JPL Publication 9-2

Disruption Tolerant Network Technology Flight Validation Report

DINET

Ross M. Jones

National Aeronautics and Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

February 2009

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

© 2009 California Institute of Technology. Government sponsorship acknowledged.

Table of Contents

1. Executive Summary 1
2. Introduction 2
3. Validation Objectives and Experiment Design 2
   3.1. Validation Objectives 2
      3.1.1. Terms of Validation 5
      3.1.2. Metric 1 – Path Utilization Rate (U) 7
      3.1.3. Metric 2 – Delivery Acceleration Ratio (G) 7
      3.1.4. Metric 3 – ION Node Storage Utilization 7
      3.1.5. Metric 4 – Multipath Advantage 7
   3.2. Experiment Design 8
      3.2.1. System Level Design 8
      3.2.2. Flight Software 15
      3.2.3. Ground Software 19
      3.2.4. Experiment Operations Center 22
4. Experiment Results 36
   4.1. Findings 36
      4.1.1. Metric 1 – Path Utilization Rate (U) 36
      4.1.2. Metric 2 – Delivery Acceleration Ratio (G) 36
      4.1.3. Metric 3 – ION Node Storage Utilization 36
      4.1.4. Metric 4 – Multipath Advantage 37
   4.2. Trace Bundles 37
   4.3. Anomalies 37
      4.3.1. DTN-Related Investigations 37
      4.3.2. Software Anomalies 40
      4.3.3. Hardware Anomalies 41
      4.3.4. Environmental Anomalies 41
      4.3.5. Procedural Anomalies 41
   4.4. Significance of Results and Comparison to State of the Art 42
      4.4.1. Current Deep Space Communications Methodology 42
      4.4.2. DTN Compared to Current Military Communications State of the Art 43
      4.4.3. Comparison with Terrestrial Internet 44
      4.4.4. Comparison with SSTC-UK-DMC Satellite Test 45
      4.4.5. Significance of Results 45
5. Lessons Learned 47
   5.1. Information Management 47
   5.2. Requirement and Time Management 48
   5.3. Configuration Management and Software 49
   5.4. Testing 49
   5.5. Team Morale 51
6. Future Work 51
   6.1. Work for DINET II 51
      6.1.1. Implementation of Unacknowledged CFDP Enabling Large File Transfers 52
      6.1.2. Implementation of Bundle Security Protocol (BSP) 52
      6.1.3. Demonstration of Dynamic Contact Graph Management 52
      6.1.4. Development of a DTN Bootstrap Function 52
      6.1.5. Inclusion of Additional Nodes in the Experimental Network 52
      6.1.6. Development of Automated FDM Switching 53
      6.1.7. Fixing Issues Remaining from DINET I 53
      6.1.8. Implementation of Extended Priority System 53
   6.2. Work Beyond DINET II 53
      6.2.1. Native AMS on Spacecraft 53
      6.2.2. Network Time Protocols 53

6.2.3. On-Board OWLT Calculation Integration 53
7. Acknowledgements 54
8. References 54
9. Appendix A – Experiment Data 55
10. Appendix B – Acronyms 61

Tables

Table 1 Technology Readiness Levels and DINET Events 2
Table 2 DINET Environmental and Resource Envelopes 4
Table 3 DTN Protocol Envelope 5
Table 4 DINET Experiment Summary 9
Table 5 DINET Topology Experiments 9
Table 6. Uplink Overhead 56
Table 7 Network Capacity 57
Table 8 Experiment Data Delivered 58
Table 9 Multipath Advantage 59
Table 10. Storage Utilization 60

Figures

Figure 1 DINET Topology 8
Figure 2 DINET Topology as Physically Implemented 10
Figure 3 Topology Experiment 1 11
Figure 4Topology Experiment 2 12
Figure 5 Topology Experiment 3 13
Figure 6 Topology Experiment 4 14
Figure 7 ION General Processing Flow 15
Figure 8 Current CFDP Architecture 18
Figure 9 CFDP with Integrated PX and ION 19
Figure 10 CFDP Operational Downlink Dataflow 20
Figure 11 CFDP Operational Uplink Dataflow 21
Figure 12 CFDP Testbed Dataflow 21
Figure 13 EOC Hardware (red boxes) in the PTL Work Area 22
Figure 14 EOC Functional Architecture 24
Figure 15 EOC Network Diagram 25
Figure 16 EOC Software Architecture - Software Overview 26
Figure 17 DINET Physical Network 27
Figure 18 Publisher / Subscriber Software System 28
Figure 19 Publisher Control Flow 29
Figure 20 Subscriber Control Flow 29
Figure 21 EOC to Administrative Node 30
Figure 22 Administrative Node Functionality 30
Figure 23 GUI Functionality 31
Figure 24 Monitor and Control GUI 32
Figure 25 Non-Realtime Query GUI 33
Figure 26 Element Function and Modules 34
Figure 27 Personnel Overview 35

Abstract

In October and November of 2008, the Jet Propulsion Laboratory installed and tested essential elements of Delay/Disruption Tolerant Networking (DTN) technology on the Deep Impact spacecraft. This experiment, called Deep Impact Network Experiment (DINET), was performed in close cooperation with the EPOXI project which has responsibility for the spacecraft. During DINET some 300 images were transmitted from the JPL nodes to the spacecraft. Then, they were automatically forwarded from the spacecraft back to the JPL nodes, exercising DTN's bundle origination, transmission, acquisition, dynamic route computation, congestion control, prioritization, custody transfer, and automatic retransmission procedures, both on the spacecraft and on the ground, over a period of 27 days. All transmitted bundles were successfully received, without corruption. All transmitted bundles were successfully received, without corruption. The DINET experiment demonstrated DTN readiness for operational use in space missions.

1. Executive Summary

In October and November of 2008, the Jet Propulsion Laboratory under contract to NASA, installed and tested essential elements (Bundle Protocol [1] and the Licklider Transmission Protocol [2]) of Disruption Tolerant Networking (DTN) technology on the Deep Impact spacecraft and on nine other computers at JPL. (Note that the terms disruption-tolerant networking and delay-tolerant networking are used interchangeably in network communications research.) This experiment, called Deep Impact Network Experiment (DINET), was performed in close cooperation with the EPOXI project which has responsibility for the spacecraft. (EPOXI is a combination of the names for the two extended mission components: the exosolar planet observations, called Extrasolar Planet Observations and Characterization (Epoch), and the flyby of comet Hartley 2, called the Deep Impact Extended Investigation (DIXI).) At the time, the spacecraft was at a distance of about 15 million miles (24 million kilometers) from Earth. During DINET some 300 images were transmitted from the JPL nodes to the spacecraft. Then, they were automatically forwarded from the spacecraft back to the JPL nodes, exercising DTN's bundle origination, transmission, acquisition, dynamic route computation, congestion control, prioritization, custody transfer, and automatic retransmission procedures, both on the spacecraft and on the ground, over a period of 27 days. All transmitted bundles were successfully received, without corruption, despite several transient unanticipated lapses in service at Deep Space Network (DSN) stations during tracking passes.

DINET can be appreciated through two complementary perspectives. The first is technical significance. Prior to the experiment, the DINET team defined four validation metrics, each of which was achieved during the experiment. The technical significance of DINET relies on the quantitatively defensible and tangible results. In summary, the Bundle and Licklider Transmission protocol elements of DTN were rigorously proven to work, as expected, in the disruptive environment of an interplanetary mission.

The second perspective takes a broader, longer view, concentrating on the strategic significance of DINET. It addresses the broader questions of the importance of the experiment's achievements, in both current and future timescales. The experiment's successful demonstration of the priority-aware relay aspect of DTN offers significant promise for better utilization of existing bandwidth and improved end-user satisfaction. A significant strategic result with both immediate and future consequences is a reduction in the reluctance within the space-flight operations community to host networking technology with a high (if not complete) degree of autonomy. Indeed, the EPOXI spacecraft team itself had to be convinced that the DINET software and operations plan posed no serious threat to the safety of their spacecraft. They later became advocates of DINET's inherent safety.

DINET showed the ability of a space network to exchange data between its constituent nodes with Internet-like automation and the resultant low operations labor costs. As networks grow in complexity, the time and effort needed to manually schedule and coordinate link activity quickly becomes unmanageable; DTN allows space networks to scale without such constraints. In addition, the ability to automatically route information between space vehicles in local proximity without incurring the potentially long one-way light time delays and Earth-based decision cycles of human-managed communications offers the possibility of new types of coordinated science

that qualitatively differ from current capabilities. DTN can help enable cooperative, reactive science functionality for remote spacecraft networks.

2. Introduction

The Deep Impact Network Experiment (DINET) was a technology validation experiment of JPL's implementation of Delay-Tolerant Networking (DTN) protocols. The DINET development produced a version of JPL's implementation of Delay-Tolerant Networking protocols in flight and ground software that is now at technology readiness level (TRL) 8 (Table 1). The DINET software (SW) is of sufficient quality that future flight projects can easily use it at low risk. DINET was implemented on the Deep Impact spacecraft and was closely coordinated with the EPOXI project. DINET operations were performed during the EPOXI spacecraft team "stand down" after Extrasolar Planet Observation and Characterization (EPOCH) operations and before the start of development for DIXI operations (i.e., during October and November 2008). DINET developments and operations were on a non-interference basis with EPOXI to the maximum extent possible. DINET was sponsored by NASA Office of Space Operations / Space Communications and Navigation (OSO/SCAN) via JPL DSN office Space Networking and Mission Automation. The total cost of DINET was $1.4M, which included support for the EPOXI spacecraft team and their contractor Ball Aerospace and Technology Corporation.

Table 1 Technology Readiness Levels and DINET Events

TRL level	Definition	DINET Event
4	Component and/or breadboard validation in laboratory environment	Achieved via a long period of testing on personal machines and then in the JPL Protocol Test Lab, through December of 2005.
5	Component and/or breadboard validation in relevant environment	Achieved during end-to-end ION traffic in the tracking, telemetry, command (TTC) test environment, using TTC V31.1. April 16.
6	System/subsystem model or prototype demonstration in a relevant environment (ground or space)	Achieved at the conclusion of DINET system test Sept. 26, 2008, the "relevant environment" in this case being the EPOXI test bed environment.
7	System prototype demonstration in a space environment	Achieved at the conclusion of DINET operations November 13, 2008.
8	Actual system completed and "flight qualified" through test and demonstration (ground or space)	Achieved at the conclusion of DINET operations November 13, 2008.
9	Actual system "flight proven" through successful mission operations	

3. Validation Objectives and Experiment Design

3.1. Validation Objectives

Flight validate the key features of the DTN protocols, by measuring DTN performance against stated metrics:

- Path Utilization Rate
- Delivery Acceleration Ratio
- ION Node Storage Utilization
- Multipath Advantage.

These metrics are described in detail in section 3.1.1.

Compare to performance predictions made prior to flight test. Measure and report on the following:

- Environment Envelope (Table 2)
- Resource Envelope (Table 2) of the end to end system
- Protocol Envelope (Table 3) used in the experiment, since adjusting these can affect the results.

Describe mission benefits of using DTN to the degree possible with this mission topology

Compare DTN performance to typical state-of-practice approaches

Table 2 DINET Environmental and Resource Envelopes

Parameter	Description	Value	Units	Extent	Source
Environment Envelope					
Propagation delay	One-way light time between EPOXI and Earth	79-49	seconds	per pass	EOC
Partition delay	Time between contacts.	300-13800	seconds	see description	EOC
File Size	All the file sizes of all the images sent during the experiment	see appendix	kBytes		EOC
Maximum size image	ION configuration setting for image size ceiling	64	kBytes		EOC
Data Rates (rate control)	Data rates (to and from S/C) set by ION as specified in global.ionrc file	to: 196 from: 17013, 90	Bytes/sec		EOC
Functional Data Rates	EPOXI sequence commanded CFDP rate (from S/C)	0.1-19.1	PDUs/sec		Rich's sequence
Physical Data Rates	Normally constant. Actual data rates (to and from S/C) experienced during operations. Actual radiation rate of S/C and ground.	to: 200K from: 867-165635	bits/sec	data rate vs time per pass period	EPOXI flight team
number of ground router nodes	ground nodes not including end nodes	6			
Number of end nodes	end node= data producer and/or consumer	3			DINET
Number of links per node	varies by node	1-3			DINET
Total number of links	Bidirectional and non-bidirectional	10 bi, 20 non-bi			DINET
Contact duration	Actual durations of contacts, with outages noted.	600-12600	seconds	per contact	EOC
Data Volume	Size of images sent and received per contact per node	see appendix	kBytes	per contact, per node	EOC
Data Completeness	Frames received on ground vs number of frames from spacecraft. Frames received on S/C vs number of frames sent from ground.	100% from spacecraft	percentage	per contact	EPOXI flight team
Data Quality	Synonymous with data completeness for this experiment, since completeness is the measure of quality.	100% from spacecraft			EPOXI flight team
Bit Error Rate	BER as function of time per pass	0		per pass	EPOXI flight team
Available buffer size	Volatile allocation, varied per node	10-24	Mbytes		EOC
Resource Envelope					
CPU utilization	Fraction of capacity of SCU-B used	unrecoverable	fraction vs time		EPOXI flight team/Ba
Storage	Image data (volatile) on EPOXI per contact.	0-3.8	MBytes		EOC for volatile
Footprint	Amount of storage on F drive of SCU-B that DINET software occupies.	~900	KBytes		EOC
Operator Time	Network operations time per pass	negligible			EOC

EOC = Experiment Operations Center, ION = Interplanetary Overlay Network, SCU-B = Spacecraft Control Unit B

Table 3 DTN Protocol Envelope

DTN Protocol Parameter	DINET Setting	Full Range
Convergence-layer protocols	the intra-region links between PTL nodes and DSOT nodes are BP over Bundle Relay Service (BRS, built on TCP/IP) and are continuously connected at 1 million bytes per second. The inter-region links between DSOT nodes and the Deep Impact spacecraft (node 7) are BP over LTP over TM/TC via the DSN.	Other possible convergence-layer protocols currently supported in ION are TCP, UDP, and DGR, but lots of others are contemplated in DTNRG: raw Ethernet, SMTP, flash drive, etc.
Bundle priority	15% of all images will be assigned Bundle Protocol priority 2 (expedited), 60% will be assigned BP priority 1 (standard), and the remaining 25% will be assigned BP priority 0 (bulk). All BP custody signals will be issued at priority 2 while all BP status report bundles will be issued at priority 1.	We are exercising all three of the defined BP priority levels.
Bundle Protocol Header Compresssion	All Bundle Protocol traffic will be CBHE-compressed.	non-compressed bundle headers
Custody of Application Bundles	All application bundles will be flagged as Source Custody Required	The custody alternatives are No Custody Transfer and Custody Requested but Source Custody Optional
Custody Status Reports	none	all combinations and permutations of reporting on bundle Reception, Forwarding, Custody Acceptance, Delivery, and Destruction.
Bundle time-to-live	7 days (604800 seconds), [so that bundles may remain queued for future transmission as needed]	any integral number of seconds greater than zero
Delivery	all bundles arriving at a closed endpoint will be queued for delivery when the endpoint is opened, rather than discarded.	discarding of currently undeliverable bundles is the alternative
Routing	All nodes in DSOT will use only static routes and default routes (groups). All other nodes will use only dynamic routing.	Static, default and dynamic There are capabilities in static routing that we are not using, though: we're not defining any special override rules that are bundle-source-specific or service-specific.
maximum clock error	1 sec [Presumed (for controlling contact start and stop timing)	any integral number of seconds greater than zero
LTP transmission acknowledgement	Red, yes acknowledged	alternative is for part of all of a given LTP block to be unacknowledged ("green")
Maximum LTP data segment size	750 bytes	any integral number of bytes greater than zero
LTP block size limit	will always be 1 second's worth of transmission at the maximum data rate at which the node can transmit at any time (on any link)	any integral number of bytes greater than zero
Maximum number of LTP sessions per node	will be the summation, over all links over which the node can transmit, of 20% more than the maximum round-trip time (in seconds) on the link at any time during the experiment.	any integral number of bytes greater than zero
LTP nominal block size	on each link over which the node can transmit will be 1 second's worth of transmission at the maximum data rate at which the node can transmit at any time on that link during the experiment.	any integral number of bytes greater than zero
Queuing time	will always be 1 second	any integral number of seconds greater than zero

CBHE = compressed bundle header encoding; PTL = (JPL) Protocol Technology Lab; TCP/IP = transmission control protocol/internet protocol;

3.1.1. Terms of Validation

Let *XYZ* denote the transmission opportunity from node *X* to node *Y* on DINET pass or configuration *Z*. The duration of *XYZ* in seconds, denoted by D_{XYZ}, is the end time of *XYZ* minus the start time of *XYZ*. The data rate of *XYZ* in bytes per second is denoted by C_{XYZ}. The raw *capacity* of *XYZ*, denoted by K_{XYZ}, is equal to $D_{XYZ} * C_{XYZ}$. (Note that this is ideal capacity; the actual capacity of the link will be the ideal capacity reduced by actual signal noise on *XYZ*. Moreover, transient outages in transmission—as were experienced during four of the eight DINET transmission opportunities—necessarily reduce the total capacity of an opportunity.)

DINET operated in two configurations, *a* and *b*; the former does not induce data loss while the latter induces loss by randomly discarding 1/32, or 3.125%, of received Licklider Transmission Protocol (LTP) segments. The total data return capacity S_{72a} from the EPOXI spacecraft (node 7) to the Earth subnet (node 2) while DINET is in configuration *a* is $\sum K_{72Z}$ for $Z = 1 \rightarrow 4$. The

total data return capacity S_{72b} from the EPOXI spacecraft (node 7) to the Earth subnet (node 2) while DINET is in configuration *b* is $\sum K_{72Z}$ for $Z = 5 \rightarrow 8$.

The total data return capacity S_{M7a} from the two Mars subnets (nodes 3 and 5) to the EPOXI spacecraft (node 7) while DINET is in configuration *a* is $\sum K_{M7Z}$ for $Z = 1 \rightarrow 4$. The total data return capacity S_{M7b} from the two Mars subnets (nodes 3 and 5) to the DI spacecraft (node 7) while DINET is in configuration *b* is $\sum K_{M7Z}$ for $Z = 5 \rightarrow 8$.

The EPOXI spacecraft is the bottleneck in the flow of data from the Mars subnets to the Earth subnet: the *total science data return capacity* of DINET in configuration *a*, S_{M2a}, is either the capacity of the transmission opportunities from the Mars subnets to EPOXI or the capacity of the transmission opportunities from EPOXI to the Earth subnet, whichever is less. That is, $S_{M2a} = S_{M7a} \perp S_{72a}$ and $S_{M2b} = S_{M7b} \perp S_{72b}$.

The volume of priority-0 science data that is received at the Earth subnet over the entire course of DINET while in configuration *a* is denoted by R_{0a}. Similarly, the volume of priority-1 and priority-2 science data received at the Earth subnet over the entire course of DINET while in configuration *a* is denoted by R_{1a} and R_{2a}. The *raw volume of science data received* at the Earth subnet over the entire course of DINET in configuration *a*, R_{Ta}, is the sum of these: $R_{Ta} = R_{0a} + R_{1a} + R_{2a}$. Similarly, $R_{Tb} = R_{0b} + R_{1b} + R_{2b}$.

The *urgency-weighted volume of science data received* at the Earth subnet over the entire course of DINET in configuration *a*, W_{Ta}, is the weighted sum: $W_{Ta} = R_{0a} + (2 * R_{1a}) + (4 * R_{2a})$. Similarly, $W_{Tb} = R_{0b} + (2 * R_{1b}) + (4 * R_{2b})$.

The *reference volume of priority-0 science data* received at the Earth subnet while DINET is in configuration *a*, denoted by Q_{0a}, is computed as R_{Ta} multiplied by the proportion of all image bundles that were published with priority 0 during this phase of the experiment. (This is the proportion of R_{Ta} that we would expect to be priority-0 data, that is, the expected value of R_{0a} if there were no reordering of data transmissions in the network due to priority.) Similarly, $Q_{1a} = .60 * R_{Ta}$ and $Q_{2a} = .25 * R_{Ta}$, and the same relationships can be expressed for the configuration-*b* phase of the experiment as well.

The *urgency-weighted reference volume of science data received* at the Earth subnet while DINET is in configuration *a*, V_{Ta}, is the weighted sum: $V_{Ta} = (.5 * Q_{0a}) + Q_{1a} + (2.0 * Q_{2a})$. Similarly, $V_{Tb} = (.5 * Q_{0b}) + Q_{1b} + (2.0 * Q_{2b})$.

The size of the Interplanetary Overlay Network (ION) data store at each node *X*, I_X, is a DINET configuration parameter. The *size of the traffic storage allocation* A_X at each node *X* is computed by $A_X = .6 * I_X$.

The *total unassigned space* N_{XZ} at each node *X* for pass *Z* was reported by each node at least once on each day during which there was a tracking pass.

The *net path capacity* P_{XYa} for any single path from node X to node Y while DINET is in configuration *a* is the smallest value of $\sum K_{ijZ}$ for $Z = 1 \rightarrow 4$ among all links (i, j) in that path; P_{XYb} is similarly defined for configuration *b*.

3.1.2. Metric 1 – Path Utilization Rate (U)

Path utilization rate for DINET in configuration *a* is given by $U_a = R_{Ta} / S_{M2a}$. It measures the effectiveness of automatic forwarding, custody transfer, and delay-tolerant retransmission.

Validation criteria:

- $\mathbf{U_a > 90\%}$. (DTN uses both high-rate and low-rate links efficiently.)
- $\mathbf{U_b > 90\%}$. (DTN remains efficient despite an increase in the rate of data loss.)

3.1.3. Metric 2 – Delivery Acceleration Ratio (G)

The delivery acceleration ratio for configuration *a* is given by $G_a = W_{Ta} / V_{Ta}$. It measures the effectiveness of the priority system.

Validation criteria:

- $\mathbf{G_a > 1.05}$ (Prioritization accelerates the delivery of urgent data.)
- $\mathbf{G_b > 1.1}$ (The advantage of prioritization increases with the rate of data loss.)

3.1.4. Metric 3 – ION Node Storage Utilization

Retention of a stable margin of unassigned space at each node measures the effectiveness of congestion control.

Validation criteria:

- The total number of bundles for which custody is refused anywhere in the network for the reason "depleted storage", throughout each configuration, is always zero. (We never run out of storage anywhere.)
- $\mathbf{N_{X7} = N_{X6}}$ for all values of X. (Storage utilization stabilizes over the course of network operations.)

3.1.5. Metric 4 – Multipath Advantage

The multipath advantage M_{XY} for traffic from *X* to *Y* during DINET operations is computed as $\sum P_{XY}$ for all paths from *X* to *Y*, divided by the largest single P_{XY} among all paths from *X* to *Y*, minus 1. Where there is only a single possible path between *X* and *Y*, multipath advantage is zero; where there are multiple possible paths between *X* and *Y* with net path capacity greater than zero, multipath advantage increases with the aggregate of those net path capacities. Multipath advantage therefore measures the effectiveness of dynamic routing.

Validation criteria:

> **The multipath advantage for traffic from node 20 to node 8 is greater than 20%**. (Dynamic routing among multiple possible paths increases the total network capacity from Phobos to Earth.)

3.2. Experiment Design

3.2.1. System Level Design

The basic topology of DINET is shown in Figure 1 (i.e., two surface assets, a relay orbiter, and Earth). The surface assets are designated Mars and Phobos, and the Deep Impact (DI) spacecraft fills the role of the relay orbiter.

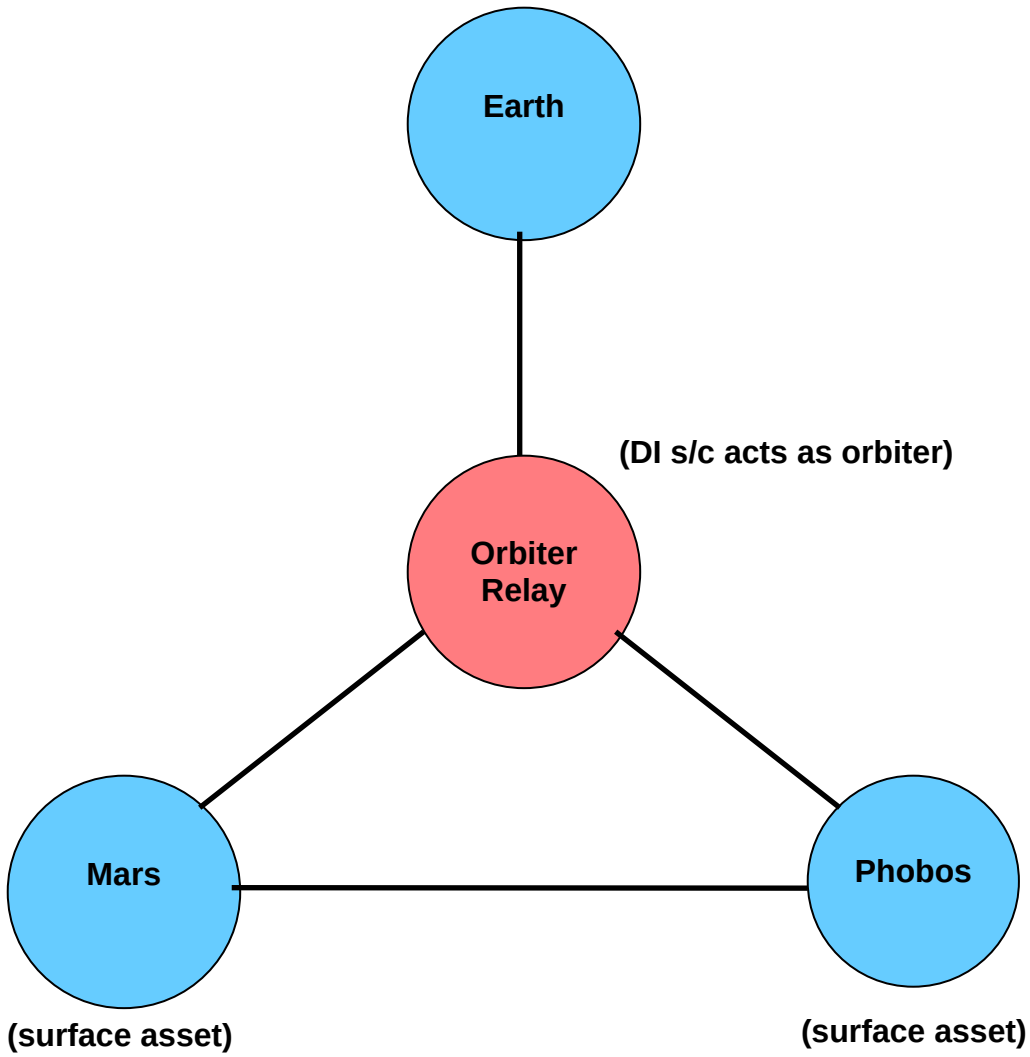


Figure 1 DINET Topology

Figure 2 shows how this topology was implemented during the experiment. The ION software with the DTN protocols was resident in each of the eleven circles, i.e. network nodes. All the nodes, except for the Deep Impact spacecraft [node 7], were physically located in the JPL Deep Space Operations Team (DSOT) area in building 264 or in the Protocol Test Laboratory (PTL) in room 238-401.

The 4-week period of DINET operations was divided into two configurations (*a* and *b*) of four tracking passes each. Configuration *a* had no injection of artificial data loss. During configuration *b*, 3.125% of all LTP segments were randomly discarded upon reception at the DI

spacecraft and at each of the three DSOT nodes. On the fourth tracking pass of each segment, the contact between Phobos and EPOXI was omitted. A brief "cross-link" contact between Phobos and Mars was scheduled for a time shortly before the 4th tracking pass of each experiment, providing an alternate path for data from Phobos. Four paths (topology experiments) were navigated using the setup shown in Figure 2. Table 4 summarizes the two configurations, the four topology experiments, and their relation to the eight DSN passes. Figures 3 through 6 and Table 5 present the paths through the network for the topology experiments.

Table 4 DINET Experiment Summary

Topology Experiments	DSN Pass	Experiment Segments		Validation Metrics							
		A	B	Path Utilization Rate		Delivery acceleration ratio		ION Node Storage Utlization		Multipath advantage	
1, 2	1	Active		Test		Test		Test			
1, 2	2	Active		Test		Test		Test			
1, 2	3	Active		Test		Test		Test			
1, 2, 3	4	Active		Test		Test		Test	Test	Test	Test
1, 2	5		Active		Test		Test	Test			
1, 2	6		Active		Test		Test	Test			
1, 2	7		Active		Test		Test	Test			
1, 2, 3	8		Active		Test		Test	Test	Test	Test	Test

Table 5 DINET Topology Experiments

Topology Experiment	Network Path
1	Send images from nodes 12 to node 8 via nodes 6, 3, 7 , 2, 4. Also send images from nodes 20 to node 8 via nodes 10, 5, 7, 2, 4.
2	Send Load/Go directive loads from node 16 to node 12 via nodes 4, 2, 7, 3, 6. Also from 16 to 20 via 4, 2, 7, 5, 10
3	Omit a contact between 7 and 5 and repeat experiment 1 & 2, forcing images from 20 to travel via 10, 6, 3, 7, 2, 4 and forcing directive loads to 20 to travel via 4, 2, 7, 3, 6, 10.

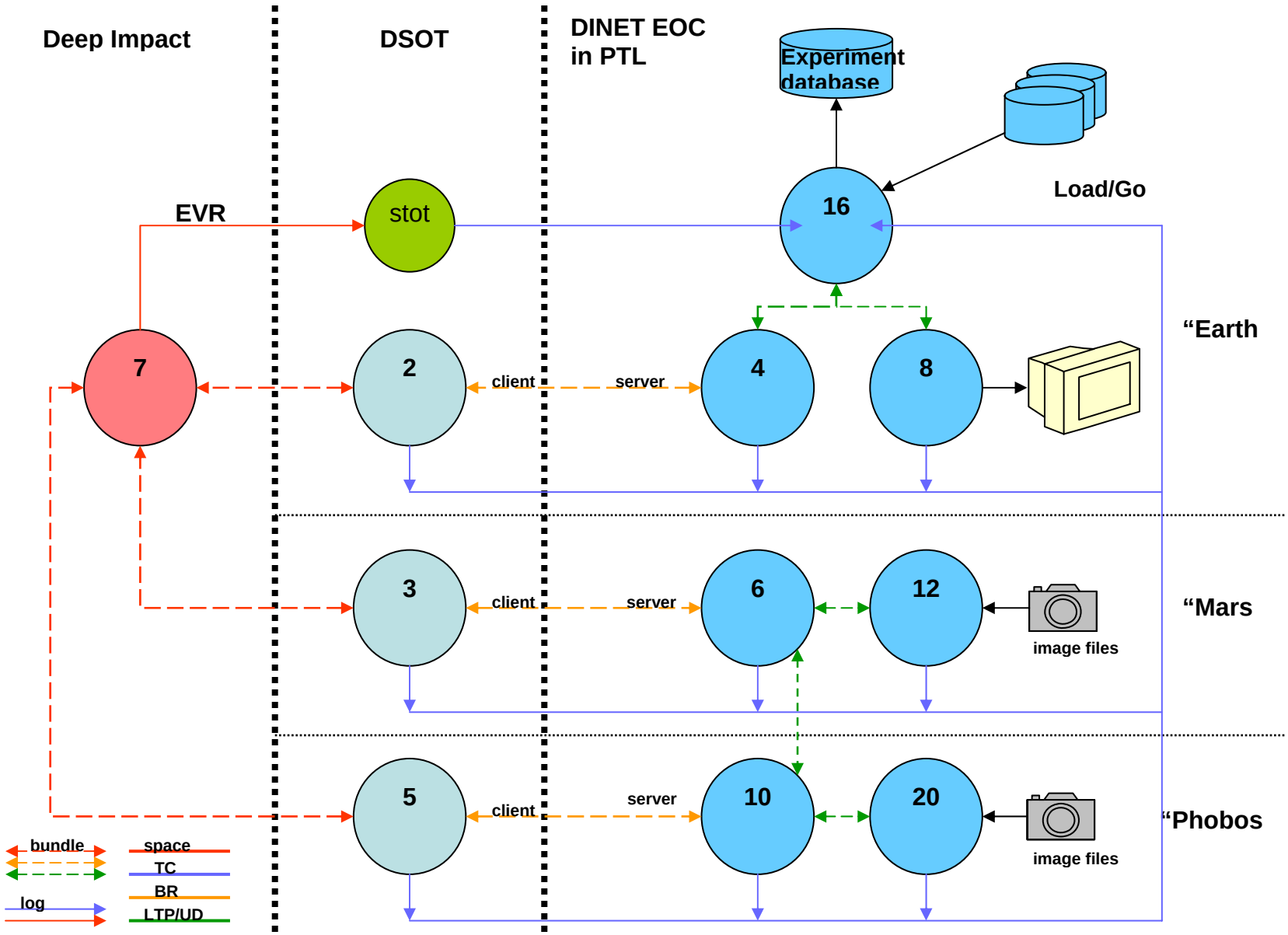


Figure 2 DINET Topology as Physically Implemented

Send images from nodes 12 to node 8 via nodes 6, 3, 7 (the Deep Impact spacecraft), 2, 4. Also send images from nodes 20 to node 8 via nodes 10, 5, 7 (the Deep Impact spacecraft), 2, 4. BRS = Bundle Relay Service; LTP = Licklider Transmission Protocol; UDP = user datagram protocol

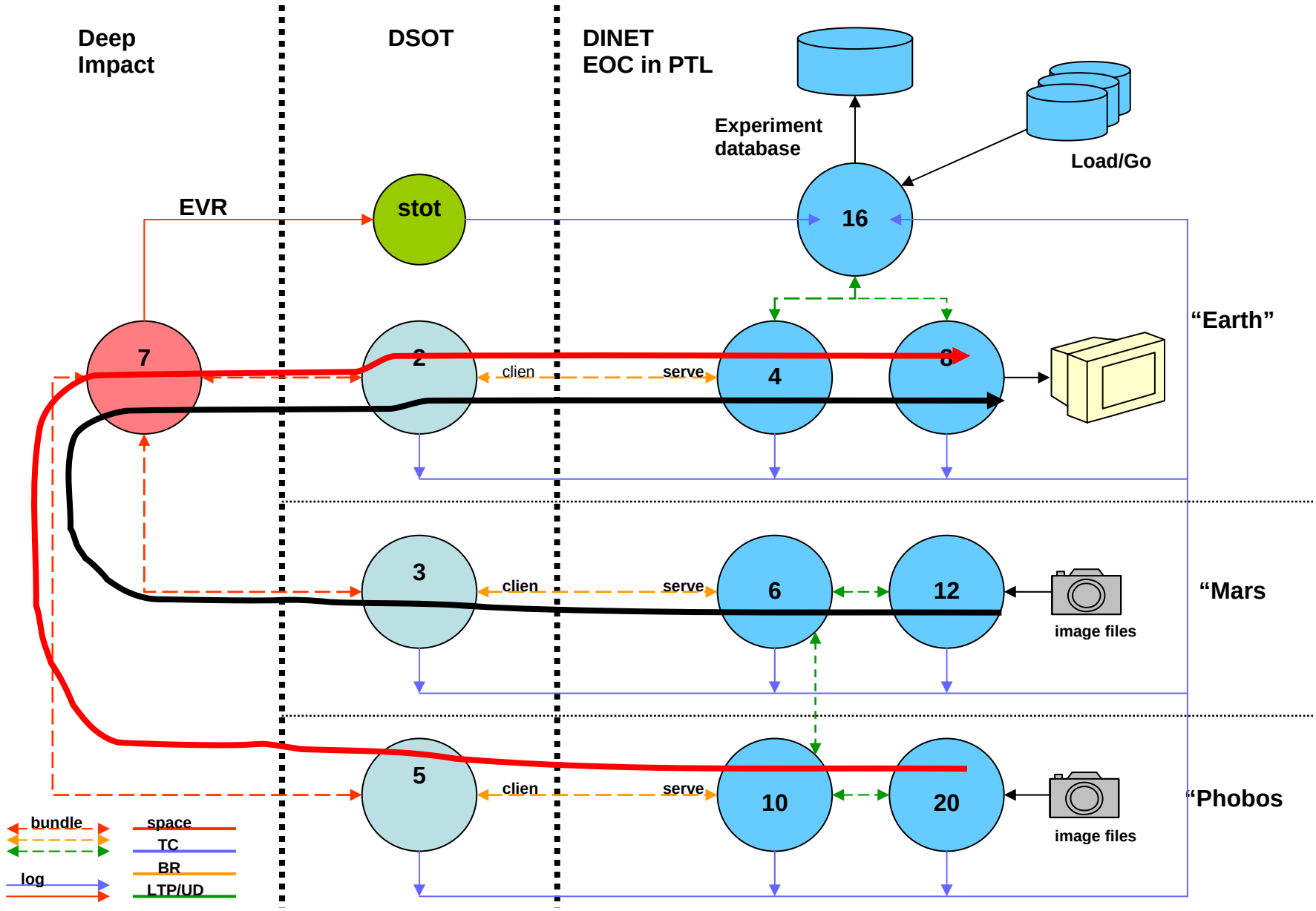


Figure 3 Topology Experiment 1

Send images from nodes 12 to node 8 via nodes 6, 3, 7 (the Deep Impact spacecraft), 2, 4. Also send images from nodes 20 to node 8 via nodes 10, 5, 7 (the Deep Impact spacecraft), 2, 4.

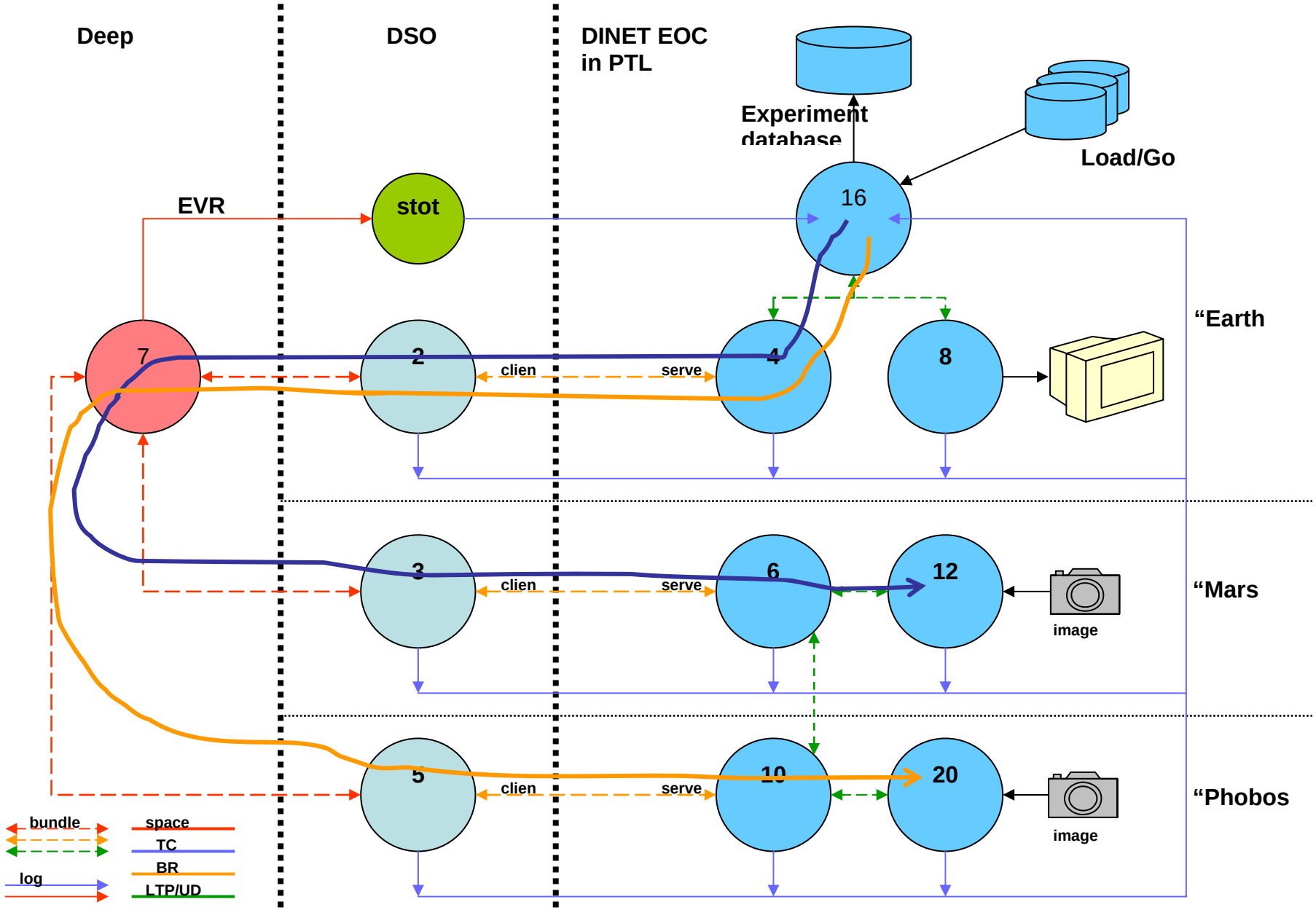


Figure 4Topology Experiment 2

Send Load/Go directive loads from node 16 to node 12 via nodes 4, 2, 7, 3, 6. Also from 16 to 20 via 4, 2, 7, 5, 10

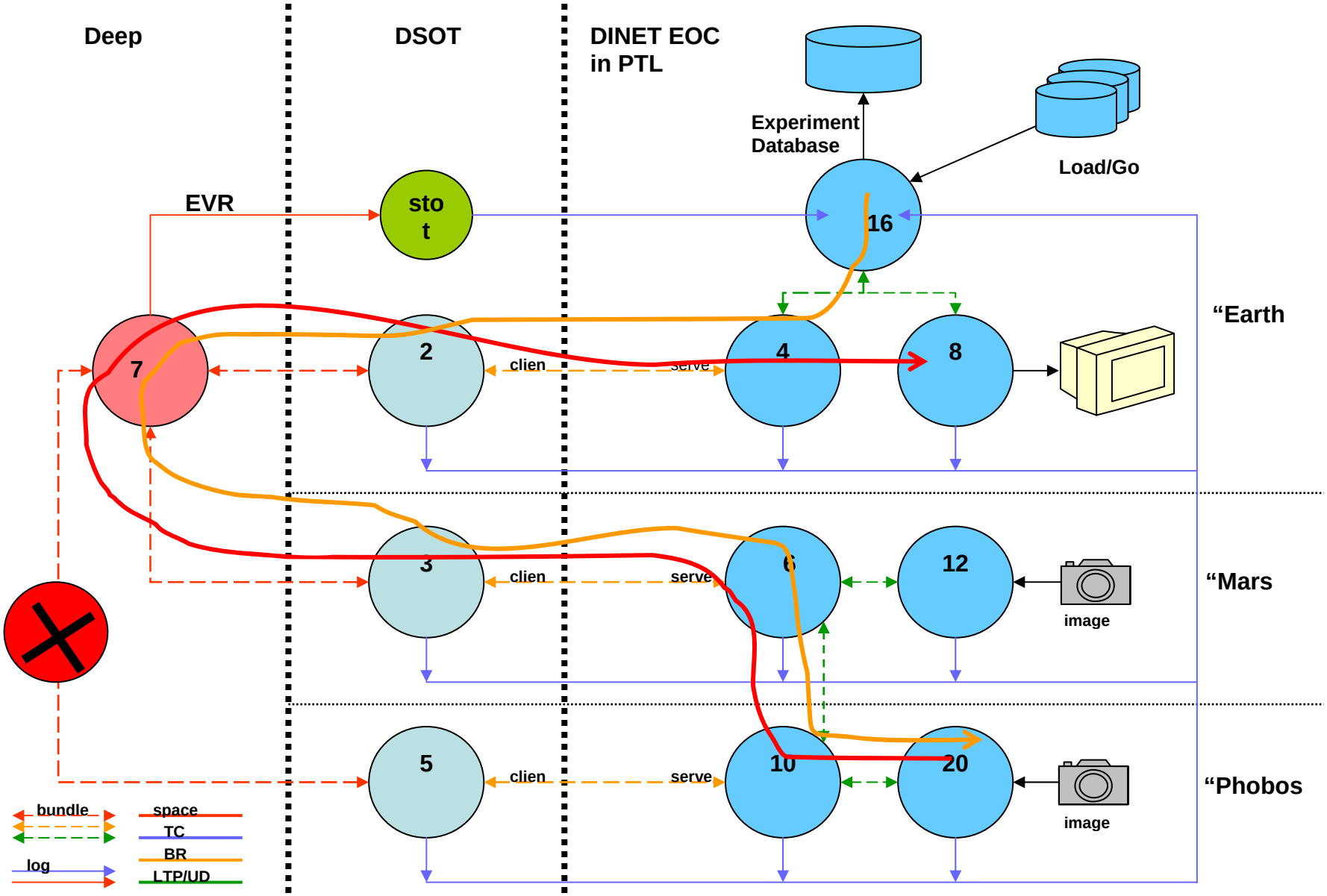


Figure 5 Topology Experiment 3

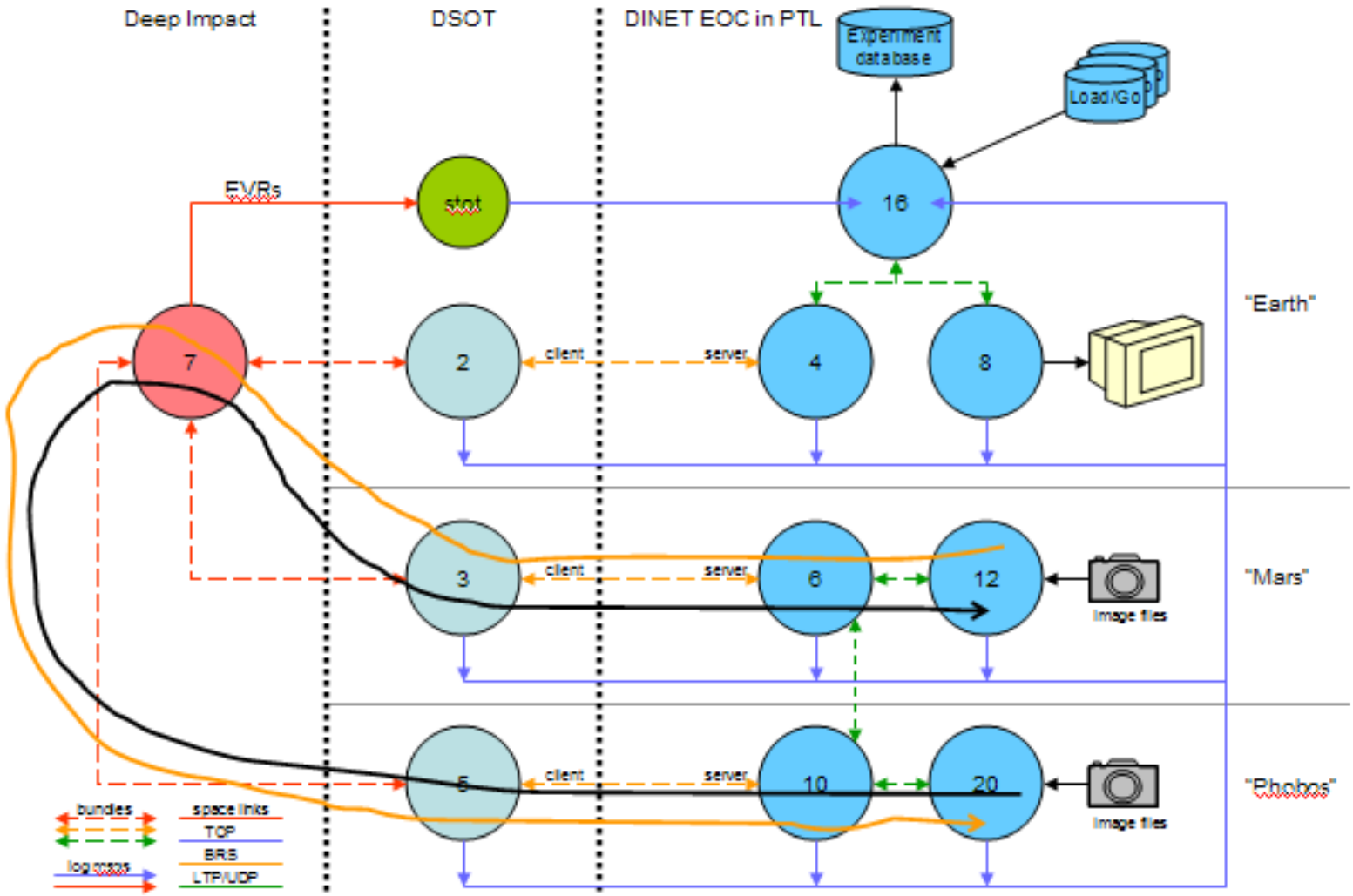


Figure 6 Topology Experiment 4

3.2.2. Flight Software

Most of the objectives of the DINET experiment are addressed by the execution of JPL's Interplanetary Overlay Network (ION) implementation of the DTN protocols.

The DTN architecture is much like the architecture of the Internet, except that it is one layer higher in the familiar International Organization for Standardization (ISO) protocol "stack". The DTN analog to the internet protocol (IP), called "bundle protocol" (BP), is designed to function as an "overlay" network protocol that interconnects "internets" – including both Internet-structured networks and also data paths that utilize only space communication links as defined by the Consultative Committee for Space Data Systems (CCSDS) – in much the same way that IP interconnects "subnets" such as those built on Ethernet, SONET, etc. The DTN analog to transmission control protocol (TCP) is the Licklider Transmission Protocol (LTP), an automatic system for the retransmission of BP data lost in transit. The ION implementation of BP/LTP is designed to work well within the constraints of the spacecraft flight software environment, emphasizing safety and efficiency. See Figure 7 for an overview of ION operations architecture.

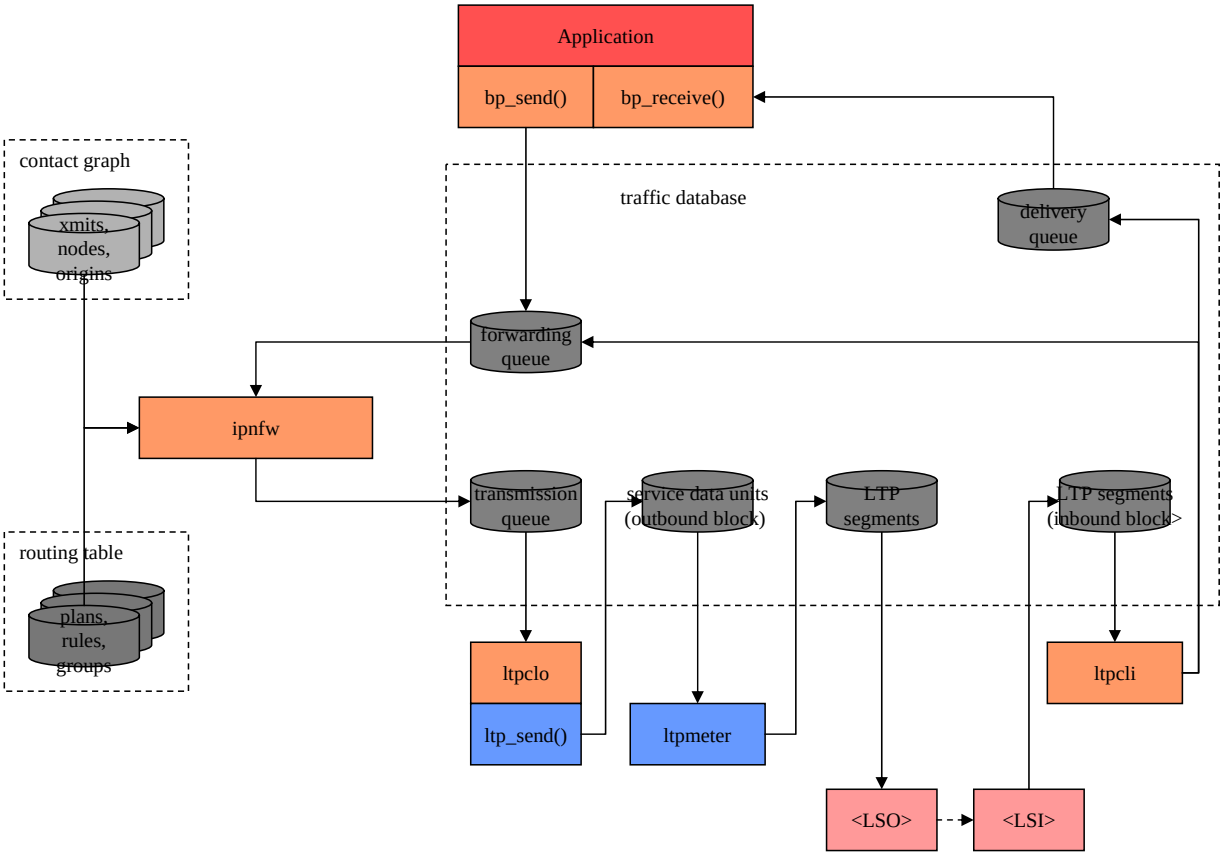


Figure 7 ION General Processing Flow

AMS = Asynchronous Messaging Service (a publish & subscribe protocol that sits on top of ION); LSI = link service input (tasks); LSO = link service output (tasks); AMS =

A few notes on this main line data flow:

- For simplicity, the data flow depicted here is a “loopback” flow in which a single BP “node” is shown sending data to itself (a useful configuration for test purposes). In order to depict typical operations over a network we would need two instances of this node diagram, such that the <LSO> task of one node is shown sending data to the <LSI> task of the other and vice versa.
- A BP application or application service (such as Remote AMS) that has access to the local BP node – for our purposes, the “sender” – invokes the `bp_send` function to send a unit of application data to a remote counterpart. The destination of the application data unit is expressed as a BP endpoint ID (EID). The application data unit is encapsulated in a bundle and is queued for forwarding.
- The forwarder task identified by the “scheme” portion of the bundle’s destination EID removes the bundle from the forwarding queue and computes a route to the destination EID. The first node on the route, to which the local node is able to transmit data directly via some underlying “convergence layer” (CL) protocol, is termed the “proximate node” for the computed route. The forwarder appends the bundle to one of the transmission queues for the CL-protocol-specific interface to the proximate node, termed an *outduct*. Each outduct is serviced by some CL-specific output task that communicates with the proximate node – in this case, the LTP output task **ltpclo**. (Other CL protocols supported by ION include TCP and user datagram protocol (UDP).)
- The output task for LTP transmission to the selected proximate node removes the bundle from the transmission queue and invokes the `ltp_send` function to append it to a *block* that is being assembled for transmission to the proximate node. (Because LTP acknowledgment traffic is issued on a per-block basis, we can limit the amount of acknowledgment traffic on the network by aggregating multiple bundles into a single block rather than transmitting each bundle in its own block.)
- The **ltpmeter** task for the selected proximate node divides the aggregated block into multiple segments and enqueues them for transmission by underlying link-layer transmission software, such as an implementation of the CCSDS Advanced Orbiting Systems (AOS) protocol.
- Underlying link-layer software at the sending node transmits the segments to its counterpart at the proximate node (the receiver), where they are used to reassemble the transmission block.
- The receiving node’s input task for LTP reception extracts the bundles from the reassembled block and dispatches them. Each bundle whose final destination is some other node is queued for forwarding, just like bundles created by local applications, while each bundle whose final destination is the local node is queued for delivery to whatever application “opens” the BP endpoint identified by the bundle’s final destination endpoint ID.
- The destination application or application service at the receiving node opens the appropriate BP endpoint and invokes the `bp_receive` function to remove the bundle from the associated delivery queue and extract the original application data unit, which it can then process.

However, the DTN protocols are at relatively high layers of the communication protocol "stack," and they rely on the support of communication software at lower layers to effect, for example, signal radiation and acquisition. Existing EPOXI operational software provides this support but is not designed to interact with the ION software, and vice versa.

An additional increment of DINET software, called Deep Impact Adaptation Software (DIAS), is therefore needed to act as an intermediary between ION and the operational software currently residing on the spacecraft and in the DI ground data system. The DIAS system enables the exchange of data between ION modules and DI operational software modules, thereby indirectly enabling the flow of DINET data, without requiring significant modification of DI flight or ground software.

The fundamental design decision underlying the DIAS design is simple. To minimize modification of DI operational software, we merely replace DI's implementation of the CCSDS File Delivery Protocol (CFDP) with a CFDP simulator, called "PX". DI operational software, both in flight and on the ground, continues to invoke the CFDP protocol data unit (PDU) transmission and reception functions exactly as it does now, but the PDUs that are transmitted and received are neither produced nor consumed by CFDP protocol engines. Instead those PDUs are artificially produced and consumed by the PX system, which simply encapsulates *segments* of DTN data in bogus CFDP file data protocol data units (FPDUs). In effect, we "tunnel" DTN traffic through underlying CFDP.

More specifically:

1. DINET test application data objects (e.g., images) are encapsulated in DTN *bundle* protocol (BP) data units for routing through the network. Outbound bundles are aggregated into *blocks* to minimize protocol overhead, and the Licklider Transmission Protocol (LTP) implementation then splits each block into segments for reliable transmission.
2. The PX system, acting as LTP's underlying "link service", encapsulates each LTP segment in a CFDP file data segment PDU. All such FPDUs have the same transaction ID, the same file data offset value (zero), and indeed the same values in all header fields except PDU length, which varies with the sizes of the encapsulated segments. Impersonating CFDP, the PX system passes these artificial FPDUs to DI operational software for transmission.
3. DI operational software accepts the PX-generated FPDUs and transmits them in the same manner that authentic FPDUs are transmitted. When the FPDUs are received by counterpart DI operational software, they are presented to a receiving PX state machine in exactly the same way that authentic received FPDUs are presented to the CFDP entity.
4. The receiving PX system extracts the encapsulated LTP segments from the FPDUs and passes them to LTP.
5. The LTP implementation reconstitutes blocks from the segments and extracts bundles from the reconstructed blocks. The BP implementation then forwards or delivers the received bundles.

This procedure is symmetrical: bundles sent both to and from the spacecraft are processed in the same way. In particular, note that the BP and LTP implementations used on-board the spacecraft

and in all ground computers (both DSOT and PTL) are the exact same code. All instances of these protocols are identical and interchangeable, except that they must be compiled differently for the various platforms on which they are executed: VxWorks on the flight computer; Solaris on the DSOT machines; and Linux on the PTL machines.

Figure 8 depicts the current operational Deep Impact CFDP architecture .

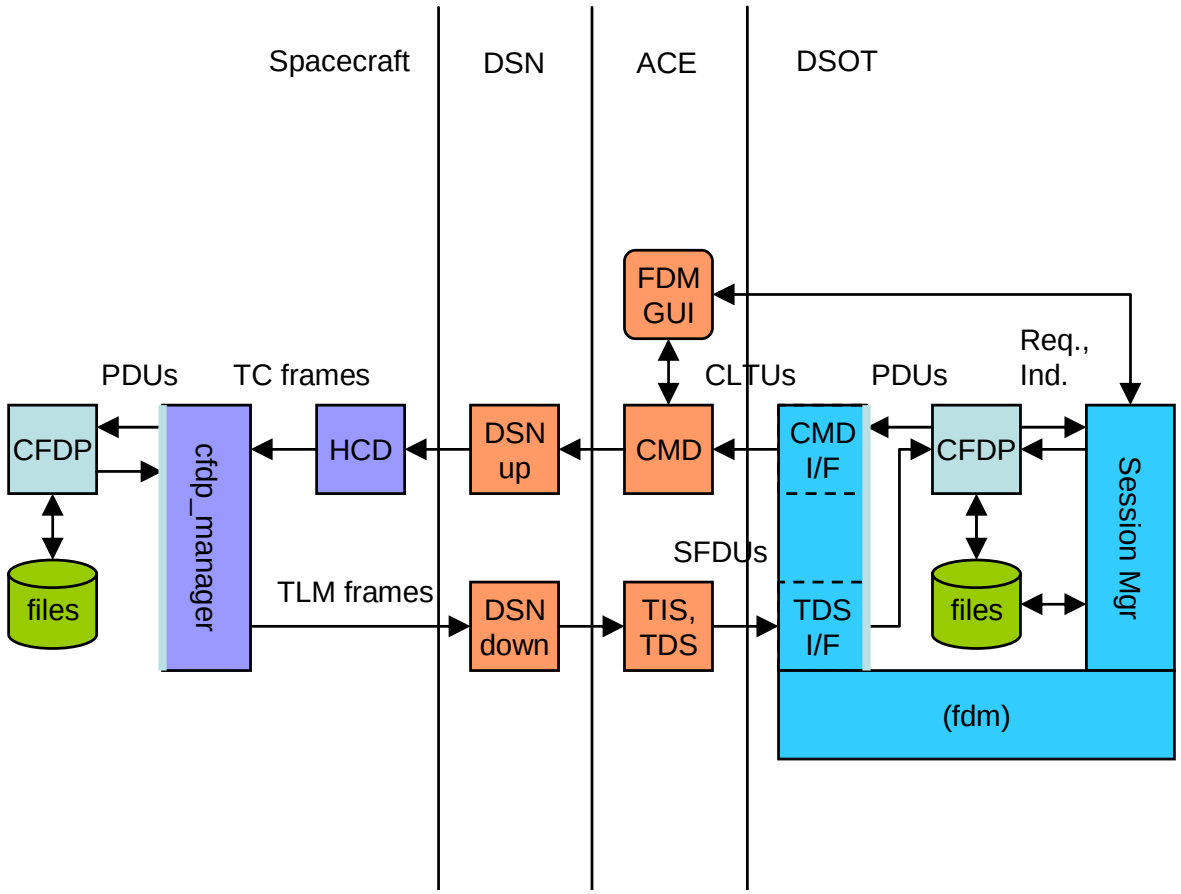


Figure 8 Current CFDP Architecture

HCD = hardware command decoder; pxisi = CCSDS File Delivery Protocol simulator input; pxiso = (output of same); TDS = Telemetry Delivery System; TIS = Telemetry Delivery System; TLM = telemetry

Figure 9 depicts the DINET software architecture including both PX and ION.

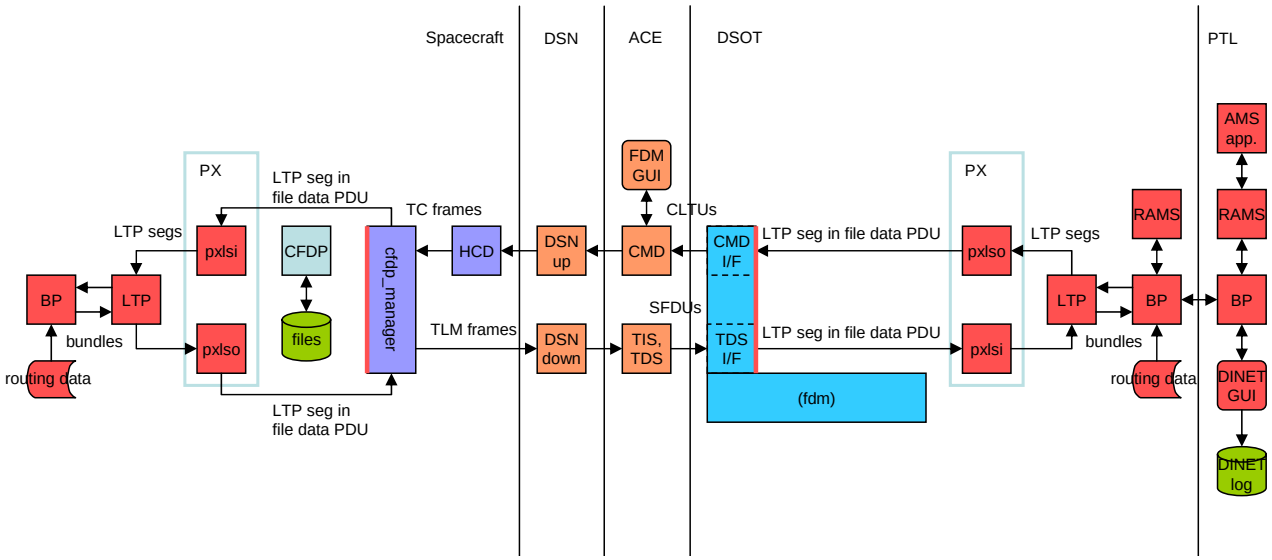


Figure 9 CFDP with Integrated PX and ION

3.2.3. Ground Software

The goal of the DINET ground software design is reducing cost by minimizing change. The DSN ground software component remains the same with exception of:

1. Create three new instances of File Delivery Manager (FDM) server that interface with ION. This interface is identical to the existing CCSDS File Delivery Protocol (CFDP) interface. These new FDM servers simply forward PDUs between ION layer and the telemetry/command subsystems by converting from telemetry packets to ION PDUs and from ION PDUs to command link transmission units (CLTUs). No CFDP uplink or downlink is performed by these FDM servers.
2. ION software used by the Ground system is identical to the software running on the EPOXI spacecraft with the exception of some differences in implementation of the DIAS layer.
3. The command Subsystem communicates with the three new File Delivery Manager (FDM) servers, one at a time, in addition to the existing FDM/CFDP server. The ACE (the person who sends commands to the spacecraft) switches the connection between the Command subsystem and the appropriate FDM server based on a fixed schedule to allow uplink dataflow.
4. In test environment, similar update is made to forward data between Flight software and three instances of FDM/ION server. There is no manual switching between uplink proxy and the three FDM servers as in real operation.
5. A new event processor task is created to forward event verification record (EVR) telemetry packets from the spacecraft to the Experiment Operation Center (EOC). These EVRs are used for tracking ION software status aboard EPOXI.

6. ION debug information from the Data Operation Center is captured and periodically forwarded to EOC through email.

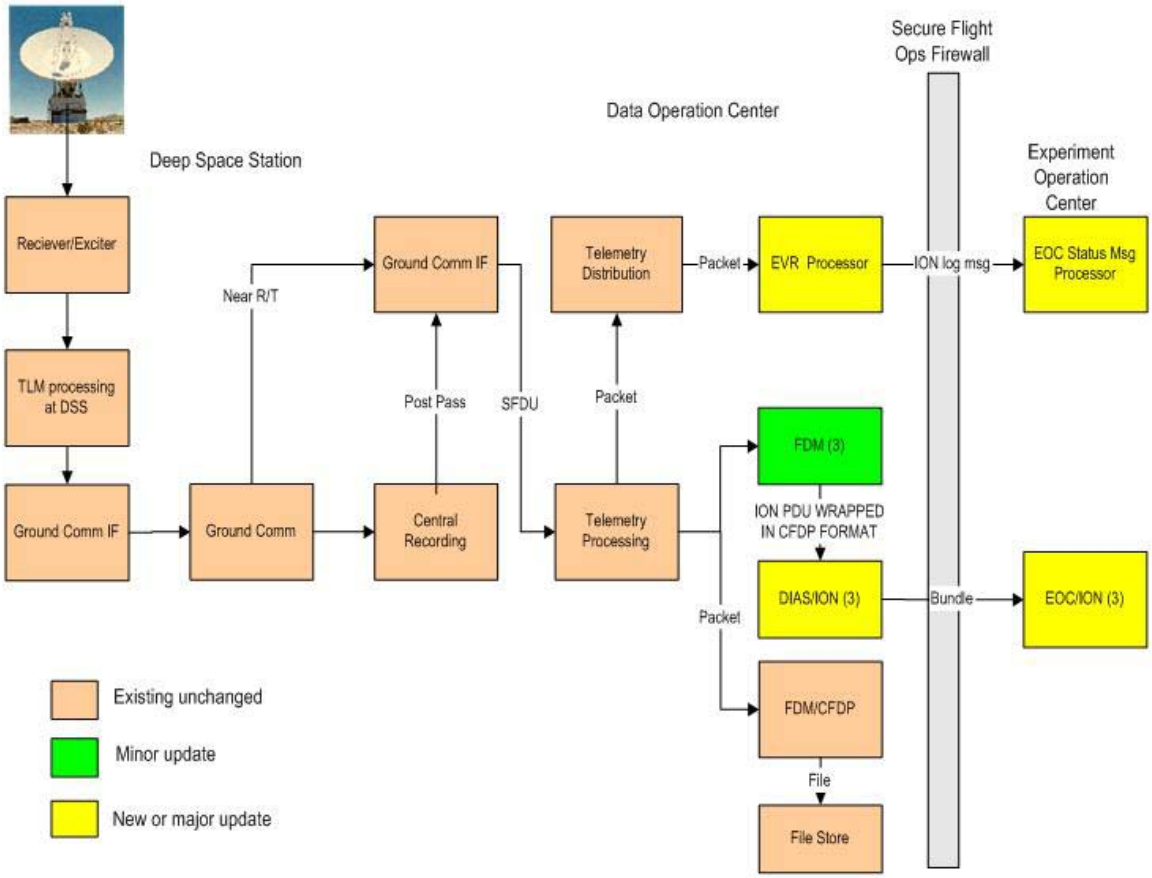


Figure 10 CFDP Operational Downlink Dataflow

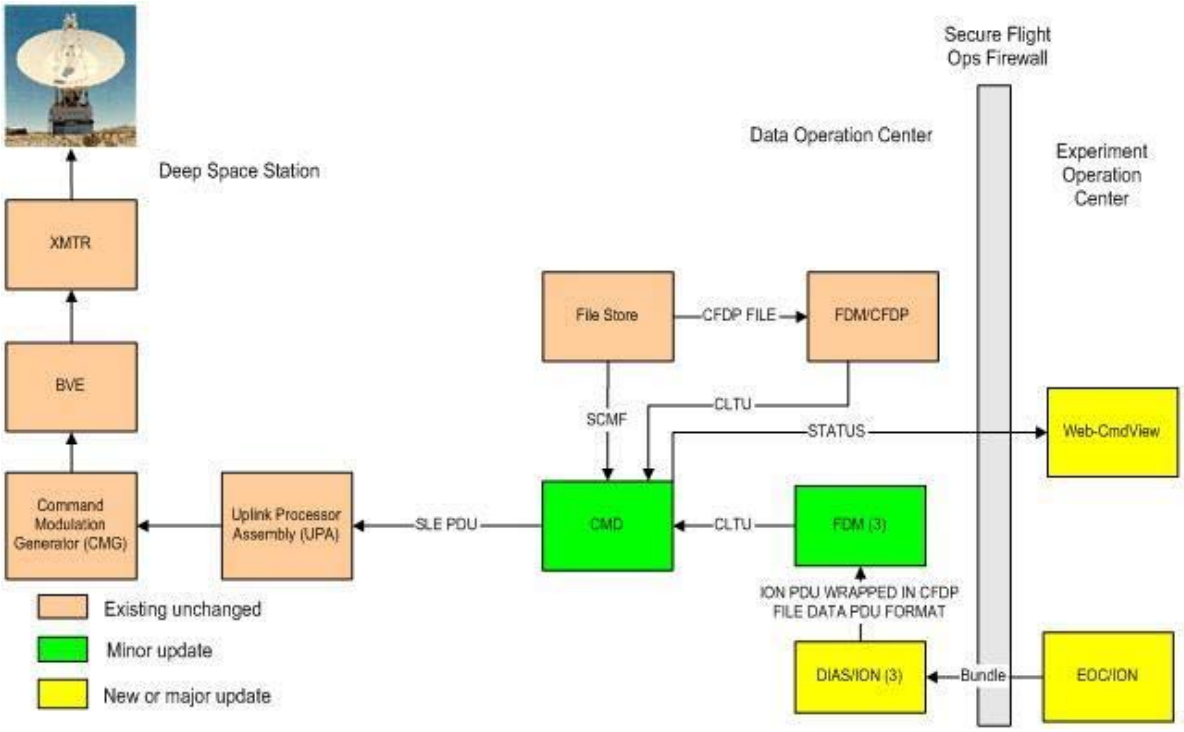


Figure 11 CFDP Operational Uplink Dataflow

BVE = Block V Exciter

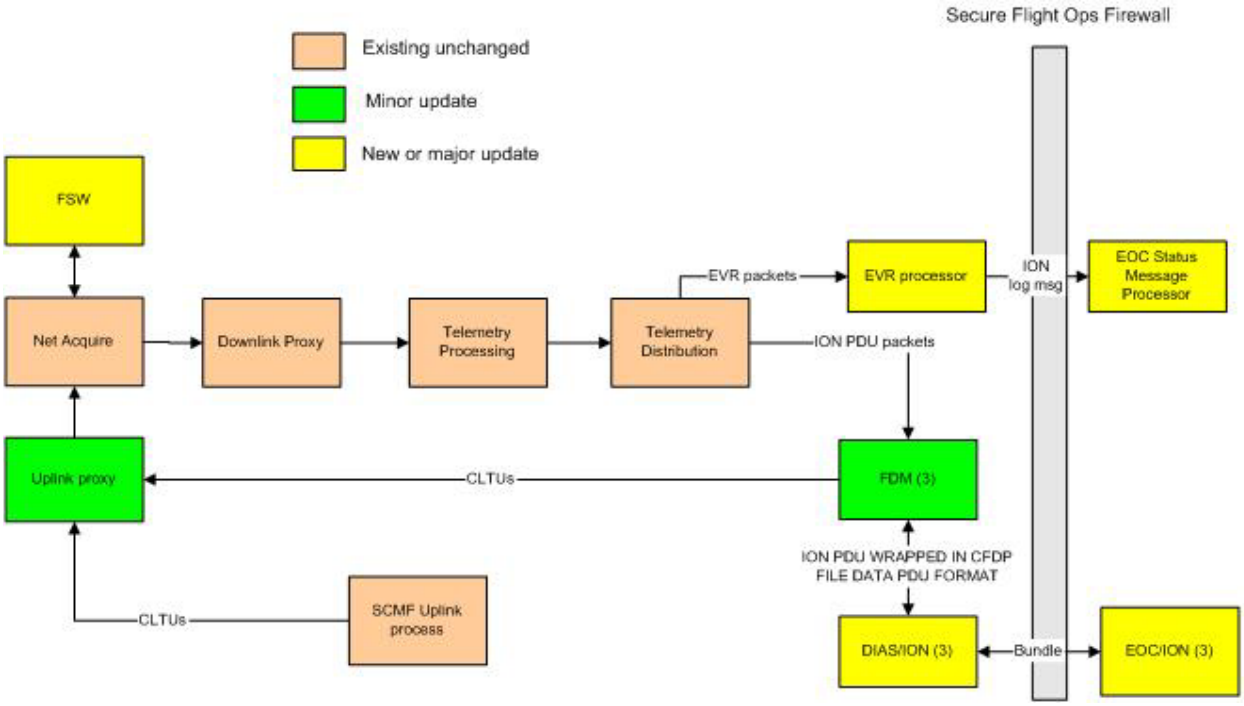


Figure 12 CFDP Testbed Dataflow

FSW = flight software

3.2.4. Experiment Operations Center

The Experimental Operations Center (EOC) is the critical point where all experiment monitoring and controlling occurred. It resided in the Protocol Technology Lab (PTL) in JPL's telecommunications laboratory facility. Since the PTL was conceived as a testbed area for space networking, the functional design revolves about projects such as DINET. The lab area includes moveable racks containing about 50 computers, three large liquid crystal display (LCD) wall-mounted displays, ample network access, and mission-control voice loop connectivity as illustrated in Figure 13.

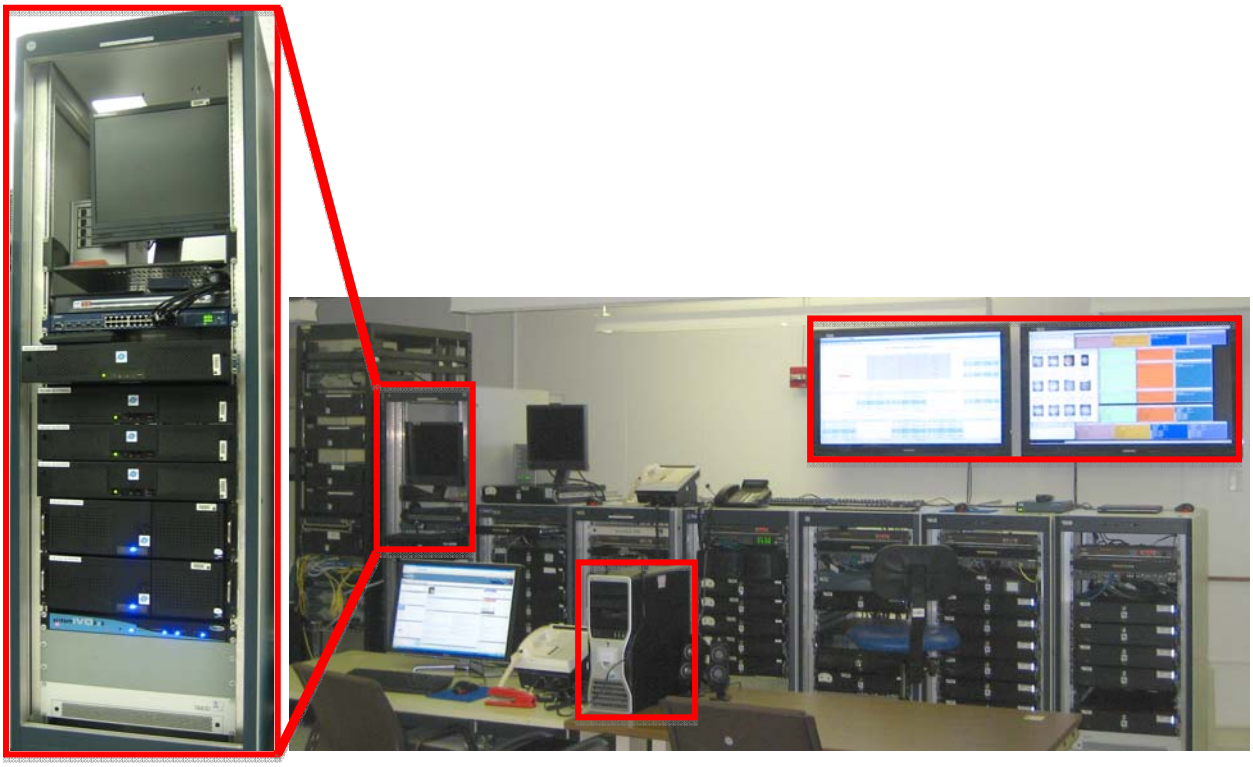


Figure 13 EOC Hardware (red boxes) in the PTL Work Area

Standard PTL hardware consists of Intel-based PCs running Fedora Linux, including DINET. All DINET computers ran 64-bit Linux except the administrative node, which ran 32-bit Linux (for plug-in compatibility).

The EOC was constructed around a set of requirements for DINET functionality. These requirements included:

Functions:

- Host three ION relay nodes and three endpoint nodes and requisite software.
- Provide Experiment Monitoring & Control (EM&C) without interfering with the operation of ION relay or endpoint nodes.

- Host a network time protocol (NTP) server to synchronize time across all EOC machines to within 1 second of Greenwich mean time (GMT) time.

Software Interfaces:

- Establish and maintain connectivity with DSOT from RELAY nodes for ION bundle transactions
- Establish and maintain connectivity with DSOT for receiving ION FSW log message EVR data from Deep Impact telemetry and other DSOT TCP/IP-relayed status information.
- Establish and maintain connectivity with DSOT for radiation status and voice coordination with DI operations using the Voice Operational Communications Assembly (VOCA)

Data Management:

- Provide data archiving of all experiment data
- Provide access to experiment operations records for on-line non-real-time query and retrieval
- Provide redundant experiment data archival capabilities to avoid loss of experiment data in the event of the failure of the primary data archive node.

User Interfaces:

- Host experiment operations display in real time for monitoring and controlling the experiment (see Figure 14).
- Provide receipt of and display of DSOT data that indicate the link status of the link to DI and indications when a bundle is radiated.

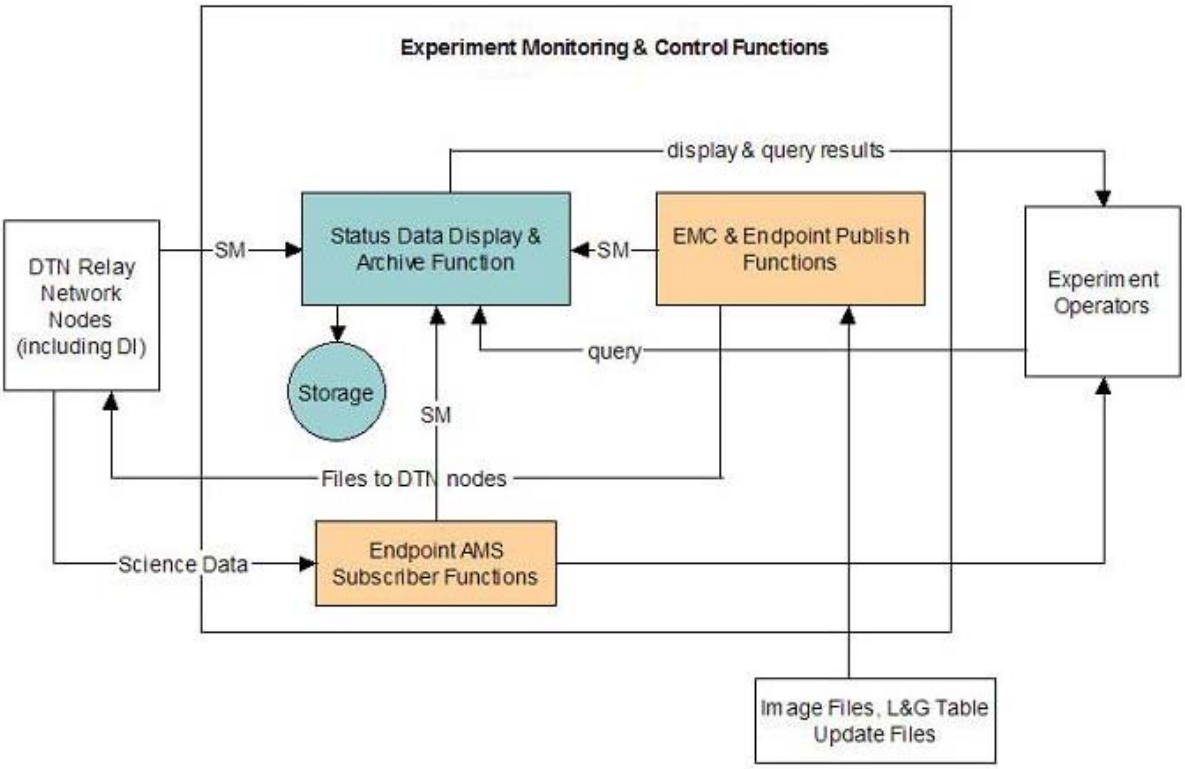


Figure 14 EOC Functional Architecture

EMC or EM&C = Experiment Monitoring & Control ; L&G = load and go (table); SM = status message

Each EOC machine in the PTL was connected to two EOC local area networks (LANs) – one private LAN for node management, network storage, and terminal traffic; and another routable LAN for experiment data and out-of-band experiment diagnostic information. The routable LAN was also accessible by DSOT machines making outbound client connections to server relay nodes in the EOC. This was accomplished using a secure authentication-based convergence layer protocol adapter of ION called Bundle Relay Service (BRS)—a TCP-enabled system allowing bundle-layer traversal through stateful firewalls such as the one protecting the DSOT nodes. One additional connection was established outside the bundle layer with the administrative node by a process running on DSOT extracting DINET EVRs and sending them to the administrative node via TCP. A Global Positioning System (GPS)-based stratum-1 NTP server on the experiment LAN kept all nodes time-synched.

The operating systems used were the 64- and 32-bit version of Fedora 9, with no special modifications beyond DINET software (ION, GUI, publish/subscribe applications). The DINET applications were developed to meet the project's top-level requirements, as previously stated. Their software location is shown in Figure 15, and functional flow is shown in Figures 16 and 17.

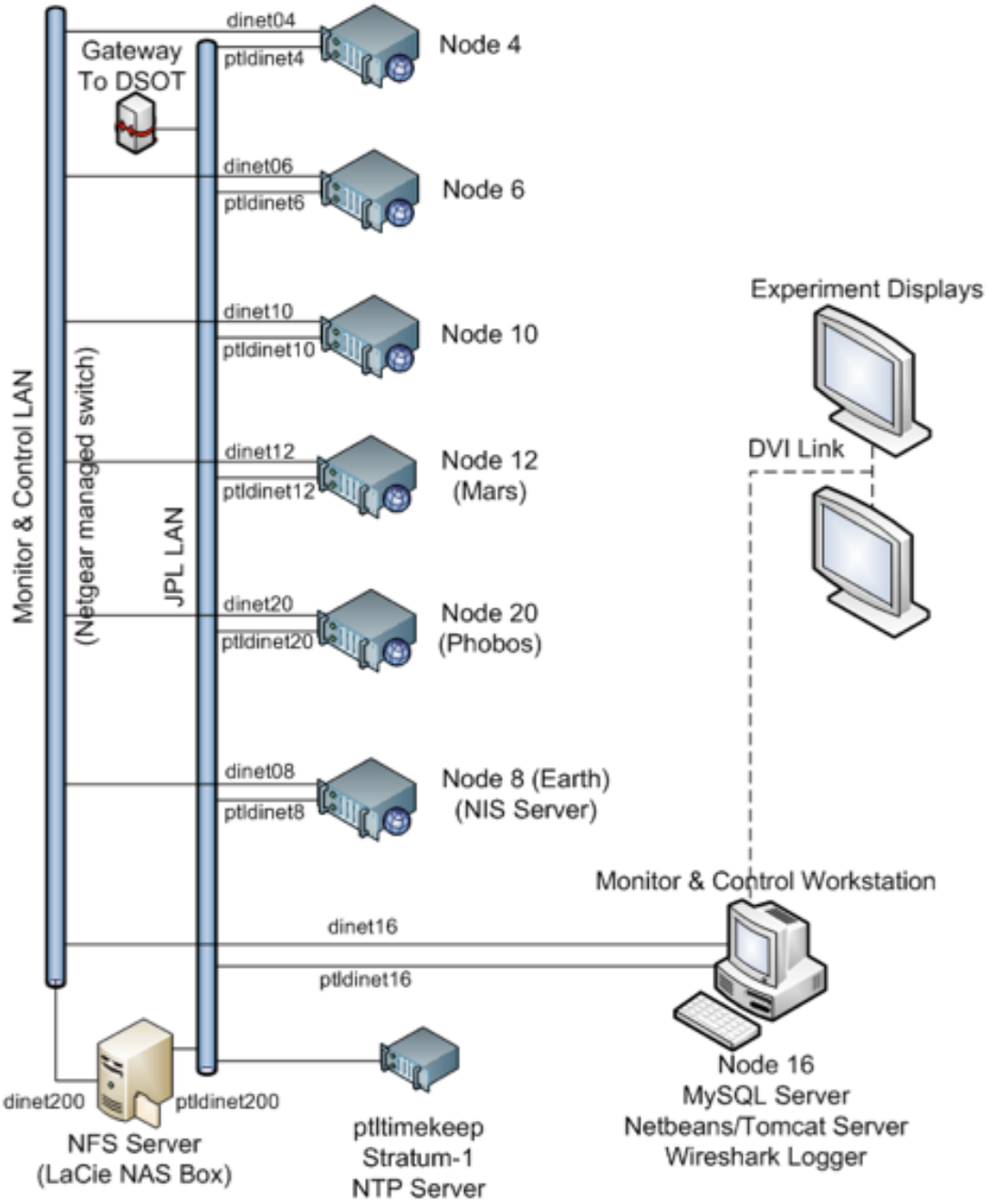


Figure 15 EOC Network Diagram

The EOC component of the PTL consisted of three ION-enabled relay nodes (4, 6, 10), three endpoint nodes (8, 12, 20), and the requisite software. In addition, the EOC contained a separate Administrative Node (16) running a GUI interface and console windows, central to the display of sending, receiving, and monitoring realtime data. Data archiving, as well as secondary redundant data archival of all experiment data and access to the experiment operations records for on-line non-realtime (NRT) query and retrieval, was also conducted through the Administrative node.

The lab provided for connectivity with DSOT from relay nodes for ION bundle transactions and for receiving ION FSW log message EVR data from Deep Impact telemetry and other DSOT TCP/IP-relayed status information. Outbound flight LAN firewall exceptions on specific ports were requested and put into place well before operations to allow outbound TCP/IP connections to EOC nodes from DSOT.

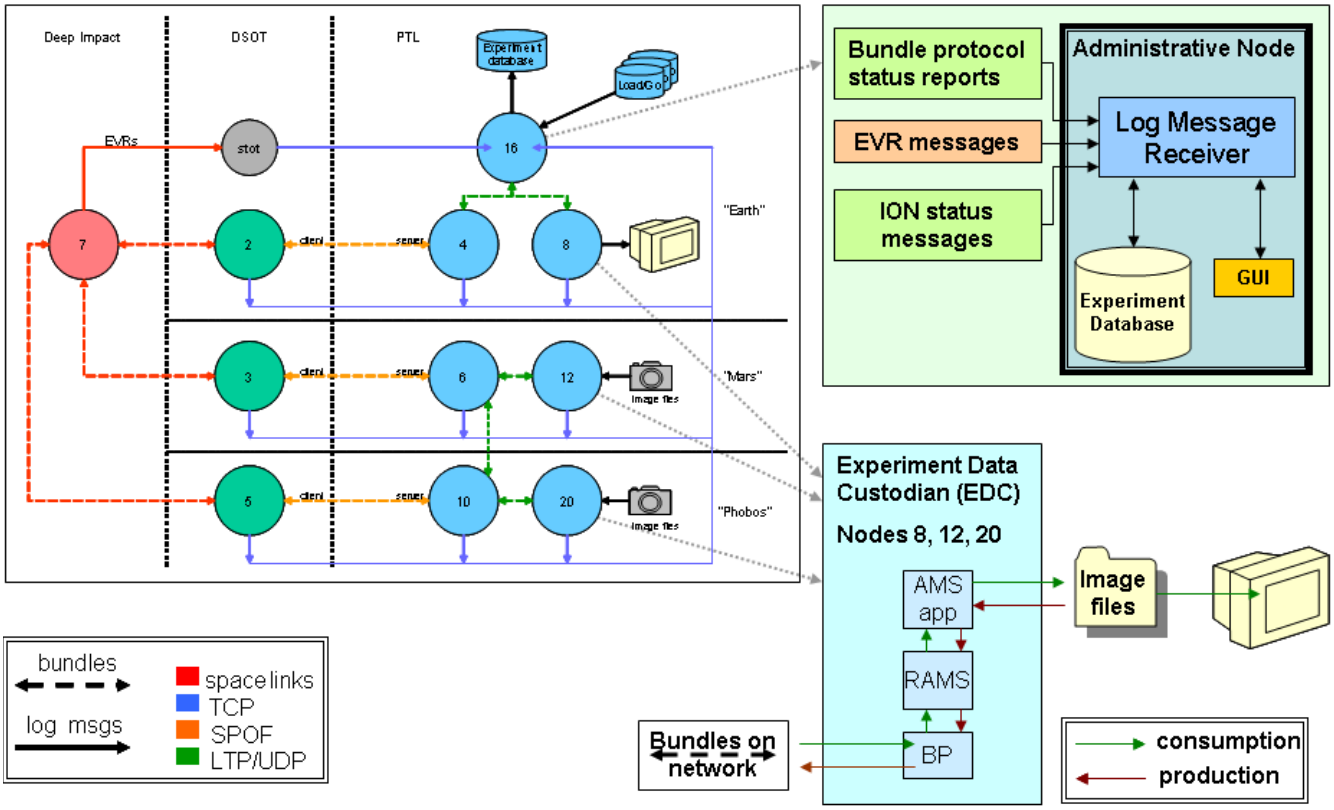


Figure 16 EOC Software Architecture - Software Overview

RAMS = Remote Asynchronous Message Service; SPOF = (has been changed to Bundle Release Service (BRS); UDP =; user datagram protocol

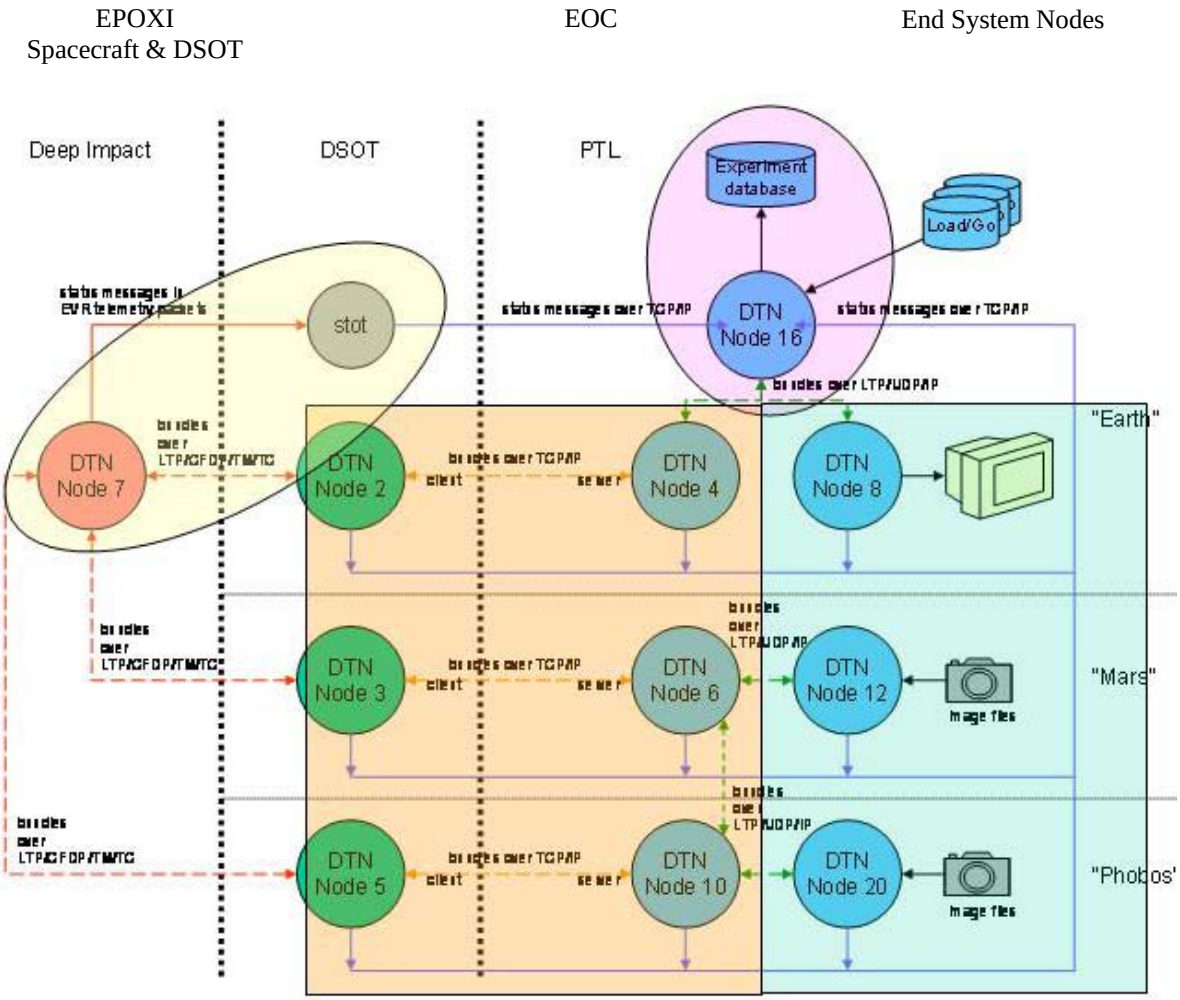


Figure 17 DINET Physical Network

stot = simple TDS output tool

EOC Publish & Subscribe and AMS

EOC Publish and EOC Subscribe tasks communicate through the Asynchronous Messaging Service (AMS). AMS is a publish & subscribe protocol that sits on top of ION and is part of the ION package (see Figures 18–20). AMS is a data system communications architecture under which the modules of mission systems may be designed as if they were to operate in isolation, each one producing and consuming mission information without explicit awareness of which other modules are currently operating.

An AMS node does not need to wait for the arrival of any message (such as a reply to the message it sent) before continuing performance of its functions. AMS might best be characterized as a messaging “middleware” protocol. As such, it relies on the capabilities of underlying Transport-layer protocols to accomplish the actual copying of a message from the memory of the sending node to the memory of the receiving node.

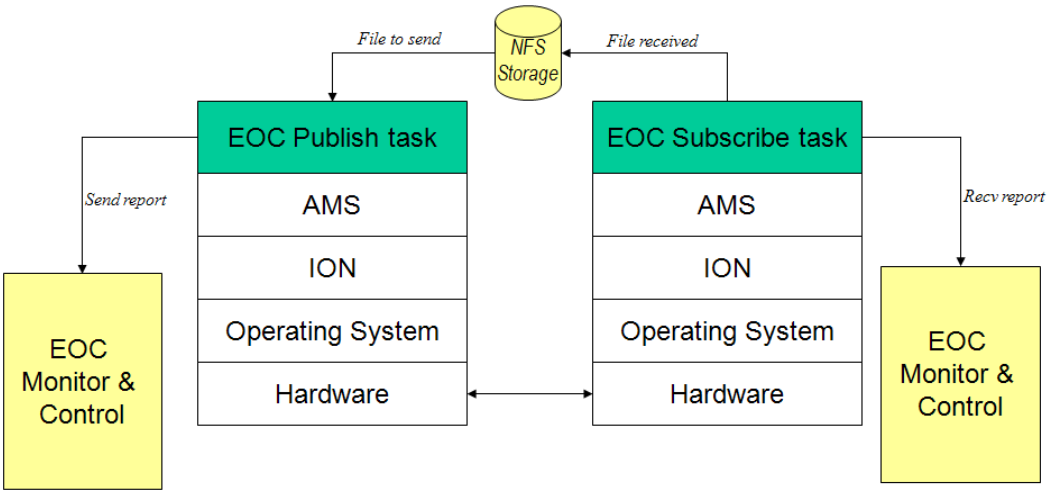


Figure 18 Publisher / Subscriber Software System

NFS = Network File System

EOC Publish and Subscribe Model:

- AMS uses a topic-based Publish & Subscribe model
  - Messages are published to "topics" or named logical channels; or in AMS's case "subject numbers"
  - Subscribers in a topic-based system will receive all messages published to the topics (subject number) to which they subscribe
- *EOC Publish* task on the *Mars* node will publish files in messages to AMS subject number 15
- *EOC Publish* task on the *Phobos* node will publish files in messages to AMS subject number 15
- *EOS Subscribe* on the *Earth* node will subscribe for the messages with AMS subject number 15
- AMS will deliver subject number 15 messages from *Mars* and *Phobos* nodes *EOC Publish* tasks to the *Earth* node *EOC Subscribe* task without any further EOC intervention

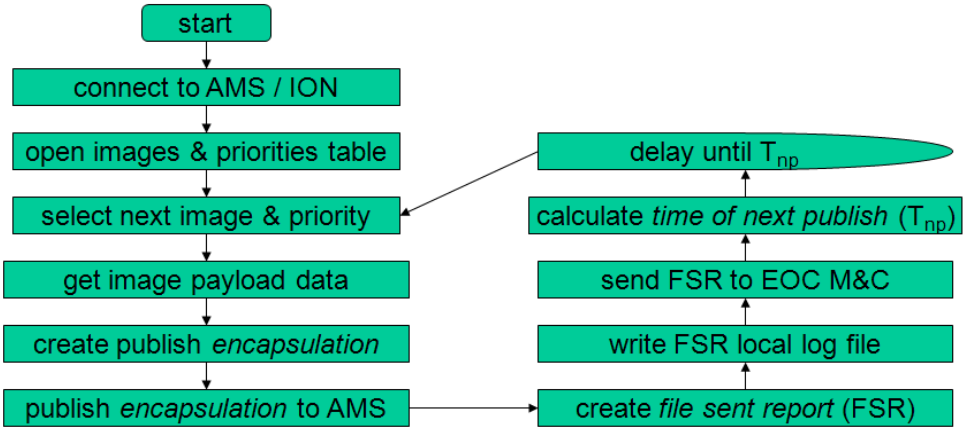


Figure 19 Publisher Control Flow

M& C = monitor and control

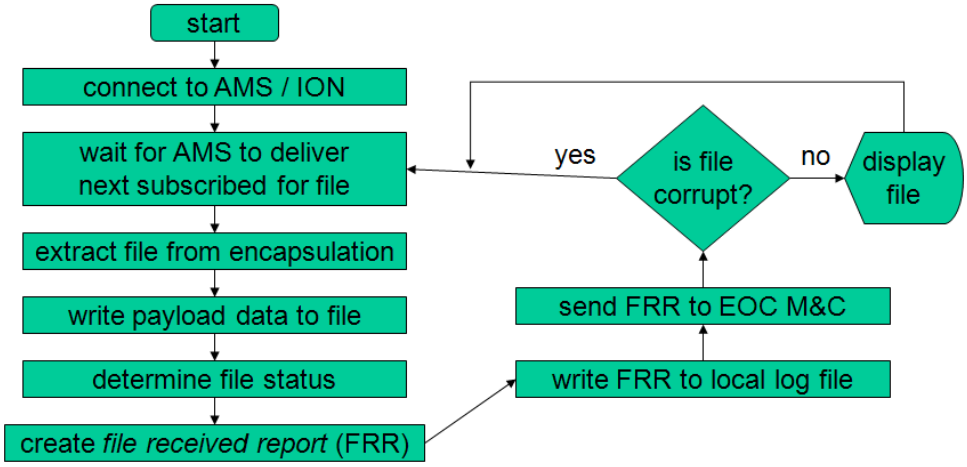


Figure 20 Subscriber Control Flow

EOC Administrative Node

The Administrative Node is a designated separate computer from the three relay computers and three endpoint ION node computers in the EOC. It was used to process and display received Protocol Diagnostic Messages (including messages announcing the arrival of Bundle Status Reports) as well as data archival and retrieval functions for normal operations. The Administrative Node provided for EOC displays and user interfaces to operate the DINET experiment in addition to running a GUI application that enabled the DINET operator to monitor and control the experiment. It also displayed a dynamic topology chart of the DTN in real time.

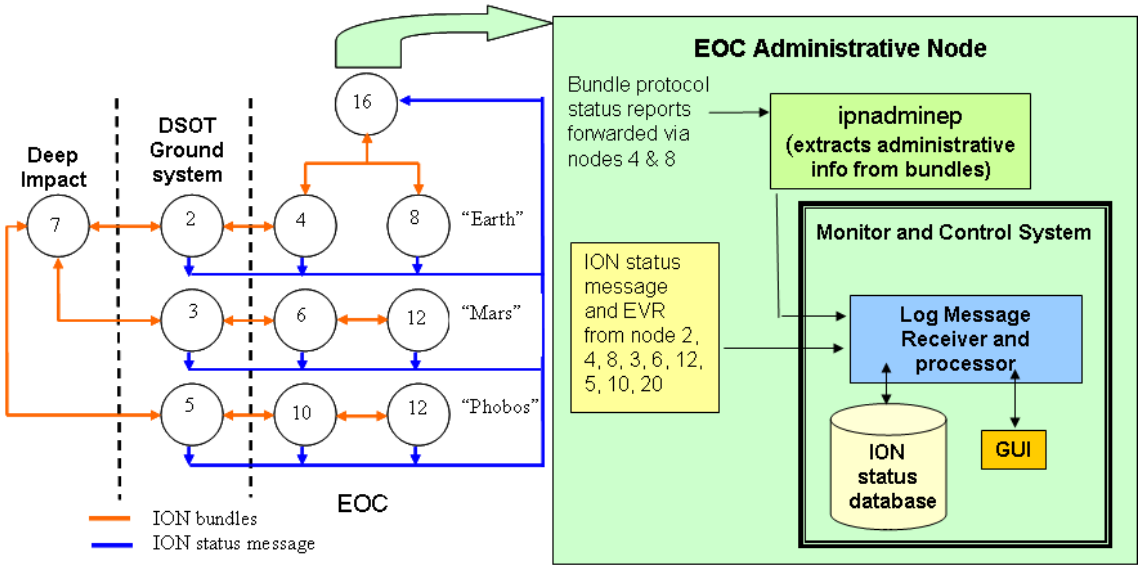


Figure 21 EOC to Administrative Node

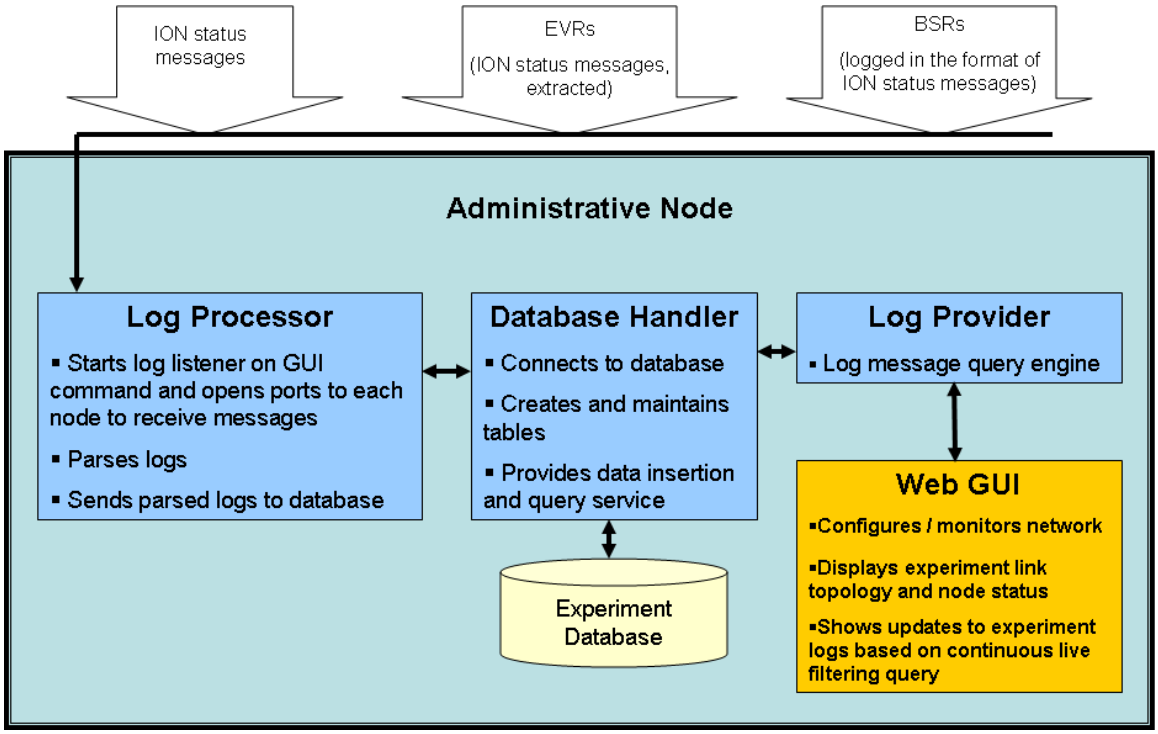


Figure 22 Administrative Node Functionality

GUI Application and Interface

The EOC bundle network is configured and monitored with the EOC GUI application (Figure 23). The application uses a combination of AJAX, html, Java, and the Netbeans 5.5 IDE to create a user interface (Figure 24) for visibility and interaction with the DTN. It primarily shows a realtime view of the topology of all nodes running, and all messages being received at each

node as they are stored in the database. A non-realtime message query GUI (Figure 25) augments this interface to enable the user to query the database for any log messages received at any time throughout the current (or any previous) experiment based on any database field.

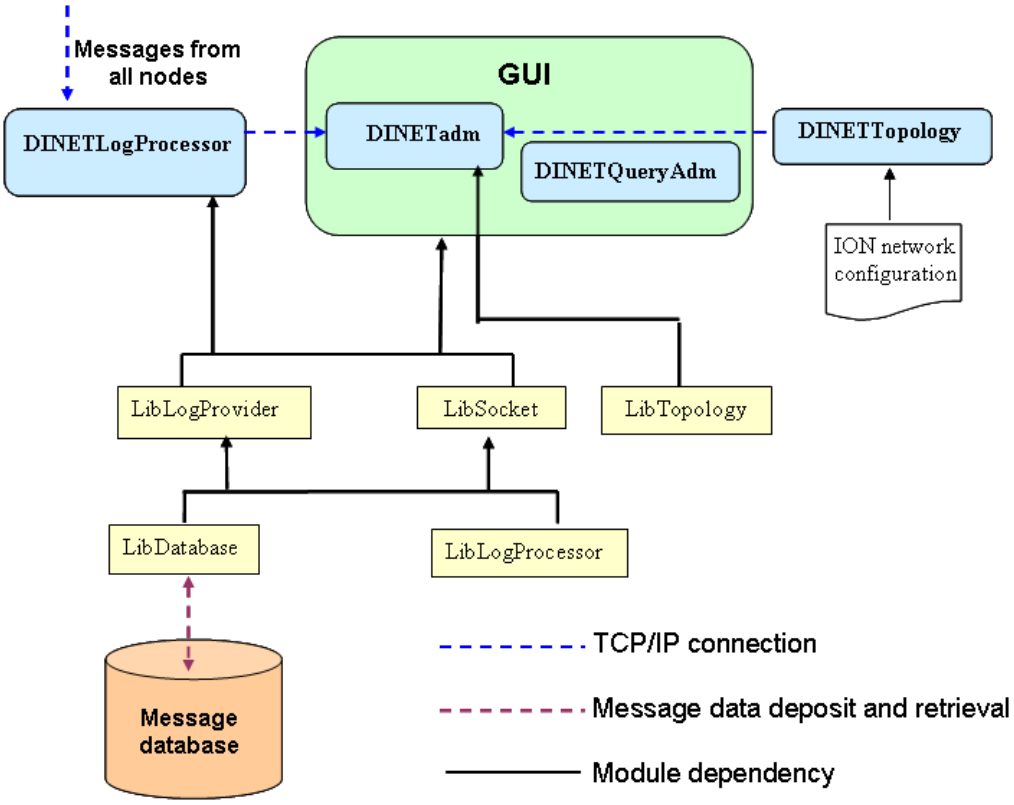


Figure 23 GUI Functionality

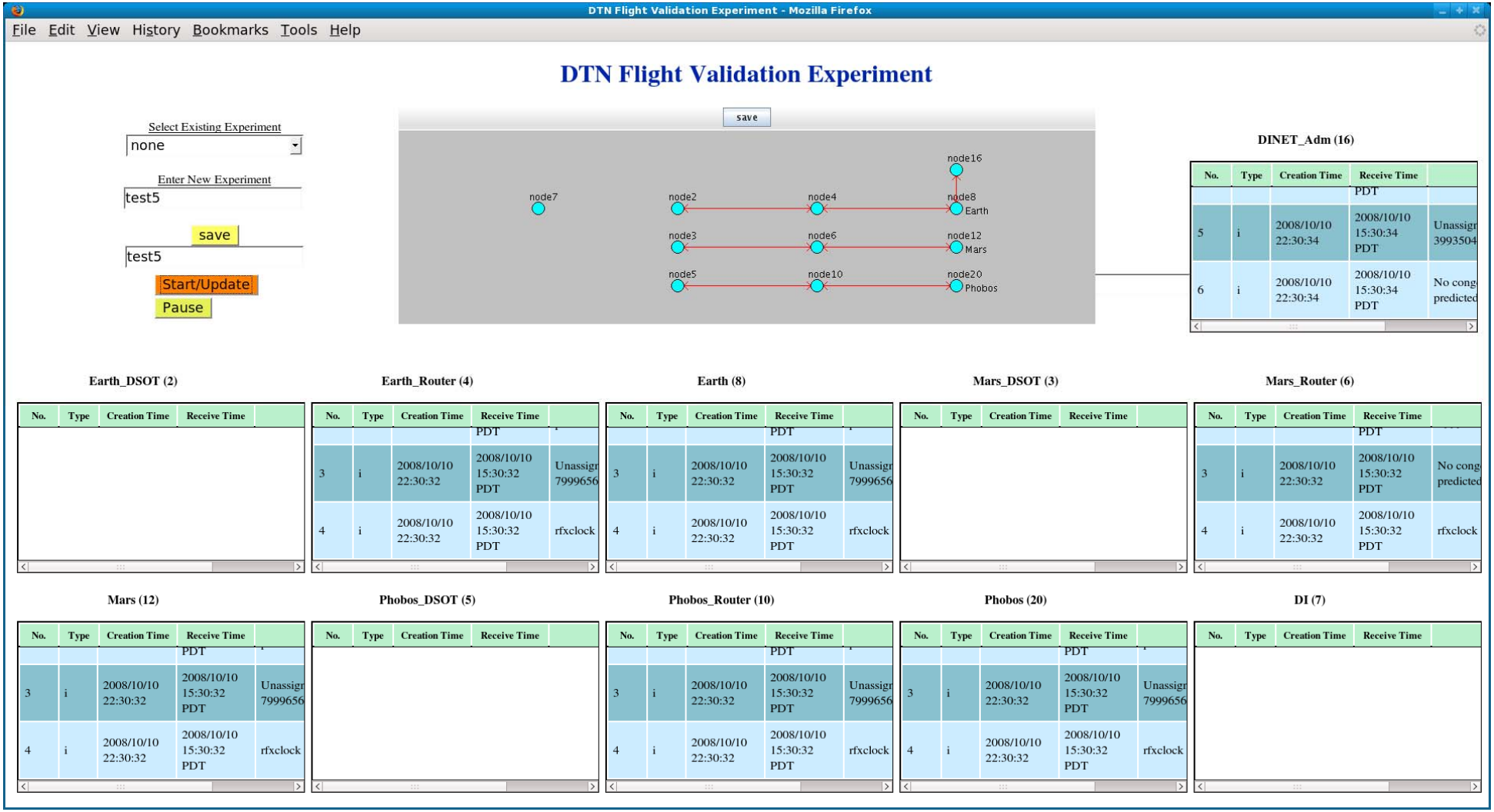


Figure 24 Monitor and Control GUI

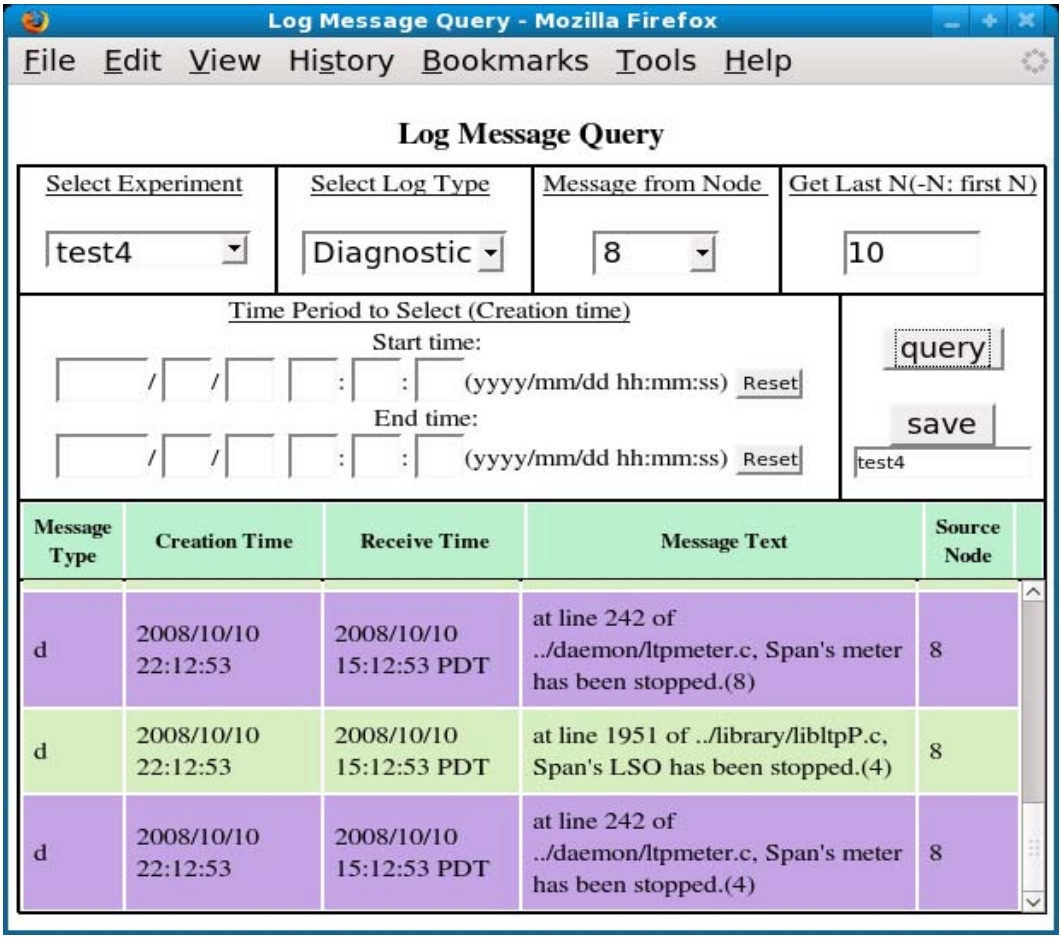


Figure 25 Non-Realtime Query GUI

SQL Database

The primary source of data archival used for the experiment was a MySQL database, which was hosted on the Administrative node. Any bundle status reports or log messages were received by the Administrative node via TCP/IP, parsed, and entered in the database according to the database record creation concept shown in Figure 26.

The backup data archival and redundant storage system used Wireshark to monitor and store data arriving at the Administrative Node, and any Wireshark logs were stored on a separate computer to guard against catastrophic failure of the Administrative Node or its SQL database.

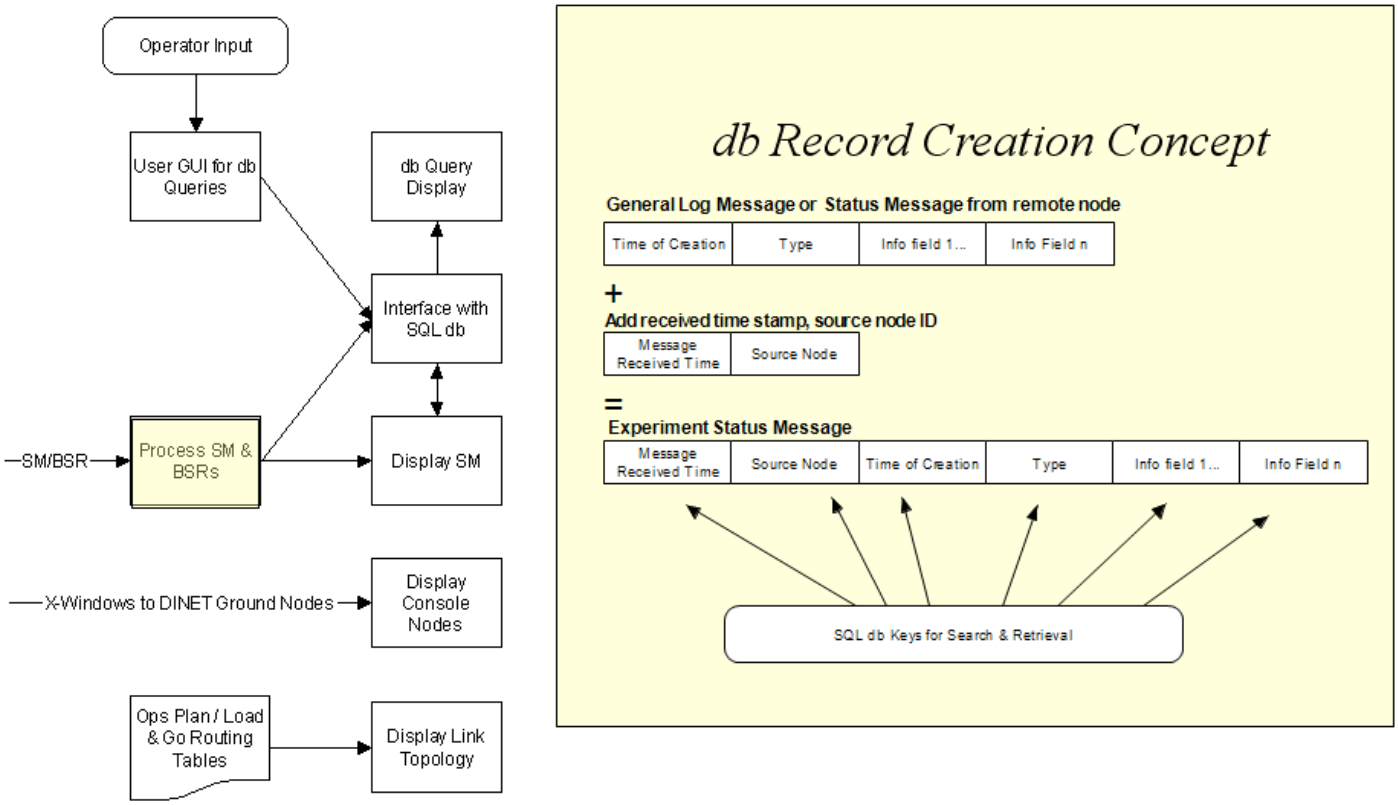


Figure 26 Element Function and Modules

Personnel

The EOC was staffed by one full-time network engineer to perform system administration and experiment operation. Four more EOC-specific staff members served as developers and testers on a part-time basis. This is illustrated in Figure 27.

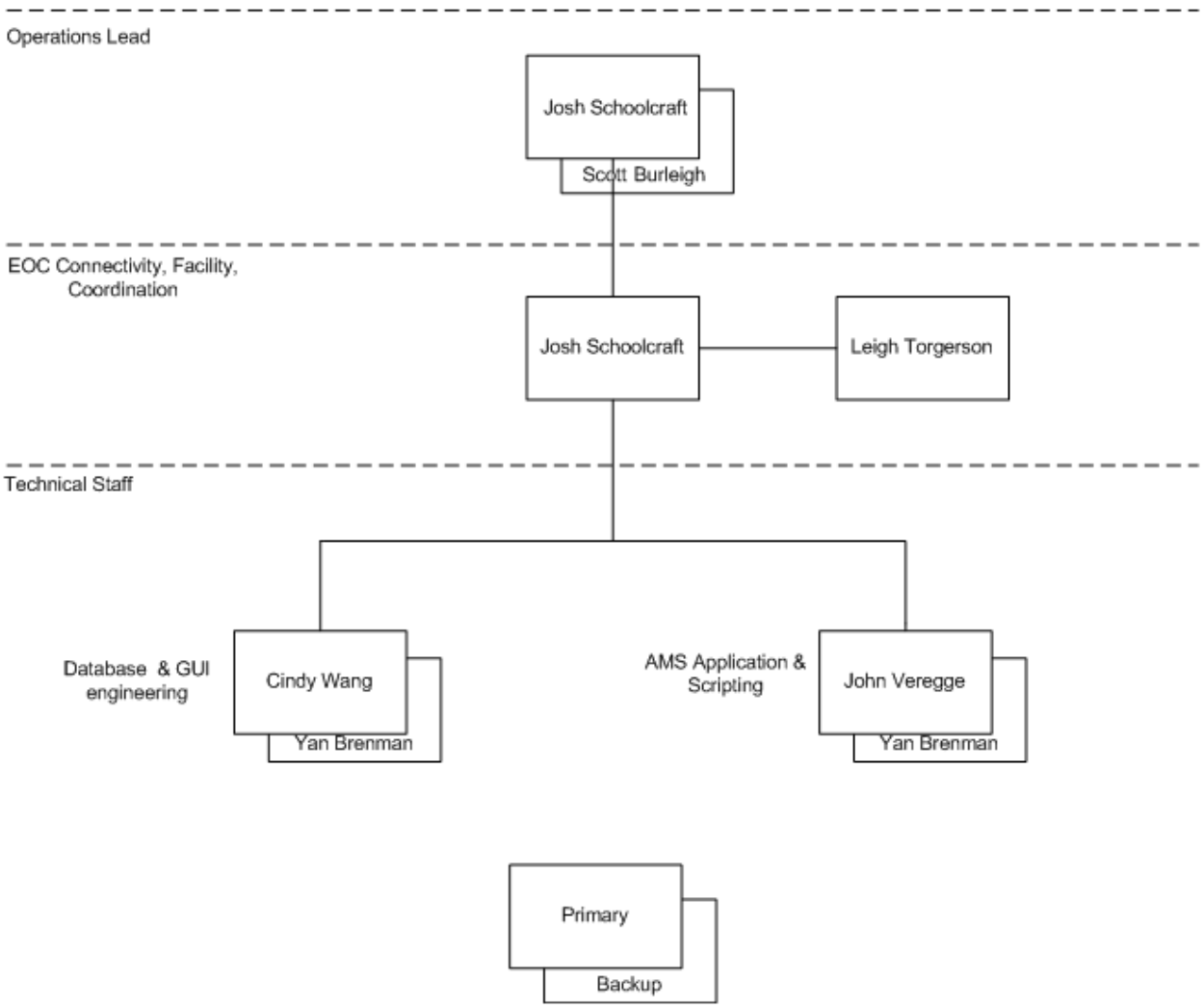


Figure 27 Personnel Overview

Operations Performance

Operations in the EOC included monitoring of data flow during experiment, data collection and distribution after experiment passes, time management of ION software aboard EPOXI, and generation of small data bundles for transfer through the DTN (in addition to automated image publishing). The EOC software GUI display and data store allowed excellent real-time visibility into network behavior, allowing rapid response to and analysis of experiment events. This success can be attributed in part to well-defined requirements guiding the EOC development team through design, assembly, and test phases. The selection of widely available open-source tools and support for code development on PTL systems also contributed to success, specifically under budgetary and temporal constraints.

4. Experiment Results

4.1. Findings

4.1.1. Metric 1 – Path Utilization Rate (U)

Validation criteria:

- $\mathbf{U_a > 90\%}$. (DTN uses both high-rate and low-rate links efficiently.)
- $\mathbf{U_b > 90\%}$. (DTN remains efficient despite an increase in the rate of data loss.)

Analysis of the DINET experiment log indicates that $\mathbf{U_a}$ was 76.2% and $\mathbf{U_b}$ was 72.4%.

Note, however, that passes 2 and 8 were underutilized due to insufficiency of offered uplink data as discussed later, so their path utilization rates do not accurately reflect protocol efficiency. Additionally, note that about 20% of available uplink capacity was consumed by link service overhead, mainly telecommand coding. When only passes 1, 3, 4, 5, 6, and 7 are considered and all non-DTN overhead is subtracted from available transmission capacity, $\mathbf{U_a}$ and $\mathbf{U_b}$ are 97.4% and 92.5% respectively. With these provisos, both validation criteria were satisfied.

Note that the increased data loss rate in configuration *b* was found to correlate to a reduced path utilization rate as expected.

4.1.2. Metric 2 – Delivery Acceleration Ratio (G)

Validation criteria:

- $\mathbf{G_a > 1.05}$ (Prioritization accelerates the delivery of urgent data.)
- $\mathbf{G_b > 1.1}$ (The advantage of prioritization increases with the rate of data loss.)

Analysis of the DINET experiment log indicates that $\mathbf{G_a}$ was 1.10 and $\mathbf{G_b}$ was 1.12. Both validation criteria were satisfied

4.1.3. Metric 3 – ION Node Storage Utilization

Validation criteria:

- The total number of bundles for which custody is refused anywhere in the network for the reason Depleted Storage, throughout each configuration, is always zero. (We never run out of storage anywhere.)
- $\mathbf{N_{X7} = N_{X6}}$ for all values of X. (Storage utilization stabilizes over the course of network operations.)

Analysis of the DINET experiment log indicates that both validation criteria were satisfied, ***except*** that $\mathbf{N_{X7}}$ was 156,816 bytes less than $\mathbf{N_{X6}}$ for node 10 (only). N_{10} had remained constant from passes 4 through pass 6. We suspect that some new functionality requiring additional storage space—possibly not related to the DTN protocols—was initially exercised on node 10 after pass 6 and prior to pass 7; analysis is continuing.

4.1.4. Metric 4 – Multipath Advantage

Validation criteria:

> **The multipath advantage for traffic from node 20 to node 8 is greater than 20%**. (Dynamic routing among multiple possible paths increases the total network capacity from Phobos to Earth.)

The computed multipath advantage for traffic from node 20 to node 8 through the entire DINET experiment is 27%. Thus, the validation criterion was satisfied. Note, however, that errors in the implementation of dynamic routing prevented the expression of this advantage in improvements in delivery acceleration ratio. This metric will be revisited in future DINET experiments.

4.2. Trace Bundles

At least one trace bundle was received by each end node (8, 12, 20) from every other end node, demonstrating the viability of traffic flow in all directions through the network, including direct exchange between science end nodes without Earth in the loop.

Fourteen trace bundles were never received, due to power failure, software restart, and/or various errors in dynamic routing as discussed below.

4.3. Anomalies

4.3.1. DTN-Related Investigations

Apparent image arrival out of priority order in pass 2

October 22, 2008

Bundles queued for transmission are forwarded in strict priority order. Why, then, was the first image received at the Earth node during pass 2 a priority-0 bundle, followed by priority-1 bundles?

During the third contact of pass 1, the EPOXI node sent all bundles it had received to Earth and then pended, waiting from another bundle to send. During the first contact of pass 2, the Phobos node completed transmission of a priority-0 bundle that it had begun transmitting at the end of the first contact of pass 1. When this transmission completed, the newly received priority-0 bundle was immediately grabbed for transmission to Earth by the EPOXI node, but there was no contact with Earth at that time, so transmission pended. Meanwhile, the Phobos node proceeded to send another 11 priority-0 bundles to EPOXI, and then the Mars node sent 7 priority-0 images and 14 priority-1 images to EPOXI during the second contact. When the third contact of pass 2 began, the EPOXI node completed transmission of the pended priority-0 image and then proceeded to send its other buffered images, starting with the priority-1 images received from the Mars node.

Underutilization of link in pass 2

October 22, 2008

Path utilization for pass 2 is sharply lower than for passes 1, 3, and 4. Why?

In short, there was too little data buffered at the Phobos node to fully consume the uplink opportunity for pass 2.

By the start of pass 2, the Phobos node had received a total of 1,129,974 bytes of image data. Many other images had been published at the Phobos science end node but had been buffered at node 10 pending the cross-link opportunity to node 6, which would enable this additional traffic to take advantage of the long contact from Mars to EPOXI during pass 4 when there would be no Phobos/EPOXI link. Only two days had elapsed since the end of pass 1, not enough time for newly published data from node 20 to fill node 5's buffers for transmission to EPOXI.

Node 5 (Phobos) transmitted 781,764 bytes to EPOXI during pass 1, leaving 411,210 bytes. It received an additional image of 49,622 bytes during its pass-2 opportunity (the cross-link buffer from 10 to 6 was fully subscribed by this time), so it transmitted a total of 460,832 bytes to EPOXI during pass 2, leaving no locally buffered data. This constituted a contact under-utilization of about 300,000 bytes.

<u>Loss of advantage provided by alternative route (cross-link between nodes 6 and 10)</u>

Throughout data production

What happened to the Phobos images, many of priority 1, that were buffered for transmission on the cross-link? Why weren't they transmitted by the Mars node during pass 4 in preference to the priority-0 Mars images?

This is the result of a software anomaly, an error in the route computation algorithm as exercised at node 6. During the cross-link contact, node 6 received all of these images and forwarded them to node 3, but node 3 refused custody: it determined that they could not be forwarded through EPOXI because its contact with EPOXI was already fully allocated to Mars images of priority-0 and priority-1. The problem is that the "backlog" to consider when making this sort of decision ought to be the backlog of all bundles of the same priority of the bundle to be routed or higher priority, rather than all bundles regardless of priority. The priority-1 images from Phobos should have "jumped the queue" ahead of the priority-0 Mars images, causing routes for those lower-priority bundles to be recomputed as necessary.

Consequently, these Phobos images remained stranded on node 6, where they eventually were destroyed due to time-to-live (TTL) expiration.

<u>Bundle expiration on EPOXI</u>

Several times throughout operations

Why did bundles expire while buffered at the EPOXI node?

It was not possible to convey them to the Earth node prior to expiration of their TTL intervals. Note that all of the expired bundles were of priority 0. All priority-1 bundles received at EPOXI were forwarded to the Earth node, but downlink contacts to Earth were highly constricted during passes 2 and 6. Insufficient contact time to clear out the buffers at EPOXI resulted in retention of the lower-priority bundles for transmission during a future contact, but TTL expired before that contact occurred.

Normally we would expect contact graph routing (CGR)-based route computation to anticipate the constrained contact opportunities and simply reject the low-priority bundles when sourced, due to "no known route". However, several contact opportunities were abbreviated for various reasons, and retransmission also consumed some contact time. The result was delayed forwarding of the low-priority bundles.

For example, a priority-0 bundle of size 39888 was sourced by the Phobos science node at 18:10 on 23 October, between passes 2 and 3. It was eligible for enqueuing on the cross-link from node 10 to node 6, because the subsequent pass-4 link from EPOXI to Earth would have delivered it prior to TTL expiration, but it was instead enqueued at node 5 because the cross-link was already fully subscribed. However, because its priority was low it was not fully transmitted to EPOXI during the first contact of pass 3; moreover, completion of its transmission session could not occur during pass 4 because that pass had no Phobos/EPOXI contact. So the last segments of this bundle arrived at EPOXI only during the first contact of pass 5—early in the morning of 4 November—by which time the bundle's 10-day time to live had expired. The bundle was reassembled from its constituent segments but then immediately destroyed.

<u>Underutilization of link in pass 8</u>

November 13, 2008

Command Modulation Generator (CMG) failure during pass 8 somehow caused the Mars node to hang, so that bundle flow did not resume even when the CMG was restarted. In an attempt to get the node running again we inadvertently restarted ION on the node when we really wanted only to restart the FDM process; this resulted in the loss of all data buffered at node 3 for transmission to EPOXI. We were able to reload the Mars node with images published at the Phobos science end node and routed to Mars over a newly created cross-link contact, but this recovery activity consumed about 1¼ hours of the Mars-EPOXI contact interval; during that time, no bundles were presented for transmission to EPOXI. This constituted a contact under-utilization of about 1,125,000 bytes.

<u>Custody refusal at node 5 due to redundant reception</u>

October 27, November 11

Why was bundle custody refused on two occasions for the reason "redundant reception"?

On two occasions, a "bptrace" text bundle sent from node 8 to node 12 via node 7 was refused by node 5 for the reason "redundant reception" following prior refusal due to "no known route" (as described in "Aggregate capacity overflow" below). This was due to a bug in custody refusal that resulted in node 5 believing that it already had taken custody of the bundle.

<u>Unexplained "watch" characters</u>

Throughout operations

What causes watch characters indicating "TTL expiration" to be printed at times when no time-to-live expirations are noted in statistics reports?

Unknown. This question remains under investigation.

<u>Aggregate capacity overflow</u>

Notably October 22 and November 6

Why can some bundles not be routed properly through the crosslink?

CGR erroneously failed to compute a route to a neighbor connected by a long-duration contact, because the aggregate capacity of this contact was so large that it overflowed the 32-bit integer in which it was stored. This caused anomalous routing activity from nodes 8 to 12. Bundles sent from 8 were routed to 5 by 7 in order to take advantage of a future crosslink between 6-12; these were always rejected, though some were able to be rerouted to 3 by 7.

This bug is listed as JIRA item DINET-107 and a more robust mechanism for computing the aggregate capacity of a contact is being developed.

4.3.2. Software Anomalies

Spontaneous statistics reports

Throughout operations

Statistics reports were produced by the DINET nodes running on Linux hosts at times other than the times of contact initiation and termination. This was due to a race condition in the `bpclock` daemon, which has already been corrected.

Unreachability of node 16

October 18, 2008

Bundle status reports produced by the EPOXI node and destined for node 16 were never transmitted because CGR was unable to compute a route from node 7 to node 16. This problem remains under investigation.

Backlog calculation must be priority-sensitive

Throughout operations

See "Loss of advantage provided by alternative route" above. A fix for this bug is being developed.

EVR forwarding connection through Data Control firewall times out and is lost

Several times through operations

The ground software utility that extracts EPOXI's DINET status messages from EVR packets and sends them through the Flight LAN firewall to the Experiment Operations Center attempts to use an open connection to EOC for this purpose. However, the Flight LAN firewall automatically closes this connection when it has been inactive for an hour, so the first EVR status message received after an hour of inactivity fails transmission to EOC and is lost. Fortunately, EVRs could still be retrieved immediately at the Mission Support Area (MSA) using Packet Show (telemetry viewing system) (PKTShow) and can also be retrieved afterwards by querying the PKTShow. In the future, the utility needs to reopen the connection to EOC for each EVR.

Incomplete bundle destruction on custody refusal

See "Custody refusal at node 5 due to redundant reception" above. This bug has already been corrected.

4.3.3. Hardware Anomalies

CMG overload

October 20, November 3, November 11, November 13

A heavy volume of command data transmission, such as DINET's Mars/EPOXI and Phobos/EPOXI contacts, can cause the CMG module at various DSN stations to overheat and shut down; switching to the backup CMG takes some time, resulting in some loss of contact opportunity. This problem is not local to a single station, and it remains under investigation.

4.3.4. Environmental Anomalies

Power failure

November 13, 2008

A JPL-wide power failure on November 12 caused all DINET nodes in the EOC to be terminated. This resulted in the loss of all data enqueued at node 10 for transmission to node 6 via cross-link, but (see "Loss of advantage provided by alternative route" above) this data would almost certainly have been refused by node 3 anyway. The material effect on the experiment was minimal.

4.3.5. Procedural Anomalies

Restarting FDM on node 3 restarted DTN as well

November 13, 2008

See "Underutilization of link in pass 8" above. According to the contingency documentation, the decision was made to restart the FDM server, but inability of readily available procedures or expertise to restart only the FDM system on node 3 caused a restart of not only the server but also the local DINET software. This had two unfortunate effects: it resulted in underutilization of the Mars/EPOXI contact, as described earlier; and it increased the difficulty of diagnosing the suspension of node operations. Procedures and/or scripts providing this detailed operational flexibility will be helpful for future DINET experiments.

Slow hand-off between DSN stations in Pass I

October 20, 2008

During the first pass, a scheduled handover from DSS 26 to 45 was performed slower than expected causing loss of data due to bundle expiration.

Incomplete FDM swapping in DSOT during FSW upload

October 18, 2008

The ACE swapped to the correct FDM during the FSW upload without selecting CFDP as the data transfer mode; rather it was left in Spacecraft Command Message File (SCMF) mode. Fortunately, this had no impact on the FSW Upload pass, and this anomaly did not reoccur during the remainder of operations. Future ground software should have automated switching between FDMs and also between data transfer modes.

4.4. Significance of Results and Comparison to State of the Art

4.4.1. Current Deep Space Communications Methodology

Data retrieval from a single spacecraft

The normal method of retrieving science and telemetry data from spacecraft is for the spacecraft team to manually schedule each contact with the DSN, decide which data are to be transmitted from the spacecraft, and then to send a command sequence built by the sequence team to command the spacecraft to transmit the selected data to the DSN at the selected time.

If, due to atmospheric conditions or other phenomena, telemetry frames are lost, a human-in-the-loop process is initiated whereby the scientists and engineers examine the data gaps and decide what data are important to retransmit; a new set of commands is then generated and sent to the spacecraft to recover the missing data. In some missions, to facilitate this procedure, thumbnail images are generated and sent along with the full sized images in the hope that if full size images are lost or corrupted, the thumbnails will help the scientists to decide whether or not to have these images retransmitted.

This process is labor intensive, as it involves the spacecraft team, the sequence team and the science team; and it can take several days to decide what should be retransmitted. Meanwhile, valuable on-board storage is unavailable since the spacecraft has to retain the images until confirmation that they have been successfully received on Earth or have been deemed unnecessary.

In addition, the use of thumbnails which facilitates data management and retransmission decisions requires additional on-board processing and uses bandwidth that could otherwise be used for the raw data.

In one current Mars mission, the desire for more downlink time resulted in the lowering of the horizon mask at the DSN station from 20 degrees to 15 degrees. While the X-band data from 20 degrees above the horizon and up was relatively error-free, the data recovered from 15 degrees to 20 degrees above the horizon was prone to errors, and many manual retransmission requests had to be generated. The sequence team ended up devising a non-standard method of automating the development of retransmission command sequences to facilitate the necessary retransmissions.

All of these labor-intensive operations can be eliminated by the use of the DTN suite with the LTP protocol providing for automatic retransmission of bad telemetry frames. The DTN data-priority scheme can automate the priorities of data sent on board, with repetitive realtime engineering telemetry sent on a best-efforts basis, for instance, while science data are always retransmitted as necessary.

The trade space of manually-commanded retransmission versus automated retransmission includes the additional bandwidth needed for retransmission of potentially useless data (and the thumbnail method of providing clues as to what is missing), the manual labor needed to decide what needs to be retransmitted, and the management of storage and processing power on board the spacecraft.

Preliminary calculations of the extra bandwidth needed for retransmission are the subject of show that the small percentage of extra bandwidth needed to retransmit potentially useless

science data (e.g., missing portions of an image showing parts of the sky) is a very small price to pay for the savings in work effort by the various teams and the benefit of being able to release storage assets immediately when using the built-in reliability features of LTP.

This percentage is also more than offset if the use of LTP eliminates the need for the generation and transmission of "thumbnail" pictures of the full size image as is done on the Mars Reconnaissance Orbiter (MRO).

The error rates on X-band links are relatively small. However when using Ka-band, automatic retransmission will be much more important since Ka is more susceptible to weather-induced errors. Depending on weather at the various DSN stations, as much as 20% of the downlinked data could be lost or corrupted, and the workload to try to manually manage that would be large. The DTN protocol stack is ideal for automating the management of data flow and will be an enabling technology for the future widespread use of Ka-band.

Good top-down mission system engineering will be necessary, as the impact of automatic retransmission and its interaction with prioritization must be carefully considered. For example, in an encounter mission, bandwidth used for retransmitting portions of older images may have adverse effects on returning closer-in images unless the newer images are given higher priority. In the case of a rover with robotic tools, the opposite may be true; complete images of the exact position of a target of interest are needed before plans on how to deploy drills or other sample tools can be made, so any retransmission of these images must be at higher priority than other traffic.

Multimission Data Relay Operations

Currently, data from surface assets (e.g., Mars Exploration Rover (MER) or Phoenix) are sent from the surface to an orbiter (under the command and control of one spacecraft/science team), and subsequently relayed back to Earth by the orbiter under the control of another spacecraft team, which may even be from another space agency (e.g., data relay via Mars Express).

The successful accomplishment of a simple two-hop relay scenario requires twice the manual work described in the previous one-hop scenario, plus coordination between teams. This method of operations simply doesn't scale well; if relay operations are being conducted through multiple spacecraft owned by multiple teams, the coordination logistics can become unmanageable very quickly.

With an approach of using standardized DTN techniques (that is to say, the use of a standard set of protocols per Internet Engineering Task Force (IETF) or CCSDS specifications to insure interoperability), the coordination problem distills down to communications schedules as defined by pre-determined orbital geometry and DSN scheduling.

DTN will be an enabling technology for previously impractical applications such as the automated and reliable relaying of data from sensor networks on the surface of other planets.

4.4.2. DTN Compared to Current Military Communications State of the Art

The use of Internet technology in the military is vital and significant effort has been expended to make Internet protocols work well over satellite relays. Field commanders rely on satellite links and networking back to the Pentagon, and laptops are ubiquitous in the field.

The problem with this reliance on use of the Internet is at the edges of the battlefield where connectivity isn't always continuous. The Internet protocols currently in use can manage loss due to corruption (using CCSDS Spacecraft Communication Protocol Standards–Transport Protocol (SCPS-TP) for instance, which is in use in military satellite communications and link asymmetries, but they cannot handle the frequent disruption of end-to-end link connectivity.

The Defense Advanced Research Projects Administration (DARPA) has had a DTN program in place for a number of years, and this program is moving rapidly towards fielding DTN capabilities for these edge networks. The Wireless Network After Next (WNAN) program is building inexpensive DTN radio nodes that will allow the use of DTN to ensure data get into the hands of soldiers in the field in highly fluid tactical environments where continuous radio connectivity to the wider military networks cannot be insured. This program is on a fast-track, as both the United States Army and the United States Marine Corps are anxious to field DTN and improve battlefield communications.

4.4.3. Comparison with Terrestrial Internet

The main point of comparison with terrestrial internet technology hinges upon the need for continuous end-to-end connectivity. With TCP/IP, a complete path through multiple routers must exist between the two computers for a connection to be established and for the reliable transmission of data. If, during the data exchange transaction, a link from one router to another goes down, or if one of the routers becomes too congested with traffic to handle the transaction, the packets are dropped and the transaction is effectively terminated. Reestablishment of the transaction must wait until once again there is a continuous end-to-end path available.

This effect and the advantage of DTN are best illustrated by relating a recent experiment by the University of California Los Angeles (UCLA) Center for Embedded Networked Sensors. A linear array of seismic sensors were placed in a local mountain range, spaced about 1 km apart, and connection between each sensor was via 802.11 wireless.

Using TCP/IP, in order to retrieve the data from the farthest sensors, all radio links had to be up and operating. As may be imagined, this was a spotty proposition; 802.11 links often went down because of atmospheric effects, trees blowing, etc., and as a result, the data from the farthest sensors was hard to recover.

UCLA took the DTN protocol and installed it in their sensors, so the data could simply be relayed from node to node when the links were up, storing the data at each link until the next hop became available. As a result, all data could be recovered. UCLA subsequently used the DTN-derived techniques in the MesoAmerican Subduction Experiment, which involves an array of sensors across Mexico for hundreds of kilometers. [3]

DTN store-and-forward techniques are also being used in a number of other terrestrial applications, such as the University of California (UC) Berkeley Tier Store project, which uses DTN to provide Internet services to remote villages in India where there is no connectivity. A similar effort is in use by the Saami-Net project in Sweden, where DTN is used to bring email and other services out to the nomadic Saami following their reindeer herds. [4]

In both these cases, the principle is that a portable DTN node on a bus or snowmobile takes all transaction requests from local users; when the portable device is driven (or ridden) to an area of

regular internet connectivity, the transactions are completed, and the results (such as received email) are stored in the DTN node to await the return trip to the remote users.

With the increased popularity of wireless applications , it is expected that DTN will play a large role in the terrestrial internet. To this end, our development of DTN over the last 10 years has been conducted in conjunction with the Internet Research Task Force (IRTF) , which established the DTN Research Group. The IRTF is an international team of researchers that has collaborated on the development and standardization of DTN. While our current use on DINET is tailored specifically for spacecraft communications, it is completely interoperable with the Delay/Disruption Tolerant Networking Research Group (Reference Implementation 2) (DTNRG DTN2) public implementation and follows the same Experimental Request for Comments (RFC) standard [1], so future use of DTN may extend from terrestrial DTN applications to our space applications.

4.4.4. Comparison with SSTC-UK-DMC Satellite Test

In September 2008, a test of some features of the DTN Bundle Protocol (BP) was performed by the NASA Glenn Research Center (GRC) using a United Kingdom satellite, UK-DMC, in low Earth orbit at an altitude of about 100 miles (160 km). In this experiment, an implementation of the bundle origination, proactive fragmentation, and transmission procedures of BP was installed on the UK satellite and was used on two occasions to transfer an image—split into two fragments Glenn Research Center from the satellite to a DTN node at Surrey, UK, over a convergence-layer protocol stack based on the specification for the Saratoga file transfer protocol; the Surrey ground station node automatically forwarded the fragments to a third DTN node at GRC. On one occasion, the image suffered data corruption in transit. The GRC convergence-layer protocols were able to detect this corruption but not correct it. On the other occasion, the image was successfully received at GRC.

In October and November of 2008, the NASA Jet Propulsion Laboratory (JPL) installed and tested fully conformant implementations of both BP and LTP on EPOXI, a NASA spacecraft in interplanetary space at a distance of 10–15 million miles (16–24 million kilometers) from Earth, and on nine other computers at JPL. In this experiment, some 300 images were transmitted from the JPL nodes to the spacecraft and then automatically forwarded from the spacecraft back to the JPL nodes, exercising DTN's bundle origination, transmission, acquisition, dynamic route computation, congestion control, prioritization, custody transfer, and automatic retransmission procedures, both on the spacecraft and on the ground, over a period of 27 days. All transmitted bundles were successfully received, without corruption, despite several transient unanticipated lapses in service at DSN stations during tracking passes.

The GRC UK-DMC test was a valuable initial proof of the concept that DTN bundles may be constructed on-board a spacecraft and used to transmit data.

JPL's EPOXI test demonstrates that JPL’s comprehensive DTN software suite is ready for operational use in flight missions.

4.4.5. Significance of Results

The significance of the Deep Impact Network Experiment can be appreciated through two complementary perspectives. The first focuses on the technical significance, with an emphasis

on the quantitatively defensible and tangible results that can be drawn from the four-week experiment. It answers the narrow question of what DINET has proved. The second perspective takes a longer view, concentrating on the strategic significance of DINET. It answers the broader question of the importance of the experiment's achievements, in both current and future timescales.

Technical Significance

DINET proved that Delay-Tolerant Networking can work in deep space. During the four-week experiment, DTN was successfully demonstrated over a variety of conditions for a ten-node network topology including the EPOXI spacecraft. The DTN protocols that governed the traffic management, data dissemination and routing functions of a network were exercised over a representative topology, with a realistic traffic pattern and characteristic application data. Software adaptations for EPOXI's flight and ground systems were developed to allow the networking protocols to act within an end-to-end information system setting.

It should be noted that the topology modeled a more complex network than today's standard of Martian communications with a two-hop maximum. Unlike current Mars practice, network functions were completely automated. It is true that some planned and unplanned actions by DINET operations personnel were necessary at times to manage the experiment, but the network/data management functions themselves did not require intervention.

All validation objectives of the experiment were met, accounting for the appearance of anomalies. One class of anomaly, the unexpected loss of DSN transmission capability that occurred on several occasions during the experiment, demonstrated the robustness of the JPL implementation of DTN. In those situations, the protocols automatically identified the missing data, and they coordinated and activated selected retransmissions, completely and rapidly recovering the data while imposing no extra operational burden. In fact, all image data received through the network were compared at the bit level to the original data injected into the network and shown to be identical, indicating the total lack of data corruption.

Strategic Significance

DINET raised the technology readiness of DTN to its highest level thus far. It clearly demonstrated a system in a space environment, and if it were only a prototype, such would qualify as TRL 7. But a case can be made that DINET went beyond the prototype stage, as the networking protocols employed were those intended to be used universally. While some needed improvements were identified during the experiment, the maturity, robust performance, and adherence to publicly available standards of the DTN software allow it to be used again on different missions with different topologies. Such a situation argues for TRL 8. While the managers of future missions may desire an additional cycle of code formalization and documentation for a higher level of supportability before committing to complete adoption, the DINET code is available for immediate use in its current form.

The experiment, by chance, uncovered a limitation of the current DSN that has significance for Internet-like operation of space networking. Historically, the downlink data volume from scientific space probes has dwarfed the uplink volume, as the tightly packaged command information has never approached the size of the resultant science measurements. So the DSN never had to accommodate significant uplink volumes. With DTN, large uplink volumes are a

possibility, and the DINET experiment attempted to send data to the EPOXI spacecraft in such high volume that the DSN's CMG system failed several times from possible overheating. Such an uplink limitation had not previously been noted. It highlights the pre-Internet mindset that many legacy communication systems have physically manifested and currently operate under.

The experiment's successful demonstration of the priority-aware relay aspect of DTN offers significant promise for better network utilization of existing bandwidth and improved end-user satisfaction.

A significant strategic result with both immediate and future consequences is a reduction in reluctance of the space flight operations community to host networking technology with a high (if not complete) degree of autonomy. Indeed, the EPOXI spacecraft team itself had to be convinced that the DINET software and operations plan posed no serious threat to the safety of their spacecraft. They later became advocates of DINET's inherent safety. Some benefits of DTN were also grasped by them as being desirable for science operations such as the EPOCH science investigations completed a few months prior to DINET.

DINET showed the ability to of a space network to exchange data between its constituent nodes with Internet-like automation and the resultant low operations labor costs. As networks grow in complexity, the time and effort needed to manually schedule and coordinate link activity quickly becomes unmanageable; DTN allows space networks to scale without such constraints. In addition, the ability to route information automatically between space vehicles in local proximity without incurring the potentially long one-way light time delays and Earth-based decision cycles of human-managed communications offers the possibility of new types of coordinated science that qualitatively differ from current capabilities. DTN should help enable cooperative, reactive science functionality for remote spacecraft networks.

5. Lessons Learned

While the development of ION took several years of work by JPL and its partners, the culmination of this effort in the form of DINET took less than a year. The rapid progression of development, integration, testing, and execution provided ample experience in quickly readying a communications protocol for use in the space environment. This experience in turn imparts numerous lessons on managing future DINET experiments and other small team technology development experiments.

5.1. Information Management

As unit testing proved successful and matured into system-level testing, information management became a vital tool to assure that all members of the team were consistent in their knowledge of testing progress, system components, and terminology. Moreover, consistent and consistently updated information management frames the basis of other lessons for future experiments.

From the beginning, a communal information source (particularly a wiki) forms the basis of proper information management. At the unit test level, those responsible for their system components should update their related wiki pages with test procedures and test results in general language. All emails referencing test progress should include links to the related wiki page. It is

crucial that updating the experiment wiki is a common task associated with formal and informal aspects of the experiment. This provides ongoing context for testing progress. Background documentation, including theoretical and previous work, should also be cached on related unit, or special, pages.

As system testing starts, the systems engineers will have a deep understanding of each unit with the context provided by each unit's wiki entries. As system testing continues and new responsibilities form, it is essential to assign individual team members to maintain those responsibilities. This way, no responsibility is left behind and a consistent voice is supporting each one. The ensuing understanding can then forge a competent and efficient testing plan with all unit specialists well-versed in related parts of the system. Administratively, consistent and properly updated documentation aids in developing coherent reviews and reports to clients.

Otherwise, the lack of proper information management may cause slow downs, halts, failed tests, and other complications that are difficult to mitigate once they appear. If individuals fail to update their unit wiki, systems engineers will not necessarily have a clear understanding of how tests are performing or details associated with that unit leading to inefficient or downright impossible timelines. If a unit's wiki page is incomplete or is not clear, test operators and systems engineers may be unable to successfully run tests that would otherwise be possible, slowing down the progress of the testing schedule.

While the wiki had existed for DINET since the beginning, it was not fully utilized until system testing. This forced systems engineers to spend extra time in consulting each unit specialist in a time-consuming and potentially inconsistent way. Moreover, information between units was exchanged occasionally through email and weekly meetings. While DINET did not face any mission-endangering consequences from not having an early, widespread adaptation of the wiki, it is possible that it would have saved time through unambiguous communication.

In general, information management through a wiki may seem trivially important, leading some team members to disregard it as a superfluous product of overzealous administrators, but it is a integral player in maintaining an effective development regiment.

5.2. Requirement and Time Management

Without proper requirement and time management the project cannot continue forward. Once requirements have been developed and refined they can be assigned to tests that are responsible for meeting those requirements. By mapping these associations, a critical path of testing will emerge that takes test dependencies into account. This critical path will provide a logical order of testing that validates the most important requirements first. This way, in the event of delays in testing process, superfluous tests can be dropped for a descoped testing regimen.

In order to successfully manage both requirements and time during testing, there are several simple rules that should be followed. First, the sooner unit tests are developed and the requirements are mapped, the sooner early system tests can be formulated and run. DINET benefited from running early system tests to diagnose issues in interfacing various units. While these issues were being resolved, other unit tests could be completed so that the testing regimen could continue unhampered. Second, the testing schedule should be constantly updated to reflect completed and delayed tests. Third, as system testing continues it is important to inspect mapped

requirements to verify that they are still relevant and assigned to the right tests. By properly maintaining the requirements, none will be missed and the testing results can be more easily integrated into reviews.

Not following these three rules, particularly through inconsistent updates of the requirement mapping and timeline, may lead to testing delays and inefficient utilization of testing staff.

5.3. Configuration Management and Software

The practice of configuration management (CM) plays a key role in all testing, and the absence of effective CM will undoubtedly lead to failed tests and a dramatically slower pace of testing.

There are two key lessons regarding CM learned based on progression of DINET testing; first the centralization of sources of key files, second the shepherding of all file-updating responsibilities to as few people as possible. Through the latter portions of the DINET experiment testing we were able to implement Subversion (SVN) file repository as a method to centralize the distribution of key files used in testing. Many of the nodes in the network would undergo an SVN update prior to a test in order to make sure everyone was utilizing the same software and script versions. This system can be improved by expanding it to all nodes on the system for all the software used in the experiment. For instance the non-ION software used on the DSOT nodes for system testing was locally maintained by programmers and operators. Configuration management was done informally through email, causing occasional miscommunication when emails had not been sent or wrong emails were read. By using a software repository system like SVN, testers will know exactly what version files they are using for each test, assuring uniformity throughout the network.

The process of CM through maintaining a software repository can also be expedited by assigning people to be responsible for particular files. All updates by anyone on the team would be routed through the person responsible for that particular file so issues of redundant and multiple different files being used simultaneously will not happen.

Lastly, the streamlining of the user interface for software plays an integral role in safe and error-free operation. During system testing, personnel shortages forced the use of untrained operation at the simulated DSOT nodes. Because the software at the nodes was rather cumbersome, a few tests were scrubbed due to incorrect execution of number of scripts, which need to be run for the test. Eventually, an overarching script was developed, which all but stopped the scrubbing of tests due to the aforementioned operator error. During operations, a simple, scripted alarm clock was developed by the DINET team to provide warnings for operational events. This intuitive software kept operators aware during operations and may have avoided human error due to forgetfulness.

5.4. Testing

For all projects, testing is a trying time during which the project team needs to integrate all the different units of the project and to quickly respond to anomalies that present themselves.

Above all, information management plays the most important role in system testing. The reality of running a system test is that several people with limited communication between them, in different buildings, are trying to simultaneously accomplish different, mutually dependent

objectives through individual efforts—simply put, human errors are inevitable. In order to minimize the errors, thorough procedures need to be written for each testing station or individual, and then these procedures must be distributed and read by all testers prior to testing. This is a responsibility not only for the systems engineer who writes and distributes the documentation, but also for each tester as well. Moreover, procedures need to be followed at all stations for even the most informal of tests. After all, procedures that are not followed will result in inconclusive test results and poor practice during operations. This was particularly noticeable in DINET's propagation of the time synchronization file which was sometimes haphazardly done in the testbed location (containing simulated DSOT and testbed nodes). Numerous tests were scrubbed due to improper synchronization, causing approximately an hour of delay each time. As testing progresses and ad hoc procedures are developed, such as calling between stations and verifying file versions, these procedures should be integrated into documentation to formalize them for future reference and for debriefs of failed tests.

The development of the testing schedule is just as important as information management in testing because it sets the pace for the entire project. The DINET experiment had three different unit testing categories: ground support software (itself divided between software for DSOT and EOC), ION software, and spacecraft adaptation. In order to streamline testing, unit tests were done in parallel as much as personnel and testing precondition constraints could allow. This enabled early unit-unit interface tests to meet our testing philosophy of

"Test early. Test often."

This decision proved very successful in rapidly adapting the various units to the DINET system as a whole. During this period, an overarching schedule was developed for system testing. This schedule provided reasonable margin for tests to take unforeseen complications into account. Our system testing was rife with complications: changes in facility, intermittent availability of a working testbed, damage to critical system hardware due to construction workers not taking safety precautions, key personnel being on leave, and technical issues that arose from testing. Despite all these complications, DINET remained on schedule for the bulk of testing due to conservative scheduling. While DINET operations occurred two weeks later than initially scheduled, it was safe within the two month operations window given by EPOXI staff.

Several important lessons were learned through various types of tests run on DINET software. One such lesson we learned was the role of long-duration testing. For most of system testing we relied on 1.5-hour tests, which were unable to expose some flaws in our code. The longer overnight tests uncovered a memory leak in the administrative node GUI at the EOC as well as routing issues in the network. Had we not done long-duration testing prior to the software freeze, our operations would have produced significantly less data. Another lesson emerged when we performed off-nominal testing on DINET. While we included some off-nominal testing early, it was trivial in scale compared to what was scheduled toward the end of system testing. This second round of off-nominal system testing uncovered many bugs in software throughout the system requiring a iterative develop-on-the-fly and testing procedure that took longer than expected.

Prior to operations, the DINET team engaged in an intensive Operational Readiness Test (ORT) period during which we ran through abridged operations twice as well as engaged in a series of post-ORT tests. The ORTs provided a very strong exercise for operations as they finalized

operational procedures but also gave the DINET team and other operators experience in dealing with anomalous events during operation. The second round of testing in the ORT period gave the DINET team extra time to understand exactly how the network would behave during operations. By observing this behavior, detecting anomalous network activity was a lot easier for operations.

5.5. Team Morale

Like any other team activity the upkeep of team morale plays a large role in its performance. While DINET did not experience any problems with morale, there were several decisions which helped maintain good performance.

Primarily, a realistic outlook on milestones and requirements potentially saved the project money and team morale. For example, while approaching our ORTs it became apparent that our pace of testing would either require multiple shifts for testing and development or else a schedule revision. After consulting with EPOXI staff, the decision for a delay was made. This choice saved the team both money and extra time, which might still not have produced the required progress for an on-time ORT.

Second, jovial attitude among the team through the final portions of testing and operations was maintained by having an ample supply of pastries and espresso on hand at the EOC. These were also shared with staff associated with the project in other stations, such as DSOT and the MSA. This simple addition helped keep spirits and energy up through long and late hours and certainly boosted performance.

6. Future Work

With the successful completion of the DINET experiment and raising the TRL of DTN to 8, we have begun to look forward to increase the functionality and the operability of the DTN protocols through further development and testing. We intend to continue our work in the DINET II experiment, which will provide several enhancements of the DTN software and will also demonstrate DTN over a larger and more complex network topology.

6.1. Work for DINET II

The key areas of development and demonstration for DINET II:

- Implementation of unacknowledged CFDP enabling large file transfers
- Implementation of Bundle Security Protocol (BSP)
- Demonstration of dynamic contact graph routing
- Development of a EPOXI bootstrap function
- Inclusion of additional nodes in the experimental network
- Development of automated FDM switching
- Fixing issues remaining from DINET I

- Implementation of extended priority system

6.1.1. Implementation of Unacknowledged CFDP Enabling Large File Transfers

The DINET I experiment was designed to be quickly adapted to the EPOXI infrastructure, which precluded the development of a CFDP implementation for DINET. Instead, DINET used preexisting AMS, which limits bundle size to approximately 64 kb. While this was sufficient to prove the efficacy of DTN, it did not establish a viable method for transferring uncompressed science data from spacecraft. For DINET II, CFDP has to be implemented as a part of the DTN stack. This will allow portability of CFDP across Portable Operating System Interface Application Programming Interface (POSIX API) supporting platforms and aid in technology infusion. This effort to adapt CFDP will likely happen at JPL.

6.1.2. Implementation of Bundle Security Protocol (BSP)

In order to expand the application of DTN in the space environment, protocol security must be developed to provide protection from outside attacks when not using secure links. Moreover, the inclusion of additional nodes in DINET II testing will require secure communications. To this end, DINET II development will include Bundle Security Protocol (BSP),[*] which includes additional blocks in either a header or trailer to authenticate the sender of the bundle. This will allow for secured communication across otherwise insecure media.

6.1.3. Demonstration of Dynamic Contact Graph Management

Throughout the DINET operations, the contact graph on the spacecraft was considered a single point of failure for the mission. We did not have enough time to developed a robust and operationally safe method to change the _ION initialization file (global.ionrc) during operation in the case of failure or to allow the establishment of another window for data transfer. During our final pass of the DINET experiment, changing of the contact graph was done within the EOC, but it has yet to become a standard practice. Changes will be made during the course of DINET II that will enable dynamic revision of the contact graph for operations and future DTN experiments. This will enable a network that is more capable of reacting to disruptions and using opportunistic contacts.

6.1.4. Development of a DTN Bootstrap Function

In order to provide a safe method of restoring connectivity in the case of a contact graph failure aboard EPOXI, the EPOXI team in conjunction with DINET will develop a bootstrap function that creates a short contact with the spacecraft to replace or restore the contact graph.

6.1.5. Inclusion of Additional Nodes in the Experimental Network

Following the success of DINET I, a logical step is to include more nodes in our network to increase network traffic and devise more experiments to test routing algorithms. We are looking into using testbeds at other NASA facilities and a computer aboard the International Space Station (ISS) as potential nodes in addition to all the nodes which participated in DINET I.

---

[*] BSP is still a draft IETF standard. It is currently available at http://www.ietf.org/internet-drafts/draft-irtf-dtnrg-bundle-security-06.txt

6.1.6. Development of Automated FDM Switching

In order to access EPOXI using the ground system software, file data management (FDM) switching needs to occur. Currently this is operated by DSOT during operations, but we intend to fully automate this based on the contact graph. Automated FDM switching is vital to realistic operations in which data seamlessly flow and switching decisions are made without a human in the loop.

6.1.7. Fixing Issues Remaining from DINET I

There are several bugs that were discovered after the software freeze but before DINET I operations. While some of these bugs have been fixed or are in the process of being fixed, the remainder will be fixed early in DINET II testing.

6.1.8. Implementation of Extended Priority System

Throughout the course of DINET I operations and discussions with other DTN partners, it became clear that it would be logical to increase the number of priority levels that are available in ION. This will provide greater usability for network clients and make sure that administrative traffic is able to go through the network without interference from other data.

6.2. Work Beyond DINET II

There are a few additional items not covered in the scope of DINET II that will be addressed in future development and experiments:

- Native AMS on spacecraft
- Network time protocols
- On-board OWLT calculation integration

6.2.1. Native AMS on Spacecraft

The DINET I experiment utilized the EPOXI spacecraft as an internet router, sending bundles from one node to another. In a realistic DTN scenario, spacecraft will be data producers and will require AMS in order to send data to their subscribers. This will require testing to adapt the current AMS software with EPOXI and possibly other spacecraft SCUs in the future.

6.2.2. Network Time Protocols

Successful networking across a delay tolerant network requires strict regulation of clocks that are accessed by the ION stack in order to manage bundles and communication during contact windows. Clock synchronization was done by receiving spacecraft clock/spacecraft event time (SCLK/SCET) files from the spacecraft and predicting clock drift through calculations. This is not a viable strategy for realistic DTN applications in a large network. Significant work will be required to develop robust network time protocols to synchronize clocks throughout a DTN network.

6.2.3. On-Board OWLT Calculation Integration

Due to the dynamic nature of interplanetary navigation, the global.ionrc as initially set will become increasingly inaccurate without changes taking the one-way-light-time (OWLT) into

account. By utilizing the Spacecraft Planetary/satellite ephemeris and constants, Instruments, C Pointing Matrix, Event Info. (Kernels) / Navigation and Ancillary Information Facility (SPICE/NAIF) toolkit, OWLTs can be automatically calculated, both at ground nodes and on-board the spacecraft, and integrated into the contact graph to enable efficient use of contact times.

JPL's EPOXI test demonstrated full end-to-end use of the DTN software suite on a deep space mission. As such this test flight validated DTN for use on space missions.

7. Acknowledgements

The work described in this report was performed at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration (NASA). Reference herein to any specific commercial product, process or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government, NASA or the Jet Propulsion Laboratory, California Institute of Technology. DINET was implemented by the following personnel: Rashied Amini, Yan Brenman, Scott Burleigh, Loren Clare, Micah Clark , Andre Girerd, Son Ho, Nuha Jawad, Ross Jones, Margaret Lam, Marisol Mercado, Amalaye Oyake, Richard Rieber, Joshua Schoolcraft, Leigh Torgerson, Shin-Ywan Wang and Jay Wyatt. The following members of the EPOXI project team were essential to the success of DINET; Steve Wissler, Richard Reiber, Greg LaBorde, Leticia Montanez and Al Nakata.

The Deep Impact Networking Experiment was sponsored by the Space Communications and Navigation Office in NASA's Space Operations Mission Directorate. NASA's Science Mission Directorate and Discovery Program provided experimental access to the EPOXI spacecraft.

The EPOXI mission team provided critical support throughout development and operations. The following members of the EPOXI project team were essential to the success of DINET; Steve Wissler, Richard Reiber, Greg LaBorde, Leticia Montanez and Al Nakata.

Finally, Rich Benson provided DSN scheduling support.

8. References

1. K. Scott and S. Burleigh, *Bundle Protocol Specification*, RFC 5050, Internet Society, Reston, VA, November 2007.

2. M. Ramadas, S. Burleigh and S. Farrell, *Licklider Transmission Protocol—Specification*, RFC 5326, Internet Society, Reston, VA, September 2008.

3. R.W. Clayton, P.M. Davis, X. Perez-Campos, "Seismic Structure of the Subducted Cocos Plate," American Geophysical Union, abstract #T32A-01, Fall Meeting 2007.

4. A. Doria, "Saami Network Connectivity:Technical Overview of SNC." Available at www.cdt.luth.se/babylon/snc; accessed February 11, 2009.

9. Appendix A – Experiment Data

This Appendix contains worksheets summarizing the performance of the DINET delay-tolerant network during its 27 days of operations.

Table 6 presents a detailed analysis of the overhead cost of DTN transmission over command links (uplink). Each layer of the protocol stack imposes some increment of overhead, typically in the form of protocol data unit "header" data. The table illustrates the calculation of the total overhead imposed by Bundle Protocol, by LTP, by the "PX" shim that enabled DINET to "tunnel" through the pre-existing support for CFDP on EPOXI, and by the CCSDS Telecommand link-layer protocol. Note that the total link bandwidth consumed by BP and LTP headers was at most 1% of available bandwidth, while the total link bandwidth consumed by the Telecommand protocol was on the order of 20%.

Table 7 details the total image transmission capacity of each contact opportunity from the simulated science nodes to the EPOXI spacecraft and from the spacecraft to the simulated mission operations center. Known periods of outage are noted. The "throttled" values in the table indicate the data rates to which DTN congestion control limited transmission based on the contact schedule; these rates were often, but not always, identical to the actual rates at which DSN and spacecraft radio frequency (RF) equipment was operating.

Table 8 details the end-to-end image delivery performance of the network. It indicates the volume of data actually received at EPOXI during each contact, the volume of data delivered to the simulated mission operations center on each pass, and the residual content of the data buffers on EPOXI at the end of each pass. Link utilization is computed from these figures in the context of the computed network capacity values from Table 7. Data volumes received at EPOXI and at "Earth" are shown for each of the three levels of priority supported by the Bundle Protocol standard. This enables the worksheet additionally to show the computed delivery acceleration ratio on each pass.

Table 9 is a calculation of the nominal multipath advantage provided by network as configured for this experiment, based again on the capacity figures from Table 7.

Table 10 shows how buffer space from the ION storage pool is allocated to DTN protocol activity. The bundle data storage allocation grows over time until a condition of peak utilization has been reached, at which point it stabilizes and residual storage margin is left untouched. As noted in 4.1.3 above, the additional allocation from residual storage at node 10 on pass 7 is an anomaly that is still being investigated. (Since the ION storage pool is used to support other activities in addition to the exercise of the DTN protocols, it is possible that this additional allocation is not attributable to DTN.)

Table 6. Uplink Overhead

BP header bytes per bundle	25
BP bundle bytes per LTP data segment	734
LTP header bytes per LTP data segment	5
LTP segment bytes per PX PDU	739
CFDP header bytes per PX PDU	11
PX PDU bytes per TC frame	750
TC segment header bytes per TC frame	6
TC frame header bytes per TC frame	5
TC frame text bytes per CLTU	761
Code blocks needed per CLTU	109
Total code block bytes per CLTU	872
Start sequence bytes per CLTU	2
Tail sequence bytes per CLTU	64
Total bytes per CLTU	938
TC overhead per PX PDU	188

Pass	Total uplink capacity	BP Payload images	Delivered bytes	Payload link consumption	Total BP overhead	Total BP bytes	# LTP data segments	Total LTP overhead	Total DTN overhead	DTN ohd link consumption	Total PX overhead	Total TC overhead	Total link svc overhead	PX/TC ohd link consumption	DTN avbl link consumption	DINET link consumption
1	2475000	38	1877402	76%	950	1878352	2560	12800	13750	1%	28160	481280	509440	21%	96%	97%
2	2700000	33	1571920	58%	825	1572745	2143	10715	11540	0%	23573	402884	426457	16%	70%	74%
3	2700000	43	2056933	76%	1075	2058008	2804	14020	15095	1%	30844	527152	557996	21%	97%	97%
4	3150000	47	2410752	77%	1175	2411927	3287	16435	17610	1%	36157	617956	654113	21%	97%	98%
config phase A	11025000	161	7917007	72%	4025	7921032	10792	53960	57985	1%	118712	2028896	2147608	19%	90%	92%
5	2276047	36	1627308	71%	900	1628208	2219	11095	11995	1%	24409	417172	441581	19%	89%	91%
6	2700000	39	2083966	77%	975	2084941	2841	14205	15180	1%	31251	534108	565359	21%	98%	99%
7	2700000	38	1909780	71%	950	1910730	2604	13020	13970	1%	28644	489552	518196	19%	88%	90%
8	3016250	28	1374751	46%	700	1375451	1874	9370	10070	0%	20614	352312	372926	12%	52%	58%
config phase B	10692297	141	6995805	65%	3525	6999330	9536	47680	51205	0%	104896	1792768	1897664	18%	80%	84%
experiment total	21717297	302	14912812	69%	7550	14920362	20328	101640	109190	1%	223608	3821664	4045272	19%	85%	88%
passes 1-7 only	18701047	274	13538061	72%	6850	13544911	18454	92270	99120	1%	202994	3469352	3672346	20%	91%	93%

Uplink consumption on passes 2 and 8 is uncharacteristically low due to insufficient traffic; the link was idle much of the time.

	Mean payload link consumption	Mean adjusted DTN link consumption	Mean total link consumption	Inferred LTP retransmission traffic
Passes 1, 3, 4 (no induced data loss):	76.2%	97.4%	97.4%	2.6%
Passes 5, 6, 7 (3.125% induced data loss):	73.1%	93.5%	93.5%	6.5%

Table 7 Network Capacity

Pass	From node	To node	Start UTC	End UTC	Raw bits/sec	Raw bytes/sec	Raw capacity	Throttle bytes/sec	Throttled capacity	
1	5	7	10/20/2008 11:00	10/20/2008 12:00	2000	250	900000	250	900000	
	3	7	10/20/2008 12:05	10/20/2008 12:19	2000	250	210000	250	210000	
	3	7	10/20/2008 12:19	10/20/2008 12:29	0	0	0	250	150000	CMG crash (est. 10 min. to repair)
	3	7	10/20/2008 12:29	10/20/2008 13:28	2000	250	885000	250	885000	
	3	7	10/20/2008 13:28	10/20/2008 13:33	0	0	0	250	75000	station handover (est. 5 min outage)
	3	7	10/20/2008 13:33	10/20/2008 14:05	2000	250	480000	250	480000	
	(sum for 3)						1575000		1800000	
	sum	7					2475000		2700000	
	7	2	10/20/2008 14:10	10/20/2008 15:00	164768	20596	61788000	20000	60000000	
2	5	7	10/22/2008 15:30	10/22/2008 16:30	2000	250	900000	250	900000	
	3	7	10/22/2008 16:35	10/22/2008 18:35	2000	250	1800000	250	1800000	
	sum	7					2700000		2700000	
	7	2	10/22/2008 18:40	10/22/2008 19:30	867	108.375	325125	110	330000	
3	5	7	10/27/2008 15:45	10/27/2008 16:45	2000	250	900000	250	900000	
	3	7	10/27/2008 16:50	10/27/2008 18:50	2000	250	1800000	250	1800000	
	sum	7					2700000		2700000	
	7	2	10/27/2008 18:55	10/27/2008 19:45	165635	20704.375	62113125	20000	60000000	
4	10	6	10/29/2008 20:00	10/29/2008 20:10	9600	1200	720000	1200	720000	
	6	10	10/29/2008 20:00	10/29/2008 20:10	240	30	18000	30	18000	
	3	7	10/30/2008 0:00	10/30/2008 3:30	2000	250	3150000	250	3150000	
	7	2	10/30/2008 3:35	10/30/2008 4:00	165635	20704.375	31056562	20000	30000000	
5	5	7	11/4/2008 0:35	11/4/2008 1:35	2000	250	900000	250	900000	
	3	7	11/4/2008 1:40	11/4/2008 2:01	2000	250	326500	250	326500	
	3	7	11/4/2008 2:01	11/4/2008 2:14	0	0	0	250	198250	CMG crash
	3	7	11/4/2008 2:14	11/4/2008 2:31	125	15.625	15047	250	240750	Reboot lowered the command rate
	3	7	11/4/2008 2:31	11/4/2008 3:40	2000	250	1034500	250	1034500	
	(sum for 3)						1376047		1800000	
	sum	7					2276047		2700000	
	7	2	11/4/2008 3:40	11/4/2008 4:35	165635	20704.375	68324437	20000	66000000	
6	5	7	11/6/2008 14:00	11/6/2008 15:00	2000	250	900000	250	900000	
	3	7	11/6/2008 15:05	11/6/2008 17:05	2000	250	1800000	250	1800000	
	sum	7					2700000		2700000	
	7	2	11/6/2008 17:10	11/6/2008 18:00	867	108.375	325125	110	330000	
7	5	7	11/11/2008 14:05	11/11/2008 15:05	2000	250	900000	250	900000	
	3	7	11/11/2008 15:10	11/11/2008 17:10	2000	250	1800000	250	1800000	
	sum	7					2700000		2700000	
	7	2	11/11/2008 17:15	11/11/2008 18:05	165635	20704.375	62113125	20000	60000000	CMG crash at 17:16:33, restored at 17:20:03, but upload to spacecraft already complete
8	10	6	11/13/2008 20:00	11/13/2008 20:10	9600	1200	720000	1200	720000	
	6	10	11/13/2008 20:00	11/13/2008 20:10	240	30	18000	30	18000	
	3	7	11/14/2008 0:00	11/14/2008 1:44	2000	250	1570500	250	1570500	
	3	7	11/14/2008 1:44	11/14/2008 1:53	2000	0	0	250	133750	CMG crash
	3	7	11/14/2008 1:53	11/14/2008 3:30	2000	250	1445750	250	1445750	
	(sum for 3)						3016250		3150000	
	7	2	11/14/2008 3:35	11/14/2008 4:00	165635	20704.375	31056562	20000	30000000	

Table 8 Experiment Data Delivered

			Reception																		
			Priority 0		Priority 1		Priority 2		Total		% link	% of total reception			Urgency-weighted				Delivery acceleration	Buffered at node 7		
Pass	From node	To node	images	bytes	images	bytes	images	bytes	images	bytes	utilization	Priority 0	Priority 1	Priority 2	total data rec'd		reference volume		ratio	images	bytes	
1	5	7	9	394965	6	274177	0	0	15	669142	74.3%	59%	41%	0%	21	943319	23	1028616	0.92	15	669142	
	3	7	7	378433	10	534661	6	295166	23	1208260	76.7%	31%	44%	24%	51	2628419	35	1857358	1.42	38	1877402	
	7	2	16	773398	16	808838	6	295166	38	1877402	3.0%	41%	43%	16%	72	3571738	58	2885974	1.24	0	0	
2	5	7	12	460832	0	0	0	0	12	460832	51.2%	82%	0%	0%	12	460832	18	708399	0.65	12	460832	
	3	7	7	363611	14	747477	0	0	21	1111088	61.7%	33%	67%	0%	35	1858565	32	1707983	1.09	33	1571920	
	7	2	1	49622	4	214424	0	0	5	264046	81.2%	19%	81%	0%	9	478470	8	405896	1.18	28	1307874	
3	5	7	3	131783	13	532555	0	0	16	664338	73.8%	20%	80%	0%	29	1196893	25	1021232	1.17	44	1972212	
	3	7	0	0	27	1392595			27	1392595	77.4%	0%	100%	0%	54	2785190	42	2140720	1.30	71	3364807	
	7	2	21	906604	50	2458203			71	3364807	5.4%	27%	73%	0%	121	5823010	109	5172438	1.13	0	0	
4	3	7	24	1248241	23	1162511	0	0	47	2410752	76.5%	52%	48%	0%	70	3573263	72	3705849	0.96	47	2410752	
	7	2	24	1248241	23	1162511	0	0	47	2410752	7.8%	52%	48%	0%	70	3573263	72	3705849	0.96	0	0	
End-to-end totals for phase A:			62	2977865	93	4643976	6	295166	161	7917007	71.8%	38%	59%	4%	272	13446481	247	12170156	1.10			
5	5	7	6	240075	9	401544	0	0	15	641619	71.3%	37%	63%	0%	24	1043163	23	986308	1.06	15	641619	
	3	7	2	89154	19	896535	0	0	21	985689	71.6%	9%	91%	0%	40	1882224	32	1515218	1.24	36	1627308	
	7	2	7	289341	28	1298079	0	0	35	1587420	2.3%	18%	82%	0%	63	2885499	54	2440209	1.18	1	39888	(*see below)
6	5	7	13	642593	0	0	0	0	13	642593	71.4%	100%	0%	0%	13	642593	20	987805	0.65	14	682481	
	3	7	1	52006	25	1389367	0	0	26	1441373	80.1%	4%	96%	0%	51	2830740	40	2215703	1.28	40	2123854	
	7	2	1	44867	3	154224	0	0	4	199091	61.2%	23%	77%	0%	7	353315	6	306046	1.15	36	1924763	
7	5	7	9	401722	0	0	0	0	9	401722	44.6%	100%	0%	0%	9	401722	14	617534	0.65	45	2326485	
	3	7	0	0	29	1508058	0	0	29	1508058	83.8%	0%	100%	0%	58	3016116	45	2318212	1.30	74	3834543	
	7	2	13	593379	51	2743201	0	0	64	3336580	5.4%	18%	82%	0%	115	6079781	98	5129047	1.19	10	497963	TTL expirations on 11/6/and 11/11
8	3	7	16	832765	12	541986	0	0	28	1374751	45.6%	61%	39%	0%	40	1916737	43	2113290	0.91	38	1872714	
	7	2	16	832765	12	541986	0	0	28	1374751	4.4%	61%	39%	0%	40	1916737	43	2113290	0.91	10	497963	
End-to-end totals for phase B:			37	1760352	94	4737490	0	0	131	6497842	65.4%	27%	73%	0%	225	11235332	201	9988592	1.12			
			Total end-to-end delivery						292	14414849												

Asserted priority ratios	0	1	2
# bundles	155	148	6
Fraction	50%	48%	2%

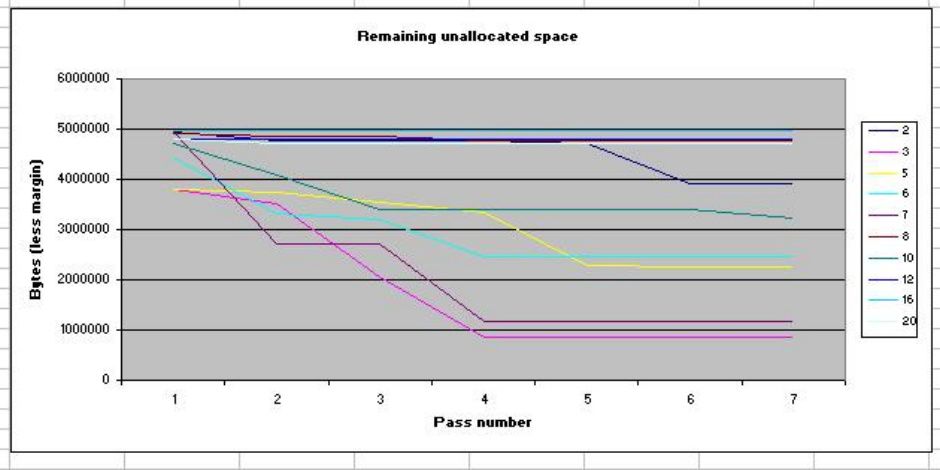
*A bundle of size 39888 was sent to 7 but expired before it could be delivered to 2 (line 6172).

Table 9 Multipath Advantage

Network from 20 to 8				
	Path 1			
		From	To	Capacity
		10	5	n/a
		5	7	5400000
		Path		5400000
	Path 2			
		From	To	Capacity
		10	6	1440000
		6	3	n/a
		3	7	17100000
		Path		1440000
	Network			6840000
	Multipath advantage:			0.27

Table 10. Storage Utilization

	Node									
	2	3	5	6	7	8	10	12	16	20
Data store size	40000000	40000000	40000000	20000000	20000000	20000000	24000000	20000000	10000000	20000000
Allocated traffic storage	24000000	24000000	24000000	12000000	12000000	12000000	14400000	12000000	6000000	12000000
Base unallocated (pre-traffic)	39960808	39960904	39960904	19935400	19999804	19920848	23768432	19797536	9966052	19813040
Base overhead	39192	39096	39096	64600	196	79152	231568	202464	33948	186960
Space utilization floor	35000000	35000000	35000000	15000000	15000000	15000000	19000000	15000000	5000000	15000000
Unassigned space remaining at start of pass:										
Pass										
1	39956792	38788944	38811000	19441208	19955428	19920848	23724672	19797536	9966052	19813040
2	39764548	38513312	38747056	18306952	17713208	19851680	23077088	19797536	9966052	19718784
3	39764548	37057432	38547472	18201648	17713208	19851680	22398328	19797536	9966052	19718784
4	39764548	35847584	38345788	17456304	16185608	19763824	22397288	19797536	9966052	19718784
5	39713716	35847584	37271948	17456304	16185608	19763824	22397288	19797536	9966052	19718784
6	38926452	35846264	37270748	17456304	16185608	19763824	22397288	19797536	9966052	19718784
7	38926452	35846264	37270748	17456304	16185608	19763824	22240472	19797536	9966052	19718784
8	38106696	35791756	37270748	(rebooted)	15820840	(rebooted)	(rebooted)	(rebooted)	(rebooted)	(rebooted)
	2	3	5	6	7	8	10	12	16	20
1	4956792	3788944	3811000	4441208	4955428	4920848	4724672	4797536	4966052	4813040
2	4764548	3513312	3747056	3306952	2713208	4851680	4077088	4797536	4966052	4718784
3	4764548	2057432	3547472	3201648	2713208	4851680	3398328	4797536	4966052	4718784
4	4764548	847584	3345788	2456304	1185608	4763824	3397288	4797536	4966052	4718784
5	4713716	847584	2271948	2456304	1185608	4763824	3397288	4797536	4966052	4718784
6	3926452	846264	2270748	2456304	1185608	4763824	3397288	4797536	4966052	4718784
7	3926452	846264	2270748	2456304	1185608	4763824	3240472	4797536	4966052	4718784
8	3106696	791756	2270748	(rebooted)	820840	(rebooted)	(rebooted)	(rebooted)	(rebooted)	(rebooted)



10. Appendix B – Acronyms

ACE	the person who sends commands to the spacecraft
AMS	Asynchronous Messaging Service (a publish & subscribe protocol that sits on top of ION)
AOS	Advanced Orbiting Systems
ASM	Asynchronous Messaging Service
BER	bit error rate
bi	bidirectional
BP	bundle protocol
BRS	Bundle Relay Service
BSP	bundle security protocol
BVE	Block V Exciter
CBHE	compressed bundle header encoding
CCSDS	Consultative Committee for Space Data Systems
CFDP	CCSDS File Delivery Protocol
CGR	contact graph routing
CL	convergence layer
CLTU	command link transmission unit
CM	Configuration Management
CMD	command
CMG	Command Modulation Generator
CPU	central processing unit
DARPA	Defense Advanced Research Projects Administration
DI	Deep Impact
DIAS	Deep Impact Adaptation Software
DINET	Deep Impact Network Experiment
DIXI	Deep Impact Extended Investigation
DSN	Deep Space Network
DSOT	Data System Operations Team
DSS	Deep Space Station
DTN	Disruption Tolerant Networking (or Delay tolerant Networking; the terms are used interchangeably in the research community)
DTNRG DTN2	Delay/Disruption Tolerant Networking Research Group (Reference Implementation 2)
EID	endpoint ID
EOC	Experiment Operation Center
EMC, EM&C	Experiment Monitoring & Control

EPOCH	Extrasolar Planet Observation and Characterization
EPOXI	EPOXI is a combination of the names for the two extended mission components: the exosolar planet observations, called Extrasolar Planet Observations and Characterization (EPOCH), and the flyby of comet Hartley 2, called the Deep Impact Extended Investigation (DIXI).
EVR	event verification record
FDM	File Delivery Manager
FPDU	CFDP file data protocol data unit
FRR	File Received Report
FSW	flight software
global.ionrc	ION initialization file
GMT	Greenwich mean time
GPS	Global Positioning System
GRC	Glenn Research Center
GUI	graphical user interface
HCD	hardware command decoder
IETF	Internet Engineering Task Force (technical body that standardizes Internet protocols)
ION	(JPL) Interplanetary Overlay Network (software suite on Deep Impact)
IP	internet protocol
ISO	International Organization for Standardization
ISS	International Space Station
JIRA	(bug, issue tracking, and project management system developed by Atlassian Software Systems)
JPL	Jet Propulsion Laboratory
LAN	local area networks
LCD	liquid crystal display
L&G, L/G	load and go (table)
LSI	link service input
LSO	link service output
LTP	Licklider Transmission Protocol
M&C	monitor and control
MER	Mars Exploration Rover
MILSATCOM	Military Satellite Communications
MRO	Mars Reconnaissance Orbiter

MSA	Mission Support Area
NASA	National Aeronautics and Space Administration
NFS	Network File System
nonbi	nonbidirectional
NRT	non-realtime
NTP	network time protocol
ORT	Operational Readiness Test
OSO/SCAN	Office of Space Operations / Space Communications and Navigation
OWLT	one-way light time
PDU	protocol data unit
PKTShow	Packet Show (telemetry viewing system)
POSIX API	Portable Operating System Interface Application Programming Interface
PTL	(JPL) Protocol Technology Lab
PX	CCSDS File Delivery Protocol simulator
pxisi	PX input
pxisi	PX output
RAMS	Remote Asynchronous Message Service (protocol)
RF	radio frequency
RFC	Request for Comments
R/T	realtime
S/C	spacecraft
SCLK/SCET	spacecraft clock/spacecraft event time
SCMF	Spacecraft Command Message File
SCPS-TP	Spacecraft Communication Protocol Standards–Transport Protocol
SCU-B	Spacecraft Control Unit B [there are units A and B]
SFDU	standard formatting data unit
SM	status message
SoA	state of the art
SPICE/NAIF	Space Planetary/satellite ephemeris and constants, Instruments, C Pointing Matrix, Event Info. (Kernels) / Navigation and Ancillary Information Facility
SPOF	(has been changed to Bundle Relay Service (BRS)
stot	simple TDS output tool
SVN	Subversion
TCP	transmission control protocol; CL protocol supported by ION

TCP/IP	transmission control protocol/internet protocol
TDS	Telemetry Delivery System? Fig. 8, P. 15]
TLM	telemetry
TIS	Telemetry Input System
TRL	technology readiness level
TTC	tracking, telemetry, command
TTL	time-to-live (of data segments)
UC	University of California
UCLA	University of California Los Angeles
UK	United Kingdom
UDP	user datagram protocol (supported by ION)
UPA	Uplink Processor Assembly
VOCA	Voice Operational Communications Assembly
WNAN	Wireless Network After Next
XMTR	transmitter