Opportunities and limits of moderate source routing in delay-/disruption-tolerant networking space networks

Opportunities and Limits of Moderate Source Routing in DTN Space Networks

*E. J. Birrane, °C. Caini, °G.M. De Cola, °F. Marchetti, °L. Mazzuca, °L. Persampieri
*Applied Physics Laboratory, Johns Hopkins University, Laurel, MD, US
°DEI/ARCES, University of Bologna, Italy

edward.birrane@jhuapl.edu, carlo.caini@unibo.it, gianmarco.decola@studio.unibo.it, federico.marchetti4@studio.unibo.it, laura.mazzuca@studio.unibo.it, lorenzo.persampieri@studio.unibo.it

***Abstract* — This paper aims to investigate the potential advantages but also the limits of source routing when applied to DTN space networks. To this end, it uses a variant of Contact Graph Routing (CGR) called Moderate Source Routing (MSR), recently proposed by the authors and fully compatible with ION, the DTN suite developed by NASA-JPL. MSR differs from standard CGR as the route to destination is not recalculated from scratch at each node, but possibly reused, if still valid, by next nodes. For this purpose, the route is saved in a dedicated extension block of the forwarded bundle (the data unit of the Bundle Protocol, used in DTN). Performance of MSR vs. CGR is assessed by considering a simple but very challenging space layout. Numerical results, obtained on a GNU/Linux testbed, show that MSR is effective at reducing the chances of loops, in particular when the source has full knowledge of the state of the network, otherwise network instabilities are still possible. In this case, they can be neutralized by means of the combined use of source routing and anti-loop tools, as shown in the paper. A further advantage of MSR is that it is compatible with standard CGR, which would facilitate a gradual or partial deployment.**

Keywords— Delay-/Disruption-Tolerant Networking, Inter-Planetary Networking, CGR, SABR, Bundle Protocol

I. INTRODUCTION

Communication links between nodes in a space network may be challenged by long signal propagation delays, planned disruptions, and frequent, unplanned service outages. Path diversity in these networks is expected to be less than the diversity achieved over terrestrial networks. Spacecraft are often constrained in their transmit power, reliance on directional communications to span long distances, and are not standardized around a common physical and data link layer. For these reasons, space network topologies are both dynamic and sparsely populated as compared to usual terrestrial networks. These challenges prevent the use of terrestrial networking protocols and algorithms such as those maintaining Internet's TCP/IP architecture.

Delay-/Disruption Tolerant Networking (DTN) defines a networking architecture [1], [2] capable of providing network communications in "challenged networks", as in space. The most significant innovation is the addition of the bundle layer between the application and the transport layers of the ISO/OSI model [3]. The aim of the new layer and of the related Bundle Protocol (BP) [4] is to provide an overlay that relies on persistent storage to cope with network interruption. After a few years of research and tests, DTN standardization has recently moved from IRTF (Internet Research Task Force) to IETF (Internet Engineering Task Force), where a new version of the BP (version 7) [5] is about to be finalized, together with new security extensions [6]. For space applications DTN protocols are standardized in parallel by the Consultative Committee for Space Data System (CCSDS) and since 2016 they have been tested on the International Space Station [9], as a first step towards a Solar System Internet.

Concerning routing, it must be emphasized that link intermittency requires a totally different approach, as it prevents the use of Internet solutions based on a continuous and fast exchange of information between nodes, clearly impossible in DTNs. Given the complexity of the problem, routing has always been one of the most important DTN research topics, with abundant literature. Routing proposals can be divided into opportunistic and deterministic solutions, depending on the kind of connectivity they must deal with [10], [11]. In space networks, transmission opportunities between nodes (i.e. contacts) are mainly deterministic, as a function of spacecraft power management, antenna pointing, and orbital dynamics. This determinism can be exploited to develop routing algorithms that accommodate time-variant topologies. One such algorithm is Contact Graph Routing (CGR) developed by NASA-JPL, which inputs a set of scheduled contacts, computes a time-variant route among them, and forwards data to the entry node of that route [12]-[16]. The latest version of CGR has recently been standardized by CCSDS as Schedule-Aware Bundle Routing (SABR) [17].

In all variants of CGR, including SABR, the algorithm is re-run at every node along the path to destination, as a safety measure to cope with the impossibility of having real-time updates of network state. Although justified, this feature increases the likelihood of encountering routing loops - particularly in space networks that experience frequent topological changes- and is also demanding in terms of computational power. A possible alternative is source-routing, which requires that data in a network follow the route as calculated by the data source [18].

This paper aims to investigate the advantages and limits of source-routing when applied to CGR. It focuses on the "Moderate Source Routing" (MSR) implementation of the source-routing principle, recently presented by some of the authors in [19]. Following the approach originally suggested by another co-author in [20], MSR captures the time-variant route calculated by CGR in a special "extension block" within a bundle. At MSR-aware nodes, the route is verified and used in lieu of a CGR route computation whenever possible, which limits route recomputation to the hopefully rare case of verification failure. MSR is at present included as an optional feature in Unibo-CGR, an implementation of CGR algorithm recently developed at the University of Bologna [21], fully compatible with ION (Interplanetary Overlay Network), the

NASA-maintained open-source collection of DTN software [22], [23]. The potential advantages of MSR are investigated in the paper by installing this software in a GNU/Linux-based testbed, designed to be particularly challenging to CGR but also representative of near-term space network deployments. The results achieved are analyzed bundle-by-bundle, to illustrate not only performance, but also the internal mechanisms of both SABR and MSR.

The remainder of this paper is organized as follows: Section II contains an overview of interplanetary networking; Section III summarizes the logical phases of SABR; Section IV describes MSR; Section V introduces the test scenario; Section VI is the core of the paper, with the analysis of numerical results. Conclusions are drawn in Section VII.

II. Interplanetary Networking and DTN

Interplanetary networks differ from Internet in that they are often partitioned because of their sparse connectivity. Frequent partitioning means that there may not exist a concurrent, end-to-end path between a message source and a message destination. The need to transmit messages even to destinations that do not belong to the network partition of the source is paramount in the design of the DTN architecture. The key aspects that differentiate it from Internet architecture are summarized below.

A. Bundle layer

The DTN architecture relies on overlay networking techniques to insulate applications from all the transport protocols that may be used in challenged environments. This overlay layer, the Bundle Layer, comprises Bundle Protocol Agents (BPAs) that accept signaling and application data on behalf of some Application Agent running on a node [24].

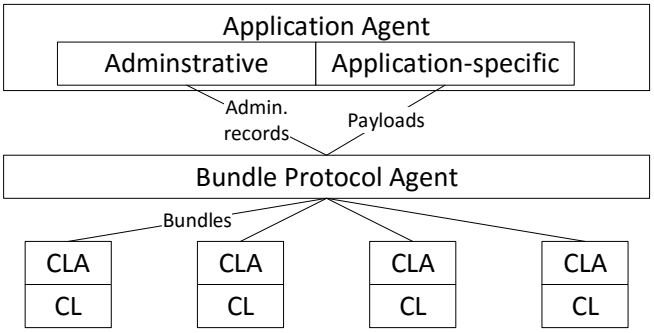

Fig. 1. The DTN architecture integrates multiple transport protocols.

That the BP overlay may use multiple underlying networks for end-to-end transmission can be abstracted from applications. As shown in Fig. 1, application agents interact with a BPA by passing administrative records and data plane information for transmission. The BPA then sends bundles to one or more underlying transport protocols, termed "Convergence Layers" (CL) in this context, through a series of interfaces called "Convergence Layer Adapters" (CLA).

B. Transport layer

In the DTN architecture, the role of transport layer is no longer end-to-end, but instead confined to a DTN hop. This allows selection of different transport protocols on each DTN hop to match its specific challenges. For example, let us consider the end-to-end path from a camera on a lander located on the surface of Mars to a science operations center located on Internet, as illustrated in Fig. 2.

Starting from the right, when an image is taken by the camera on board of a Martian rover, it is encapsulated into a bundle and then sent (when possible) to a Mars Orbiter by means of the LTP [25] convergence layer, on top of CCSDS Encapsulation Packet Protocol [26] and Unified Space Data Link Protocol [27]. The Orbiter keeps the bundle in its memory until the next contact to a ground station on Earth. From there, is forwarded to its destination, a science operation center on Internet via the usual TCP/IP stack. While each transport protocol in this chain is customized to each hop of the path, all nodes implement a BPA and process data messages as bundles.

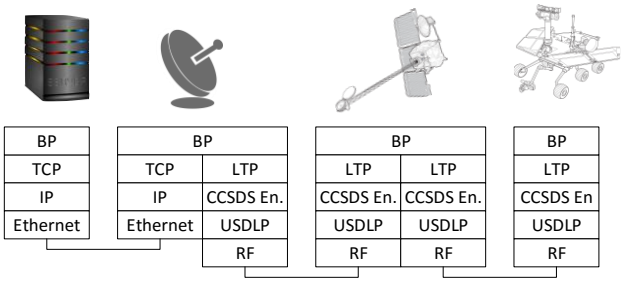

Fig. 2. In the example, the DTN architecture allows the use of diverse, specialized, transport protocols on different hops from Mars to Earth.

C. Bundle storage and retranmission

As previously discussed, the frequent partitioning of an interplanetary network requires persistent storage. While terrestrial Internet traffic can be buffered for milliseconds, DTN traffic requires longer storage (seconds, minutes, hours, or even days) while waiting for a next viable transmission opportunity, i.e. for next contact. Persistent storage also changes the ways in which retransmissions occur. The TCP/IP "end-to-end principle" states that retransmission should come from the data source, as in Internet this solution is simple, robust and fast, thanks to continuous connectivity and short round trip times. However, this does not hold true in challenged networks where long delays and network partitioning make retransmission from intermediate nodes preferable. This may happen at both transport and bundle layers, as we can see by again considering the previous example. First, we can note that the redefinition of transport scope allows transport protocol to recover segment losses occurring in one DTN hop. Second, in special circumstances, e. g. when the custody option is enabled, or in case of a reliable CL failure, a bundle itself can be retransmitted by intermediate nodes [2], [4]. In both cases, costly end-to-end retransmissions between Earth and Mars are generally avoided.

D. Bundle staus reports

Finally, information about the transmission status of a bundle can be communicated to a "report-to" node in the network in the form of administrative records. These records are generated by the application agent's administrative element, as shown in Fig. 1. These administrative bundles, named "status reports", notify the status of each bundle processed (forwarded, received, delivered and deleted) [4]. Once collected in a file by a monitor node, status reports offer researchers a very convenient way to study the path followed by each bundle. We will make extensive use of them in the numerical results section.

III. Contact Graph Routing

A fundamental difference between Internet and DTN routing is that an Internet route consists of a series of

intermediate nodes (the routers), while a DTN route consists of a series of contacts between nodes. In DTN space networks, these contacts can be known a priori, because they are due to the deterministic motion of planets and spacecraft. Scheduled contacts and expected ranges (propagation delays between nodes) are thus inserted into a "contact plan" and disseminated by a Mission Control Centre (MCC) to all nodes. Starting from this, the task of CGR is to find the most suitable route to destination, based on specific metrics, such as the earliest delivery time [12]-[16]. Note that the "CGR route" (sequence of contacts) implies a "geographical route" (sequence of intermediate nodes to be visited in order), but not vice versa, as many different contacts between two nodes are possible. While routing in Internet is similar to the search for the best itinerary for vehicles (roads can usually be assumed to be always available), CGR routing is analogous to planning the best sequence of flights to a remote destination (flights, as contacts, have a source, a destination and are operated only at scheduled times). In this analogy we will often use, DTN nodes are airports, contacts are flights and bundles are passengers [15].

Routing in intermittent networks is per se an arduous problem. In this regard it must be stressed that CGR is best-effort, not optimal, being a necessary compromise between accuracy and computation load. Here it is only possible to provide the reader with a brief overview, referring to the SABR version [17].

A. SABR algorithm

To facilitate comprehension, we will divide the algorithm into three logical phases, following the approach originally presented in [28]. These phases, moreover, correspond to the three core modules of the Unibo-CGR implementation used in tests (Fig. 3).

1) Phase I: route computation

Starting from contacts and ranges declared in the contact plan, the first time a bundle heading for node D appears, "computed" routes, i.e. routes that offer the shortest arrival time for this destination, are calculated. The search on the graph of contacts uses Yen's variant [29] of Dijkstra's algorithm, to facilitate the search for new routes stemming from a previously computed route. ION implementation stops the search of computed routes, after the first (the fastest) is found, while SABR specifications leave the decision on when to stop to the implementation. In fact, Unibo-CGR could insert more routes, but this option is not enabled in tests for consistency with ION behavior. As computation time of Dijkstra's search is significant, this phase is not performed for subsequent bundles destined to D, unless considered necessary by Phase II (see the feedback loop in Fig. 3).

2) Phase II: route validation

Computed routes, calculated in Phase I, are independent of bundle characteristics and of the state of the network, thus their arrival times are optimistic. The effective ability of a computed route to bring the current bundle to destination D in time needs to be validated in Phase II. In particular, the PBAT (Projected Bundle Arrival Time) of the route is calculated, considering both bundle characteristics and local queues, to be sure that it is lower than bundle lifetime. A very important SABR innovation is that the check on the residual volume availability is now extended to all contact of the route (in the analogy, if there are enough seats in all flights). This is done by comparing the current bundle dimension with the local MTV (Maximum Transmission Volume) counters (one for each level of priority) of the route contacts, representing the residual volumes. A long list of other checks is also performed; if all are passed, the computed route becomes a "candidate" route (a viable route for the bundle). Otherwise Phase I is performed again, to find a new computed route to be added to the initial set, and so on.

3) Phase III: bundle forwarding or replication

Unless the bundle is "critical", the best route is selected from among the "candidate" routes (provided that there is a choice) by choosing the one with the shortest PBAT, and the bundle is then forwarded to the neighbor indicated by the first contact of the route. If the bundle is critical, however, SABR performs a replication scheme where one copy of the bundle is sent to all neighbors for which there is at least a viable route.

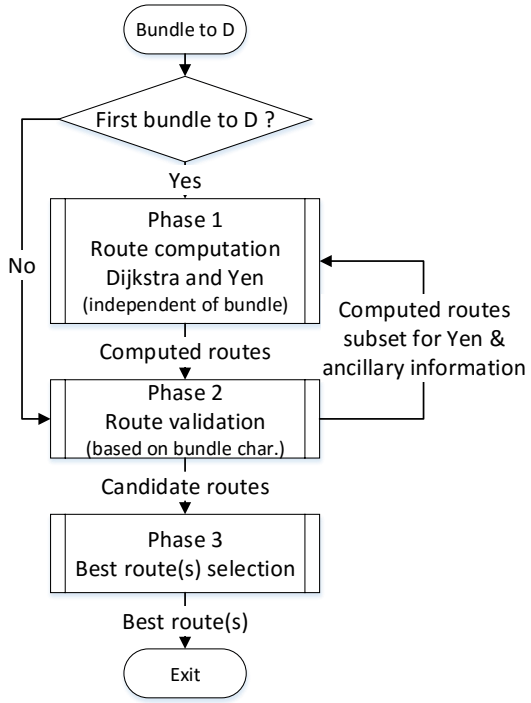

Fig. 3. SABR logical flow chart, as implemented in Unibo-CGR.

B. Rationale and limits of route recomputation

In SABR the best route is used to find which local MTV counters must be decreased and to select the proximate node to which the current bundle must be enqueued. The route is not saved and is then recomputed from scratch at each node (as if after having found the best sequence of flights to a remote destination, only the first was booked). This is a safety measure, dictated by the fact that in a DTN network the state of other nodes, such as traffic generated, contact availability, etc., cannot be exchanged in real time (in contrast with flight bookings in the Internet). Although this concern is justified, recomputing the route at all nodes may be computationally intensive, especially when Phase I must be re-entered, and in fact useless whenever the new route coincides with the residual part of the original one. More importantly, it can increase routing instabilities, because CGR uses not only "common" information available to all nodes, such as the contact plan, but also "local" information, such as actual link availability, local queues (ETO on first hop [14]), estimates on contact residual volumes (MTV counters), etc. On the one hand, local information is essential to validate a route and

select the best; on the other, it varies from node to node by definition, which may lead to inconsistent routing decisions.

IV. Moderate Source Routing

Moderate Source Routing (MSR) aims to conjugate the advantages of the source routing principle with the flexibility and safety of route recomputation performed by CGR. The key is enforcing the CGR route computed by the source at intermediate nodes, but only if this route is still viable, otherwise a new route must be computed from scratch, as in standard CGR.

With respect to the original idea of applying source routing to DTN [12], MSR presents two significant differences. First the extension block used to convey the CGR route is different, second, the route verification in MSR is no longer limited to the control of residual volume availability in the contact of the saved route, but is identical to route validation of SABR, with all checks of Phase II applied.

A. The CGRR extension block

The "CGR-Routes" extension block (CGRR) carries the route selected by SABR in Phase III with the bundle, possibly to be applied by following nodes along the path to destination. CGRR differs from the analogous "CGR extension block" [20] in that it contains only the essential elements, i.e. the contacts of the route and uses another format for route encoding. The CGR route is saved as an ordered series of contacts (or "hops"), identified by the two contact endpoints and the contact start time. Thus, the length of CGRR extension is directly proportional to the number of hops, which may largely vary, but the total length is usually only a few tens of bytes, an absolutely negligible value for data bundles, normally much larger than ordinary IP packets.

A second possible use of CGRR extension, recently discovered, is to reintegrate local MTVs when a bundle is reprocessed by CGR before leaving the local node, either because not transmitted in time (before the forfeit time expiration), or because the intervention of the overbooking management option (enabled by default in ION) [12]. In this case, it is obvious that the bundle will not consume the volume in next contacts of the planned route, thus the corresponding MTV counters need to be reintegrated. This is impossible in standard SABR, because the planned route is not saved, but can be easily accomplished by exploiting the CGRR extension, as done by the latest versions of Unibo-CGR.

B. The MSR algorithm: basic steps

Although the idea of source routing is simple, its implementation is not. Preliminary versions [19] were implemented by modifying the existing ION code, but later MSR was included in Unibo-CGR. Although the code has evolved, the MSR algorithm is almost the same, therefore here we will limit the treatment to a brief summary, referring the interested reader to the cited paper for more details.

In MSR we can distinguish between two phases. In the first phase, three "applicability" checks are performed: a) the bundle must not be flagged as critical; b) the CGRR extension must be enabled; c) the route carried by the extension must contain a contact with the local node as entry point. If all tests are passed, the algorithm continues, otherwise SABR is called.

In the second phase, the saved route must be validated. Two preliminary steps are performed: a) the residual route must be extracted; b) then it must be converted into the SABR format, possible only if all contacts of the residual route are present in the contact plan (this step, route conversion, is actually quite elaborate, see [19]). Eventually, the route is validated by calling the same routine used in SABR Phase II. If validation is successful, the bundle is forwarded to the next node, as planned; otherwise, a new route is computed, by entering SABR, and the old route is replaced in the CGRR extension block.

In accordance with [20], the current MSR version can optionally limit route validation to the first hops. A node is called "wise" if the control encompasses all hops, otherwise, "unwise". In the latter case, the maximum number L of hops to validate is set by the user. Note that unwise nodes need not to know the destination node, while this knowledge is essential to SABR and wise nodes. Thus, unwise nodes could be provided with a contact plan limited to neighbors, which would limit the computational effort required by MSR routing to a minimum. In this way MSR could be used in spacecraft with limited computational power, such as in nano satellites [30].

V. Test Scenario

The test environment used in this paper is similar to that used in [19], but with different contacts. It was explicitly designed to be particularly challenging for CGR, but also to be as simple as possible to facilitate accurate analysis of results. Although this scenario does not pretend to be fully representative of any specific space environment, we have maintained application-oriented node-names, to stress the fact that the situations we are going to examine could happen in a real deployment.

DTN topology

The DTN topology of the test scenario is shown in Fig. 4. It consists of 5 nodes, a Mission Control Centre (MCC), two Ground Stations (GS1 and 2), one Orbiter and one Lander. Terrestrial nodes are connected by continuous links, (represented by continuous lines in the figure), while those between space and terrestrial nodes are scheduled intermittent (dotted and dashed lines). TCP is used at convergence layer on continuous links and LTP on the others.

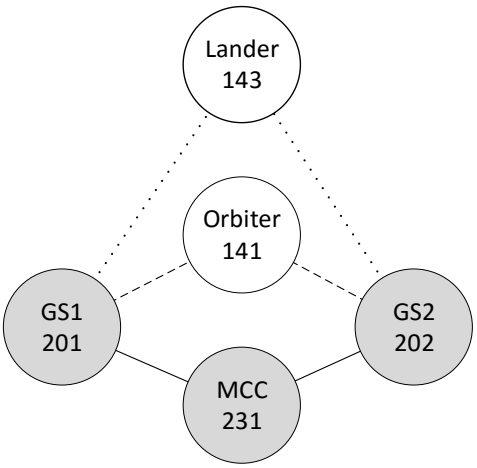

Fig. 4. The DTN Layout used in this paper. Terrestrial nodes, in grey, are connected by continuous lines that denote continuous terrestrial links (TCP). Space links (LTP) are denoted by dashed or dotted lines, depending on their level of connectivity, high or low, respectively.

A propagation delay of 1s is inserted between space and terrestrial nodes and all links are assumed error free, essential in order to have actual transfer rates close to the nominal speeds declared in the contact plan. Note that the symmetry of the layout considered and the presence of very good connectivity between the Orbiter and GSs (almost continuous) are instrumental to increasing the chances of loops (e.g. MCC->GS1->Orbiter->GS2->MCC, or vice versa), especially for uplink traffic, as bandwidth and thus contact volumes tend to reduce the further from earth.

A. Contact plan

The contact plan used in tests is shown in Table I. It differs from that used in [19], in three significant aspects: first, an additional contact between GS2 and the Space Asset has been inserted (#4), to make more complex tests possible; second, the first two contacts between GSs and Lander (#1 and #2) no longer overlap, to improve the reading of plots; last, contacts to/from orbiter (#6 and #7) are no longer continuous, but start at +25s, which helps to simplify analysis. We have also adopted a different contact numbering criterion, to focus attention on the most significant contacts, the first four, now happening in temporal order. Other notes:

- Times are expressed differentially with respect to a reference time (ION startup).
- Space contacts are often asymmetric in space, thus they must be declared as unidirectional contacts, as in Table I, which follows ION syntax. However, for the sake of simplicity, all contacts in Table I have a corresponding symmetrical contact.
- The Intermittent contacts from GS to Lander have the same short length and the same low Tx rate, thus the same small volume.
- In the absence of a specific notation in ION, continuous contacts between MCC and GSs are declared as very long contacts; they have the same high Tx rates and large contact volumes.
- For the sake of simplicity, as they are totally uninfluential regarding loops, all nominal propagation delays ("ranges" in ION) are set to 1s and therefore not included in the table.

TABLE I. CONTACT PLAN (ION FORMAT).

Cont.	***From***	***To***	***Start (s)***	***End (s)***	***Rate (byte/s)***
1	201	143	+30	+42	4000
2	202	143	+48	+60	4000
3	201	143	+70	+82	4000
4	202	143	+88	+100	4000
5-8	The same as 1-4 but in the opposite direction				
6	201	141	+25	+36000	125000
7	202	141	+25	+36000	125000
8-9	The same as 6-7 but in the opposite direction				
10	231	201	+0	+36000	1250000
11	231	202	+0	+36000	1250000
12-13	The same as 10-11 but in the opposite direction				

B. Emulation platforms

We implemented the test scenario on two emulation platforms: CORE (Common Open Research Emulator) [30], based on Linux containers, and Virtualbricks [32], based on Virtual Machines (VM). In both cases we have one GNU/Linux virtual device for each DTN node of the test layout, plus one additional "Monitor" node, used to launch the test and collect status reports. The advantage of CORE is that it allows the user a faster switch between CGR variants, because the BP implementation that contains the CGR code must be recompiled only on the host machine (containers share the operating system and the application of the host). By contrast, with CORE it is almost impossible to use different CGR variants on different nodes (e.g. to test the "unwise" MSR feature), as there is only one ION implementation running, that on the host. To have the best of two worlds, we used both emulation platforms.

Together with Unibo-CGR, we used the latest available release of ION (3.7.2); this allows the user the choice between BPv6 [4] or the still experimental BPv7 [5]; we opted for BPv6, but routing decisions are independent of the BP versions, thus all results presented here would be valid with BPv7. We did not insert any security feature, but is worth mentioning that in a real deployment with possible security concerns, the CGRR extension could be protected from tampering by the insertion of security blocks [6].

In our tests we compared 3 CGR variants, namely SABR, MSR and MSR-AL (MSR plus Anti-Loop options), all easily obtainable by changing Unibo-CGR settings. To generate bundles and collect bundle status reports we used several concurrent instances of the DTNperf_3 tool [33], launched on different nodes by means of a "do test" script file on the Monitor node. Status reports were complemented by Unibo-CGR logs, essential for investigating not only which routing decisions were taken, but why.

VI. NUMERICAL RESULTS

We will start the analysis by considering the ideal case of a source that has perfect knowledge of the network when CGR is called the first time and later examine a more complex situation where this knowledge is only partial. Considerations on possible computational savings offered by MSR will close the section.

In all tests, the focus is on the uplink flow generated by the MCC and destined to Lander, consisting of 5 bundles of 50 kB each, generated at 1s interval, with lifetime 120s, i.e. longer than the expected duration of the experiment.

A. Source with full knowledge of network state

In this first test, there is only one source, the MCC, the network is unloaded, and all contacts happen as expected. The 5 bundles are generated immediately after completion of ION startup and are processed by CGR one-by-one as soon as generated. They are represented by the "Generated 231" time series in Fig. 5, where the four contacts to Lander are also plotted. From test layout (Fig. 4) and contacts (Table I) it is evident that there are 4 obvious routes to get to Lander from MCC:

- Route 1, via GS1 consisting of contacts 10 (continuous) and 1 (starting at +30s).
- Route 2, via GS2, - contacts 11 (continuous) and 2 (starting at +48s).
- Route 3, the second via GS1, - contacts 13 (continuous) and 3 (starting at +70s).
- Route 4, the second via GS2, - contacts 11 (continuous) and 4 (starting at +70s).

The four routes are in order of convenience, i.e. the first is the fastest. This, however, holds true only at the beginning of the experiment, when contact volumes are intact. In fact, we must recall that the volume of the four contacts to Lander can only contain two full bundles, considering bundle overheads. With this essential constraint in mind, we can presume that the first two bundles will follow Route1, the next two Route 2 and the last Route 3. We will see however that this prediction is not always confirmed. From now on, it is convenient to proceed by distinguishing between SABR and MSR (MSR-AL is not considered in this first test, as results would coincide with MSR).

1) SABR (general analysis, from status reports)

Examining SABR results in Fig. 5, we can observe that although all bundles are delivered in the first two contacts and a half, as expected, the last three bundles are delivered out of order. Moreover, bundle 3 performs a loop, which indicates severe routing problems. From GS1 and GS2 "received" status report ("GS rcv" in figures), we can see that decisions of the source meet expectations (bundle 1 and 2 are sent to GS1, 2 and 3 to GS2, 5 again to GS1, in accordance with Routes 1, 2 and 3). The problem is that the expected routes are confirmed on intermediate nodes only for bundles 1 and 2, the only ones directly delivered as expected, i.e. on contact 1. Bundles 4 and 5 are redirected to the alternate station, via the orbiter, instead of being directly delivered on contact 2; finally, bundle 3 performs a loop, as said. This example clearly shows how SABR recomputation at intermediate nodes may lead to routing instabilities. A much more detailed analysis, for interested readers, is given in the Appendix. The same applies to all other tests considered in this section.

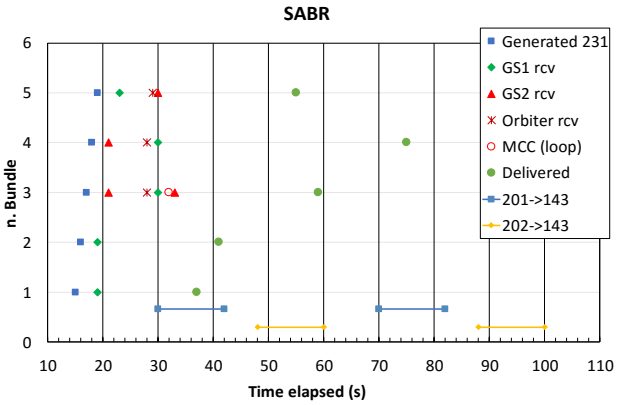

Fig. 5. Full knowledge of network state; SABR, one flow from MCC to Lander: time sequences of bundle generation, forwarding and delivery. Note the loop performed by bundle 3.

2) MSR (general analysis, from status reports)

By repeating the same experiment with MSR (Fig. 6) all bundles are now delivered in order, and what is more important, without any loops. As a general remark, we can state that MSR is effective at preventing the instabilities that may derive from route recomputation at each node, at least in this ideal case.

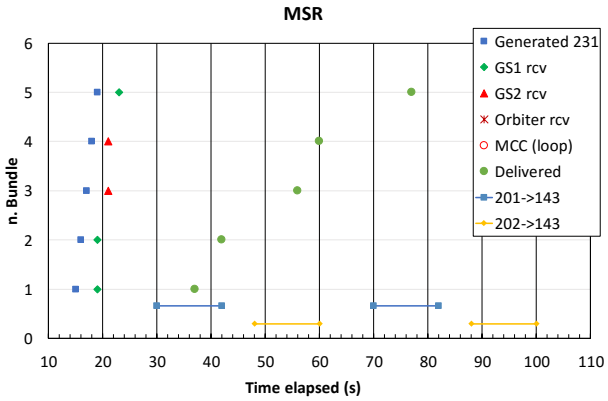

Fig. 6. Full knowledge of network state; MSR, one flow from MCC to Lander: time sequences of bundle generation, forwarding and delivery. All bundles delivered in order, without loops, as expected.

B. *Source with partial knowledge of network state*

This second test differs from the previous one because of the presence of concurrent traffic. Now two bundles, still destined to Lander, are generated by GS1 (Generated 201 time series, Fig. 7) immediately before the MCC five, now renumbered as 3-7 (Generated 231).

1) SABR (general analysis, from status reports)

The introduction of concurrent traffic in other nodes, as here, means that the source can have only partial knowledge of the state of the network, resulting in inaccurate predictions. As MCC is unaware of the two bundles introduced in GS1, on MCC all locally generated bundles are processed as before. The first two (now 3 and 4) are consequently sent to GS1 (Route 1), the other two (now 5 and 6) to GS2 (Route 2), and the last (now 7) to GS1 again (Route 3), as shown by GS1 and GS2 rcv markers, at around + 22s). Looking at Fig. 7 again, however, we can see that only one bundle (ironically the last, 7) is enqueued to Lander for direct delivery (on contact 3), as planned by MCC. For the other 4 bundles (3-6) the recalculated route brings them to the opposite GS. Here, 3, 4 and 5 are enqueued to Lander and later delivered on contact 2 and 3 respectively (see 3, 4 and 5 Delivered markers on Fig. 7). Unfortunately for bundle 6 the outcome is the worst: after looping twice, it is blocked on GS1 and never delivered.

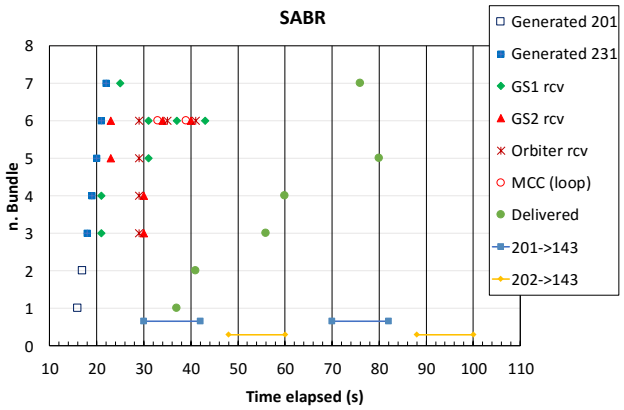

Fig. 7. Partial knowledge; SABR, one flow from GS1 (201) and one from MCC (231) to Lander: time sequences of bundle generation, forwarding and delivery. Bundle 6 loops twice and is never delivered.

2) MSR (general analysis, from status reports)

The introduction of concurrent traffic from other nodes is also challenging to MSR, because this time decisions taken by

the source, which are the same as those of the ideal case, are no longer optimal, as shown by the looping of bundles 3 and 4 (Fig. 8). In brief, by jeopardizing the routes planned by the source, concurrent traffic has exposed the inherent limits of MSR. However, in spite of the loops all bundles are delivered, which is an important point in favor of MSR.

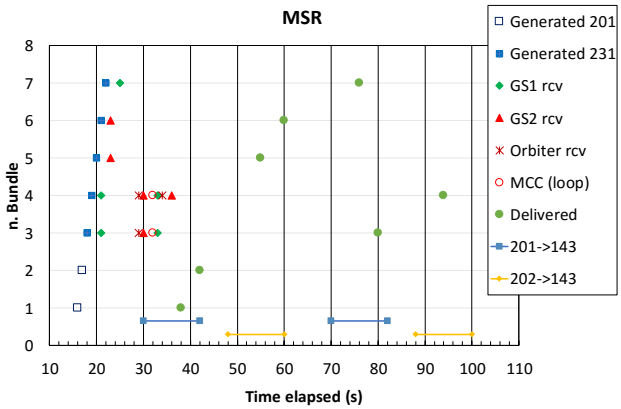

Fig. 8. Partial knowledge; MSR, one flow from GS1 (201) and one from MCC (231) to Lander: time sequences of bundle generation, forwarding and delivery. Bundles 3 and 4 loops once before being delivered.

3) MSR-AL (general analysis form staus reports)

Unibo-CGR has an experimental feature to counteract loops, which comes in two variants, reactive and proactive [21]. They are both based on a second experimental bundle extension, called RGR (Record Geographical Route), which records the nodes already visited by the bundle. By inspecting it, the proactive variant controls whether a candidate route contains a visited node. If so, the route is marked as "closing-loop", because it would result in a loop. In Phase III, closing-loop routes are only selected in the absence of any alternative. This way, it is possible to avoid loops by influencing the decision process of CGR. Conversely, the reactive variant intervenes only a posteriori, after a loop has happened. This variant too is based on the RGR extension: if a loop is detected (the current node appears in the RGR list), the node visited after the current one should be avoided not to repeat the same geographical loop. To this end, the candidate route starting with this node are selected in Phase III only as a last resort, as before. The difference is that the reactive variant cannot avoid the first loop and thus is generally less performant.

To see if it is possible to eliminate the loops in the MSR test, we repeated the current test with both MSR and AL features enabled. From Fig. 9 we can see that this time all bundles are delivered without entering in any loop, thus proving the effectiveness of MSR when associated to the anti-loop features.

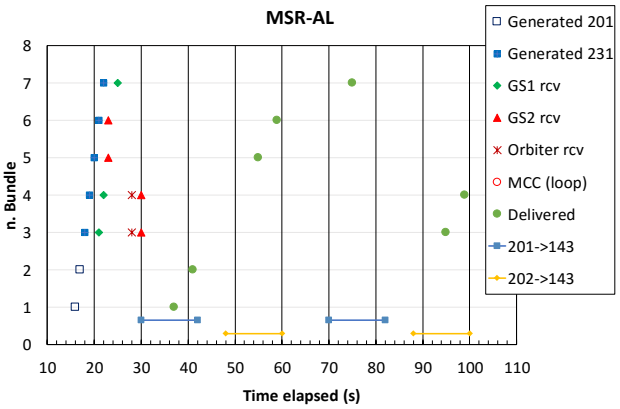

Fig. 9. Partial situation awareness; MSR-AL, one flow from GS1 (201) and one from MCC (231): time sequences of bundle generation, forwarding and delivery.

C. Computation time analysis

Source routing presumably leads to a significant reduction in the processing time. To assess this theoretical advantage in practical terms, we have "instrumented" Unibo-CGR to produce time-logs and then repeated all our experiments. Overall corresponding computation times are presented in Fig. 10. The relative values (absolute values are of little significance, as they depend on the hardware) confirm the presumed advantage of MSR, with a reduction of about 60% in both cases. This reduction depends on two factors. First, by reducing instabilities, MSR also reduces the times a bundle is processed by CGR. The actual number of total CGR calls can be obtained approximately by summing up the number of markers in previous figures, except those of delivered series. We have 13 calls for SABR versus only 5 for MSR in the full knowledge case, (Fig. 5 and Fig. 6); 21 for SABR, 15 for MSR and 9 for MSR-AL in the partial knowledge one (Fig. 7, Fig. 8 and Fig. 9). The second factor is that every time the saved route is validated, instead of recomputed, we have an additional gain, contributing to the overall reduction of Fig. 10. The amount of this second gain, however, is particularly significant only in cases when SABR would have performed Phase I, as Dijkstra's computation time dominates validation times.

The results presented here are preliminary, and do not pretend to be general. In this regard, we think that a general quantitative evaluation of MSR's impact on real deployments is actually almost impossible, because of the heavy dependence on the specific characteristics of the scenarios considered, in terms of layout, contact plan and traffic flows. With this caveat in mind, we can however observe that as SABR and MSR perform the same on source nodes, in more complex layouts, with longer paths, the MSR advantage might be greater than that found in this paper, where the layout is challenging in terms of stability, but the best path from source to destination consists of only two hops.

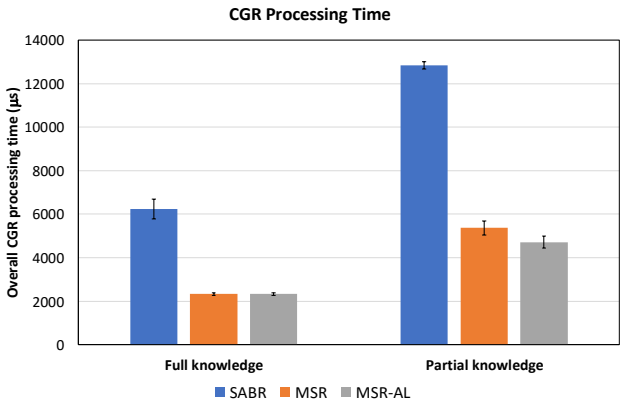

Fig. 10. Comparison of CGR variants computation times. The values are comprehensive of all CGR calls performed on the different nodes, during the full length of the experiment. The values presented are the mean on four launches, to average off oscillation due to operating system scheduler. Error bars are also plotted.

VII. Conclusions

In this paper the possible advantages, but also the limits, of Source Routing in space DTN networks are investigated, using the MSR variant of CGR, recently implemented by some of the authors. Tests were carried out on an emulated environment, with GNU/Linux machines running the latest version of ION with Unibo-CGR added, and this in turn includes MSR. The results show that MSR can offer ideal performance in terms of ordered and fast delivery, when the source has perfect knowledge of the network. In particular, MSR can prevent the loops that affect SABR in the challenging test scenario considered in the paper. However, if knowledge of the source is partial, MSR can no longer prevent loops, although it continues to offer better performance than original SABR. Loops are eventually eliminated by the joint use of MSR and the anti-loop mechanism offered by Unibo-CGR. Preventing routing instabilities, however, it is not the sole possible advantage of source routing, it can also lead to a significant reduction in the processing time, as quantified in the paper. As a final remark, however, we must stress that given the complexity of the routing problem in intermittent scheduled networks, source routing cannot be considered optimal per se, but only a promising alternative/complement to SABR. MSR implementation of source routing is compatible with standard SABR in ION, which would facilitate gradual or partial deployment.

Appendix

The inspection of status reports, carried out in the Numerical Results section, describes what happens, but cannot explain why. In other words, we can see routing decisions, but not the internal process that leads to these decisions. The comprehensive analysis that follows, based on the very informative Unibo-CGR logs, fills this gap. However, this analysis is only destined to those readers who are really interested to know all SABR details, being necessarily quite long and intricate. For the reader's convenience, we will adopt the same structure already used in the Numerical Results section.

A. Source with full knowledge of network state

1) SABR (detailed analysis from logs)

When the first bundle is processed on MCC, CGR performs route computation to Lander using Dijkstra's algorithm (Phase I), in which neither bundle characteristics nor available contact volumes are considered. It finds that Route 1 is the fastest and inserts it in the set of "computed" routes, until now empty. Then, Route 1 is validated in Phase II, where a long list of checks is performed, the most important of which is the control of MTV counters, to be sure that there is enough volume for the current bundle in all contacts of the route. Route 1 passes all checks, so becoming a "candidate" route and is then obviously selected, being the sole one, in Phase III. The bundle is put in the queue to GS1 where it is immediately sent and received (see "GS1 rcv" time series in Fig. 5).

The same holds true for bundle 2, with the notable exception that Phase I is now skipped, as Route 1 is already among the computed routes. When bundle 3 is processed, Phase I is temporarily skipped, but this time Route 1 is barred in Phase II, because its MTV counters indicates that there is not enough residual volume on contact 1 to accommodate a third bundle. After this failure, Phase I is re-entered and Route 2 is added to computed routes. Route 2 is then found valid in Phase II, selected (it is actually the sole one) in Phase III, and the bundle is accordingly sent to GS2 ("GS2 rcv" time series in Fig. 5). As before for bundle 4, except that Phase I is skipped. At this point, on MCC the MTVs of contacts 1 and 2 are almost depleted and Phase I must be re-entered for bundle 5. Route 3 is thus added to computed routes, validated and selected, and bundle 5 is immediately sent to GS1, as were bundles 1 and 2. To make a long story short, everything happens on the source as expected, but this detailed explanation was necessary to understand why now, on GS1 and GS2 things are not going to happen as predicted by the source.

As soon as the first bundle arrives at GS1, CGR is called, and the route recomputed from scratch, as prescribed by SABR. The obvious decision is in favor of the direct route consisting of the sole contact 1, therefore the bundle is enqueued to Lander to be successfully delivered after contact 1 starts (marker 1, Delivered series). The same holds true for bundle 2, but not for bundle 5, because contact 1 is now full. Route 2, selected by the source for bundle 5, indicates the use of contact 3 (i.e. to stay here, waiting for contact 3 to start), but GS1 has a different opinion. After failure of Route 1, the Alternate Route 2 via Orbiter and GS2 (contacts 6, 9, 2) is preferred by Dijkstra to contact 3, because it offers a shorter arrival time. The Alternate Route 2 is then validated in Phase II, selected in Phase III and the bundle is enqueued to Orbiter, waiting for the opening of contact 6 (at +25s). The origin of different routing decisions, with respect of what was planned by the source, lies in the fact that GS1 is unaware of not locally processed traffic, thus its MTV counters for contact 2 are not depleted by bundles 3 and 4, as those of the source were. This is why Alternate Route 2 is not barred in Phase II.

After this key observation, we can move on to GS2, where the situation is almost specular to bundle 5 processing. At the arrival of bundle 3 Dijkstra prefers the local Alternate Route 1 (7, 8, 1) to contact 2 because it is seen as faster, thus is validated and eventually selected, and the same happens for bundle 4. When contacts to Orbiter open (6 and 7), bundles 3 and 4 from GS2 to GS1, and bundle 5, in the opposite direction, cross on board the orbiter at about the same time, ("Orbiter rcv" markers, at around +28s), which clearly proves that the decisions taken on GS1 and GS2 are far from optimal. Here, on Orbiter, CGR must confirm the direction of

movement for all bundles, as the SABR anti ping-pong mechanism prevents CGR from sending back any bundle.

Let us continue from bundle 5, re-processed by CGR on arrival at GS2. This time, the MTV for contact 1 appears depleted (by bundles 3 and 4, just processed), which forces CGR to re-enter Phase I, add the direct route consisting of contact 2, validate and select it. Bundle 5 is therefore enqueued to Lander and eventually delivered after contact 2 starts (marker 3 of the Delivered time series), in third position.

In the meantime, bundles 3 and 4 have arrived at GS1. When bundle 4 is processed, the local MTVs of contact 1 are full (depleted by bundles 1 and 2), while there is still room for one bundle on contact 2, departing, however, from the other GS. Therefore, the selected route for bundle 3 is Looping Route 2 (contacts 12, 11, 2). Markers on the figure show that bundle 3 passes first through the MCC, thus completing the loop, and then it reaches GS2 again, to be eventually delivered after bundle 5, on contact 2 (marker 3, Delivered time series). The decision taken by GS1 for bundle 3 depletes the MTV of contact 2, thus for bundle 4 the Looping Route 2 is no longer viable and the decision is to wait for contact 3, during which the bundle is actually delivered (marker 4, Delivered time series).

2) MSR (Detailed analysis, from logs)

The MSR story is, luckily, much shorter than the previous one. Source decisions are the same as before, but this time the original route is saved in the CGRR extension. Next nodes check the saved route by directly calling the SABR Phase II routine. All checks are passed and thus all bundles follow the original route planned by the source (Fig. 6).

B. Source with partial knowledge of network state

1) SABR Detailed analysis, from logs

The bundles generated by GS1 are immediately enqueued to Lander associated to contact 1, making it unavailable to other bundles. As MCC is unaware of that, all its bundles are processed as before and then forwarded either to GS1 (bundles 3, 4, 7), or GS2 (bundles 5 and 6), see Fig. 7. By contrast to the full knowledge case, the first two bundles (4 and 5) are now redirected to GS2, as local MTV counters for contact 1 are depleted (by bundles 1 and 2). Bundles sent to GS2 (now 5 and 6) are redirected to GS1 as in the previous SABR test, as GS2 MTV for contact 1 is intact. The last bundle (now 7), when received by GS1, finds local MTVs of both contact 1 and 2 depleted (by bundles 1, 2, 3 and 4) and directly routed to Lander on contact 3, as planned by the source. After the contacts to Orbiter start, four bundles cross in opposite directions (Orbiter rcv markers, around 30s), a clear symptom of routing problems. When bundles 3 and 4 arrive at GS2 (about +30s), they are directly routed to contact 2, as the local MTV counter of contact 1 is now depleted (by bundles 5 and 6, which arrived at about +22s). At this point the only bundles not yet enqueued are 5 and 6. When they arrive at GS1 from the Orbiter, local MTVs of contacts 1 and 2 are depleted (by bundles 1, 2, 3 and 4) and there is only room for one bundle on contact 3 (partially used by bundle 7). Bundle 5 takes the residual volume first and it is enqueued to Lander and delivered (marker 5, Delivered time series). From now on (from about +31s), we can focus on bundle 6, the only one not yet enqueued to Lander. It is waiting to be routed by GS1, where local MTVs of contacts 1, 2, and 3 are depleted, while the MTV of contact 4 is intact. This is correct, as contacts 1-3 are already been fully allocated to the other 6 bundles. In the absence of any alternative, the route chosen for bundle 6 is the counterclockwise Looping Route 4 (contacts 12, 11, 4), which brings it to GS2, for the second time, after two hops (marker 6 of GS2 rcv time series, at about 32s). What appears to GS2, however, is wrong: although local MTVs of contact 1 and 2 are depleted, the contact 3 appears empty but is not. Therefore, Alternate Route 3 (contacts 7, 8, 3) is preferred to contact 4. The counterclockwise loop thus continues; when bundle 6 arrives at GS1 for the second time, local MTVs of contacts 1-3 are all depleted as before, the only difference is that contact 4 MTV is no longer intact (because of previous redirection of bundle 6 itself). The story repeats itself, as Looping Route 4 is chosen again, and bundle 6 reaches GS2 for the third time and is sent to GS1 on Alternate Route 3, as in the previous loop. When bundle 6 arrives at GS1 for the third time, all four MTVs are now depleted (ironically, that of contact 4 as a result of bundle 6's two loops, but GS2 is unaware of that). Consequently, bundle 6 is put in "limbo", re-processed by CGR every 4 s, and eventually deleted at lifetime expiration.

2) MSR Detailed analysis, from logs

When the first two bundles generated by MCC (now 3 and 4) arrive at GS1, Route 1 is extracted from the CGRR extension and the remaining part, consisting only of contact 1 is checked, by calling the Phase II routine. The check however fails, as contact 1 is not available, thus CGR is called to find a new route from scratch, as with SABR. Bundles 3 and 4 are thus sent to Lander via GS2, after selection of Alternate Route 2 (contacts 6, 9 and 2), see Fig. 8.. This route is saved in the CGRR extension, then validated by the Orbiter but not by GS2, because bundles 5 and 6 have arrived in the meantime ("GS2 rcv" time series) and been routed to contact 2, as planned by MCC. After this second failure, the route is recomputed for both bundles 3 and 4 and the clockwise Looping Route 3 (13, 10, 3) is the only one possible. It is validated on all nodes, for both bundles, until they arrive at GS2. Here it is confirmed for bundle 3 but not for bundle 4, as on local MTV for contact 3 there is space for one bundle only (bundle 7 had arrived in the meantime and forwarded on it). Consequently, the route is recomputed for the third time for bundle 4 and Alternate Route 4 (contacts 6, 9 and 4) is selected. It is confirmed everywhere, and bundle 4 too can be eventually be delivered, last of all, on contact 4.

3) MSR-AL Detailed analysis, from logs

Let us focus on bundles 3 and 4, the only ones for which the original route is recalculated (see Fig. 9). In GS1 ("GS1 rcv" markers 3 and 4) the candidate route has no loops, and the same decisions are taken as before. Vice versa, in GS2 ("GS2 rcv" markers 3 and 4) this time Looping Route 3 is discarded, in favor of contact 4, although the latter is less appealing in terms of projected bundle arrival time.

ACKNOWLEDGMENT

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

REFERENCES

[1] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets", in Proc. of Conf. on Applications, Technologies, Architectures, and Protocols For Computer Commun. SIGCOMM '03, Karlsruhe, pp. 27-34, 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] "ISO/IEC 7498-4:1989 -- Information technology -- Open Systems Interconnection Basic Reference Model: Naming and addressing",

standards.iso.org/ittf/PubliclyAvailableStandards/s014258_ISO_IEC_7498-4_1989(E).zip

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

[5] S. Burleigh, K. Fall, E. Birrane, “Bundle Protocol Version 7”, IETF Draft, Dec. 2020, https://tools.ietf.org/html/draft-ietf-dtn-bpbis-30, work in progress.

[6] E. Birrane, K. McKeever “Bundle Protocol Security Specification”, IETF Draft, Jan. 2021, https://tools.ietf.org/html/draft-ietf-dtn-bpsec-26, work in progress.

[7] CCSDS 734.0-G-3 “Rationale, Scenarios, and Requirements for DTN in Space.” CCSDS Green Book, Issue 3, Jul. 2014. https://public.ccsds.org/Pubs/734x0g1e1.pdf

[8] CCSDS 734.2-B-1 “CCSDS Bundle Protocol Specification”, CCSDS Blue Book, Issue 1, Sept. 2015. https://public.ccsds.org/Pubs/734x1b1.pdf

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

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

[11] 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, Nov. 2011. doi: 10.1109/JPROC.2011.2158378

[12] 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, Mar. 2015. doi: 10.1109/MCOM.2015.7060480

[13] 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

[14] N. Bezirgiannidis C. Caini,V. Tsaoussidis, “Analysis of contact graph routing enhancements for DTN in space”, Int. J. Satell. Commun. Network., 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] M. Marchese and F. Patrone, "E-CGR: Energy-Aware Contact Graph Routing Over Nanosatellite Networks," IEEE Trans. on Green Commun. and Network., vol. 4, no. 3, pp. 890-902, Sept. 2020, doi: 10.1109/TGCN.2020.2978296.

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

[18] S. Previdi, C. Filsfils, B. Decraene, S. Litkowski, M. Horneffer and R. Shakir, “Source Packet Routing in Networking (SPRING)”, Internet RFC 7855, May 2016, https://tools.ietf.org/html/rfc7855

[19] C. Caini, G. M. De Cola, F. Marchetti, L. Mazzuca, “Moderate Source Routing for DTN Space Networks”, in Proc. of ASMS 2020, Virtual, Oct.. 2020, pp. 1-7. doi: 10.1109/ASMS/SPSC48805.2020.9268926

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

[21] C. Caini, G. M. De Cola, L. Persampieri, “Schedule-Aware Bundle Routing: Analysis and Enhancements”, Int J Satell Commun Network, 2020. doi: 10.1002/sat.1384

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

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

[24] Birrane, Ed, and Jason Soloff. Designing Delay-Tolerant Applications for Store-and-Forward Networks. Artech House, 2020.

[25] M. Ramadas, S. Burleigh and S. Farrell, "Licklider Transmission Protocol – Specification," Internet RFC 5326, Sept. 2008, http:/www.rfc-editor.org/rfc/rfc5326.txt

[26] CCSDS 133.1-B-3, “Encapsulation Space Protcol” CCSDS Blue Book, Issue 3, May 2020. https://public.ccsds.org/Pubs/133x1b3e1.pdf

[27] CCSDS 732.1-R-3, “Unified Space Data Link Protcol” CCSDS Red Book, Issue 3, Sept. 2017. https://public.ccsds.org/Lists/CCSDS%207321 R3/732x0r3.pdf

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

[29] 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.

[30] M. Marchese and F. Patrone, "A Source Routing Algorithm Based on CGR for DTN-Nanosatellite Networks," GLOBECOM 2017 - 2017 IEEE Global Commun. Conf., Singapore, 2017, pp. 1-6, doi: 10.1109/GLOCOM.2017.8255092.

[31] CORE (Common Open Research Emulator) code and documentation: https://github.com/coreemu/core

[32] 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. doi: 10.1007/978-3-319-47081-8_7

[33] 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, Dec. 2013, pp. 3009 - 3015. doi: 10.1109/GLOCOM.2013.6831533