Received: 30 March 2016 | Revised: 12 June 2016 | Accepted: 8 March 2017

DOI: 10.1002/sat.1210

WILEY

RESEARCH ARTICLE

Congestion management techniques for disruption-tolerant satellite networks

Pablo G. Madoery | **Juan A. Fraire** | **Jorge M. Finochietto**

Digital Communications Research Lab - IDIT, Universidad Nacional de Córdoba - CONICET (FCEFyN), Argentina

Correspondence
Juan Fraire, Digital Communications Research Lab - IDIT, Universidad Nacional de Córdoba - CONICET (FCEFyN) Argentina.
Email: juanfraire@gmail.com

Summary

Delay and disruption-tolerant networks are becoming an appealing solution for extending Internet boundaries toward challenged environments where end-to-end connectivity cannot be guaranteed. In particular, satellite networks can take advantage of a priori trajectory estimations of nodes to make efficient routing decisions. Despite this knowledge is already used in routing schemes such as contact graph routing, it might derive in congestion problems because of capacity overbooking of forthcoming connections (contacts). In this work, we initially extend contact graph routing to provide enhanced congestion mitigation capabilities by taking advantage of the local traffic information available at each node. However, since satellite networks data generation is generally managed by a mission operation center, a global view of the traffic can also be exploited to further improve the latter scheme. As a result, we present a novel strategy to avoid congestion in predictable delay- and disruption-tolerant network systems by means of individual contact plans. Finally, we evaluate and compare the performance improvement of these mechanisms in a typical low Earth orbit satellite constellation.

KEYWORDS

contact graph routing, congestion management, delay and disruption tolerant networks, satellite networks

1 | INTRODUCTION

Traditionally, Earth observation satellites have been designed to periodically gather data from large ground areas. However, the increasing need of timely and on-demand data (images, videos, etc) is demanding a paradigm shift toward better acquisition rates and improved data delivery. To this end, recent research has shown that low Earth orbit (LEO) satellite networks can meet these requirements by significantly enhancing both coverage and revisit time among other benefits.[1] In particular, this spatial diversity not only allows for unprecedented applications by combining sensors for wider aperture, better footprint, or sensing diversity but also presents additional opportunities for data downlink to ground stations.[2]

Indeed, by relying on intersatellite links (ISLs) among the orbiting assets, traffic can flow through multiple hops toward its destination on Earth improving both system capacity and data delivery time. However, maintaining a persistent end-to-end connection between the origin of the data and its destination in orbiting constellations demands strict flight-formation requirements[3] and might require prohibitive amounts of communication resources.[4] As a result, embracing delay- and disruption-tolerant networks (DTNs)[5] as the underlying communications architecture has recently been recognized as an alternative solution for building future satellite applications.[6]

Originally studied to develop a network architecture for the interplanetary Internet,[7] DTN has been specified as a communication architecture for environments where communications can be challenged by either latency, bandwidth, errors, or stability issues.[8] In particular, to overcome disruption, DTN nodes implement a temporary storage where data is kept until forthcoming communication opportunities (ie, *contacts*) become available. As a result, DTN traffic travels in a store carry-and-forward fashion toward its final destination. However, the expected data flow can be disturbed and deteriorated by storage or link exhaustion in intermediate nodes, generating a congestion problem. In general, the *congestion problem* has been defined as the attempt to send more data than a given contact or node buffer allows for.[9] Therefore, congestion is provoked by a combination of topology constraints and excessive network traffic, which needs to be solved to avoid unnecessary packet drops or retransmissions.

In contrast to traditional Internet-based networks, DTN cannot rely on broadcasts or stable end-to-end feedback to implement congestion control due to the sporadic nature of contacts. An in-depth survey with the state-of-the-art on DTN congestion control is provided in Fall.[10] Among the contributions of this survey, a taxonomy for classifying congestion control mechanisms is proposed. On the one hand, *reactive* mechanisms such as custody transfer procedures[11] have been proposed for implicit congestion control by monitoring of buffer occupancy.[12] Most recently, the use of header inspection and reactive feedback messages has also been analyzed to mitigate congestion,[13] but its performance is degraded in highly disrupted scenarios.[9] On the other hand, *proactive* mechanisms based on network capacity of scheduled contacts have been proposed and deployed.[14] However, these mechanisms only solve DTN congestion partially as they only consider local capacities. Instead, these mechanisms can be extended with capacities beyond those available on local contacts, exploiting the predictable nature of their time-evolving contact topology.[9]

Since routing and forwarding schemes are in charge of dispatching DTN traffic through a given network path, they become a key point to proactively avoid or mitigate congestion. Among them, we highlight contact graph routing (CGR)[15] that was designed to take advantage of the a priori knowledge of a *contact plan* comprising the forthcoming communication opportunities to compute efficient routes to the destination. In particular, a contact plan can be derived from node trajectory, orientation, and contacts duration and capacity that can be accurately predicted beforehand in satellite systems.[16] Indeed, this plan is provisioned to the DTN nodes in advance that can later execute the routing algorithm to obtain candidate neighbors to forward locally generated or in-transit data. Contact graph routing has been flight validated[17] and received increasing attention and enhancements from the research community.[18–21]

Contact graph routing proactively mitigates congestion as it avoids forwarding data to next-hop neighbors through locally congested links. This is achieved by keeping status of the *residual capacity* of the future contacts of the local node. Supposedly, there is no benefit in tracking the rest of the hops in the route path as the forwarding decision is not necessarily deterministic in the following nodes. Despite this congestion mitigation mechanism resulted a reasonable approach for initial CGR evaluations, it does not consider the complete path capacity nor storage limitation of intermediate nodes, leading to unwanted traffic-bouncing effects in several scenarios as reported in Fraire et al.[22,23] Moreover, the latter effect becomes more severe when traffic scheduled by remote nodes is also expected to flow through these congested links or buffers.

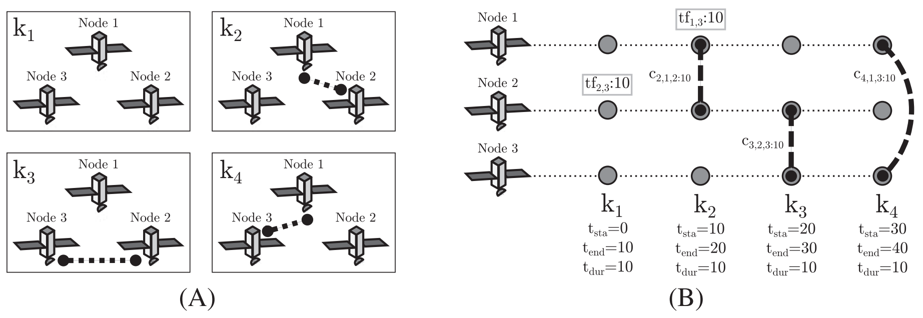
The contributions of this paper are twofold. First, we introduce local path-aware CGR (LPA-CGR) that enhances current CGR congestion mitigation capabilities by combining local traffic information in each node with the complete path capacity. This can be attained with a negligible increase in complexity with respect of the original CGR algorithm. Second, we consider the fact that satellite networks for Earth observation are generally managed by a mission operations and control center that determines a schedule for data acquisitions and reserves system resources in advance.[24] In these specific scenarios, contributions from different traffic sources on a given path can be estimated; thus, link congestion can be proactively predicted. For this type of networks, we describe global path-aware CGR (GPA-CGR) that integrates a global view of the expected traffic to generate congestion-free contact plans. Furthermore, we explore specific optimization techniques based on evolutionary algorithms to fine-tune the results generated by GPA-CGR. Finally, we evaluate and compare these strategies in a typical low Earth orbit satellite constellation.

The paper is presented as an extended and archival quality version of Fraire et al[9] and is organized as follows. Section 2 describes the DTN model as a time-evolving topology by using a finite state machine. Sections 3 and 4 provides a thorough description of LPA-CGR and GPA-CGR, respectively. Next, we evaluate and analyze the performance of the proposed strategies in Section 5 to finally draw the final conclusions and future work in Section 6.

2 | DISRUPTION-TOLERANT NETWORK MODEL

2.1 | Finite state machine modeling

In DTN, a *contact* stands for a forthcoming transmission opportunity and is defined by at least a start and end time and a source and destination node pair. To determine these parameters, communications subsystem attributes such as transmission power, modulation, bit error rate, among others can be combined with orbital dynamics[16] such as position, range, and attitude (orientation of the spacecraft and antenna in the inertial system) of each node. As a result, the set of all feasible and implementable contacts for all nodes in the system within a given *time interval* conforms a time-evolving topology as shown in Figure 1A that can be later imprinted in a *contact plan*.



k_1
k_2
k_3
k_4
Node 1
Node 2
Node 3
$tf_{1,3}$:10
$tf_{2,3}$:10
$c_{2,1,2:10}$
$c_{4,1,3:10}$
$c_{3,2,3:10}$
t_{sta}=0 t_{end}=10 t_{dur}=10
t_{sta}=10 t_{end}=20 t_{dur}=10
t_{sta}=20 t_{end}=30 t_{dur}=10
t_{sta}=30 t_{end}=40 t_{dur}=10
(A)
(B)

FIGURE 1 Topology model. A, Time-evolving topology. B, Finite state machine modeling

To model DTN traffic flows and contact plans, a finite state machine (FSM) formulation has been proposed[4] and illustrated in Figure 1B. In FSM, the time-evolving nature of the contact topology is captured by means of graphs, whose vertices and edges symbolize DTN nodes and their respective contacts. In particular, the topology is discretized by a set of K time intervals $[t_{sta}, t_{end}]$ where each k_i state has a graph representing the communication opportunities within its interval duration t_{dur}. As a result, a single contact can span multiple k states, and for every start and end of a contact, there is a k_i to a k_{i+1} state evolution. Next, contacts of capacity d between node i and j at state k are represented by $c_{k,i,j:d}$ arcs as shown in dotted lines in Figure 1B. Indeed, the contact average data rate can be obtained by dividing traffic volume over the contact duration (d/t_{dur}). Finally, traffic flow sources at node i with destination j and volume d are represented by $tf_{i,j}$: d labels enclosed in boxes on top of the corresponding generation state. It is worth clarifying that despite the example illustrates a topology with a single contact per node, the stated FSM formulation also supports modeling several overlapping contacts.

2.2 | Traffic flows and congestion

In FSM topology modeling, traffic flows can be studied and visualized as depicted in the examples in Figure 2. In this particular case, only a single traffic source $tf_{1,3}$ is considered for analysis, and contact $c_{3,2,3}$ is set to a maximum capacity of 5 traffic units. In this scenario, Node 1 executes CGR using the contact plan of Figure 1 to determine the best route toward the destination. Therefore, the result is that the best route to Node 3 is through contacts $c_{2,1,2}$ and $c_{3,2,3}$. Furthermore, the local congestion avoidance capability included in CGR as stated in Burleigh[15] effectively checks that local contact $c_{2,1,2}$ has enough capacity (10 traffic units) to accommodate the traffic flow $tf_{1,3}$: 10. However, as it does not evaluate the rest of the contacts in the path, it is not able to realize that contact $c_{3,2,3}$ has only got a capacity of 5 deriving in a congestion problem. As a result, 5 units of $tf_{1,3}$: 10 remain stuck in Node 2 until a new route becomes available.

Indeed, this is a remarkable evidence of how congestion can arise even in simplistic scenarios when each node applies CGR to its locally generated traffic. It is worth noticing that reactive mechanisms such as custody transfer could have warned Node 1 about the congestion in this case, but these schemes tend to derive in unwanted bouncing effects in paths with several hops.[22,23] If CGR were able to foresee all contact capacities in the path, a forwarding strategy like the one in Figure 2B would have delivered the complete traffic volume to its destination with the same contact plan but without congestion issues.

A second example with 2 traffic flows and a contact $c_{3,2,3}$ with a maximum capacity of 10 traffic units is also depicted in Figure 3A. In particular, it illustrates the resulting traffic flow after each node executes CGR with the same contact plan of Figure 1 to forward the traffic at state k_2 in Node 1 and at state k_3 in Node 2. The result is that Node 2 attempts to forward its local $tf_{2,3}$ through the direct path Nodes 2 to 3 by using the contact $c_{3,2,3}$, and Node 1 attempts to forward $tf_{1,3}$ through the path Nodes 1 to 2 to 3 by using the contacts $c_{2,1,2}$ and $c_{3,2,3}$. In consequence, in state k_3, Node 2 will have 20 traffic units of 2 different traffic flows but a contact capacity with Node 3 of only 10 traffic units. Since CGR is not aware of traffic flows from other nodes, Node 1 is not able to foresee the overbooking of contact $c_{3,2,3}$ provoking traffic $tf_{1,3}$ to be stuck in Node 2 until a new route becomes available in a future contact plan.

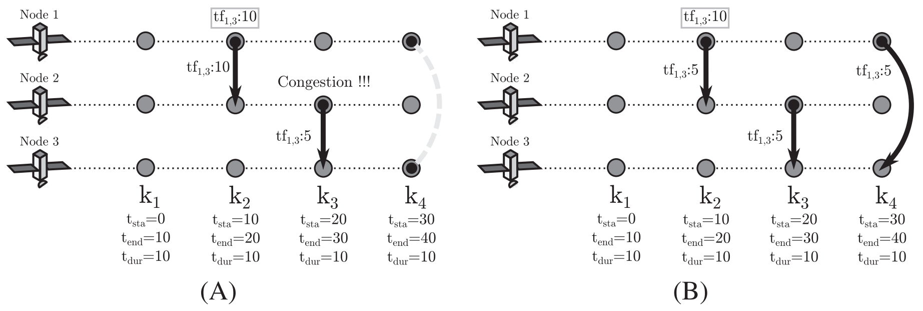

Node 1
Node 2
Node 3
tf1,3:10
tf1,3:10
Congestion !!!
tf1,3:5
k1 tsta=0 tend=10 tdur=10
k2 tsta=10 tend=20 tdur=10
k3 tsta=20 tend=30 tdur=10
k4 tsta=30 tend=40 tdur=10
(A)
tf1,3:5
tf1,3:5
tf1,3:5
(B)

FIGURE 2 Traffic flows with a single source. A, Congested contact graph routing traffic flow. B, Congestion-free traffic flow

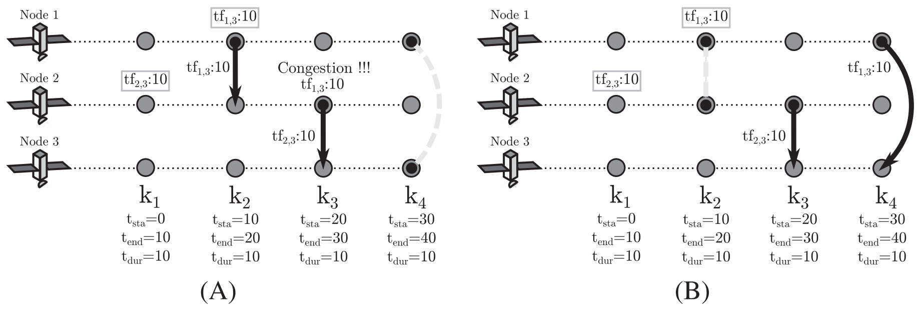

Node 1
Node 2
Node 3
tf1,3:10
tf2,3:10
tf1,3:10
Congestion !!!
tf1,3:10
tf2,3:10
k1 tsta=0 tend=10 tdur=10
k2 tsta=10 tend=20 tdur=10
k3 tsta=20 tend=30 tdur=10
k4 tsta=30 tend=40 tdur=10
(A)
tf1,3:10
tf2,3:10
(B)

FIGURE 3 Traffic flows with multiple sources. A, Congested contact graph routing traffic flow. B, Congestion-free traffic flow

On the other hand, Figure 3B illustrates an alternative congestion-free traffic flow for the same contact plan that allows traffics $tf_{2,3}$ and $tf_{1,3}$ to be effectively delivered to Node 3 at state k_4. Interestingly, this forwarding uses the same amount of communication resources as Figure 3A (2 contacts $c_{3,2,3}$ and $c_{4,1,4}$ of capacity 10) while improving the overall delivery ratio.

Finally, it is worth noticing that for both single and multiple traffic sources examples, nodes storage capacity was sufficient to accommodate each of the traffic flows. However, if this is not the case, another possible source of congestion in DTN arises. In the following sections, we will describe specific strategies to tackle these congestion problems provoked by (1) storage exhaustion, (2) excessive local traffic in a route path, and (3) excessive traffic from other nodes in the system.

3 | LOCAL PATH-AWARE CGR

While CGR algorithm maintains an updated status of the residual capacity of local contacts, LPA-CGR is designed to consider the complete path or *route capacity*. We define the latter as the traffic volume than can be forwarded through a path comprised by a set of 1 or more contacts. Indeed, route capacity is determined by the contact with least *residual capacity* or the node with less available buffer space along the route. As a result, a given neighbor will only be considered feasible by LPA-CGR if and only if the associated route capacity can accommodate the size of the forwarded packet. This allows LPA-CGR to avoid congestion generated by local traffic in advance, improving overall CGR performance.

In general, LPA-CGR can easily be integrated with existing CGR implementations as it only implies minor additions as shown in the detailed procedures of Algorithms 1 and 2. In addition to CGR,[15] LPA-CGR includes 2 new global variables in Algorithm 1: a route list (*Route*) and route capacity (*RouteCap*) that will store the current route information through the algorithm recursions. The recursion stack is initialized in lines 1 to 4 before iterating through all contacts (*xmits*) in the contact plan. Within the loop, each contact is evaluated for the destination node. To this end, route capacity and time-related variables such as forfeit (*Forfeit*) and best delivery (*BestDel*) are initialized in lines 8 to 10 if the contact connects to destination *D* (final contact in path); otherwise, these parameters are updated accordingly in lines 12 to 17. Then, line 18 evaluates if the route as calculated can accommodate the required data (*Ecc*). If route capacity is enough, either we found a feasible end-to-end path in line 19 (the xmit source is the local node), or we need to recurse in lines 21 to 23. If the capacity is depleted, the current contact is discarded in lines 24 to 25, and time variables restored in lines 26 to 27 before returning from the current recursion.

Algorithm 1: Local path-aware CGR contact review procedure

```
global : $Forfeit = \infty$; $Ecc, Route, RouteCap = \emptyset$; $Cplan$; $BestDel$; $ExclNodes \leftarrow PrevHop$
input  : $D, X$
output: $ProxNodes$
1  $ExclNodes \leftarrow D$;
2  $PrevForfeit = Forfeit$;
3  $PrevBestDel = BestDel$;
4  $PrevRouteCap = RouteCap$;
5  for $xmit \in Cplan \mid xmit_D = D, xmit_{t.end} \leqslant X$ do
6      $Route \leftarrow xmit$;
7      if $D == xmit_D$ then
8          $RouteCap = xmit_{Rate} * (xmit_{t.end} - max(xmit_{t.str}, T))$;
9          $Forfeit = xmit_{t.end}$;
10         $BestDel = max(xmit_{t.str}, T)$;
11     else
12         if $xmit_{ResCap} < RouteCap$ then
13             $RouteCap = xmit_{ResCap}$;
14         if $xmit_{t.end} < PrevForfeit$ then
15             $Forfeit = xmit_{t.end}$;
16         if $xmit_{t.str} > PrevBestDel$ then
17             $BestDel = xmit_{t.str}$;
18     if $RouteCap > Ecc$ then
19         if $xmit_S == ThisNode$ then
20             $ProxNodes \leftarrow [D, Route]$;
21         else
22             if $xmit_S \notin ExclNodes$ then
23                 LPA-CGR-CRP($xmit_S, xmit_{t.end}$)
24     $Route.Pop$;
25     $RouteCap = PrevRouteCap$;
26     $Forfeit = PrevForfeit$;
27     $BestDel = PrevBestDel$;
28 $ExclNode.Pop$;
29 return $ProxNodes$;
```

In addition to the variables considered in Algorithm 1, LPA-CGR also needs to keep track of the nodes buffer occupancy in each topology state so it can update correctly the residual capacity of each contact of the contact plan. A source node cannot send more data through a contact than the amount of data the destination node can store in its buffer, no matter the transmission data rate or the contact duration. That is the reason the forwarding Algorithm 2 has a global variable named *BufferCap* that take into account the occupancy of the buffer of all nodes in the network as a function of time by considering the begin and end of each topology state. This structure is initialized with the buffer size of each node; therefore, this information must be present in the contact plan provisioned to the nodes. The functioning of this algorithm is as follows: initially, a node needs to know the neighbor nodes *proxNodes* through which it can send a packet *B* to reach the destination node *D*. This step is achieved in line 1 by calling the LPA-CGR-CRP procedure. Then, if *proxNodes* is not empty (line 2), there are 2 possibilities to consider depending on the packet service type. On one side, if the packet has to be sent to all the neighbor nodes *N* (also known as *critical* packets in DTN[11]), it is forwarded to all feasible routes *R* (line 5) obtained in the previous step. Then, the structure *BufferCap* has to be updated accordingly with the calling to the *UpdateBuffersCapacity* routine. Finally, *DecreaseContactsCapacity* reduce the capacity of each contact in the contact plan. On the other side, if the packet is of unicast type, it has to be sent only to the best neighbor node *N* through route *R* present in *proxNodes* (line 10). The best neighbor node is chosen by the *sortNodes* subroutine in line 9 and make use of some predefined metric like the earlier delivery time of the packet as in this case. Lastly, lines 11 and 12 have the same purpose than lines 6 and 7 of updating *BufferCap* and reducing the residual capacities of the contacts in the contact plan in concordance with the planned routing for the packet.

Algorithm 2: Local path-aware CGR forward bundle procedure

```
   global: Cplan, BufferCap
   input : B
1  ProxNodes ← LPA-CGR-CRP(D, X);
2  if ProxNodes ≠ ∅ then
3      if Bundle_priority == Critical then
4          for [N, R] ∈ ProxNodes do
5              TransmitBundle(B, N);
6              UpdateBuffersCapacity();
7              DecreaseContactsCapacity();
8      else
9          [N, R] ← PopFront(SortNodes(ProxNodes));
10         TransmitBundle(B, N);
11         UpdateBuffersCapacity();
12         DecreaseContactsCapacity();
13 else
14     No Route Error
```

3.1 | Forward-back to previous node considerations

Despite LPA-CGR provides a congestion-free approach toward local traffic, it does not consider resource overbooking provoked by other nodes. In general, congestion provoked by traffic from other nodes can be amended at the cost of reforwarding the data until a capable route is found. However, in several cases, this implies returning the packet to the previous hop, which is forbidden in the specification of CGR[15] so as to avoid routing loops. As a result, this policy can make nodes to fail on reacting and recovering from an incorrect forwarding.

For example, as illustrated in the circular topology of Figure 4, a forthcoming contact $c_{2,2,3}$ with a capacity of 100 is considered by Nodes 1 and 2 as part of the best (fastest) path toward Node 3. Also, a future $c_{4,1,3}$ can be considered as an alternate yet later route for Node 1 traffic. Due to the fact that Node 1 forwards 50 traffic units to Node 2 at k_1, a congestion problem arise at $k2$ because of the limited capacity of $c_{2,2,3}$ for carrying both traffics ($tf_{1,3}$: 50 and $tf_{2,3}$: 100) to Node 3. At this point, Node 2 must reforward $tf_{1,3}$ and Node 1 results as the next best neighbor in the route with contacts $c_{3,2,1}$ and $c_{4,1,3}$. However, the nonreturn policy combined with the incorrect forwarding from Node 1 have negative consequences since the data are now stuck in Node 2 when a feasible and underused path through $c_{4,1,3}$ exists for k_4.

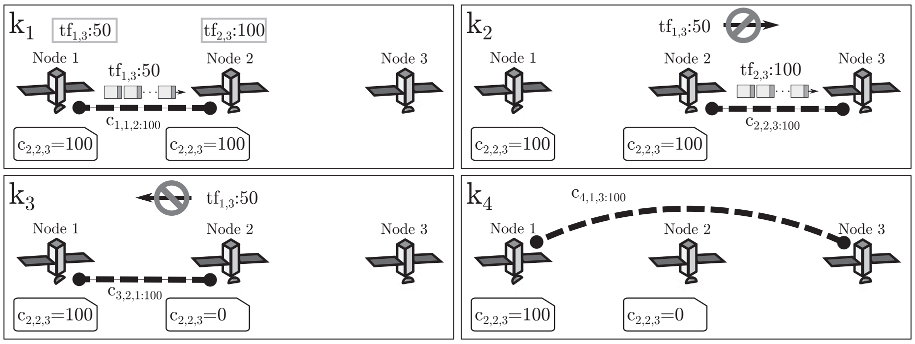


FIGURE 4 Forward-back policy blocking congestion reaction

Despite this negative outcome, the nonreturn policy is mandatory when implementing a single-hop greedy strategy such as CGR. However, its effects can be detrimental when LPA-CGR needs to react to congestion provoked by traffic from other nodes, and it is therefore discouraged. Nonetheless, if the DTN system traffic can be predicted as in most satellite sensor networks, alternative approaches such as the proposed in the next section can be considered to completely avoid this type of congestion.

4 | GLOBAL PATH-AWARE CGR

Contact graph routing has a congestion management limited to local contacts while LPA-CGR accounts for the same overhead as CGR, yet avoiding congestion on not local contacts. However, none of them accounts for traffic congestion avoidance provoked by nonlocal traffic. In this section, we describe GPA-CGR: a novel technique that has a limited in-band overhead and avoids traffic congestion in networks with predictable traffic. The latter is a typical feature of space networks applications where, in general, a mission operations center (MOC) determines in advance the use of on-board instruments and payloads. Therefore, the amount of data and generation time in the constellation system might be predictable enough to allow for a proper contact plan design as we describe hereafter.

Algorithm 3: Global path-aware CGR

```
    input  : Topology, Traffic
    output: Cplans[ ], RoutedTraffic
    global : RoutedTraffic, BufferCap
1   Generators ←GetSortedGenerators(Traffic);
2   RouteTable ←ComputeRouteTable(Topology, Generators);
3   for $g_i \in$ Generators do
4       while $g_i$.GetCurrentDataLength() > 0 do
5           Route ←getBestRoute($g_i$, RouteTable);
6           if Route $\neq \emptyset$ then
7               RouteGenerator($g_i$, Route);
8               UpdateBuffersCapacity();
9               DecreaseContactsCapacity();
10              UpdateRouteTable();
11          else
12              skip $g_i$;
13  for $n_i \in$ Topology.Nodes do
14      Cplans.At($n_i$) ←GetContactPlan(RoutedTraffic, $n_i$);
15  return [Cplans[ ], RoutedTraffic]
```

Global path-aware CGR is illustrated in Figure 5 and described in Algorithm 3. The processing intense part of GPA-CGR takes place in the contact plan design stage (Stage 1), where a centralized node can take advantage of a priori knowledge of both the predicted *Topology* and the expected *Traffic* that the procedure receives as input. The purpose is to generate a custom contact plan for each node by reserving capacities in the network in such a way that congestion can be later avoided when nodes apply simpler routing algorithms (Stage 2). The procedure begins in line 1 by obtaining the traffic flows (*generators*) sorted by generation time from the planned *Traffic* Matrix. A *generator* g_i consist of a traffic volume, generation time, and a source and destination node. Then, a global *RouteTable* is computed in line 2 that contains all possible routes each *generator* can use in a store carry-and-forward manner to reach its destination node. In lines 3 to 12 ,each *generator* is "routed" with a process that consist first in obtaining the

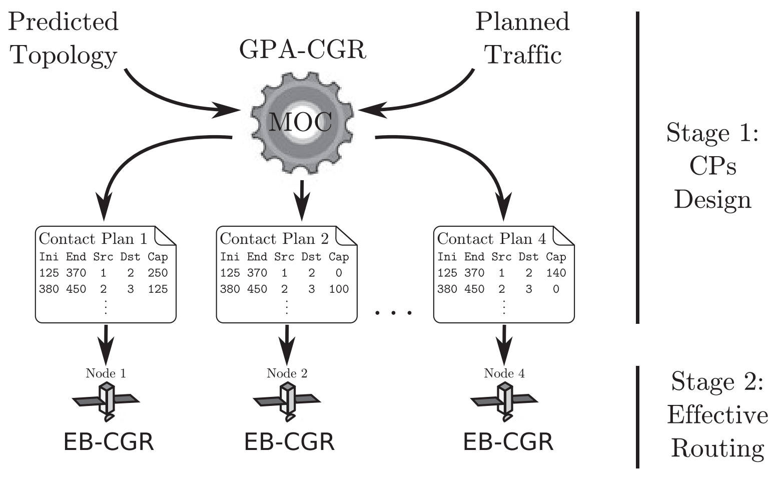


FIGURE 5 Global path-aware CGR (GPA-CGR) procedure. EB-CGR, extension block contact graph routing; MOC, mission operations center

best *Route* from the *RouteTable* in line 5 and then making a flow assignment to the corresponding contacts of the route in the *RoutedTraffic* structure in line 7. Next, the *BufferCap* is updated in line 8, the contacts capacity reduced in line 9, and the *RouteTable* updated in line 10. The *BufferCap* structure take account of the nodes buffer occupancy as a function of time in the same way it does in LPA-CGR scheme. The only difference here is that this structure is a global one and is aware of all the network nodes and traffic. Finally, in lines 13 and 14, a custom contact plan is obtained for each node by considering only the traffic generated by that node through all contacts in the topology. All other contacts shall remain with a capacity of 0 implying that the contact exist and that it can course traffic yet forbidding the local node to take it into consideration for local route calculations. Consequently, the summation of all derived contact plans result in a contact plan with less or equal capacity than the original topology.

Therefore, once all contact plans are disseminated, each node can use plain CGR over its specific contact plan. However, since the origin node is the only 1 with access to its reserved capacity to perform calculations, intermediate nodes will have to rely and honor this original route path. Indeed, this path needs to be encoded in the transmitted packet by specific protocols such as extension block CGR (EB-CGR).[20] Since route path is in the header, traffic flowing through intermediate nodes does not require a route recalculation neither need to know the topology capacity value originally reserved to other nodes. A simple path validation can be enough to assure the required contacts exist in the local contact plan.

Additionally, it is worth noticing that as GPA-CGR contact plans are designed to accommodate predicted traffic, it accounts with a limited capacity to further route unpredicted traffic or mitigate topology or traffic prediction inaccuracies. To avoid this, GPA-CGR allows to incorporate error margins in the capacity calculations or even distribute a second and global backup contact plan with the marginal remaining capacities in the system for all nodes to consider in case of these unplanned events. However, this backup approach will not account with the congestion management feature of the original GPA-CGR. We leave the analysis and definition of this strategy as further research.

4.1 | Contact and neighbor-based queueing considerations

One aspect to pay special attention is the fact that CGR-based algorithms such as EB-CGR pass through 3 different stages: routing a packet, queuing a packet in the node memory, and forwarding a packet. Although the entire route is calculated in the routing phase, only the first neighbor node of the route is saved for forwarding. Next, the queuing of the packet in the node memory is labeled with the next neighbor node the packet has to be forwarded. Finally, when a contact opportunity is established with some node, the local node searches in its storage for packets that were queued for that neighbor node and performs the effective forwarding to the next hop. In other words, the forwarding in CGR as stated in Burleigh[15] and EB-CGR is executed on a per neighbor basis.

This behavior could lead to unintended traffic flows when using GPA-CGR as we show with the example illustrated in Figure 6. The topology is composed of 4 nodes, 4 states, and 2 traffic flows: $tf_{1,4}$ and $tf_{2,3}$. When Node 1 uses EB-CGR to forward $tf_{1,4}$, it chooses the reserved capacity of contacts $c_{1,1,2}$, $c_{2,2,3}$, and $c_{3,3,4}$, while node 2 only sees capacity on contact $c_{4,2,3}$. The beforementioned routing phase is performed by EB-CGR in each node by respecting the planned routing made by GPA-CGR in Stage 1. However, a conflict arises in the forwarding phase due to the queuing by next neighbor node policy. Although Node 2 calculates properly that $tf_{2,3}$ has to use the contact $c_{4,2,3}$, as it is the only with some capacity, it does the queuing of that packet for Node 3 as next neighbor. Then, when the $c_{2,2,3}$ is established, Node 2 has 2 packets queued for Node 3 but not enough capacity for both. Note that if traffic $tf_{2,3}$ is sent instead of $tf_{1,4}$ the effective forwarding is not the same as the planned by GPA-CGR in Stage 1.

To avoid this kind of situations, the queuing policy in GPA-CGR must be changed in a way that routing decisions can be respected in the forwarding phase. Therefore, the queuing of the packets must be done with a queuing-by-contact policy instead of a queuing-by-neighbor-node policy. A somewhat different but analogue effect called head-of-line blocking occurs in First In First Out (FIFO)-buffered network switches when an older packet cannot be forwarded to the corresponding output due to a contention effect. In that case, one way to overcome the limitation is by using virtual output queues.

4.2 | Evolutionary optimizations to GPA-CGR

As it was described in the last section, the GPA-CGR procedure consists in routing the traffic generators by start time over a global route table and then producing a flow assignment to the contacts of the topology. Next, a custom contact plan is derived for each node based on that flow assignment.

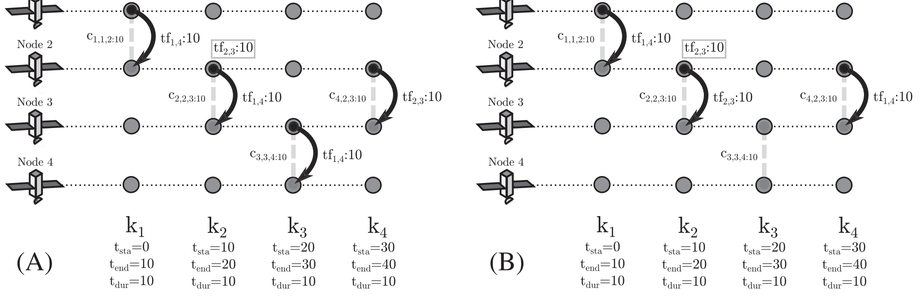


FIGURE 6 Queueing conflict. A, Global path-aware CGR planned routing (Stage 1). B, Extension block CGR effective routing (Stage 2).

Therefore, a solution to the first stage of the routing scheme consist of a set of contact plans, one for each node in the network. Although the solution obtained in this way is free from congestion, there might exist different solutions that can be obtained by other mechanisms that are also free of congestion. As a result, another aspect to take under consideration is the question of whether the solution provided by GPA-CGR can be optimized in function of some metric like delivery time or network resource usage.

One way of exploring among the different solutions could be by varying some contacts from the topology and then applying the GPA-CGR algorithm to the modified topology. As we will see, this method can lead to solutions that besides being free of congestion are better solutions in some predefined metric like delivered packets or network resource usage. This results in a complex combinatorial search problem that can be addressed with population-based metaheuristics like evolutionary algorithms.[25] In general, evolutionary algorithms are based on the notion of competition, imitating the evolution of species.[26] In Algorithm 4, we describe evolutionary GPA-CGR (EGPA-CGR) that receives as input the expected *Topology*, the planned *Traffic* matrix, the crossing over probability (*PCr*), the mutation probability (*PMt*), the population limit (*PopLim*), and the numbers of iterations (*Iters*) as stopping criteria. The output is the same as GPA-CGR: a custom contact plan for each node of the network and a flow assignment that is useful to know when a solution is better than another one.

Algorithm 4: Evolutionary GPA-CGR

```
    input  : Topology, Traffic, PCr, PMt, Iters, PopLim
    output: Cps[ ], RoutedTrf
    global : Pop, NewPop
    global : BestTop, BestTopI
    global : BestRoutedTrf, BestRoutedTrfI
    global : BestCps[ ], BestCpsI[ ]
1   InitializePopulation(PopLim);
2   SortPopulation(Pop);
3   for i ← 1 to Iters do
4       j ← Pop.Begin();
5       CurTop ← Pop.At(j);
6       j ← j + 1;
7       while j ≠ Pop.End() do
8           NewTop ← Pop.At(j);
9           if Random(0,1) ⩽ PCr then
10              TopA, TopB =CrossOver(CurTop, NewTop);
11              Mutate(TopA, PMt);
12              Mutate(TopB, PMt);
13              NewPop.PushBack(TopA);
14              NewPop.PushBack(TopB);
15              if NewPop.Size() ⩾ PopLim then
16                  break;
17              Pop.PopFront();
18              j ← Pop.Begin();
19              if j ≠ Pop.End() then
20                  curTop = Pop.At(j);
21                  j ← j + 1;
22          else
23              j ← j + 1;
24              if j = Pop.End() then
25                  Pop.PopFront();
26                  j ← Pop.Begin();
27                  CurTop ← Pop.At(j);
28                  j ← j + 1;
29      SortPopulation(NewPop);
30      BestTopI = NewPop.Front();
31      [BestCpsI[ ], BestRoutedTrfI] = GPA-CGR(BestTopI, Traffic);
32      if IsBetter(BestRoutedTrfI, BestRoutedTrf) then
33          BestRoutedTrf = BestRoutedTrfI;
34          BestCps[ ] = BestCpsI[ ];
35      Pop = NewPop;
36      NewPop.Clear();
37  return [BestCps[ ], BestRoutedTrf]
```

In line 1, the initial population (*Pop*) is filled with mutated topologies generated by modifying the input topology. In line 2, the initial population is sorted based on some fitness criteria that consists in the prioritization of some metrics over others as follows: the GPA-CGR algorithm is applied to every topology, and 3 metrics are calculated taking into account the *RoutedTrf* structure obtained as a result: packet delivery ratio, normalized system contact time, and delivery time. These metrics will be explained in Section 5, and the user has the possibility of weighting each one, thereby changing the fitness criteria and adjusting the focus of the objective solution obtained in the search process. The for loop in lines 3 to 36 traverses the population structure with the aim of obtaining a new population (*NewPop*) that can lead to better solutions by performing operations of crossing over, mutation, and selection. Lines 4 to 8 obtain 2 individuals of the population (*CurTop* and *NewTop*) that will be subjected to operations of crossing over and mutation. The while loop between lines 7 and 28 has the objective of changing the individual *NewTop* with the aim of trying different combinations. The crossing over is performed in concordance with the *PCr* probability in lines 9 and 10, and the *TopA* and *TopB* children topologies are obtained. Lines 11 and 12 mutate those children in agreement with the *PMt* probability that are finally saved in *NewPop* in lines 13 and 14. Lines 15 and 16 allow to break any iteration if the population grows more than a certain limit. Line 17 removes an individual from *Pop* one time that it achieves a crossing over operation with other individual. Lines 18 to 21 restore the iterators to obtain new individuals of the population when the crossing over was effectively accomplished, and lines 23 to 28 do the same task when no crossing over was performed. Line 29 sorts *NewPop* individuals once an iteration has ended, and line 30 gets the best iteration individual topology (*BestTopI*). Line 31 obtains the flow assignation (*BestRoutedTrfI*) and the derived contact plans array (*BestCpI*) by calling the GPA-CGR procedure. Lines 32 to 34 update the best global solution. Lines 35 to 36 assign *NewPop* to *Pop* and clear *NewPop* to begin a new iteration of the genetic algorithm. Finally, the best global solution consisting of a flow assignment, and a contact plan array is returned in line 37.

5 | PERFORMANCE EVALUATION

To evaluate the described congestion avoidance techniques, we propose a particular case study of a realistic linear DTN constellation of low Earth orbit satellites. Several reasons support the linear flight formation such as NASA's A-Train constellation.[27] Among them, if spacecrafts are close enough in this formation, they perceive quite the same gravity perturbations allowing significant propellant savings (generally used for station keeping). Furthermore, it is a desirable formation from a launcher point of view as it does not require a transfer orbit to deliver the cluster nodes to their final position. Additionally, from an earth observation mission perspective, it allows to obtain stereoscopic or wide-angle earth observation images. If supplied with intersatellite links (ISL) and with a DTN data management approach, the proposed constellation can share downlink transponders optimizing the usage of limited resources such as frequency allocations and power.

In particular, we propose a 4 satellite linear formation with the orbital parameters and time lapses detailed in Table 1. By using SGP-4 (a well-known satellite propagator), combined with a communication range of 1000 km for ISL and 2500 km for Earth-to-satellite links (ESL), we obtain a real-time–evolving topology suitable for DTN applications. Henceforth, ISL and ESL are solely thought as full duplex and point-to-point, disregarding shared medium access schemes that fail to perform properly in extensive networks as they assume physical adjacency of many nodes. Therefore, if nodes are equipped with a single communication subsystem, further topology design and fractionation such as the proposed in Fraire and Finochietto[4,28] are mandatory. Also, to evaluate a total of 4 flyby over a ground station located in Córdoba, Argentina (−32° Latitude, −64° Longitude), the evaluation interval spans a total duration of 78 199 seconds, where Nodes 1 and 4 alternate the usage of their downlink transponder in each pass. As a result, the final topology for the system, modeled by a FSM, is illustrated in Figure 7.

To complete the description of the case study, we now consider the traffic to be delivered from all satellites to the ground station identified as Node 0. Combining a 100 Kbps link speed with a packet size of 12.5 KBytes (1 packet per second) and a total ground contact time of 1149 seconds for Node 1 and 1222 for Node 4, the whole system should be able to deliver a total payload of 29.637 MBytes or 2371 packets in the proposed interval. In other words, the maximum traffic that the topology can handle for each node is 7.4 MBytes or 592 packets. Henceforth, we consider this

TABLE 1 Case study time lapses and orbital parameters

Topology Interval Start	Jan 1, 2016, 0 h 0 min 0 s
Topology Interval End	Jan 1, 2016, 21 h 43 min 18 s
Bstar Coefficient, /ER	90039
Inclination, deg	98°
RAAN, deg	0°
Eccentricity	9152
Argument of perigee, deg	0°, 5°, 10°, and 15°
Mean anomaly, deg	0°
Mean motion, rev/d	15.07561758 rev/d

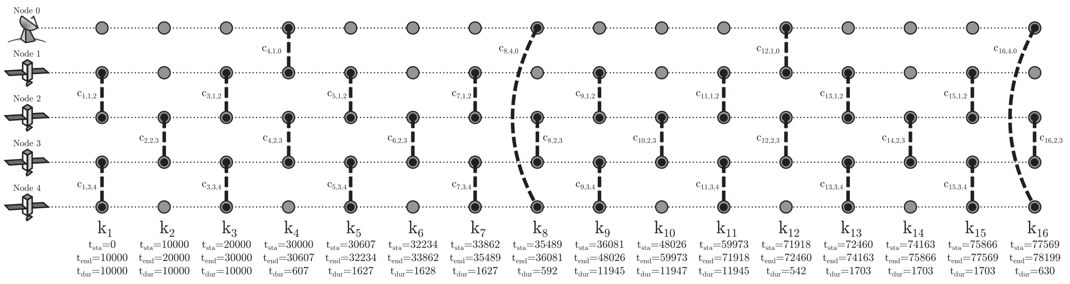

Node 0
Node 1
Node 2
Node 3
Node 4
$c_{4,1,0}$
$c_{8,4,0}$
$c_{12,1,0}$
$c_{16,4,0}$
$c_{1,1,2}$
$c_{3,1,2}$
$c_{5,1,2}$
$c_{7,1,2}$
$c_{9,1,2}$
$c_{11,1,2}$
$c_{13,1,2}$
$c_{15,1,2}$
$c_{2,2,3}$
$c_{4,2,3}$
$c_{6,2,3}$
$c_{8,2,3}$
$c_{10,2,3}$
$c_{12,2,3}$
$c_{14,2,3}$
$c_{16,2,3}$
$c_{1,3,4}$
$c_{3,3,4}$
$c_{5,3,4}$
$c_{7,3,4}$
$c_{9,3,4}$
$c_{11,3,4}$
$c_{13,3,4}$
$c_{15,3,4}$
k_1 $t_{sta}=0$ $t_{end}=10000$ $t_{dur}=10000$
k_2 $t_{sta}=10000$ $t_{end}=20000$ $t_{dur}=10000$
k_3 $t_{sta}=20000$ $t_{end}=30000$ $t_{dur}=10000$
k_4 $t_{sta}=30000$ $t_{end}=30607$ $t_{dur}=607$
k_5 $t_{sta}=30607$ $t_{end}=32234$ $t_{dur}=1627$
k_6 $t_{sta}=32234$ $t_{end}=33862$ $t_{dur}=1628$
k_7 $t_{sta}=33862$ $t_{end}=35489$ $t_{dur}=1627$
k_8 $t_{sta}=35489$ $t_{end}=36081$ $t_{dur}=592$
k_9 $t_{sta}=36081$ $t_{end}=48026$ $t_{dur}=11945$
k_{10} $t_{sta}=48026$ $t_{end}=59973$ $t_{dur}=11947$
k_{11} $t_{sta}=59973$ $t_{end}=71918$ $t_{dur}=11945$
k_{12} $t_{sta}=71918$ $t_{end}=72460$ $t_{dur}=542$
k_{13} $t_{sta}=72460$ $t_{end}=74163$ $t_{dur}=1703$
k_{14} $t_{sta}=74163$ $t_{end}=75866$ $t_{dur}=1703$
k_{15} $t_{sta}=75866$ $t_{end}=77569$ $t_{dur}=1703$
k_{16} $t_{sta}=77569$ $t_{end}=78199$ $t_{dur}=630$

FIGURE 7 Linear formation time-evolving topology

network load as 1 ($\rho = 1$), for which 2371 packets are the maximum packet delivery ratio (ie, 1). However, such a throughput assumes that congestion is properly managed within the constellation. The latter is not a trivial achievement since the contact to ground station can easily become a traffic bottleneck due to its limited capacity in comparison to ISL capacities. Therefore, the proposed case study becomes a valid case study scenario for comparing different congestion avoidance techniques, and it was in fact used in Fraire et al[9] with 1 important difference: in this case, there exists an additional constraint on each node buffer capacity with the objective of analyzing its impact in the proposed algorithms performance. Particularly, each satellite node has a small-sized 9 MBytes buffer and the ground station a large-sized 100 GBytes buffer to be able to receive all the generated traffic.

Probably, the most important metric considered to compare CGR, LPA-CGR, GPA-CGR, and EGPA-CGR is the total payload effectively delivered to ground station; however, it is also necessary to understand how efficiently such a delivery was achieved. Therefore, we also measure the overall constellation contact time usage (ESL and ISL communication resources) and the payload delivery time. Henceforth, we will refer to these metrics as *packet delivery ratio*, *normalized system contact time*, and *delivery time*, respectively. It is worth clarifying that the *packet delivery ratio* is calculated as the packets delivered over the packets generated and the *normalized system contact time* as the contact time usage for communication over the delivered packets, as a way to make better and fairer comparisons among the routing schemes when packets cannot reach its destinations. Another aspect to mention is that in this study, we put the focus on proactive schemes for avoiding congestion disregarding reactive mechanisms like custody

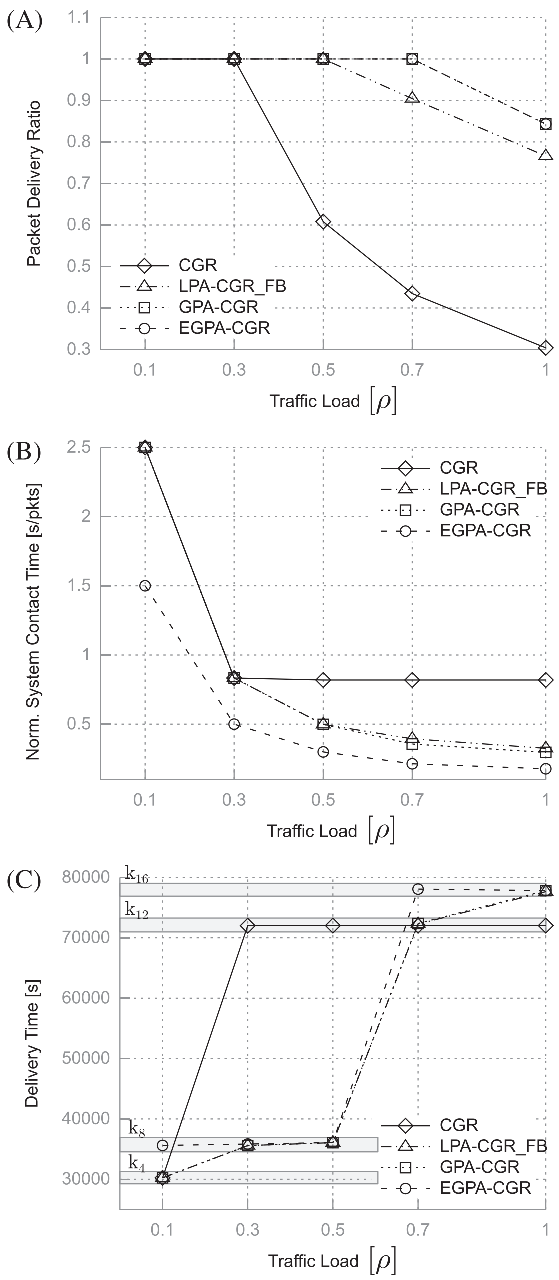


FIGURE 8 Simulation results. CGR, contact graph routing; EGPA-CGR, evolutionary global path-aware contact graph routing; GPA-CGR, global path-aware contact graph routing; LPA-CGR_FB, local path-aware contact graph routing with forward-back enabled

transfer. Therefore, when packets arrive at a node with an overflowed buffer, they are discarded. Finally, to determine and analyze these metrics, all algorithms and topologies were implemented in an Omnet++ based simulator specifically developed for DTN congestion evaluation with the results described in Section 5.1.

5.1 | Result analysis

The obtained simulation results are summarized in Figure 8, where the abscissa axis represents an increasing traffic load from $\rho = 0.1$ (59 packets or 737.5 KB per node) up to $\rho = 1$ (592 packets or 7.4 MB per node). On the ordinate axis, *packet delivery ratio*, *normalized system contact time*, and *delivery time* are plotted for plain CGR, LPA-CGR with forward-back (FB) enabled, GPA-CGR, and EGPA-CGR. In particular, FB is disabled for CGR since routing loops cause its *normalized system contact time* metric to drastically increase. The EGPA-CGR scheme is configured to improve the mentioned metrics of the GPA-CGR solutions in the following order: first delivered packets, then normalized system contact time, and finally delivery time. Figure 8C also have 4 highlighted areas representing k_4, k_8, k_{12}, and k_{16} states where the constellation have contacts with the ground segment through arcs $c_{4,1,0}$, $c_{8,4,0}$, $c_{12,1,0}$, and $c_{16,4,0}$, respectively.

In general, for low throughput, all congestion management mechanisms provides an optimum delivery as shown in Figure 8A. However, an inflection point is evidenced for CGR beyond $\rho = 0.3$ (177 packets per node totaling 708 in the system). To properly explain this behavior, Figure 9A illustrates the traffic flows for plain CGR for the particular case of $\rho = 1$. Here, CGR is only using contacts $c_{4,1,0}$ and $c_{12,1,0}$ to reach the destination Node 0 with a decreasing packet delivery ratio as the traffic load increases. This is because when each node routes its traffic, it considers a usable contact in its contact plan ($c_{4,1,0}$), ignoring its capacity can be overwhelmed by others traffic. Only Node 1 is able to notice the overbooking of $c_{4,1,0}$ as it receives a traffic flow of $tf_{2,0}$: 592 in k_1, and $tf_{3,0}$: 592 plus $tf_{4,0}$: 128 in k_3. Nevertheless, despite this node can calculate alternative routes through $c_{8,4,0}$, it is unable to FB any data due to the policy explained before. As a result, the traffic is stuck until a second favorable contact is used in k_{12}. Note also that many packets get lost due to the buffers capacity restriction of each node as there is not a custody mechanism available.

Local path-aware CGR improves CGR packet delivery ratio performance as it allows intermediate Nodes 4, 3, and 2 to also predict and react to congestion in advance. In contrast to CGR flows illustrated in Figure 9A, intermediate nodes 3 and 2 foresee future contacts capacity that allows them to determine that $c_{4,1,0}$ is fully used before the traffic arrives at node 1. As a result, packets are forwarded back through alternate routes including $c_{8,4,0}$, $c_{12,1,0}$, and $c_{16,4,0}$. However, since LPA-CGR ignores other node's traffic, an important amount of communication resources is wasted in bouncing data back and forth until each node's local contact plan capacities are depleted. At this stage, some packets might end stuck in intermediate nodes making LPA-CGR packet delivery ratio to rise up to 76.6% of the total throughput. In general, CGR use less system contact time than LPA-CGR, but the fact that LPA-CGR delivers a much higher quantity of packets to destination makes the normalized system contact time metric to be always better for LPA-CGR as it is seen in Figure 8B. On the other hand, this comes at the expense of obtaining a higher delivery time for LPA-CGR when the traffic load is $\rho = 1$ as shown in Figure 8C.

The best results are clearly obtained by GPA-CGR and EGPA-CGR as these schemes are capable of avoiding congestion by providing a specific contact plan to each node. Figure 9B evidences that the fact of eliminating the contact $c_{2,2,3}$ and making an adequate capacity reserve allows Nodes 3 and 4 to take the convenient decisions of routing its traffic through contacts $c_{8,4,0}$ and $c_{16,4,0}$ while Nodes 1 and 2 can freely use contacts $c_{4,1,0}$ and

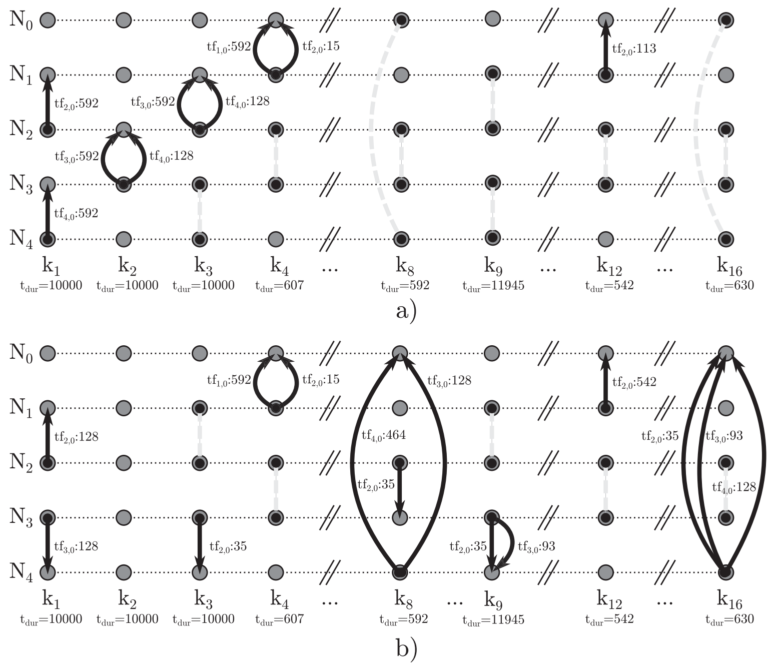


FIGURE 9 Traffic flows for, A, contact graph routing and, B, evolutionary global path-aware contact graph routing when $\rho = 1$

$c_{12,1,0}$. This kind of procedures allows to also obtain better results than CGR or LPA-CGR in normalized system contact time. Note that EGPA-CGR was configured to improve the delivered packets and the resource use before delivery time. Therefore, it will always obtain better or equal metrics than GPA-CGR in Figures 8A,B and maybe a worst metric in Figure 8C. This is due to the fact that when normalized system contact time is prioritized instead of delivery time, fewer contacts will be used with the aim of reducing the network usage at the expense of delivering the traffic later. In this way, EGPA-CGR provides a flexible and configurable scheme for obtaining customized contact plans that optimize certain metrics.

Another effect evidenced in the results is how the buffer capacity constraint affects the network performance as the traffic load increases. Noticeably, when $\rho \geqslant 0.7$, even GPA-CGR and EGPA-CGR schemes are stressed in a way that cannot deliver all the generated packets in the topology period. To make a deeper analysis on the buffer restriction, we set the traffic load to $\rho = 1$ and vary the buffer capacity of each satellite node between 7.4 MBytes and 14.4 MBytes. The packet delivery ratio metric is obtained by using the described routing schemes. Furthermore, the PA-CGR scheme that does not take into account the buffer limitations is also incorporated to the analysis with the aim of comparing the improve obtained with LPA-CGR. As it is clearly shown in Figure 10, GPA-CGR and EGPA-CGR can deliver all the generated payload when the buffers are large enough while PA-CGR results more stressed than LPA-CGR for the same scenario.

5.2 | Algorithm analysis

To summarize and compare the different routing strategies, we will discuss the main aspects embodied in Table 2. Both CGR and LPA-CGR avoid congestion by considering only the locally generated traffic. CGR takes into account only the first contact of the route while LPA-CGR extends that scope to all the contacts and buffers capacity in the same route. The improvement in the network metrics obtained by using LPA-CGR is achieved at the expense of an increment in the computational complexity of the algorithm. On the other hand, both GPA-CGR and EGPA-CGR obtain even better metrics because they avoid congestion at a global level by assigning a custom contact plan for each node based on the assumption that traffic generators can be planned beforehand by a central node. Another advantage of using the global schemes is in the algorithms execution number. Global path-aware CGR and EGPA-CGR are applied only once at a central node, and then the EB-CGR associated algorithm is applied once per packet

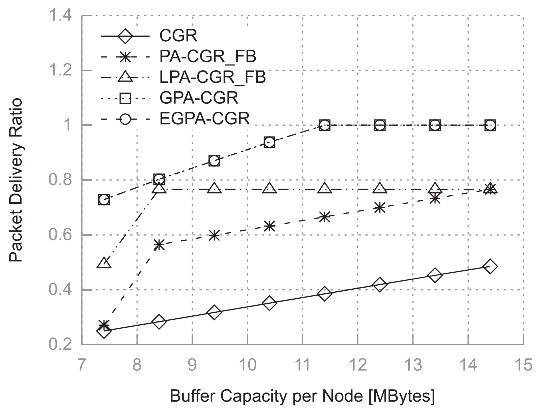


FIGURE 10 Buffers variation results. CGR, contact graph routing; EGPA-CGR, evolutionary global path-aware contact graph routing; GPA-CGR, global path-aware contact graph routing; LPA-CGR_FB, local path-aware contact graph routing with forward-back enabled; PA-CGR_FB, path-aware contact graph routing with forward-back enabled

TABLE 2 Algorithms comparison table

	Algorithm		
Feature	CGR	LPA-CGR	GPA-CGR / EGPA-CGR
Traffic Congestion Level	Local (first contact)	Local (all route contacts)	Global
Required Information	Topology	Topology	Topology + Traffic
Contact Plan	The same for all nodes	The same for all nodes	One different per node
Packet Overhead	None	None	Computed route
Executions Number	Once per packet at each node	Once per packet at each node	Once at central node + once per packet only at source node

Abbreviations: CGR, contact graph routing; EGPA-CGR, evolutionary global path-aware contact graph routing; GPA-CGR, global path-aware contact graph routing; LPA-CGR, local path-aware contact graph routing.

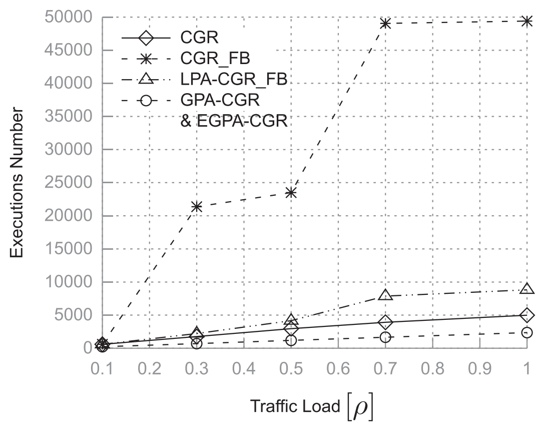


FIGURE 11 Algorithm executions. CGR, contact graph routing; CGR_FB, contact graph routing with forward-back enabled; EGPA-CGR, evolutionary global path-aware contact graph routing; GPA-CGR, global path-aware contact graph routing; LPA-CGR_FB, local path-aware contact graph routing with forward-back enabled

only at the traffic source node, and the computed route is carried in the packet overhead. Figure 11 shows the executions number performed with each algorithm as a function of a varying traffic load for the scenario described previously in this section. The best performance in terms of executions number is achieved by the global schemes. Contact graph routing is considered both with and without the FB option enabled. Local path-aware CGR_FB is executed less times than CGR_FB thanks to the extended congestion avoidance capability of the former while the apparent disadvantage of LPA-CGR_FB regarding CGR is considered to be a trade-off for the great improvement obtained by LPA-CGR_FB in the delivery packets ratio.

6 | CONCLUSION

In this article, the congestion problem in disruption-tolerant satellite networks was introduced and described. In particular, we showed that this phenomenon has not been completely considered by current algorithms such as CGR despite that it can drastically reduce the overall data delivery performance of DTN systems. Furthermore, we validated that congestion might arise either by excessive local traffic along a route path or excessive remote traffic both deriving in an overbooking of contacts or nodes storage.

As a result, we contributed with 2 novel strategies to mitigate and avoid congestion in a proactive manner. On one hand, LPA-CGR was introduced as mean of extending current CGR solution to consider the complete path capacity to mitigate congestion provoked by local traffic. Second, we proposed GPA-CGR that takes advantage of traffic predictability to completely avoid congestion by reserving communication resources in advance. Indeed, this approach allowed us to explore unique optimization opportunities with EGPA-CGR.

Finally, by evaluating these solutions in a reference satellite constellation system, we demonstrated that LPA-CGR can significantly outperform CGR in data delivery time and ratio with a minimal increment in algorithm complexity. Also, when assuming predictable traffic, a more important betterment was observed for GPA-CGR and its EGPA-CGR optimization at the expense of the provision of 1 specific contact plan per node. Finally, we leave as further work the research of GPA-CGR extensions to cope with predictability uncertainties and system communication failures.

REFERENCES

1. Rashvand HF, Abedi A, Alcaraz-Calero JM, Mitchell PD, Mukhopadhyay SC. Wireless sensor systems for space and extreme environments: a review. *IEEE Sensors J*. 2014; 14(11):3955-3970.
2. Brown O, Eremenko P. Fractionated space architectures: A vision for responsive space. In *4th Responsive Space Conference, American Institute of Aeronautics and Astronautics. Paper no. AIAA RS4 2006 1002*: Los Angeles, CA, 2006, Paper RS4-2006-1002.
3. Krishnamurthy A, Preis R. Satellite formation, a mobile sensor network in space. In *19th IEEE International Parallel and Distributed Processing Symposium*. Denver, CO, USA, 2005, pp. 7-, https://doi.org/10.1109/IPDPS.2005.387
4. Fraire JA, Finochietto JM. Design challenges in contact plans for disruption-tolerant satellite networks. *IEEE Commun Mag*. 2015; 53(5):163-169.
5. Cerf V, et al. RFC-4838: Delay-tolerant networking architecture, Network Working Group, IETF, April 2007.
6. Caini C, Cruickshank H, Farrell S, Marchese M. Delay and disruption tolerant networking (DTN): an alternative solution for future satellite networking applications. *In Proceedings of the IEEE*, vol. 99, no. 11, pp. 1980-1997, NOV. 2011, https://doi.org/10.1109/JPROC.2011.2158378.
7. Burleigh S, Hooke A, Torgerson L, et al. Delay-tolerant networking: an approach to interplanetary internet. *IEEE Commun Mag*. 2003; 41(6):128-136.
8. Fall K. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '03. ACM: New York, NY, USA, 2003;27-34.
9. Fraire J, Madoery P, Finochietto J, Birrane E. Congestion modeling and management techniques for predictable disruption tolerant networks. In *40th IEEE Conference on Local Computer Networks (LCN 2015)*. Clearwater Beach, FL, USA, 2015, pp. 544-551, https://doi.org/10.1109/LCN.2015.7366369.

10. Silva AP, Burleigh S, Hirata CM, Obraczka K. A survey on congestion control for delay and disruption tolerant networks. *Ad Hoc Networks*. Volume 25, Part B, 2015, pp. 480-494, https://doi.org/10.1016/j.adhoc.2014.07.032.
11. Scott K, Burleigh S. RFC-5050: Bundle protocol specification, Network Working Group, IETF, November 2007.
12. Burleigh S, Jennings E, Schoolcraft J. Autonomous congestion control in delay-tolerant networks. In *9th International Conference on Space Operations*: Rome, Italy, June 2006.
13. Birrane E. Congestion modeling in graph-routed delay tolerant networks with predictive capacity consumption. In *Global Communications Conference (GLOBECOM)*, Atlanta, GA, 2013, pp. 3016-3022, https://doi.org/10.1109/GLOCOM.2013.6831534.
14. Burleigh S. Interplanetary overlay network: an implementation of the DTN bundle protocol. In *Consumer Communications and Networking Conference*, Las Vegas, NV, USA, 2007, pp. 222-226, https://doi.org/10.1109/CCNC.2007.51.
15. Burleigh S. Contact graph routing, IETF-Draft, 2010.
16. Vallado DA. *Fundamentals of Astrodynamics and Applications* (4th edn.) Microcosm: Hawthorne, CA, 2007.
17. Wyatt J, Burleigh S, Jones R, Torgerson L, Wissler S. Disruption tolerant networking flight validation experiment on NASA's epoxi mission. In *Proceedings of the 2009 First International Conference on Advances in Satellite and Space Communications*, SPACOMM '09. IEEE Computer Society: Washington, DC, USA, 2009;187-196.
18. Araniti G, Bezirgiannidis N, Birrane E, et al. Contact graph routing in DTN space networks: overview, enhancements and performance. *IEEE Commun Mag*. 2015; 53(3):38-46.
19. Bezirgiannidis N, Caini C, Padalino Montenero D, Ruggieri M, Tsaoussidis V. Contact graph routing enhancements for delay tolerant space communications. In *Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, Livorno, 2014, pp. 17-23, https://doi.org/10.1109/ASMS-SPSC.2014.6934518.
20. Birrane E, Burleigh S, Kasch N. Analysis of the contact graph routing algorithm: bounding interplanetary paths. *Acta Astronaut*. 2012; 75:108-119.
21. Segui J, Jennings E, Burleigh S. Enhancing contact graph routing for delay tolerant space networking. In *2011 IEEE Global Telecommunications Conference (GLOBECOM 2011)*. IEEE: Houston, TX, USA, December 2011;1-6.
22. Fraire J, Ferreyra P. Assessing DTN architecture reliability for distributed satellite constellations: preliminary results from a case study. In *2014 IEEE Biennial Congress of Argentina (ARGENCON)*: Bariloche, Argentina, June 2014;564-569.
23. Fraire J, Madoery P, Finochietto J. Leveraging routing performance and congestion avoidance in predictable delay tolerant networks. In *International Conference on Wireless for Space and Extreme Environments (WISEE)*, Noordwijk, 2014, pp. 1-7, https://doi.org/10.1109/WiSEE.2014.6973079.
24. Lary DJ. An objectively optimized earth observing system. In *Aerospace Conference, Big Sky, MT*, USA, 2007, pp. 1-3. https://doi.org/10.1109/AERO.2007.353089
25. Talbi EG. *Metaheuristics : From Design to Implementation*. J. Wiley & Sons: Hoboken (N.J.), 2009.
26. Holland JH. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press: Cambridge, MA, USA, 1992.
27. NASA: A-Train, 10.26.10 (2017, April 25). Retrieved from http://atrain.nasa.gov/.
28. Fraire J, Finochietto J. Routing-aware fair contact plan design for predictable delay tolerant networks. *Elsevier Ad-Hoc Networks*. Volume 25, Part B, 2015, pp. 303-313, https://doi.org/10.1016/j.adhoc.2014.07.006.

Juan A. Fraire received the Telecommunications Engineering degree at the Instituto Universitario Aeronico (IUA) and his PhD grade in Engineering and Applied Sciences at the Universidad Nacional de Cba (UNC). His main focus of research is the implementation and optimization of communication algorithms and protocol solutions for near-Earth and interplanetary space networks. He coauthored more than 30 leading papers published in journals and international conferences. Since 2014, Juan is the chair of the annual Space-Terrestrial Internetworking (STINT) workshop and is currently an invited researcher at TIMA laboratories (Grenoble, France)

Pablo G. Madoery received the Telecommunications Engineering degree from the Instituto Universitario Aeronáutico, Argentina, in 2012. He has worked at the Digital Communication Laboratory of the National University of Córdoba in the field of satellite communications, and currently, he is finishing a PhD in Engineering Sciences focusing his interest in models, algorithms, and protocols for delay and disruption-tolerant networks.

Jorge M. Finochietto is a full professor at Universidad Nacional de Córdoba, Argentina (UNC) and senior researcher at the National Research Council (CONICET) of Argentina. Dr Finochietto has been involved in several national and international research projects in the fields of communication networks. He has coauthored over 50 papers published in international journals and presented in leading international conferences.

How to cite this article: Madoery P, Fraire J, Finochietto J. Congestion management techniques for disruption-tolerant satellite networks. *Int J Satell Commun Network*. 2018;36:165–178. https://doi.org/10.1002/sat.1210