DTN for LEO Satellite Communications

Carlo Caini and Rosario Firrincieli

DEIS/ARCES, University of Bologna, Bologna, Italy
{carlo.caini,rosario.firrincieli}@unibo.it

Abstract. Satellite communications are an interesting and promising application field for Delay/Disruption Tolerant Networking (DTN). Although primarily conceived for deep space communications and sensor networks, it was immediately recognized that DTN was applicable to satellite environments, in particular to cope with the intermittent channels typical of LEO (Low Earth Orbit) constellation satellite systems. The aim of this paper is to assess the advantages of DTN when applied to LEO satellites. Qualitative assessments are supported in selected cases by preliminary results obtained on a testbed based on GNU/Linux machines. In particular, two application scenarios have been considered, both using a single LEO satellite. In the former, we have one LEO satellite for Earth observation, connected to its gateway stations only at intermittent scheduled intervals due to its orbital motion. The latter is one LEO satellite acting as a "data mule" between a terrestrial sensor network and a remote satellite gateway station, which are never in the satellite coverage area at the same time. The results show the feasibility and the advantages of DTN in LEO satellite communications.

Keywords: DTN, LEO satellites, Satellite communications, Challenged networks, DTN2, ION.

1 Introduction

Delay/Disruption Tolerant Networking (DTN) aims to provide interoperable communications in "challenged networks", i.e. those networks where one or more of the usual assumptions implicit in the use of the TCP/IP stack (short delays, negligible PER, existence of a continuous path between source and destination), no longer hold true. Such networks include deep space communications, a large variety of terrestrial and maritime sensor networks, satellite and airborne communications e.g. Unmanned Aerial Vehicles (UAVs) and many other in both the civil and military fields [1]-[7].

Concerning satellite communications, DTN represents an interesting alternative to the use of PEPs (Performance Enhancing Proxies) in GEO (Geosynchronous Earth Orbit) satellite systems, as shown in [8], but its use is particularly appealing in LEO (Low Earth Orbit) systems, because of DTN ability to cope with intermittent channels, disruption and lack of end-to-end connectivity, typical of both single LEO satellites and incomplete constellations [6], [9]. Hence, this paper aims to evaluate the

G. Giambene and C. Sacchi (Eds.): PSATS 2011, LNICST 71, pp. 186–198, 2011.

advantages of DTN when applied to LEO satellites, supporting our assessments with results and logs obtained on a real testbed, i.e. on DTN implementations running on GNU/Linux machines.

In the tests we considered two possible applications, both using a single LEO satellite. In the first, a LEO satellite for Earth observation is connected to its gateway stations only at intermittent scheduled intervals. In the second, a single LEO satellite acts as "data mule" between a terrestrial sensor network and a remote control centre. In both cases, to cope with intermittent channel availability (and also the lack of a continuous path in the latter), file transfers using TCP/IP stack would require manual intervention. By contrast, as shown in the paper, file transfers can be performed automatically by DTN even in these challenging scenarios.

2 DTN Outline

2.1 Origin and Motivation

DTN was first conceived to address space communications impairments, as it was glaringly obvious that the usual TCP/IP stack, alone, could not cope [1]. Later, DTN scope was enlarged to cover all challenged networks, whether spatial or terrestrial. To this end, in 2002 the IRTF Delay Tolerant Networks Research Group (DTNRG) was established to promote DTN. As the new architecture must tolerate not only long delays, but also link disruptions, the DTN acronym is often expanded as Delay/Disruption Tolerant Networking. The interested reader is referred to [2] for an informative study of TCP limits in challenged networks, and to [3] and [4] for an exhaustive survey of DTN development. Tutorials and other references can be found on the DTNRG website [5], which is the major source of DTN documentation and software. Although the DTN architecture based on the introduction of the Bundle protocol, described in [6], and [7], is not the sole possible option, it is the most common and we will refer to it in this paper.

2.2 DTN Bundle Protocol Architecture

In order to support communication in challenged environments, the Bundle protocol DTN architecture [6], [7] is based on a new layer, located between Transport and Application, called "Bundle layer". The related protocol (the Bundle protocol) can interface with various transport protocols (including TCP [10] and UDP, but also with new protocols, like Licklider [11], [12] or Saratoga [13]), through "convergence layer adapters". In this new architecture (see Fig. 1), transport protocol end-to-end features are confined to homogeneous network segments (A, B and C), while end-to-end data transfer across the heterogeneous network is provided by the bundle layer; large data packets called "bundles" are exchanged between DTN nodes through a store-and-forward relay. The main innovations of DTN architecture are summarized below.

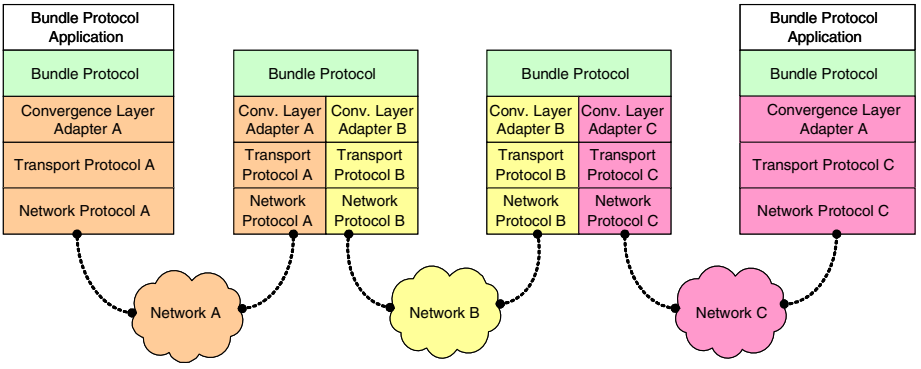

Fig. 1. DTN architecture and protocol stack

2.3 DTN Overlay

First, although TCP/IP protocols are not necessarily replaced, their role is changed. By installing the Bundle protocol on end-points and some intermediate nodes, (e.g. on satellite gateways), the end-to-end path is divided into multiple DTN hops. On each DTN hop a different protocol stack can be used, or, when the same stack is retained, which is the most common case, just different protocols, like TCP, UDP, or different versions of the same protocol (e.g. different TCP variants). Readers familiar with satellite PEPs [14], [15] can easily realize that the DTN multi hop architecture can be seen as a generalization of the TCP splitting concept. In particular, both DTN and TCP splitting PEPs allow the use of specific protocols (or specific versions of the same protocol) on the satellite segment. In such a way, the same advantages of TCP splitting PEPs can be achieved, in terms of goodput, also by DTN [8]. However, it must be stressed that while in the DTN architecture the "splitting" is a direct consequence of the new protocol stack, in PEPs it implies a severe violation of the end-to-end TCP semantics. Concerning security, TCP splitting is incompatible with IPsec, while the DTN architecture has the advantage of a greater flexibility (both end-to-end and hop-by-hop security can be provided). On the other hand, by contrast to PEPs, the DTN architecture is not transparent to end nodes.

2.4 Storage at Intermediate Nodes

The second difference between DTN and customary TCP/IP network is related to information storage. In standard networks, because of usual assumptions of continuous connectivity and short delays, information is supposed to be stored only at end nodes. By contrast, in DTN networks, where the usual assumptions do not hold anymore, information (i.e. data bundles) must be stored at intermediate DTN nodes for long period of times and, when the custody option [6], [7] is enabled, only on persistent memory (e.g. on local hard disks). This feature actually differentiates DTN architecture from usual PEPs, as it makes DTN much more robust against disruptions, disconnections, and temporary node failures [16].

2.5 Bundle Fragmentation

An interesting feature of DTN Bundle protocol is the possibility of fragmenting bundles. Fragmentation can be performed a priori (proactive fragmentation) or a posteriori (reactive fragmentation). The former has been conceived to cope with intermittent periodic connectivity, where there may be a stringent constraint on the maximum amount of data that can be transferred (contact volume) on a DTN hop at each availability time window (contact time). It allows large bundles to be divided "a priori" into multiple fragments compatible with the contact volume. This feature could be useful in single LEO satellite systems, where the contact volume is known in advance. By contrast, reactive fragmentation works a posteriori, triggered by a relatively long disruption. It could be advantageous in satellite communications (both GEO and LEO) with mobile terminals, when obstacles (buildings, tunnels, etc.) may disrupt the satellite signal reception.

3 LEO Satellite Communications and DTN

LEO satellites are characterized by low orbits with a reduced distance from the Earth's surface (160 – 2000 km). Compared to GEO systems they offer the obvious advantage of reduced attenuation loss and a shorter propagation delay. On the other hand, to an observer on the Earth's surface they do not appear fixed in the sky, but fast and constantly moving; for example, at an altitude of 520 km the revolution period necessary to counteract the Earth's gravity is about ninety minutes. As a result, a single satellite can only provide intermittent connectivity with a fixed ground station, while continuous connectivity can be provided only by constellations of several tens of satellites, like those used in Iridium [17] or Globalstar [18], the two most well-known commercial systems. Because of their different implications, we will treat single and multiple satellite coverage separately.

3.1 Single Satellite Coverage

In the case of a single satellite we further distinguish between two possible applications. The first, a data transfer from a LEO satellite to a remote control centre; the second, a "data mule" data collection from a sensor network.

3.2 Earth Observation Scenario

Due to their low orbits, LEO satellites pass over a fixed ground station for short intervals (some minutes) many times a day, thus providing scheduled intermittent connectivity. In this scenario we consider data transmission from a LEO satellite to a terrestrial destination. For instance, a LEO satellite devoted to Earth observation which has to transfer large image files to a remote operation control centre. Here, we have to cope with intermittent scheduled end-to-end connectivity because of satellite motion. The short transmission time window and the possible limited channel bandwidth pose limits to "contact volume", i.e. the total amount of data that can be transferred at each link availability interval. Image files larger than contact volume cannot be transferred during a single pass, and require to be divided into multiple

segments for transmission during consecutive passes. In this case DTN could benefit from the "proactive fragmentation" feature of the Bundle protocol. Alternatively, if this feature is not available in the implementation in use (or to avoid the complexity that can derive from the concurrent use of fragmentation and Bundle protocol security extensions), it is possible to use a DTN application, like DTNperf, which can segment a file into a series of bundles of the desired dimension. Further details will be provided in the numerical results section.

3.3 Data Mule Scenario

Here, we consider a source and a sink both located on Earth and connected through two ground stations and a LEO satellite. The two ground stations are a long distance apart, so are never concurrently in line of sight from the satellite. Consequently, there is not a continuous path between them and the LEO sat must act as a data mule. For example, imagine a remote sensor network connected to its control centre via satellite. Data must be first collected at a central node of the sensor network, with DTN and satellite capabilities (the first ground station); the data are then to be transferred to the second ground station. The LEO satellite is alternately in line of sight from one or other ground station, and data transfer can only performed by storing data on the satellite, which must therefore have adequate storage capacity. Note that as this scenario is the most challenging, it is also the most favorable to DTN. The total absence of end-to-end connectivity prevents the establishment of TCP (or TCP-like) connections, while it is perfectly suited to the DTN "store-and-forward" approach.

3.4 Multi Satellite Coverage

Unlike GEO systems, where one satellite can offer continuous coverage of a large area, with LEO systems continuous connectivity requires the deployment of a constellation of satellites (50-70). First generation LEO systems like Globalstar and Iridium, designed in the early '90s and still in use, were primarily designed to provide voice communication and can offer only secondary data capabilities at low bit rates (max 128 kbit/s for the Iridium system). They are soon to be replaced by second-generation systems designed mainly for data communications and Internet access. The present Iridium system, for example, will be replaced by "Iridium Next", which will make use of the same orbits and number of satellites, but its enhanced payloads will be able to offer data communications at various rates up to 8 Mbit/s. Deployment of this second generation will require the launch of 66 active satellites (plus some spares), and is expected to take a couple of years.

Until a LEO constellation is fully deployed, it is difficult to make use of the satellites already in orbit, because gaps in the, moving, coverage area cause intermittent connectivity. DTN could cope with this problem, thus enabling the first satellites deployed to enter into operation, with obvious economic advantage. For example, incomplete constellations could be used for file transfers and non real-time data exchange, thanks to the DTN ability to function despite intermittent connectivity and disruption. Moreover, even after complete constellation deployment, DTN could still offer significant advantages. It could, for example, counteract link disruptions frequently met when using mobile terminals, or remedy the possible temporary lack in

free channels during handovers between satellites (in LEO system handover are necessary even for fixed terminal due to the satellite motion).

DTN use in LEO constellations will be the object of future research, and will therefore not further treated in the numerical result sections.

4 DTN Implementations and Tools

4.1 DTN2: Bundle Protocol Reference Implementation

DTN2 is the Delay Tolerant Networking reference implementation. In addition to the reference Bundle protocol implementation, the DTN2 package also contains some DTN basic applications (DTNping, DTNsend, etc.) and DTNperf_2, the DTN evaluation tools used in our experiments and described below. The DTN2 goal is twofold: "to clearly embody the components of DTN architecture, while also providing a robust and flexible software framework for experimentation, extension, and real-world deployment" [5]. In other words, DTN2 aims to be suitable for both study and real use. It runs on Linux (x64 and x86) and other platforms as well. DTN2 can be downloaded from Source Forge (see [5]). The latest release is 2.7. Installation is complex, but configuration is relatively simple, being based on one configuration file for each DTN node. To enable DTN capabilities, it is enough to launch DTN2 as a daemon, which can be done at boot time. Once launched, all users can easily start DTN applications, like DTNperf, on top of it.

4.2 ION: NASA Bundle Protocol Implementation

ION (Interplanetary Overlay Network) is an implementation of the Bundle protocol developed by NASA JPL (Jet Propulsion Laboratory), with the contributions of Ohio and other Universities, and explicitly focused on deep space applications [20]. As in these environments TCP cannot be used because of excessive RTTs, ION distribution also contains an implementation of Licklider Transport Protocol (LTP), which was designed to offer reliable service in environments characterized by very long delays, and can be suitably used as convergence layer in DTN architecture [11], [12]. Although some features, like DTN node naming, have been specifically designed for space applications, ION software can be used in other environments as well. Moreover, it offers a good interoperability with DTN2 nodes. ION is written in C and currently runs on various Linux platforms, OS/X, FreeBSD, Solaris, VxWorks, and RTEMS.

The ION source code is available as open source from the Open Channel Foundation [21]. The latest release is 2.3 and includes implementations of Contact Graph Routing and several convergence-layer adapters, including TCPCL (interoperable with DTN2), UDPCL (likewise interoperable with DTN2) and LTPCL. ION configuration and use appears somewhat more complex than DTN2; however, it offers some features of particular interest here, like intermittent links, which have not yet been implemented in DTN2. It should be noted, however, that scheduled links in ION require the use of LTP at convergence layer. Moreover, ION offers limited support of bundle fragmentation.

4.3 DTNperf_2

DTNperf is a client-server evaluation tool designed to assess goodput and to provide logs in DTN bundle layer architectures [22]. It is named after the famous Iperf application, widely used to test TCP and UDP performance in ordinary (i.e. non DTN) networks, and it is included in the official DTN2 package released by DTNRG. As DTNperf versions 2.x are significantly improved with respect to previous 1.x versions, they are called DTNperf_2, to stress this difference. The latest DTNperf_2 versions are available for downloading from the "bleeding edge" DTN2 version using Mercurial [23]. They are under an open-source license (Apache License 2.0).

DTNperf is intended to complement other debugging and testing tools included in DTN2, like DTNping (the DTN equivalent of "ping"), DTNsend and DTNrecv (to create, send and receive one bundle), or basic applications, like DTNcat (to send standard input data to another DTN node) or DTNcp (to copy a file between DTN nodes). By contrast, however, and like Iperf, DTNperf is focused on performance evaluation in terms of goodput. Moreover, it allows the user to easily collect the informative DTN "status reports" sent by DTN nodes, (i.e. sent, forwarded, received, custody accepted, delivered, deleted, etc), which are essential in the study of bundle transmission on complex DTN networks.

DTNperf_2 is written in C language, to maintain full compatibility with the DTN2 bundle layer reference implementation APIs. A version also compatible also with ION is envisaged but at present has not been developed. A distinctive feature of DTNperf_2 is examined in detail below because of its relevance in our tests.

4.4 DTNperf_2 Transmission Window

The first release of DTNperf, like other DTN tools, did not allow the source to send more than one bundle at a time, i.e. it was necessary to wait for the reception of an "acknowledgment" of the bundle sent before starting the transmission of a new bundle. This resulted in an obvious goodput ceiling of one bundle per RTT, and a less obvious additional delay for each intermediate DTN nodes due to the store and forward transmission mechanism. To overcome these limitations, which had a significant impact on goodput [8], DTNperf_2 introduced a transmission window that allows multiple bundle transmissions. The length of the Tx window, W, represents the maximum number of bundles that can be concurrently in-flight (i.e. sent but not acknowledged yet). By default, bundle acknowledgments are represented by the "delivery status report" [6], [7], ("status delivered", in short) sent by the receiver node. The DTNperf_2 transmission window is similar to TCP transmission window [8], with the difference that in-flight bundles can be non-consecutive to cope with the non-ordered delivery of the bundle protocol.

5 Numerical Results

In this section, experimental results obtained according to the scenarios presented in 3.2 and 3.3 are discussed. The experiments were carried out by means of a DTN testbed consisting of five GNU/Linux OS machines, with either DTN2 or ION installed. The rationale for the concurrent use in the same testbed of two Bundle

protocol implementations, although on different machines, is to take advantage of both advanced DTNperf_2 features, including multiple bundle transmission (W>1), bundle logs, and bundle reordering, and ION link management capabilities, such as scheduled links and transmission speed regulation. General assumptions and scenario characteristics are summarized in Table 1.

Table 1. General assumptions and scenario characteristics

Characteristic	Value
LEO-ground station link type	Intermittent (10 min every 100 min); first contact 5 min after transfer start (Earth observation case). Intermittent (10 min every 50 min); first contact 5 min after transfer start (data mule case).
Ground stations-other terrestrial nodes link type	Wired link, always available, 100 Mbit/s, negligible delays.
LEO-ground station RTT	130 ms
LEO-ground station Bandwidth	1 Mbit/s (symmetric)
LEO-ground station PER	Not present
Number of ground stations	1 (Earth observation case) 2 (data mule case)
Number of total contacts between LEO and ground stations	2 (Earth observation case) 3 (data mule case)
Max contact volume	75 MB
File to transfer	80 MB (Earth observation case) 20 MB (data mule case)
Bundle size	200 kB
Bundle number	400 (Earth observation case) 100 (data mule case)
DTNperf_2 transmission window, W	200 (Earth observation case) 100 (data mule case)
Custody option	ON (all nodes)

5.1 Earth Observation Scenario

Here we assume that the LEO satellite takes images of Earth and, as soon as passes over the ground station, sends them toward the control center (Fig. 2-a). The corresponding testbed topology is shown in Fig. 3. It is worth noting that the LEO satellite has both a DTN2 and an ION node on board. The first acts as DTNperf_2 source (client), while the second is necessary to establish an LTP scheduled link with the ground station. Note that the use of LTP on scheduled links is mandatory in ION.

According to Table 1, maximum contact volume on the LEO-ground station link, obtained using full-speed transfer for the entire contact time, is 75 Mbyte. Depending on file length, file transfer can be completed in one or more passes. In the first case the transfer is quite simple and can be completed as soon as the LEO satellite comes into line of sight with its ground station. The second case is more interesting, and

therefore is the sole considered here. Assuming an 80 MB file transfer, two satellite passes are necessary. At the control center, the DTNperf_2 server application, running on a DTN2 machine, has to reassemble the transmitted file by collecting and reordering all arriving bundles.

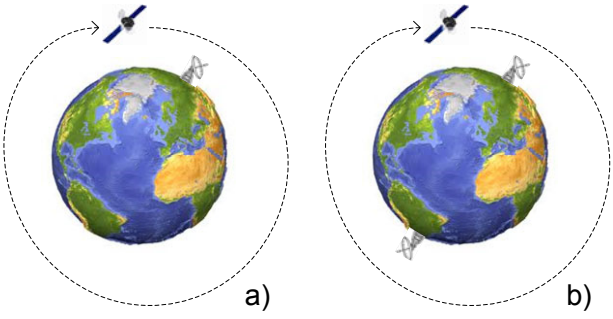

Fig. 2. Experimental cases: a) Earth observation, b) data mule

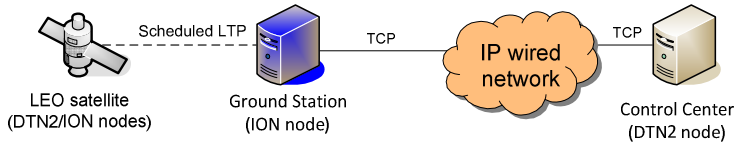

Fig. 3. Earth observation topology

Some details of bundle transfer are given in Fig. 4, taken from DTNperf_2 logs. At time zero the first bundles are transferred by the DTNperf_2 client to the Bundle protocol of the source DTN2 node and from here to the ION node inside the LEO satellite, where are taken into custody waiting for satellite link availability (first part of the "SENT/Custody on LEO" series). In order not to exceed the ION node storage limits (about 60 MB), we used a W=200 DTNperf_2 transmission window, which limits to 40 MB (half of the file) the amount of data to be stored on the ION node. As soon as the LEO-ground station link becomes available (at 300 s from time zero), bundles start to be progressively transferred to the ground station (at 1 Mbit/s) and from there to the control center (at 100 Mbit/s). Bundle deliveries are immediately confirmed to the DTNperf_2 client on the source by "delivered" status reports. At the sender side, the arrival of each status report ("DELIVERED ACK" series in the chart) triggers a corresponding sliding of the DTNperf_2 transmission window, thus allowing the remaining 200 bundles to be progressively sent and then taken into custody by the ION node inside the LEO satellite (second part of the "SENT/Custody on LEO" series). When the LEO link closes (at 900 s) all the bundles (400) have been sent by the source and taken into custody by the LEO ION node, but only a part (344, i.e. 68.8 MB) have actually been transferred to the control center as yet. Consequently, it is necessary to wait for the second contact (at 6300 s) to transmit the bundles still in custody (56, i.e. 11.2 MB) and complete the file transfer. LEO link

availability is highlighted through horizontal segments in the figure. As a final remark, note that the transmission of 68.8 MB on the first pass, given a theoretical contact volume of 75 MB, is an excellent result, as link utilization efficiency is greater than 0.9.

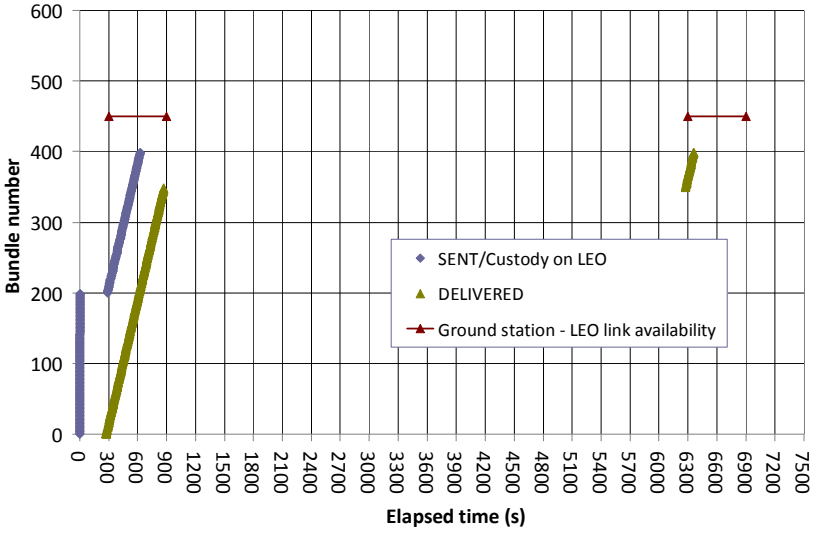

Fig. 4. Earth observation: bundle transmission logs

5.2 Data Mule Scenario

Here (Fig. 2-b) a 20 MB data transfer from two terrestrial nodes connected via one LEO satellite is considered. The LEO satellite forwards data from the first ground station to the second, alternately in line of sight with the satellite. The corresponding testbed topology is shown in Fig. 5.

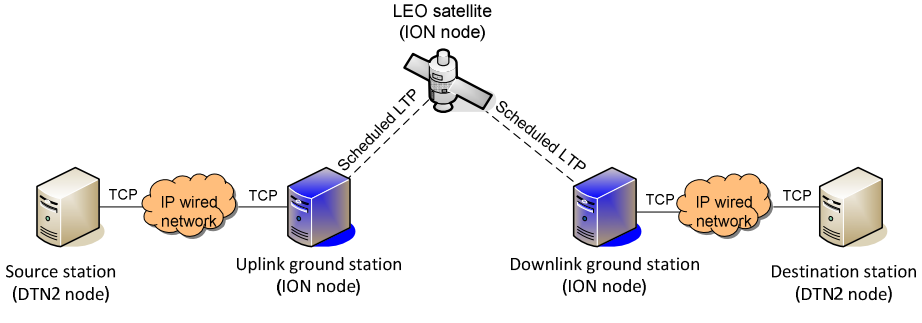

Fig. 5. Data mule topology

The file dimension has been assumed here lower than the maximum contact volume (75 MB as before, Table 1), which allows the file to be transferred in a single pass. Therefore, when LEO passes over the first ground station it is able to get the entire file; then, as soon as it is in line of sight with the second ground station, the file is transferred entirely toward the destination station. A second and last contact with the first ground station has the sole purpose of transmitting bundle acknowledgments (i.e., "delivered" status reports) to the source station running the DTNperf_2 client. As in the previous case, in the destination station the DTNperf_2 server application reassembles the transmitted file by collecting and reordering the arriving bundles.

Bundle transfer is illustrated in Fig. 6. At time zero all bundles are immediately transferred by the DTNperf_2 client to the source station bundle layer ("SENT" series) and from here to the first ground station, where they wait in custody for the next satellite link contact. "Custody" status reports generated by both the source and the first ground station are not shown, as they would overlap the "SENT" series. When the LEO satellite passes over the first ground station (300 s after time zero), bundles are progressively transferred on board (at 1 Mbit/s) and taken into custody ("Custody on LEO" series). The transfer time is about 160 s. When the satellite comes into line of sight with the second ground station (at 3300 s), bundles are downloaded (at 1 Mbit/s) and transferred at high speed (100 Mbit/s) to destination, which, in turns, sends back "delivered" status reports. The "DELIVERED" series in the chart represents here the time at which bundles are actually delivered (this information is contained in the "timestamp" field of "delivered" status reports). On their way back, the "delivered" reports have to stay on board the satellite until the next pass on the first ground station (at 6300 s), when they can finally be transferred to the source ("DELIVERED ACK" series). Their transfer time is almost instantaneous (vertical slope in the chart) because status reports consist of only few tents of bytes.

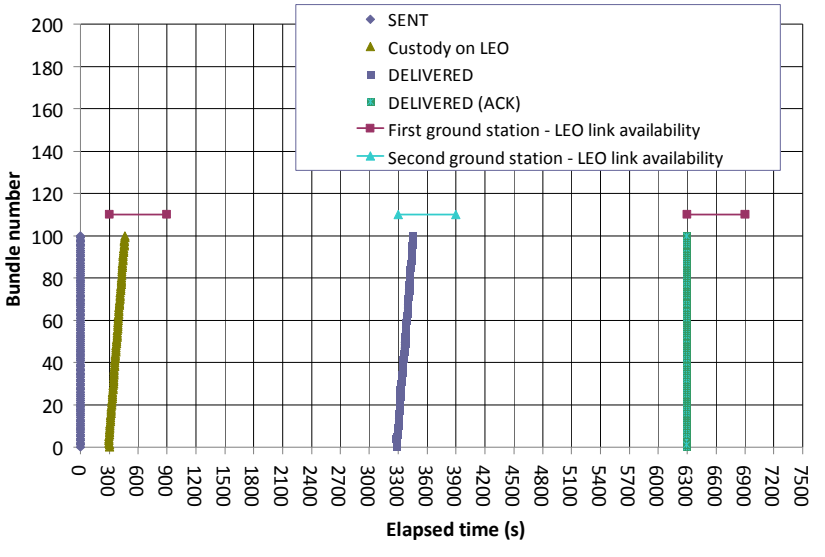

Fig. 6. Data mule: bundle transmission logs

6 Conclusions

In this paper the advantages of DTN when applied to LEO satellite communications have been assessed, considering both single satellites and constellations. In the former case, some preliminary results, obtained on a real testbed based on GNU/Linux machines are discussed. The tests were performed using both DTN2 and ION Bundle protocol implementations, and the DTNperf_2 evaluation tool. In both the applications considered, namely Earth observation and data mule communications, the results show the feasibility and the advantages of DTN in LEO satellite communications. In fact, both cases are characterized by intermittent connectivity on scheduled intervals, a challenge that would prevent the use of ordinary file transfer protocols and TCP/IP stack, but which is effectively tackled by DTN, as shown in the paper.

References

1. Burleigh, S., Hooke, A., Torgerson, L., Fall, K., Cerf, V., Durst, B., Scott, K., Weiss, H.: Delay-tolerant networking: an approach to interplanetary Internet. IEEE Communications Magazine 41(6), 128–136 (2003)
2. Farrell, S., Cahill, V., Geraghty, D., Humphreys, I., McDonald, P.: When TCP Breaks: Delay- and Disruption- Tolerant Networking. IEEE Internet Computing 10(4), 72–78 (2006)
3. McMahon, A., Farrell, S.: Delay- and Disruption-Tolerant Networking. IEEE Internet Computing 13(6), 82–87 (2009)
4. Fall, K., Farrell, S.: DTN: an architectural retrospective. IEEE Journal on Selected Areas in Commun. 26(5), 828–836 (2008)
5. DTNRG web site, http://www.dtnrg.org/wiki (last visited January 11, 2010)
6. Cerf, V., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., Weiss, H.: Delay-Tolerant Networking Architecture. IETF RFC 4838 (April 2007)
7. Scott, K., Burleigh, S.: Bundle Protocol Specification. IETF RFC 5050 (November 2007)
8. Caini, C., Cornice, P., Firrincieli, R., Lacamera, D.: A DTN Approach to Satellite Communications. IEEE Journal on Selected Areas in Communications, special issue on Delay and Disruption Tolerant Wireless Communication 26(5), 820–827 (2008)
9. Ivancic, W., Eddy, W.M., Stewart, D., Wood, L., Northam, J., Jackson, C.: Experience with Delay-Tolerant Networking from Orbit. Int. J. of Satell. Commun. And Networking 28(5-6), 335–351 (2010)
10. Allman, M., Paxon, V., Stevens, W.: TCP Congestion Control. IETF RFC 5681 (September 2009)
11. Burleigh, S., Ramadas, M., Farrell, S.: Licklider Transmission Protocol —Motivation. IETF RFC 5325 (September 2008)
12. Ramadas, M., Burleigh, S., Farrell, S.: Licklider Transmission Protocol —Specification. IETF RFC 5326 (September 2008)
13. Wood, L., McKim, J., Eddy, W., Ivancic, W., Jackson, C.: Using Saratoga with a Bundle Agent as a Convergence Layer for Delay-Tolerant Networking. IETF Internet draft, work in progress, http://tools.ietf.org/id/draft-wood-dtnrg-saratoga (last visited January 11, 2010)

14. Border, J., Kojo, M., Griner, J., Montenegro, G., Shelby, Z.: Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations. IETF RFC 3135 (June 2001)
15. ETSI TR 102 676: Satellite Earth Stations and Systems (SES); Broadband Satellite Multimedia (BSM): Performance Enhancing Proxies (PEPs)
16. Caini, C., Firrincieli, R., Cruickshank, H., Marchese, M.: Satellite Communications: from PEPs to DTN. In: Proc. of ASMS 2010, Pula, Italy, pp. 62–67 (September 2010)
17. Iridium website, http://www.iridium.com (last visited January 11, 2010)
18. Globalstar website, http://www.globalstar.com (last visited January 11, 2010)
19. DTN2 Reference Implementation, http://www.dtnrg.org/wiki/Code (last visited January 11, 2010)
20. Burleigh, S.: Interplanetary Overlay Network (ION) an Implementation of the DTN Bundle Protocol. In: The Proc. of 4th IEEE Consumer Communications and Networking Conference, pp. 222–226 (2007)
21. ION code, http://www.openchannelfoundation.org/projects/ION, (last visited January 11, 2010)
22. Caini, C., Cornice, P., Firrincieli, R., Livini, M.: DTNperf_2: a Performance Evaluation tool for Delay/Disruption Tolerant Networking. In: Proc. of ICUMT 2009 (E-DTN session), St.-Petersburg, Russia, pp. 1–6 (October 2009)
23. DTNperf_2 source code: DTN2 Mercurial repository, http://dtn.hg.sourceforge.net/hgweb/dtn/DTN2 (last visited January 11, 2010)