# Schedule-Aware Bundle Routing: Analysis and Enhancements

Carlo Caini, Gian Marco De Cola, Lorenzo Persampieri

**Abstract — The Delay-/Disruption-Tolerant Networking architecture (DTN) was designed to cope with challenges such as long delays and intermittent connectivity. To exploit the a priori knowledge of contacts, typical of space networks, NASA-JPL designed and included in ION (its DTN protocol suite) the Contact Graph Routing (CGR) algorithm. This paper studies the latest version, recently standardized as Schedule-Aware Bundle Routing (SABR) within the Consultative Committee for Space Data Systems (CCSDS). The first part of the paper is devoted to the algorithm analysis, which distinguishes three logical phases to examine sequentially. Following this comprehensive study, three enhancements are proposed, which aim to improve SABR accuracy and resistance against possible loops. They are studied on a simple but challenging DTN topology, implemented on a virtual GNU/Linux testbed. Tests are performed by running the latest version of ION and an independent implementation of SABR developed by the authors, Unibo-CGR. The numerical results are then examined in detail to highlight both SABR mechanisms and the advantages offered by the proposed enhancements.**

***Index Terms— Delay-/Disruption-Tolerant Networking, Inter-Planetary Networking, CGR, SABR, Bundle Protocol.***

## I. INTRODUCTION

SPACE links are characterized by many challenges, such as long propagation delay, scheduled link intermittency, possible link losses and asymmetric bandwidth. In the late 90s' NASA-JPL researchers, who aimed to build an interplanetary internet, realized that it would be necessary to build a new networking architecture to cope with space problems. As some of the challenges are common to terrestrial "challenged networks", such as those in remote areas, military tactical networks, underwater networks, etc., the research was almost immediately broadened to find a common solution, called Delay-/Disruption-Tolerant Networking, (DTN) [1], [2].

DTN architecture extends TCP-IP architecture by inserting the new Bundle layer between Application and (usually) Transport. This new layer, and corresponding Bundle Protocol (BP) [3], should be implemented on the end-nodes and on some selected intermediate nodes. The new layer forms an overlay, with its own addresses and routing.

DTN standardization started inside IRTF (Internet Research Task Force), but after several years of research and experiments, it moved to the IETF (Internet Engineering Task Force) DTN group [4], where a new BP version (bpv7) is about to be standardized [5]. In parallel, DTN protocols are tailored for space applications by CCSDS (Consultative Committee for Space Data Systems) [6], [7], a standardization body consisting of all major space agencies. DTN protocols have been tested in space for many years and are at present used on the International Space Station (ISS) for experimental data delivery to Earth [8]; they should be part of future space missions.

Concerning DTN routing, intermittent connectivity and possible long delays demand ad hoc solutions, as a timely exchange of information between nodes is generally impossible [9]. In this regard, we must distinguish between terrestrial and space environments [10]: the former are characterized by random connectivity, as opportunity of transmissions (contacts) are generally related to the random motion of terrestrial nodes (cars, pedestrians, etc.); the latter, by scheduled intermittent connectivity, where contacts are known in advance, because they are due to the predictable motion of planets and spacecraft. While for terrestrial applications, there is a wide variety of possible solutions [10], generally based on some form of moderate flooding , for space environments the almost sole solution is Contact Graph Routing (CGR), developed by NASA-JPL [11]-[15]. This algorithm uses scheduled contacts to find the best path to destination, a much more challenging task than it may appear, as will be shown in the paper. Given the complexity of the problem, CGR has evolved continuously since its appearance, in parallel with new versions of ION, the NASA-JPL implementation of DTN protocols [16], which is the de facto reference for space applications, although not the sole. This is largely due to the fact that ION is available as free software [17]. Recently, a milestone was reached when the latest version was standardized by CCSDS under the name SABR (Schedule-Aware Bundle Routing) [18], now included in the latest ION releases.

The first aim of this paper is to make an in-depth analysis of the new SABR version, distinguishing between its logical phases and discussing conditions for optimality. This analysis is then followed by the proposal of three enhancements, to improve the accuracy of SABR predictions and to introduce both reactive and proactive measures against possible loops. These enhancements have been included in Unibo-CGR, an independent implementation of SABR, developed by the authors and then analysed by means of a GNU/Linux testbed consisting of several virtual machines running ION-4.0.0. A detailed study of numerical results highlights the possible

Manuscript received XX, 2020.

Carlo Caini, Gian Marco De Cola and Lorenzo Persampieri are with the University of Bologna, Italy, and may be contacted at carlo.caini@unibo.it, gianmarco.decola@studio.unibo.it and lorenzo.persampieri@studio.unibo.it.

benefits of these enhancements.

## II. DTN Architecture – Delay-/Disruption-Tolerant-Networing

DTN architecture [1], [2] is based on the introduction of the Bundle layer, usually above the Transport layer of the ISO/OSI model, whose scope is however redefined. A node that implements the BP [3] is called a DTN "node" (not all nodes of the network are required to be DTN nodes), and a DTN hop is the segment of the end-to-end path between two consecutive DTN nodes. In DTN architecture, the Transport is no longer end-to-end but restricted to one DTN hop. This is essential to allow the use of different transport protocols on different segments of the end-to-end path, as shown in Figure 1 with reference to a 3-hop path. An example is useful to clarify this point. Let us consider an Earth-to-Moon connection, consisting of three segments: one terrestrial segment, from the source to a terrestrial gateway to space, a space segment between one Earth and one Moon gateway, a lunar segment, between the lunar gateway and destination. While the first and last segments are not challenged, the space segment is characterized by an RTT of about 2.5s, incompatible with TCP. To cope with this delay, it is necessary to use a specialized Transport protocol, such as LTP [19],[20]. Conversely, the use of LTP on terrestrial and lunar segments is contraindicated, as LTP lacks congestion control. The use of different transport protocols on different network segments is the key to satisfactory performance in most challenged networks, and this is the first major DTN architecture benefit. The second, maybe more obvious, is the ability to store data on board DTN nodes, essential to cope with link intermittency [10].

BP is allowed to work on top of whichever transport protocol for which a "Convergence Layer Adapter" (CLA) is implemented (Figure 1). This implies that the bundle protocol can work as an overlay over different networks, even implementing a non-TCP/IP stack.

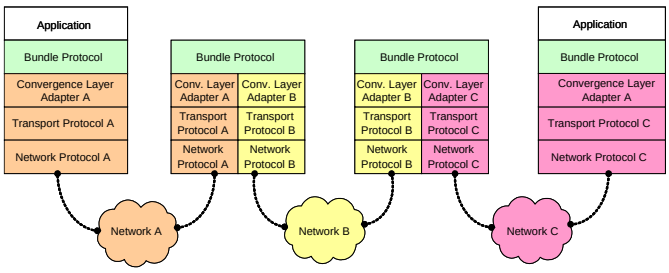


Figure 1: DTN architecture protocol stack on a possible 3-hop path.

Bundles have three "cardinal" priority classes: bulk, normal and expedited [1]. The ECOS (Extended Class Of Service) draft [21], implemented in ION, has, however, introduced a further subdivision of the expedited class into many "ordinal" priorities and other quality-of-service flags, among which the "critical" one is of particular interest to SABR. The aim of this priority-based strategy is to deliver bundles with the highest priority first. To this end, each node implements three queues towards each neighbour node, one for each cardinal priority, to send higher priority bundles first when a contact opens (analogously to first class passengers at boarding). Cardinal priorities, however, also have an impact on routing, as the presence of previous contact allocations to lower priority bundles is neglected by SABR, as if lower priority bundles were completely non-existent, or "transparent", to higher-priority ones. Thus, it may happen that a high priority bundle is allocated by SABR to a contact that is already fully booked by lower priority bundles (as if first-class passengers could grab the seats of tourist-class ones).

## III. Analysis of the SABR algorithm

Since its introduction, the CGR algorithm has incorporated new features at nearly each new version of ION [11]-[15]. Here we will focus on the SABR version [18].

### A. The Contact Graph

Every node uses contact plan information, i.e. contacts and ranges between nodes, to build a contact graph. This is searched, e.g. by Dijkstra's algorithm, to find the best succession of contacts leading to bundle destination. The criterion for the best route is the earliest arrival time, The contact graph is a direct acyclic graph, where vertex are contacts (not DTN nodes!) and edges represent episodes of data retention at a given node, waiting for a subsequent contact to start. An analogy with planes can help. Nodes of the graphs are flights, and edges are the waiting times in airports. Because vertices are contacts, a CGR route is a sequence of contacts (the flights that a passenger must take to get to destination), not a geographical route, i.e. the sequence of DTN nodes to be visited (the airports). A CGR route implies a geographical route, but not vice versa [15], [18].

CGR is a sophisticated algorithm, thus for a comprehensive treatment we shall divide it into three main logical phases, as in [22].

### B. Phase 1: route computation

Starting from the graph of contacts, the routes (sequence of contacts) that offer the earliest arrival time are computed. The SABR standard leaves most aspects of this phase to the implementation, including the way the contact graph is scanned and how many routes have to be computed, i.e. when to stop the search. The ION implementation, our reference, uses Yen's variant [23] of Dijkstra's algorithm and stops after computing only one route. It is worth stressing that Dijkstra contact search does not consider either bundle characteristics or the state of the network (queues, etc.), but leaves them to the next phase. For computational reasons, it is opportune to compute routes to destination D only when necessary; therefore, routes to D are computed for the first time and inserted in a *computed route* list, only when the first bundle to D starts to be processed by CGR on the current node. As shown in the simplified SABR flow chart (Figure 2), this phase is generally skipped by subsequent bundles directed to the same destination, essentially to avoid further Dijkstra searches. However, phase 1 is re-entered if phase 2 is unable to find a viable route among the computed routes (see below).

### C. Phase 2: selection of candidate routes

The second phase is validating the computed routes (one or more) calculated in phase 1, considering the specific characteristics of the current bundle, e.g. priority, lifetime and dimension. The routes that pass all checks are inserted into a *candidate route* list, for phase 3. Some of the exclusion rules from the standard are listed below:

- "Each route whose entry node is a member of the excluded nodes list shall be ignored". As the previous node is normally inserted into the excluded nodes list, this rule prevents a bundle form being sent back, to avoid "ping-pong" instabilities.
- "Each route for which projected bundle arrival time (PBAT) is after the bundle's expiration time shall be ignored". This is probably the most important check; it can only be carried out in this phase because both PBAT and expiration time depend on bundle characteristics (dimension, priority, lifetime).
- "Each route that includes any contact indicating transmission to X (the local node) should be avoided, unless X and D (the destination node) are identical". This rule is necessary to exclude routes implying a geographical loop. These loops are actually possible only because DTN routes calculated by Dykstra consist of a series of contacts, instead of nodes.
- "Each route that is depleted with regard to the bundle's level of priority shall be ignored". In practice, all contacts of the route must have enough residual volume to accommodate the current bundle, unless the "anticipatory fragmentation" option is on. As the utility of this option is questionable, from now we will consider it off. Note that the residual volume, called MTV (Maximum Transmission Volume) in the standard, depends on current bundle priority, as bundles with lower priority are neglected by SABR, as if they were non-existent, or "transparent" to the current bundle. The check on residual volumes of all contacts on the route is one of the most important novelties introduced by SABR (previous versions checked only the residual volume of the first contact) [14].

If after all the checks the candidate route list is empty, phase 1 is re-entered to find additional computed routes, i.e. new chances of finding candidate routes (Figure 2).

An important point is that if the set of computed routes consists of only one route, as normally in ION, phase-2 is reduced to a viability check on the sole computed route.

### D. Phase 3: bundle forwarding

This phase distinguishes between standard and critical bundles.

#### 1) Standard bundles

If the bundle is not flagged as critical, the best candidate route is determined by applying, sequentially, 4 figures of merit, i.e. PBAT, number of contacts, route termination-time and entry node number [18]. In practice, the fastest route to destination is chosen among all candidate routes (shortest PBAT), and, in case of parity, the route with the fewest contacts. The application of the last two tiebreak rules, based on route termination-time and entry node number, is quite uncommon. The bundle must only be forwarded to the entry node of the best candidate route, therefore there is no bundle replication for standard bundles.

#### 2) Critical bundles

If the bundle is critical, (hopefully a rare event), phase 3 selects one route for each neighbour, and a bundle copy is forwarded to each of them. If there is not any candidate route for a specific neighbour, phase 1 is re-entered, with the constraint that only routes starting from this neighbour must be looked for. The aim is twofold:

- to have at least one candidate route to each neighbour, provided that the contact plan allows for it.
- to send a copy to all these neighbours.

Actually it would be better to perform the first point in phase 2, in order to have a candidate route for each neighbour for critical bundles before entering phase 3. This is done in Unibo CGR implementation, to which Figure 2 refers.

This policy is an ingenious form of controlled flooding, where the a priori knowledge of contacts is used to avoid sending bundle copies to neighbours for which there is no chance of getting to destination in time. As for all forms of flooding, however controlled, its use should be strictly limited to exceptional cases.

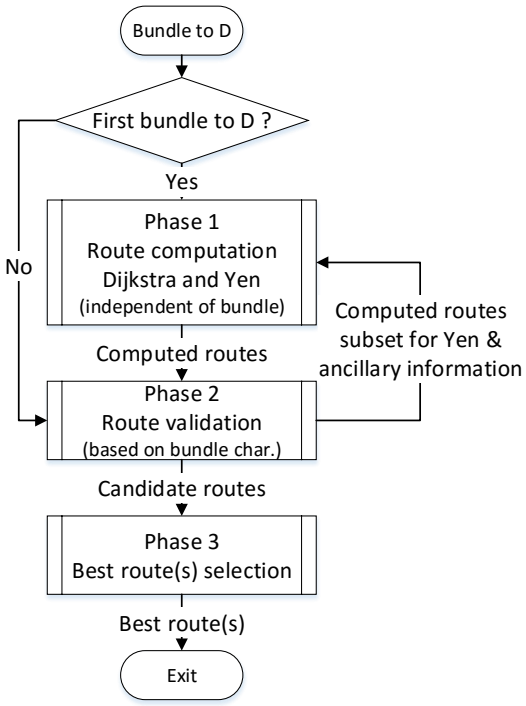


Figure 2: SABR logical flow chart, as implemented in Unibo-CGR.

### E. Route recomputation

In SABR, route computation and selection serve only to determine the neighbour (or neighbours for critical bundles) to which forward the current bundle. Once the bundle arrives at the next node, the route is recomputed from scratch. The rationale is that in DTN networks it is impossible to know the actual state of other nodes, because prompt updates are impeded

by long delays and link intermittency. Although a different approach, based on a form of moderate source routing, is possible [24], the focus here is on the standard.

### *F. On SABR optimality*

The next step is to discuss under which conditions SABR is able to select the best route. A full discussion of this complex topic can be found in [22]; here we will limit ourselves to the most significant points.

First, we must stress that phase 1 considers neither bundle properties (dimension, priorities, etc.) nor network state (expected bundle queues), so it is equivalent to a Dijkstra search considering a bundle of one byte (B) on an unloaded network. Therefore, if bundle length is negligible and the network unloaded, the route found by Dijkstra is the best; this cannot be taken for granted in any other circumstances.

Moving to phase 2, note that even if the route computed in phase 1 is the best for the first bundle to D, it could no longer be optimal for subsequent bundles (for which phase 1 is normally skipped), for a variety of reasons. Suffice to say that even for 1B bundle and a still unloaded network, the best route at time t0 and time t1 may differ. Keeping the old route for new bundles only because it is still viable is computationally efficient but clearly suboptimal. Particularly, with a loaded network, expected queuing delays due to already processed traffic can make the difference, thus they should theoretically be considered in the Dijkstra search (i.e. in phase 1) as envisaged in [12], [14]. This however, would require constant re-computation of routes, which is not a practical option. A trade-off between optimality and computational load is necessary, and that is what is actually done by SABR, which is therefore not optimal, but simply best effort.

## IV. Proposed Enhancements

The enhancements we are about to propose aim to improve the terms of this trade-off, by making SABR generally more accurate at the expenses of a limited computational effort.

### *A. "One-route-per-neighbor" enhancement*

In SABR, phase 1 is not re-entered unless necessary, to minimize Dijkstra searches. To be effective, this approach would require multiple candidate routes among which to select, in phase 3, the best, which can change depending on bundle characteristics and contact residual volumes. In the standard, when to stop phase 1 (i.e. after how many computed routes) is not specified, leaving this critical point to the implementation (in ION after one).

The "one-route-per neighbor" enhancement is inspired by the current treatment of critical bundles, and by pre-SABR versions of ION, which had a multiplicity of computed routes (ideally, one for each contact departing from the source node). When enabled, the one-route-per neighbor enhancement forces the calculation of a candidate route for each neighbor, even for standard bundles, to enhance the chances of selecting a better candidate route in phase 3. In brief, it aims to have a true election here, with multiple candidates, instead of a "Bulgarian" one. The drawback is that multiple candidate routes must be checked in phase 2, with an increase of the computational cost, depending on the number of proximate nodes, i.e. on the operational scenario. However, we think that this cost is well worth paying in most cases, to achieve satisfactory performance.

### *B. The "queue-delay" enhancement*

Traffic load, key for optimality, is only partially addressed by SABR in phase 2, where the Expected Volume Consumption (EVC) of the current bundle is compared with residual volumes of all contacts of the computed route, given by MTVs down-counters, one for each level of priority. However, the contact volume already allocated to previous bundles (i.e. the original contact volume minus the MTV) is not converted into an expected queuing delay. The queue-delay enhancement fills this gap by adding an expected queuing delay to the PBAT calculation, for each contact after the first. This delay is simply calculated as the total amount of bytes already allocated divided by the Tx rate declared in the contact. This expected delay can differ from what will really be encountered for contacts after the first, i.e. not departing from the current node (essentially because in DTN the current node cannot receive fast updates of traffic processed by other nodes). Adding these delays is a conservative strategy and by improving PBAT accuracy SABR performance should improve as well. Moreover, the additional computational cost of this enhancement is null.

Note that the first contact is skipped because its queuing delay is already precisely calculated by the ETO (Earliest Transmission Opportunity) variable. The present queue-delay enhancement simply extends the same concept to further contacts, inspired by the idea behind the ETO original proposal [12], [14], which worked on all contacts, not only on the first. Note, however, that the queue-delay enhancement significantly differs from the original ETO, as the delays are now applied after Dijkstra, i.e. in phase 2 rather than in phase 1. This is clearly less optimal, but, on the other hand, it saves further Dijkstra calculations.

### *C. Anti-loop enhancement*

SABR decisions are based on two kinds of information: general and local. While the information in the contact plan is general, as it is supposed to be common to all nodes, local information is given by all parameters that are known locally and that cannot be rapidly shared with other nodes, because of the well-known limitations of challenged networks. The more local information is used, the better the accuracy, but also the higher the risk of routing instabilities, such as loops. To counteract these, a prerequisite is being aware of their presence, when they happen. To this end, we introduced in ION a bundle extension, called RGR (Record Geographical Route) [22], which lists all nodes visited, thus extending the already existing "previous hop" extension, which reports only the last one [5]. By inspecting the RGR extension, it is possible to know if the current bundle has performed a loop and, if so, to take countermeasures to avoid a second loop, such as sending the bundle to a different neighbor. This is done by the reactive mechanism of the anti-loop enhancement.

A more ambitious goal, such as loop prevention, is pursued by the proactive anti-loop mechanism. When this is enabled, the list of already visited nodes is compared in phase 2 with those associated to contacts of a computed route. If one of the contact end-nodes coincides with a visited node, the checked route will close a loop. On the other hand, it should be considered that the "closing-loop" route could be the sole candidate route and that there is no certainty that the bundle is destined to loop, because following nodes might take different routing decisions. Therefore, we decided to still consider a closing-loop route as viable in phase 2, but introduced a loop warning flag, to be considered in phase 3. This flag prevents a closing-loop route from being selected, if other (non-flagged) candidate routes exist. The computational cost of the anti-loop enhancement is irrelevant, but the inclusion of the RGR extension necessarily increases the total bundle dimension. However, as the RGR consumes only very few bytes per visited node, this cost is negligible too.

## V. System Model Used in Tests

The primary aim of the test environment considered here is to highlight the different SABR behavior that the three enhancements induce. Although it does not pretend to be representative of any particular operational scenario, it has some relationship with both the LEO satellite scenario considered in [14] and the Martian scenario recently investigated in [25]. From the former it derives most contacts, from the latter the presence of an additional node. The high level of symmetry in this scenario is useful to test SABR ability to find the best route in the presence of nearly equivalent choices. Although simple, symmetry and the presence of nested contacts make it challenging for SABR, i.e. very suitable for testing the enhancements. To the same end, we considered ideal, i.e. without losses, the environment, which is instrumental to give a bundle by bundle description of the enhancements, as tests results are deterministic without losses.

### A. DTN layout

DTN layout (Figure 3: The DTN Layout used in this paper. Continuous lines denote continuous terrestrial links, dashed lines continuous space links, dotted lines intermittent space links. The Orbiter node is present only in anti-loop tests.) consists of 5 nodes, a Mission Control Centre (MCC), two Ground Stations (GS1 and 2), one Orbiter and a Space Asset (SA). Contacts between the space asset and GSs are intermittent.

### A. Reference scenario: DTN Layout and contact plan

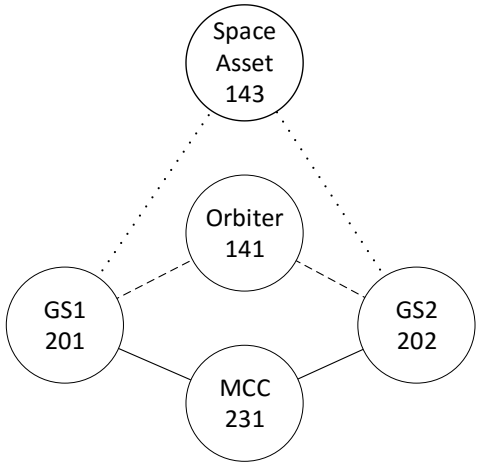


Figure 3: The DTN Layout used in this paper. Continuous lines denote continuous terrestrial links, dashed lines continuous space links, dotted lines intermittent space links. The Orbiter node is present only in anti-loop tests.

### B. Reference Contact Plan

We used the same reference contact plan in all tests (Table I), with the exception of Orbiter contacts, present only in anti-loop tests. In the Table:

- Contacts are listed in ascending order of sender and receiver EIDs, as in ION internal structures.
- Times are expressed differentially with respect to a reference time (ION startup).
- In the absence of a specific notation, continuous contacts are declared as very long contacts; all have the same high Tx rate and corresponding large volumes.
- A bi-directional contact is represented by a pair of unidirectional contacts, as in ION. For simplicity, all contacts here are symmetric, i.e. for each contact in one direction there is a corresponding contact in the opposite direction.
- The only intermittent contacts are between the Space Asset and the Ground Stations. To GS1 there is only one contact (3), short but fast. It is nested in the first contact to GS2 (4), which is longer but also 4 times slower. There is a second contact to GS2 (5), which makes CGR's task a bit more complex. It should not be used if CGR took the right decisions.
- For the sake of simplicity, all "range" delays are set to 1s, therefore they are not included in table

TABLE I. Reference Contact plan (ION format).

| Cont. | Sender (IPN) | Rec. (IPN) | Start Time [diff](s) | End Time [diff](s) | Data Rate (byte/s) |
|---|---|---|---|---|---|
| 1 | 141 | 201 | +0 | +36000 | 1250000 |
| 2 | 141 | 202 | +0 | +36000 | 1250000 |
| 3 | 143 | 201 | +60 | +85 | 64000 |
| 4 | 143 | 202 | +30 | +100 | 16000 |
| 5 | 143 | 202 | +120 | +140 | 16000 |
| 6 | 201 | 141 | +0 | +36000 | 1250000 |
| 7 | 201 | 143 | +60 | +85 | 64000 |
| 8 | 201 | 231 | +0 | +36000 | 1250000 |
| 9 | 202 | 141 | +0 | +36000 | 1250000 |
| 10 | 202 | 143 | +30 | +100 | 16000 |
| 11 | 202 | 143 | +120 | +140 | 16000 |
| 12 | 202 | 231 | +0 | +36000 | 1250000 |
| 13 | 231 | 201 | +0 | +36000 | 1250000 |
| 14 | 231 | 202 | +0 | +36000 | 1250000 |

### B. *ION CGR and Unibo-CGR*

All enhancements (but the anti-loop proactive mechanism) were first introduced in an experimental version of ION CGR (i.e. by modifying the original "libcgr.c" file). This was a development step towards the independent Unibo-CGR implementation of SABR, used in tests. A specific interface isolates the Unibo-CGR core from the ION environment, preserving full compatibility with it. All enhancements (and other features) can be selectively enabled, by means of "define" switches in the code. When they are off, Unibo-CGR behaves as ION CGR, except for the anticipatory fragmentation mechanism, which is not implemented. Unibo-CGR is aligned with the latest ION release (4.0.0), including the support for the new version of the bundle protocol, bpv7 [5]. Unibo-CGR is compatible with both the new and the old BP versions.

### C. *Virtual testbed, ION settings and generation tools*

Tests were carried out on a GNU/Linux virtual testbed created with Virtualbricks [26]. It consists of five Virtual Machines (VMs), one for each DTN node, connected by virtual switches and channel emulators. A propagation delay of 1s is inserted on all links and all links are assumed error free. In tests we used ION 4.0.0, with bpv7. At convergence layer, we have TCP and LTP on terrestrial and space links, respectively. To generate bundles and collect status reports [3] we used the DTNperf_3 tool [27]. Figures were obtained by plotting bundle status reports, but the analysis is largely supported by the new, very informative logs provided by Unibo-CGR. These proved essential in order to understand the rationale of SABR decisions in the most complex cases.

## VI. TESTS

In the following, we examine the three enhancements one-by-one. In all tests, layout and contacts are the same, but the Orbiter is present only in anti-loop tests. There is always only one traffic flow, consisting of 20 bundles of 100kB each, generated at the beginning of each experiment, either by the Space Asset or the MCC.

### A. *"One-route-per-neighbor"*

To evaluate the first enhancement, we consider the downlink case (SA as a source and MCC as a destination) and start from the SABR case (Figure 4). Bundles are routed as soon as generated (generated time series in the figure). In particular, phase 1 is performed immediately after the generation of the first bundle, at +17s, and stops once the first route (route A in the following) is computed by Dijkstra; this route is via GS2, consisting of contacts 4 (SA-GS2, starting at +30, see the figure) and 11 (GS2-MCC, not plotted as continuous). Route A is then verified in phase 2 and eventually selected in phase 3. Even though this is the sole candidate route, it is actually the best for the first bundle. Once the second bundle is generated, SABR is called again, but this time phase 1 is skipped (to save computational time). Route A is checked for the second bundle and selected as before, and this goes on until the residual volume available on contact number 4 becomes too small to accommodate a further bundle, which happens for bundle 11. After route A has failed, there are no more routes and phase 1 is re-entered, leading to the computation of route B, via GS1. Route B consists of contacts number 3 (the only SA-GS1 contact, starting at +60, see the figure) and 8 (GS1-MCC, continuous). Route B is then verified and selected for this and all remaining bundles. Summarizing, SABR (as implemented in ION) always keeps the first route selected until it fails, which may lead to suboptimal results, as proved here (unnecessary delayed delivery of many bundles, such as the last delivered via 202, disordered delivery, unbalanced load, early exhaustion of the SA-GS2 contact). Note that, although ordered bundle delivery is not an RFC requirement, it is always preferable from an operational point of view, as for UDP in Internet, to limit the degree of disorder as much as possible.

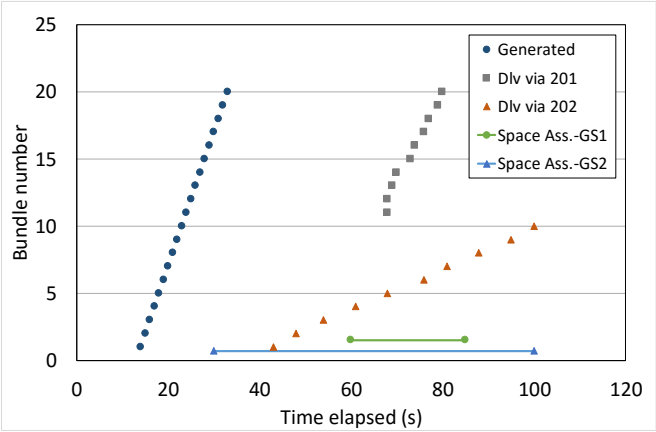


Figure 4: Downlink (SA to MCC), no Orbiter, SABR. Generated and delivered (Dlv) time series. The first 10 bundles are routed via GS2, the last 10 via GS1.

Better performance, in primis a faster delivery of all bundles, can be obtained by introducing some path diversity, in order to have multiple candidate routes in phase 3. The previous test is thus repeated by enabling the one-route-per-neighbor enhancement. The difference is now that when route A is passed to phase 2, phase 1 is immediately re-entered to find an alternate route via a different neighbor. This is accomplished by removing all contacts leading to GS2 before the second Dijkstra search. This way, route B, via GS1, is now computed before the

failure of route A. As there are no more neighbors, phase 2 goes on and validates both routes by calculating PBAT (i.e. the expected delivery time) of the current bundle for each route. A third route, consisting of contacts number 5 (the second SA-GS2 contact) and 11 would be possible, but as there is already a candidate route for each neighbor, it is not considered. This differs from pre-SABR versions of CGR, which computed one route for each contact departing from the source node, with huge computational effort. In phase 3, route A is then selected, as it offers a shorter PBAT than route B, for the first bundle. The same happens for all bundles, but the choice alternates between Routes A and B, depending on the queuing delay on the first hop, given by the ETO variable. This eventually leads to a more efficient concurrent use of both routes, as shown in Figure 5. There are three clear advantages: bundle delivery is much more ordered (very close to optimal); all bundles are delivered before the end of the nested contact; both space contacts still have enough residual volume to accommodate possible further bundles, should they ever be generated.

Finally, let us note that the other two enhancements would make no difference here, because queuing delays on second hops (terrestrial) are negligible. This would also make loops impossible, should the Orbiter be added (results would be exactly the same as here).

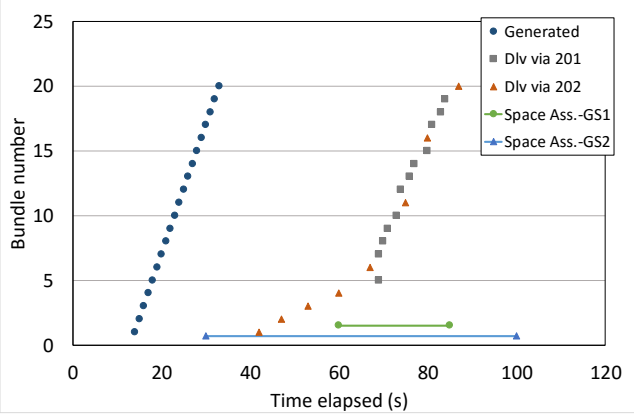


Figure 5: Downlink (SA to MCC), no Orbiter, SABR with one-route per neighbor. Generated and delivered (Dlv) time series. Bundles are routed via GS1 and GS2 concurrently thanks to the enhancement.

### *B. "Queue-delay"*

To demonstrate the utility of the second enhancement, it is necessary to reverse direction and examine the uplink case (MCC as a source and SA as a destination). The Orbiter is still temporarily absent, and we start as before from the SABR case (Figure 6). Looking at the figure, we can see that the results are actually the same as in the downlink case (Figure 4). The difference is that now the routes consist of the opposite-direction contacts: route A, via GS2, of contacts number 14 and 10; route B, via GS1, of contacts 13 and 7.

Now, if the one-route-per-neighbor enhancement was enabled alone, as in the downlink case, here the results would not improve, because what really matters is the expected queuing delay on intermittent contacts, which are now on the second hop, and not on the first as before, thus neglected here by the ETO variable. To include them in the PBAT calculation it is thus necessary to enable the "queue-delay" enhancement as well. Thanks to the more accurate PBAT calculation that follows, delivery results (Figure 7) are now qualitatively the same as those obtained in the downlink case (Figure 5) and with the same advantages.

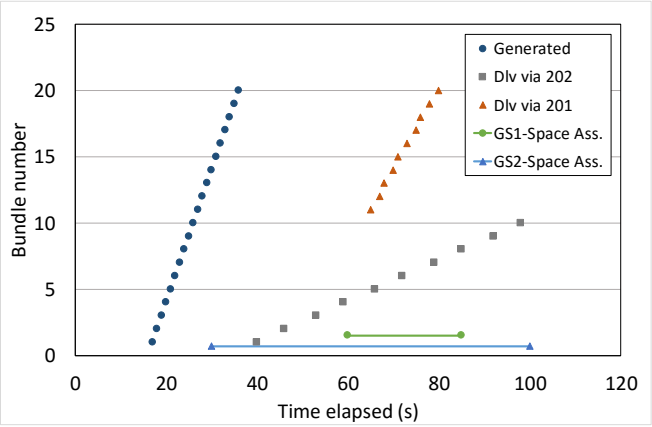


Figure 6: Uplink (MCC to SA), no Orbiter, SABR. Generated and delivered (Dlv) time series. The first 10 bundles are routed via GS2, the last 10 via GS1. Results are the same as in downlink (Figure 4).

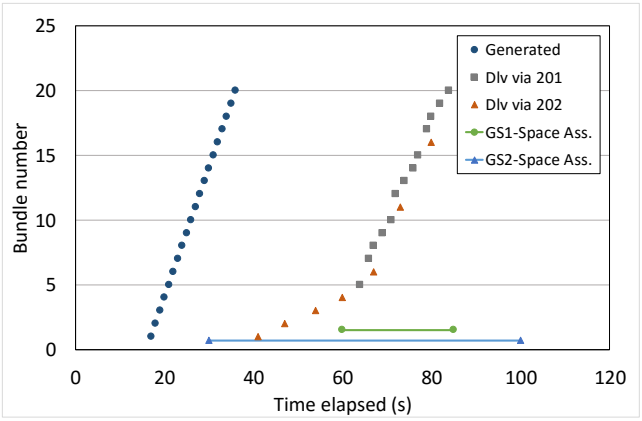


Figure 7: Uplink (MCC to SA), no Orbiter, SABR with one-route per neighbor and queue-delay. Generated and delivered (Dlv) time series. Bundles are routed via GS1 and GS2 concurrently, as in downlink (Figure 5).

### *C. "Anti-loop" enhancement*

Loops in SABR are unavoidable. Their chances are increased by the presence of a high level of symmetry in the layout and by connectivity restrictions, more common in uplinks. By adding the Orbiter here, with continuous contacts, we deliberately make the uplink scenario unstable. In detail, when bundles routed by MCC arrive either at GS1 or GS2, the path to destination is recalculated from scratch. As each ground station is aware solely of the traffic processed locally, GS1 can think that the "grass is greener", i.e. that the better path is actually via GS2, and vice versa. This was true in previous queue-delay tests too, without the Orbiter, but forwarding a bundle to the other GS was then possible only via the MCC, i.e. going back, which was prevented by the anti-ping-pong feature. The insertion of the Orbiter now makes a second (continuous) path between GS1 and GS2 available. This makes the anti-ping-pong mechanism useless, and facilitates loops (MCC-GS1-Orbiter-GS2-MCC or vice versa). Conversely, we can say that in the absence of this second path, GS1 and GS2 can only confirm the route chosen by the source, which explains why we neglected routing on ground stations in all previous

experiments.

*1) SABR*

As usual, we study the SABR case first (Figure 8). The MCC selects route A for the first ten bundles and starts selecting route B for the others, as before. When the first ten arrive at GS2, the route is recomputed and the MCC choice confirmed, as the alternative, via Orbiter and GS1 is less appealing than direct delivery. Therefore, the bundles are enqueued to the Space Asset to be delivered regularly when contact 7 (GS2-SA) starts at +30. The problems arise when the remaining bundles are received and rerouted by GS1, starting from bundle 11. In fact, SABR on GS1 erroneously thinks that waiting here for the start of contact number 7 (GS1-SA) at +60, is worse than moving to GS2 to take contact 10 (GS2- SA) starting only at +30. This contact, however, is already fully booked by the first ten bundles (already arrived and processed at GS2 in the meantime), but GS1 is unaware of this, as traffic processed by other nodes is not visible. As a result, these bundles start being sent to GS2 via the Orbiter, triggering a series of loops (shown in the figure by the "Rcv 231", i.e. received by 231, series). The situation becomes chaotic when the first looping bundles arrive on MCC, and in SABR mix with newly generated bundles, starting from bundle 17. Unibo-CGR logs show what happens, but a detailed explanation would be too long here. Suffice to say that at the end of the day, two bundles loop twice (bundles 11 and 12) and five once (13-17) before being delivered. The most unlucky bundle (18), however, is never delivered. When it reaches MCC after the first loop (Rcv 231 marker at 37s) it is blocked, as SABR is no longer able to identify any candidate route for it, since the volume allocated by MCC on space contacts (7, 9 and 11) is never reintegrated if not used, as for looping bundles. When bundle 18 arrives on MCC, the MTVs of all space contacts appear exhausted by previously routed bundles, including the looping ones. However, we can see from Figure 9, the GS1-SA contact is still largely unused (only 5 bundles use it).

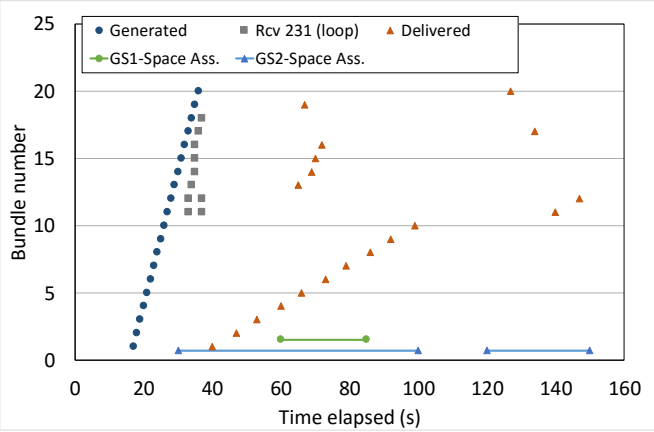


Figure 8: Uplink (MCC to SA) with Orbiter, SABR. Generated, received by 231 (loop indicator), and delivered time series. Bundles 1-10 are initially forwarded to GS2, bundle 11-19 to GS1, 20 to GS2. Of the 9 sent to GS1, two loop twice (11-12) and six once (13-18) (Rcv-231 series), before being delivered. One (18) is blocked on MCC and never delivered.

*2) Anti-loop proactive mechanism*

To solve the loop problem, it is necessary to add the anti-loop enhancement. Therefore we repeated the experiment with all enhancements enabled (Figure 9). From the logs we can fully understand what happens and why. Here, we can summarize saying that MCC routes the first ten bundles to GS2 (route A, contacts 14 and 10) and the others to GS1 (route B; contacts 13 and 7). Once on the ground stations, the bundles are rerouted and some are directly enqueued to the Space Asset (either with contact 7 or 10) while the others are sent to the opposite ground station via the Orbiter (Rcv-141 series). The proactive mechanism of the anti-loop enhancement proves to be effective and the bundles passed to the alternate ground stations are stopped, to prevent any loops. The results are much better than before: no loops, more ordered delivery (although much worse than in the analogous tests without the Orbiter), all bundles but one delivered in the first two contacts to SA (7 and 10), and only bundle 17 needs to wait for the start of the second GS2-SA contact (11).

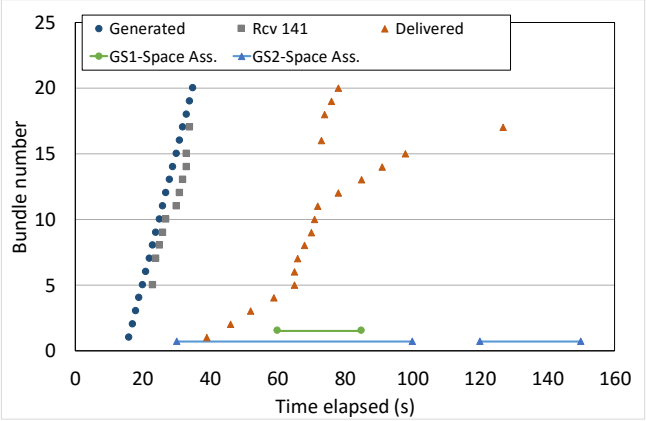


Figure 9: Uplink (MCC to SA) with Orbiter, SABR with one-route-per neighbour, queue-delay and anti-loop enhancements. Generated, received by 141 and delivered time series. The first 10 bundles are initially forwarded to GS2, the last 10 to GS1. Although many are swapped between GS1 and GS2 via the Orbiter (Rcv-141 series), the proactive anti-loop succeeds in preventing any loops. Only one bundle (17) needs the second GS2-SA contact to be delivered and none is deleted.

*3) Anti-loop reactive mechanism*

In the previous experiment, the proactive mechanism proved so successful that it prevented us from testing the reactive mechanism, for which the occurrence of a loop is obviously necessary. To test it and gain a better insight, we repeated the same experiment but disabled the proactive mechanism (Figure 10). Although some loops are present again (bundles 5, 7, 8, 9, 12 and 17), consecutive loops of the same bundle are avoided, as hoped. What is unexpected, is that now all bundles are delivered in the first two contacts to SA (contacts 7 and 10). This positive but counterintuitive result, however, cannot be generalized, depending on the peculiar situation of bundle 17, which actually benefits from the first loop. Instead of being stored on GS2 waiting for the second GS2-SA contact (11, starting at +120) as in the previous test (Figure 9), it is sent to MCC (closing the first loop) which in turn forwards it to GS1 (there is no other choice, as going back is prevented by the anti-ping-pong feature). When the bundle reaches GS1 for the second time, the reactive anti-loop forces it to be enqueued to the Space Asset instead to the Orbiter, which is (and was) the

right choice. Bundle 17 is eventually delivered on the GS1-SA contact (7), at about +80, i.e. much earlier than in the previous test, by what seems pure chance.

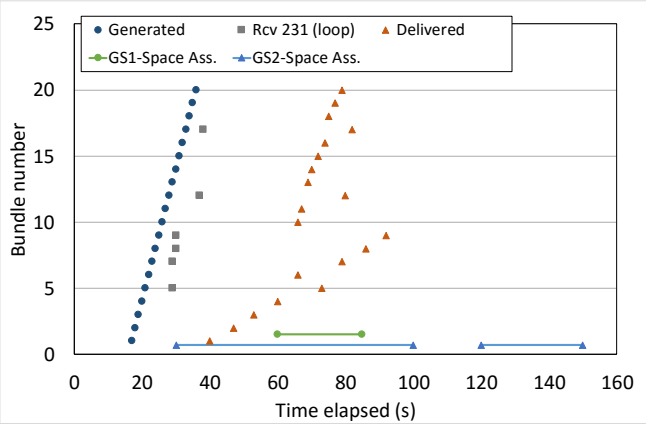


Figure 10: Uplink (MCC to SA) with Orbiter, SABR with one-route-per neighbour, queue-delay and anti-loop (reactive only) enhancement. Generated, received by 231 (loop indicator) and delivered time series. The first 10 bundles are initially forwarded to GS2, the last 10 to GS1. Many are swapped between GS1 and GS2 and often start looping (Rcv-231 series), but only once, as further loops are prevented by the reactive anti-loop mechanism. All bundles are eventually delivered in the first two contacts.

## VII. Conclusions

The paper focuses on the SABR version of CGR. The analysis in the first part of the paper illustrates the details of the algorithm and discusses conditions for optimality; these are not generally met, as SABR is a trade-off between optimality and computational complexity. In brief, SABR is a best-effort algorithm, based on some heuristics, a fact that is often misunderstood by inexpert users. The SABR analysis paves the way to the introduction of three enhancements, presented in the second part of the paper. The first two aim to improve the accuracy of SABR predictions, the last to counteract loops. All enhancements were included in Unibo-CGR, an independent SABR implementation developed by the authors, fully compatible with ION. Numerical results, achieved on a GNU/Linux testbed running the latest version of ION and Unibo-CGR, confirm the validity of the enhancements proposed.

## Acknowledgements

The authors would like to thank Scott Burleigh of NASA-JPL for his continuous support on ION and all the many valuable discussions on SABR.

## References

[1] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst and K. Scott, “Delay-tolerant networking: An approach to interplanetary internet", IEEE Commun. Mag., vol. 41, no. 6, pp. 128–136, Jun. 2003.

[2] V. Cerf , A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, H. Weiss “Delay-Tolerant Networking Architecture”, Internet RFC 4838, Apr. 2007.

[3] K. Scott, S. Burleigh, “Bundle Protocol Specification”, Internet RFC 5050, November 2007, http://tools.ietf.org/html/rfc5050.

[4] IETF DTN web site: https://datatracker.ietf.org/group/dtn/about/

[5] S. Burleigh, K. Fall, E. Birrane, “Bundle Protocol Version 7”, IETF Draft, May 2020, https://datatracker.ietf.org/doc/draft-ietf-dtn-bpbis/

[6] CCSDS 734.0-G-1 “Rationale, Scenarios, and Requirements for DTN in Space.” CCSDS Green Book, Issue 1, Aug. 2010. https://public.ccsds.org/Pubs/734x0g1e1.pdf

[7] CCSDS 734.2-B-1 “CCSDS Bundle Protcol Specificatons” CCSDS Blue Book, Issue 1, Sept.2015. https://public.ccsds.org/Pubs/734x2b1.pdf

[8] A. Schlesinger, B. M. Willman, L. Pitts, S. R. Davidson and W. A. Pohlchuck, “Delay/Disruption Tolerant Networking for the International Space Station (ISS),” 2017 IEEE Aerospace Conference, Big Sky, MT, 2017, pp. 1-14.

[9] S. Jain, K. Fall, and R. Patra, Routing in a delay tolerant network, in Proc. ACM SIGCOMM Portland, Aug./Sep. 2004, pp. 145–157.

[10] C. Caini, H. Cruickshank, S. Farrell, M. Marchese, “Delay- and Disruption-Tolerant Networking (DTN): An Alternative Solution for Future Satellite Networking Applications”, Proceedings of IEEE, Vol. 99, N. 11, pp.1980-1997, November 2011.

[11] E.Birrane, S.Burleigh and N. Kasch, “Analysis of the contact graph routing algorithm: Bounding interplanetary paths”, Acta Astronautica, Vol. 75, pp. 108-119, June-July 2012

[12] N. Bezirgiannidis, F. Tsapeli, S. Diamantopoulos, V. Tsaoussidis, “Towards Flexibility and Accuracy in Space DTN Communications”, in Proc. of of the 8th ACM MobiCom workshop on Challenged networks (ACM CHANTS 2013), Miami, FL, USA, Sept. 2013, pp1-7.

[13] G. Araniti, N. Bezirgiannidis, E. Birrane, I. Bisio, S. Burleigh, C. Caini, M. Feldmann, M. Marchese, J. Segui, K. Suzuki. “Contact Graph Routing in DTN Space Networks: Overview, Enhancements and Performance”, IEEE Commun. Mag., Vol.53, No.3, pp.38-46, March 2015.

[14] N. Bezirgiannidis C. Caini,V. Tsaoussidis, “Analysis of contact graph routing enhancements for DTN in space”, International Journal of Satellite Commun. and Networking, pp.695-709, No.34, Sept./Oct. 2016, DOI: 10.1002/sat.1138.

[15] S. Burleigh, C. Caini, J. J. Messina, M. Rodolfi, “Toward a Unified Routing Framework for Delay-Tolerant Networking”, in Proc. of IEEE WiSEE 2016, Aachen, Germany, Sept. 2016, pp. 82 - 86, DOI: 10.1109/WiSEE.2016.7877309

[16] S. Burleigh, "Interplanetary Overlay Network (ION) an Implementation of the DTN Bundle Protocol, In the Proc. of 4th IEEE Consumer Commun. and Networking Conference, 2007, pp. 222-226.

[17] ION code and manual: http://sourceforge.net/projects/ion-dtn/.

[18] CCSDS 734.3-B-1 “Schedule-Aware Bundle Routing”, recommended standard, Blue Book, July 2019, https://public.ccsds.org/Pubs/734x3b1.pdf

[19] M. Ramadas, S. Burleigh and S. Farrell, “Licklider Transmission Protocol – Motivation”, Internet RFC 5326, Sept. 2008.

[20] M. Ramadas, S. Burleigh and S. Farrell, “Licklider Transmission Protocol – Specification”, Internet RFC 5326, Sept. 2008.

[21] S. Burleigh “Bundle Protocol Extended Class Of Service (ECOS)”, IETF Draft, July 2013, https://tools.ietf.org/html/draft-irtf-dtnrg-ecos-05

[22] G.M. De Cola, “Contact Graph Routing Enhancements in ION 3.7.0”, Bachelor’s thesis, University of Bologna, Fabruary 2020 (available on request)

[23] Jin Y. Yen, “Finding the K Shortest Loopless Paths in a Network”, Management Science, Vol. 17, No. 11, pp. 712-716, Theory Series (Jul. 1971).

[24] E. Birrane “Contact Graph Routing Extension Block”, IETF Draft, October 2013, https://tools.ietf.org/html/draft-irtf-dtnrg-cgreb-00

[25] N. Alessi, C. Caini, T. de Cola, S. Martin, J. Pierce Mayer, “DTN Performance Analysis of Multi-Asset Mars Earth Communications”, International Journal of Satellite Commun. and Networking, Aug. 2019; doi: 10.1002/sat.1326; Open access.

[26] P. Apollonio, C. Caini, M. Giusti and D. Lacamera, "Virtualbricks for DTN satellite communications research and education”, in Proc. of PSATS 2014, Genoa, Italy, July 2014, pp. 1-14.

[27] C. Caini, A. d’Amico and M. Rodolfi, “DTNperf_3: a Further Enhanced Tool for Delay-/Disruption- Tolerant Networking Performance Evaluation”, in Proc. of IEEE Globecom 2013, Atlanta, USA, December 2013, pp. 3009 - 3015.