ION IMPLEMENTATION OF THE DTN ARCHITECTURE

Scott Burleigh
Jet Propulsion Laboratory, California Institute of Technology
Scott.C.Burleigh@jpl.nasa.gov

Day 1 Agenda – Afternoon

Key Topics

o **DTN: some technical details**

o **Survey of DTN implementations**

o **ION architecture**

o **ION structure and functions**

o **The core ION modules**

DTN Details

Implementing the DTN architecture

"The architecture embraces the concepts of occasionally-connected networks that may suffer from frequent partitions and that may be comprised of more than one divergent set of protocols or protocol families." (RFC 4838).

So how exactly does this work?

It's not really new

The DTN architecture is much like the architecture of the Internet, except that it is one layer higher in the familiar ISO protocol "stack".

The 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 – in much the same way that IP interconnects "subnets" such as those built on Ethernet, SONET, etc.
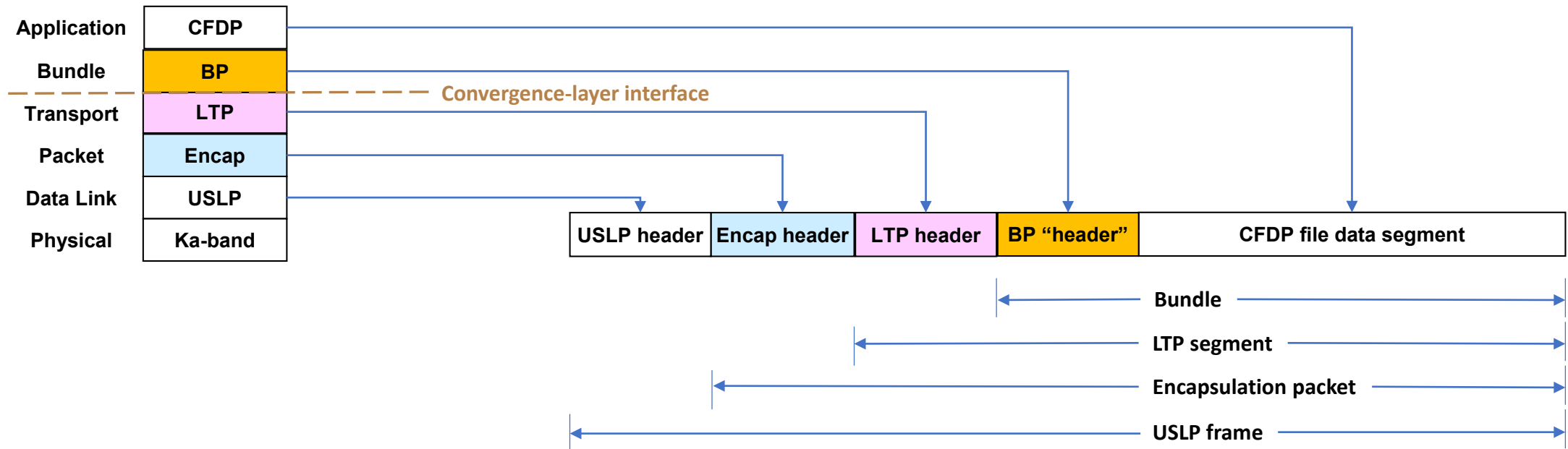
The DTN protocol stack
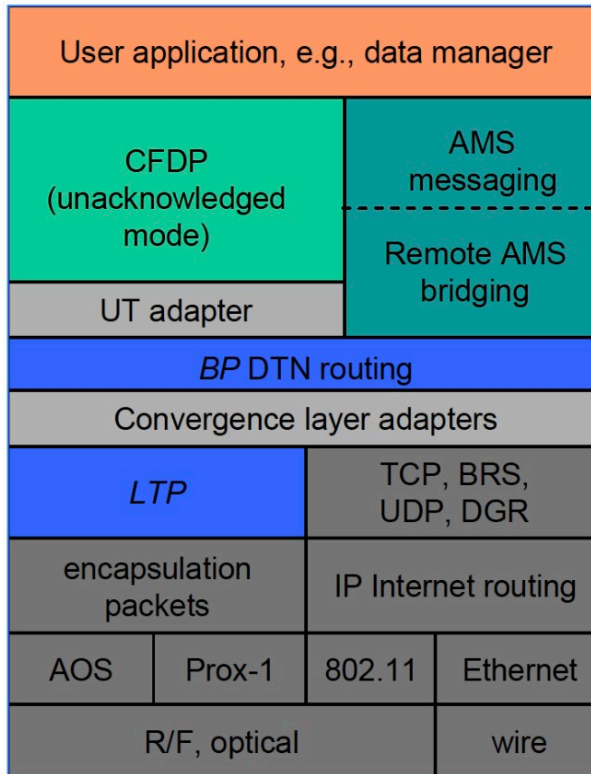
Data traversing a DTN are conveyed in DTN *bundles* – which are functionally analogous to IP packets – between BP *endpoints* which are functionally analogous to sockets.

A single BP *node* may communicate via multiple BP endpoints, just as a single IP node (host or router) may communicate via multiple sockets.
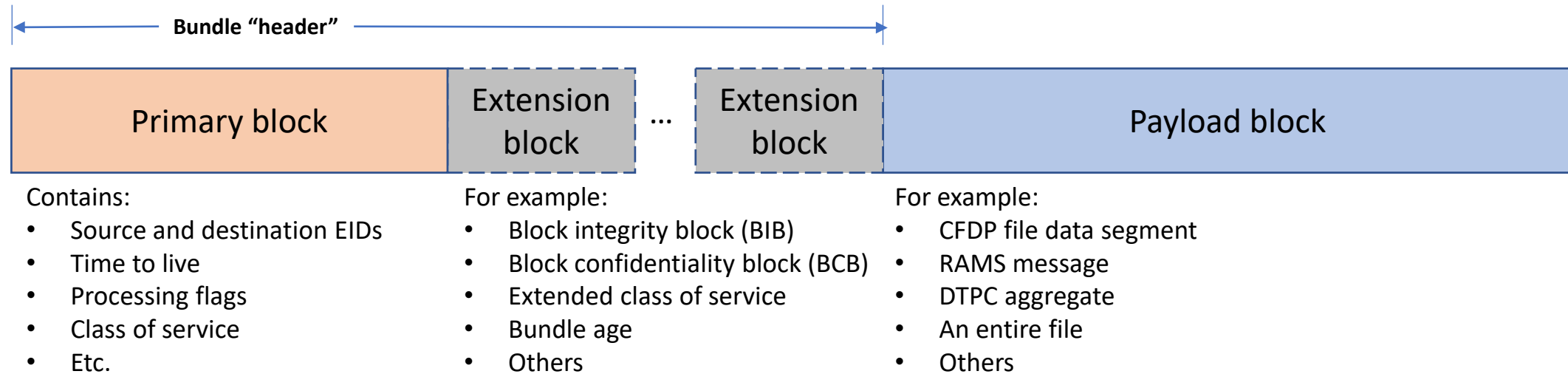
"Convergence-layer" protocols

Just as in the Internet, the stacking of DTN protocols is reflected in the structure of the protocol data units that are transmitted over the network.

Application	CFDP
Bundle	BP
Transport	LTP
Packet	Encap
Data Link	USLP
Physical	Ka-band

Convergence-layer interface

| USLP header | Encap header | LTP header | BP "header" | CFDP file data segment |

Bundle

LTP segment

Encapsulation packet

USLP frame

Structure of a Bundle

Bundle "header"

Primary block	Extension block	...	Extension block	Payload block

Contains:
- Source and destination EIDs
- Time to live
- Processing flags
- Class of service
- Etc.

For example:
- Block integrity block (BIB)
- Block confidentiality block (BCB)
- Extended class of service
- Bundle age
- Others

For example:
- CFDP file data segment
- RAMS message
- DTPC aggregate
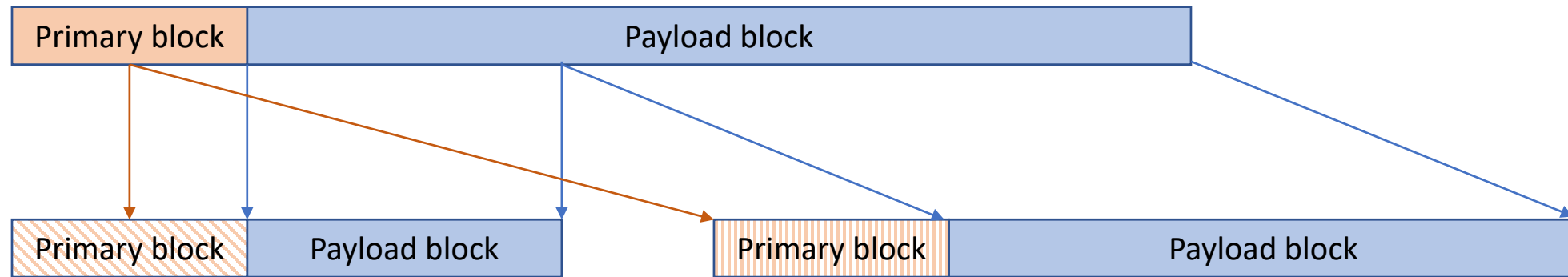- An entire file
- Others

Every bundle is a sequence of *blocks*:

- **Primary block**
- **Zero or more extension blocks**
- **Payload block**

BP is designed for bandwidth efficiency where possible, expansion as necessary. Most fields are variable-length.

Bundle fragmentation

When a bundle is too large to fit into a convergence-layer protocol data unit, the bundle can be *fragmented* into two smaller bundles.

- Only the payload is fragmented. Each fragment is a full-fledged, routable bundle with a primary block that is mostly copied from the original bundle.
- Fragmentary bundles may be further fragmented as necessary.
- Fragments are reassembled at the final destination to reconstitute the original payload, which is then delivered.

Endpoint IDs (1 of 2)

The sources and destinations of bundles are endpoints, identified by **endpoint IDs** that are functionally similar to IP addresses.

But BP endpoint IDs are names, not addresses; they have no topological significance and contain no hints about the route the bundle should take.

Instead they are Uniform Record Identifiers (URIs), i.e., ASCII text strings of the general form:
        scheme_name:scheme_specific_part

For example: **dtn://topquark.caltech.edu/mail** conforms to the "dtn" scheme.

Endpoint IDs (2 of 2)

To reduce transmission overhead, endpoints may instead be named in conformance to the alternative "ipn" scheme:

ipn:*node_number*.*service_number*

Endpoint IDs formed in the "ipn" scheme can be abbreviated to pairs of unsigned binary integers by means of a technique called Compressed Bundle Header Encoding (CBHE).

ipn-scheme BP endpoint IDs (EIDs) look a little bit like Internet addresses (for example, "ipn:3982.64"), but they are still just names (written mostly in numbers rather than letters).

Node numbers identify the flight or ground data system computers on which network software executes, and service numbers are roughly analogous to TCP and UDP port numbers, but node numbers are not topologically aggregated to indicate routes.

Node numbers

💡 **Remember!** CBHE-conformant BP endpoint IDs (EIDs) are functionally and structurally similar to Internet socket addresses. Node numbers are roughly analogous to Internet node numbers (IP addresses), in that they typically identify the flight or ground data system computers on which network software executes, and service numbers are roughly analogous to TCP and UDP port numbers.

In the ION architecture there is a natural one-to-one mapping not only between BP node numbers and BP nodes but also between BP node numbers and:

- LTP engine IDs
- CFDP entity numbers
- AMS continuum numbers

By convention, the same numerical value is usually used for a BP node number, the engine number of the LTP engine used by that BP node, the entity number of the CFDP entity that uses that BP node, and the continuum number of the AMS continuum that uses that BP node.

Security

DTN was designed with security in mind from the very first day. Bundle security is not a separate layer of protocol. It is a mechanism that is built directly into BP itself:

- Every block of a bundle can be signed by a Bundle Integrity Block (BIB) enabling detection of tampering with block data. When there a BIB for the Primary block, which identifies the source node, the authenticity of the bundle can be verified.
- Every block other than the Primary block can be separately encrypted by a Block Confidentiality Block (BCB).
- For defense against traffic analysis, an entire block including Primary block may be the payload of a Bundle In Bundle Encapsulation (BIBE) PDU, and that BIBE's payload can be encrypted.

Licklider Transmission Protocol (LTP)

LTP is a convergence-layer transport protocol designed for use over space links.

LTP divides a data "block" (containing a higher-layer protocol data unit, typically a Bundle) into individual segments for transmission and, much like TCP, detects segment transmission failures, **automatically retransmits lost segments** as necessary, and reassembles segments into the original block at the receiver.

Unlike TCP, LTP transmits segments for multiple blocks **concurrently**. Data sent by LTP will not necessarily arrive in transmission order and there is no suppression of duplicate data arrival.

An LTP block may contain "red" data (for which loss detection and retransmission is required), "green" data (for which there is no loss detection or retransmission), or red data followed by green data in the same block.

LTP provides reliable transmission like BP custody transfer, but the unit of retransmission is the segment rather than the entire bundle; more efficient use of transmission bandwidth.

DTN Core Protocol Implementations

DTN2 (1 of 2)

DTN2, written in C and C++, comprises the reference implementation of the Bundle Protocol (RFC 5050) together with implementations of supporting protocols including the Licklider Transmission Protocol. It is the foundational software for the Networking for Communications Challenged Communities (N4C) platform and project.

DTN2 Platform Applications:

DTNmailex is an Email application for DTN.
HTML requester is a web-caching application for DTN.
PyMail is a nomadic Email Application for the DTN.

DTN2 Platform Tools:

The N4C Integration platform provides an easy and simple way for the DTN user to install and use all the modules of DTN2.

[Download Link](#)

DTN'S Core Protocols Implementation

DTN2 (2 of 2)

DTN2 includes an implementation of PRoPHET, a powerful DTN routing protocol that is highly portable due to being based on the QT cross-platform framework: it runs on Windows, Linux, OSx, Embedded Linux and Symbian platforms.

PRoPHET is included in the DTN2 software distribution along with applications and tools that take advantage of its capabilities.

PRoPHET-enabled Applications:

NotSoInstatMessaging (NSIM): NSIM service can be used for sending/receiving the messages within a DTN network, sending out SMS text messages to a GSM network and sending/receiving Email. It is based on the **Nuntius Leo** source code, a free open-source Email client.

PRoPHET-enabled Tools:

PLogParser: Software tool for analyzing field test traces of PRoPHET routing activity.

Download Link

DTN Core Protocols Implementation

POSTELLATION

Postellation is a DTN implementation that runs on Windows, MacOSX, Linux, BSD, and RTEMS. It is packaged for easy installation and instant registration to the dtnbone for end-users. It implements the Bundle Protocol [RFC5050].

POSTELLATION Applications:

    a. dtnping/dtnpong

    b. dtnsend/dtnrecv

    c. HTTP/HTTPS Proxy

    d. RSS news service delivery, such as NASA news over DTN

Download Link

DTN Core Protocols Implementation

IBR-DTN

This implementation of the bundle protocol (RFC5050) and supporting protocols, written in C++, is designed for embedded systems like the RouterBoard 532A or Ubiquiti RouterStation Pro and can be used as framework for DTN applications.

The module-based architecture with multiple interfaces makes it possible to change functionalities like routing or storage of bundle just by inheriting from the applicable class.

Applications:
- ➢ **dtnsend**
- ➢ **dtnrecv**
- ➢ **dtntrigger**
- ➢ **dtnping**
- ➢ **dtntracepath**
- ➢ **dtninbox**
- ➢ **dtnoutbox**
- ➢ **dtnstream**

IBR-DTN was developed at TU Braunschweig. It supports the TCP and UDP convergence layers, the Bundle Security Protocol, and IPND neighbor discovery specifications.

IBR-DTN aims to be very portable; it is designed to run on embedded systems using **OpenWrt**. It has been successfully tested on x86, MIPS, Raspberry Pi, BeagleBone, and various ARM platforms. Additionally IBR-DTN supports standard x86/x64 Linux distributions (Debian, Ubuntu), OS X, and Android smartphones. Source code and packages for various distributions are available at the **IBR-DTN project page.**

DTN Core Protocols Implementation

JDTN

The JDTN implementation, written in Java by Cisco, contains a DTN 'core' that runs on any platform that supports Java; it also contains a set of UIs for Android. It was developed for mobile platforms, such as Android.

Implemented Protocols:

a. Bundle Protocol (BP) - RFC 5050

b. Licklider Transport Protocol (LTP) - RFC 5326

So why ION?

Flight Environment Constraints (1 of 3)

Link constraints – wireless links enabling interplanetary network communication are generally slow and are usually asymmetric.

- Limited electrical power, relatively small antennae.
    - So signals are weak. This has historically limited transmission from the spacecraft to rates on the order of .25 Mbps to 6 Mbps.
    - Additionally, reception sensitivity is limited. Rates of transmission to the spacecraft have historically been even lower, on the order of 1 or 2 Kbps.
- So the cost per octet of data is on the links is high, and the links are heavily subscribed.
- **Economical use of reception and transmission opportunities is important.**

Flight Environment Constraints (2 of 3)

Processor constraints:

- Limited electrical power, limited mass allowance.
- Intense radiation environment, mandating radiation-hardening, which is time-consuming and expensive.
- Relatively small market, limiting incentive to do radiation-hardening engineering for the latest advances in processor technology.
- So flight processors are always slower than engineering workstations.
- So the cost per processing cycle is high and the processors are heavily subscribed.
- **Economical use of processing resources is important.**

Flight Environment Constraints (3 of 3)

***Operation constraints*:**

- **Hands-on repair is impossible, so reliability is key.**
- **Predictability enhances reliability, so flight software usually must meet hard real-time deadlines. So typically real-time operating systems (RTOS) are used: all software has historically run in "kernel" (rather than "user") mode, without memory protection.**
- **Dynamic allocation of system memory is difficult to predict, so it is typically prohibited except in certain well-understood spacecraft states, e.g., start-up. Software must live within static memory allocations.**

ION is DTN designed for Space Flight

	Terrestrial DTN	**DTN for Space Flight**
Links	Ethernet or WiFi Fast, cheap, symmetrical	Directed, highly attenuated Relatively slow, very expensive, asymmetrical Must use reception/transmission contacts efficiently.
CPU, memory	Commodity generic chips Fast, cheap	Limited-production radiation-hardened chips Relatively slow, very expensive Must use processing resources efficiently.
Resource management	Reboots are easy. Dynamic management of memory is routine.	Hands-on repair is impossible; must minimize risk. Dynamic memory management is unpredictable. Fixed memory allocation is provided at startup.
Operating System	Commercial O/S with memory protection; tasks run in user space.	Real-time O/S, normally no memory protection – all tasks run in kernel space. Must be RTOS-compatible.

ION's ARCHITECTURE

◆ **Design principles**
◆ **Structure**
◆ **Modules**
◆ **Features**

ION's Design

The design of ION addresses all of those constraints:

- **Built-in private dynamic management** of **memory allocated at startup**.
- High-speed **shared direct access** to **built-in object database**.
- **System-wide transaction mechanism**, for safety:
  - Ensures mutual exclusion, preventing lockouts and race conditions.
  - Enables reversal of all database updates made within the current transaction in case of software failure.
- **Compressed bundle headers**, for transmission economy.
- **Zero-copy objects**, for processing and storage economy.
- **Written in C**, for processing economy and small footprint.
- **Portable** among POSIX (and similar) operating systems, including RTOS:
  - Linux, Windows, Android, Solaris, OS/X, RTEMS, VxWorks, FreeBSD

ION's Design Principles (1 of 2)

- ## **Use shared memory.**
    - Often there's no protected memory, so we have no option.
    - But this can be turned to advantage: shared memory is a highly efficient way to pass data between flight software tasks.
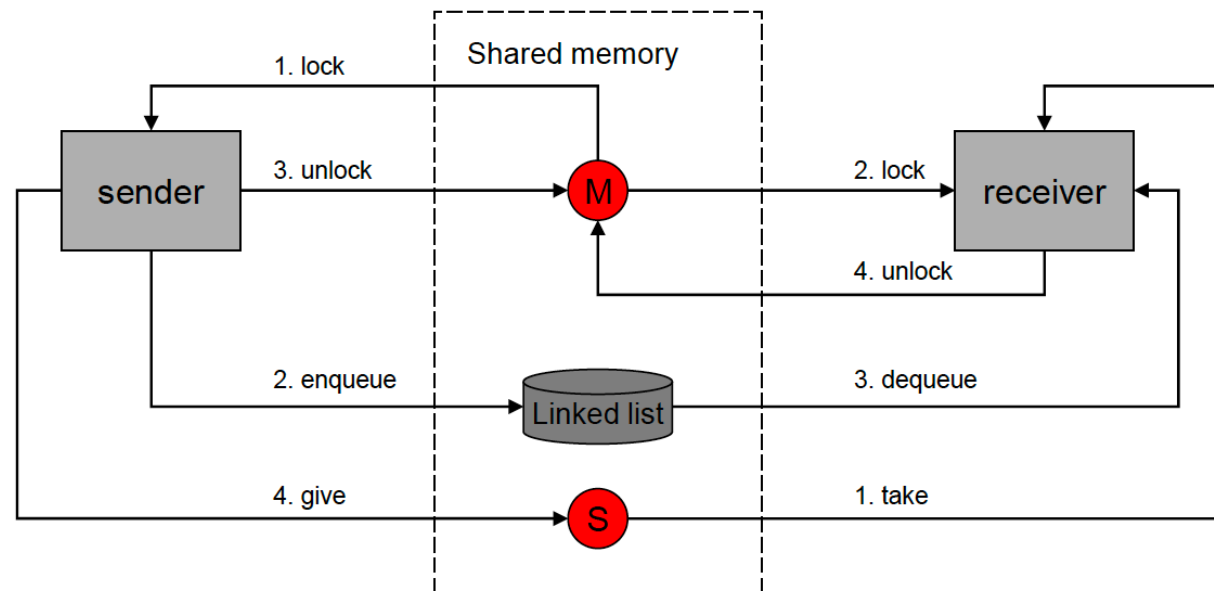
ION's Design Principles (2 of 2)

- **Zero-copy procedures**: leverage shared memory to minimize processing overhead.
  - Encapsulation in layers of protocol overhead (headers and trailers) can be done by reference rather than by copy.
  - The same data object can be shared by multiple tasks, provided reference counting prevents premature deletion.
- **Portability**: this is an unfamiliar programming model, so we must make it easy to develop in an environment with good programming support (e.g., Linux) and then deploy – without change – in the target RTOS environment.

General Design Overview
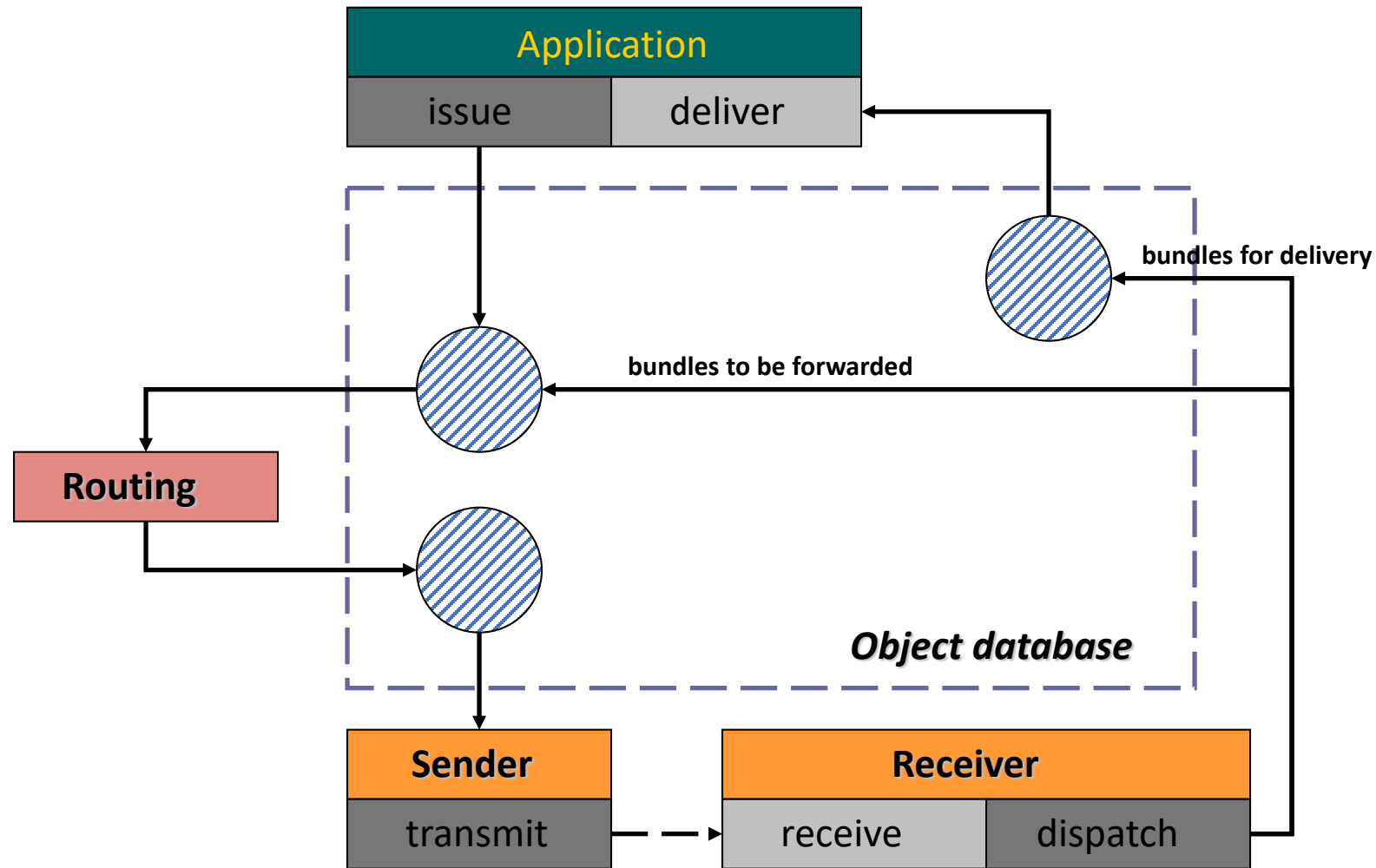
ION Design Goals

o **Reliable conveyance of data over a delay-tolerant network (dtnet)**

o **Built on this capability, reliable data streaming, reliable file delivery, and reliable distribution of short messages to multiple recipients (subscribers) residing in such a network**

o **Inexpensive management of traffic through such a network**

o **Inexpensive facilities for monitoring the performance of the network**

o **Robustness against node failure**

o **Portability across heterogeneous computing platforms**

o **High speed with low overhead**

o **Easy integration with heterogeneous underlying communication infrastructure, ranging from Internet to dedicated spacecraft communication links**

ION Modules Overview (1 of 2)

The ION distribution comprises the following software modules:

o **bp (Bundle Protocol)**, a core DTN protocol that provides delay-tolerant forwarding of data through a network in which continuous end-to-end connectivity is never assured, including support for delay-tolerant dynamic routing.

o **ltp (Licklider Transmission Protocol)**, a core DTN protocol that provides transmission reliability based on delay-tolerant acknowledgments, timeouts, and retransmissions.

o **dgr (Datagram Retransmission)**, an alternative implementation of LTP that is designed for use in the Internet. It enables data to be transmitted via UDP with reliability comparable to that provided by TCP.

o **ici (Interplanetary Communication Infrastructure),** a set of general-purpose libraries providing common functionality to the other modules.

ION Modules Overview (2 of 2)

o **cfdp (CCSDS File Delivery Protocol)**, application-layer service which utilizes underlying DTN protocols for file transfer. CFDP performs the segmentation, transmission, reception, reassembly, and delivery of files in a delay-tolerant manner.

o **ams (Asynchronous Message Service)**, an application-layer service which utilizes underlying DTN protocols for publication of short messages.

o **bss (Bundle Streaming Service)**, a system for efficient data streaming over a delay-tolerant network. It includes (a) a convergence-layer protocol (bssp) that preserves in-order arrival of all data that were never lost en route, yet ensures that all data arrive at the destination eventually, and (b) a library for building delay-tolerant streaming applications.

BP/LTP

The base **Bundle Protocol [RFC5050]** (BP) was developed by the **Internet Research Task Force (IRTF) Delay Tolerant Networking research group** (dtnrg).

The **Consultative Committee for Space Data Systems [CCSDS]** produced profiles of the BP [CCSDS BP] and LTP [LTP for CCSDS] specifications and standardized them for use by CCSDS member agencies.

The profiles are **interoperable with the base specification** and include mechanisms for increased bit efficiency (e.g. limiting the range of some random values in the protocols) as well as protocol extensions such as the Extended Class of Service mechanisms for BP.

The ION BP/LTP code, conforming to those profiles, is available from SourceForge.

Modules: a Closer Look

ION Software Structure
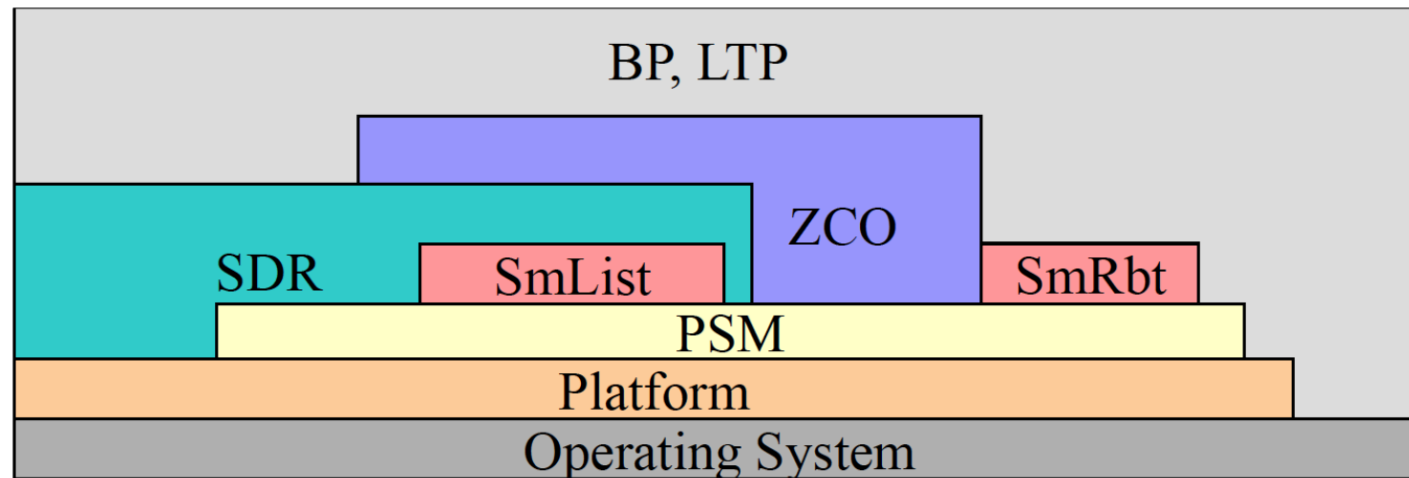
- ○ **BP, LTP Bundle Protocol and Licklider Transmission Protocol libraries and daemons**
- ○ **ZCO Zero-copy objects capability: minimize data copying up and down the stack**
- ○ **SDR Spacecraft Data Recorder: persistent object database in shared memory, using PSM and SmList**
- ○ **SmList linked lists in shared memory using PSM**
- ○ **SmRbt red-black trees in shared memory using PSM**
- ○ **PSM Personal Space Management: memory management within a pre-allocated memory partition**
- ○ **Platform common access to O.S.: shared memory, system time, IPC mechanisms**
- ○ **Operating System POSIX thread spawn/destroy, file system, time**

THE
BUNDLE PROTOCOL
(BP)

Bundle Protocol (BP)

Full implementation of the BP spec as developed by the DTN Research Group. (RFC 5050)

Includes support for:

- Prioritization of data flows
- Bundle reassembly from fragments
- Flexible status reporting
- Custody transfer

Additional features:

- Rate control provides support for congestion forecasting and avoidance.
- Bundle headers are compressed, to reduced protocol overhead and improve link utilization.
- Also includes an implementation of Contact Graph Routing, a system for dynamic routing over interplanetary links.

Bundle Protocol (BP)

BP Compile-time options

Declaring values for the following variables, by setting parameters that are provided to the C compiler (for example, –DION_NOSTATS or –DBRSTERM=60), will alter the functionality of BP (see the ION Deployment Guide).

- **TargetFFS**
- **BRSTERM=*xx***
- **ION_NOSTATS**
- **KEEPALIVE_PERIOD=*xx***
- **ION_BANDWIDTH_RESERVED**
- **ENABLE_BPACS**
- **ENABLE_IMC**

Bundle Protocol (BP)

Configuring BP

Reference files

The BP administration command (bprc) file: configuration of BP on a given ION node.

The IPN scheme administration command (ipnrc) file: configuration of static and default routes for endpoints whose IDs conform to the "ipn" scheme.

The DTN scheme administration command (dtn2rc) file: configuration of static and default routes for endpoints whose IDs conform to the "dtn" scheme as instantiated in the DTN2 reference implementation.

Bundle Protocol (BP)

Testing BP

Test executables

- **bpdriver** is a continuous source of bundles.

- **bpcounter** is a bundle receiver that counts bundles as they arrive .

- **bpecho** is a bundle receiver that sends an "echo" acknowledgment bundle back to bpdriver upon reception of each bundle.

- **bpsource** is a simple console-like application for interactively sending text strings in bundles to a specified DTN endpoint, nominally a bpsink task .

- **bpsink** is a simple console-like application for receiving bundles and printing their contents.

THE
LICKLIDER TRANSMISSION PROTOCOL (LTP)

Licklider Transmission Protocol (LTP)

Full implementation of the LTP spec as developed by the DTN Research Group. (RFC 5326)

Additional features:

- Aggregation of multiple service data units into a single block, to minimize the volume of acknowledgment traffic over highly asymmetric links.

- Implements delay-tolerant, non-conversational flow control by limiting the number of transmission sessions that can be in progress concurrently.

ION ARCHITECTURE
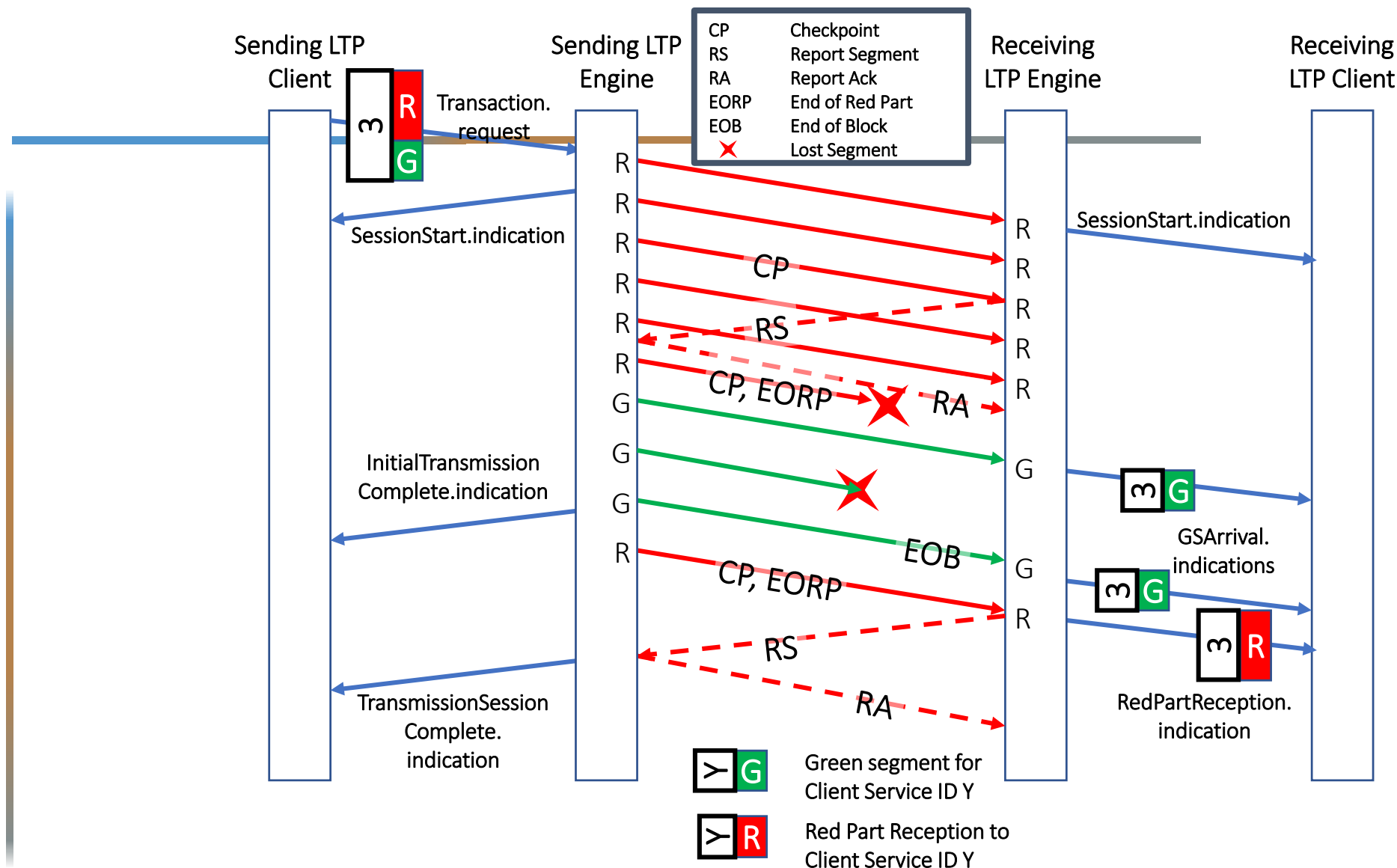
Licklider Transmission Protocol (LTP)

Concurrent LTP Transmissions

- LTP is designed to enable multiple block transmission sessions to be in various stages of completion concurrently, to maximize link utilization. There is no requirement to wait for one session to complete before starting the next one.

- The maximum number of transmission sessions that may be concurrently managed by LTP constitutes a transmission "window" – once the limit is reached, no new transmission can be initiated until an existing one completes or is canceled.

- Allowing too few concurrent sessions could result in an artificial constraint on link utilization, while allowing too many could result in multiple LTP blocks left incomplete at the end of a communications pass. Therefore it is important to configure the aggregation size limits and session count limits of spans during LTP initialization.

Licklider Transmission Protocol (LTP)

Configuring LTP on a deployment platform

The LTP administration command (ltprc) file provides the information needed to configure LTP on a given ION node.

Licklider Transmission Protocol (LTP)

Running LTP on a deployment platform

The executable programs used in operation of the ltp component of ION include:

- The ltpadmin protocol configuration utility → invoked at node startup time and as needed thereafter.
- The ltpclock background daemon → its effects schedule LTP events such as segment retransmissions.
- The ltpmeter block management daemon → segments blocks and effects LTP flow control.
- The udplsi and udplso link service input and output tasks → handle transmission of LTP segments encapsulated in UDP datagrams (mainly for testing purposes).

ltpadmin starts/stops the ltpclock and ltpmeter tasks and, as mandated by configuration, the udplsi and udplso tasks.

Licklider Transmission Protocol (LTP)

Testing LTP on a deployment platform

Test executables

- **ltpdriver** is a continuous source of LTP segments.

- **ltpcounter** is an LTP block receiver that counts blocks as they arrive.
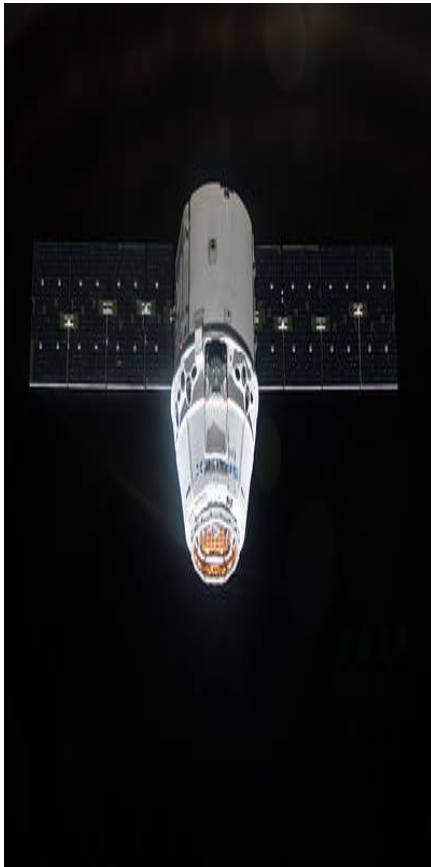
DATAGRAM RETRANSMISSION (DGR)

Datagram Retransmission (DGR)

The **DGR** module is an alternative implementation of LTP that is designed to operate responsibly in the internet.

It is provided primarily as a candidate "primary transfer service" in support of AMS operations in a non-delay-tolerant environment.

The DGR design combines LTP's concept of concurrent transmission transactions with congestion control and timeout interval computation algorithms adapted from TCP.

Datagram Retransmission (DGR)

Testing DGR on a deployment platform

Test executables

- **file2dgr** repeatedly reads a file of text lines and sends copies of those text lines via DGR to **dgr2file**, which writes them to a copy of the original file.

💡 **Remember!** No configuration files or administration utilities are required for the operation of the DGR component of ION.
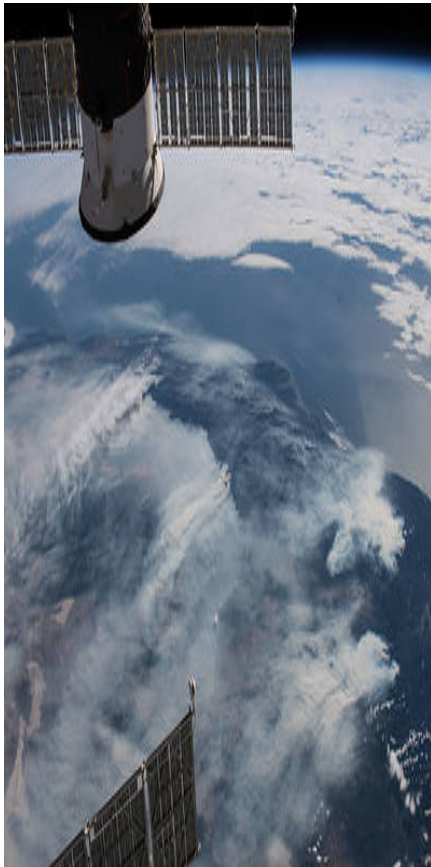
THE INTERPLANETARY COMMUNICATION INFRASTRUCTURE (ICI)

Interplanetary Communication Infrastructure (ICI)

- Insulation of ION software elements from the differences among operating systems.
- Flexible, dynamic private management of a fixed block of pre-allocated system memory.
- Coexistence of multiple memory management instances (e.g., multiple shared-memory partitions).
- Standardized management of linked lists in private and shared memory.
- Flexible, dynamic private management of a fixed block of non-volatile storage, such as battery-backed memory or a pre-allocated file in a flash file system.
- Protocol data encapsulation by reference rather than by copy, plus a reference counting system to enable safe concurrent access to a single non-volatile storage object by multiple tasks.

Interplanetary Communication Infrastructure (ICI)

Core services included in ICI:

a. **Platform:** contains operating-system-sensitive code that enables ICI to present a single programming interface on all platforms.

b. **Personal Space Management (PSM):** performs high-speed dynamic allocation and recovery of variable-size memory objects within an assigned memory block of fixed size.

c. **Memmgr:** enables multiple memory managers – for multiple privately-managed blocks of system memory – to coexist within ION and be concurrently available to ION software elements.

d. **Lyst system:** manages doubly-linked lists in private memory.

e. **Llcv (Linked-List Condition Variables) system:** is an inter-thread communication abstraction that integrates POSIX thread condition variables with doubly-linked lists in private memory.

f. **Smlist:** a doubly-linked list management service which resides in shared DRAM.

g. **SmRbt service:** provides mechanisms for populating and navigating "red/black trees" (RBTs) residing in shared DRAM.

Interplanetary Communication Infrastructure (ICI)

h. **Simple Data Recorder (SDR) system**: manages non-volatile storage.

i. **Sptrace system:** is a diagnostic facility that monitors the performance of the PSM and SDR space management systems.

j. **Zco (zero-copy objects) system**: leverages the SDR system's storage flexibility and implements a reference counting system.

k. **Ionsec** (ION security) system: manages information that supports the implementation of security mechanisms in the other packages.

Interplanetary Communication Infrastructure (ICI)

Configuring ICI on a deployment platform

Files used to provide the information to perform global configuration of the ION protocol stack:

- the ION system configuration (ionconfig) file
- the ION administration command (ionrc) file
- the ION security configuration (ionsecrc) file

The instantiation of ION on a given computer establishes a single ION node on that computer, for which hard-coded values of wmKey and sdrName (see ionconfig(5)) are used in common by all executables to assure that all elements of the system operate within the same state space.

Interplanetary Communication Infrastructure (ICI)

Running ICI on a deployment platform

The executable programs used in operation of the ici component of ION include:

- The **ionadmin** system configuration utility and **ionsecadmin** security configuration utility, invoked at node startup time and as needed thereafter.

- The **rfxclock** background daemon, which effects scheduled network configuration events.

- The **sdrmend** system repair utility, invoked as needed.

- The **sdrwatch** and **psmwatch** utilities for resource utilization monitoring, invoked as needed.

Interplanetary Communication Infrastructure (ICI)

Testing ICI on a deployment platform

Test executables are provided to support testing and debugging of the ICI component of ION

- The **file2sdr** and **sdr2file** programs exercise the SDR system.

- The **psmshell** program exercises the PSM system.

- The **file2sm**, **sm2file**, and **smlistsh** programs exercise the shared-memory linked list system.

ION IN-DEPTH KNOWLEDGE
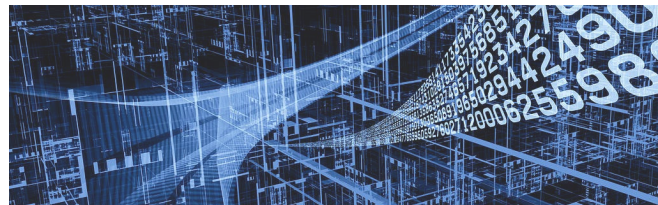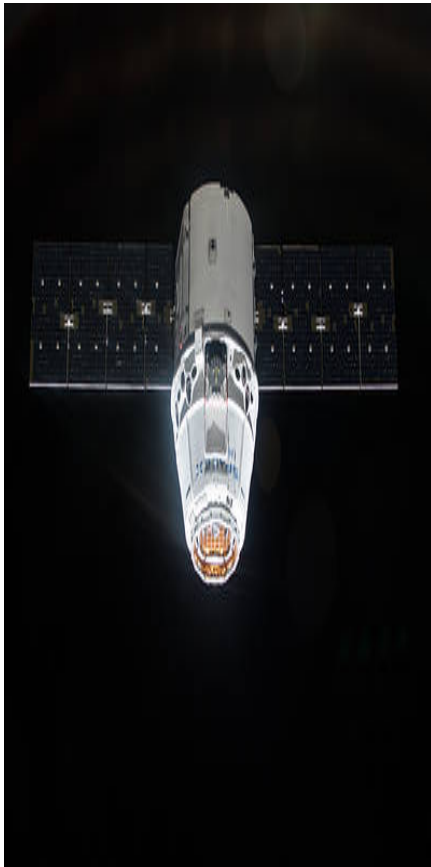
The DTN DevKit

Dev Kit Overview

NASA's DTN development kit includes a **configuration tool** written by NASA's Jet Propulsion Laboratory and a **set of mission-relevant scenarios** using the Common Open Research Emulator (CORE).

CORE provides a set of management functions on top of Windows and Linux Virtual Containers that house the various ION node implementations (including configurations). The Linux virtual machine that is part of the package contains the CORE emulation environment, the ION BP implementation, and the DTN scenarios.
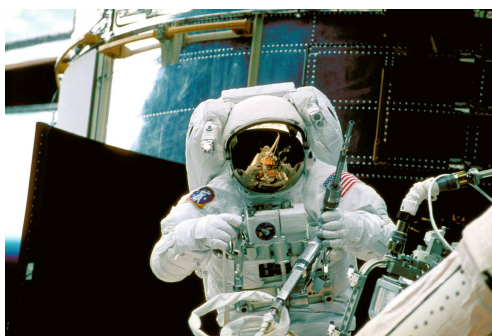
BP and CORE Relationship

In ION scheduled connectivity is controlled via the **contact plan** which controls any links that use the **Licklider Transmission Protocol (LTP) convergence layer** via the contact plan.

Mobility scripts in CORE can automatically move nodes according to predefined paths, and that movement may bring them into and out of contact with other nodes.
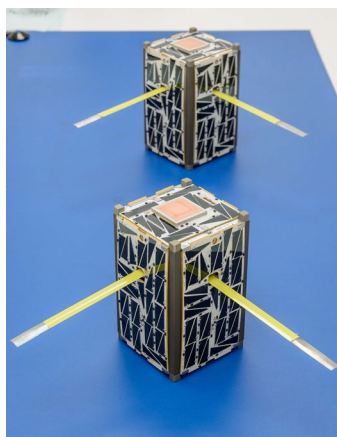
Remember! There is NO INTRINSIC RELATIONSHIP between CORE's notion of connectivity and that in the ION contact plans. For scenarios that include changing connectivity, the mobility of the CORE nodes is orchestrated to be synchronous with the ION contact plans.

Use Case: BP applications

Running executable programs as part of BP with data convergence from multiple sources:

All scenarios included with the development kit start ION processes on each of the nodes, and run a **bpecho** server on BP service ID 1 of each node, and a **bprecvfile** server on BP service ID 2 of each node.

Any node should respond to a bping command of the form:

bping ipn:src_node_id.unused_service_ID ipn:dst_node_id.unused_serviceID

e.g. to ping from n2 to n3:

bping ipn:2.5 ipn:3.1

Send a file to any node using:

bpsendfile ipn:src_node_id.unused_service_ID ipn:dst_node_id.unused_serviceID FILENAME

e.g. to send the file README from n2 to n3:
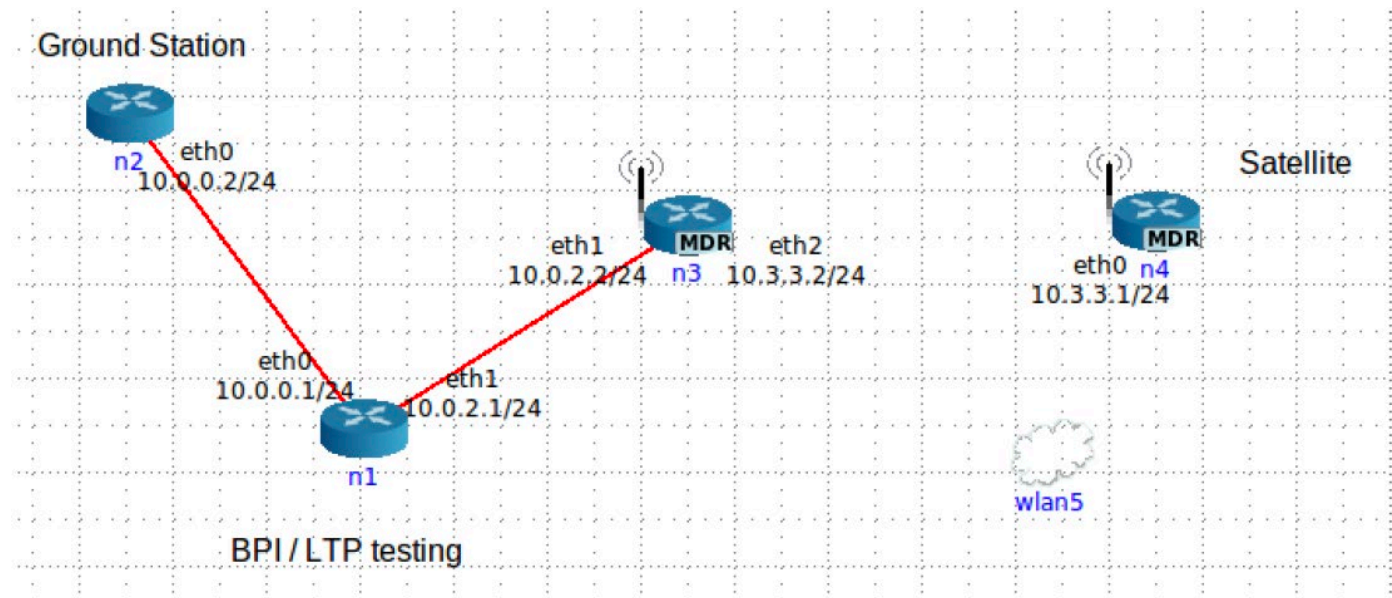
bpsendfile ipn:2.5 ipn:3.2 README

*Use Case: BP applications

The Base Scenario (as a linear topology)

It will automatically start the mobility script which moves the satellite into and out of range periodically, and will launch a bping command from node 2 to node 4.

The ION contact plan is synchronized with the mobility.

Pings will be received whenever the satellite is connected to the fixed network. When the satellite is disconnected, bundles are queued for later delivery.
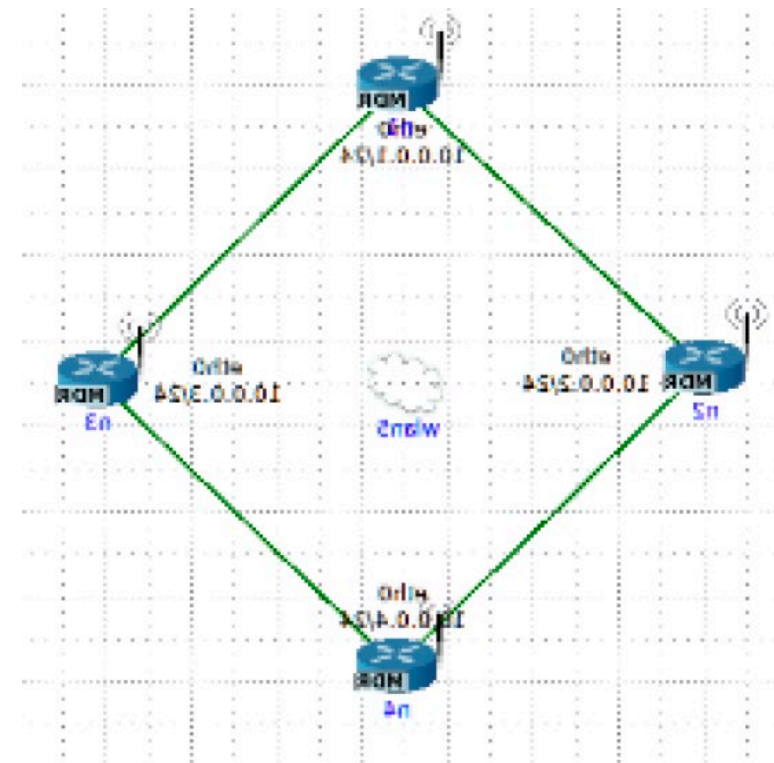
Ground Station

n2   eth0
  10.0.0.2/24

eth1   eth2
10.0.2.2/24   n3   10.3.3.2/24

Satellite

eth0 n4
10.3.3.1/24

eth0
10.0.0.1/24   eth1
              10.0.2.1/24

n1

wlan5

BPI / LTP testing

Refer to "DTN development Kit v1.3"

*Use Case: BP applications

The Diamond Scenario (topology with alternating connectivity between nodes and the destination)

It has no mobility and two paths from the source (n1) to the destination (n4), with connectivity alternating between the last links of the left hand and right hand paths every 30s.

n1 is connected to n2 and n3, and n2 and n3 are each intermittently connected to n4 with 50% duty cycles and 180 degrees out of phase.
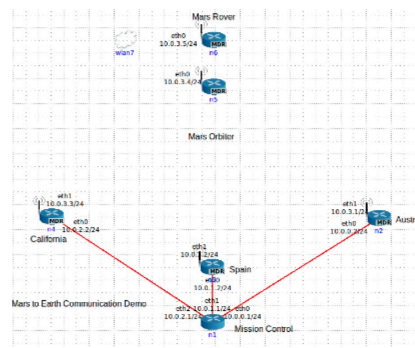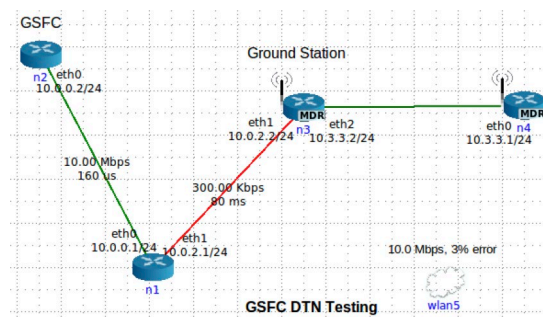

Refer to "*DTN development Kit v1.3*"

*Use Case: BP applications

Other scenarios:

- **GSFC (Automatic BPI demo):** a scenario where a satellite has a high-rate downlink to a ground station, and the ground station has a lower-rate link across the terrestrial network.

- The **"Mars"** scenario: uses an emulated Mars orbiter as a 'data mule' to transfer information from the emulated rover to the mission control center via one of the emulated DSN stations.

- The **"Planet"** scenario: emulates a LEO spacecraft communicating with a number of different ground stations, where all ground stations can receive from the satellite but only two (node n2 and n5) can transmit to the satellite .

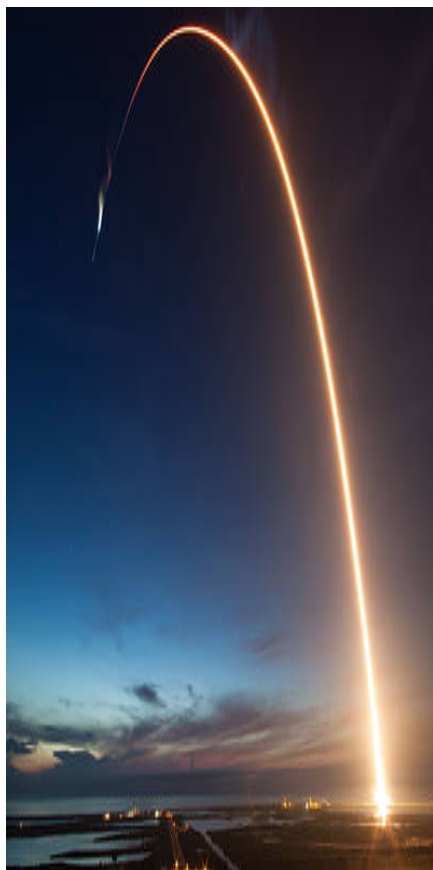Refer to "*DTN development Kit v1.3*"
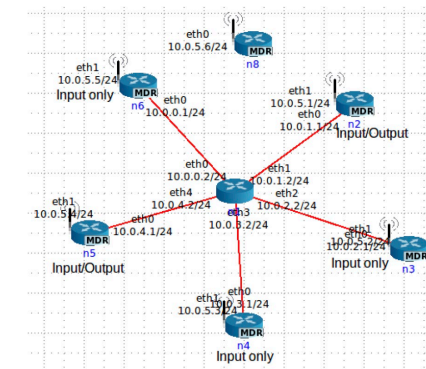
Topics discussed in this session:

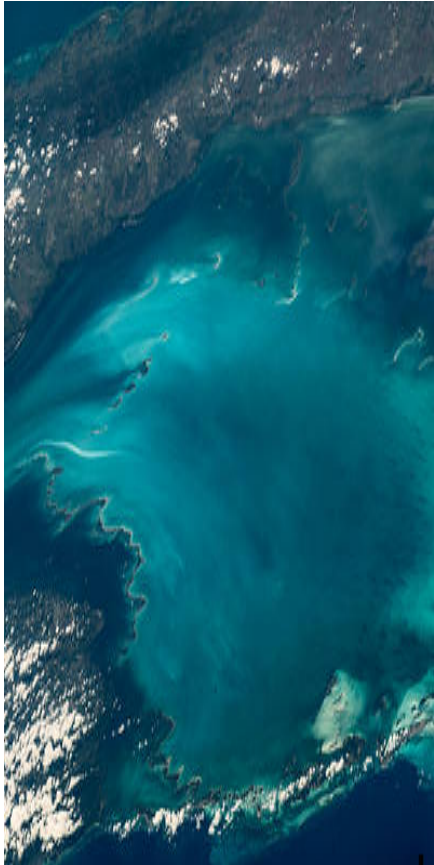| Theory |
| --- |
| Space communication and challenges |
| DTN implementations |
| ION Architecture |
| The Basic ION Structure and Functions |
| ION Modules |
| Introduction to the DevKit |

Any questions?

Thank you!

Goodbye for now!