Ad Hoc Networks 27 (2015) 147–158

Contents lists available at ScienceDirect

Ad Hoc Networks

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

The Interplanetary Internet implemented on a terrestrial testbed ☆

Joyeeta Mukherjee, Byrav Ramamurthy *

Department of Computer Science and Engg., University of Nebraska–Lincoln, Lincoln, NE 68588-0115, USA

ARTICLE INFO

Article history:
Received 11 April 2014
Received in revised form 18 September 2014
Accepted 30 December 2014
Available online 9 January 2015

Keywords:
Interplanetary Internet
Interplanetary Overlay Network
ORBIT testbed

ABSTRACT

Future space exploration demands a space network that will be able to connect spacecrafts with one another and in turn with Earth's terrestrial Internet and hence efficiently transfer data back and forth. The feasibility of this technology would enable everyone to directly access telemetric data from distant planets and satellites. The concept of an Interplanetary Internet (IPN) is only in its incubation stage and considerable amount of common standards and research is required before widespread deployment can occur to make IPN feasible.

Delay is an important factor in space communication and its nature is completely different from terrestrial delay environments. Moreover, space deployments and testing in space environments are very costly and time consuming. We propose a design of the IPN and implement it with the Interplanetary Overlay Network (ION) software module on physical nodes on the terrestrial ORBIT testbed. Two space network scenarios are designed and experimentally evaluated to verify the correctness of the network implementation. We also focus on the study of bundle transmission delay and separately evaluate the effect of bundle size and number of bundles. The experimental evaluation provides insights into the factors which caused delay in bundle transmission such as custody refusal, expiration of bundle lifetime and congestion.

1. Introduction

Space communication and networking research has added a new engineering and scientific era to the history of space exploration. The early phase of space communication used radio signal shot towards spacecraft antennas whenever they came into view. Telecommunications software lacked universality and differed from one mission to another. This, in turn, led individual flight projects to acquire and operate their own specialized space communication networks. To overcome these problems we need to develop a space network that can be interconnected, standardized and evolved over the future decades and such motivations led to novel networking architectures and technologies that could support space communication networks – such as the Interplanetary Internet (IPN).

The IPN is a store-and-forward network of Internets in support of a deep space exploration that is often disconnected, has a wireless backbone with error-prone links and delays ranging to tens of minutes, even hours, even when there is a connection. Moreover, there are a number of physical phenomena in outer space such as solar storms and magnetic interferences which interrupt normal communication between spacecrafts. Moreover, the spacecrafts which are farther away from Earth have back dated technology than the ones launched recently. The existing

☆ An earlier version was presented at 2013 IEEE International Conference on Communications (ICC) [1].

* Corresponding author. Tel.: +1 402 472 7791.
E-mail addresses: joyeetam@cse.unl.edu (J. Mukherjee), byrav@cse.unl.edu (B. Ramamurthy).

http://dx.doi.org/10.1016/j.adhoc.2014.12.007
1570-8705/

terrestrial Internet and the TCP/IP suite will not be able to handle the constraints posed by such extreme conditions. These critical issues are the motivation [2] for our work and thus we propose and study a future deep space network architecture which will survive such extreme conditions. The remote networks of the solar system such as the Earth based Internet and other planetary networks support various protocols and they hook up to the IPN backbone by choosing among mobile satellite gateways that would seamlessly convert between these protocols [3,4].

Delay Tolerant Networking (DTN) is viewed as an overlay network on top of such regional planetary networks. It incorporates a new protocol layer called as the bundle layer on top of heterogeneous region specific lower layers. The NASA Jet Propulsion Laboratory (JPL), has developed the Interplanetary Overlay Network (ION) to implement DTN in Interplanetary environments [5–7]. It is open source, modular, easy to modify and we can also plug in our own routing protocol. It implements the Bundle Protocol as in [8] along with the CCSDS File Delivery Protocol (CFDP) [9] and the Licklider Transport Protocol (LTP) found in IRTF RFCs 5325 [10], 5326, and 5327. Three experiments on space DTN were recently conducted, namely – the UK-Disaster Monitoring Constellation (UK-DMC) Experiment (2008), the Deep Impact Network Experiment (DINET) (2008) and Experiment on-board the International Space Station (ISS) (2010) and all these experiments were done in space. Considering the critical issues and cost for space deployments and testing, we propose designs of IPN deployments on the Open Access Research Testbed for Next-Generation Wireless Networks (ORBIT testbed) so as to examine the network and its operations more easily.

2. Interplanetary Overlay Networks

The Interplanetary Overlay Network (ION) [11] is a product of the Jet Propulsion Laboratory (JPL) to implement DTN in Interplanetary environments [12] and an overview of its functional dependencies are shown in Fig. 1. ION uses a simple header compression scheme to improve transmission efficiency called the Compressed Bundle Header Encoding (CBHE) scheme and it is database centric unlike its predecessor DTN2 [13]. Fragmentation and reassembly is also well taken care of in the ION infrastructure. To minimize transmission overhead and to accommodate asymmetric links in an IPN, ION wants to send downlink data in the largest possible aggregations and termed it as coarse-grained transmission. But again to minimize head-of-line blocking (delay in transmission of a newly presented high-priority item) and data delivery latency by using parallel paths (i.e., to provide fine-grained partial data delivery, and to minimize the impact of unexpected link termination), ION sends downlink data in the smallest possible aggregations and has termed it as fine-grained transmission. ION achieves both these functions at two different layers of the software stack – the application layer and at the BP/LTP convergence layer. At the application layer small Application Data Units (ADUs) are generated on the order of 64 Kb which enables fine grained transmission. Lower in the stack at the BP/LTP convergence layer the small bundles which are of similar kind (i.e.,same priority level or destined for the same LTP engine) are aggregated into single blocks for delivery. This enables coarse-grained transmission. ION also supports Contact Graph Routing (CGR) which is a dynamic routing scheme used to compute the routes through a time varying topology of scheduled communication contacts in an IPN. Each node builds a contact graph data structure from the range and contact timeline entries and uses it to make the routing decisions. ION implements the concept of One Way Light Time (OWLT) which is the time taken for an electromagnetic signal to travel one way between Earth and a spacecraft or some other celestial body. The node architecture and processing within the node has been further elaborated in [11]. Currently the Delay Tolerant Networking Research Group (DTNRG) is in an effort to establish a worldwide collection of nodes running Bundle Protocol Agents (BPAs) that represent the DTN implementations of DTN2 and the ION [14].

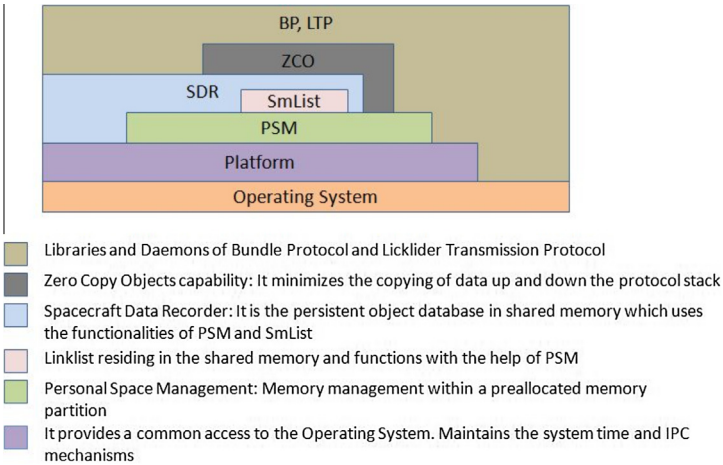

Fig. 1. ION software functional dependencies.

3. ORBIT testbed

The ORBIT(Open Access Research Testbed for Next-Generation Wireless Networks) testbed is a flexible wireless network testbed, supported by NSF and located at WINLAB in Rutgers University. It contains an indoor radio grid of 400 nodes arranged in a 20 by 20 matrix as shown in Fig. 2. The ORBIT testbed allows users to set the various properties and parameters of the wireless network and capture all dependencies and most environmental conditions (especially complex radio link layer issues) to facilitate repeatable experiments. It even allows users to remotely control the experiments. ORBIT may be viewed as a set of services where we can input experimental definitions and receive the experimental results as output. Each ORBIT Radio Node is a real PC sitting at WINLAB with a 1 GHz VIA C3 processor, 512 Mb of RAM, 20 GB of local disk, two 100BaseT Ethernet ports, two IEEE 802.11 a/b/g cards and a Chassis Manager to control the node. Fig. 3 shows the flow of experiment execution from the user's point of view and how the user can interact with the testbed. The Nodehandler component functions as an Experiment Controller and multicasts 30 commands to the nodes at the right time and keeps track of the experiment flow. Nodeagents reside at each node which then executes these commands and also reports back to the Nodehandler. With the help of these two components the user controls the testbed and enables automated collection of experimental results through the ORBIT Measurement Framework and Library (OML) [15]. It uses a SQL database to persistently store and retrieve these results for analysis as and when required.

Fig. 2. Orbit testbed (from [16]).

For experiment setup we choose a group of nodes on the ORBIT testbed. In ORBIT testbed, a grid is a set of nodes together with the controlling console which can be used to run experiments. The testbed consists of a single large grid (main grid) with 400 nodes and an array of sandboxes i.e. "grids" with only 2 nodes and a console, which are development and test environments intended to reduce the time experimenters need on the main grid. Ideally, experimenters develop their software on off-site machines and then use the sandboxes for integration with the orbit environment and orbit software infrastructure. Once the experiment runs successfully in the sandbox environment, it can be moved to the main grid. For our experiment we randomly selected a group of nodes on the ORBIT testbed. We used sandbox-9 as our testing environment. Among the nodes of sandbox-9, any single node is chosen and tailored to support the DTN architecture. The node runs Linux operating system on which we install the ION software module. The Expat XML parser library written in C language is also required to support ION execution. Once the system is ready to implement the DTN architecture we save the system image on the remote testbed. Later we load the saved image on the other chosen nodes in the group. We need to modify the config.rc file of each node to set up the network topology. For DTN-based space networks ad-hoc information discovery is costly and it becomes backdated by the time the information is communicated to a second node.

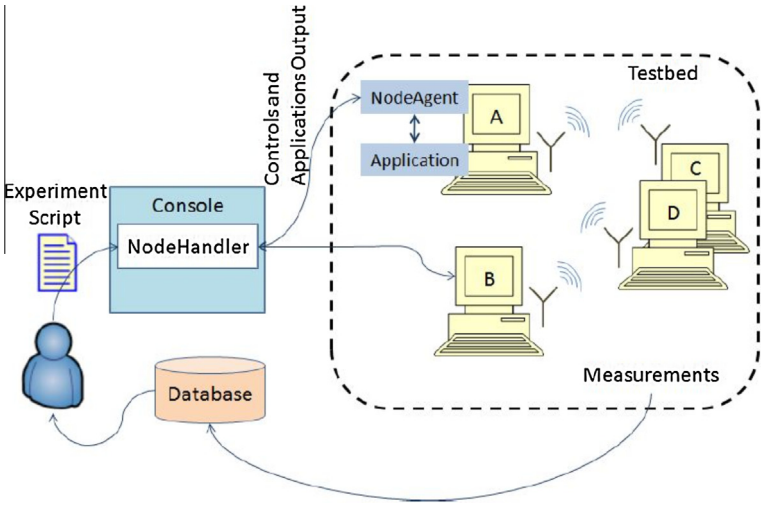

Fig. 3. Execution of an ORBIT experiment from the User's point of view.

It is more effective to pre-place information at the network nodes tagged with the dates and times. This is done in the configuration file by implementing egress plans and contact graphs.

ION can only forward bundles to a neighboring node by queuing them on some explicitly specified outduct. Specifications that associate the neighboring nodes with these outduct comprise the egress plan. Contact graphs on the other hand specify the time duration during which a transmitting and receiving node remains in contact. Each contact is characterized by its start time, its end time, the identities of the transmitting and receiving nodes, and the rate at which data is expected to be transmitted by the transmitting node during the indicated time period. Finally, we specify a range which is the time period during which the displacement between two communicating nodes is expected to vary by less than 1 light second from a stated anticipated distance.

The ION configuration file is a mixture of different configuration files like ionadmin, ltpadmin, bpadmin, ipnadmin and dtnadmin. The ionadmin configuration file specifies contact bandwidths and one-way transmission times. The ltpadmin configuration file specifies spans, transmission speeds, and resources for the Licklider Transfer Protocol convergence layer. The bpadmin configuration file specifies all of the open endpoints for delivery on your local end, which convergence layer that you intend to use. The ipnadmin configuration file maps endpoints to convergence-layer addresses. The dtnadmin configuration file acts as the routing table for the DTN naming scheme. A sample example of the configuration file is shown in Table 1 where two nodes communicate with one another using User Datagram Protocol (UDP). This file is made of ionadmin, bpadmin and ipnadmin configuration files. In the ionadmin section, we specify the contact plan and the range between nodes. In the sample configuration file, we add a range between node 1 and node 2. Node 1 will start connecting to Node 2 at +0 s from now and will end at +600 s from now. Data on the link is expected to take 1 s to reach the other end (One Way Light Time). We also added a contact plan for both the nodes. This plan states that each node will be in contact with the other node starting from +0 s to +600 s and will transmit data at the rate of 10,000 bytes per second. In the bpadmin section, we add the Endpoint Identifier (EID) scheme, endpoints, induct and outduct. We also add the protocol that is used for communication. In the ipnadmin section we add an egress plan so that bundles are transmitted from node 1 to node 2. Implementing these specifications help us to build the network topology. The design of the network is developed from observation of ephemeris data [17,18]. Tables 2 and 3 show a sample of data subset for Mars Reconnaissance Orbiter (MRO) over a specific period of time while Table 4 shows the ephemeris data label. From this information we can retrieve the position and velocity vectors of a given celestial structure relative to its center of motion and a reference frame.

Table 1
Configuration File for 2 Node Communication using UDP.

```
## begin ionadmin
1 1 ''
s
a range +0 +600 1 2 1
a contact +0 +600 1 2 100000
a contact +0 +600 2 1 100000
## end ionadmin
## begin bpadmin
1
a scheme ipn 'ipnfw' 'ipnadminep'
a endpoint ipn:1.1 q
a protocol udp 1400 100
a induct udp 10.19.1.1:4556 udpcli
a outduct udp * udpclo
s
## end bpadmin
## begin ipnadmin
a plan 2 udp/*, 10.19.1.2:4556
## end ipnadmin
```

Orbit Testbed supports ORBIT Management Framework(OMF). OMF is a framework for controlling and managing networking testbeds [19]. OMF defines and uses OMF Experiment Description Language (OEDL), a domain-specific language to describe an experiment. OEDL is based on a ruby scripting language.Our experiment is based on two simple scenarios of Earth to Moon and Earth to Mars communication which have been elaborated in the next section. Once we set up the network topologies we run the experiment for a fixed duration of time, execute applications on an ORBIT node through Ruby script files and collect experimental data.

A sample example of the Ruby script file is shown in Table 5 which starts the ION application on 4 nodes at the same time taking the configuration file at each node as input. In this example, the *defApplication* block is used to define a new application. Here we define and name our application as *startApp*. In this block we provide a short description for our application and provide a location of the config file to the application. The *defGroup* block defines a set of resources to be used in an experiment. Here we define a group of four nodes and name it as *worker*. In this block we use a sub-command *addApplication* to install and configure the instance of the above defined application on the four nodes. We use the *whenAllUp* block to run a set of commands when all the resources in the experiment are up and running. In this block we start all the applications which are associated with the resources in all the defined groups.

The Ruby script file helps in creating a wrapper class around the application ionstart. After the ION application is started we wait for a few seconds for the nodes and the network to prepare itself for communication. Other ION applications like *bpsource*, *bpsink* and *bpsendfile* are then used for bundle transmission.

4. Network scenario

In this section we discuss the problem of setting up a space network in a terrestrial environment and then varying the network parameters to gain an insight into the setup. We design and implement two different network scenarios to study the problem and measure the delay

Table 2
Ephemeris data sample for MRO.

Structure	ID	Center	Frame
MRO_HGA_OUTER_GIMBAL		−74,213	−74,000
MRO_SPACECRAFT	MRO_HGA	−74,214	−74,213
MRO_HGA_OUTER_GIMBAL	MRO_LGA1	−74,220	−74,213
MRO_HGA_OUTER_GIMBAL	MRO_SAPX_OUTER_GIMBAL	−74,313	−74,000
MRO_HGA_OUTER_GIMBAL	MRO_SAPX_C1	−74,315	−74,313
MRO_HGA_OUTER_GIMBAL	MRO_SAMX	−74,324	−74,323
MRO_HGA_OUTER_GIMBAL	MRO_SAMX_C1	−74,325	−74,323

Table 3
Ephemeris data sample for the different MRO structures.

	Name	Relative to	Type	Naif ID
Non Built-in Mars Frames	MRO_MME_OF_DATE	rel.to J2000	DYNAMIC	−74,900
Spacecraft frame	MRO_SPACECREAFT	rel.to MME_OF_DATE	CK	−74,900
Science Instrument frames	MRO_CRIM_BASE	rel.to SPACECRAFT	FIXED	−74,011
Antenna frames:	MRO_HGA_BASEPLATE	rel.to SPACECRAFT	FIXED	−74,211

Table 4
Ephemeris data label.

PDS_VERSION_ID	=PDS3
RECORD_TYPE	=FIXED_LENGTH
RECORD_BYTES	=1024
SPICE_KERNEL	=“mro_sc_psp_101214_101220.bc”
MISSION_NAME	=“MARS_RECONNAISSANCE_ORBITER”
SPACECRAFT_NAME	=“MARS_RECONNAISSANCE_ORBITER”
DATA_SET_ID	=“MRO-M-SPICE-6-V1.0”
KERNEL_TYPE_ID	=CK
PRODUCT_ID	=“mro_sc_psp_101214_101220.bc”
PRODUCT_CREATION_TIME	=2012-06-05T20:57:04
PRODUCER_ID	=“NAIF/JPL”
MISSION_PHASE_NAME	=“PRIMARY SCIENCE”
PRODUCT_VERSION_TYPE	=ACTUAL
PLATFORM_OR_MOUNTING_NAME	=“NA”
START_TIME	=2010-12-18T12:28:28
STOP_TIME	=2010-12-19T12:28:28
SPACECRAFT_CLOCK_START_COUNT	=“2/0860010026.245”
SPACECRAFT_CLOCK_STOP_COUNT	=“2/0860025679.058”
TARGET_NAME	=MARS
INSTRUMENT_NAME	=“MRO_SPACECRAFT”
NAIF_INSTRUMENT_ID	=−74,000
SOURCE_PRODUCT_ID	=“N/A”
NOTE	=“See comments in the file for details”
OBJECT	=SPICE_KERNEL
INTERCHANGE_FORMAT	=BINARY
KERNEL_TYPE	=POINTING
DESCRIPTION	=“MRO SPICE CK file providing actual telemetry-based orientation of the MRO spacecraft bus for a part of the Primary Science phase of the mission, created by NAIF, JPL.”
END_OBJECT	=SPICE_KERNEL

of transmitting a bundle from source to destination. The first scenario implements a network from Earth to Moon and the second scenario is a network from Earth to Mars. We keep alive both the scenarios for a specific duration and transmit bundles from the source to the destination while dynamically changing the topology from time to time.

4.1. Scenario 1: Lunar Mission

The Lunar network scenario is shown in Fig. 4 where Node 1 represents the Mission Control Center (MCC) – an entity managing aerospace flight operations from the point of launch to the end of a mission. It is an Earth based node and all telemetric data is destined for this node after it is

Table 5
Ruby script file to start ION application.

```
defApplication('ionstartWrapper','startApp'){|app|
app.shortDescription = ''This is a simple wrapper
  application around ionstart"
app.path = ''/root/ion-open-source/ionstart -I/root/
  config.rc"
}
defGroup('worker',[[1,1],[1,2],[1,3],[1,4]]){|node|
node.addApplication('ionstartWrapper')
}
whenAllUp(){|node|
wait10
allGroups.startApplications
wait20
Experiment.done
}
```

received by the DSN stations. Nodes 2 and 3 act as DSN stations at Goldstone and Madrid respectively. Finally, Node 4 represents the Shackelton station on the south pole of Moon. Any information either scientific data or image collected by Node 4 is sent through the DSN stations to Node 1. We consider One Way Light Time (OWLT) from Earth to Moon as 1.28 light seconds which is an average of the OWLTs for Earth to Moon communication at perigree (closest = 360,000 km) and at apogee (farthest = 405,000 km). The uplink datarate from Earth to Moon is set as 1 Kbps while the downlink datarate is 128 Kbps. The network remains alive for 10 min. The contact graphs over this period are shown in Fig. 5. For the first 5 min the Madrid DSN station is not in sight with the Moon Shackelton station. It is in Line of Sight (LoS) with the Goldstone DSN station. The Earth rotates and after 5 min the Goldstone station moves out of LoS while Madrid comes into view. These two contact graph snapshots are fed into all the 4 nodes which gives them the network topology so as to construct the dynamic route, which a bundle should take during transmission.

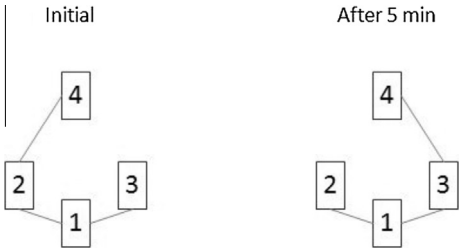

Fig. 5. Contact graphs for Scenario 1: Lunar Mission.

4.2. Scenario 2: Mars Mission

The Mars network scenario is shown in Fig. 6 and it consists of 6 nodes. Nodes 1, 2 and 3 are the same as Scenario 1. Node 4 represents a Mars orbiter called Mars Atmosphere and Volatile EvolutioN (MAVEN). It was part of NASAs' Mars Scout 2013 mission. Node 5 represents the Mars Reconnaissance Orbiter (MRO). Finally, Node 6 represents a Mars rover called the ExoMars which is an autonomous six-wheeled terrain vehicle planned for a future robotic mission to Mars. Nodes 2 and 3 act as the major elements in the network to receive all telemetric signals sent from the Mars orbiters i.e., nodes 4 and 5 and then they are sent to the MCC Node 1. The OWLT for Earth to Mars communication is taken as an average of 324 light seconds. It varies as Mars moves to the

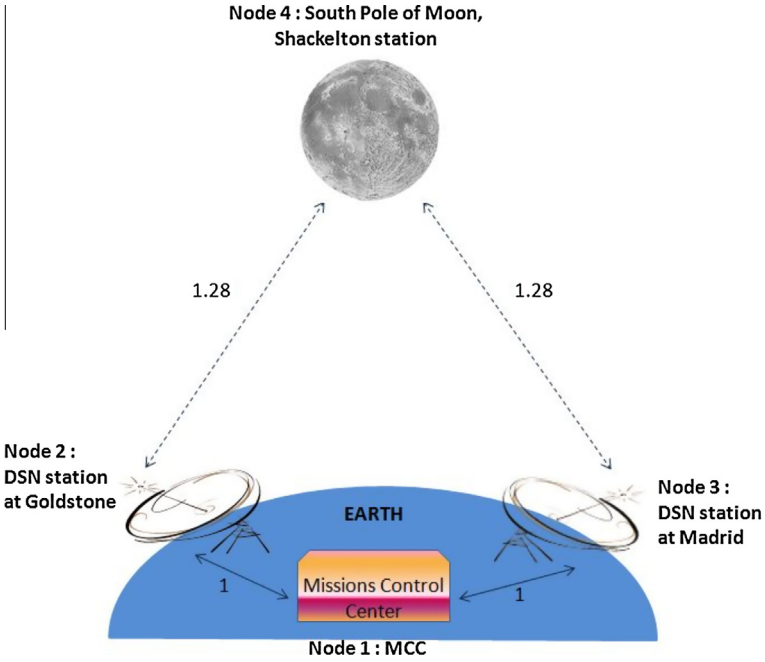

Fig. 4. Scenario 1 depicting the Earth to Moon communication configuration.

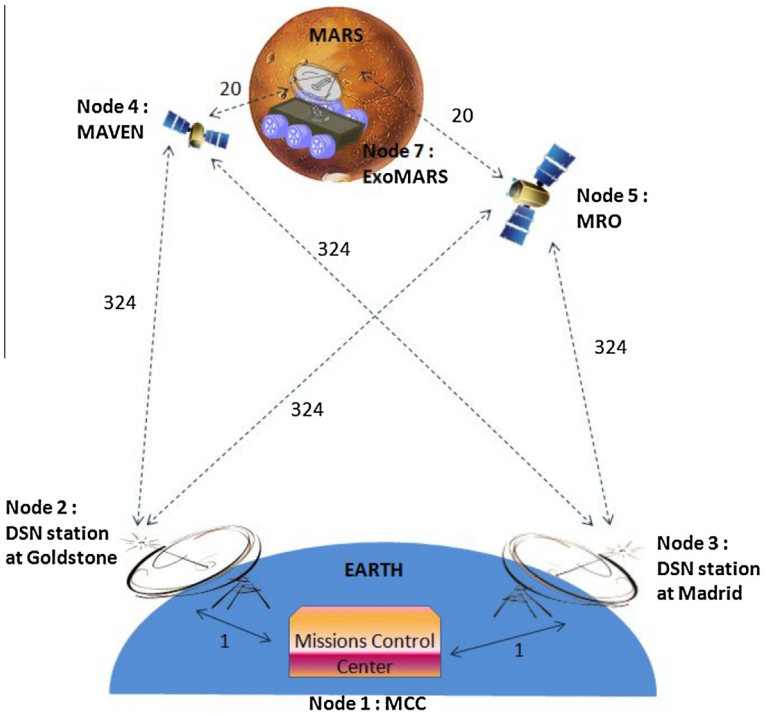

Fig. 6. Scenario 2 depicting the Earth to Mars communication configuration.

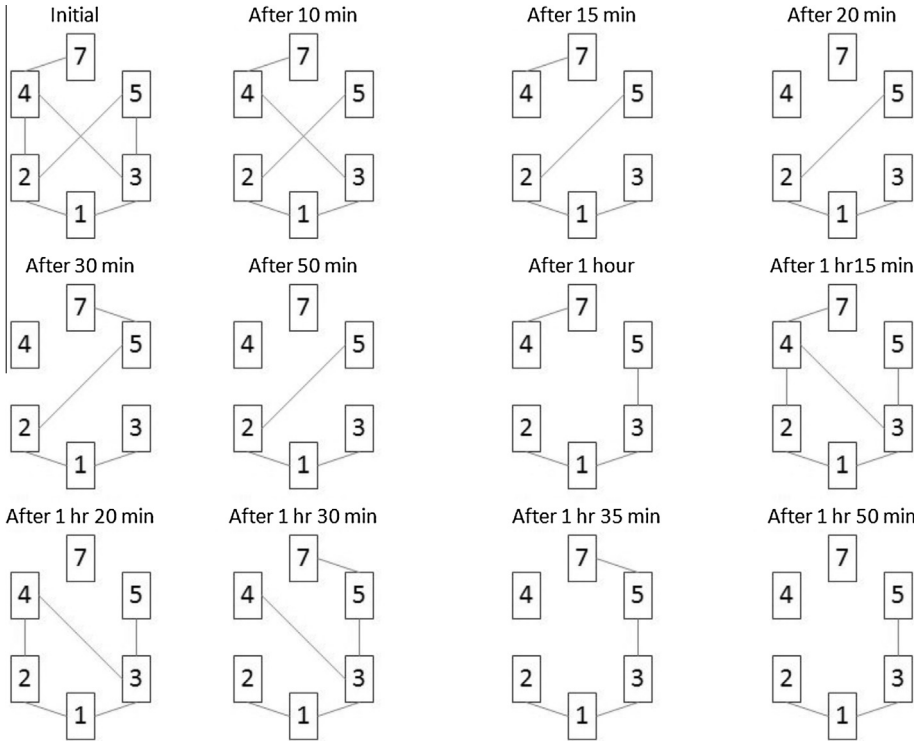

Fig. 7. Contact graphs for Scenario 2: Mars Mission.

farthest and closest to Earth. The OWLT between a Mars orbiter and the Mars surface is taken as 20 light seconds and the communication time between MCC and DSN stations is a maximum of 1 light second which is the normal terrestrial Internet speed. The communication datarates are as follows:

- Mars rover ↔ Mars orbiter = 16 Kbps.
- Earth → Mars = 1 Kbps.
- Mars → Earth = 128 Kbps.

The network is alive for a duration of 2 h. Fig. 7 shows the contact graphs over this time period. Communication becomes a challenge [20] as both Earth and Mars rotate and bringing the celestial bodies in LoS is difficult. Furthermore the Mars orbiters have their own orbital periods – MAVEN has an orbital period of 4.5 h and MRO has an orbital period of 35 h. Keeping these numbers and ephemeris data in mind we design the contact graphs so that within the small time duration of 2 h each node has a chance to be part of a route for bundle transmission.

5. Experimental results

5.1. Verifying network operation

In this section, we present the network statistics that help us to understand IPN and its implementation. We

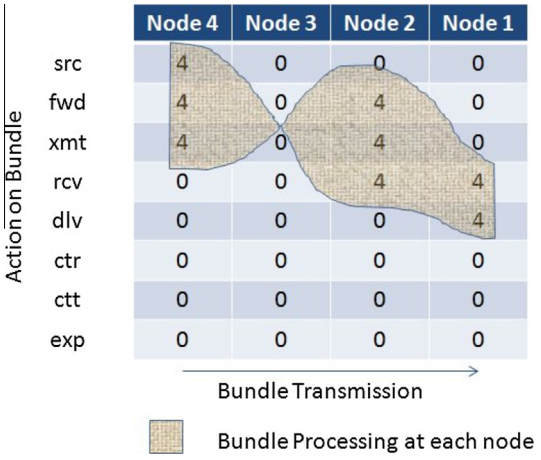

Fig. 8. Test A: network statistics for transmitting 4 bundles.

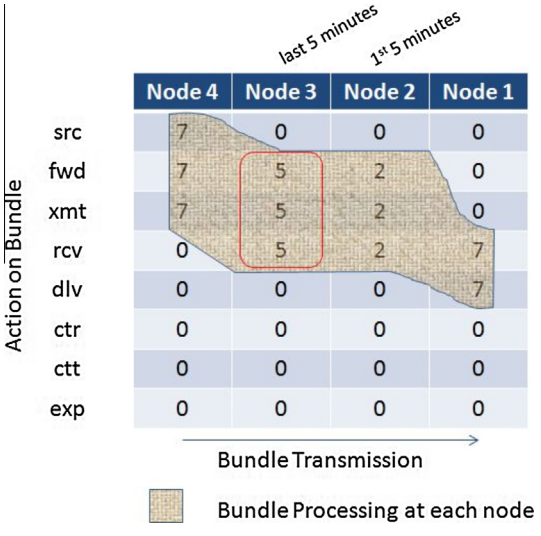

Fig. 9. Test B: network statistics for transmitting 7 bundles.

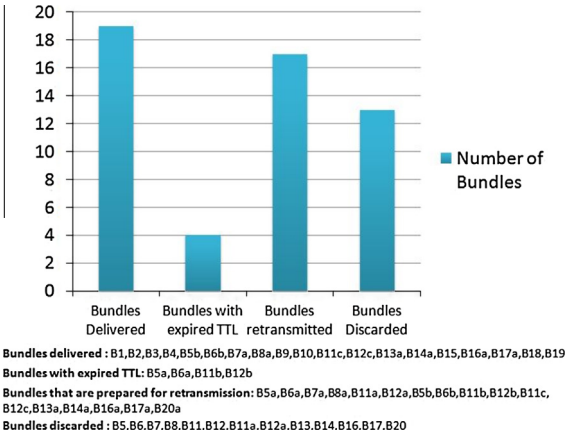

Fig. 10. The status of the bundles in the network and their corresponding counts.

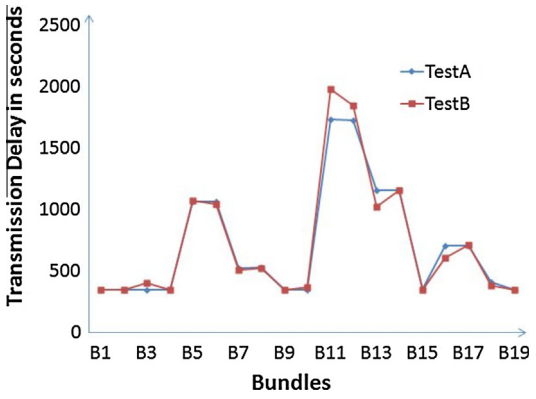

Fig. 11. Transmission delay of bundles through the DTN.

run Scenario 1 for a duration of 10 min over 2 tests. In the first Test A, 4 bundles are sent from the source Node 4 to the destination Node 1 during the first 5 min. After that the network is left idle for the next 5 min. In the second Test B, 7 bundles are sent from the source Node 4 to the destination Node 1. In the first 5 min we send 2 bundles and in the last 5 min we send 5 more bundles. After running the tests the statistics are drawn from both the networks (Earth–Moon, Earth–Mars) and they are shown in Figs. 8 and 9 respectively.

The network statistics gives the bundle transmission route and 8 different types of network activities as shown in Table 6 and as described in [2]. The figures verify the contact graphs of the Lunar Mission and the correctness of network operation. We also notice that for Test A the bundle processing at Node 3 is zero while the bundle processing at Node 4 and Node 2 are equal and higher than that required at Node 1. Similarly if we interpret the Figure for Test B, the red border around Node 3 indicates a higher processing requirement among the two intermediate nodes. Resources in space networks are very expensive. Hence we try to gain an insight into what factors lead to the exhaustion of a node. We can clearly see that a balance is needed on the number of bundles sent over alternative

routes and it also requires a knowledge of the contact times over a given network. Appropriate load balancing with contact graph knowledge can increase the lifetime of the network. However, there are also other factors that might affect the balance such as spacecraft size, link capacity and asymmetric contact durations at different parts of the network, which we do not look into. In the next section we study the bundle transmission delay based on Scenario 1 and Scenario 2.

5.2. Studying bundle transmission delay

A bundle is transmitted from source to destination over a given route based on contact graph snapshot at that instant of time. In Delay Tolerant Networks a bundle may leave the source for its destination but there might be a disconnection in the middle of the network. In such conditions ION has the provision to store and take custody of the bundle at the local node until a connection is established. In our second scenario we try to study the delay in bundle transmission over a duration of 2 h during which network partitions are created. The bundle lifetime or TTL (Time to Live) is set to a value of 600 s. A total of 20 bundles are sent over the duration of 2 h and the bundles to corresponding time slots are shown in Table 7. Each bundle is sent from the source Node 7 to the destination Node 1.

After running the experiment we show the status of the bundles in the network along with their corresponding counts in Fig. 10. We find that there are a total of 19 bundles that are successfully delivered to the destination. Due to the disconnection of the source node from the network in the last time slot as shown in the contact graph, the 20th bundle remains undelivered. Transmission of a bundle through the DTN can fail mainly due to two reasons - contact failure and custody refusal. Situations may arise where a contact between two nodes does not occur for a long time period during which the bundle TTL expires or a bundle cannot be transmitted within the set contact period of the two communicating nodes. For both cases the bundle is removed from its outbound transmission queue and the Dynamic Route Computation Algorithm is re-applied to the bundle so that an alternate route can be computed. For our experiment bundles which undergo this kind of failure are given a small alphabet version (such as B5a) when its route is recomputed. There are a total of 17 bundles whose route needs to be recomputed. However, a node might also refuse custody of a bundle because it finds it impossible to forward the bundle. Such bundles are simply discarded, but discarding any such bundle that is marked for custody transfer will cause a custody refusal signal to be returned to the bundle's current custodian. Again the bundle route needs to be recomputed.

Fig. 11 shows the delay in transmission for each bundle. We perform two tests – Test A and Test B on Scenario 2. All parameters for both the tests are kept same. We get similar results except for a few differences which is accountable to the multiprocessing capability of a node leading to different bundle processing times. However, in both the tests we find that the bundles 11 and 12 take the maximum time to reach the destination. It is almost 1732.3/345.69 = 5.011 times the nominal delay for the network. It is a consequence of these two bundles being repeatedly discarded at different times in different parts of the network.

Figs. 12–16 demonstrates the delay in bundle transmission based on a number of different factors. We now use Scenario 1 to study the bundle transmission delay. Two new applications are developed, implementing delay calculation and bulk bundle transmission. They are named as *send* and *recv* for sending and receiving bundles respectively and are designed to take the source and destination eids as input. Calculating delay seems to be quite challenging since it requires the whole network to be synchronized. We achieve this by writing a script to synchronize the whole network through ORBIT console. As shown below we subtract the sum of the bundle's creation time (provided in the BpDelivery structure that gets populated when we call bp_receive function in the recv application) and 946684800 value (required to convert from the internal DTN Epoch 2000 time to Unix epoch time), from the current time at the moment bp_receive function returns (as provided by the getUTCTime function in the ici library of

Table 6
Description of different network activities for an ION implementation.

src	The measure of the number of bundles for which the local node is the source
fwd	The number of bundles forwarded from the local node. Re-forwards due to custody transfer timeouts are also counted
xmt	The number of bundles passed to the convergence layer protocol(s) for transmission from this node. Re-forwards due to custody transfer timeouts are also counted as retransmissions at the convergence layer
rcv	The measure of the number of bundles received by the local node
dlv	The number of bundles delivered to applications via endpoints on the local node
ctr	This message reports on the custody refusal signals received at the local node
ctt	The number of custodial bundles for which convergence-layer transmission failed at this node
exp	The number of bundles destroyed at this node due to TTL expiration

Table 7
Bundles sent at different time slots (in hours:minutes) for Scenario 2: Mars Mission.

0–10	10–15	15–20	20–30	30–50	50–60	1–1:15	1:15–1:20	1:20–1:30	1:30–1:35	1:35–1:50	1:50–2
B1	B3	B5	B7	B9	B11	B13	B15	B16	B18	B19	B20
B2	B4	B6	B8	B10	B12	B14		B17			

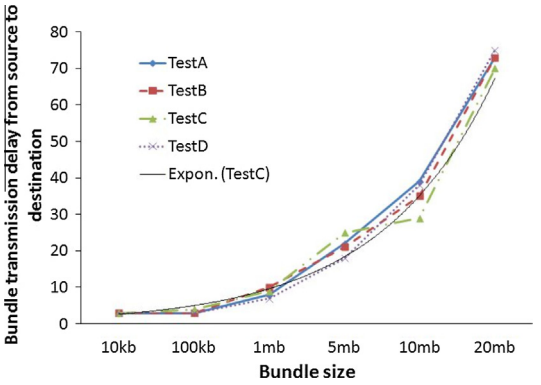

Fig. 12. Bundle transmission delay with varying bundle size.

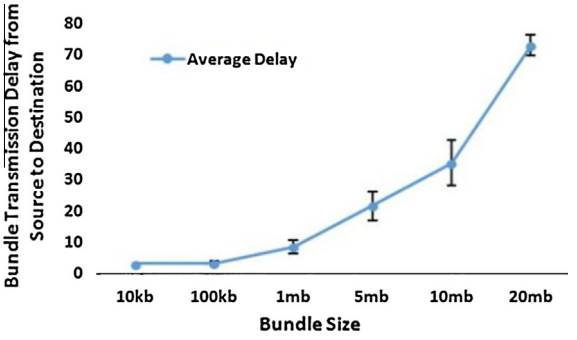

Fig. 13. Average bundle transmission delay with varying bundle size along with 95% confidence interval.

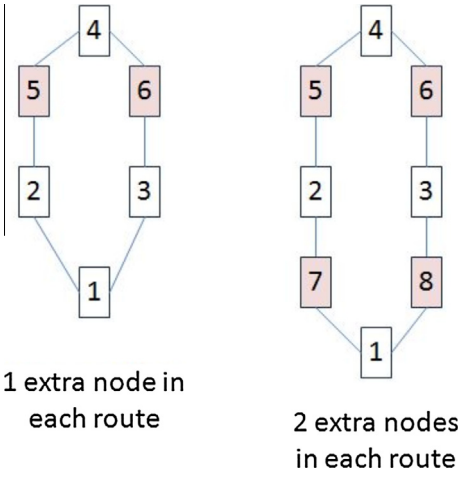

Fig. 14. Variation in Scenario 1 by adding extra nodes.

ION software module). This finally gives us the time taken by a bundle to reach from the source to the destination in seconds.

In Fig. 12 we perform 4 tests – Test A, Test B, Test C and Test D with the same environmental parameters and topology while varying the size of the bundle from 10 Kb to 20 Mb. We can clearly notice the increase in bundle transmission delay time with increasing bundle size. Henceforth, we always take the average of 4 tests, to obtain a single delay value as shown in Fig. 13. This removes the unpredictability due to node multiprocessing that was observed for Test A and Test B in Fig. 11.

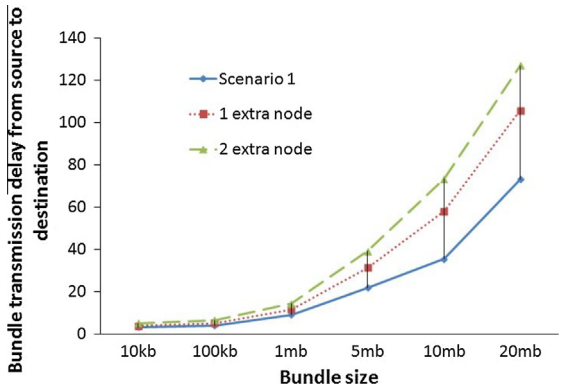

Fig. 15. Bundle transmission delay with varying bundle size along with the implementation of extra nodes.

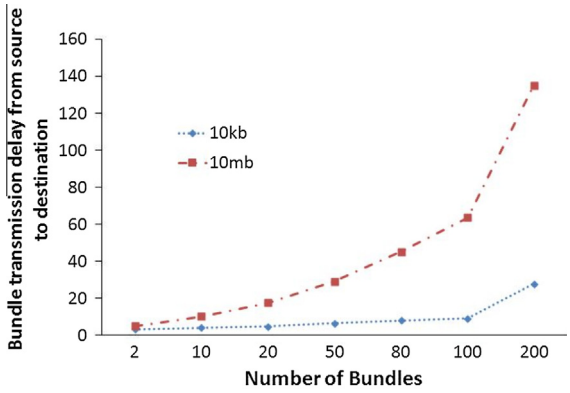

Fig. 16. Transmission delay of bundles through the DTN.

In our next experiment we introduce a variation in Scenario 1 topology as depicted in Fig. 14. An extra node – Node 5 and Node 6 is included in each of the routes, 4-2-1 and 4-3-1 respectively. In another variation we include 2 extra nodes in each route – nodes 5, 7 and nodes 6, 8 respectively. The contact graph routes remain the same as in Scenario 1 over the duration of 10 min.

The plot in Fig. 15 shows the delay in bundle transmission based on bundle size which is varied from 10 Kb to 20 Mb. It is seen that the bundle transmission delay gradually increases as more number of nodes are added to the topology. An average increase of 43.25% is noticed when we add 1 extra node to the route and about 87.92% when we add 2 extra nodes to the route. This clearly depicts the effect of increasing hops on the delay of bundle transmission.

In order to evaluate the impact of the increase in number of bundles on the transmission delay, we repeat the

experiment of Scenario 1 with varying number of bundles while keeping the bundle size fixed at 10 Kb and 10 Mb. Fig. 16 shows us the data plot. We find that for 10 Kb bundles the transmission delay and its variation is low till the number of bundles increase to 100. When the number is increased to beyond 100 to 200 the delay shows a steeper rise. A similar behavior is also noticed for 10 Mb bundles. This can be explained as a result of congestion in the network due to increase in data volume. Congestion in a DTN is the imbalance between data enqueuing and dequeuing rates that results in exhaustion of queuing (storage) resources at a node, preventing continued operation of the protocols at that node [21–23]. ION implement methods to compute congestion forecast and admission control mechanisms. Whenever a congestion is predicted by ION it sets an alarm which prevents further data being pushed from the application layer. Data rate control at the link layer and revised contact plans can help to avert the anticipated resource exhaustion and in turn congestions. Furthermore, we notice that the delay takes a steeper rise for 10 Mb bundles than for 10 Kb bundles with increasing number of bundles. Hence, we can conclude that delay is affected by both the bundle size and the number bundles pushed towards the destination.

6. Conclusion

We present an approach for distributed emulation of Interplanetary networks where we implement the Interplanetary Overlay Network (ION) software distribution on real time physical nodes on the ORBIT testbed. Two space network scenarios are designed which help us to emulate the Interplanetary Internet (IPN). We first test the experimental data to verify the correctness of the network implementation and also discuss the bundle processing required at each node. Resources in space networks are very expensive. This discussion gives us an insight into what factors lead to the exhaustion of a node. We next focus on the study of delay in bundle transmission. Disconnection in network leads to almost 5-fold increase in bundle transmission delay as bundles repeatedly get discarded. Finally we study the impact of increasing the number of bundles, increasing the bundle size and deploying extra nodes on bundle transmission delay.

7. Future work

In our work, we tried to address the basic operations of an Interplanetary Internet (IPN). Several other parameters can be brought in for experimental evaluation like storage capability and processing power of a node, varying link capacities, varying bundle priorities, effect of the number of nodes on transmission delay, percentage of time a node dedicates to communication and other celestial parameters. Till date and for long in the future, it is likely for contact plans to remain mostly manual because communication contact planning involves complex mathematics: science operations plans, thermal and power constraints, etc. However, one can work towards making these automated so that a fully dynamic contact plan can be generated at run time. Future work can be done on developing algorithms which would aid in load balancing based on time dependent contact graph knowledge. Future work can also include looking at open issues, such as potential weakness of protocols and sensitivity of delay to various communication parameters.

Acknowledgments

This work was supported in part by an NSF FIA Grant (CNS-1040765) and a NASA Nebraska Space Research Mini-Grant. The authors would also like to acknowledge the useful discussions on relevant topics with Loren P. Clare and Philip C. Tsao from the NASA Jet Propulsion Lab (JPL) in Pasadena California and Saichand Palusa, UNL.

References

[1] J. Mukherjee, B. Ramamurthy, The interplanetary internet implemented on a terrestrial testbed, in: 2013 IEEE International Conference Communications (ICC), 2013, pp. 4298–4303.
[2] J. Mukherjee, Routing Over the Interplanetary Internet. Master's Thesis, University of Nebraska–Lincoln, 2012.
[3] J. Mukherjee, B. Ramamurthy, Communication technologies and architectures for space network and interplanetary internet, in: IEEE Communications Surveys & Tutorials, 2012.
[4] L. Clare, J. Agre, T. Yan, Considerations on communications network protocols in deep space, in: Aerospace Conference, IEEE, 2001.
[5] L. Clare, S. Burleigh, K. Scott, Endpoint naming for space delay/disruption tolerant networking, in: Aerospace Conference, 2010 IEEE, 2010, pp. 1–10, doi: http://dx.doi.org/10.1109/AERO.2010.5446949.
[6] K. Fall, Delay-tolerant network architecture for challenged internet, in: SIGCOMM, 2003.
[7] J. Burgess, B. Gallagher, D. Jensen, B.N. Levine, MaxProp: routing for vehicle-based disruption-tolerant networks, in: Proceedings of 25th IEEE International Conference on Computer Communications, INFOCOM 2006, 2006, pp. 1–11, doi: http://dx.doi.org/10.1109/INFOCOM.2006.228.
[8] K. Scott, S. Burleigh, Bundle Protocol (BP). IETF Request for Comments, RFC 5050. <http://tools.ietf.org/html/rfc5050>, 2007.
[9] NASA, CCSDS File Delivery Protocol (CFDP) Part 1: Introduction and Overview. CCSDS 720.1-G-3, Green Book. <http://public.ccsds.org/publications/archive/720x1g3.pdf>, 2007.
[10] S. Burleigh, M. Ramadas, S. Farrell, Licklider transmission protocol (LTP) – motivation. IETF Request for Comments, RFC 5325. <http://www.rfc-editor.org/rfc/rfc5325.txt>, 2008.
[11] S. Burleigh, Interplanetary overlay network: an implementation of the DTN bundle protocol, in: 4th IEEE Consumer Communications and Networking Conference, 2007. CCNC 2007, 2007, pp. 222–226, doi: http://dx.doi.org/10.1109/CCNC.2007.51.
[12] I. Psaras, L. Wood, R. Tafazolli, Delay-/Disruption-Tolerant Networking: State of the Art and Future Challenges, Technical Report, 2010. Technical Report, University of Surrey, UK.
[13] K. Nichols, M. Holbrook, R. Pitts, K. Gifford, A. Jenkins, S. Kuzminsky, DTN implementation and utilization options on the international space station, in: SpaceOps 2010 Conference, 2010.
[14] Delay Tolerant Network Research Group, DTNBone. <http://www.dtnrg.org/wiki/DtnBone>.
[15] WINLAB, Rutgers Univ., ORBIT. <http://www.orbit-lab.org/wiki/WikiStart>.
[16] WINLAB, Rutgers Univ., ORBIT Indoor Testbed. <http://www.winlab.rutgers.edu/docs/about/resources.html>.
[17] JPL, NASA, Ephemeris Data. <http://www.ephemeris.com/space-time.html>.
[18] JPL, NASA, (HORIZONS System). <http://ssd.jpl.nasa.gov/horizons>.
[19] T. Rakotoarivelo, M. Ott, G. Jourjon, I. Seskar, OMF: a control and management framework for networking testbeds, ACM SIGOPS Oper. Syst. Rev. (2010)
[20] S. Farrell, V. Cahill, Security considerations in space and delay tolerant networks, in: Second IEEE International Conference on Space Mission Challenges for Information Technology, 2006. SMC-IT

2006, 2006, pp. 8–38, doi: http://dx.doi.org/10.1109/SMC-IT.2006.66.
[21] D. Katabi, M. Handley, C. Rohrs, Congestion control for high bandwidth-delay product networks, SIGCOMM Comput. Commun. Rev. 32 (2002) 89–102.
[22] C. Casetti, M. Gerla, S. Mascolo, M. Sanadidi, R. Wang, TCP westwood: end-to-end congestion control for wired/wireless networks, ACM/Kluwer Wireless Netw. (WINET) 08 (2002) 467–479.
[23] I.F. Akyildiz, G. Morabito, S. Palazzo, TCP peach: a new congestion control scheme for satellite ip networks, IEEE/ACM Trans. Netw. 09 (2001) 307–321.

Joyeeta Mukherjee graduated with MS at the University of Nebraska–Lincoln (UNL) under the supervision of Dr. Byrav Ramamurthy. She completed her B.Sc. and Masters in Computer Application from the University of Calcutta, India in 2007. From August 2007 to April 2009 she was with HCl Technologies at New Delhi, India working as Software Engineer to test and develop web applications. Currently she is involved with the Mobilityfirst project to design novel concepts for future Internet and working closely with the Optical Communication group. Her research interests also span space communication and routing over Interplanetary networks.

Byrav Ramamurthy is currently a Professor and Graduate Chair in the Department of Computer Science and Engineering at the University of Nebraska–Lincoln (UNL). He has held visiting positions at the Indian Institute of Technology-Madras (IITM), in Chennai, India and at the A&TT Labs-Research, New Jersey, U.S.A. He is author of the book "Design of Optical WDM Networks – LAN, MAN and WAN Architectures" and a co-author of the book "Secure Group Communications over Data Networks" published by Springer in 2000 and 2004 respectively. He has authored over 125 peer-reviewed journal and conference publications. He serves as Editor-in-Chief for the Springer Photonic Network Communications journal. He was Chair of the IEEE ComSoc Optical Networking Technical Committee (ONTC) during 2009–11. He served as the TPC Co-Chair for the IEEE INFOCOM 2011 conference to be held in Shanghai, China. He is a recipient of the College of Engineering Faculty Research Award for 2000 and the UNL CSE Dept. Student Choice Outstanding Teaching Award for Graduate-level Courses for 2002–2003 and 2006–2007. He has graduated 10 Ph.D. and 40 M.S. students under his research supervision. His research has been supported by the U.S. National Science Foundation (NSF), the U.S. Department of Energy (DOE), the U.S. Department of Agriculture (USDA), National Aeronautics and Space Administration (NASA), AT&T Corp., Agilent Tech., HP, OPNET Inc. and the University of Nebraska–Lincoln (UNL).