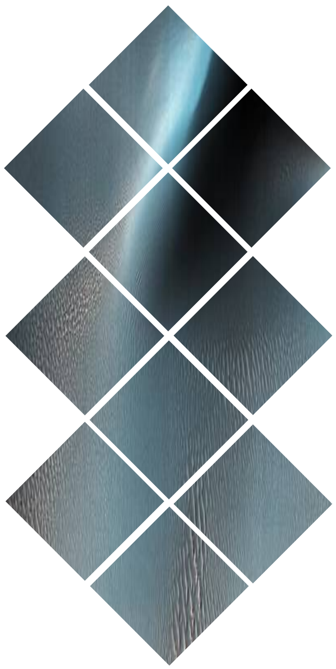
# ION IMPLEMENTATION OF THE DTN ARCHITECTURE

# Day 2 Agenda – Afternoon

### Key Topics

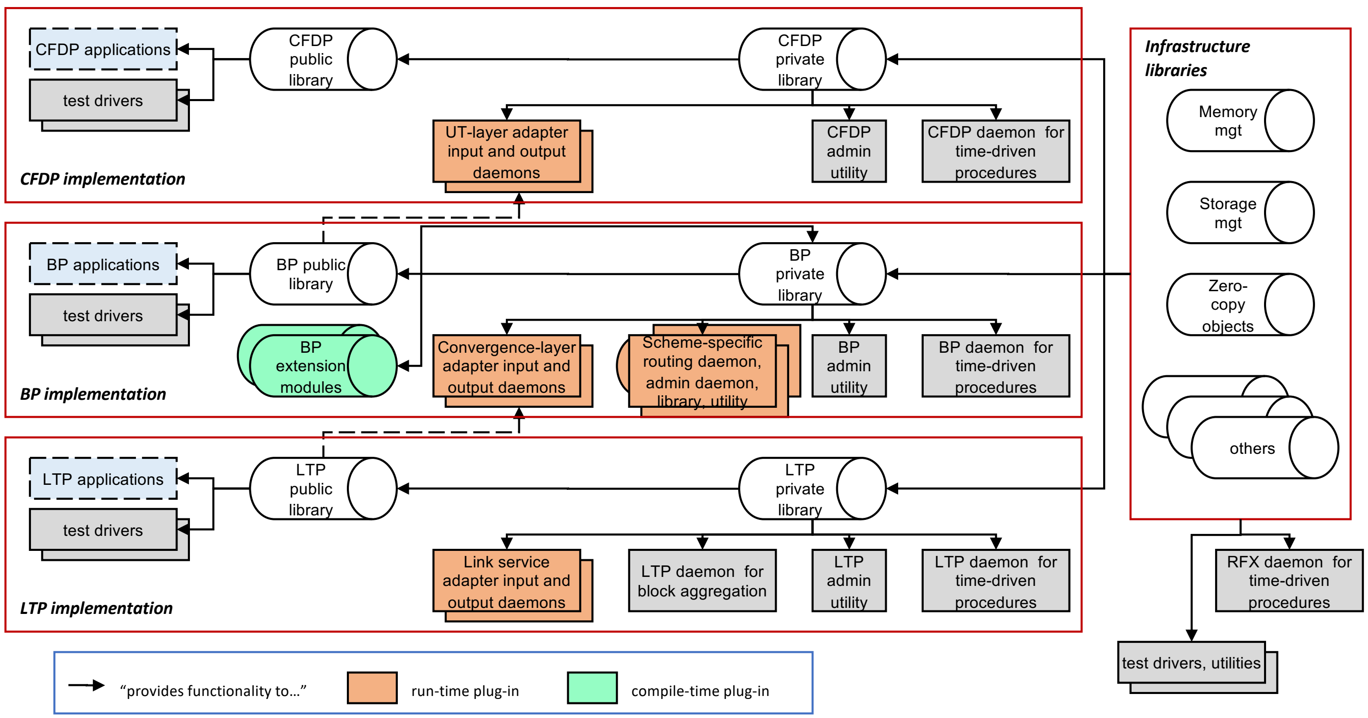
- ION technical details
- Concept of operations



# ION: Under the Hood

- Core components
- The mechanics

# Core Components of ION



# Features of ION

- **Private dynamic management** of **fixed, pre-allocated memory** – to satisfy flight software rules
- High-speed **non-volatile data store** – for processing efficiency
- **Compressed Bundle Header Encoding** – for link efficiency and processing efficiency
- **Managed aggregation** – for transmission efficiency over links with asymmetric data rates
- **Transaction** mechanism – for system safety
- **Zero-copy objects** – for processing efficiency
- **Contact Graph Routing** – for high bandwidth utilization

# ION Resource Management

**Delay-Tolerant Networking** relies on retention of bundle protocol agent state information – including protocol traffic that is awaiting a transmission opportunity – for potentially lengthy intervals.

The nature of that state information will fluctuate rapidly as the protocol agent passes through different phases of operation, so **efficient management of the storage resources allocated to state information** is a key consideration in the design of ION.

General classes of storage resources managed by ION:

- volatile "working memory"
- non-volatile "heap"

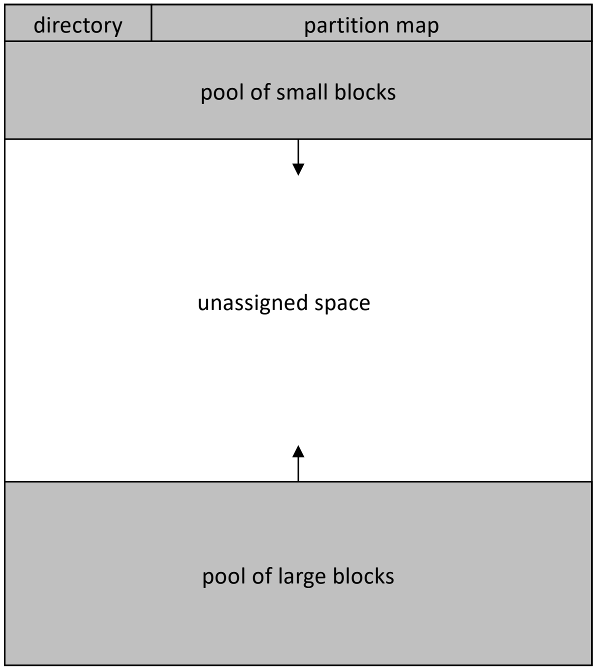
*The Jet Propulsion Laboratory (JPL) installed and tested elements of the DTN on the Deep Impact/ Extrasolar Planet Observation and Characterization (EPOCh) and Deep Impact eXtended Investigation (DIXI) - EPOXI spacecraft. This experiment, called Deep Impact Network Experiment (DINET), transmitted about 300 images from the JPL nodes to the spacecraft and back to the nodes in October 2008. In the following years, DTN has also been deployed on the International Space Station and exercised through a pulsed laser beam from the Moon.*

# Volatile Storage

## ➢ ION's Working Memory

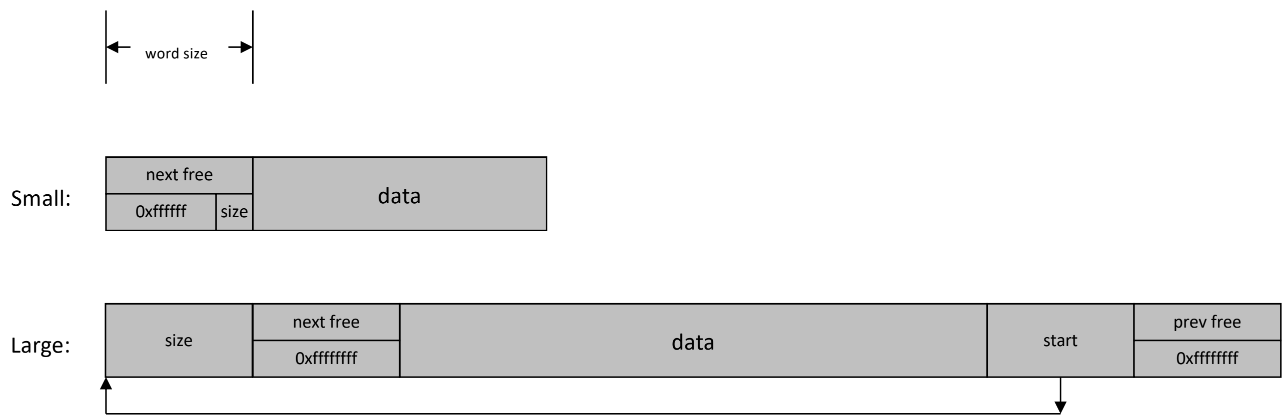
- Represents a fixed-size pool of shared memory (dynamic RAM) that is allocated from RAM at the time the bundle protocol agent commences operation.
- Is used by ION tasks to store temporary data (e.g. linked lists, transient buffers, volatile databases, etc.).
- Intermediate data products and temporary data structures that ought not to be retained in the event of a system power cycle are written to working memory.
- Data structures residing in working memory may be shared among ION tasks/may be created and managed privately by individual ION tasks.
- All of the working memory for any single ION bundle protocol agent is managed as a single Personal Space Management (PSM) "partition". The size of the partition is specified in the wmSize parameter of the ionconfig file supplied at the time ION is initialized.

# Private Dynamic Memory Management




Note: this system has been in continuous use for JPL projects since 2004, on the EO-1 spacecraft for the Autonomous Science Experiment, and on autonomous sea surface and undersea vehicles for the Navy.

# Memory Management Blocks




Trailing overhead of large block enables a newly freed block to be merged with the adjacent free block(s), if any, to minimize fragmentation.

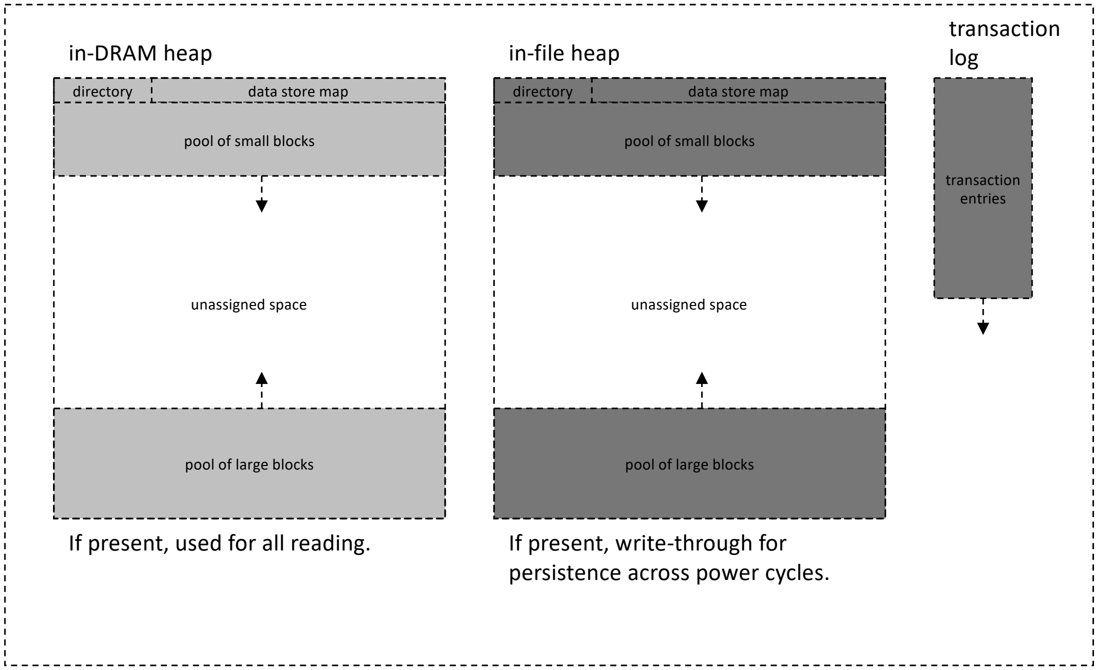
# Non-Volatile Storage

## ➢ ION's Heap

- Represents a fixed-size pool of notionally non-volatile storage that is likewise allocated at the time the bundle protocol agent commences operation.
- The allocated space may occupy a fixed-size pool of shared memory (dynamic RAM → might not be battery-backed), it may occupy only a single fixed-size file in the file system, or it may occupy both.

  In the latter case, all heap data are written both to memory and to the file but are read only from memory → reliable non-volatility of file storage coupled with the high performance of retrieval from dynamic RAM.
- The ION heap is used for storage of data that in some deployments would have to be retained in the event of a system power cycle to ensure the correct continued operation of the node.
- The dynamic allocation of heap space to ION tasks is accomplished by the Simple Data Recorder (SDR) service.
- The total number of bytes of storage space in the heap is computed as the product of the size of a "word" on the deployment platform multiplied by the value of the heapWords parameter of the ionconfig file supplied at the time ION is initialized.

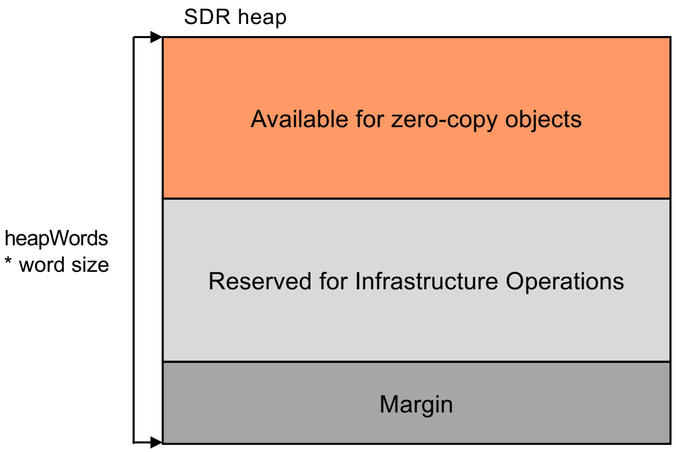
# Non-volatile Data Store (SDR "heap")



# Congestion in ION

## ➢ Heap utilization

- *Space within the ION heap is apportioned as it follows: 20% is normally reserved as margin, 40% is normally reserved as infrastructure operations (both of these percentages are macros that may be overridden at compile time), and 40% is available for storage of protocol state data in the form of "zero-copy objects".*




- *At any given moment, the data encapsulated in a zero-copy object may "belong" to any one of the protocols in the ION stack (AMS, CFDP, BP, LTP), depending on processing state → the available heap space is a single common resource to which all of the protocols share concurrent access.*
- *The heap for an ION node must be large enough to contain the maximum volume of data (e.g. bundles awaiting processing, etc.) that the node will be required to retain during operations.*

# Resource management

### ➢ *Resource exhaustion*

The design of ION is required to prevent resource exhaustion by refusing to enqueue additional data that would cause it.

In ION the affected queuing resources are allocated from notionally non-volatile storage space in the SDR data store and/or file system.

The ION design attempts to prevent potential resource exhaustion by forecasting levels of queuing resource occupancy and reporting on any predicted congestion.

**Remember!** Congestion in a dtnet is the imbalance between data enqueuing and dequeuing rates that results in exhaustion of queuing (storage) resources at a node, preventing continued operation of the protocols at that node.

# Resource management

➢ ***Static vs Dynamic resource allocation***

In general, effective utilization of non-volatile storage can be performed in one of two ways:

1. Static pre-allocation of storage resources (can be less efficient and labor-intensive to configure)
2. Storage resource pooling and automatic, adaptive dynamic allocation

As noted above, ION data management design combines the above approaches:

- 40% of the total SDR data store heap size is statically allocated to the storage of protocol operational state information. This is critical to the operation of ION.
- 20% of the total SDR data store heap size is statically allocated to "margin", a reserve that helps to insulate node management from errors in resource allocation estimates.
- The remainder of the heap and all pre-allocated file system space are allocated to protocol traffic, in the form of zero-copy objects.

# Resource management

➢ ***Congestion detection***

A forecast of a node's estimated maximum resource occupancy is computed by adding to the current occupancy all anticipated net increases and decreases during the horizon for the forecast (this period of time is indefinite unless explicitly declared via ionadmin).

**The ionwarn utility program:**

- Performs congestion forecasting.
- May be run independently at any time.
- Is automatically run by ionadmin immediately before exit.
- Is automatically run by the rfxclock daemon program whenever any of the scheduled reconfiguration events it dispatches result in contact state changes that might alter the congestion forecast.

# Resource management

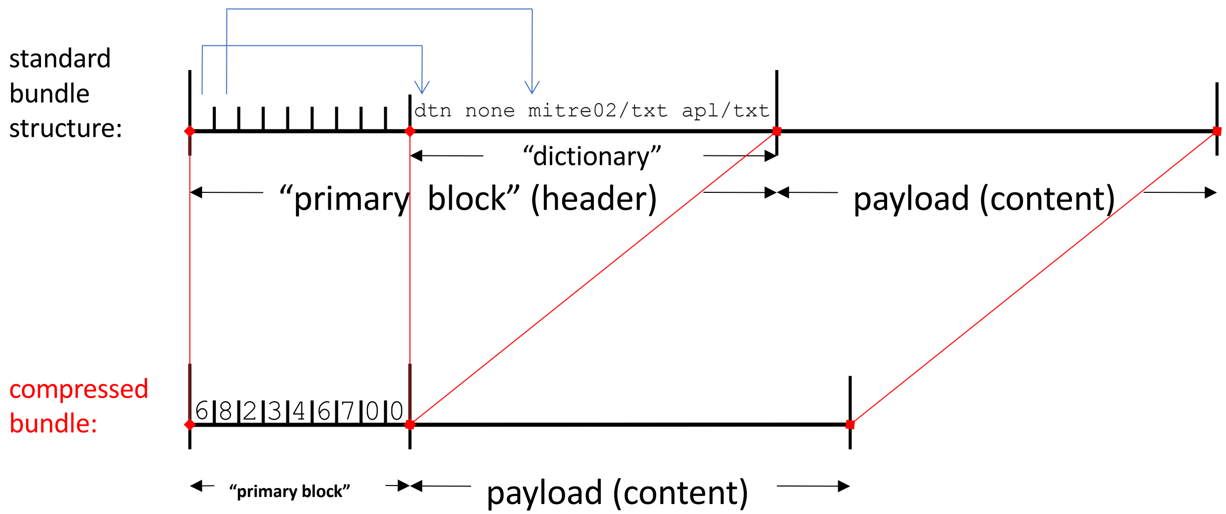
➢ ***Congestion control***

**ION's congestion detection is anticipatory rather than reactive as in the Internet.**

**Anticipatory congestion detection is important because congestion mitigation has to be anticipatory.**

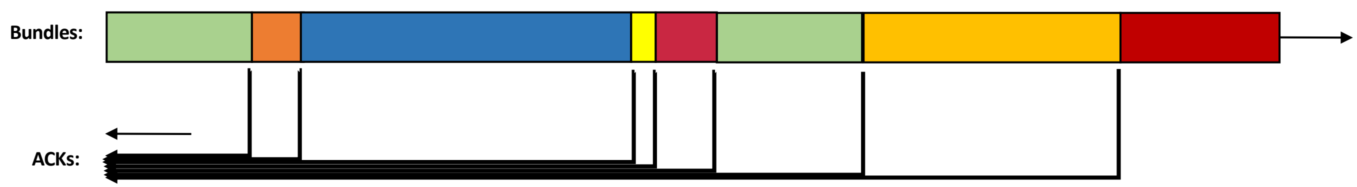
**It is the adjustment of communication contact plans by network management, via the propagation of revised schedules for future contacts.**

# Compressed Bundle Header Encoding

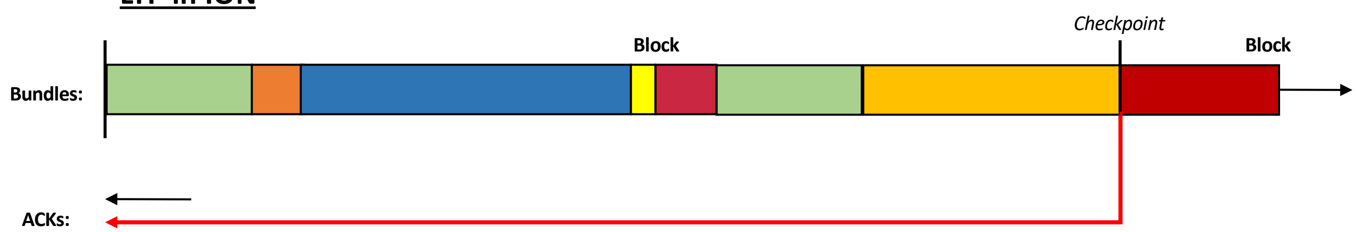


# Managed Aggregation

**BP custody transfer (or CFDP)**

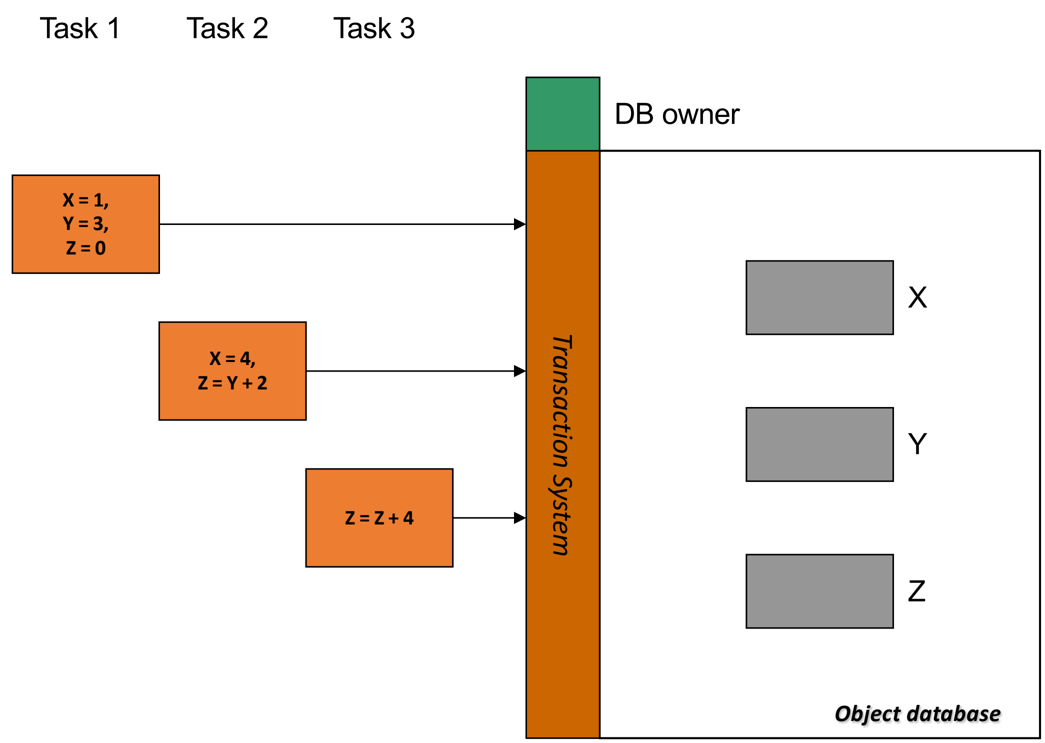



**LTP in ION**

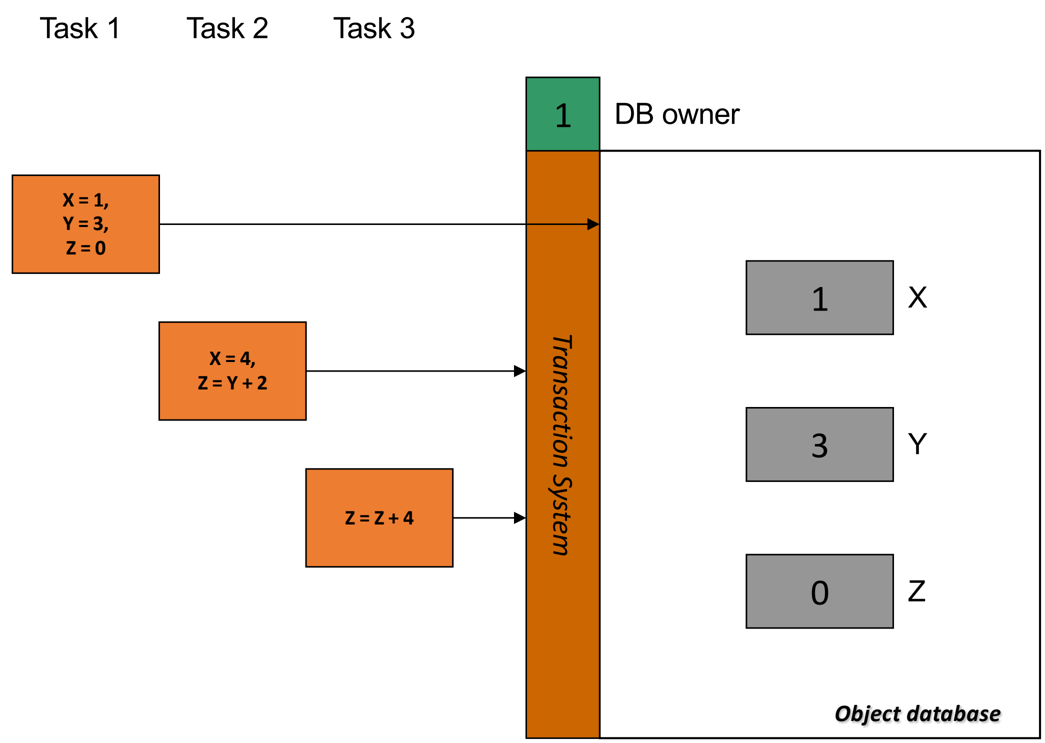



Block size is configurable, so ACK rate can be tuned to the return data rate.

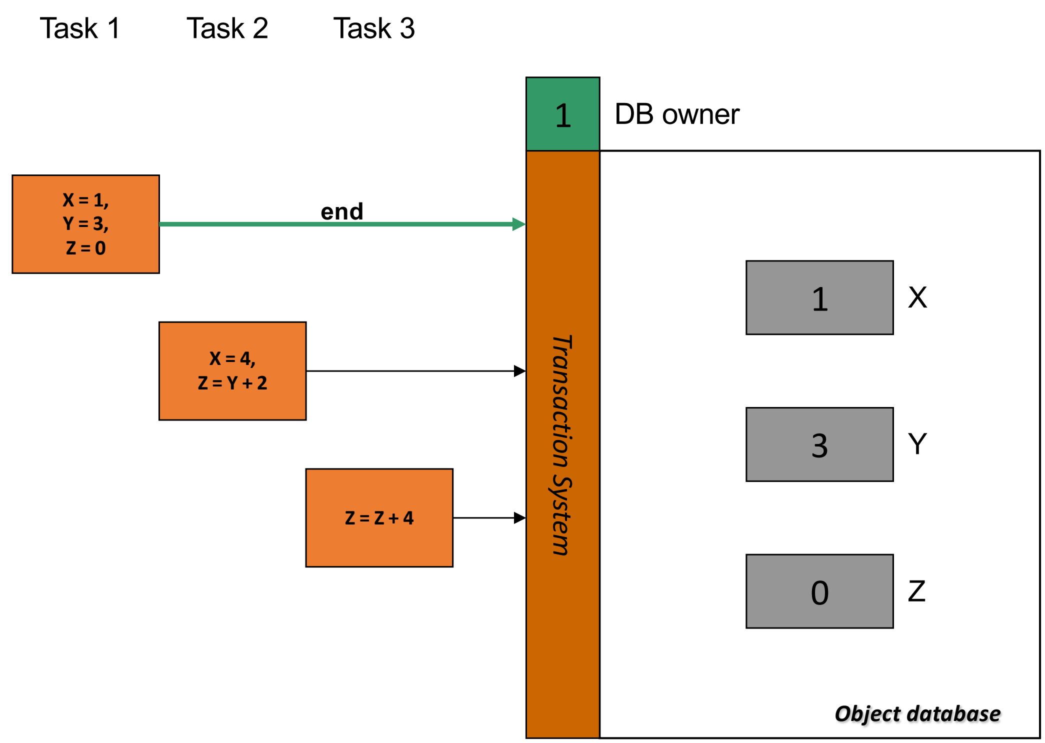
# Transaction system



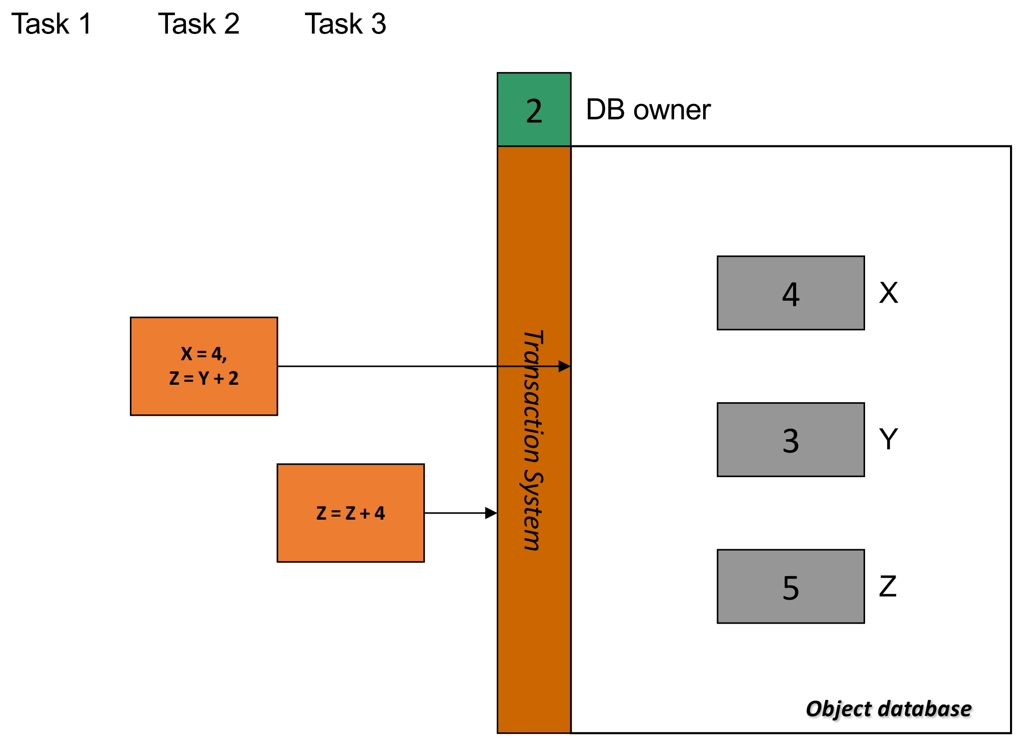
# Transaction system



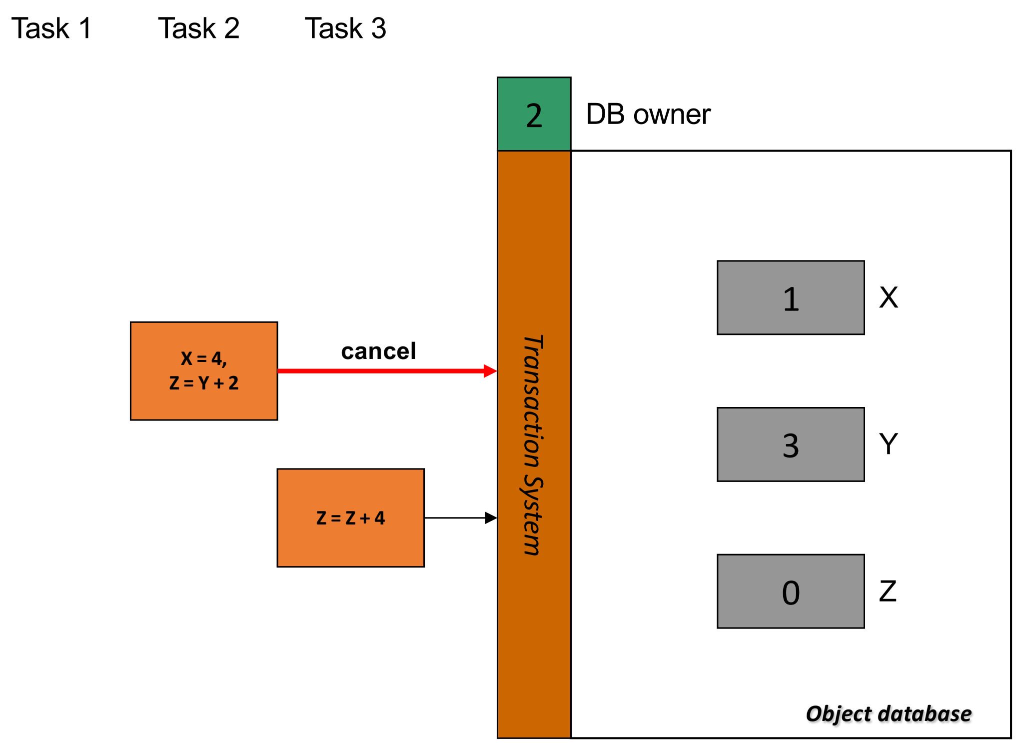
# Transaction system



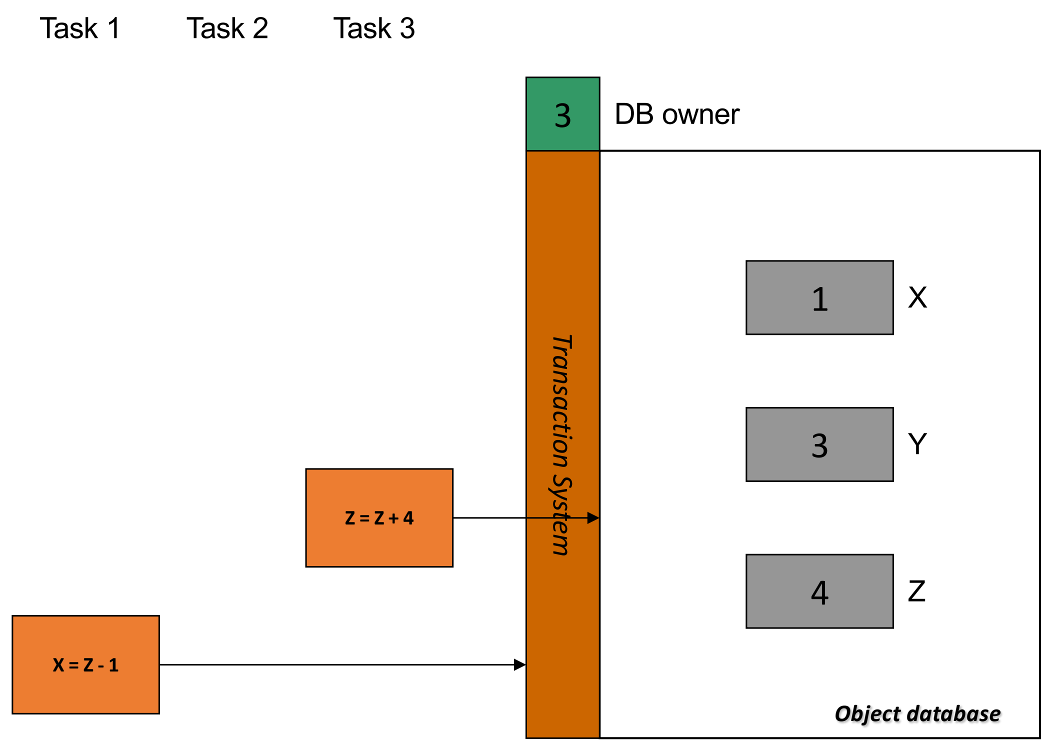
# Transaction system



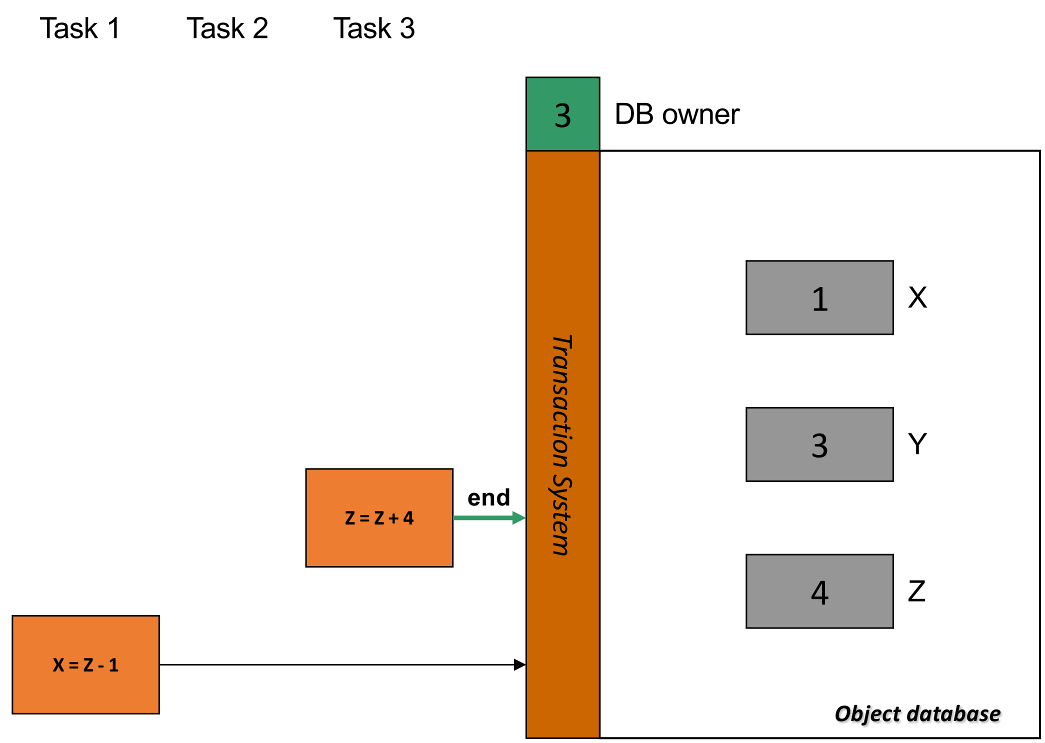
# Transaction system



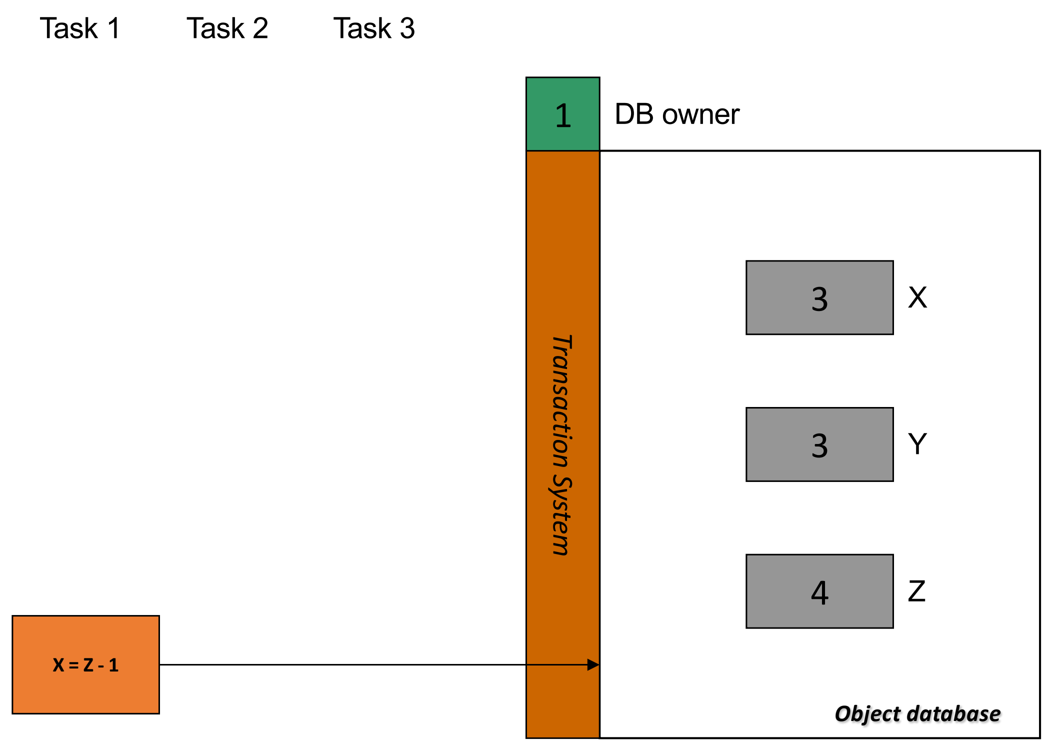
# Transaction system



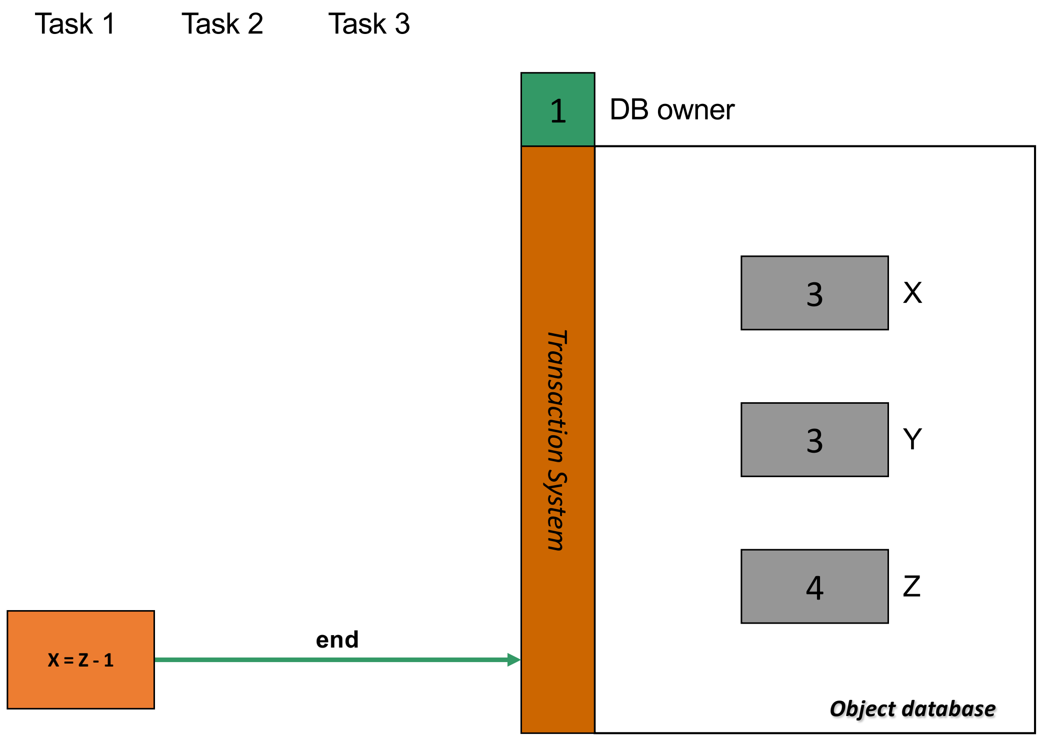
# Transaction system



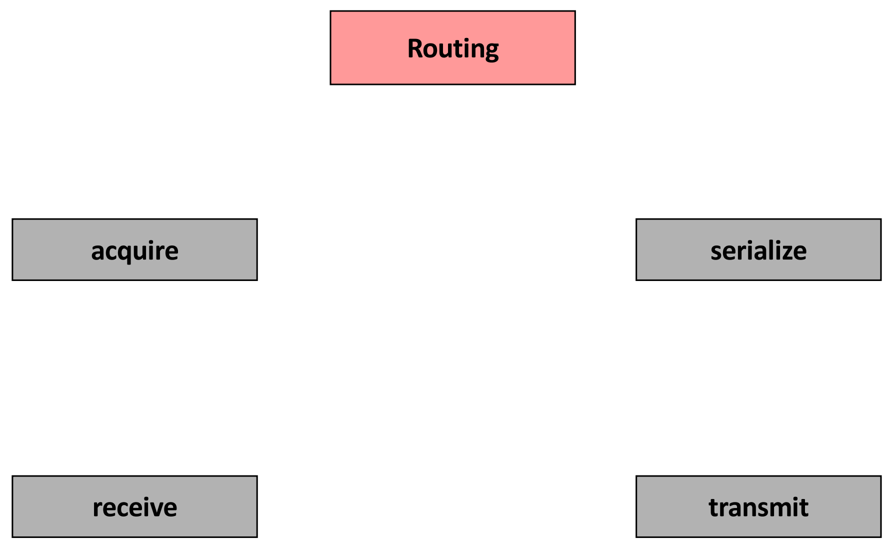
# Transaction system



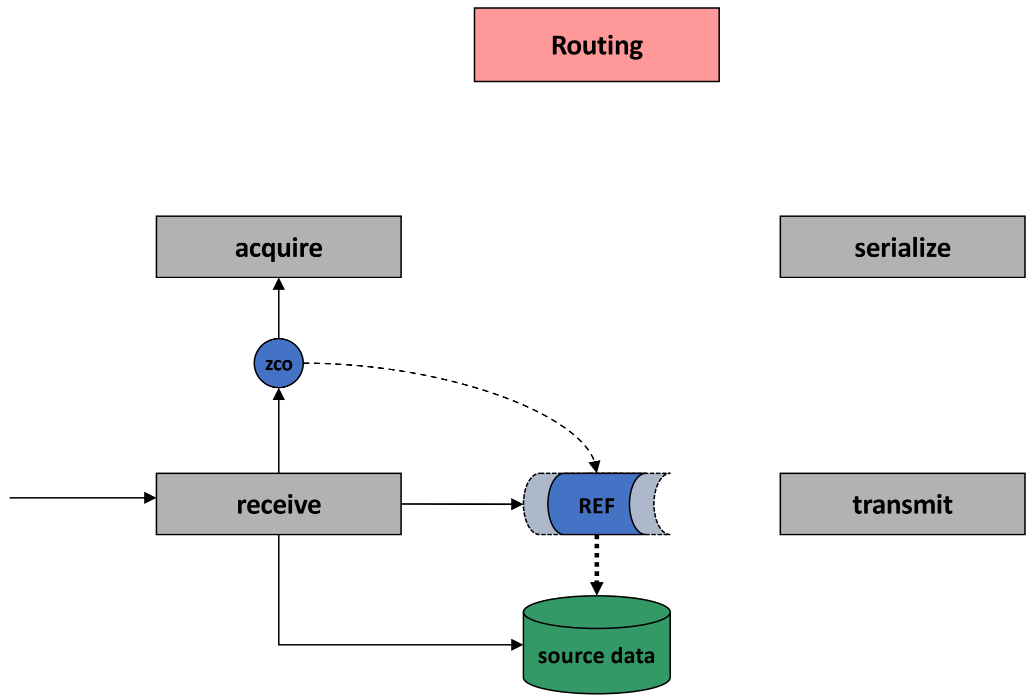
# Transaction system



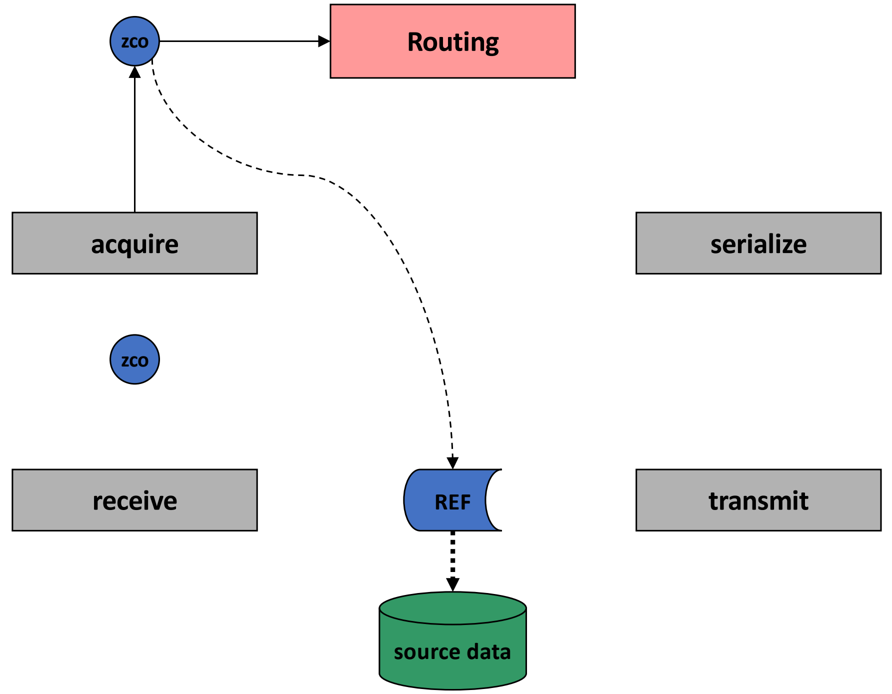
# Zero-Copy Objects



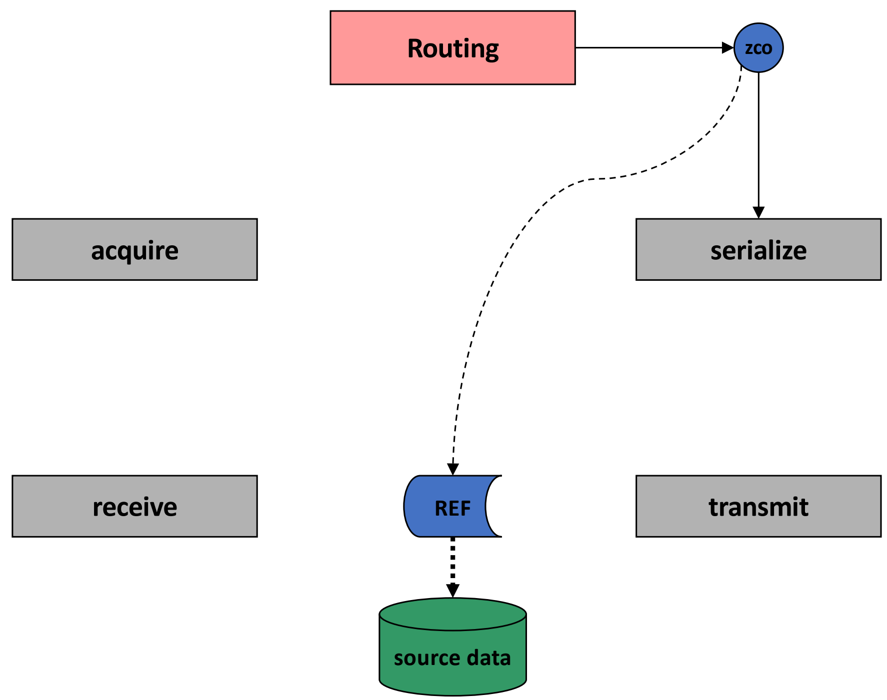
# Zero-Copy Objects



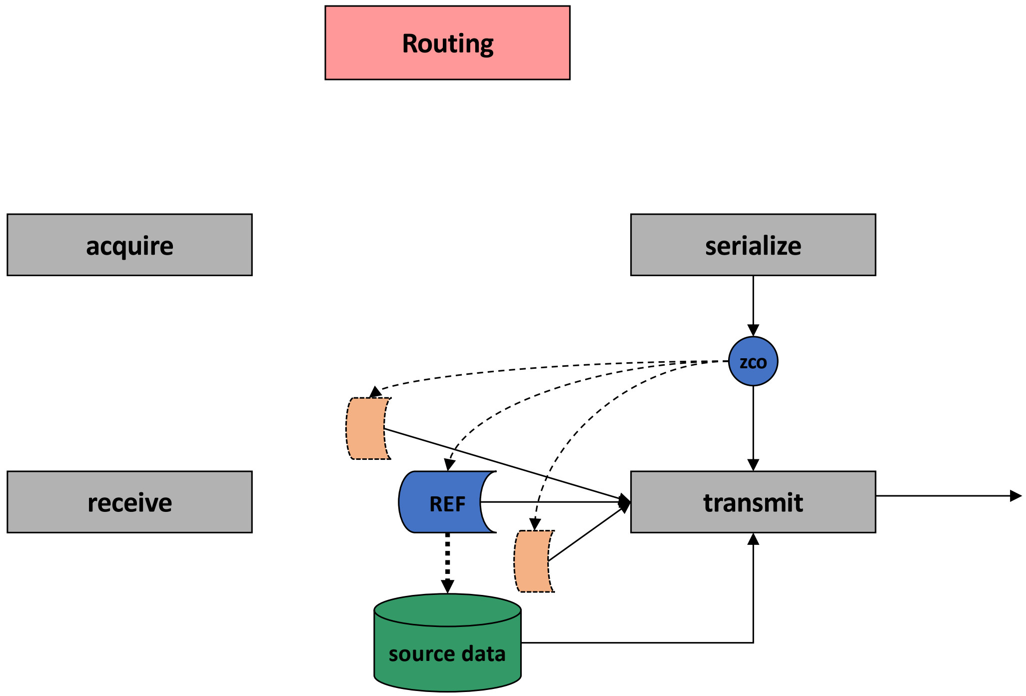
# Zero-Copy Objects



# Zero-Copy Objects



# Zero-Copy Objects



# Contact Graph Routing (CGR)

Contact Graph Routing (CGR) is a dynamic routing system that computes routes through a time-varying topology of scheduled communication contacts in a DTN network.

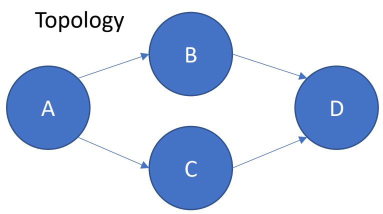
CGR is designed to support operations in a space network based on DTN, but it also could be used in terrestrial applications.

**Remember!** CGR routing procedures respond dynamically to the changes in network topology that the nodes are able know about → e.g. those changes that are subject to mission operations control and are known in advance rather than discovered in real time.

Outcome:

- more accurate routing
- increased total data return
- reduced mission operations cost and risk

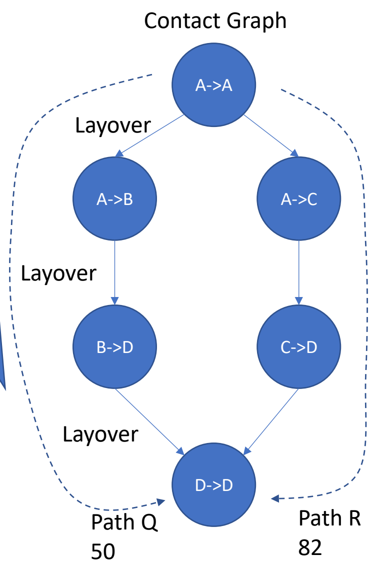
# Contact Graph Routing (CGR)




Contact Plan

| Link | Start | End | Capacity |
|---|---|---|---|
| A -> B | 10 | 20 | 1000 |
| B -> A | 10 | 20 | 1000 |
| B -> D | 40 | 50 | 1000 |
| D -> B | 40 | 50 | 1000 |
| A -> C | 60 | 70 | 1000 |
| C -> A | 60 | 70 | 1000 |
| C -> D | 80 | 90 | 1000 |
| D -> C | 80 | 90 | 1000 |

| Link | OWLT |
|---|---|
| A -> B | 1 |
| B -> D | 10 |
| A -> C | 30 |
| C -> D | 2 |




| Size | TTL | Send ON | Amt Queued for Q |
|---|---|---|---|
| 100 | 30 | --- | |
| 100 | 60 | Q | 100 |
| 100 | 150 | Q | 200 |
| 200 | 60 | Q | 400 |
| 100 | 70 | Q | 500 |
| 300 | 150 | Q | 800 |
| 300 | 150 | R | |

# Key Concepts: Contact Graph Routing (CGR)

| Concept | Definition |
| --- | --- |
| Bundle's time-to-live (TTL) | Represents the length of time after which the bundle is subject to destruction if it has not yet been delivered to its destination. |
| Bundle's expiration time | The time at which a bundle is subject to destruction, computed as its creation time plus its TTL. |
| One way-light time (OWLT) | The distance between the sender and receiver of data, measured in light seconds. Keep in mind that OWLT can change during the time a bundle is en route, because either the sender or receiver (or both) may be space vehicles that are in motion. |
| Contact volume | Represents the product of a contact's data transmission rate (bytes per second) and its duration (stop time minus start time, in seconds). |
| Bundle's estimated volume consumption | A bundle's size is the sum of its payload size and its header size.<br>The total estimated volume consumption (or "EVC") for a bundle is the sum of the sizes of the bundle's payload and header and the estimated convergence-layer protocol overhead. |

# Key Concepts: Contact Graph Routing (CGR)

| Concept | Definition |
|---|---|
| Residual volume (of a given contact) | A contact's residual volume is the sum of the volumes of that contact and all prior scheduled contacts between the local node and the entry node, less the sum of the ECCs of all bundles with priority equal to or higher than the priority of the subject bundle that are currently queued for transmission to that neighbor. |
| Excluded neighbours | A neighboring node C that refuses custody of a bundle destined for some remote node D is termed an excluded neighbor for D. In this case no bundles destined for D will be forwarded to C – except that occasionally (once per lapse of the RTT between the local node and C) a custodial bundle destined for D will be forwarded to C as a "probe bundle". |
| Critical bundles | A bundle which absolutely has to reach its destination. The CGR dynamic route computation algorithm causes each Critical bundle to be inserted into the outbound transmission queues for transmission to all neighboring nodes that can plausibly forward the bundle to its final destination. The bundle is guaranteed to travel over the most successful route and all other plausible routes. This may result in multiple copies of a Critical bundle arriving at the final destination. |

# Contact Graph Routing (CGR)

## Contact plan messages:

CGR relies on contact plan messages that are read from .ionrc files and processed by ionadmin, which retains them in a non-volatile contact plan in the RFX database within the ION SDR heap.

There are two types of contact plan messages:

Contact messages with the following content:

- The starting time of the interval to which the message pertains
- The stop time of this interval
- The Transmitting node number
- The Receiving node number
- The planned rate of transmission from node A to node B over this interval, in bytes per second

Range messages with the following content:

- The starting time of the interval to which the message pertains
- The stop time of this interval
- Node number A
- Node number B
- The anticipated distance between A and B over this interval, in light seconds

# Contact Graph Routing (CGR)

**Remember!** Range messages may be used to declare that the "distance" in light seconds between nodes A and B is different in the B→A direction from the distance in the A→B direction. While direct radio communication between A and B will not be subject to such asymmetry, it's possible for connectivity established using other convergence-layer technologies to take different physical paths in different directions, with different signal propagation delays.

# Contact Graph Routing (CGR)

**Routing Tables:**

Each node uses Range and Contact messages in the contact plan to build a "routing table" data structure.

Routing table:

- Is constructed locally by each node in the network.
- Is a list of route lists, one route list for every other node D in the network that is cited in any Contact or Range in the contact plan.
    - A “route” from node A to node D is a sequence of contacts that will enable a bundle residing at A to be delivered, ultimately, at D.

# Routing Tables: How do they operate?

**Computing lists**

Route lists are computed as they are needed (the maximum number of route lists resident at a given time is the number of nodes that are cited in any Contacts or Ranges in the contact plan).

**Route lists**

Each entry in the route list for node D is a list of routes to D from local node X. The receiving node for the initial contact of a given route is termed the “entry node” for that route.

**Hops**

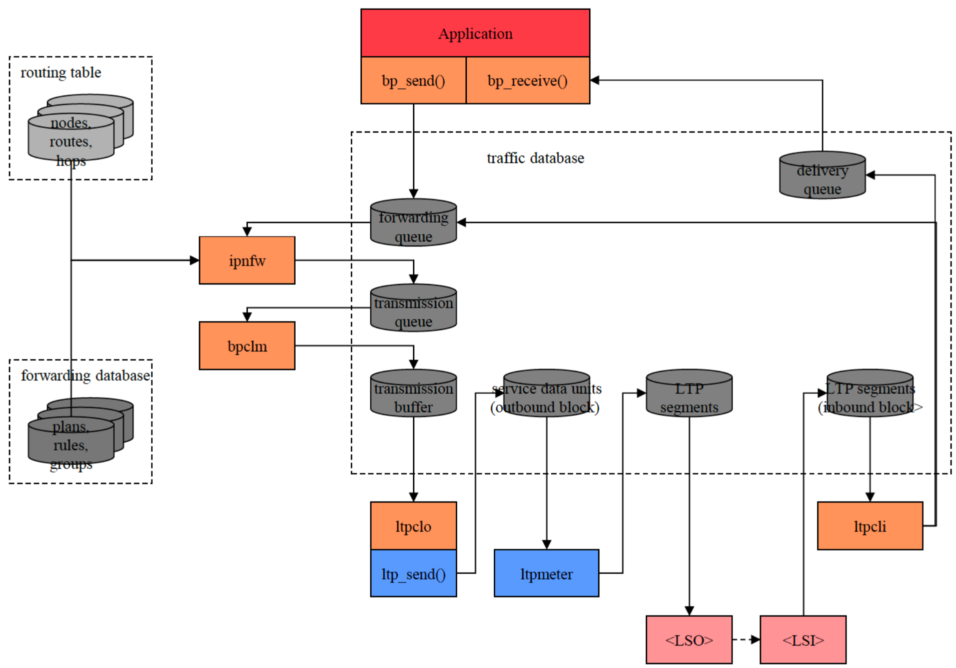
For any given route, the contact from the local node to the entry node constitutes the initial “hop”of the end-to-end path to the destination node. Additionally noted in each route are all of the other contacts that constitute the remaining hops of the route’s end-to-end path.

**Route objects**

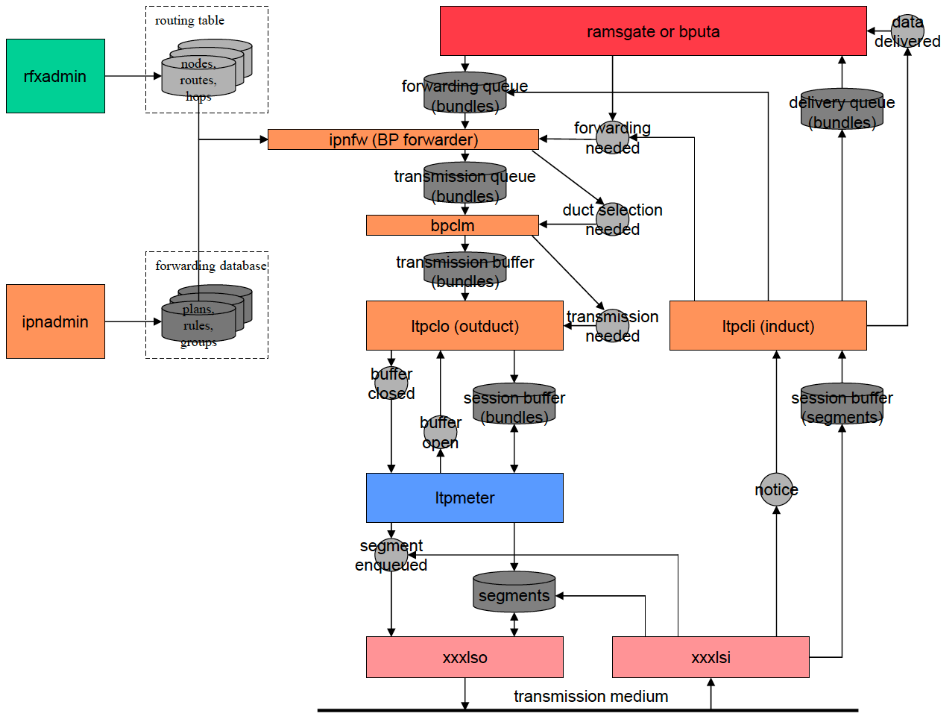
Each route also notes the forwarding cost for a bundle that is forwarded along this route. CGR is configured to deliver bundles as early as possible, so best-case final delivery time is used as the cost of a route. Each route also notes the route’s termination time, the time after which the route is no longer usable because one or more of its contacts have ended.

# Concept of operations

# ION's Main Line of Data Flow



# Core ION Design

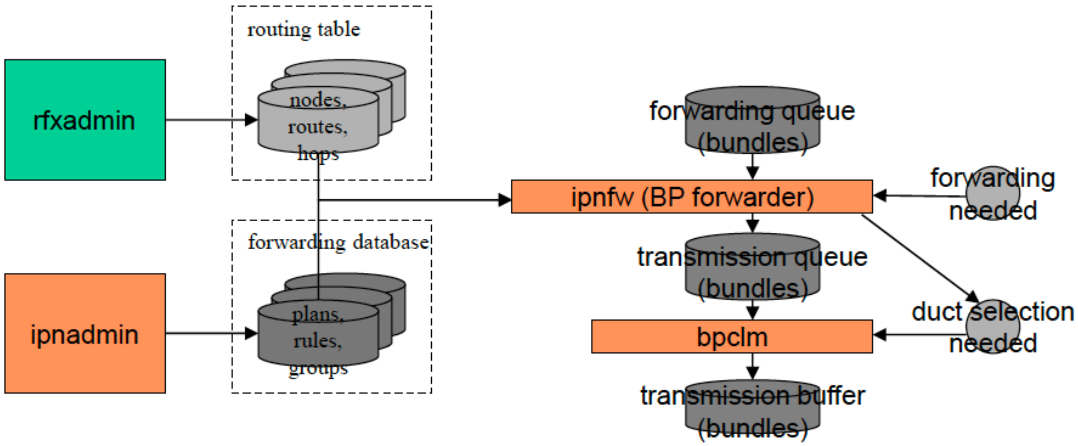



ION node functional overview

*Interplanetary Overlay Network (ION) Design and Operation v. 3.6*

# Bundle Protocol Design

Control of dataflow at the BP level looks like this:




*Interplanetary Overlay Network (ION) Design and Operation v. 3.6*

# Operational features

The concept of operations for ION, based on these modules and mechanisms, encompasses these features:

- ***Fragmentation and Reassembly***
- ***Bandwidth Management***
- ***Contact Plans***
- ***Route Computation***
- ***Delivery Assurance***
- ***Rate Control***
- ***Flow Control***

**Remember!** Fragmentation and reassembly, bandwidth management, and delivery assurance (retransmission) – can potentially be addressed at multiple layers of the protocol stack, in different ways for different reasons.
In stack design it's important to allocate this functionality so that the effects at lower layers complement the effects imposed at higher layers of the stack.

# Operational features

> ***Fragmentation and Reassembly***

ION needs to minimize both head-of-line blocking (which is exacerbated by large PDUs) and transmission overhead (which is exacerbated by small PDUs). The ION protocol stack reconciles these demands at multiple layers of the stack:

- At the application layer (AMS, CFDP, etc.)
- At the bundle layer (BP)
- At the convergence layer (LTP)

# Guidelines for transmission overhead and head-of-line blocking reconciliation:

**At application layer and bundle layer**

- By presenting relatively small application data units (ADUs) – on the order of 64 KB – to BP for encapsulation in bundles, we place an upper bound on head-of-line blocking when bundles are de-queued for transmission.
- Proactive fragmentation of bundles has the same effect.
- This also provides perforations in the data stream at which forwarding can readily be switched from one link (route) to another, enabling partial data delivery at relatively fine, application-appropriate granularity.

**At LTP convergence layer adapter**

- By aggregating these small bundles into blocks for presentation to LTP, we can reduce overhead and minimize report traffic (which is at block granularity).
- Block size threshold places an upper limit on the amount of data that would need to be re-transmitted over a given link at next contact in the event of an unexpected link termination that caused delivery of an entire block to fail.

**At TLP level**

- By segmenting the block we reduce the probability of losing any single PDU and accommodate the Maximum Transfer Unit (MTU) frame size of the underlying link service.

# Operational features

## ➢ *Bandwidth management (1 of 2)*

The allocation of bandwidth to application data is requested by the application task that's passing data to DTN. This is accomplished only at the lowest layer of the stack at which bandwidth allocation decisions can be made and in the context of node policy decisions that have global effect.

The transmission queue interface to a given neighbor in the network is actually three queues of outbound bundles: one queue for each of the defined levels of priority ("class of service") supported by BP.

When an application presents an ADU to BP for encapsulation in a bundle, it indicates its own assessment of the ADU's priority. Upon selection of a proximate forwarding destination node for that bundle, the bundle is appended to whichever of the queues corresponds to the ADU's priority.

# Operational features

## ➢ *Bandwidth management (2 of 2)*

**Default:** the convergence-layer manager (CLM) task servicing a given proximate node extracts bundles in strict priority order from the heads of the three queues → the bundle at the head of the highest-priority non-empty queue is always extracted.

**Preventing starvation:** If the ION_BANDWIDTH_RESERVED compiler option is selected at the time ION is built, the convergence-layer manager task servicing a given proximate node extracts bundles in interleaved fashion from the heads of the node's three queues:

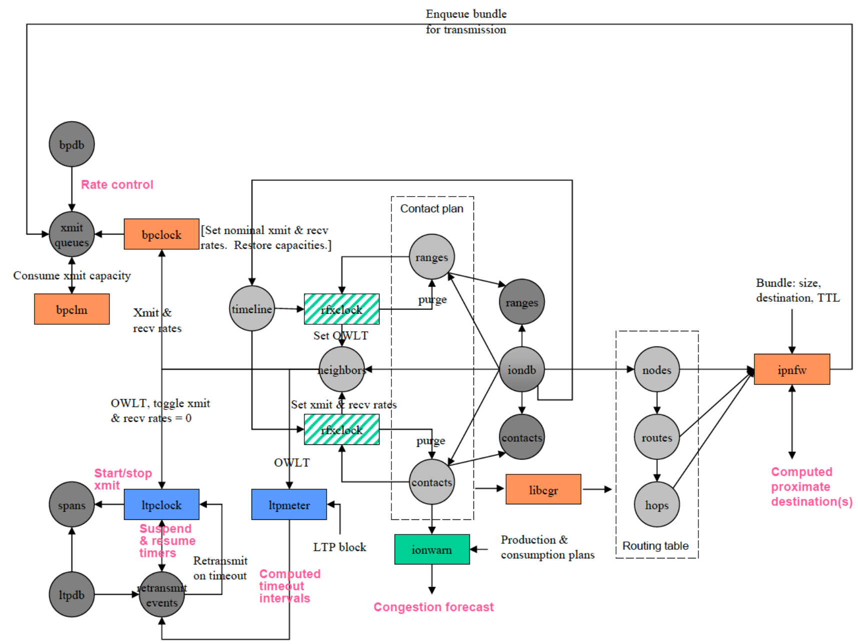
- Whenever the priority-2 queue is non-empty, the bundle at the head of that queue is the next one extracted.
- At all other times, bundles from both the priority-0 -1 queues are extracted, but over a given period of time twice as many bytes of priority-1 bundles will be extracted as bytes of priority-0 bundles.

# Operational features

➢ ***Contact Plan Database***

**Remember:** In a DTN-based network ad-hoc information discovery can take too much time and the information might lose currency and effectiveness.

Protocol operations must be largely driven by information that is pre-placed at the network nodes and tagged with the dates and times at which it becomes effective. This information takes the form of *contact plans* that are managed by the R/F Contacts (rfx) services of ION's ici package.



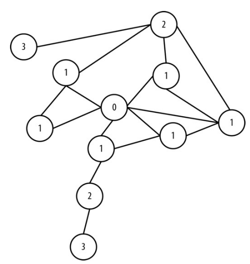

The structure of ION's RFX (contact plan) database

# Operational features

➢ ***Route computation***

The route computed by ION for a bundle with a given destination endpoint may be either **unicast** or **multicast**, depending on the destination endpoint ID.

**Remember!** The result of successful routing is the insertion of the bundle into an outbound transmission queue (selected according to the bundle's priority) for one or more neighboring nodes.



# Operational features

## Unicast

Unicast is the transmission of a bundle whose destination is a single node registered within a known "singleton endpoint".

The destination endpoint ID must be a URI formed in either the "dtn" scheme (e.g.: dtn://bobsmac/mail) or the "ipn" scheme (e.g.: ipn:913.11).

Procedure:

1. Begin unicast route computation by attempting to compute a dynamic route to the bundle's final destination node, by CGR.
2. If no dynamic route can be computed, but the final destination node is a "neighboring" node → taking a direct route to that node is best strategy if the transmission is not blocked by management.
3. Otherwise, look for a static route ("exit"). If the bundle's destination node number is in one of the ranges of node numbers assigned to exits, then forward the bundle to the gateway node for the smallest such range.
4. If you can't determine a dynamic route or a static route for this bundle, the bundle is placed in a "limbo" list for future re-forwarding when transmission to some node is "unblocked".

# Operational features

**Multicast**

The topology of the single network-wide multicast distribution tree is established by management library functions that declare the children and parents of each node. These functions are invoked from the imcadmin utility program.

Each relative of each node in the tree must also be a neighbor in the underlying dtnet.

A bundle whose destination endpoint cites a multicast group, whether locally sourced or received from another node:

- Is delivered immediately if the local node is a member of the indicated endpoint.
- Is queued for direct transmission to every immediate relative in the multicast tree other than the one from which the bundle was received (if any).

NOTE: revisions to multicast procedures are coming in ION version 3.6.3 but are not yet ready to discuss.

# Operational features

## ➢ *Delivery assurance (1 of 2)*

- ION is designed to enable retransmission at multiple layers of the stack in case data delivery fails, depending on the preference of the end user application
- At the convergence layer (the lowest stack layer that is visible to ION), failure to deliver one or more segments due to segment loss or corruption will trigger segment retransmission if a "reliable" convergence-layer protocol is in use: LTP or DGR "red-part" transmission or TCP (including Bundle Relay Service, which is based on TCP).
- <u>LTP retransmission:</u> Timer interval computation is non-trivial in an environment of scheduled contacts as served by LTP.
  - The round-trip time for an acknowledgment dialogue may be twice the OWLT between sender and receiver at one moment, but it may be hours or days longer at the next moment due to cessation of scheduled contact until a future contact opportunity.
  - The ltpclock task infers the initiation and cessation of LTP transmission from contacts in the contact plan. This controls the dequeuing of LTP segments for transmission by underlying link service adapter(s) and it also controls suspension and resumption of timers, removing the effects of contact interruption from the retransmission regime.

# Operational features

## ➢ *Delivery assurance (2 of 2)*

- At the BP layer:
  - If the nominally reliable convergence-layer protocol fails altogether (e.g., an LTP transmission might be canceled due to excessive retransmission activity), BP detects the CL protocol failure and re-forwards all bundles whose acquisition by the receiving entity is presumed to have been aborted by the failure.
    - If re-forwarding is impossible because transmission to all potentially viable neighbors is blocked, the affected bundles are placed in the limbo list for future re-forwarding when transmission to some node is unblocked.
  - If the bundle is flagged for custody transfer service, the forwarding of a bundle may be explicitly refused by the receiving entity: a "custody refusal signal" (packaged in a bundle) is sent back to the sending node, which must re-forward the bundle in hopes of finding a more suitable route. Potential causes:
    - The receiving node has insufficient resources to store and forward the bundle.
    - The receiving node has no route to the destination.
    - The receiving node will have no contact with the next hop on the route before the bundle's TTL has expired.

# Operational features

➢ ***Rate Control***

On deep space links, signal propagation delays might prohibit effective dynamic negotiation of transmission rates.  Instead, deep space links are operationally reserved for use by designated pairs of communicating entities over pre-planned periods of time at pre-planned rates.

ION avoids congestion in the network by adhering to the planned contact periods and data rates if there is no congestion inherent in the contact plan.

ION's rate control system will enable data flow only while contacts are active.

# Operational features

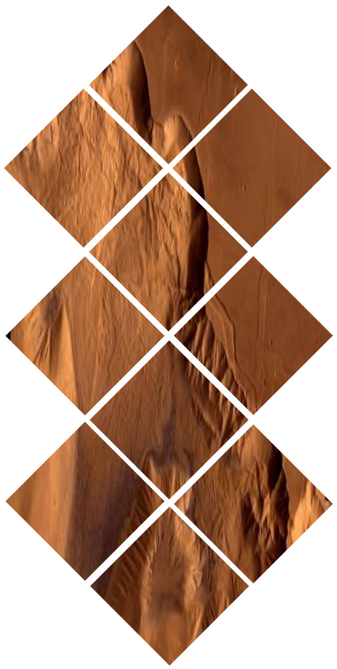
## ➢ *Flow Control*

**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.**

## Topics discussed this afternoon:

| Theory |
| --- |
| Overall Data Flow in ION |
| ION Resource Management |
| Compressed Bundle Headers |
| Managed Aggregation |
| Transactions |
| Zero-copy Objects |
| Contact Graph Routing |
| Concept of Operations |



# Any questions?

# Thank you!

# Goodbye for now!