

CCSDS File Delivery Protocol (CFDP) JPL's Reference Implementation An Overview

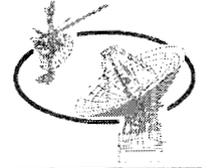
Navid Dehghani

Deputy Manager, Mission Software Systems

May 13, 2003



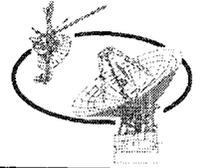
CFDP Reference Implementation



- JPL is developing Reference Implementation of the CCSDS File Delivery Protocol
- Started as a prototype in the Flight System Testbed
- Work is being partially funded by the JPL Standards Organization
- JPL is participating on the CCSDS CFDP Panel (P1F) which is responsible for the development of its specification.



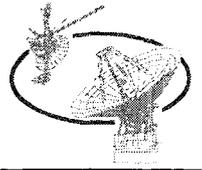
What is CFDP?



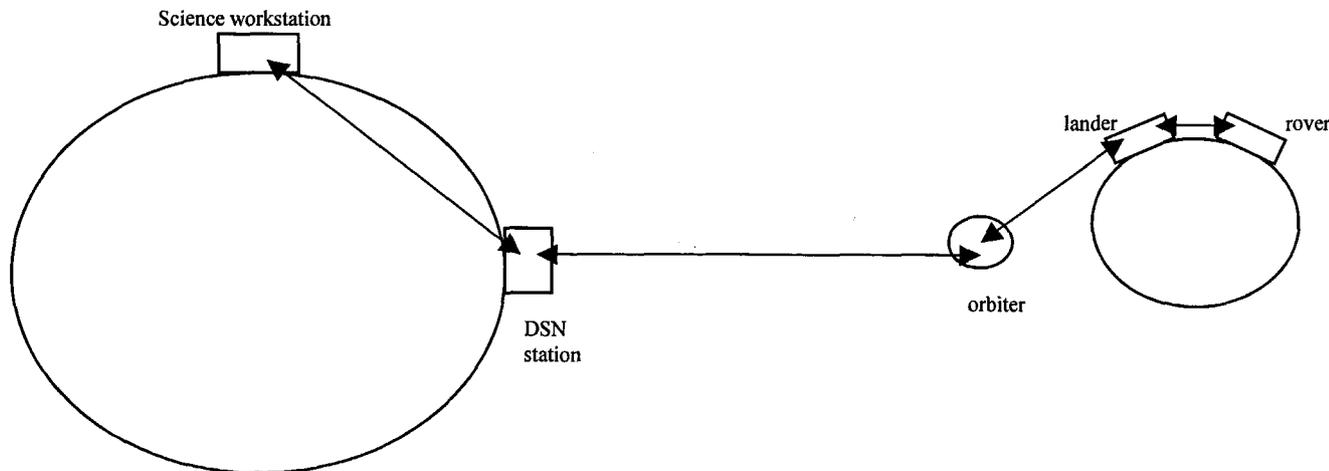
- An international standard for automatic, reliable file transfer between spacecraft and ground (in both directions), built on top of the familiar CCSDS protocols.
- Able to route data across multiple relay points, some of which may be separated by interplanetary distances.
- A single protocol, that relies on the services of underlying Link-layer protocols.
- Handles concurrent file transfers in both directions
- Handles out of order PDU delivery
- Supports acknowledged and unacknowledged transmission modes
- In reliable transmission mode, automatically retransmits lost or corrupted data using one of several lost segment detection modes
- Provides several data integrity measures including file checksums and optional CRCs for each PDU



Core & Extended Procedures



- Core Procedures
 - Single point to point file transmission
- Extended Procedures
 - Multi-hop transfers from waypoint to waypoint
 - Each entity takes custody of the file
 - Retransmission is point-to-point, but transaction state is tracked end-to-end [e.g., rover to desktop]





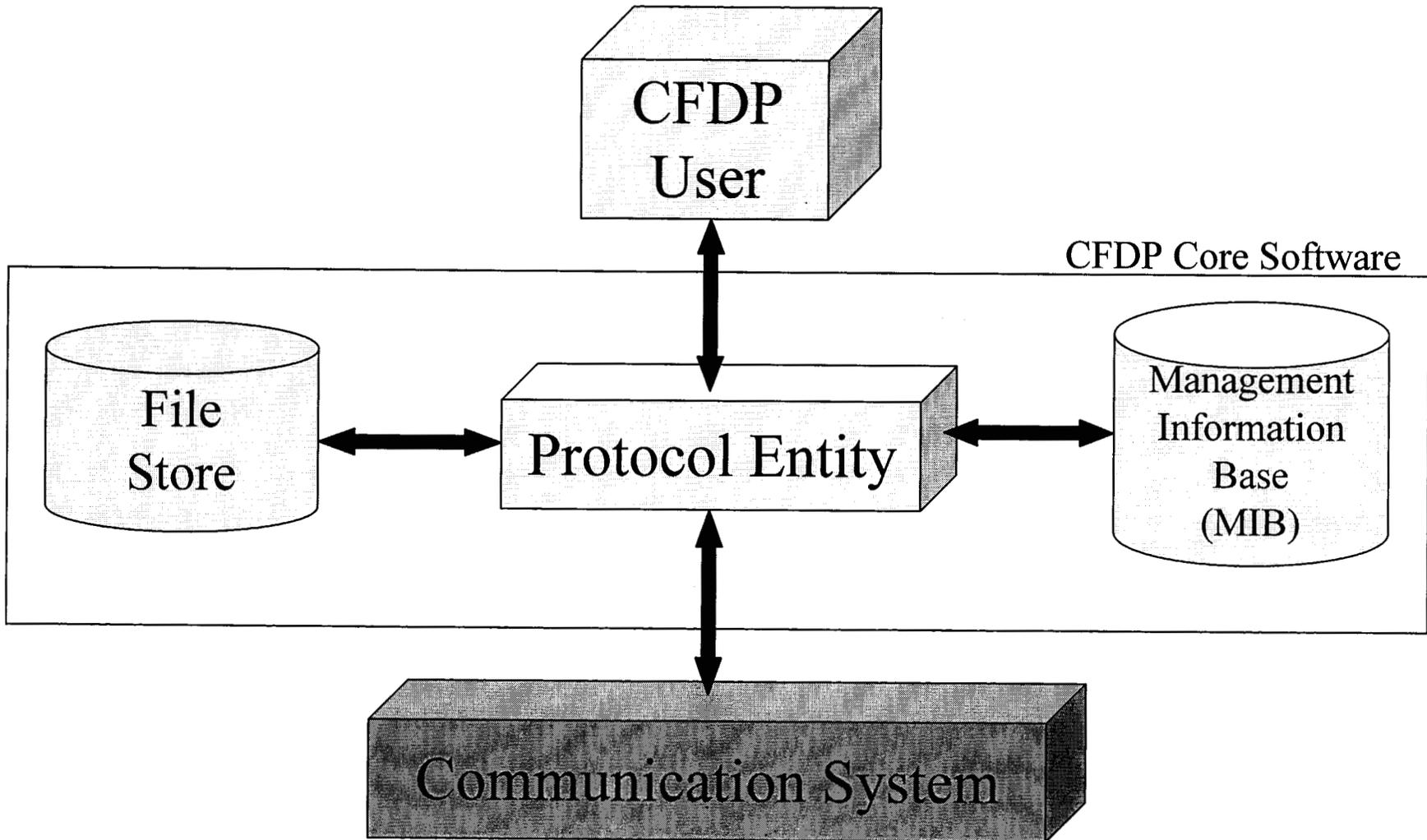
CFDP Classes of Service



- Class 1 - Core procedures
 - Unacknowledged mode file transfer
- ❖ **Class 2 - Core procedures**
 - **Unacknowledged or Acknowledged mode file transfers**
 - **Bounded or Unbounded Data File**
- Class 3 - Extended procedures
 - Unacknowledged transfer of bounded or unbounded data files from a source to destination via waypoints
- Class 4 - Extended procedures
 - Acknowledged transfer of a bounded or unbounded data file from a source to destination via waypoints

❖ Current Class of JPL Reference Implementation

Basic Architecture of a CFDP Implementation





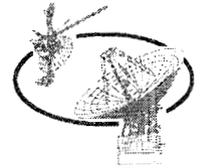
CFDP Terminology



- Entity – a functioning implementation of the CFDP protocol that implements all of its procedures.
- CFDP User – a software task that makes services requests of its the local entity
- Filestore – the media used by the entity to store its files. The protocol operates by copying files from the filestore on the local entity to a filestore on a remote entity.
- Management Information Base – The database used to configure the local CFDP entity.
- Communication System – The underlying communication system referred to as the “Unitdata Transfer Layer” to which all entities in a given domain have access.
- File Data Unit – the concatenation of the filedata and metadata required to transfer a single file between 2 entities.
- Transaction – An end to end transmission of a single File Data Unit between 2 entities.



CFDP User Application



- Each CFDP entity has an associated user application which is not part of the core software.
- The user application performs requests and receives events (or indications) and status from the core application via interface libraries.
- Requests include: put file, cancel transaction, abandon transaction suspend transaction, resume transaction, and report transaction
- Indications include the transaction indication, transaction finished indication, metadata received indication, EOF sent indication, file segment received indication, transaction suspended / resumed indications, a report indication and a fault indication.
- The user application is not standardized by a CCSDS specification and can be a complex application or a simple test program.



Communication System



- The CFDP Entity communicates with other entities via transport layer protocols.
- The UT adapters have been developed which interface to the underlying transport protocols via interface libraries
- These adapters handles mapping of CFDP entity names to protocol level addresses using information stored in the Management Information Base (MIB)
- Each CFDP entity may interface to one or more UT adapters to communicate with different remote entities.
- We used User Datagram Protocol (UDP) adapters in our test environment.



Management Information Base



- Contains information about the local entity:
 - Entity ID
 - Default Fault handlers
 - Default indication configuration
- Provides information about remote entities
 - Mapping between entity names and protocol level addresses
 - One way light time,
 - Timer Values
 - Maximum file segment length
 - Start and end of transmission opportunities



Types of Operations



- Put Request
 - Transfer a file from one filestore to another
 - Perform filestore requests on another filestore
 - Initiate proxy operations
- Cancel Transaction – cancel the specified transaction
- Abandon Transaction – abandon the specified transaction
- Report on Transaction – request a report on a specified transaction
- Suspend/Resume Transaction – suspend or resume a specified transaction
- Set / Request MIB Values



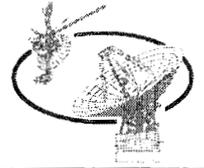
Types of PDUs



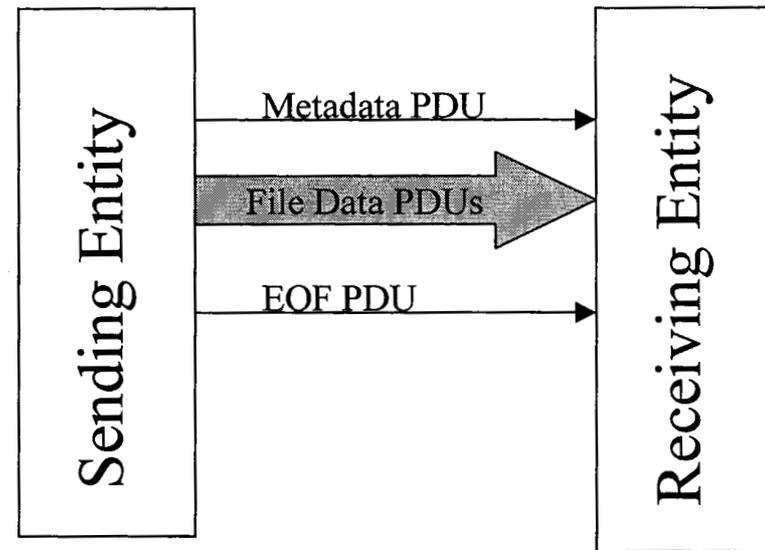
- Metadata PDU – first PDU sent from sender to receiver on start of a file transfer transaction
- File Data PDUs – contain file data
- EOF PDU – sent after all file data has been sent
- Finished PDU – sent by receiver when transaction is complete.
- ACK (EOF or Finished) – acknowledgement of receipt of EOF or Finished PDU (acknowledged mode only)
- NAK PDU – sent by receiver to sender to indicate that a given range of data needs to be resent.
- Keep Alive PDU – sent by sending receiving to tell the sending entity how much data has been received
- Prompt PDU – (Keep Alive or NAK) – prompts receiving entity to sent a Keep Alive PDU or NAK PDU



Unacknowledged Transmission Mode



- One way communication
- No Retransmission
- The sending Entity sends a Metadata PDU, followed by on or more file data PDUs and an EOF PDU.
- The receiving entity does not acknowledge receipt of the EOF PDU and stores the file on receipt of the EOF PDU.

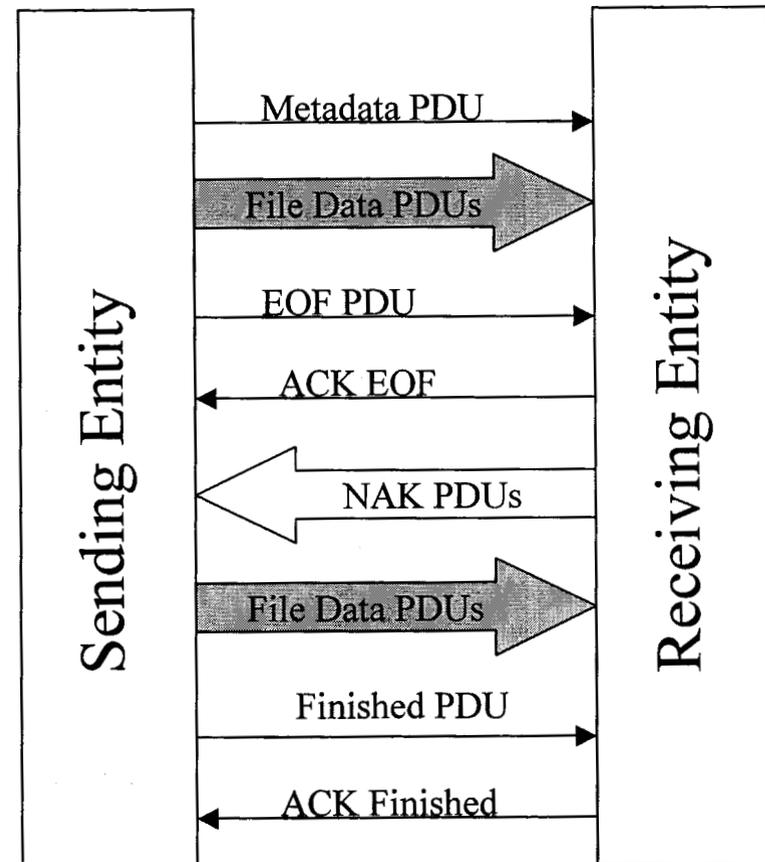




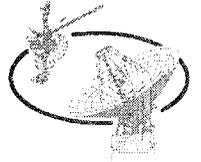
Acknowledged Transmission Mode



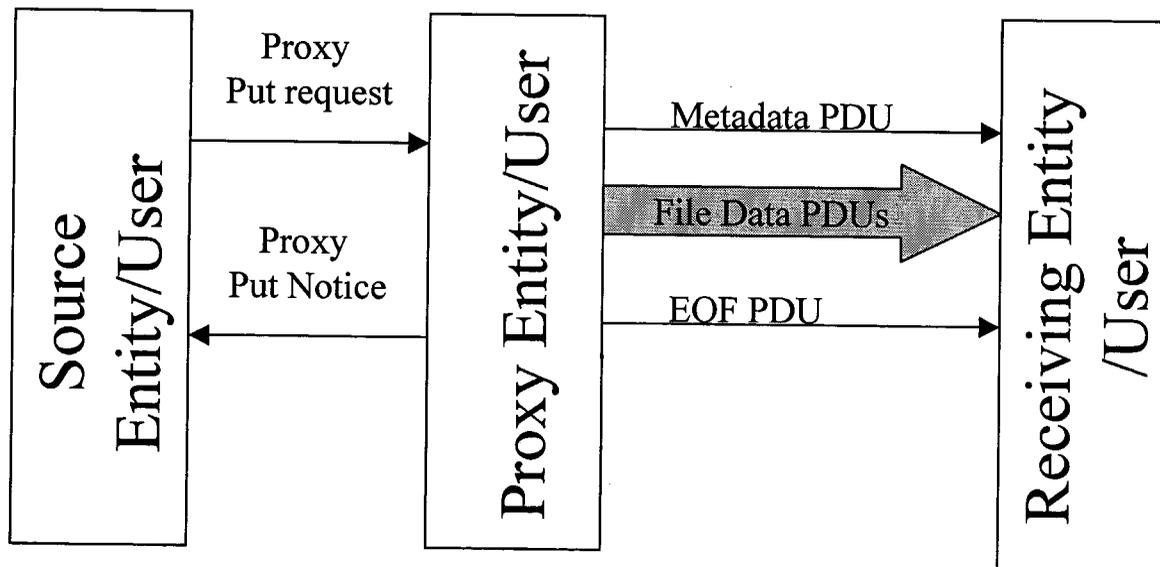
- The sending Entity sends a Metadata PDU, followed by on or more file data PDUs and an EOF PDU.
- The receiving entity acknowledges receipt of the EOF PDU and will send NAK PDUS for those data segments which have not been received.
- Missing data will be NAK'd by the receiving entity a user specified number of times.
- The receiving entity will send a Finished PDU when file transfer is complete with status information regarding the transfer.
- The file is stored to the filestore on sending the Finished PDU.
- The Finished PDU is acknowledge by the sender.



Proxy Operation



- Source Entity User requests proxy entity user to transfer a file to the receiving entity.
- Can be used like a “get” request to get a file from another entity.





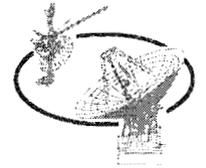
Retransmission Modes



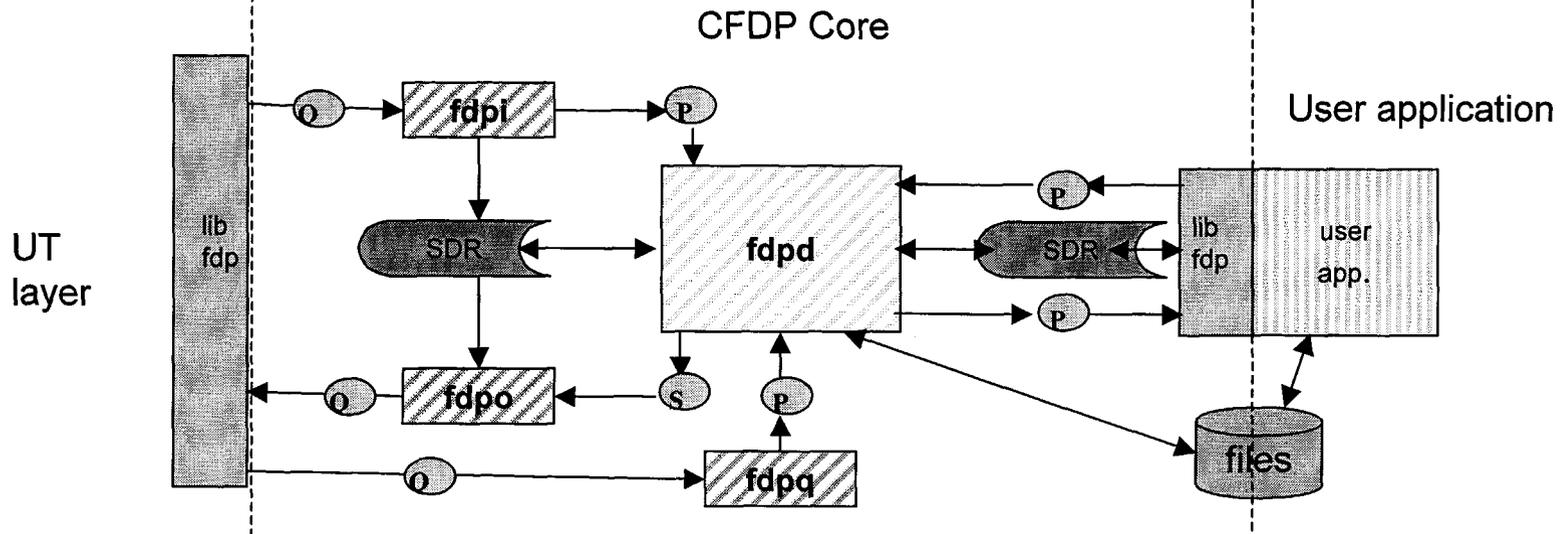
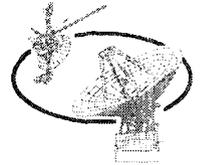
- Immediate Mode – A NAK PDU is issued by the receiving entity as soon as a gap is detected in the data sequence.
- Prompted – A NAK PDU is issued by the receiving entity when prompted by a PROMPT PDU from the sending entity
- Asynchronous – A NAK PDU is issued by the receiving entity when an implementation specific event occurs (opening up of a window of opportunity, timer etc)
- Deferred – A NAK PDU is issued by the receiving entity upon receipt of the EOF PDU if gaps are detected.



Faults and Fault Handling



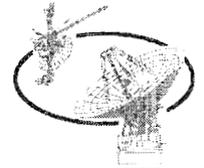
- Types of CFDP Faults
 - Too Many ACKS
 - Too Many KeepAlives
 - Invalid Transmission Mode
 - Filestore Rejection
 - Checksum Error
 - FileSize Error
 - Too Many NAKs
 - Inactivity
 - Invalid File Structure
- CFDP Fault Handlers
 - Abandon Fault Handler
 - Cancel Fault Handler
 - Suspend Fault Handler
 - Ignore Fault Handler



- P pipe
- Q message queue
- S semaphore
- UT layer Universal Transfer layer - interface to transport layer
- SDR Simple Data Recorder (persistent storage for user data objects)
- files source/destination files on virtual file system
- libfdp fdpd and user application API
- user app. Template user application used for testing (sends requests to fdpd and accepts event responses)
- fdpi reads PDUs from message queue, writes them to SDR, pipes notification to fdpd
- fdpd transmission and retransmission daemon
- fdpo responds to semaphore, reads PDUs from SDR, writes them to message queue
- fdpq Notifies fdpd that a critical PDU has been sent.



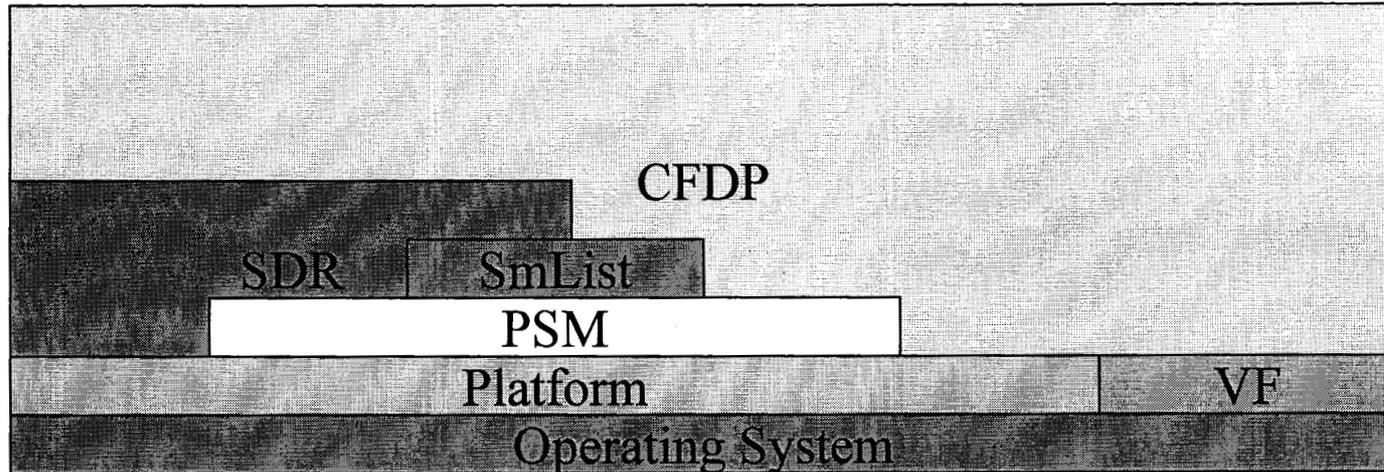
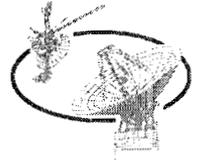
Core Process Architecture



- Processes
 - **fdpd** is the main CFDP process and performs the bulk of protocol's work.
 - **fdpo** process is created to communicate with each contact (a destination entity with which this entity can directly communicate).
 - **fdpi** tells the fdpd task that there is an incoming PDU by sending a token over a pipe
 - **fdpq** tells the fdpd task that a critical PDU has been sent so that fdpd can start its timers.



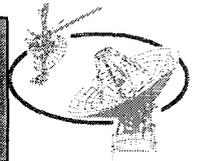
Layered Architecture



CFDP	baseline = fdpd, fdpi, fdpo, fdpq
SDR	Simple Data Recorder = persistent object database in shared memory, using PSM and SMList
SmList	linked lists in shared memory using PSM
PSM	Personal Space Management = memory management within a pre-allocated memory partition
Platform	access to O.S. such as shared memory, system time, P, Q, S
VF	Virtual Filestore = isolates CFDP from I/O
Operating System	thread/spawn/destroy, file system, time, inter process communications



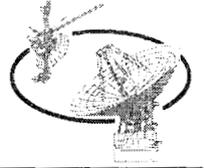
SDR Layer - Persistent Storage



- Simple Data Recorder (SDR) is a rudimentary persistent object system which insulates client code from differences in mass storage implementations (file and DRAM versions available)
- Six layers of functionality:
 - Catalog service for retrieval of object by name.
 - Collection object management: linked lists, self-identifying tables, self-identifying strings.
 - Heap management services: create and destroy arbitrary “objects”.
 - Low-level heap and bulk storage I/O functions.
 - Transaction mechanism for data integrity.
 - Adaptation layer re-implemented separately for each mass storage technology.



SDR Layer - Persistent Storage



- Architecture:
 - Heap area for small, volatile objects, such as linked list elements.
 - Bulk area for large, static data, such as the contents of linked list elements. A large ring buffer.
 - “N” write-ahead logs.
 - Heap is mirrored in DRAM.
- All heap write()s are applied to mirror and to current log.
- All heap read()s access mirror.
- When end of log A is reached, we switch to log B and all committed transactions in log A are applied to heap.
 - Multiple writes to any single area of heap are resolved to a single write before application.



PSM

SmList



- PSM (Personal Space Management)
 - Implements private, high-speed dynamic memory management
 - a pre-allocated memory partition of fixed size.
 - Managed partition may be either private or shared memory.
- SmList uses PSM to implement linked lists in shared memory.



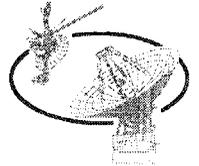
VF (Virtual Filestore)



- Implements C “standard I/O”, insulating CFDP from direct access to operating system functions.
- Enables support for all “virtual filestore” functions identified in CFDP (reading and writing files, renaming files and directories, deleting files and directories, creating files and directories...)
- Implemented as libvf.a, libvf.so
- Header file vf.h



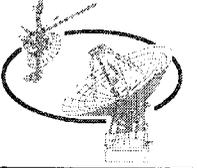
Platform Layer



- Current Platforms
 - Unix
 - Solaris5, Solaris 7, Solaris 8, Solaris 9
 - Linux
 - VxWorks 5.2, VxWorks 5.4
- File Systems
 - NFS, DOS



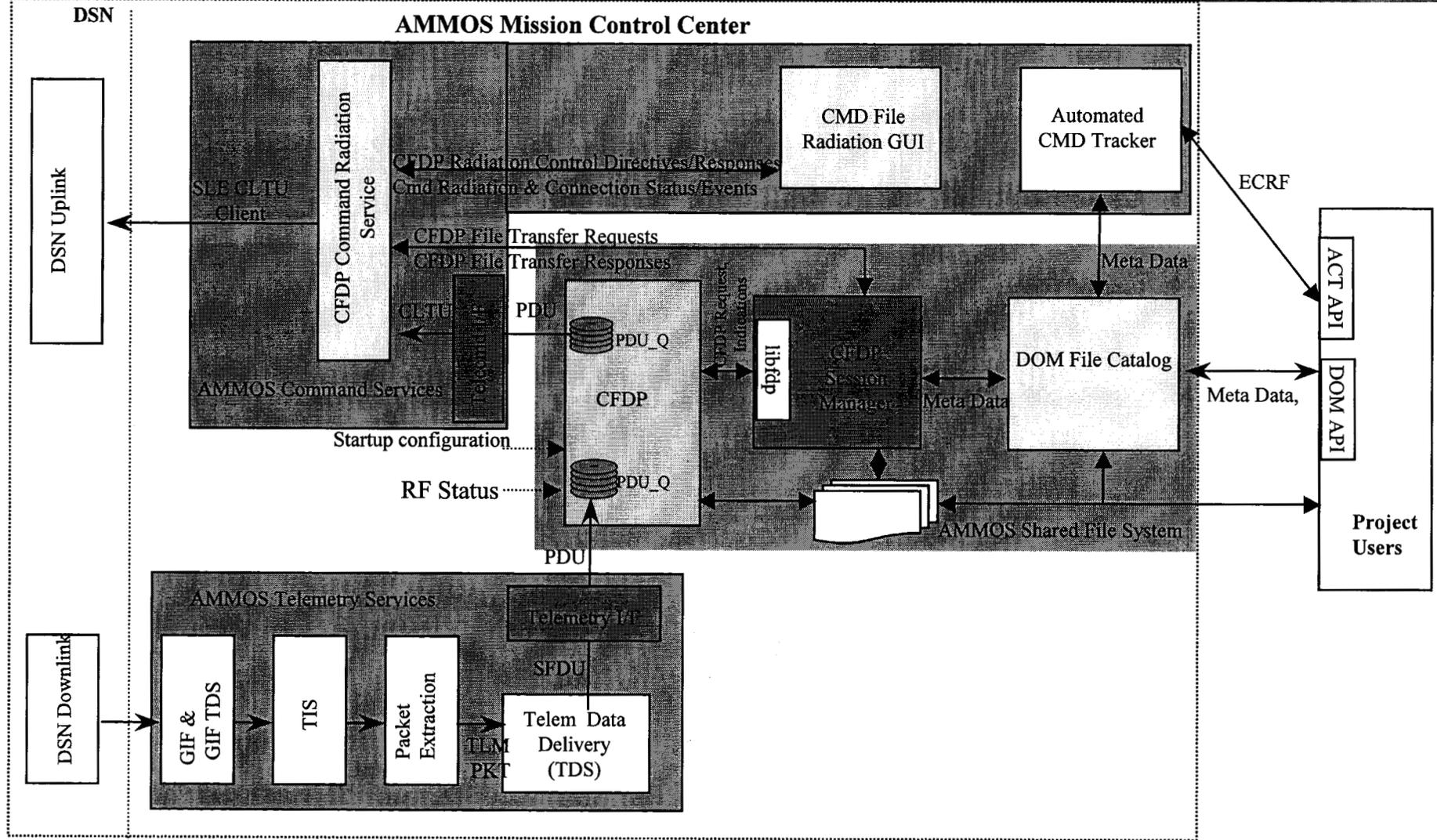
Who Uses our Implementation?



- Current Customers at JPL:
 - Deep Impact Flight Project (Ball)
 - TC&DM Ground System – FDM Subsystem
 - MRO
- Potential Customers at JPL:
 - MDS
 - DAWN Mission
 - MMO
- Non-JPL Customers:
 - Messenger Ground System at the Applied Physics Laboratory Ames Research Center
 - Various Universities



GDS S/W Design and Dataflow





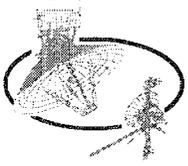
Other Implementations of CFDP Core Software



- European Space Agency (ESA)
 - DOS based implementation in Object Pascal
- Japanese Space Agency (NASDA)
 - DOS based implementation in C++
- Goddard (NASA)
 - Unix based implementation in C
- British Space Agency (BNSD)
 - Linux based implementation in C



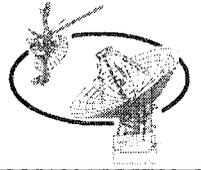
Near Term Goals



- Delivery of flight ready software to Deep Impact – January 2003 with continuing support through launch.
- Delivery to MMO (V28.0) June 2003
- Delivery to MRO (V28.1) in October 2003
- Delivery to TTC&DM Ground System V28.0 (June 2003)
- Full Blue Book compliance and Extended Procedures implemented by CCSDS Panel 1F meeting in Oct. 2003
- Integration of Store and Forward Overlay and Proxy functions into the user application by CCSDS Panel 1F meeting in Oct. 2003



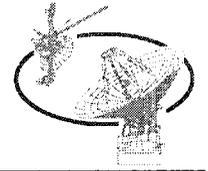
The Team...



- Task Lead - Mike Levesque
- System Engineer - Scott Burleigh
- CFDP CDE / Developer – Kathy Bryan Rundstrom
- CFDP Subsystem Tester - John Veregge
- FDM CDE / Developer - Son Ho
- FDM Subsystem Tester – Ken Lam
- Deep Impact Project Rep. - Felicia Sanders



References



- CCSDS Web Page - CCSDS documents and specifications
 - www.ccsds.org
- JPL CFDP Web Pages for development status and docs
 - www.jpl.nasa.gov/eis/cfdp
- A Standard for the Transmission of IP Datagrams on Avian Carriers
 - <http://www.ietf.org/rfc/rfc1149.txt>.