding processes discussed in the following sections. In particular, *maximum available transmission volume* ($MAV(p)$) variables are used to keep track

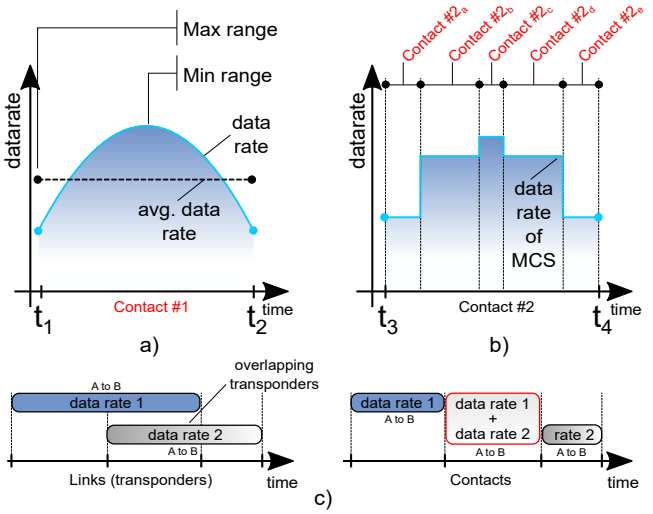


Figure 5. Data volume considerations: a) The data $rate$ parameter in a contact can be obtained by averaging the data rate over the period. b) Specific contacts for each modulation and coding scheme (MCS) class can be considered. c) Data rate over overlapping links with the same source-destination pair must be modeled by a single contact.

Table III
CONTACT PARAMETERS

Fixed parameters	
$C.snd$, $C.rcv$	Sending and receiving nodes
$C.start$, $C.end$	Start and end transmission time
$C.rate$	Data transmission rate
$C.owlt$	Distance (range) expressed in light-seconds
$C.volume$	Contact volume $((end - start) * rate)$
Variable parameters	
$C.MAV(p)$	Maximum available volume for priority p
Route search working area:	
$C.arr\_time$	Data arrival time at the destination (dst)
$C.visited$	Contact was visited in a previous loop
$C.visited\_n[\,]$	List of previously visited nodes
$C.pred$	Pointer to the predecessor contact in route
Route management working area:	
$C.suppr$	Contact suppressed by management
$C.suppr\_nh[\,]$	List of suppressed Next hop contacts
Forwarding working area:	
$C.fbtx$	First Byte transmission time for a route
$C.lbtx$	Last Byte transmission time for a route
$C.lbrx$	Last Byte arrival time for a route
$C.EVL$	Contact's effective volume limit

of the remaining available volume of a contact as bundles are assigned to be forwarded through it[2]. Because different priorities p exist, different $MAV(p)$ values express the volume availability for each of them. Indeed, while a contact can be fully booked for a lower priority p_{low} (i.e., $MAV(p_{low}) = 0$), a higher priority bundle could still be allocated to C provided that $MAV(p_{high}) > B.EVC$. Other variables are managed in the algorithm's *working areas*. There is a working area for route search, route management and forwarding that will be further described and used in Section IV, V and VI respectively. All the contact parameters used in this tutorial are summarized in Table III.

B. Contact Plan

Arranging all the anticipated contacts into a contact plan allows the operator to capture the time-evolving nature of a space network. A contact plan can naturally be expressed in a table such as the one in Fig. 3 a), but different representation and models can be considered for routing computation. On the one hand, a traditional *static graph* representation is shown in Fig. 3 b). Although this familiar representation provides a direct match with the physical disposition of nodes (i.e., ground stations, satellites etc. are the actual nodes in the graph), it hinders an intuitive understanding of the time dynamics (the presence of edges between nodes depends on the time). Indeed, trying to integrate time effects into such a model (i.e., by aggregation or other simplifications) can result in non-trivial model inaccuracies as reported in [99]. A *time-line* view, as illustrated in Fig. 3 c), is useful as an intuitive visualization of the contact plan, but it cannot be used for graph-based algorithm calculations. *Time-evolving graphs*, as shown in Fig. 3 d) are an appealing representation and modeling technique for space networks [100], [101], where time dynamics are captured in discrete states. For each state snapshot, a static graph is used to represent the (stable) connectivity of the network during that period of time. Naturally, a connectivity change (i.e., the initiation or termination of any contact) requires of a new state. The main drawback of time-evolving graphs is that they scale poorly, not only with larger numbers of nodes and contacts but also especially with time, hindering the efficient modeling of space networks with long time horizons. Furthermore, the modeling of delay effects requires yet more states. For example, the two contacts in Fig. 4 would require 4 states: $(1, 6)$, $(6, 11)$, $(11, t)$ and $(t, t + 5)$, each with a connectivity graph of the whole network [65].

[2] $C.MAV(p)$ corresponds with the Maximum Transmission Volume (MTV) terminology in SABR [17]. We have chosen a different naming to avoid confusion with $C.volume$, which is the **maximum** volume the contact can carry.

C. Contact graph

Contact graphs, illustrated in Fig. 6, overcome the drawbacks of other models. 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. Edges E in a contact graph can be seen as episodes of data retention at a node i, between the end of the earlier contact and the start of the subsequent contact. Fig. 6 illustrates the CG_A^E based on the contact plan example of Fig. 3. The structure of the contact graph may seem somewhat counter-intuitive as it bears almost no relation to the topology of the network as illustrated in Fig. 3 b). The vertices of the graph are not the satellites or ground stations at which data reside, but rather the episodes of contact during which data can be transferred; the edges of the graph are not paths enabling data transfer but rather the periods of time during which data must be stored while awaiting successor contacts. In compensation, though, this static graph representation facilitates the execution of network algorithms over time-evolving networks on simple graph structures.

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. 6, the receiving node of contact 1 ($C^{0,60}_{A,B}$) is the same as the transmission node of contact 3 ($C^{0,60}_{B,C}$). An edge between them represents a temporal storage in the connecting node B which is 0 when contacts are overlapped in time since an immediate transmission is possible; otherwise it can take any value. Besides storage time modeling, propagation delay or $owlt$ can be conveniently considered within each vertex on the graph. In other words, time can pass on edges (storage) but also on vertices (propagation). Finally, notional contacts from node A to itself and from node E to itself (a.k.a. root and terminal contacts) are included as part of the contact graph, enabling consistent notation throughout. As stated in [17], the root and terminal vertex may be thought of as corresponding to delivery from the application to the source node's Bundle Protocol agent (the root vertex) and delivery from the destination node's Bundle Protocol agent to the application (the terminal vertex).

A different contact-graph data structure is used for each source-destination pair. This might be seen as a disadvantage with respect to time-evolving graphs, where a single data structure serves for all source-destination node pairs. However, in practical operations on flight computers with modest computing power, it is desirable to compute routes to a single destination and not all or several destinations. For computation of routes from a single source node to a single destination node, a simple contact graph as described here is all that is needed. Furthermore, as discussed in [70] and detailed in [71], adapted Dijkstra's searches can be used over contact graphs to efficiently determine optimal routes. For these reasons contact-graph based routing had received increasing attention from the space networking community and is now a CCSDS recommended procedure for space networks [17].

D. Routes

A route R^E_A for a bundle with node A as current location and node E as destination is defined as some sequence $hops[]$ of the contacts in a contact plan 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 [17].

In the example of Figs. 3 and 6, one of the fastest routes, route (1) $R^E_A = \{C^{0,60}_{A,C}, C^{0,30}_{C,D}, C^{0,10}_{D,E}\}$, is highlighted in red. A different route can also be obtained via a different last contact: route (3) $R^E_A = \{C^{0,60}_{A,C}, C^{0,30}_{C,D}, C^{30,40}_{D,E}\}$, marked as light blue. As mentioned, DTN assumes no persistent end-to-end connectivity, thus a route might require temporal storage at intermediate nodes (e.g., in node D until time 30 in route (3)). Also, contact propagation delays ($owlt$) are aggregated along with storage times to calculate the best delivery time (BDT) for each route. Route (1), for example, exhibits a

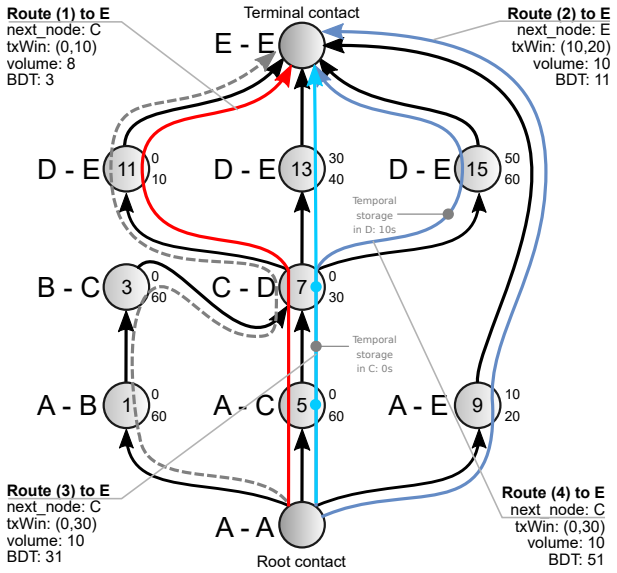


Figure 6. Contact graph CG^E_A for the topology of Fig. 3. Example routes from A to E are highlighted and ordered by best delivery time (BDT) (others are possible, see Section V).

$BDT = 3$ despite immediate transmission at each node because the $owlt = 1$ in each of the three constitutive contacts. Storage and propagation times in a route are computed during the contact graph exploration as presented in Section IV.

The receiving node for the first contact of a route is termed the route's *entry node* (or $next\_node$, e.g., node C), and the destination node is noted as to_node. Each route is characterized by a valid *transmission window* (tx_win) defined by the earliest and latest time where transmission from sending node A to the $next\_node$ can happen. As mentioned, a *best-case delivery time* or BDT indicates the earliest possible time the first byte of data can arrive at the destination node (that is, the earliest time at which delivery of the data can begin, not the time at which delivery will be completed). Also, similar to contacts, the $volume$ of a route indicates the maximum amount of data the route can carry. Indeed, as discussed below, there is a correlation between $R.volume$ and constitute contacts' $C.volume$. These are parameters that remain fixed once the route is discovered.

A route also has a working area used to store variable metrics in forwarding time, which depend on the bundle to be forwarded and the node's queuing status. For example, the earliest transmission opportunity (ETO) parameter is used to estimate the real transmission time of data given the actual backlog of bundles [78].

Projected arrival time or PAT is the adjusted time at which the bundle, of a given size, can be delivered to the destination. While BDT is a best-case delivery time, PAT considers ETO and bundle size, which are only available at forwarding time.

Finally, the effective volume limit or EVL stores the actual volume of data the route can carry considering ETO[3].

[3] $R.EVL(p)$ corresponds to the Route Volume Limit (RVL) terminology in SABR [17]. We have chosen a different naming to avoid confusion with $R.volume$, which is the **maximum** volume the route can carry.

Table IV
ROUTE PARAMETERS

Fixed parameters	
$R.hops[\,]$	List of contacts in the route
$R.to\_node$	Final node in the path ($hops[-1].dst$)
$R.next\_node$	First neighbour in the path ($hops[1].dst$)
$R.tx\_win(s,e)$	Interval of time (s,e) where the route is valid
$R.BDT$	Best time at which data can arrive to dst
$R.volume$	Maximum data the route can carry
Variable parameters	
Forwarding working area:	
$R.ETO$	Earliest Transmission Opportunity
$R.PAT$	Projected arrival time
$R.EVL$	Route's effective volume limit

These variables will be referenced and further described in Section VI. Table IV summarizes the route parameters used in the following sections.

E. Volume

Contacts' and routes' volume modeling is an important aspect of CGR that takes especial relevance in congestion control at the forwarding stage. Both contacts and routes have fixed $volume$, which indicates the maximum volume of data they can carry. In the case of a contact, the volume is directly obtained from $(end - start) * rate$; but the route volume depends on the sequence of hops and their $owlt$. An example is illustrated in Fig. 7 a), where the contact volumes are computed as $C^{0,60}_{A,C} = 60$, $C^{0,30}_{C,D} = 30$, and $C^{0,10}_{D,E} = 10$, but due to $owlt$, the resulting route volume of $R^{E}_{A} = 8$. This is the maximum amount of data that the route can carry from A to E, and does not change throughout the contacts and route's lifetime. On the other hand, as discussed in Section VI, when data is forwarded through contacts, available resources are consumed and maximum available volume $C.MAV$ (initialized to $C.MAV(p) = C.volume\ \forall\ p$) is decreased. As discussed in Section VI, $C.MAV$ affects the route's $R.EVL$, which can preclude the forwarding of a bundle if the available volume is not enough to accommodate the bundle as indicated by $B.EVC$.

The simplest approach towards variable volume annotation is to use *linear modeling*, as currently implemented in ION and assumed throughout this paper. In a linear volume model, the residual volume of a contact C after forwarding a bundle B_1 with priority p is updated by $C.MAV(p)_{t2} = C.MAV_{t1}(p)_{t1} - B_1.EVC$. When a subsequent B_2 with EVC larger than $C.MAV(p)_{t2}$ needs to be forwarded, any route through C will not be considered a candidate. This simplistic volume modeling assumes that the resource utilization of the contact always begins at the contact start time, and that subsequent bookings will always come immediately after that. This approach might not reflect real utilization of resources in time, especially in the case of networks with long (e.g., continuous) contacts. Fig. 7 b) illustrates the inaccurate annotation this assumption might generate. As such, an incorrect volume annotation can result in sub-optimal bundle forwarding. Indeed, in a linear volume modeling, there is no record of the time within the contact on which the volume is being consumed. An alternative and more accurate *list modeling* would address $C.MAV$ as a list of volume windows instead of single numeric characterization. In such a way, volume consumption in contacts and routes can be represented by time-bounded elements with $(start, end)$ intervals. The trade-off between accuracy and the computational resources required to implement the routines discussed below remains an open research issue.

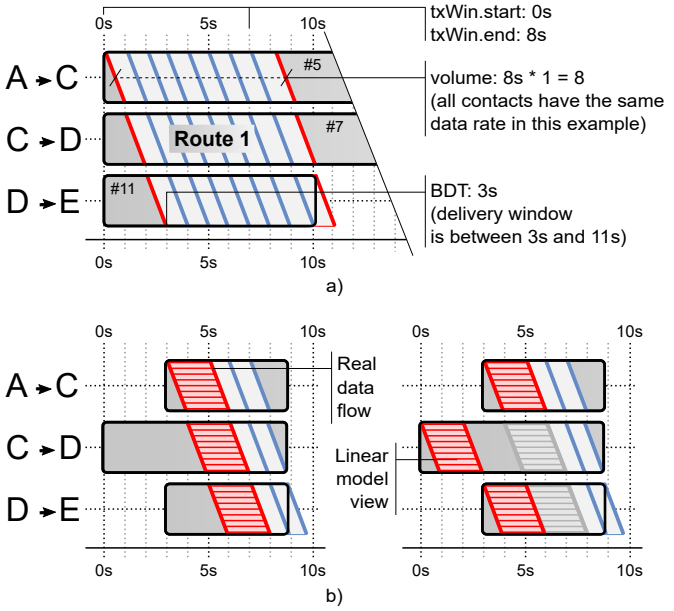


Figure 7. a) Volume illustration of route (1) on a time line view. Transmission and reception windows are $(0s, 8s)$ and $(3s, 11s)$ for a total route volume $R.volume = 8s * 1 = 8$, where $rate = 1$ for all contacts in the example CP. b) Linear volume modeling inaccuracies. Although the real data flow (in red) occupies contact $C_{C,D}$ from 4s to 6s, the linear model would represent a utilization between 0s to 2s. The incorrect depiction of the volume placement within a contact can result in incorrect route selection at forwarding time.

IV. ROUTE SEARCH

In this section we present the route search algorithmic approach, the core of the CGR routing ecosystem.

A. Contact Graph Dijkstra Search

The main advantage of contact graph data structures is that they can be used as input for traditional shortest-path algorithms. In particular, Dijkstra's shortest path algorithm [102] can be adapted to find a path from a source to a destination node with the best delivery time in a contact graph [70]. The resulting calculation can either be used to determine the next hop (in distributed routing such as ION [15]) or serve as the complete route path (in source or centralized routing [71]).

Main Algorithm: The modified Dijkstra search for the best route through a contact graph is presented in Alg. 1. The final outcome of the procedure is a route R^{D}_{S} from source S to destination D, with an arrival time BDT. The route is computed starting from the root contact (C_{root}) to the destination D in the contact plan CP. As mentioned earlier, the root contact is an artificial contact $C^{0,\infty}_{S,S}$ from source S to itself. The arrival time in the root contact $C^{0,\infty}_{S,S}.arr\_time$

is set to the starting time of the expected route (i.e., the time at which the Dijkstra search is called). The route search will thus ignore contacts ending before $C^{0,\infty}_{S,S}.arr\_time$. The *working area* of each contact in the CP must be cleared before each Dijkstra search. A cleared working area means: (*i*) contact arrival time $C.arr\_time = \infty$, (*ii*) contact visited flag $C.visited = False$, (*iii*) contact predecessor $C.pred = \varnothing$, and (*iv*) visited nodes list $C.visited\_n \leftarrow \{\}$. Note that the list of visited nodes is a CGR enhancement that goes beyond the basic algorithm described in SABR [17], as discussed below.

The first part of the algorithm is a loop that explores the CP while keeping track of the current contact in C_{curr} (line 5). C_{curr} is initially C_{root}, the root contact of the contact graph.

The *successor* (or "next hop") contacts of the current contact C_{curr} are those contacts C that have sender node $C.snd$ equal to the receiver node of the current contact $C_{curr}.rcv$ (see Fig. 8). Within each iteration of the exploration loop, (*i*) a Contact Review Procedure (`CRP`) updates the arrival time metric (cost) through all contacts that are successors to C_{curr} (line 6), (*ii*) a Contact Selection Procedure (`CSP`) then selects the contact with best cost metric (line 7), C_{next}, and (*iii*) current contact C_{curr} is set to the best contact C_{next} before proceeding with the next iteration (line 9). The Contact Review Procedure (`CRP`) and Contact Selection Procedure (`CSP`) are described below. Finally, when no C_{next} contact can be determined, the CP exploration loop ends (line 11).

A "final contact" is a contact whose receiving node is the destination node D. If the destination node D was reached during the CP exploration loop, the best final contact was stored in C_{fin} and the best final arrival time in BDT, indicating the best route has been discovered. If this is the case, the route reconstruction loop takes place (line 12). The sequence of contacts in the route is recovered from predecessor contacts ($C.pred$) starting from C_{fin} back to the root contact $C^{0,\infty}_{S,S}$, while populating the resulting route $R^D_S.hops$ list (line 14-16). The $R^D_S.BDT$ was computed during `CRP`, and the to_node and $next\_node$ parameters are directly accessible via the first hop $R^D_S.hops[0].rcv$ and last hop $R^D_S.hops[-1].rcv$[4]. However, the route volume limit ($R^D_S.volume$) and the valid transmission window ($R^D_S.tx\_win$) remain to be be computed at the end of this phase (line 17).

Example: In the example network of Fig. 6, the fastest route is Route (1), for which $C_{fin} = C^{0,10}_{D,E}$ and $BDT = 4$. The iterations followed by the adapted Dijkstra search to obtain this route are illustrated in Fig. 8. After the last iteration, the `CRP` and `CSP` procedures will have populated $C^{0,10}_{D,E}.pred = C^{0,30}_{C,D}$, $C^{0,30}_{C,D}.pred = C^{0,60}_{A,C}$ and $C^{0,60}_{A,C}.pred = C^{0,\infty}_{A,A}$, the latter being precisely the root contact. As a result, Route (1) will be constructed as $R^E_A = \{C^{0,60}_{A,C}, C^{0,30}_{C,D}, C^{0,10}_{D,E}\}$, with a $BDT = 3$, $tx\_win = (0; 8)$ and $R.volume = 8$ in the second part of the algorithm. A timeline view illustration of route 1) and the exposed metrics is offered in Fig. 7.

[4]In the algorithms listings throughout this paper we use Python-like list notations namely: $list[-1]$ indicates the last element of the list, $list[:x]$ returns a sub-set of elements in list up to x and $list[x:]$ a sub-set starting from x. This facilitates the study of these algorithms with the provided pyCGR library. Note that we use a coding style suitable for low-level programming languages (i.e., skips, breaks, etc.), as required for flight-grade software.

Algorithm 1: Contact Graph Dijkstra Search

Data: root contact C_{root}, destination D, contact plan CP (with cleared working area)
Result: Route R^D_S from source S to destination D

1 $R^D_S \leftarrow \{\}$
2 $C_{fin} \leftarrow \{\}$ `// final contact`
3 $BDT = \infty$ `// final arrival time`
4 $C_{curr} = C_{root}$ `// current contact is root`
`/* contact plan exploration loop */`
5 **while** *True* **do**
`/* contact review procedure */`
6 $C_{fin}, BDT \leftarrow$`CRP` (CP, C_{curr}, C_{fin}, BDT)
`/* contact selection procedure */`
7 $C_{next} \leftarrow$`CSP` (CP, C_{curr}, BDT)
8 **if** $C_{next} \neq \{\}$ **then**
9 $C_{curr} \leftarrow C_{next}$
10 **else**
11 break `// review and selection completed`
`/* route reconstruction loop */`
12 **if** $C_{fin} \neq \{\}$ **then**
13 $C = C_{fin}$
14 **while** $C \neq C^{0,\infty}_{S,S}$ **do**
15 $R^D_S.hops \leftarrow \{C\}$
16 $C = C.pred$ `// previous contact in path`
17 `Compute` ($R^D_S.tx\_win$, $R^D_S.volume$)

Contact Review Procedure: The specific steps of the Contact Review Procedure (`CRP`) are listed in Alg. 2. These steps are performed for each successor contact C of the current contact C_{curr}. The `CRP` reviews the CP and updates the working areas of those successor contacts.

The following disqualification conditions are tested in the first part of the algorithm (lines 1-11): (*i*) C ended before the arrival time in C_{curr}, (*ii*) C has already been visited in a previous call to `CRP`, (*iii*) C leads to an already visited node, (*iv*) C or its destination node have been administratively suppressed ($suppr$ and $suppr\_nh$ variables), (*v*) C does not comply with data volume requirements. Regarding this last point, contacts for which available volume for the lowest priority has been depleted ($C.MAV(0) = 0$) should be ignored. In other words, this disqualification considers the case where route computation happens after enough low-priority data has been forwarded through the evaluated contact consuming all its available volume. In such case, there is no more possible data that can flow through this contact and it should be disqualified.

The best-case arrival time $C.arr\_time$ (cost) for transmission of a bundle during contact C is then computed (lines 12-16). $C.arr\_time$ is initialized to the start time of C or the arrival time of the current contact C_{curr}, whichever is later. Then arr_time needs to be increased by the contact's propagation delay ($C.owlt$). Since OWLT is approximated, a safety margin is suggested in [17] equal to $owlt_{mgn} = 125 * C.owlt/186282$.

Next, if the computed arrival time is earlier than any previously computed arrival time for C (line 17), indicating that the arrival time for C is improved by transmitting from C_{curr},

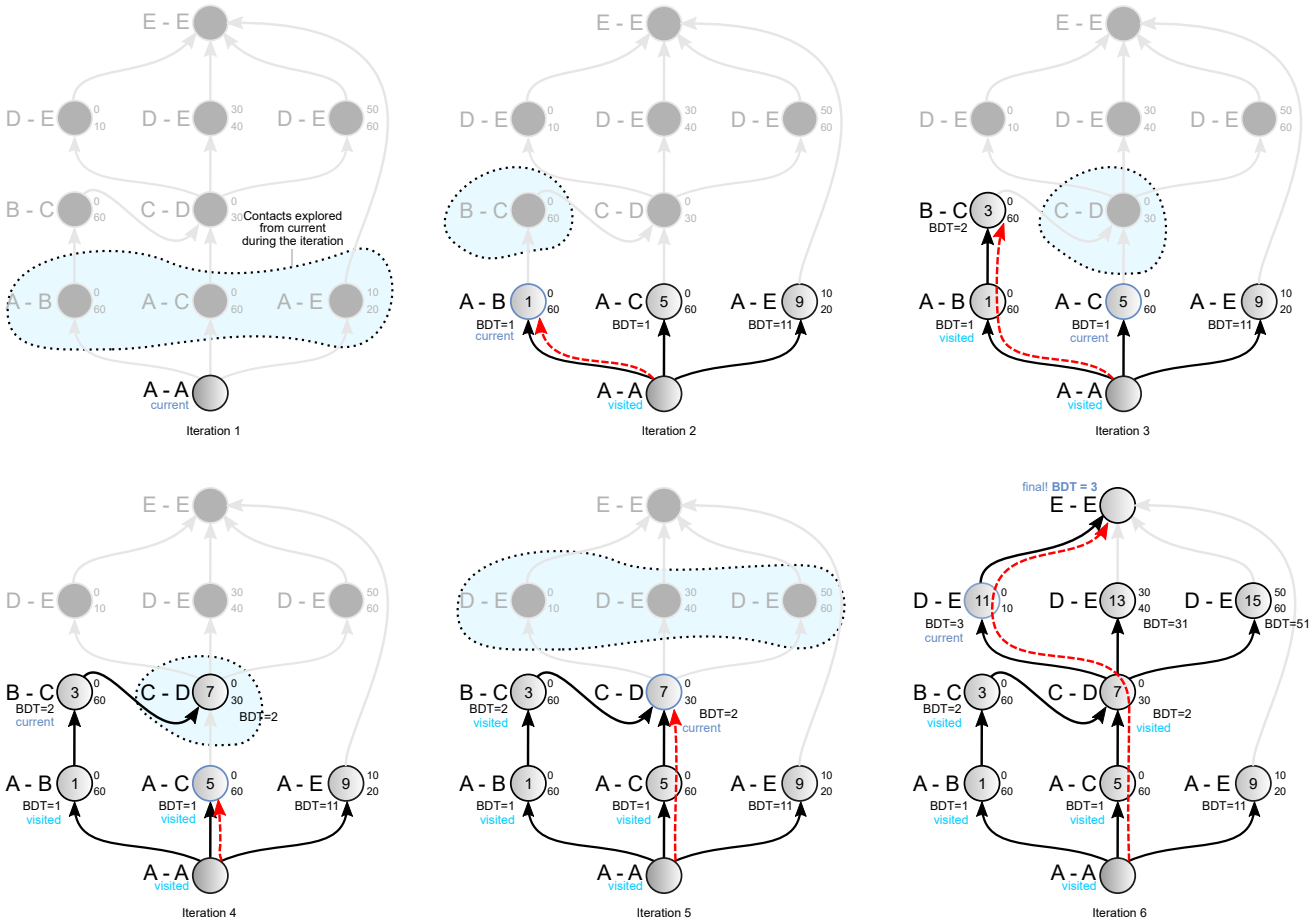


Figure 8. Dijkstra search iterations over the contact graph on Fig. 6. Current and visited contacts are annotated at each iteration. A light blue highlight indicates which are the visited contacts, while a red dashed arrow indicates which is the lowest cost path (lowest BDT) on each stage. As per Dijkstra algorithm properties, the best route is found even though some contacts (e.g., $C^{10,20}_{A,E}$) are never visited.

then the new arrival time and corresponding predecessor node C_{curr} are noted in C's working area.

After the CRP steps have been performed for each successor contact C of the current contact C_{curr}, C_{curr} is marked as "visited" and the Contact Selection Procedure begins.

Example: In the first call to CRP in the example network of Fig. 6, the current contact is set to the root contact ($C_{curr} = C^{0,\infty}_{A,A}$). As illustrated in Fig. 8, three neighbor contacts are reviewed in iteration 1: $C^{0,60}_{A,B}$, $C^{0,60}_{A,C}$ and $C^{10,20}_{A,E}$. Since arrival times at these contacts were initialized to ∞, all of them are updated to $C^{0,60}_{A,B}.arr\_time = 1$, $C^{0,60}_{A,C}.arr\_time = 1$ and $C^{10,20}_{A,E}.arr\_time = 11$ (labeled as BDT in the figure). Note that this last contact is able to reach E, so $C_{fin} = C^{10,20}_{A,E}$ and $BDT = 11$ after this call to CRP terminates. But this is not the best route. As seen below, the next call to CRP will select $C_{next} = C^{0,60}_{A,B}$, from which potential neighbors are $C^{0,60}_{B,A}$ and $C^{0,60}_{B,C}$. The former, however, will be skipped as A is already in the $C^{0,60}_{A,B}.visited\_n$ list (see iteration 2 in Fig. 8). As discussed below, the best route is found on iteration 6 with $C^{0,10}_{D,E}$ as final contact and a $BDT = 3$.

An alternative and slightly different formulation of CRP is possible if the information of the specific bundle B to be forwarded is available when calling the routine. Since this approach is not standard SABR and is not implemented in ION, we discuss it separately in Appendix A.

Contact Selection Procedure: The Contact Selection Procedure (CSP) is detailed in Alg. 3. Once the CP has been reviewed and its contacts' working areas updated by CRP, the procedure in CSP revisits all contacts in CP to determine which is the one with the best annotated arrival time ($C.arr\_time$). This set of nodes is known as the *frontier* in Dijkstra terminology [102]. Each contact in the frontier that has already been visited or is administratively suppressed is immediately skipped (lines 4-5). Each contact whose arrival time is later than the current BDT is likewise immediately skipped (lines 6-7). It is at this point where Dijkstra assumption of a monotonically increasing cost metric must hold. Indeed, since time cannot go back, it makes no sense to proceed with exploration through such a contact. This phenomenon can also be appreciated in Fig. 8, where a $BDT = 3$ forces CSP to never select contacts such as $C^{10,22}_{A,E}.arr\_time = 11$. From the remaining contacts, the one with the earliest arrival time $t_{earliest\_arrival} = min(C.arr\_time \forall C \in CP)$ is noted in C_{next}. If no C_{next} contact is selected, then the contact exploration loop terminates. Alternatively, if the receiver node $C_{next}.rcv$ of the newly selected next contact C_{next} is the destination node D, then the newly selected C_{next} is recognized

Algorithm 2: Contact Review Procedure (CRP)

Data: CP, C_{curr}, C_{fin}, BDT
Result: revised CP, C_{fin}, BDT

1 **for** *contact* $C \in CP \mid C.src = C_{curr}.dst$ **do**
/* Ignore conditions test */
2 **if** $C.end \leq C_{curr}.arr\_time$ **then**
3 skip C // (i) ignore due contacts
4 **if** $C.visited$ **then**
5 skip C // (ii) ignore visited
6 **if** $C.to \in C_{curr}.visited\_n$ **then**
7 skip C // (iii) ignore visited nodes
8 **if** $C.suppr$ **or** $C \in C_{curr}.suppr\_nh$ **then**
9 skip C // (v) ignore suppressed contacts
10 **if** $C.MAV(0) = 0$ **then**
11 skip C // (iv) ignore overbooked contacts
/* Calculate arrival time */
12 **if** $C.start < C_{curr}.arr\_time$ **then**
13 $arr\_time = C_{curr}.arr\_time$
14 **else**
15 $arr\_time = C.start$
16 $arr\_time+ = C.owlt + owlt_{mgn}$
/* Update arrival time if better */
17 **if** $arr\_time < C.arr\_time$ **then**
18 $C.arr\_time = arr\_time$
19 $C.pred = C_{curr}$
20 $C.visited\_n = C_{curr}.visited\_n + C.dst$
/* Mark if destination reached */
21 **if** $C.dst = D$ **and** $C.arr\_time < BDT$ **then**
22 $BDT = C.arr\_time$
23 $C_{fin} = C$
24 $C_{curr}.visited = True$ // contact review completed

Algorithm 3: Contact Selection Procedure (CSP)

Data: CP, BDT
Result: C_{next}

1 $C_{next} \leftarrow \{\}$
2 $t_{earliest\_arrival} = \infty$ // earliest delivery time
3 **for** *contact* $C \in CP$ **do**
4 **if** $C.suppr$ **or** $C.visited$ **then**
5 skip C // ignore suppressed or visited
6 **if** $C.arr\_time > BDT$ **then**
7 skip C // ignore worst arrival time
8 **if** $C.arr\_time < t_{earliest\_arrival}$ **then**
9 $t_{earliest\_arrival} = C.arr\_time$
10 $C_{next} =\leftarrow C$

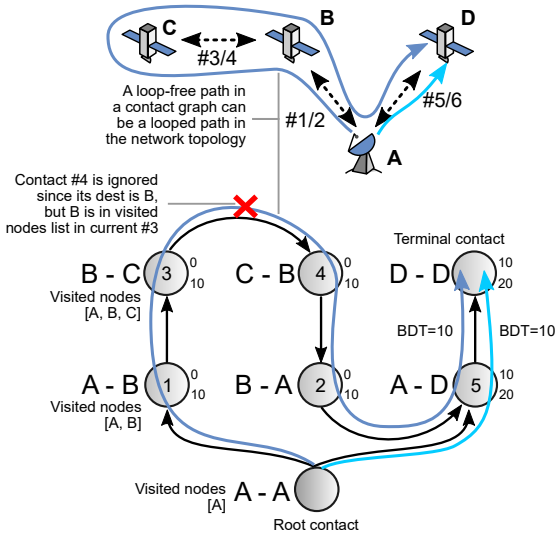


Figure 9. A loop-free path in a contact graph is not necessarily a loop-free path in the topology. In order to build a truly acyclic contact graph, additional measurements shall be taken by a visited nodes list ($C.visited\_n[\,]$).

as the final contact of the best route; C_{fin} is set to the new C_{next}, BDT is set to the arrival time of the new C_{next}, and the contact plan exploration loop ends. Otherwise, C_{curr} is set to C_{next} and the next iteration of the contact plan exploration loop begins.

Example. Contact $C_{D,E}^{0,10}$ is noted as final with a $BDT = 3$ on iteration 6 as shown in Fig. 8). Since not of the other possible contacts ($C_{D,E}^{30,40}$, $C_{D,E}^{50,60}$ and $C_{A,E}^{10,20}$ with respective arr_time = 31, 51 and 11) can improve that BDT, the routine terminates and deliver Route (1) as the best route.

B. Loops

Contacts in the CP are marked as *visited* to avoid choosing as the next current contact C_{curr} a contact that has already been considered as the current contact. Because time is monotonic (strictly increasing as the contact graph is explored), the arr_time of a visited contact that cannot improve the current BDT metric and so this contact can be safely ignored. This is the fact that makes Dijkstra suitable for computing loop-free routes over the contact graph. Nevertheless, a loop-free path in a contact graph is not necessarily a loop-free path in the real network topology. An illustrative example is shown in Fig. 9. Both paths offers the same $BDT = 10$ ($owlt = 0$ for all contacts), but one exhibits a forwarding loop through nodes A, B and C. Note that the latter is not a loop in the contact graph, as no contact appears more than once in the path. To avoid this problem in the adapted Dijkstra search, the $visited\_n$ list keeps the visited nodes along each contact working area. The list is considered on the CRP procedure to avoid revising contacts that lead to visited nodes (lines 6-7 in Alg. 2). The visited node list disqualification is not part of SABR standard [17].

C. Complexity

The overall complexity of the CGR adapted Dijkstra search can be extrapolated from the base Dijkstra structure [102]. The original Dijkstra algorithm is known to have a time complexity of $\mathcal{O}(|V|^2)$, where $|V|$ is the number of vertices in the graph. In this case, the time complexity of the CGR Dijkstra call is $\mathcal{O}(|C|^2)$, where $|C|$ is the size of the contact plan. Indeed, in the worst case, all contacts are visited in Alg. 1 and all contacts are reviewed in Alg. 2, resulting in a quadratic

complexity (route reconstruction is linear with respect to $|C|$ and is thus disregarded). Optimizations based on min-priority queues and Fibonacci Heaps are known to reduce the worst-case complexity of Dijkstra searches [103]. With this in mind, a time complexity of $\mathcal{O}(|C|log(|C|))$ can be achievable in CGR Dijkstra if, instead of iterating over all contacts in Alg. 3, a suitable priority queue is used when annotating the contact arrival times in Alg. 2. We have kept these processes separated for the sake of clarity.

In most practical cases, however, the average complexity should be much lower as many contacts are immediately ignored for the reasons noted above. Furthermore, the contact iterations in line 1 of Alg 2 can be encoded into a hash table of contact lists with $C.snd$ as key, reducing the processing effort on subsequent Dijkstra calls. Disregarding the time to build the hash table (which can be indeed built offline) the overall complexity of the adapted Dijkstra can be reduced. Note that the complexity studies in [83] consider the whole route table computation and not a single Dijkstra computation, thus they are applicable to Section V.

D. Multiple Destinations

As with the base Dijkstra algorithm, the adapted CGR algorithm can be trivially adapted to calculate the best route to all possible destinations. To this end a C_{fin} and BDT need to be independently tracked and verified for each potential destination D (lines 21-23 in Alg. 2). In `CSP`, the maximum BDT among all destinations should be considered (lines 7-8 in Alg. 3), and route reconstruction must occur for all final contacts found (lines 12-17 in Alg. 1). This approach has been recently explored in conjunction with a node-based spanning tree approach for contact plans in [93].

V. Route Management

At any DTN node, multiple routes to any single destination node may need to be computed and managed. The main reasons are these: (*i*) each route expires after $tx\_win.end$, thus others will be needed afterwards; (*ii*) routes have limited volume, thus others might provide the required extra capacity; (*iii*) routes might be overbooked for a given priority class, thus others might provide a means to reduce congestion; and (*iv*) routes might not occur as expected due to uncertainties or failures, thus others might provide necessary redundancy. Route management is the approach by which multiple routes to a single destination are properly computed, stored, used, and pruned.

As with Internet routing, storing computed routes in *route tables* enables the node to avoid repeating Dijkstra searches. For CGR, a route 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 [17]. A route list $[R_S^D]$ from S to D can be of size 0 (no routes for destination) up to all possible routes in the contact plan for that node pair.

A. Classification

As introduced in Section II, the routing scheme for a space network can be *centralized* or *distributed*. Also, *source routing* can be a middle-ground approach.

1) **Centralized**: In a centralized approach, a mission operation and control center generates the contact plan, computes all necessary route lists, and uploads those route lists to the nodes of the network (option B in Fig. 2). The number of routes to be retained for each source-destination pair at any given moment in time is a mission decision. Centralizing the computation of routes is a conservative approach that enables tight control and detailed debugging, as all possible decisions taken by potentially unreachable nodes are already present and available on ground. Furthermore, centralized routing reduces the route computation effort on limited flight computers.

2) **Distributed**: In distributed routing, the topological information in the contact plan is distributed to nodes (option A in Fig. 2), which then autonomously compute the route lists based on local state and policies. A DTN node with this capability can decide the exact number of routes to compute based on traffic shape. This is a more aggressive but scalable approach as DTN nodes are the only source of truth regarding the traffic status and route demand at any given moment.

3) **Source routing**: Source routing can be an intermediate solution where only the sender node computes the route and stores it in the bundle so that next hop nodes can forward the bundle without recomputing the route [67], [69]. Although example applications using this approach were studied [77], an in-depth quantitative comparison of the centralized, source routing and distributed routing approach remains a topic for future research.

While in centralized routing all required routes are *statically* computed, the approach towards route management in distributed routing can be either *static* or *dynamic* [92]. In static route management, all possible routes for a destination are calculated for the contact plan duration period. Following this calculation, the route list remains unchanged until an update of the contact plan is received. In dynamic route management, a limited number of routes are initially computed from the contact plan. The route list can then be extended and updated as per traffic demand (or prediction) from the local node. In other words, a limited number of new best routes are calculated on-demand by the DTN node. The reader interested in implementations of these CGR variations is referred to DtnSim code [20].

ION. In older ION versions (v3.6 and older), the route management was distributed and static. All routes to a given destination D were fully computed as soon as a bundle was to be forwarded to D for the first time [15], [87], The main drawback was that valuable compute time was required when long route lists were to be populated on resource-constrained processors (i.e., several Dijkstra calls), and any modification to the contact plan forced the deletion and re-computation of all route lists. Since computations were made at forwarding time, the processing blocked any transmission until all routes were computed; this reduced the effective utilization time of a contact [84]. Therefore, in recent ION implementations (v3.7 and later), route lists are dynamically computed. When required, ION computes the next best route for a given destination and adds it to the route list. When such route is no longer valid (i.e., $tx\_win.end$ time passed, or $R_S^D.EVL(p)$ is exhausted for the priority class p, the best next route is

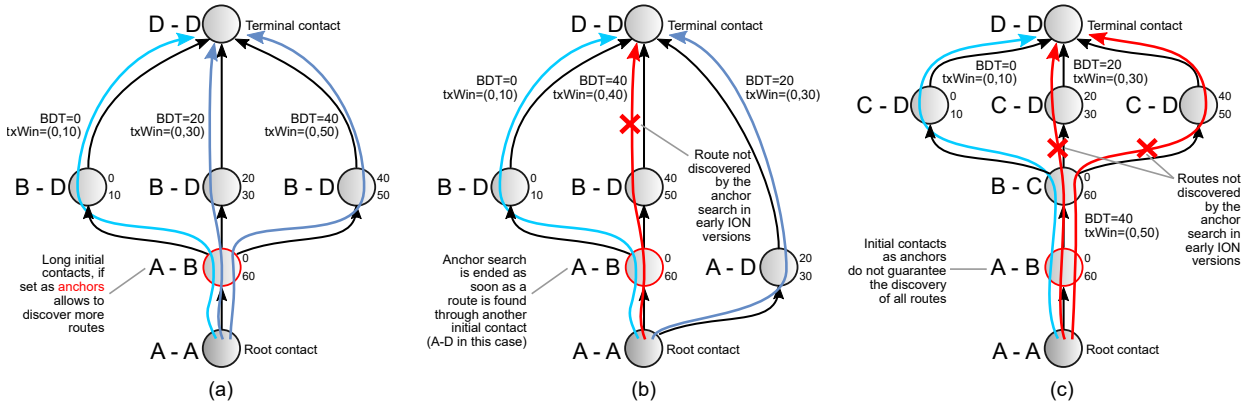


Figure 10. Anchor search limitations. (a) Anchor search anchors on long initial contacts ($C^{0,60}_{A,B}$ in this case) and explores further routes. (b) However, the anchor search can be prematurely terminated when a next best route is found outside the anchor contact (e.g., through $C^{20,30}_{A,D}$). (c) Also, anchoring only on initial contacts excludes the discovery of all routes branching from contacts further ahead (e.g., through $C^{0,60}_{B,C}$).

calculated (but the previous R^D_S is retained in the route list if valid, as $R^D_S.EVL$ might be suitable for higher priority traffic).

Whether centralized or distributed, static or dynamic, some number of routes have to be obtained from a contact plan for a given source-destination pair. Clearly, the first route can be obtained from a contact graph Dijkstra search, as presented in Section IV. However, modifications or annotations to the contact graph need to be made so that each subsequent Dijkstra search returns the next best route. The contacts' route management working area as shown in Table III serves this purpose.

B. Anchor Search

Older versions of ION leveraged a custom route management heuristic which serves as an explanation for what a route management process should do. The original approach was to remove from consideration the *initial contact* of the previously computed route ($R^D_S.hops[0].suppr = True$). Subsequent calls to the Dijkstra route search algorithm would thus ignore such contact from that moment on. As a result, in the best case, one route would be added to the route list per initial contact in the contact graph. However, the suppression of long initial contacts (i.e., ground contacts such as in Fig. 3) hindered the discovery of alternative routes starting later in time through that same contact. The first heuristic to overcome this issue was to apply the so-called *anchor-based* search. In an anchor search, a long initial contact (i.e., a contact $C \mid C.end = R^D_S.tx\_win.end$) was temporarily considered as *anchor*, meaning that several routes could be obtained from it before its suppression [87]. However, the limitations of the anchor approach were soon discovered and addressed in [92]. Among them, anchor search was prone to miss the detection of routes by prematurely terminating the anchor phase, or by ignoring routes branching from long contacts ahead in the path as summarized in Fig.10.

C. Yen's algorithm

The correct and complete approach to construct the route table is to use Lawler's modification of Yen's algorithm [104], [105]. This method was implemented in ION v3.7 and is still being used at the time of this writing (v4.0). Yen's algorithm performs a Dijkstra search in a nested loop to deliver a set of the K best routes ($len([R^D_S]) = K$), where K is provided as argument.

The adaptation of Yen's algorithm is listed in Alg. 4. The algorithm delivers a route list $[R^D_S]$ with at most K best routes from source S to D as per contact plan CP. After clearing the Dijkstra and route management working areas in the CP, the first best route is computed by Dijkstra and added to the route list (lines 1-2). The root contact $C^{t,\infty}_{S,S}$ is also added as one hop in the route for the reasons explained below. A potential route list $[P^D_S]$ is then initialized (line 3) and populated with candidate best routes in a loop ranging from 1 to K (line 4). The best route at the end of the outer loop is added to the final route list in $[R^D_S]$. The inner loop iterates over each contact of the last route added to $[R^D_S].[-1]$ (line 5). Each of the chosen contacts becomes a so-called *spur contact* (C_{spur}). In order to properly detect all routes, the root node $C^{0,\infty}_{S,S}$ should also be treated as a spur node, thus, it should also be part of the route data structure during Yen's procedure. A *root path* is then defined from the root contact to the spur contact, but without including the latter ($hops[0]$ to $hops[C_{spur} - 1]$ in line 6). The next step is to compute the best route from the spur contact to the destination terminal node. To this end, the CP working areas are again cleared (line 7). Then, all contacts in the root path are suppressed from the CP (lines 8-9). Also, all edges from C_{spur} to neighbouring contacts which are already considered in previous routes with the same root path in $[R^D_S]$ must be suppressed (lines 10-12). The C_{spur} is then prepared as root contact for a subsequent Dijkstra call by setting a proper arr_time and $visited\_n$ list composed by all visited hops $hops.to$ in the root path P_{root} (lines 13-14). If a route from the spur contact to the destination is found, the CGR Dijkstra call returns a *spur path* P_{spur} (line 15), which is combined with the root path rendering a new route to be included in the potential route list (lines 16-17). Routes in $[P^D_S]$ are then sorted by BDT such that the first route in the potential route list ($P^D_S[0]$) has the best arrival time among

Algorithm 4: Contact Graph Yen's algorithm

Data: source-dest. S-D, contact plan CP, K routes
Result: Route list $[R_S^D]$ with K routes

1 `Clear(`CP`)`
2 $[R_S^D] \leftarrow C_{S,S}^{0,\infty}$ + `Dijkstra(`$C_{S,S}^{t,\infty}$, D, CP`)`
3 $[P_S^D] \leftarrow \{\}$ `// potential routes`
4 **for** k **from** 1 **to** $K-1$ **do**
5 **for** $C_{spur} \in [R_S^D].[-1]$ **do**
 `/* root path from` $C_{S,S}^{0,\infty}$ `to spur contact */`
6 $P_{root} = [R_S^D].[-1].hops[0, C_{spur} - 1]$
7 `Clear(`CP`)`
 `/* suppress all contacts in root path */`
8 **for** $C \in P_{root}$ **do**
9 $C.suppr = True$
 `/* suppress` C_{spur} `edges in any` R `in` R_S^D `*/`
10 **for** $R \in [R_S^D]$ **do**
11 **if** $P_{root} = R.hops$ **then**
12 $C_{spur}.suppr\_nh \leftarrow R.hops[len(P_{root})]$
 `/* compute spur path from` C_{spur} `to` D `*/`
13 $C_{spur}.arr\_time = P_{root}.arr\_time$
14 $C_{spur}.visited\_n \leftarrow \forall P_{root}.hops.to$
15 P_{spur} =`Dijkstra(`C_{spur}, D, CP`)`
 `/* if any, insert new potential route */`
16 **if** $P_{spur} \neq \{\}$ **then**
17 $[P_S^D] \leftarrow \{P_{root} + P_{spur}\}$
18 Sort $[P_S^D]$ by arrival time
 `/* move best potential route to` $[R_S^D]$ `*/`
19 **if** $[P_S^D]$ *is not empty* **then**
20 $[R_S^D] \leftarrow P_S^D[0]$
21 **else**
22 **finish** `// no more potential routes`

all routes (lines 18). If the potential route list is not empty, the best route in $[P_S^D]$ is moved to the route list $[R_S^D]$ (line 20). Otherwise, if the potential route list is empty, no more routes can be found and the algorithm terminates (line 22). The resulting list in $[R_S^D]$ will contain the best K routes in the contact plan, ordered by BDT.

Example. The step-by-step iterations and the results of Yen's computations for the example network in Fig. 3 are illustrated in Fig. 11. As mentioned, the best route is found by Dijkstra on the initialization stage ($R_A^E = \{C_{A,C}^{0,60}, C_{C,D}^{0,30}, C_{D,E}^{0,10}\}$). Then, the figure presents one graph for each iteration and spur contact on which a new potential route is found. For instance, at iteration 0 and spur contact 0 (root contact), $R_A^E = \{C_{A,B}^{0,60}, C_{B,C}^{0,60}, C_{C,D}^{0,30}, C_{D,E}^{0,10}\}$ is found. Although another potential route is found on the second spur node $C_{spur} = C_{C,D}^{0,30}$, the first one makes it to the final route list. Indeed, its $BDT = 4$ is better than $BDT = 31$, so it is the next route moved from $[P_S^D]$ to $[R_S^D]$ at the end of iteration 0. The next one added to $[R_S^D]$ is $R_A^E = \{C_{A,E}^{10,20}\}$, at the end of iteration 1. No potential routes can be found branching out from the latter on iteration 2. Two more potential routes are found on iteration 3 and 4 (routes using the latest $C_{D,E}^{50,60}$ contact rendering a $BDT = 51$). The algorithm terminates in iteration 6 when no more potential routes are found in $[P_S^D]$. Actually, Yen's algorithm is able to find the best routes from A to E in the example contact plan in Fig. 3.

ION. Yen's algorithm is part of ION in order to control the number of best routes to be computed. Indeed, in practice, only a few out of many possible routes are typically used from a contact plan. As a result, ION's routing routine (in `libcgr.c`) operates on a stateful Yen's implementation. In other words, k is advanced until a *workable* route is obtained (the conditions under which a route is declared workable are discussed in Section VI). Then, the loop is put on hold, storing the state of the algorithm until the computation of a next route is commanded. This is indeed a Yen's implementation that dynamically computes routes. Finally, as stated in [17], a *route pruning* is recommended every time the contact plan is changed or updated. A route pruning implies that earlier route computations might be invalidated and thus all route lists shall be discarded and recomputed as needed. Efficient strategies to selectively update route tables without pruning remain an open research topic.

D. Complexity

Evidently, if all possible routes from a graph are to be determined, then *breadth first* or *depth first* algorithms followed by a sorting of the obtained list would outperform Yen's algorithm. Such an approach would have been appropriate for the example in Fig. 11 if all 7 routes were to be computed. The worst case complexity of a breadth first search (BFS) is known to be $\mathcal{O}(|C|^2)$. Plus, a route sorting algorithm of $\mathcal{O}(|R|log(|R|))$ is required afterwards, where $|R| = |C|$ in the worst case [106]. A BFS route computation approach would thus render a worst case time complexity of $\mathcal{O}(|C|^2 + |C|log(|C|)) \approx \mathcal{O}(|C|^2)$. Instead, when only a few (and controllable number of) best routes need to be extracted, Yen's is an adequate solution. Yen's time complexity depends on the Dijkstra's complexity ($\mathcal{O}(|C|log(|C|))$ when using a Fibonacci Heap). A total of $K \times s$ Dijkstra calls are made in Yen's loop, where K is the number of routes requested and s is the average size of the spur path (which is estimated to be $log(|C|)$ but is $|C|$ in the worst case [107]). As a result, the CGR-adapted Yen's worst case time complexity is estimated to be $\mathcal{O}(K|C|^2log(|C|)) \approx \mathcal{O}(|C|^2log(|C|))$ (as K is supposed to be a small value, thus amenable to a constant).

E. Alternative Methods

Alternative route management methods with practical advantages had been proposed in [92] before Yen's was adopted by CCSDS in the SABR recommendation. Alternative methods are introduced below and compared in Table V. Their implementation is publicly available in DtnSim [20].

First ended: One of those methods suggests the suppression of the first contact in the path to terminate (the one with $min(C.end)$) before calling Dijkstra again. This approach was coined *first-ended* route management strategy. The first-ended

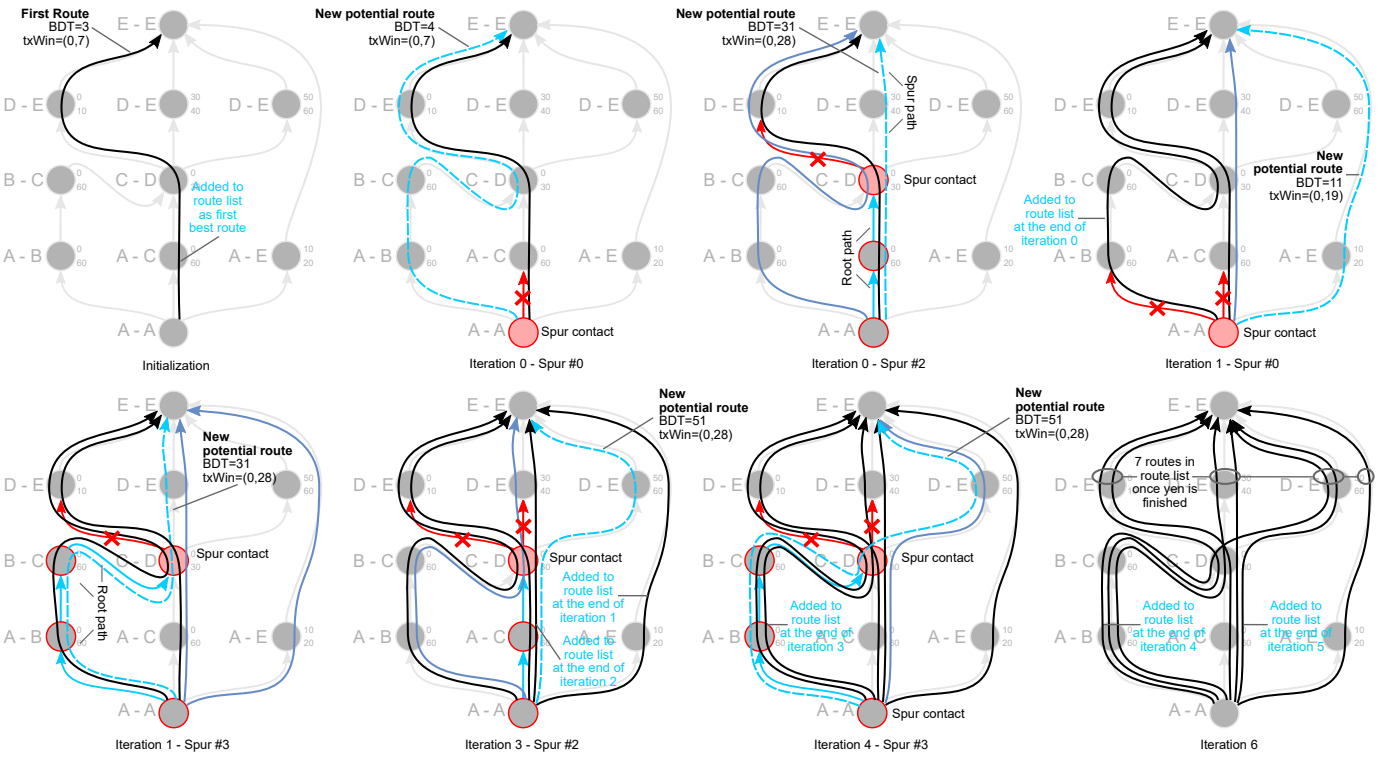


Figure 11. The iterations of the Yen's algorithm when applied to the example network in Fig.3 with $K \geq 7$. Spur contact and contacts in root path are highlighted in red. Computed potential routes never overlap with suppressed edges crossed out in the graphs. Potential routes are inserted into $[P_S^D]$, the list of potential routes, and are moved to the route list $[R_S^D]$ one at a time as needed. As a result, up to 7 routes in order of BDT are found and added to $[R_S^D]$.

Table V
ROUTES DISCOVERED BY ROUTE MANAGEMENT METHODS FOR THE EXAMPLE NETWORK IN FIG.6.

$R.BDT$	$R.EVL$	$R.tx\_win$	Hops in R_A^E	Anchor	Ended	Depleted	Yens'	BFS
3	8	$(0, 10)$	$\{C_{A,C}^{0,60}, C_{C,D}^{0,30}, C_{D,E}^{0,10}\}$	✓	✓	✓	$K \geq 1$ ✓	✓
4	7	$(0, 10)$	$\{C_{A,B}^{0,60}, C_{B,C}^{0,60}, C_{C,D}^{0,30}, C_{D,E}^{0,10}\}$				$K \geq 2$ ✓	✓
11	10	$(10, 20)$	$\{C_{A,E}^{10,20}\}$	✓	✓	✓	$K \geq 3$ ✓	✓
31	10	$(0, 30)$	$\{C_{A,C}^{0,60}, C_{C,D}^{0,30}, C_{D,E}^{30,40}\}$		✓	✓	$K \geq 4$ ✓	✓
31	10	$(0, 30)$	$\{C_{A,B}^{0,60}, C_{B,C}^{0,60}, C_{C,D}^{0,30}, C_{D,E}^{30,40}\}$				$K \geq 5$ ✓	✓
51	10	$(0, 30)$	$\{C_{A,B}^{0,60}, C_{B,C}^{0,60}, C_{C,D}^{0,30}, C_{D,E}^{50,60}\}$				$K \geq 6$ ✓	✓
51	10	$(0, 30)$	$\{C_{A,C}^{0,60}, C_{C,D}^{0,30}, C_{D,E}^{40,50}\}$			✓	$K \geq 7$ ✓	✓

strategy guarantees that for each interval of time, the route management method will provide one route (the fastest) to the destination. As shown in Table V, the first-ended approach delivers at least one route for each of the $(0, 10)$, $(10, 20)$ and $(20, 30)$ intervals. This is indeed a simpler technique which reduces computation effort at the expense of not accessing redundant paths during each time window.

First depleted: Alternatively, the *first-depleted* approach bases the suppression decision on the volume bottleneck in the previously computed route. In particular, the contact that is limiting the route volume is suppressed before subsequent Dijkstra searches. For example, in $R_A^E = \{C_{A,C}^{0,60}, C_{C,D}^{0,30}, C_{D,E}^{0,10}\}$, the contact $C_{D,E}^{0,10}$ is the limiting contact with 8 seconds of effective transmission time (see Fig. 7). Suppressing such contact from a new Dijkstra search delivers the route $R_A^E = \{C_{A,E}^{10,20}\}$, and so on. As described in [92], the scheme also annotates the volume reduction in all contacts on the route. This allows an accurate detection of limiting contacts in subsequent routes. As a result, the "first depleted" approach mimics the time-evolving booking of contacts as if traffic were actually being allocated in the contact plan. However, as discussed in Section VI, volume annotation is a forwarding stage matter.

VI. FORWARDING

Once routes are computed and stored in a route table, the forwarding of outbound data can take place. Local information on queue status at current time t and the parameters of bundle B are consulted in order to select the best candidate route for a target destination. Finally, placement of the bundle in the outbound queue triggers volume annotations and consequent (immediate or deferred) transmission of the data.

A. Candidate Routes

The conceptual list of routes that can be used at node S at time t to forward a bundle B is termed the *candidate routes* list. The construction of such a list implies the selection of a sub-set of the routes in the route list $[R_S^D]$ at forwarding time. The $[R_S^D]$ list is computed and maintained as discussed in Sections IV and V respectively. While formal rules for the candidate routes list construction procedure are provided in the

SABR recommended standard [17], in this section we describe how to realize the candidate route list construction in Alg. 5. We advise the reader to rely on Tables II, III and IV to follow the procedure divided into four distinct phases as described below.

1) **Basic checks**: The procedure evaluates all routes in the route list $[R_S^D]$ (line 1). In order to reduce the computation, basic checks that can detect and skip non-plausible routes with minimal processing are executed first. In particular, (*i*) any route for which the best-case delivery time is later than the bundle deadline and (*ii*) any route for which $next\_node$ (the receiver node of the first contact in the route) is in the *excluded nodes* list ($[E_n]$) will be immediately ignored (line 2). As indicated in [17], the excluded nodes list identifies nodes which must not be considered for forwarding due to any of the following causes. **Cause 1**) the node is $B.sender$, i.e., that node sent the bundle to node S; that is, *back-propagation* is normally excluded. In particular, forwarding loops resulting from differences in topological information (i.e., contact plan or volume annotations) at different nodes can be avoided by forbidding back-propagation. However, if the bundle is being reforwarded as consequence of a custody refusal, return to the sender can be authorized in hopes of finding an alternative path [17]. **Cause 2**) nodes that have previously refused custody of bundles with this destination. Under the assumption that nodes refusing custody of a bundle are unavailable or overbooked, forwarding through them is not recommended. Periodic probing can detect the recovery of the neighbour and remove it from the excluded nodes list. **Cause 3**) nodes can be manually added to the excluded nodes list via configuration for management purposes.

2) **Earliest transmission opportunity**: The earliest transmission opportunity or ETO is then computed based on the volume already in queue (named *applicable backlog volume* in [17], and here represented by $v_{blog}(B.p)$). The applicable backlog volume is defined as the sum of the $EVCs$ of all bundles currently queued for transmission to the route's $next\_node$ whose priority p is greater than or equal to that of the bundle that is to be forwarded. For instance, in ION, the local node learns about $v_{blog}(B.p)$ by inspecting the status of the CLA used to reach the $next\_node$. To compute the ETO, the adjusted start time t_{start} is first determined as either the current time t or the start time of the first contact in the route $R.hops[0].start$, whichever is later (line 4). t_{start} is the earliest possible start time for the transmission of B, which needs to be adjusted by the *residual backlog* already allocated for $R.next\_node$ ($v_{blog}(priority)$). The residual backlog is the portion of the applicable backlog affecting the route's first contact. These bundles will have to be transmitted in $R.hops[0]$ before transmission of bundle B can begin. To compute the residual backlog, all contacts whose receiver is the next hop node ($R.next\_node$), that have not ended and that start earlier than the first hop ($R.hops[0].start$) are evaluated (line 6). In simpler words, these are contacts that can at least partially clear the applicable backlog volume before the route R starts. The *applicable prior contact volume* (v_{prior}) that each of these contacts can clear out of the backlog before current time t is computed and accumulated (line 7). To this end, the applicable start time of the contact is set to the current time t if the contact started before the forwarding phase. The residual backlog volume affecting R's first hop can be obtained by subtracting the volume accumulated in v_{prior} from the applicable backlog volume ($v_{blog}(B.p)$). The time shift (termed *backlog lien* in [17]) imposed by the residual backlog volume is then computed (t_{blog}, line 8) and used to determine the ETO under current queuing conditions considering the bundle priority (line 9). In order to declare R a potential route, the obtained ETO must not be later than the end of R's first

Algorithm 5: Candidate Routes Construction

Data: Route List $[R_S^D]$, Bundle B, excluded nodes list $[E_n]$, current node S, current time t
Result: Candidate route list $[C_S^D]$

1 **for** *route* $R \in [R_S^D]$ **do**
  /* 1) Basic checks */
2 **if** $R.BDT > B.dline$ **or** $R.next\_node \in [E_n]$ **then**
3 skip R // bundle expires or node excluded
  /* 2) R.ETO computation and check */
4 $t_{start} = max(t, R.hops[0].start)$
5 $v_{prior} = 0$
6 **for** $C \in CP \mid C.src = S$ **and** $C.dst = R.next\_node$ **and** $C.end > t$ **and** $C.start < R.hops[0].start$ **do**
7 $v_{prior} += (C.end - max(t, C.start)) * C.rate$
8 $t_{blog} = max(0, v_{blog}(B.p) - v_{prior})/R.hops[0].rate$
9 $ETO = t_{start} + t_{blog}$
10 **if** $ETO > R.hops[0].end$ **then**
11 skip R // first hop is overbooked
  /* 3) R.PAT computation and check */
12 $R.hops[0].fbtx = ETO$
13 **for** $C \in R.hops[1:]$ **do**
14 $C.fbtx = max(C.start, C_{prev}.lbrx)$
15 $C.lbtx = C.fbtx + B.EVC/C.rate$
16 $C.lbrx = C.lbtx + C.owlt$
17 $R.PAT = R.hops[-1].lbrx$
18 **if** $R.PAT > B.deadline$ **then**
19 skip R // arrival time later than deadline
  /* 4) R.EVL computation and check */
20 **for** $C \in R.hops[]$ **do**
21 $succ_{min\_end} = min(C.end \,\forall\, C \in R.hops[C:])$
22 $eff_{stop} = min(C.end, succ_{min\_end})$
23 $eff_{dur} = eff_{stop} - C.fbtx$
24 $C.EVL = min(eff_{dur} * C.rate, C.MAV(B.p))$
25 $R.EVL = min(C.EVL \,\forall\, R.hops[])$
26 **if** $R.EVL \leq 0$ **then**
27 skip R // no volume left
28 **if** $R.EVL \leq B.size$ **and** $B.frag = False$ **then**
29 skip R // no volume for whole bundle
30 $[C_S^D] \leftarrow R$

hop (line 10).

3) **Projected arrival time**: Based on the ETO computations, the projected arrival time or PAT of the bundle can be obtained. ETO is used as the first byte transmission time of R's first hop ($C.fbtx$, line 12); subsequent contacts in the path can then be considered (line 13). The $fbtx$ of each subsequent contact in the route is set to the contact's start time or the previous contact's last byte arrival time ($C_{prev}.lbrx$), whichever is greater. This is roughly equivalent to computing the ETO for each hop in the route path, assuming that forwarding at each node can begin as soon as the full bundle (i.e., last byte) has been received[5]. The time required to transmit the bundle on the channel is then calculated ($B.EVC/C.rate$) and used to compute the last byte transmission time of the current contact ($C.lbtx$, line 15). The last byte reception time for the contact indicates the point in time at which the bundle is expected to be completely received at the next hop node (line 16). Finally, the $C.lbrx$ of the last contact in the path is the projected arrival time of the bundle (line 17). If PAT is greater than the bundle deadline $B.deadline$, the route must be ignored (line 18).

4) **Effective volume limit**: Based on the PAT timing results, the effective volume limit of the route $R.EVL$ can be determined and evaluated. For this purpose, each contact C on the route is examined (line 20) to compute its individual effective volume limit, as follows. First, the minimum end time among all of C's successor contacts ($succ_{min\_end}$, line 22) is determined. Then the latest moment at which data can be transmitted from C, termed the contact's *effective stop time*, is set to $succ_{stop}$ or the current contact end time, whichever is less (line 22). The contact's *effective start time* is the contact's first byte transmission time $C.fbtx$. The contact's *effective duration* - the interval during which data transmitted from C can potentially reach the destination node - can then be computed as the difference between *effective stop time* and *effective start time* (line 23). The contact's *effective volume limit* ($C.EVL$) is the maximum volume of data that can be transmitted during such period, or the available volume for priorities equal or greater than p in C ($C.MTV(B.p)$), whichever the less (line 24). The route volume limit is then determined as the lowest value of EVL among all contacts in the path (line 24). In the case that $R.EVL$ is depleted (that is, not greater than zero), the route is ignored (line 26). Even if $R.EVL$ is not depleted, if $R.EVL$ is less than the size of the bundle and the bundle cannot be fragmented then the route must be ignored (line 28).

All routes that passed the 1) $basic$, 2) ETO, 3) PAT and 4) EVL validations are considered candidate routes for the destination (line 30). If the resulting list contains no candidate routes and the routing approach is dynamic, the calculation of new routes (i.e., a new call to Yen's algorithm) should take place. If no candidate routes were found and either the route calculation is static or dynamic route calculation fails to identify a new route, then no valid route has been found in the forwarding stage and the bundle cannot be forwarded.

[5]It is worth mentioning that we (and ION) assume that the queuing status of remote nodes is not available, as timely access to such information cannot be guaranteed in a DTN. Note, though, that works such as [67], [68], [94] have exploited this knowledge at forwarding time in non-delayed networks to affect the computation of effective transmission times of the bundle at remote nodes.

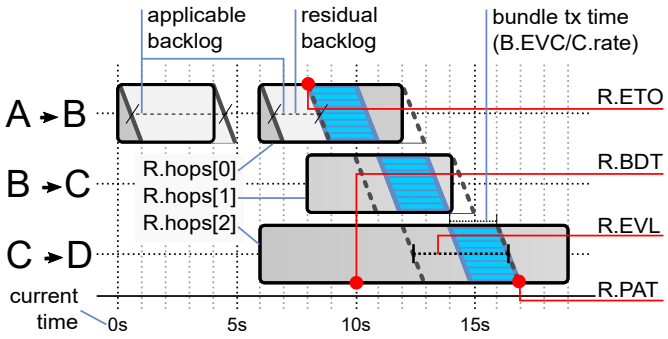


Figure 12. Illustration of parameters computed in Alg. 5. Applicable backlog is the total volume in queue at current time, while residual backlog is the part of that volume affecting the current route. Bundle transmission time is the time needed to transmit the whole bundle on the channel. These parameters are needed to evaluate the route considering bundle size and queue status.

Example. Fig. 12 illustrates the parameters computed in Alg. 5 for an example route R. Assuming $C.rate = 1 \; \forall \; C \in R.hops$, the applicable backlog for $B.priority$ is 6, from which 4 corresponds to a previous contact and 2 is the residual backlog affecting the route first hop ($R.hops[0]$). The resulting $R.ETO$ is thus $8s$, which is less than $R.hops[0].end$ indicating the route meets the initial condition. If $B.size = 2$, its transmission time is 2 for all contacts with $rate = 1$ and the $R.PAT$ results $17s$ ($R.hops[2].lbrx$). This is the maximum bundle deadline allowable through this route. Finally, $R.EVL$ is the lowest $C.EVL$, which in this case is 4, limited by $R.hops[0]$. Indeed, $R.hops[0].volume = 6$, but a residual backlog of 2 should imply a $C.MTV = 2$ and $C.EVL = 4$. Bundles larger than 4 can be transmitted via this route if and only if fragmentation is enabled.

ION. In ION, bundles for which no candidate routes can be found are placed in a $limbo$ list. The limbo list is a memory space where bundles can be stored with the expectation that a future contact plan update will provide valid routes. To meet the Bundle Protocol specification and avoid congestion issues, bundles in the limbo are deleted when their expiration times are reached.

B. Route Selection and Enqueuing

When the candidate route list for a destination contains multiple routes, the best candidate is defined as the route with the smallest value of projected arrival time (PAT) among all candidate routes. If several routes in the candidate route list offer the same PAT, then the one with the fewest contacts (hops) is selected among them. If tied, the route with latest termination time (end of tx_win) is selected. Finally, the route with the smallest first hop node number is selected if all other conditions are equal.

If the best candidate route's effective volume is less than the size of the bundle ($R.EVL(p) < B.EVC$) and anticipatory fragmentation is allowed and available, then the bundle may be fragmented at the local node. In this case the first fragment's payload is the portion of the bundle payload that results in a fragmentary bundle size that is no greater than the route's

available volume ($B.size = R.EVL(p)$), and the second fragment, containing the remainder of the bundle's payload, should be forwarded through a different route.

If the bundle is flagged as a critical ($B.critical$), then a copy of this bundle should be enqueued for transmission to every neighboring node (to_node) in the candidate route list. In the example list of Table V, a critical bundle from A to E would be copied twice and forwarded via C, B and E, with best-case delivery times of $BDT_C = 3$, $BDT_B = 4$ and $BDT_E = 11$.

Notionally, the bundle being forwarded will be enqueued in the transmission buffer allocated to the $next\_node$ marked in the selected route, a neighbor in the topology of the network. An alternative approach is to enqueue the bundle being forwarded in a transmission buffer more specifically allocated to the initial contact in the the selected route, to guarantee that the bundle is sent through the right contact despite possible contention from other locally or remotely sourced traffic [54].

Whenever a bundle B is enqueued for transmission via a particular route, the $C.MAV(p)$ of all contacts in that route, for that bundle's level of priority p and every lower level of priority, needs to be reduced by $B.EVC$. This will affect the EVL of all future candidates routes using these contacts (see volume modeling discussion in Section III).

In the enqueuing process, a bundle may be assigned to a route that is already fully subscribed by bundles of lower priorities. This is an oversubscription, which will cause some low priority bundles in the queue to miss their contact to accommodate higher priority bundles. The procedure by which these *bumped* bundles are handled is known as *overbooking management* [78]. Overbooking management takes care of quickly and efficiently re-forwarding the bumped bundles through alternative candidate routes that have enough effective volume for their (lower) priority level.

Finally, when a contact does not occur as expected, bundles need to be reforwarded by means of a procedure that is similar to the aforementioned overbooking management.

VII. Trends Outlook

The CGR framework presented here, comprising planning, route search, route management, and forwarding, has been shown to be capable of managing the routing of DTN bundles in a delayed and disrupted space network in concrete practical applications. However, we can imagine improvements in CGR scalability and resilience.

A. Scalability

Throughout the text we have alluded to the processing-constrained on-board computers (OBC) used in most spacecraft. Although high-performance COTS computing hardware is being evaluated for use in spacecraft, the electrical power available for flight computers is limited and dissipation of the heat generated by powerful processors is difficult in the vacuum of space. Additionally, the hostile radiation environment above Earth's atmosphere results in frequent single-bit upsets that are especially troublesome for machines operating on high-frequency clocks [108]. The plot in Fig. 13 shows the execution time of the CGR route search routine as a function of contact plan size on a commercial nano-satellite OBC. The execution time reaches several dozens of seconds on realistic contact plans with a few hundreds of contacts. This computational cost is in agreement with the complexity analysis presented in Section IV and the results published in [83]. While software optimizations might mitigate this problem somewhat, adaptations that would enable CGR to scale up to larger networks over extended time-horizons should be explored.

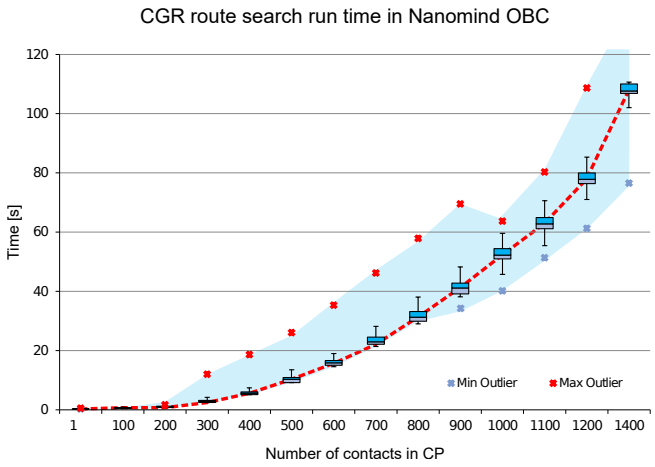


Figure 13. CGR route search (Alg. 1) run time in GomSpace's Nanomid A712C on-board computer with Free RTOS clocking at 10 MHz (implemented in C language). The contact plan corresponds to a Walker constellation of 12 satellites, 10 ground stations, and 100 ground spots for a duration of 24 hrs. [87]. Results are averaged among traffic from all ground sites to a mission control center reachable through any of the ground stations. The number of contacts considered ranges from 1 to 1400, at which point the OBC's 2MB of static RAM are exhausted. Courtesy of the Argentinian Space Agency (CONAE, Unidad de Formación Superior).

Hierarchical inter-regional routing: Hierarchical inter-regional routing (HIRR), proposed in [85], [86], aims at reducing the scope of route computation by limiting the topology knowledge provided to each DTN node in the network - in other words, reducing the contact plan size. In order to control the routing computation effort, the network can be divided into multiple *regions* as illustrated in Fig. 14. A region is defined as the set of all BP nodes that are either the senders or receivers of all contacts in some single contact plan; the size of the contact plan necessarily limits the size of the region. A region is thus a bounded sector of network topology; it might also correspond to some bounded sector of geographical space, but this is not required. Just as each node is required to have a unique identifier within the network, so too must each region be uniquely identified with the network. Regions nest. A given node is always a member of at least one region (termed the node's *home region*) and may additionally be a member of the region that encompasses the node's home region (the node's *outer region*). Note that a given node might be cited either as a receiver or as a sender in the contacts of the contact plan that defines a region of which it is a member. When the contact plan indicates that a node N can receive bundles as a member of region A and transmit those bundles as a member

of region B, and vice versa, N is termed the *passageway* node between regions A and B; the contacts that enable this traffic flow between regions A and B are termed *vents*. The advantage of hierarchical inter-regional routing is that each node sees only the contact plans describing the connectivity of the region(s) of which it is a member; contact plan storage requirements are small and the scope of CGR computation is sharply constrained, yet the size of the network is unlimited. A bundle that is destined for another node in the source node's own region is routed to that node by CGR; a bundle that is destined for a node is another region is routed via CGR to the passageway node that is a member of the destination region (and thence to the destination node by CGR) - or is a member of a region from which the bundle can ultimately be routed to the destination region by these same mechanisms. An initial inter-region routing algorithm was proposed in [85], and an auto-discovery approach was described in [86]. Further evaluation, potential optimization, and efficient region design procedures motivate further research efforts in this important domain.

B. Resilience

Standard contact graph routing rests on relatively relaxed modeling assumptions such as linear volume annotations, estimated bundle volume consumption ($B.EVC$), and averaged contact capacity approximations. Furthermore, the communication opportunities that arise in real time may not exactly follow the plan: divergences might be introduced by inaccuracies in the predictions at the planning stage, by unexpected interference, or by node failure. New, unplanned contacts might be established opportunistically as nodes come into unanticipated proximity. Also, distributed CGR can result in sub-optimal forwarding because not all nodes will have access to the same information regarding contact volume allocations. For these reasons, research into adaptations that would make CGR more resilient to deviations from its predictions is important. Specifically, making CGR resilient implies (*i*) considering probability in the *uncertain* components of the procedure and (*ii*) efficiently reacting to unexpected *opportunistic* events.

Uncertain contacts: Early reliability studies indicated that CGR was able to react to divergence from contact plans by late bundle reforwarding [87], [88]. Heuristic improvements to routing in DTN under uncertain contact plans were introduced in [89], where alternative CGR metrics and copy-based forwarding were explored. In particular, considering $hops$ count with respect to PAT proved to increase the delivery ratio. As a result, a dual-copy CGR variant proposed to send one copy of the bundle through the route with best PAT and another through the one with less $hops$, but these were early efforts outperformed by proper formalism as discussed below.

Another approach is to assume that each contact's $start$ and end times are probability distributions rather than known time values. Fig. 15 illustrates both cases.

Recent studies have proposed formal methods to model these probabilistic routing and copy-based forwarding questions [90]. The probabilistic contact plan may be modeled as a Markov Decision Process (MDP), from which optimal copy-based routing policies can be obtained [91].

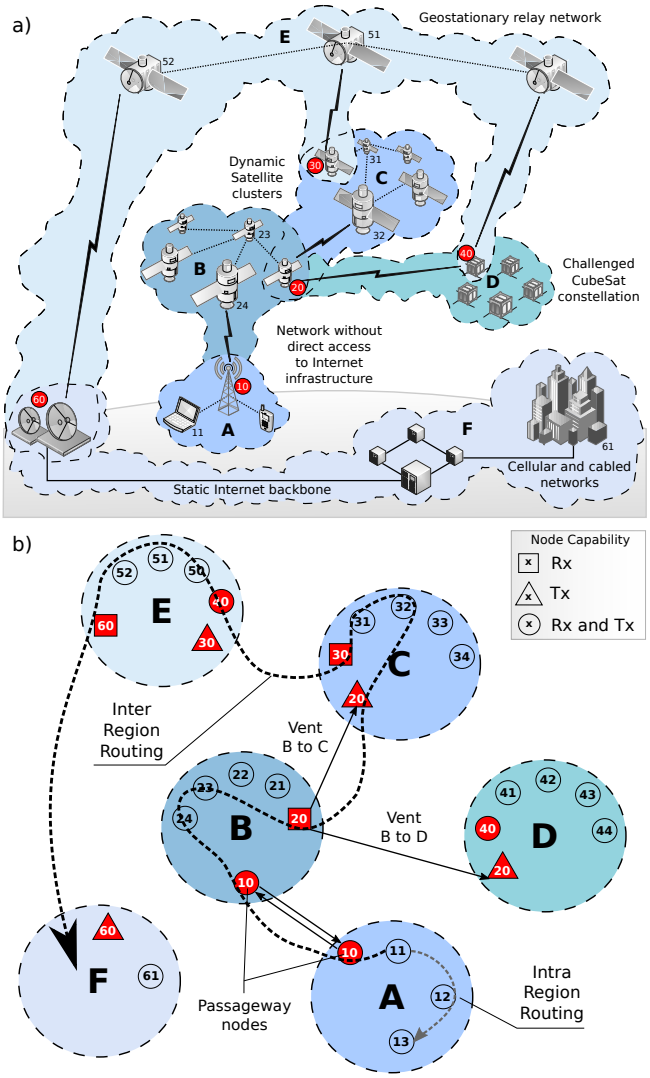


Figure 14. An example space network partitioned into six regions (A, B, C, D, E, F) in a). The criteria for composing this contact plan and, hence, the composition of the region were physical proximity and connection type (i.e., permanent, episodic), but others can be considered. An abstract diagram of the regions is shown in b). Depending on the contacts asserted in the contact plan, a given node may be able only to receive (Rx), only to transmit (Tx), or both within a region of which it is a member. When node 11 has a bundle destined for node 13, CGR is enough to route that bundle towards its destination. If node 11 has a bundle destined for node 61 (region F), 1) Inter-RR is used to identify a trans-regional route and a valid passageway node in region A (node 10), and 2) CGR is used to find a route to the passageway.

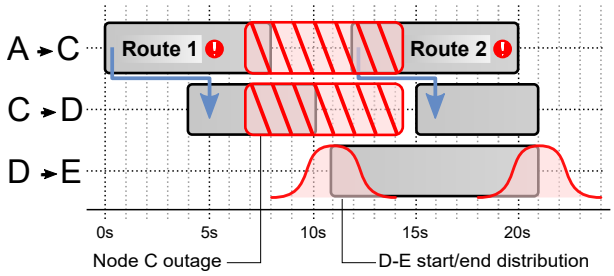


Figure 15. Contact plan becomes inaccurate due to a) node's transient or permanent contact outages, hindering transmission or reception of data, and b) uncertain start and end time of a contact due to unreliable orbit determination, incorrect orbit propagation (i.e., trajectory prediction), or errors in orientation estimation. The parameters of computed routes can be affected by these inaccuracies; even entire routes can be rendered invalid.

Opportunistic Contacts: While in considering uncertain contacts we deal with the problem of inaccuracy in computation based on *scheduled contacts*, the problem of opportunistic contacts arises from the discovery of *non-scheduled contacts*, potentially with new neighbor nodes. In other words, there is a learning and prediction strategy in opportunistic CGR (OCGR), as presented in [82]. The process starts with the recording in a contact log, at each node, of the $start$ and end times and $volume$ of all discovered contacts. Contact logs are exchanged between neighboring nodes when contacts (whether scheduled or discovered) begin. Contact prediction can then be performed for each snd, rcv pair in the updated contact log. Prediction involves the computation of mean and standard deviation of the duration both of contacts and of the gaps between contacts. A base confidence B value is obtained based on the standard deviation, and a new predicted contact C_{new} for the snd, rcv pair is added to the CP. $C_{new}.start$ is set to current time t_{curr}, and $C_{new}.end = t_{curr} + t_{elog}$, where t_{elog} is the earliest applicable entry in the contact log for the sender/receiver pair. The predicted transmission $rate$ for the new contact is the sum of the $volumes$ of all applicable log entries, divided by C_{new} duration. Also, the contact is configured with a net confidence $C_{conf} = 1 - (1 - B)^N$, where N is the number of applicable entries in the log (net confidences of scheduled contacts remains always 1). This enables a route R to exhibit a route confidence R_{conf} that is the product of the computed confidences for all contacts in that route. When a bundle is forwarded through R, its delivery confidence is incrementally increased by R_{conf} and, unless a predetermined threshold is reached, another copy is created for future forwarding.

Both uncertain and opportunistic CGR variants are recent research and must be considered preliminary. Room for improvement is considerable, making CGR resilience one of the most interesting research areas in delay-tolerant networking.

VIII. Closing Remarks

The fundamental problem of DTN is simply to maximize the volume of data that can be delivered to its destination in timely fashion through a network characterized by lengthy round-trip times. Solutions to that problem have multiple dimensions, but one of the most important is routing: how to best decide whether or not - and if so, when - to forward a copy of a given bundle to a given topologically adjacent node. While a very wide range of strategies for DTN routing have been proposed and studied, Contact Graph Routing is the only one that has been shown to be effective in network communications beyond Earth orbit. CGR is complex in order to accommodate the constraints imposed by the space environment, but that very complexity suggests that many opportunities for improving the algorithms remain. We expect research on this challenging topic to continue. To this end, we highlight possible axes on which research could extend the state-of-the-art frontier of routing in space networking.

(*i*) Devise efficient strategies to update route tables without a full pruning of the table after the contact plan is updated.

(*ii*) Perform an in depth quantitative comparison of the centralized and distributed routing approaches.

(*iii*) Derive estimations of the numbers of routes to be computed for each source-destination pair in centralized routing.

(*iv*) Study the trade-off between accuracy and processing-memory resources in linear and list volume modeling.

(*v*) Integrate opportunistic and uncertain CGR into a unified routing framework fitting space and ground networks.

(*vi*) Develop effective region operations methods for CGR, including region definition and self-learning approaches.

Appendix
Capacity-Oriented Search

In cases where the Dijkstra algorithm (see Section IV) is requested to obtain a route for a particular bundle B, the parameters of that bundle can be considered in the computation. However this is not standard CGR, where information pertaining to B is only available at the forwarding stage (see Section VI of the algorithm and not before. Since bundle-sensitive CGR goes beyond what is implemented in ION [15] and documented in the SABR specification [17], it is discussed separately in this appendix.

In some implementations, the CGR route search can be triggered for a specific bundle B. If this is the case, the overall search algorithm can be adapted to filter contacts based on the parameters of B. In particular, $B.EVC$ and $B.p$ can be used to condition the `CRP` (Alg. 2) as discussed below. On the one hand, in line 10 of Alg. 2, an extra volume check can be added in the 'if' statement. Instead of just checking that the contact retains some non-zero capacity for bundles of the lowest priority ($C.MAV(0) = 0$), $B.EVC$ and $B.p$ can further constrain the disqualification condition (($C.MAV(B.p) < B.EVC$)). This simple check is applicable for any bundle for which fragmentation is disabled ($B.frag = False$); for a bundle that can be fragmented, we instead check $MAV(p) > 100Bytes$ indicating that at least a fragment of the bundle can be allocated to the contact. By this means, every contact with too little volume to convey bundle B will be skipped during the Dijkstra search. Moreover, the arrival time variable in line 16 in Alg. 2 can be affected by the time required to transmit bundle B, much as in the computation done in the forwarding stage. Specifically, line 16 could be replaced by $arr\_time+ = C.owlt + owlt_{mgn} + (B.EVC/C.rate)$, where $(B.EVC/C.rate)$ is the total time required to transmit bundle B given the contact's asserted transmission data rate.

Acknowledgments

This research has received support from the ERC Advanced Grant 695614 (POWVER) and by the DFG Grant 389792660, as part of TRR 248 (https://perspicuous-computing.science). Part of this work has been developed while Dr. Juan Fraire was visiting Politecnico di Torino. Part of this research was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Government sponsorship acknowledged.

REFERENCES

[1] M. N. Sweeting, "Modern small satellites-changing the economics of space," *Proceedings of the IEEE*, vol. 106, no. 3, pp. 343–361, 2018. [Online]. Available: https://doi.org/10.1109/JPROC.2018.2806218

[2] G.-P. Liu and S. Zhang, "A survey on formation control of small satellites," *Proceedings of the IEEE*, vol. 106, no. 3, pp. 440–457, 2018.

[3] R. Radhakrishnan, W. W. Edmonson, F. Afghah, R. M. Rodríguez-Osorio, F. Pinto, and S. C. Burleigh, "Survey of inter-satellite communication for small satellite systems: Physical layer to network layer view," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 4, pp. 2442–2473, 2016. [Online]. Available: https://doi.org/10.1109/COMST.2016.2564990

[4] O. Brown and P. Eremenko, "The value proposition for fractionated space architectures," in *AIAA-2006-7506, AIAA Space 2006*, San Jose, CA, 2006.

[5] J. Alvarez and B. Walls, "Constellations, clusters, and communication technology: Expanding small satellite access to space," in *2016 IEEE Aerospace Conference*, 2016, pp. 1–11.

[6] J. R. Kopacz, R. Herschitz, and J. Roney, "Small satellites an overview and assessment," *Acta Astronautica*, vol. 170, pp. 93 – 105, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0094576520300540

[7] A. T. Klesh, J. D. Baker, J. Bellardo, J. Castillo-Rogez, J. Cutler, L. Halatek, E. G. Lightsey, N. Murphy, and C. Raymond, "Inspire: Interplanetary nanospacecraft pathfinder in relevant environment," in *AIAA SPACE 2013 Conference and Exposition*, 2013, p. 5323.

[8] J. Schoolcraft, A. Klesh, and T. Werne, "Marco: interplanetary mission development on a cubesat scale," in *Space Operations: Contributions from the Global Community*. Springer, 2017, pp. 221–231.

[9] N. Bosanac, A. D. Cox, K. C. Howell, and D. C. Folta, "Trajectory design for a cislunar cubesat leveraging dynamical systems techniques: The lunar icecube mission," *Acta Astronautica*, vol. 144, pp. 283–296, 2018.

[10] J. O. Burns, B. Mellinkoff, M. Spydell, T. Fong, D. A. Kring, W. D. Pratt, T. Cichan, and C. M. Edwards, "Science on the lunar surface facilitated by low latency telerobotics from a lunar orbital platform-gateway," *Acta Astronautica*, vol. 154, pp. 195–203, 2019.

[11] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, "Delay-tolerant networking: An approach to interplanetary internet," *Comm. Mag.*, vol. 41, no. 6, pp. 128–136, June 2003. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2003.1204759

[12] Z. Zhang, "Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges," *IEEE Communications Surveys Tutorials*, vol. 8, no. 1, pp. 24–37, 1Q 2006.

[13] R. Diana, E. Lochin, L. Franck, C. Baudoin, E. Dubois, and P. Gelard, "A dtn routing scheme for quasi-deterministic networks with application to leo satellites topology," in *2012 IEEE Vehicular Technology Conference (VTC Fall)*, September 2012, pp. 1–5.

[14] G. Araniti, N. Bezirgiannidis, E. Birrane, I. Bisio, S. Burleigh, C. Caini, M. Feldmann, M. Marchese, J. Segui, and K. Suzuki, "Contact graph routing in DTN space networks: overview, enhancements and performance," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 38–46, 2015.

[15] S. Burleigh, "Interplanetary overlay network: An implementation of the dtn bundle protocol," 2007.

[16] M. Feldmann and F. Walter, "μPCN—A bundle protocol implementation for microcontrollers," in *2015 International Conference on Wireless Communications & Signal Processing (WCSP)*. IEEE, 2015, pp. 1–5.

[17] Consultative Committee for Space Data Systems (CCSDS), "Schedule-aware bundle routing (SABR) (blue book, recommended standard CCSDS 734.3-B-1," https://public.ccsds.org/Pubs/734x3b1.pdf, July 2019.

[18] S. Burleigh, "Contact graph routing, IETF-Draft draft-burleigh-dtnrg-cgr-00," December 2009.

[19] S. Burleigh, "Contact graph routing, IETF-Draft draft-burleigh-dtnrg-cgr-01," July 2010.

[20] J. A. Fraire, P. G. Madoery, F. Raverta, J. M. Finochietto, and 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*, September 2017.

[21] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in uav communication networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1123–1152, 2015.

[22] N. Benamar, K. D. Singh, M. Benamar, D. E. Ouadghiri, and J.-M. Bonnin, "Routing protocols in vehicular delay tolerant networks: A comprehensive survey," *Computer Communications*, vol. 48, pp. 141 – 158, 2014, opportunistic networks. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366414001212

[23] J. Hom, L. Good, and Shuhui Yang, "A survey of social-based routing protocols in delay tolerant networks," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, January 2017, pp. 788–792.

[24] F. Z. Benhamida, A. Bouabdellah, and Y. Challal, "Using delay tolerant network for the internet of things: Opportunities and challenges," in *2017 8th International Conference on Information and Communication Systems (ICICS)*, April 2017, pp. 252–257.

[25] J. Partan, J. Kurose, and B. N. Levine, "A survey of practical issues in underwater networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 11, no. 4, pp. 23–33, October 2007. [Online]. Available: http://doi.acm.org/10.1145/1347364.1347372

[26] J. A. Fraire, M. Feldmann, and S. C. Burleigh, "Benefits and challenges of cross-linked ring road satellite networks: A case study," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–7.

[27] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-tolerant networking architecture," Internet Requests for Comments, RFC Editor, RFC 4838, April 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4838.txt

[28] J. A. Fraire, M. Feldmann, F. Walter, E. Fantino, and S. C. Burleigh, "Networking in interstellar dimensions: Communicating with trappist-1," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 4, pp. 1656–1665, August 2019.

[29] 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, November 2011.

[30] C. Caini and R. Firrincieli, *DTN for LEO Satellite Communications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 186–198. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23825-3_18

[31] C. Caini and R. Firrincieli, "Application of contact graph routing to leo satellite dtn communications," in *2012 IEEE International Conference on Communications (ICC)*, June 2012, pp. 3301–3305.

[32] M. Ramadas, S. Burleigh, S. Farrell *et al.*, "Licklider transmission protocol-specification," Internet Requests for Comments, RFC Editor, RFC 5326, September 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5050.txt

[33] Consultative Committee for Space Data Systems (CCSDS), "Proximity-1 space link protocol - rationale, architecture and scenarios (green book, informational report CCSDS 210.0-G-2," https://public.ccsds.org/Pubs/210x0g2.pdf, December 2013.

[34] Consultative Committee for Space Data Systems (CCSDS), "Unified space data link protocol (blue book, recommended standard CCSDS 732.1-B-1," https://public.ccsds.org/Pubs/732x1b1.pdf, October 2018.

[35] K. Scott and S. Burleigh, "Bundle protocol specification," Internet Requests for Comments, RFC Editor, RFC 5050, November 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5050.txt

[36] Consultative Committee for Space Data Systems (CCSDS), "Ccsds bundle protocol specification (blue book, recommended standard CCSDS 734.2-B-1," https://public.ccsds.org/Pubs/734x2b1.pdf, September 2015.

[37] W.-B. Pöttner, J. Morgenroth, S. Schildt, and L. Wolf, "Performance comparison of dtn bundle protocol implementations," in *Proceedings of the 6th ACM workshop on Challenged networks*. ACM, 2011, pp. 61–64.

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

[39] "Micro Planetary Communication Network (uPCN) software home page," https://www.upcn.eu/.

[40] W. Ivancic, W. M. Eddy, D. Stewart, L. Wood, P. Holliday, C. Jackson, and J. Northam, "Experience with delay-tolerant networking from orbit," in *2008 4th Advanced Satellite Mobile Systems*, August 2008, pp. 173–178.

[41] K. Suzuki, S. Inagawa, J. Lippincott, and A. Cecil, "Jaxa-nasa interoperability demonstration for application of dtn under simulated rain attenuation," in *SpaceOps 2014 Conference*, 2014, p. 1920.

[42] A. Jenkins, S. Kuzminsky, K. Gifford, R. Pitts, and K. Nichols, "Delay/ disruption-tolerant networking: Flight test results from the international space station," in *2010 IEEE Aerospace Conf.*, March 2010, pp. 1–8.

[43] J. Wyatt, S. Burleigh, R. Jones, L. Torgerson, and 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*, July 2009, pp. 187–196.

[44] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. New York, NY, USA: ICST, 2009.

[45] J. Morgenroth, S. Schildt, and L. Wolf, "Hydra: Virtualized distributed testbed for dtn simulations," in *Proceedings of the fifth ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, 2010, pp. 71–78.

[46] Z. Zhang, Z. Jin, H. Chen, Y. Shu, and C. Zhao, "Design and implementation of a delay-tolerant network emulator based in qualnet simulator," in *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*. IEEE, 2009, pp. 1–4.

[47] C. Caini, R. Firrincieli, D. Lacamera, and M. Livini, "Virtualization technologies for dtn testbeds," in *International Conference on Personal Satellite Services*. Springer, 2010, pp. 276–287.

[48] I. Komnios, I. Alexiadis, N. Bezirgiannidis, S. Diamantopoulos, S.-A. Lenas, G. Papastergiou, and V. Tsaoussidis, "Spice testbed: A dtn testbed for satellite and space communications," in *International Conference on Testbeds and Research Infrastructures*. Springer, 2014, pp. 205–215.

[49] Y. Li, P. Hui, D. Jin, and S. Chen, "Delay-tolerant network protocol testing and evaluation," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 258–266, 2015.

[50] J. A. Fraire, P. G. Madoery, and J. M. Finochietto, "On the design and analysis of fair contact plans in predictable delay-tolerant networks," *IEEE Sensors Journal*, vol. 14, no. 11, pp. 3874–3882, 2014.

[51] J. A. Fraire and J. Finochietto, "Routing-aware fair contact plan design for predictable delay tolerant networks," *Ad Hoc Networks*, vol. 25, pp. 303 – 313, 2015.

[52] J. A. Fraire, P. G. Madoery, and J. M. Finochietto, "Traffic-aware contact plan design for disruption-tolerant space sensor networks," *Ad Hoc Networks*, vol. 47, pp. 41 – 52, 2016.

[53] J. A. Fraire, P. G. Madoery, J. M. Finochietto, and G. Leguizamón, "An evolutionary approach towards contact plan design for disruption-tolerant satellite networks," *Applied Soft Computing*, vol. 52, pp. 446–456, 2017.

[54] P. G. Madoery, J. A. Fraire, and J. M. Finochietto, "Congestion management techniques for disruption-tolerant satellite networks," *International Journal of Satellite Communications and Networking*, vol. 36, no. 2, pp. 165–178, 2018.

[55] J. A. Fraire, G. Nies, H. Hermanns, K. Bay, and M. Bisgaard, "Battery-aware contact plan design for leo satellite constellations: The ulloriaq case study," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.

[56] J. A. Fraire, G. Nies, C. Gerstacker, H. Hermanns, K. Bay, and M. Bisgaard, "Battery-aware contact plan design for leo satellite constellations:the ulloriaq case study," *IEEE Transactions on Green Communications and Networking*, pp. 1–1, 2019.

[57] D. Zhou, M. Sheng, X. Wang, C. Xu, R. Liu, and J. Li, "Mission aware contact plan design in resource-limited small satellite networks," *IEEE Transactions on Communications*, vol. 65, no. 6, pp. 2451–2466, June 2017.

[58] J. A. Fraire and J. M. Finochietto, "Design challenges in contact plans for disruption-tolerant satellite networks," *IEEE Communications Magazine*, vol. 53, no. 5, pp. 163–169, May 2015.

[59] M. Demmer and K. Fall, "Dtlsr: Delay tolerant routing for developing regions," in *Proceedings of the 2007 Workshop on Networked Systems for Developing Regions*, ser. NSDR '07. New York, NY, USA: Association for Computing Machinery, 2007. [Online]. Available: https://doi.org/10.1145/1326571.1326579

[60] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, April 2006, pp. 1–11.

[61] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, ser. WDTN '05. New York, NY, USA: ACM, 2005, pp. 252–259. [Online]. Available: http://doi.acm.org/10.1145/1080139.1080143

[62] A. Vahdat and D. Becker, "Epidemic routing for partially-connected ad hoc networks," Tech. Rep., 2000.

[63] Y. Wu, S. Deng, and H. Huang, "Performance analysis of epidemic routing in dtns with limited forwarding times and selfish nodes," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 13, no. 3/4, pp. 254–263, July 2013. [Online]. Available: http://dx.doi.org/10.1504/IJAHUC.2013.055474

[64] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, July 2003. [Online]. Available: http://doi.acm.org/10.1145/961268.961272

[65] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 145–158, August 2004. [Online]. Available: http://doi.acm.org/10.1145/1030194.1015484

[66] S. Burleigh, "Dynamic routing for delay-tolerant networking in space flight operations," in *SpaceOps 2008 Conference*, 2008, p. 3406.

[67] E. J. Birrane, "Improving graph-based overlay routing in delay tolerant networks," in *2011 IFIP Wireless Days (WD)*. IEEE, 2011, pp. 1–6.

[68] E. J. B. III, "Building routing overlays in disrupted networks: inferring contacts in challenged sensor internetworks," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 11, no. 2-3, pp. 139–156, 2012.

[69] E. Birrane, "Contact graph routing extension block," Internet RFC, Internet Draft, October 2013. [Online]. Available: https://tools.ietf.org/id/draft-irtf-dtnrg-cgreb-00.txt

[70] J. Segui, E. Jennings, and S. Burleigh, "Enhancing contact graph routing for delay tolerant space networking," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, December 2011, pp. 1–6.

[71] 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.

[72] Z. Laitao, L. Yong, Z. Junxiang, W. Jing, T. Xiao, and Z. Jianguo, "Application of contact graph routing in satellite delay tolerant networks," *Chinese Journal of Space Science*, vol. 35, no. 1, pp. 116–125, 2014.

[73] J. A. Fraire, P. G. Madoery, J. M. Finochietto, P. Ferreyra, and R. Velazco, "Internetworking approaches towards along-track segmented satellite architectures," in *Wireless for Space and Extreme Environments (WiSEE), 2016 IEEE International Conference on*, October 2016, in Press.

[74] M. Marchese and F. Patrone, "A source routing algorithm based on CGR for DTN-nanosatellite networks," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[75] C. Krupiarz, C. Belleme, D. Gherardi, and E. Birrane, "Using smallsats and DTN for communication in developing countries," in *Proc. International Astronautical Congress (IAC-08. B4. 1.8)*, 2008.

[76] 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*, ser. ACWR '11. New York, NY, USA: ACM, 2011, pp. 219–224. [Online]. Available: http://doi.acm.org/10.1145/2185216.2185280

[77] M. Feldmann, J. A. Fraire, and F. Walter, "Tracking lunar ring road communication," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–7.

[78] N. Bezirgiannidis, C. Caini, D. D. P. Montenero, M. Ruggieri, and V. Tsaoussidis, "Contact graph routing enhancements for delay tolerant space communications," in *2014 7th Advanced Satellite Multimedia Systems Conf. and the 13th Signal Processing for Space Comms. Workshop (ASMS/SPSC)*, September 2014, pp. 17–23.

[79] N. Bezirgiannidis, C. Caini, and V. Tsaoussidis, "Analysis of contact graph routing enhancements for DTN space communications," *Int. Journal of Satellite Coms. and Networking*, vol. 34, no. 5, pp. 695–709, 2016.

[80] J. A. Fraire, P. Madoery, and 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*, October 2014, pp. 1–7.

[81] J. A. Fraire, P. Madoery, J. M. Finochietto, and E. J. Birrane, "Congestion modeling and management techniques for predictable disruption tolerant networks," in *Local Computer Networks (LCN), 2015 IEEE 40th Conference on*, October 2015, pp. 544–551.

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

[83] G. Wang, S. C. Burleigh, R. Wang, L. Shi, and Y. Qian, "Scoping contact graph-routing scalability: Investigating the system's usability

in space-vehicle communication networks," *IEEE Vehicular Technology Magazine*, vol. 11, no. 4, pp. 46–52, December 2016.

[84] P. Madoery, P. Ferreyra, J. Fraire, F. Gomez, J. Barrientos, and R. Velazco, "Enhancing Contact Graph Routing Forwarding Performance for Segmented Satellites Architectures," in *1st IAA Latin American Symposium on Small Satellites*, Argentina, March 2017.

[85] P. G. Madoery, J. A. Fraire, F. D. Raverta, J. M. Finochietto, and S. C. Burleigh, "Managing Routing Scalability in Space DTNs," in *2018 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, December 2018, pp. 177–182.

[86] N. Alesi, "Hierarchical Inter-Regional Routing Algorithm for Interplanetary Networks," Master's thesis, School of Engineering and Architecture, Department of Computer Science and Engineering, Bologna, Italy, 2018.

[87] J. A. Fraire, P. Madoery, S. Burleigh, M. Feldmann, J. Finochietto, A. Charif, N. Zergainoh, and R. Velazco, "Assessing contact graph routing performance and reliability in distributed satellite constellations," *Journal of Computer Networks and Communications*, vol. 2017, 2017.

[88] P. Madoery, F. Raverta, J. Fraire, and J. Finochietto, "On the performance analysis of disruption tolerant satellite networks under uncertainties," in *Proceedings of the 2017 XVII RPIC Workshop*, September 2017.

[89] P. G. Madoery, F. D. Raverta, J. A. Fraire, and J. M. Finochietto, "Routing in space delay tolerant networks under uncertain contact plans," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[90] F. D. Raverta, R. Demasi, P. G. Madoery, J. A. Fraire, J. M. Finochietto, and P. R. D'Argenio, "A Markov Decision Process for Routing in Space DTNs with Uncertain Contact Plans," in *2018 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, December 2018, pp. 189–194.

[91] P. R. D'Argenio, J. A. Fraire, and A. Hartmanns, "Sampling distributed schedulers for resilient space communication," in *NASA Formal Methods - 12th International Symposium, NFM 2020, Moffett Field, CA, USA, May 11-15, 2020, Proceedings (In Press)*, 2020.

[92] J. A. Fraire, P. G. Madoery, A. Charif, and J. M. Finochietto, "On route table computation strategies in delay-tolerant satellite networks," *Ad Hoc Networks*, vol. 80, pp. 31–40, 2018.

[93] O. De Jonckère, "Efficient contact graph routing algorithms for unicast and multicast bundles," in *2019 IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, July 2019, pp. 87–94.

[94] S. Dhara, C. Goel, R. Datta, and S. Ghose, "CGR-SPI: A New Enhanced Contact Graph Routing for Multi-source Data Communication in Deep Space Network," in *2019 IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, July 2019, pp. 33–40.

[95] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *2005 ACM SIGCOMM Workshop on DTN*, 2005, pp. 252–259.

[96] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility," in *Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07)*. IEEE, 2007, pp. 79–85.

[97] M. Demmer and K. Fall, "Dtlsr: delay tolerant routing for developing regions," in *Proceedings of the 2007 workshop on Networked systems for developing regions*, 2007, pp. 1–6.

[98] S. Burleigh, "Compressed Bundle Header Encoding (CBHE)," Internet Requests for Comments, RFC Editor, RFC 6260, May 2011, http://www.rfc-editor.org/rfc/rfc6260.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6260.txt

[99] T. Hossmann, F. Legendre, and T. Spyropoulos, "From contacts to graphs: Pitfalls in using complex network analysis for DTN routing," in *IEEE INFOCOM Workshops 2009*. IEEE, 2009, pp. 1–6.

[100] S. Merugu, M. H. Ammar, and E. W. Zegura, "Routing in space and time in networks with predictable mobility," Georgia Institute of Technology, Tech. Rep., 2004. [Online]. Available: http://hdl.handle.net/1853/6492

[101] A. Sekhar, B. S. Manoj, and C. S. R. Murthy, "MARVIN: movement-aware routing over interplanetary networks," in *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.*, October 2004, pp. 245–254.

[102] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, December 1959. [Online]. Available: http://dx.doi.org/10.1007/BF01386390

[103] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *Journal of the ACM (JACM)*, vol. 34, no. 3, pp. 596–615, 1987.

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

[105] E. L. Lawler, "A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem," *Management science*, vol. 18, no. 7, pp. 401–405, 1972.

[106] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001.

[107] *Mesh Routing and Recovery Framework*. John Wiley & Sons, Ltd, 2007, ch. 3, pp. 61–80. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470032985.ch3

[108] E. Stassinopoulos and J. P. Raymond, "The space radiation environment for electronics," *Proceedings of the IEEE*, vol. 76, no. 11, pp. 1423–1442, 1988.