Ad Hoc Networks 80 (2018) 31–40

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

On route table computation strategies in Delay-Tolerant Satellite Networks

Juan A. Fraire[a,*], Pablo G. Madoery[a], Amir Charif[b], Jorge M. Finochietto[a]

[a] IDIT, Universidad Nacional de Córdoba, CONICET, Córdoba, Argentina
[b] Computing and Design Environment Laboratory, CEA LIST, Gif-sur-Yvette, F-91191, France

ARTICLE INFO

Article history:
Received 1 December 2017
Revised 7 May 2018
Accepted 2 July 2018
Available online 3 July 2018

Keywords:
Delay-Tolerant Satellite Networks
Contact Graph Routing
Schedule Aware Bundle Routing

ABSTRACT

Delay-Tolerant Networking (DTN) has been proposed for satellite networks with no expectation of continuous or instantaneous end-to-end connectivity, which are known as Delay-Tolerant Satellite Networks (DTSNs). Path computation over large and highly-dynamic yet predictable topologies of such networks requires complex algorithms such as Contact Graph Routing (CGR) to calculate route tables, which can become extremely large and limit forwarding performance if all possible routes are considered. In this work, we discuss these issues in the context of CGR and propose alternatives to the existing route computation scheme: *first-ending, first-depleted, one-route*, and *per-neighbor* strategies. Simulation results over realistic DTSN constellation scenarios show that network flow metrics and overall calculation effort can be significantly improved by adopting these novel route table computation strategies.

1. Introduction

There is an increasing interest of the space community (commercial, civilian and military) in deploying large-scale satellite networks with the purpose of providing high quality imagery, video and communication services [1,2]. This trend has pushed for more efficient space-terrestrial communication techniques and technologies capable of successfully moving large volumes of data between space and ground networks. In this context, Delay- Tolerant Networking (DTN) has been identified as a disruptive approach which can meet this goal in a cost-effective way by means of loose communication requirements.

Inspired in deep-space applications, the DTN architecture [3] assumes no upper bound on the signal propagation delay nor an expectation of a continuous nor bidirectional end-to-end connectivity through the network. This certainly differs from traditional Internet-based networks where end-to-end connectivity is generally assumed stable enough to pass data from source to destination. Since no instantaneous end-to-end feedback can be assumed, data might be temporarily stored (i.e., delayed) at intermediate nodes [4,5]. To determine when to store or when to forward data to a given neighbor, existing DTN forwarding schemes have sought to acquire the best possible knowledge of the network [6]. When applied in space, episodes of communications in DTN can be precisely computed in advance based on orbital elements. Also, power-conserving spacecrafts may communicate on infrequent, fixed intervals established by configuration. In any case, forthcoming episodes of communications (a.k.a. *contacts*) are typically scheduled weeks or months before they occur and can be imprinted in a *contact plan*. The resulting contact plan can be either distributed in advance to DTN nodes, or used by a centralized node (i.e., mission control) to execute route determination procedures.

A DTN paradigm can indeed be used to forward data on near-Earth satellite networks with sporadic satellite-to-satellite and satellite-to-ground communication opportunities. If so, we define them as Delay-Tolerant Satellite Networks (DTSNs). DTSNs differ from other space DTNs in the size of the topology and the speed at which it changes. In particular, interplanetary networks are rather scarce in terms of spacecrafts as a few rovers on a remote planet plus some orbiters are typically assumed in the literature [7]. While this density of deep-space nodes is unlikely to change in the near future, DTSNs topologies are expected to be promptly based on dozens or even hundreds of satellites [1]. Furthermore, while in interplanetary DTNs the topological changes are dictated by planetary dynamics, communication opportunities in DTSNs typically occur much more frequently between satellites in Low-Earth Orbit (LEO). As a result, the scalability limits of current DTN protocols and algorithms are likely to be met sooner in DTSN than in deep-space applications. Thus, DTSNs become an immediate object of study for evaluating efficient routing strategies which will also, in the long term, be valuable in the interplanetary domain.

* Corresponding author.
E-mail addresses: juanfraire@unc.edu.ar, jfraire@sti-tech.com.ar (J.A. Fraire).

https://doi.org/10.1016/j.adhoc.2018.07.002
1570-8705/© 2018 Elsevier B.V. All rights reserved.

In order to run traditional network algorithms in DTNs, previous studies sought to derive suitable graph structures out of contact plans. Initial approaches translated the contact plan to *time-expanded graphs* [8]. In a time-expanded graph, topological changes are modeled by a succession of graphs each representing the connectivity of the whole network during an interval where it is considered stable. Although convenient to calculate all-to-all paths, algorithms based on such data structure scale poorly with time (require more graphs) and nodes quantity (require more nodes in all graphs). A not so intuitive, yet more efficient modeling was proposed under the name of *contact graph*, a suitable structure that facilitates the distributed execution of adapted Dijkstra's searches [9]. The convenience of contact graph models motivated the study [7], implementation [10] and flight-validation [11] of early versions of distributed Contact Graph Routing (CGR) algorithms. Also, a reference version of CGR has been seamlessly integrated with Bundle Protocol [12] (DTN standard protocol) in the Interplanetary Overlay Network (ION) stack developed by JPL (NASA) [10]. ION is currently on version 3.6 and is operative in the International Space Station [13], and its CGR statement is being considered for the Schedule Aware Bundle Routing (SABR) recommendation book at the Consultative Committee for Space Data Systems (CCSDS) [14].

In spite of the increasing interest of the space community in CGR route determination algorithms [15], only recently there has been attention to the computation strategies that are used in CGR to populate its route table [16]. Indeed, since the route table computation method is responsible for triggering and setting subsequent CGR calls parameters (i.e., current network status), it has a direct impact on the final network flow and overall route processing effort. Previous works have sought to construct static route tables (left unchanged throughout the contact plan duration) similarly to those used in Internet [10]. As a first contribution, this work analyzes existing techniques, discusses their weaknesses and introduces *first-ending* and *first-depleted* which improve the construction of complete and accurate route tables. Next, we discuss that different approaches can be proposed to cope and to better adapt to the large and rapidly changing topologies of DTSNs. To validate this hypothesis, we also introduce *one-route* and *per-neighbor* methods which are presented as novel alternatives to dynamically update route tables on-demand, minimizing calculation effort without sacrificing data-forwarding efficiency. Finally, the resulting performance of each of these techniques are compared with the reference CGR implementation in ION v3.6 and analyzed by means of simulations over two appealing Low-Earth-Orbit (LEO) DTSN constellations. At the time of writing, the solutions explored in this paper are being rolled out in ION v3.7 and included in CCSDS's SABR specification.

This paper is structured as follows. In Section 2 relevant CGR concepts are defined and overviewed. Section 3 describes current route table calculation limitations and propose novel methodologies. Performance analysis by means of simulations are provided in Section 4 and discussed in Section 5. Final conclusions are drawn in Section 6.

2. Contact Graph Routing

Routing in DTNs is typically divided in three stages: planning (performed by mission control on ground), routing (typically distributed but can be centralized on ground as well) and forwarding (executed locally by each DTN node). In the planning stage, contact plans are determined based on the estimation of future episodes of communications. This task involves the physical disposition and orientation of nodes as well as their communication system configuration (antenna, modulation, transmission power, etc.). As a result, orbital propagators and communication models are combined to determine the final contact plan which can be further tuned to reduce energy consumption or remove conflicting contacts [17]. Whether kept in a centralized planning node or distributed to all nodes, the contact plan is then used by algorithms such as CGR to derive efficient routes. The forwarding process is then responsible for selecting the best route, out of many available on the route table, when local or in-transit data need to be queued for transmission. In DTN, data units are known as bundles. Then, a bundle can be either instantaneously transmitted or enqueued until a contact with the next hop node takes place.

CGR takes as input a contact plan, a data structure comprised of a series of contacts. A contact $C_{A,B}^{t_1,t_2}$ is defined as a time interval $(t_1; t_2)$ during which it is expected that data will be transmitted by node A (the contact's sending node) and received by node B (the contact's receiving node)[1]. In Fig. 1(a), each contact is identified by a number (#1 ... 16) and characterized by its start time, its end time, the identities of the sending and receiving nodes, and the rate at which data is expected to be transmitted by the sending node throughout the indicated time period. Furthermore, each contact is characterized by an approximate range value between nodes A and B expressed in light seconds. Since contacts are unidirectional, a pair is needed to describe a bidirectional link. In Fig. 1, contacts $C_{A,B}^{0,60}$, $C_{B,C}^{0,60}$ and $C_{A,C}^{0,60}$ represent permanent links (e.g., between ground stations connected through Internet). Contacts $C_{C,D}^{0,30}$ and $C_{A,E}^{10,20}$ might stand for sporadic Ground to Space Links (GSLs) while $C_{D,E}^{0,10}$, $C_{D,E}^{30,40}$ and $C_{D,E}^{50,60}$ for opportunistic Inter-Satellite Links (ISLs). The data volume of a contact is given by the product of its duration and its data transmission rate. Thus, a contact plan captures the time-evolving nature of a dynamic topology which can also be presented in a static graph as in Fig. 1(b) or in a time line view as in Fig. 1(c).

To compute data paths, CGR creates a contact graph model based on the contact plan. Specifically, a contact graph for destination node D at source node S is a conceptual directed acyclic graph $CG_S^D = (V, E)$ where vertices V correspond to contacts $C_{A,B}^{t_1,t_2}$ in the contact plan while edges E can be seen as episodes of data retention at a node i. Fig. 2 illustrates the CG_A^E based on the contact plan example of Fig. 1. Indeed, the structure of the contact graph may seem somewhat counterintuitive as it bears almost no relation to the topology of the network as illustrated in Fig. 1(b). In return, this static graph representation facilitates the execution of network algorithms over time-evolving networks. Specifically, a contact graph is formed by one vertex for each contact in the contact plan that signifies transmission either directly or indirectly (i.e., through other contacts) from A to node E. Edges are then added between contacts where destination and source nodes correspond (i.e., the receiving node of a contact matches the source node of the next contact in the path). In the example of Fig. 2, the receiving node of contact 1 ($C_{A,B}^{0,60}$) is the same as the transmission node of contact 3 ($C_{B,C}^{0,60}$). An edge between them represents a temporal storage in the connecting node which could be 0 when contacts are overlapped in time, meaning a direct transmission is possible. Finally, notional contacts from node A to itself and from node E to itself (a.k.a. root and terminal contacts) are also included as part of the contact graph. As discussed in [7] and detailed in [9], adapted Dijkstra's searches can be used to determine optimal routes over contact graphs.

Reference CGR implementation in ION 3.6 stores resulting routes in *route tables*. A routing table is a list of *route lists*, one route list for every other node in the network that is cited in any contact in the contact plan. A route list can be of size 0 (no routes

[1] A contact is specifically not an episode of activity on a link. Episodes of activity on different links (e.g., different radio transponders operating on the same spacecraft) may well overlap, but contacts by definition cannot. Therefore, all concurrent links should be considered together in a single contact.