Experiences from architecting a DTN Testbed

Efthymios Koutsogiannis, Diamantopoulos Sotirios, Papastergiou Georgios, Komnios Ioannis, Aggelis Aggelis and Nestor Peccia

***Abstract*— As the number of space elements increases, space communications enter a new era, where internetworking gradually replaces traditional telecommunication protocols. Delay Tolerant Networking (DTN) has emerged as the most prominent solution for such challenged networks; however, extensive testing and evaluation of all mechanisms is required prior to the wide deployment of DTN. In this paper, we present our experiences from architecting a state-of-the-art DTN testbed, capable of emulating future space communications and appropriate to evaluate proposed protocols and mechanisms. Several evaluation results have been drawn so far, including reliability (CFDP over LTP), routing (Contact Graph Routing versus other DTN routing protocols) and interoperability issues (communication between DTN and Space Packet nodes).**

***Index Terms*– Delay Tolerant Networking, Testbed, Space communications, CFDP**

I. INTRODUCTION

Scope of this paper is to provide insight in the development and implementation of a Delay Tolerant Networking (DTN) [1] testbed specifically designed to evaluate the benefits and risks of future space internetworking technologies. As DTN becomes a standard architecture included in the Consultative Committee for Space Data Systems (CCSDS) [2] standardization procedures as well as the Internet Engineering Task Force (IETF) [3], a testing and verification infrastructure emerges along with a set of scenarios, operations and evaluation procedures. The deployment of a DTN testbed allows for cost-effective optimization and evaluation of space communication designs for reliable and efficient data delivery, exposes the corresponding constraints, and uncovers potential tradeoffs. In essence, a DTN testbed provides a vehicle for the transition to reliable and flexible communications in Space as well, through internetworking, reducing mission risks and enhancing space data transfer rates.

The design of a DTN testbed is associated with two major challenges:

a) to functionally satisfy the nature of the scheduled experiments and scenarios
b) to allow for ad hoc, flexible and reliable administration of network, protocol and application parameters.

Taking the aforementioned into consideration, the focus of this paper is on:

i) the experiments and scenarios that need to be deployed within a DTN ttestbed, inline with the open issues of Space DTN
ii) the protocol stack that satisfies the diversity of applications, including future application scenarios that are scheduled for Space beyond 2020
iii) the architecture that allows for flexible administration of the testbed itself; that is, the testbed architecture is a challenge in its own right.

In this context, we discuss our experiences from two major perspectives: first, the design perspective, where we highlight the major design challenges and technical obstacles such as immaturity of existing technology; and second, the experimental perspective, where we discuss the appropriate scenarios to evaluate DTN protocols for Space along with experiences from using DTN itself, based on the results we have obtained by using the testbed.

Unlike a simulation tool, a testbed allows for more realistic scenarios, uncovers practical software or hardware malfunctions and, in general, constitutes an environment where behavior of participating entities is unpredictable, unscheduled and involves a wide spectrum of parameters. In this context, a testbed can be far more reliable than a simulation tool; however, comparatively, it suffers from limited flexibility and limited scalability - let alone the higher costs of construction. In order to cancel that major disadvantage of the DTN testbed, we took two specific measures:

- We incorporated heterogeneity - to deal with scalability. In particular, the testbed includes satellite links capable of repeating communication patterns that emulate realistic propagation delays and space conditions. Therefore, space time can be realistically achieved in the experiments through ”ping-pong” operations. Furthermore, internetworked links, including intercontinental links between Europe (DUTH, Xanthi, Greece) and America (MIT, Boston, USA) are also included, allowing for evaluation of scenarios of space-data distribution on Earth. According to its design objectives, DTN may form an overlay that binds earth and space communications; in this context, data dissemination on Earth, possibly of huge volume and occasionally over congested links, will be possible.
- We integrated simulation/emulation tools - to deal with flexibility. The testbed should be able to emulate fundamental network parameters such as bandwidth, packet error rate, propagation delay, and available connectivity, and dynamically adapt to changes of those parameters in real-time. Moreover, it should be able to scale well over a larger number of communication nodes to allow for emulation of any future Deep-Space communication scenario, which will include several planetary surface networks and relay satellites. As far as transparency is concerned, network emulation should be transparent to upper layer protocols and applications, thus permitting their use and evaluation without need for modification. Finally, the testbed should be flexible enough to emulate any space communication topology, incorporate new protocols, applications and mechanisms, and interoperate with other similar DTN testbeds. With respect to Space, DTN is envisioned to support Internet-like services across interplanetary distances; Deep-Space missions are expected to sufficiently exploit an infrastructure that allows the efficient communication between in-space entities, such as explorer spacecraft, landed vehicles and orbiters. To satisfy all the aforementioned scenarios, Netem [4]

Manuscript received June 25, 2009; revised October 14, 2009.

has been integrated into the testbed, along with the ION [5] kernel and the DTN-2 [6] kernel, which have been customized accordingly.

Although the infrastructure is capable of being reformed to support Earth-oriented DTN applications, our focus presently is on space-oriented applications. Space DTN poses new questions on DTN architecture, primarily on the appropriateness of the current bundle implementation for Space. In particular, a number of issues need to be addressed through extensive evaluations. These issues may be categorized into functional evaluation, efficiency of the architecture and reliability enhancements that indeed justify the deployment of internetworking in Space. Our evaluation goes beyond the typical issues of DTN functionality to address more fundamental questions for space communications:

- Is the bundle protocol header (RFC 5050) sufficient for space applications? Several fields may be simply overhead for Space and some space-oriented mechanisms, such as priority classes, may be lacking header support.
- Is custody transfer a limiting factor for store and forward procedures when connectivity is severely constrained? Is the tradeoff between reliability and throughput administered in an unbalanced manner in favor of reliability and at the cost of throughput?
- Is fragmentation sufficiently scheduled to deal with more dynamic routing scenarios in the future?
- How far does the resource-sharing policy of each agency limit the internetworking capabilities of space resources - will it be possible for an agency to cancel the benefits of space internetworking through the deployment of wrong resource sharing policies. In that case, the investment may not worth it.
- What form of routing may be appropriate for Space?
- How much overlap does it exist between CFDP [7] and DTN?

The remainder of the paper is organized as follows. Section 2 includes related work on networking testbeds. In Sections 3 and 4 we elaborate on the requirements and the architecture of a DTN testbed, respectively. Experiments conducted utilizing our DTN testbed are presented in Section 5. Next, we detail our experiences from building and using the DTN testbed in Section 6. Finally, we conclude the paper in Section 7, and outline future work.

II. Delay Tolerant Networking Efforts

Currently, all space communications are static, inflexible and, due to limited communication among space entities, also unreliable. In this context, less sophistication is required from communication protocols; space operations are dominated by human-operated procedures; communication is predetermined, utilizing fixed allocated routes; application services are determined manually and for limited duration; and a possible update due to dynamic events is theoretically enough to bring schedules and management activities to a possible collapse. However, as the number of space assets continuously increases, missions become more complex and new communication architectures and protocols for backbone, access, and proximity networking need to be designed, validated and optimized.

Delay-Tolerant Networking has been proposed to overcome relevant problems that arise from current Deep-Space communication networks, since it embraces the concept of occasionally-connected networks that may suffer from frequent partitions and that may be comprised of more than one divergent set of protocols. Long propagation delay values, increasing number of alternative communication paths that may be used to reach a single receiver, demand for interoperability among space agencies, requirement for 100% reliable delivery of all kinds of scientific data, different type of applications (e.g. file transfers, telemetry and command and control), delivery of large volumes of scientific data, audio and visual information from in-space entities to terrestrial stations, and missions with varying link architectures, orbit types, frequency bands, coding schemes, data rates and end-to-end protocols used, is only part of the challenges DTN needs to address. Under these circumstances, in order for the DTN testbed to be effective, specific requirements need to be satisfied.

A number of network testbeds have been designed and deployed so far, aiming to test various protocols and mechanisms, prior to their commercial deployment. That is, network research testbeds have been the crucial proving grounds in which new networking research ideas have be tested, stressed, observed, reformulated, and ultimately proven before making their way into operational systems.

One of the most widely known network testbeds is Emulab [8]. Emulab is designed to accurately emulate, on low-end PCs, virtual topologies over an order of magnitude larger than the physical hardware, when running typical classes of distributed applications that have modest resource requirements. Emulab virtualizes hosts, routers, and networks, while retaining near-total application transparency, good performance fidelity, responsiveness suitable for interactive use, high system throughput, and efficient use of resources.

Another successful and widely used testbed is Defense Advanced Research Technology Network (DARTnet) and its follow-on Collaborative Advanced Interagency Research Network (CAIRN) [9]. DARTnet was a wide-area experimental environment for research on the network layer and transport layer protocols and on advanced applications that would drive this research. The primary advanced application turned out to be packet audio and video, especially video teleconferencing. The CAIRN backbone and research networks constitute a testbed for the development of experimental architectures, protocols and algorithms for real-time multimedia applications and related technology. Applications developed in this testbed include distributed audio and video conferencing systems, such as used in the MBONE [10], as well as real-time interactive simulation systems.

Similarly to the aforementioned testbeds, we describe a testbed architecture able to emulate and validate various protocols and mechanisms. However, we focus on Delay-Tolerant Networks instead of IP-related networks. There are several testbeds deployed to allow experimentation on DTN networks, most of which employ mobile nodes to emulate link disruptions. Some are briefly described below.

Diverse Outdoor Mobile Environment (DOME) [11] is a testbed designed and built to support large-scale mobile experimentation. DOME consists of 40 computer equipped buses, numerous battery-powered nomadic nodes, hundreds of organic WiFi access points, and a 26-node municipal WiFi mesh network. Employing a DTN architecture, the researchers managed to cover an area of 150 square miles.

The Zebranet [12] is a project conducted in Kenya by the University of Princeton to track wild Zebras in an area

via wireless transmissions without stationary antennas, based on the DTN framework. The animals were equipped with GPS receivers and small solar powered transmitters. The data content was transmitted every two hours to the animals in reach, stored and relayed on with the new animal's own position information until the information was finally collected by mobile units and transported via a gateway (ranger station) to the Internet.

A testbed that will employ DTN to extend Internet access to remote regions is the main objective of N4C project [13]. Taking into consideration the large distances involved, the 'always on' paradigm of constant connectivity and essentially synchronous access enjoyed in many urban areas today is not available to these regions.

Compared with the aforementioned testbed deployments, our approach focuses on accurately emulating space communications; our main objective is research and experimentation on DTN protocols and mechanisms to be used in Space. This differs from the previous approaches, since we need to emulate constrained conditions such as long propagation delays and high bit error rates, evaluate possible overlap with space protocols and investigate the efficiency of space applications.

III. Requirements of a DTN testbed

In order for a DTN testbed to accurately emulate real networking conditions, from terrestrial networks to Deep-Space links, several requirements need to be defined and satisfied.

A. Design goals

The main objectives of the DTN testbed regarding its accuracy and efficiency are:

i) Dynamic control of network parameters. The testbed should be able to emulate fundamental network parameters (such as bandwidth, packet error rate, propagation delay, and available connectivity), and adapt realistically and dynamically to changes in those parameters in real-time.

ii) Scalability. Currently terrestrial networks comprise large numbers of nodes. Deep-Space communications involve a limited number of communication nodes, although in future they will include several planetary surface networks and relay satellites. Therefore a DTN testbed should be able to scale well over a larger number of communication nodes to allow for emulation of multiple nodes' communication scenarios.

iii) Transparency. Network emulation should be transparent to upper layer protocols and applications, thus permitting their use and evaluation without need for modification.

iv) Flexibility. The testbed should be flexible enough to: emulate any communication topology; incorporate new protocols, applications and mechanisms; interoperate with other DTN testbeds; and provide a reusable infrastructure towards an actual hardware testbed.

B. Software requirements

Network parameters vary, not only among emulated networks, but also between any couple of nodes. Any pair of nodes may have unique link characteristics, such as bandwidth, packet error rate, propagation delay in a multiple node topology. Consequently, a DTN testbed should provide link emulation mechanisms in order to adjust the corresponding parameters. A sophisticated link emulation software needs to be implemented in order to emulate a variety of environments and scenarios.

Since a DTN testbed may be consisted of a large number of nodes, a node control mechanism is required to control and monitor node operations. Control operations allow for the realization of the scenarios, while monitor operations enable the generation of statistics and results.

A state-of-the-art testbed capable of emulating a wide range of topologies and corresponding scenarios, calls for the design of an advanced, yet user-friendly Graphical User Interface. In essence, the GUI should provide total control of the testbed, as well as detailed monitoring of the system.

C. Protocol requirements

A DTN testbed should emulate different environments, adjusting parameters correspondingly. In this context, a DTN testbed should provide an extended protocol stack, supporting various protocols for space communications, the corresponding convergence layer protocols, as well as the required mechanisms for accurate link emulation.

Bundle protocol [14] is the core protocol of DTN architecture, comprising mechanisms in order to cope with intermittent connectivity and long propagation delays. Taking advantage of scheduled, predicted, and opportunistic connectivity, custody-based retransmission mechanisms and late binding of overlay network endpoint identifiers to constituent underlying (currently internet) network addresses are key capacities of the Bundle protocol. The aforementioned mechanisms affect the operation of the underlying transport and network protocols.

D. Tools

Deep-Space communications are based on predetermined, scheduled contacts, due to the orbits of planets and assets. In order to enhance Deep-Space emulation, applying accurate link characteristics is imperative. Using tools to emulate planet and space asset orbits, such as STK/Space Environment and Effects Tool (STK/SEET) [15], and evaluate the effects of the space environment to communications (e.g. sun's solar flare activities) can elevate realism.

Besides software link emulation, the integration of link emulation hardware into the testbed is also possible. Space Agencies implement data link layer in hardware and, in conjunction with space channel emulation hardware, accurately emulate space links (f.e. ESA Portable Satellite Simulator (PSS)). Such devices can also be integrated within the testbed, facilitating the deployment of DTN protocols over link layer protocols currently used for space communications.

E. Scenarios to be deployed

Since DTN interconnects different types of networks, a testbed should be able to emulate and evaluate a wide range of complex, terrestrial and/or space scenarios.

Regarding terrestrial scenarios, mobile nodes with intermittent connectivity, low propagation delays and opportunistic contacts need to be emulated. DTN can provide connectivity to the edges of current Internet infrastructure. In that sense, the DTN architecture is a potential candidate for ad-hoc, sensor and military networks.

Current space communications are static, since the majority of space assets are not designed to communicate with each

other, but only with terrestrial infrastructure. Networking space equipment will allow for increased connectivity time, faster delivery of data and more efficient exploitation of space infrastructure. There is an emerging need for automated space communications and efficient transport and network layer protocols. DTN architecture is a prominent candidate for Deep-Space communications, in order to cope with intermittent connectivity and long propagation delays, and utilize predetermined contacts to deliver files such as scientific data, telemetry and control commands between space assets and control centers.

In the future, terrestrial and Deep-Space networks are expected to be interconnected as in order to increase connectivity time and data delivery efficiency among terrestrial and space assets. DTN, being an overlay architecture, hides any dissimilarities between the underlying network characteristics and enables interoperable operation.

IV. Architecture of a DTN Testbed

The DTN testbed is implemented in a modular fashion and each component can be developed independently. It includes emulated network links, real and modeled protocol implementations and is enhanced with actual satellite link interfaces offered by HellasSat satellite. Network parameters and evaluation results can be easily controlled through a Graphical User Interface.

Figure 1 illustrates the main DTN testbed components and their interconnections. Graphical User Interface (GUI), Central Management System (CMS) and Kinematics Modeling System (KMS) compose the administrative part of the testbed. A number of distributed emulation nodes are responsible for emulating space communication links and protocols. Components are described in more detail in Section 4.3.

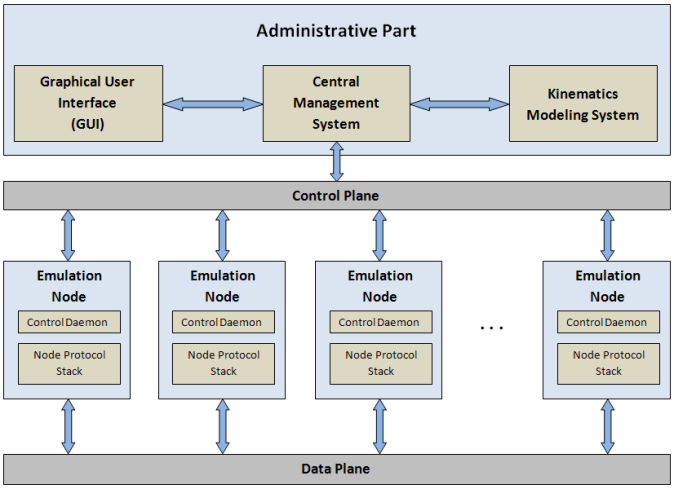

Fig. 1. DTN testbed architecture

A. Emulation Scenarios

As mentioned in Section 2, the DTN testbed focuses mainly on the concept of Delay Tolerant Networking in Space and is designed to emulate the properties of current and future Deep-Space communications. Emulation scenarios can include planetary surface networks that are characterized by low error rates and propagation delays of the order of milliseconds, near-planet relay networks with higher error rates and propagation delays of the order of seconds, as well as Deep-Space low-bandwidth backbone networks of long-haul links, characterized by high error rates and propagation delays.

An emulation scenario consists mainly of a number of emulation nodes and a time-based network model produced by the Kinematics Modeling System (KMS). Apart from the emulated links, a real geostationary link through Hellas Sat satellite is also included in the testbed and is used to provide actual link conditions for short-haul satellite links.

B. Supported Protocol Stack Configurations

Figure 2 shows the protocol stack configurations available in the testbed. The fundamental architecture supported by the testbed is the generic DTN Architecture as defined in RFC 4838 [16]. This includes support of the Bundle Protocol along with its convergence layer protocols such as Licklider Transmission Protocol [17], TCP and UDP. In addition, Bundle Protocol is compatible with several experimental routing protocols such as Delay Tolerant Link State Routing (DTLSR) [18], Contact Graph Routing (CGR) [19] and PRoPHET [20]. An alternative architecture that differs conceptually from the RFC-based one is implemented based on CCSDS File Delivery Protocol (CFDP). CCSDS File Delivery Protocol (CFDP) has been developed for automatic and reliable file transfer between spacecraft and ground, build on top of the supplementary CCSDS protocols. Although an application protocol, CFDP includes also transport functionalities. It is a ready-to-flight product and constitutes the ancestor of Delay Tolerant Networking in Space, as it is the first protocol that introduced the concepts of store-and-forward and custody transfer. Within the testbed, CFDP sits on top of UDP and in this case reliability is achieved utilizing CFDP mechanisms. Additionally, CFDP can also be used only as a file transfer application, while the Bundle Protocol enables file transfer in a store-and-forward fashion and ensures reliability. Alternatively, Asynchronous Messaging Service or simple test applications can be used on top of Bundle Protocol.

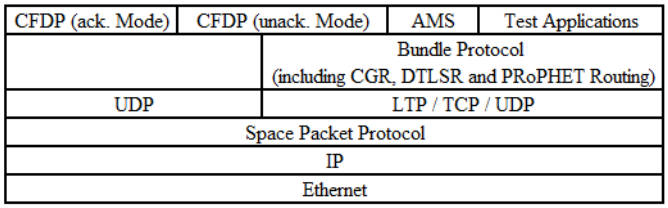

Fig. 2. Testbed's protocol stack

In all the above protocol stack configurations Space Packet Protocol (SPP) [21] can optionally be used as a common layer below DTN-specific protocols. A basic SPP protocol functionality is thus included in the testbed. This configuration is closer to the existing space communication infrastructure and allows for legacy mission support.

C. Components

1) Graphical User Interface: The Graphical User Interface (GUI) is the main input and output system of the ttestbed. In cooperation with the CMS it allows for effortless and dynamic configuration of the testbed emulation nodes, as well as real-time monitoring of the testbed. At the moment, GUI is fully compatible only with nodes running ION DTN reference implementation, however it can be easily extended to support

a number of alternative protocol implementations such as DTN2 and ESA/ESTEC CFDP. That is, GUI cannot configure remotely any DTN2 or ESA/ESTEC CFDP implementation parameters; however, it can still dynamically configure any emulated link and fully monitor the respective nodes.

The details of an experiment can be described as follows: A user initially configures the number of the required nodes and the respective IP addresses. Then, each available link and its characteristics (contact schedules, bandwidth, PER, propagation delay), as well as the required convergence layer protocols (LTP, TCP, etc) are configured. Finally, the user selects at least one transfer application, along with the desired time of its execution. The GUI creates all the necessary configuration files and utilizes CMS to distribute them along the testbed. Apart from this initial static configuration of an experiment, link characteristics can also be dynamically modified during the experiment's execution.

The GUI is also responsible for the presentation of information about the status of each node, as well as testbed statistics. This is achieved in cooperation with the CMS, which stores all log information it receives from the testbed nodes. The GUI presents this information in a separate log window. The process of logging is described in more detail in the following section.

The GUI is developed under an Ubuntu Linux environment, using two tools: Glade [22] and GTK+ [23]. Glade is a Rapid Application Development (RAD) tool that enables quick development of user interfaces for the GTK+ toolkit and the GNOME desktop environment. GTK+ is a highly usable and feature-rich toolkit for creating graphical user interfaces, which are written in C languages, having also bindings to many other popular programming languages such as C++, Python and C#. The testbed GUI is totally developed in C. Glade files are converted into XML files using GtkBuilder GTK+ object and are loaded by the C application, dynamically.

2) Central Management System: The GUI software creates all the necessary configuration files for each experiment node. The job of the control system is to transfer the configuration files to the appropriate nodes, control the execution of software and gather experimental data generated by the nodes participating at the testbed. The functionality of the Central Management System is achieved utilizing a list of software facilities provided already from UNIX standard utilities, along with some glue-code written in python. Also, ION software was modified so that events generated at execution time would be redirected to the syslog unix facility instead of the local machine standard output.

For the development and execution of the Central Management system, the following unix facilities and software are used:

- Secure Copy (SCP) for file transfer between the central and testbed nodes.
- SSH remote execution for executing software on the ION nodes.
- AT unix facility for scheduling application execution on the nodes at specific times.
- NTP facility for synchronizing the clock of all the nodes participating on the testbed. This is mandatory because all results generated from ION software are tagged with the precise time they occurred.
- The capability of remote logging with RSYSLOG unix daemon. Software events from ION software are logged via remote sysloging at the central management node.
- A combination of shell scripts and python programs are used as glue-code for all the above facilities.

The use of the above facilities reduced the required time to develop the testbed, since they are already proven and tested facilities in the UNIX world.

The logic diagram of the communication among GUI, CMS and the client nodes is depicted in Figure 3.

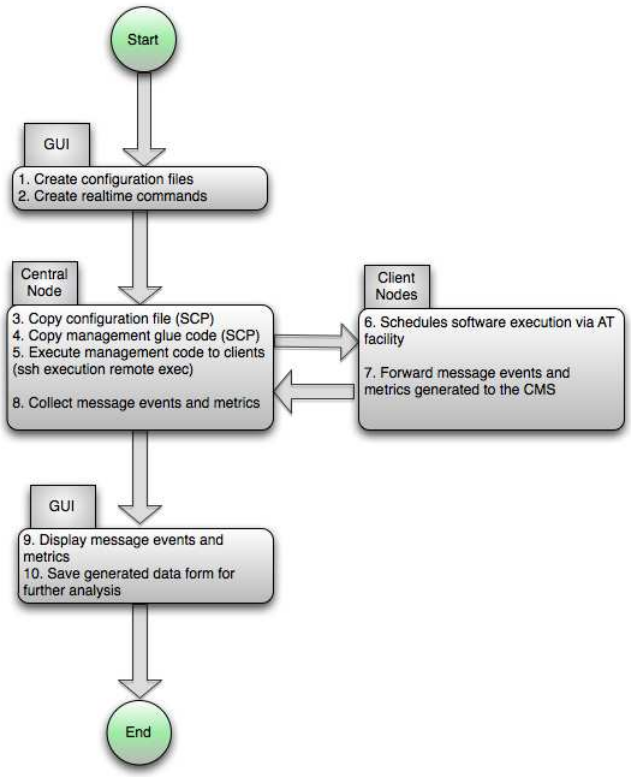

Fig. 3. DTN testbed logic diagram

3) Kinematics Modeling System: The purpose of the Kinematics Modeling System (KMS) is to produce an accurate time-based network model, focusing on a more generic scenario description. KMS could take a scenario description as an input that defines, for example, a multi-node communication path from Earth to Mars, the initial position of each node and the communication protocols used; and produce a time-based network model that is based on simulated nodes or planet movement, as well as simulated space weather conditions. This network model includes specific information on link connectivity and availability, along with link characteristics (PER, available bandwidth and propagation delay).

4) Emulation nodes: Emulation nodes are the most important part of the testbed and their role is twofold: they implement a variety of communication protocol stacks and emulate network conditions; each node is physically mapped to a single machine.

Functionally, each node consists of two parts: the control part, and the data part. The control part is responsible for receiving the configuration files from the CMS and executing them. The data part implements the various protocol stacks available on the testbed, which means actual protocol implementations at application, transport and network layer. Since GUI is currently only compatible with ION implementation, the control part of each node configures only the ION-related protocols (BP, LTP, TCP) and relevant test applications, as

well as all the necessary network emulation functions among nodes.

The process of communication between the control part of each node and the Central Management System is described in Section 4.3.2. We add here a brief description on how link emulation is applied to each node. An important characteristic of the testbed is link layer emulation that enables transparent network emulation. That is, at each node, outgoing IP packets are captured and controlled prior to their transmission to the link layer. Network emulation, in terms of bandwidth, PER, corruption, duplication, re-ordering and delay is applied using Netem, the Linux network emulator module, in conjunction with TC, the Linux traffic control tool, which in turn is part of the Linux IProute2 package of tools. Each emulation node of the testbed supports any protocol stack configuration described in Section 4.2.

5) HellasSat satellite link: HellasSat satellite link can provide actual link conditions for short-haul satellite links among emulation nodes. This link is provided by Hellenic Aerospace Industry (HAI), which manages part of the available bandwidth using a hub station. A satellite terminal is established at the ComNet group laboratory of the Democritus University of Thrace (DUTH) and each testbed node located at DUTH can utilize the satellite interface to communicate with testbed nodes located at HAI.

V. Experiments with the testbed

A. Experimental results

The testbed comprises several network, transport and application layer protocols; these are evaluated using appropriate scenarios and topologies, in order to study their performance either independently or combined, as to increase the efficiency of the system.

1) Application layer protocol - CFDP: Even though CFDP provides both transport and network protocol functionality, in order to integrate it into the DTN testbed, we utilized its advanced application layer functionality, combined with the transport and network layer capabilities of the underlying DTN architecture. CFDP can provide both unreliable data transmission for one-way links, and a reliable transmission mechanism similar to DTN/LTP. CFDP is integrated into the testbed in unreliable mode; reliability is assured by the underlying DTN LTP and TCP protocols for space and terrestrial environments respectively. Moreover, in the absence of line-of-sight between the two endpoints, CFDP offers two mechanisms for storing data to intermediate nodes, called Store-and-Forward and Extended Procedures; we have evaluated them both using the testbed.

In order to evaluate the performance of CFDP over terrestrial delay tolerant networks with low propagation delays and space links up to the distance of geostatic satellites, we use 50 - 300 ms propagation delay over BP/TCP. At higher propagation delays, TCP performance is severely decreased and the protocol timers collapse; TCP is designed for communication over terrestrial links with propagation delay at the order of some milliseconds. The packet loss rates vary from 0 to 10 percent, while bandwidth is asymmetrical, using a 1 Mbps link to deliver a file of 10 MB in a two-node topology. The size of each CFDP PDU equals to 1024 Bytes. In Figure 4, we depict Task Completion Time for increasing loss values and we notice that the performance of TCP under increased packet loss and propagation delay was captured, as TCP is not designed to operate over such connections. The results show that CFDP can perform under these conditions quite successfully, especially if we consider that CFDP, as a stand-alone application, is designed for much higher delays, in the order of seconds.

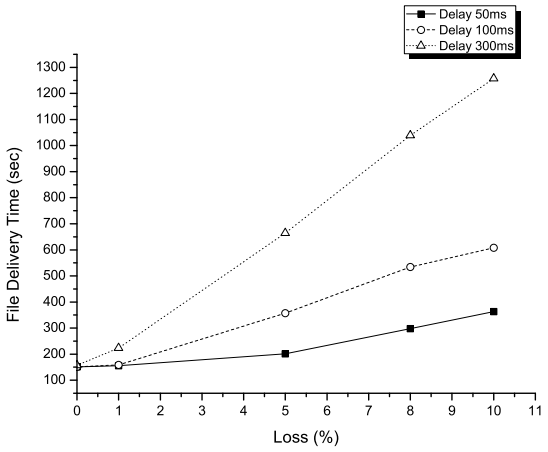

Fig. 4. CFDP over DTN-TCP

Space rovers and satellites are able to send data at rather high speed, unlike Earth to Space links which are characterized by lower speeds. In this context, we emulate space links using asymmetric bandwidth and variable error rates at forward and reverse links, respectively. Specifically, we use a 1 Mbps downlink and a 256 Kbps uplink, with a 30 seconds propagation delay each, and packet loss rates 0, 0.1, 1, 5 and 10 percent both in downlink and uplink. We evaluate CFDP in unreliable mode over ION with LTP protocol selected for reliability, and compare it to CFDP in reliable mode over UDP. In order for LTP's and CFDP's retransmission mechanism to be comparable, CFDP is set in immediate NACK mode.

The LTP implementation we utilized is designed to operate in Deep-Space links where data loss due to corruption (radiation, limited transmission power) is generally minimized by heavy forward error correction coding at link level. Moreover, LTP is designed for moderate data rates and the high data rates that are used in our experiments resulted in high burst rates that caused the malfunction of the receiver's retransmission timers. In our scenarios, LTP could not function properly at high packet loss rates; therefore, we present LTP results only up to one percent packet loss.

File deliveries of five and ten MB CFDP PDUs of 1024 Bytes over UDP were evaluated in a single hop topology, using packet loss rates up to ten percent. In Figure 5, we capture file delivery time for increasing loss values; file delivery time of CFDP-LTP combination appears better than individual CFDP for packet loss rates up to one percent for 5 MB filesize. In the case of 10 MB filesize, higher number of packets causes much higher burst rates, more retransmissions and, as a result, LTP underperforms. The outcome of the experiments was that the integration of CFDP over DTN architecture is feasible, allowing the exploitation of the benefits of both protocols simultaneously. The new version of LTP, which is designed for higher data loss rates and includes an improved retransmission mechanism, is expected to perform much better, resulting in an efficient integration of CFDP and Bundle protocol.

The last set of experiments regarding CFDP, involves Store-and-Forward (SFO) and Extending Procedures mechanisms.

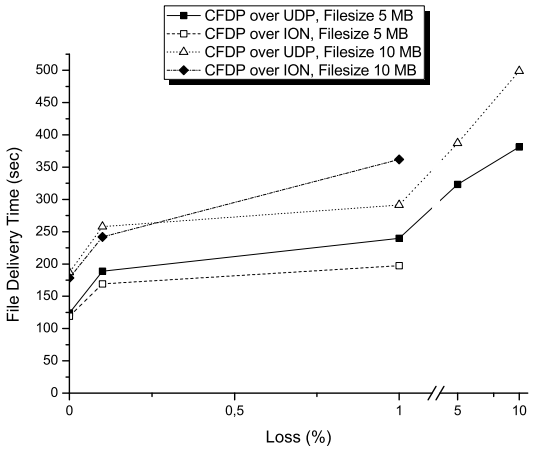

Fig. 5. CFDP over DTN-LTP

The main difference between the two aforementioned mechanisms is that using SFO, each transmitted file is received, stored and forwarded in a hop-by-hop manner by intermediate waypoint users rather than CFDP entities. Additionally, Extended Procedures allow for incremental forwarding of the file and, thus, allowing portions of the file to be forwarded by a waypoint upon receipt, rather than upon the receipt of the entire file. It appears that, in case there is an end-to-end path to the destination, the higher the propagation delay, the losses and the file size, the better the performance of Extended Procedures versus SFO. In fact, Extended Procedures complete the transmission of a 10 MB file in a two-hop topology using 512 Kbps links with 30 seconds propagation delay in two minutes, instead of the four minutes required in the case of SFO.

2) Routing protocols: We have evaluated some of the most prominent routing protocols for Delay-Tolerant Networks: Epidemic, Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET), and Contact Graph Routing (CGR), in space environment. Using the DTN testbed, we showed that for increasing delay, Contact Graph Routing significantly outperforms the two alternative routing schemes.

According to Epidemic routing protocol, transmitted data is continuously replicated until all nodes receive a copy. In particular, upon the receipt of a new packet, a node first checks whether itself is the final destination of the packet; if not, it multicasts the received packet to every other node it shares a link with. PRoPHET incorporates the novel idea of utilizing knowledge of previous encounters between any two nodes, in order to select the best path towards the receiver. This is achieved by calculating delivery predictability for each node. CGR, on the contrary, is a dynamic routing protocol which takes advantage of the fact that contact information between any nodes in space communications is predetermined. In particular, all nodes utilize knowledge of both their current state and all scheduled future communication contacts and, thus, achieve the computation of successful routes. Moreover, CGR includes responsive mechanisms to react to any communication anomaly that may occur.

In Figure 6, we summarize the characteristics of the aforementioned routing protocols as far as energy consumption, autonomy, contact exploitation and Quality of Service are concerned. As noticed, Epidemic and PRoPHET routing may be autonomous and able to exploit both scheduled and opportunistic contacts; however none of them is energy-efficient or provides Qos, like Contact Graph Routing does. Epidemic and PRoPHET Routing are incorporated into DTN2 implementation, while CGR is part of the ION platform.

	Epidemic Routing	PRoPHET Routing	Contact Graph Routing
Energy-Efficient	No	No	Yes
Autonomous	Yes	Yes	No
Contacts Exploited	All	All	Scheduled Only
QoS	No	No	Yes

Fig. 6. Characteristics of Epidemic, PRoPHET and CGR

In the first set of our experiments, we emulated some fundamental scenarios of space communications, i.e. a sensor in Space transmitting temperature measurements to Earth. We used a topology of four nodes to compare Epidemic, PRoPHET and Contact Graph Routing under increasing RTT values, using Task Completion Time as a metric. The topology of the experiment consists of one sender, one receiver and two intermediate nodes, as shown in Figure 7 below. The intermediate nodes provide two alternative routing paths, while only one of them is connected to the endpoints at any given time. The sender transmits 100 packets of 10 KB each with a time interval of five seconds between any two consecutive transmissions, while the bandwidth between any two connected nodes is 10 Mbps. For each network configuration, we calculated the results by averaging the output of 50 emulation runs.

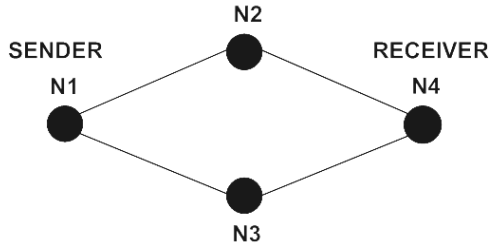

Fig. 7. Two-hop alternate path topology

In Figure 8, we measure Task Completion Time of each protocol as Round Trip Time increases. As depicted, CGR outperforms both PRoPHET and Epidemic routing as RTT values increase; for almost zero-second delay we observe that the performance of all three protocols is almost identical, whereas for RTT values greater than one second PRoPHET's performance degrades in contrast to the relatively stable performance of both CGR and Epidemic routing. The reason behind this, is PRoPHET's requirement to exchange routing information before each transmission, resulting in wasting valuable time and resources. CGR, however, achieves better performance by utilizing predetermined information on each node's position and movement.

In our next set of experiments, we emulated the transfer of a large file over the same topology of four nodes (Figure 7), using Epidemic, PRoPHET and CGR as the routing protocol, respectively. The sender transmits a file of a variable size (starting from 1 MB and up to 20 MB) and each time we measure Task Completion Time. The link characteristics remain the same as in the previous set of experiments.

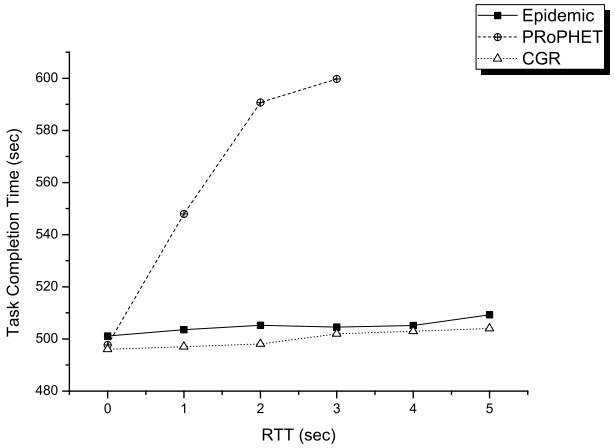

Fig. 8. Round-Trip Time impact on Task Completion Time

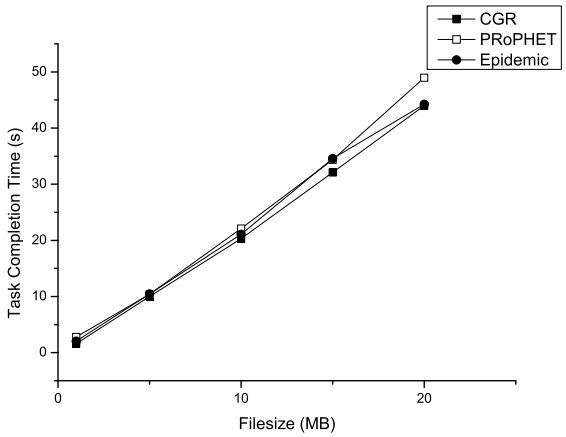

Fig. 9. Task Completion Time for varying Filesize

As illustrated in Figure 9, CGR outperforms both Epidemic and PRoPHET Routing, although with a minor variation. This can be attributed to the fact that CGR is the most ”static” protocol among all three, since it knows *a priori* the available contacts and links, and does not perform active link discovery. Even Epidemic routing results to lower Task Completion Time, in comparison to PRoPHET, as each node quickly multicasts all the packets it receives.

Using our DTN testbed, we showed that CGR achieves considerably better performance than Epidemic and PRoPHET Routing, when delay is in the order of seconds. We also observed that PRoPHET does not perform well even for short delay values, neither for small, nor for large filesizes. This can be justified by PRoPHET’s mechanism to exchange routing information prior to each data transmission.

B. Future work

1) Transport Protocol: We have already evaluated LTP for space scenarios and TCP for terrestrial. During this phase, several issues, which we intend to solve, arose. Therefore, we have designed a space-oriented DTN Store-and-Forward Transport Protocol (SF-TP) that has the following unique characteristics:

- Proactive retransmission
  SF-TP includes three different proactive retransmission mechanisms. The first one is inherited from Deep Space-Transport Protocol (DS-TP) [24] and its Double Automatic Retransmission (DAR) technique, and transmits a fraction of the total number of packets twice, importing some delay Rd between the original transmissions and the retransmissions. Retransmission rate is loosely coupled with packet error rate, as measured by the sender using SNACK information. Rd is an increasing function of the packet sequence number, and the receiver utilizes two different types of SNACKs: SNACK-1 for faster PER calculation and SNACK-2 for triggering retransmissions. The second retransmission technique, called DAR2, is similar to the first one, however more simple. The main difference is that the second mechanism adds a maximum fixed low delay Rd between original transmissions and retransmissions. This has the advantage that alleviates the need for two different types of SNACKs and thus simplifies receiver operations. The drawback of this approach is that it cannot enhance reliability for a continuous fraction of data.
  The third retransmission approach incorporates packet level erasure coding to allow for advanced recovering from packet losses. Coding rate is dynamically adjusted and based on network measurement. Several coding schemes can be applied, such as Reed-Solomon and Low Density Parity Check (LDPC) [25], focusing mainly on Long Erasure Codes (LEC). Although the use of packet error correction reduces retransmission overhead for the same packet recovering capability, it essentially increases processing overhead. That is, there is a trade-off between processing overhead, retransmission overhead and recovering capability.
- Cut-through forwarding
  SF-TP includes the functionality of cut-through forwarding. Since the number of space network nodes increases, as new missions are scheduled, there is a desire among space agencies to interconnect spacecraft, landers, rovers, and other space assets. As a result, the new space internetworking era departs from the long-established Direct-to-Earth communication model, and adopts multi-hop paths. Since a given node can exploit potentially more than one path to final destination, SF-TP focuses on redundant data transmission via parallel data paths. Controlling the redundant transmission rate to a preset threshold, the proposed solution can offer robust reliable services. Sequence of application data is resumed at the receiver. Parallel data transfer can be implemented by preserving the original sequence number space. This feature requires explicit definition in the protocol header, so that the final destination can anticipate and merge data packets coming from different paths.
- Sending rate adaptivity
  SF-TP’s sending rate can be accurately characterized as temporarily constant. Sending rate adaptation can follow network events, such as storage capacity exhaustion. These network events can be either perceived by senders through advanced mechanisms or explicitly signaled by receivers.
- End-to-end functionality
  DTN, as described in RFC5050, does not provide the typical end-to-end functionality. Since no mechanism or protocol does have a total view of the network, DTN employs a hop-by-hop approach to provide reliability and allows for limited routing and flow control decisions. More specifically, mechanisms such as custody transfer

provide a reliable trail between the sender and the receiver, but this does not ensure total reliability. As far as routing is concerned, decisions are made inside the network, while endpoints do not have information about the conditions met there. The same applies to flow control mechanisms. An end-to-end mechanism should take into consideration any shortage of communication resources in the intermediate nodes and adjust the sender's transmission window, correspondingly. We intend to inject end-to-end characteristics into the DTN architecture, rendering the architecture not only more reliable, but also more efficient.

2) Routing: Our evaluation of current routing protocols uncovered the need for a space-oriented advanced routing scheme. The routing scheme currently deployed and used in DTN is rather static. It utilizes knowledge of all scheduled future communication contacts to calculate possible routes. The main limitation of this approach is that opportunistic contacts cannot be exploited. Moreover, any anomaly in the network is not detectable, possibly leading to unsuccessful transmission efforts, while other more suitable routes may be available. Another drawback of the current routing scheme is that it takes into consideration only the time needed to transfer data between two nodes. Other metrics, such as financial cost, link reliability (providing custody transfer etc.), security and usage of agency resources, are not used by the protocol to provide a suitable route towards the receiver. Furthermore, there is no provision for exploitation of parallel data transfers. All in all, current DTN routing schemes provide limited sophistication.

Furthermore, DTN prioritization system supports only three priority classes, with one more reserved for future use. This is not sufficient for complex space missions, since they usually produce many different types of data not of the same value (scientific, financial, etc). To cover this need, we will enhance DTN by supporting more priorities. Parameters such as policies imposed by space agencies, custody availability, and timely delivery versus financial cost, will be used to calculate the best available route between two nodes.

In that context, we plan to design and deploy a sophisticated routing mechanism that will cover all the aforementioned issues, namely dynamic route calculation, policies, parallel data path computation, and sufficient priorities.

3) DTN DNS server: Currently, the translation of DTN addresses is performed in a manual basis, since there does not exist an automatic mechanism to provide this functionality. We plan to deploy a DNS server that will allow the user-friendly identification of nodes, followed by a resolution scheme that identifies corresponding DTN sending and receiving node. A number of security alternatives for the DNS solution will be investigated. It is envisaged that the preferred choice will be based on the principles of DNSSEC (DNS Security Extensions). This will ensure authentication of data origin and data integrity for the domain names, enabling dynamic updates of locations throughout the DTN infrastructure, while maintaining open access to clients. The DNS and DNSSEC will be subsequently adapted to the DTN addressing structure.

VI. Experiences from building and using the testbed

A. Challenges

We faced several challenges during the deployment of the DTN testbed. A description of the most important ones is provided below.

- Accurate emulation of space conditions
  A major objective of our testbed was to accurately recreate the network conditions met in Space. That is, link characteristics, such as propagation delay and packet error rate, as well as the orbital movement of planets and space assets needed to be accurately emulated in the testbed. To support this objective, we decided to use the Netem tool which is installed in each node of the testbed. This distributed architecture provides several advantages; it minimizes packet processing overhead and, at the same time, increases the throughput of the emulation system - unlike centralized architecture which suffers from increased processing overhead. The very long propagation delays related to Deep-Space communications pose the requirement for handling a large number of packets prior to transmission and, thus, amplify this problem. Consequently, the choice for a distributed architecture complies with our initial system design goal of scalability. Hardware channel emulators would be an alternative to achieve realistic link emulation. Such an emulator enables the creation of a mathematical model representing the physical radio signal transmission medium, replacing the real-world radio channel between a radio transmitter and a receiver.
  Another issue regarding the link emulation is the exact values used to emulate specific space links. We use a flexible tool that allows us to insert link delay values and probability distributions for packet error rates taken from accurate tables. A different approach would be to use appropriate software, such as the Satellite Toolkit (STK), that calculates planet and spacecraft trajectories, providing information regarding link characteristics.
- Implementation of a space protocol stack
  In order to achieve accurate results, we need to implement a protocol stack that is as close as possible to the protocols actually used in Space. In that context, our emulation ranges from Network layer protocols, with the deployment of the Space Packet Protocol, to Applications, such as CFDP. A more advanced approach would be to deploy appropriate link layer protocols as well, such as TM/TC [26][27].
- Scalability of the testbed
  Another important requirement that should be met is the scalability of the Testbed, since complex scenarios call for many nodes to be available. In our approach, one emulation node per PC is used, avoiding the deployment of virtual machines, which increase the processing overhead and pose a limitation to the testbed.
- Reliable/realistic platform hosting the emulation software
  We use Linux machines to host our emulation software and tools. Although a very reliable platform, it is, however, not used in Space. Currently, most spacecrafts use real time operating system, such as RTEMS and VxWorks. Hence, a different approach, and at the same time a possible extension for our testbed, would be to install nodes that use the aforementioned operating systems.

B. Limitations

As sophisticated as our testbed may be, some limitations still exist. In the following paragraphs, we briefly describe some of them.

- GUI
  The Graphical User Interface is a valuable tool, since it allows for easy set-up of the experiments and provides output results. Nevertheless, it only supports ION software and link emulation tools. CFDP provides its own GUI, while DTN2 is configured using scripts.
- Remote nodes and network conditions
  Since there exists an Internet link between some remote nodes, we expect some variance in our test results, due to the lack of control over this link. Indeed, network conditions, such as link congestion, can alter the outcome of an experiment regarding the performance of protocols under evaluation. To escape this limitation, we exclude these remote nodes when rather sensitive metrics are used.
- TM/TC protocol stack
  Currently the testbed lacks link layer protocols are similar to those used in Space. TM/TC and AOS [28] protocol stacks would provide more realistic results and more accurate upper layer protocols implementations. For example, since TM/TC can provide information on the upper layers about link availability, it would be possible to implement a sophisticated transport protocol that would utilize the information provided by link layer to decide whether it should trigger a transmission.

C. Differences between our approach and a simulation approach

A simulation approach regarding the experimentation and development of new protocols appears to have some advantages. First of all, incorporating many nodes in an experiment is an easy task, rendering a simulation approach scalable in comparison to testbed approach. That is, in a testbed, one needs to use individual PCs to implement each node. Moreover, simulation uses models of the protocols and the underlying hardware being tested and allows for complex scenarios to be created on a single workstation. However, simulation neither works in real time, nor considers the complexities of the real system. Furthermore, since simulation is based on models, rather than on the actual implementations, it may lead to inaccurate performance evaluations. On the contrary, emulation uses actual protocol implementations on hardware being able to emulate the complexities of an actual system.

VII. CONCLUSIONS

Delay Tolerant Networking constitutes the key technology for future space internetworking, able to utilize alternative routing paths, provide better exploitation of the available resources and allow for dynamic management of currently static procedures. However, extensive testing and evaluation is required prior to the wide deployment of DTN on space assets, in order to uncover possible malfunctions.

In this paper, we have described step-by-step the procedure of architecting a state-of-the-art DTN testbed for space communications, since it supports accurate, real-time experiments. The main infrastructure of the testbed consists of three basic components; the graphical interface provides users with the functionality to configure network parameters and follow the evolution of the experiment; the central management system is responsible both for the distribution of input data to all nodes and collection of results; and the kinematics modeling system implements the necessary conditions as far as delay, loss etc., between any two nodes are concerned.

We have also shown that we can evaluate space protocols and mechanisms in real-life conditions. CFDP has been tested both over TCP and LTP, showing significant performance increase in the latter case. Moreover, three distinct routing protocols, namely Epidemic, PRoPHET and Contact Graph Routing have been evaluated, showing that CGR achieves better performance, lacking, however, the ability to discover opportunistic contacts or unplanned link outages.

In future work, we will design a space-oriented DTN Store-and-Forward Transport Protocol that will incorporate mechanisms such as proactive retransmission, parallel data transfer and end-to-end functionality. As far as routing is concerned, we plan to design and deploy a sophisticated routing mechanism, able to provide multiple priority classes and satisfy specific policies, imposed by space agencies. Activities have also been scheduled to evaluate cross-testbed activity with NASA and MIT.

REFERENCES

[1] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst and K. Scott, Delay- Tolerant networking: an approach to InterPlaNetary Internet, *IEEE Communications Magazine*, 41(6), pp. 128-136, June 2003.
[2] Consultative Committee for Space Data Systems (CCSDS), Official Website: http://public.ccsds.org/default.aspx.
[3] Internet Engineering Task Force (IETF), Official Website: http://www.ietf.org/
[4] S. Hemminger, Network emulation with netem, *Linux Conf Au*, April 2005.
[5] Interplanetary Overlay Network (ION) Design and Operation, Jet Propulsion Laboratory, California Institute of Technology, 2008 URL: https://ion.ocp.ohiou.edu/
[6] DTN2 Reference Implementation, Delay Tolerant Networking Research Group, URL: http://www.dtnrg.org/
[7] CCSDS File Delivery Protocol, Recommendation for Space Data System Standards, CCSDS 727.0-B-2, Blue Book, Issue 2, Washington, D.C.: CCSDS, October 2002.
[8] M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb and J. Lepreau, Large-scale Virtualization in the Emulab Network Testbed, in *Proceedings of the 2008 USENIX Annual Technical Conference*, pp. 113-128, Boston, MA, June 2008.
[9] Defense Advanced Research Technology Network (DARTnet) and Collaborative Advanced Interagency Research Network (CAIRN), URL: http://www.ece.udel.edu/ mills/dartnet.html.
[10] H. Eriksson, MBONE: the multicast backbone, *Communications of the ACM* , Volume 37, Issue 8, pp. 54 - 60, 1994.
[11] H. Soroush, N. Banerjee, M. D.Corner, B. N.Levine and B. Lynn, DOME: A Diverse Outdoor Mobile Testbed, Department of Computer Science, University of Massachusetts, Amherst.
[12] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S.Peh and D. Rubenstein, Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet, in *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, San Jose, California, 2002.
[13] Networking for Communications Challenged Communities: Architecture, Test Beds and Innovative Alliances project (N4C), Official Website: http://www.n4c.eu/Home.htm
[14] K. Scott and S. Burleigh, Bundle Protocol Specification, IETF RFC 5050, experimental, November 2007, URL: http://www.ietf.org/rfc/rfc5050.txt
[15] Satellite Tool Kit Software, Analytical Graphics Inc., Official Website: www.stk.com
[16] V. Cerf et al., Delay-Tolerant Network Architecture, IETF RFC 4838, information, April 2007, URL: http://www.ietf.org/rfc/rfc4838.txt
[17] M. Ramadas, S. Burleigh, and S. Farrell, Licklider Transmission Protocol - Specification, IETF RFC 5326, experimental, September 2008, URL: http://www.ietf.org/rfc/rfc5326.txt
[18] D. Demmer and K. Fall, DTLSR: Delay tolerant routing for developing regions, *ACM NSDR*, 2007.
[19] S. Burleigh, Dynamic Routing for Delay-Tolerant Networking in Space, *Flight Operations SpaceOps Conference*, 2008.
[20] A. Lindgren, Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET), Internet Draft, URL: http://www.dtnrg.org/docs/specs/draft-lindgren-dtnrg-prophet-02.txt

[21] Space Packet Protocol, Recommendation for Space Data System Standards, CCSDS 133.0- B-1, Blue Book, Issue 1, Washington, D.C.: CCSDS, September 2003.
[22] GLADE - A User Interface designer, Official Website: http://glade.gnome.org/
[23] The GTK+ Project, Official Website: http://www.gtk.org/
[24] I. Psaras, G. Papastergiou, V. Tsaoussidis, and N. Peccia, DS-TP: Deep-Space Transport Protocol, *IEEE Aerospace Conference 2008*, Montana, USA, March 2008.
[25] Y. Wang, S. Jain, M. Martonosi and K. Fall, Erasure- Coding Based Routing for Opportunistic Networks, in *Proceedings of the 2005 35ACM SIGCOMM workshop on Delay-tolerant networking*, pp. 229-236, 2005.
[26] TM Space Data Link Protocol, Recommendation for Space Data System Standards, CCSDS 132.0-B-1, Blue Book, Issue 1, Washington, D.C.: CCSDS, September 2003.
[27] TC Space Data Link Protocol, Recommendation for Space Data System Standards, CCSDS 232.0-B-1, Blue Book, Issue 1, Washington, D.C.: CCSDS, September 2003.
[28] AOS Space Data Link Protocol, Recommendation for Space Data System Standards, CCSDS 732.0-B-1, Blue Book, Issue 1, Washington, D.C.: CCSDS, September 2003.

Efthymios Koutsogiannis received a B.Sc. in Computer Science from University of Ioannina, Greece in 2004. He also received a M.Sc degree in Communication Systems and Networks from National and Kapodistrian University of Athens, Greece, in 2006. He is currently a Ph.D. candidate in the Department of Electrical and Computer Engineering, Democritus University of Thrace, Greece. His research interests lie in the area of Delay Tolerant Networking, mainly in transport protocols and mechanisms.

Sotirios Diamantopoulos obtained a Diploma in Electrical and Computer Engineering from Democritus University of Thrace, Greece in 2008, where he is currently a M.Sc student. He also participates in the Extending Internet into Space, ESA-funded project. His main research interests include reliability and routing mechanisms in delay-tolerant networks.

Georgios Papastergiou received his diploma in Electrical and Computer Engineering from Democritus University of Thrace, Greece in 2005. He also received his M.Sc degree in Informatics from Aristotle University of Thessaloniki, Greece in 2007. He is currently working toward the Ph.D degree in the Department of Electrical and Computer Engineering, Democritus University of Thrace, Greece. His current research interests are in the area of transport protocols for deep space communication networks.

Ioannis Komnios received his Master Degree in Computer Networks from Democritus University of Thrace, Greece, in October 2009. Since October 2008 Ioannis Komnios has been participating in the ”Extending Internet into Space - Phase II” ESA-funded project. His main research interests include reliability conflict among different network layers and routing in delay-tolerant networks.

Aggelis Aggelis obtained his BSc in Electrical and Computer Engineering from Democritus University of Thrace, Greece; and MSc in Telecommunications From Democritus University of Thrace. He is currently a PhD candidate in Electrical and Computer Engineering in Democritus University, Greece. Aggelis is currently working as a Telecommunications Engineer in Hellenic Aerospace Industry. His research interests include DVB-RCS, embedded computing and signal processing.

Nestor Peccia Nestor Peccia is the Head of the Data Systems Infrastructure Division, Department of Engineering at the Operations Centre of the European Space Agency (ESA / ESOC), in Darmstadt, Germany. Also, he currently serves as the Deputy Chairman CCSDS Engineering Steering Group and Mission Operations and Information Management Services (MOIMS) Area Director and as a Chairman of the European Technology Harmonisation on Space Ground Software Systems.