The Consultative Committee for Space Data Systems

Report Concerning Space Data System Standards

CCSDS FILE DELIVERY PROTOCOL (CFDP)— PART 1: INTRODUCTION AND OVERVIEW

INFORMATIONAL REPORT

CCSDS 720.1-G-4

GREEN BOOK
May 2021

The Consultative Committee for Space Data Systems

Report Concerning Space Data System Standards

CCSDS FILE DELIVERY PROTOCOL (CFDP)— PART 1: INTRODUCTION AND OVERVIEW

INFORMATIONAL REPORT

CCSDS 720.1-G-4

GREEN BOOK

May 2021

AUTHORITY

Issue:	Informational Report, Issue 4
Date:	May 2021
Location:	Washington, DC, USA

FOREWORD

This document is a CCSDS Report, which contains background and explanatory material to support the CCSDS Recommended Standard, *CCSDS File Delivery Protocol* (reference [1]).

Through the process of normal evolution, it is expected that expansion, deletion, or modification to this Report may occur. This Report is therefore subject to CCSDS document management and change control procedures, which are defined in reference [2]. Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this report should be addressed to the CCSDS Secretariat at the address on page i.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 720.1-G-1	CCSDS File Delivery Protocol (CFDP)—Part 1: Introduction and Overview	January 2002	Original issue, superseded
CCSDS 720.1-G-2	CCSDS File Delivery Protocol (CFDP)—Part 1: Introduction and Overview	September 2003	Issue 2, superseded
CCSDS 720.1-G-3	CCSDS File Delivery Protocol (CFDP)—Part 1: Introduction and Overview	April 2007	Issue 3, superseded
CCSDS 720.1-G-4	CCSDS File Delivery Protocol (CFDP)—Part 1: Introduction and Overview	May 2021	Current issue

CONTENTS

Section Page

1 INTRODUCTION 1-1

1.1 PURPOSE 1-1
1.2 SCOPE 1-1
1.3 ORGANIZATION OF THIS REPORT 1-1
1.4 CONVENTIONS AND DEFINITIONS 1-1
1.5 REFERENCES 1-4

2 SUMMARY AND OVERVIEW 2-1

2.1 BACKGROUND 2-1
2.2 OPERATIONAL CONTEXT 2-3
2.3 DESIGN CONCEPT 2-4
2.4 ARCHITECTURE ELEMENTS 2-6
2.5 PROTOCOL OPERATIONS 2-8
2.6 PROTOCOL RELIABILITY OPTIONS 2-13
2.7 PRIMITIVES, PDUS, AND PIPES 2-20

3 EXAMPLE CONFIGURATIONS 3-1

3.1 OVERVIEW 3-1
3.2 FILE DELIVERIES 3-1

ANNEX A ABBREVIATIONS AND ACRONYMS A-1
ANNEX B MAJOR REVISIONS TO VERSION 3 BLUE BOOK B-2

Figure

1-1 Bit Numbering Convention 1-2
1-2 Octet Convention 1-2
2-2 The CFDP Operates over a Wide Range of Underlying Protocols 2-4
2-3 Core and Extended Interactions 2-4
2-4 Architectural Elements of the File Delivery Protocol 2-6
2-5 Possible Virtual Filestore Structure 2-7
2-7 Unreliable Service Mode 2-14
2-8 Deferred NAK Mode 2-15
2-9 Immediate NAK Mode 2-16
2-10 Prompted NAK Mode 2-17
2-11 Asynchronous NAK Mode 2-18
2-12 Time-out Triggered NAK Retransmission 2-19
2-13 Time-out Triggered EOF and Finished Retransmissions 2-19

CONTENTS (continued)

Section Page

2-14 Request Primitives, PDUs, and Indication Primitives—Operational View 2-21
2-15 CFDP Pipe Diagram 2-22
3-1 Example Configuration 1 3-2
3-2 Example Configuration 1a 3-2
3-3 Unreliable Download.. 3-4
3-4 Reliable Download 3-7
3-5 Reliable Upload 3-10
3-6 Two Party Proxy (Get)... 3-12
3-7 Example Mission Configuration 2 (Three Party Proxy)... 3-14
3-8 Three Party Proxy 3-15

Table

2-1 CFDP Service Primitives 2-9
2-2 Abbreviations.. 2-14

1 INTRODUCTION

1.1 PURPOSE

This Report is an adjunct document to the Consultative Committee for Space Data Systems (CCSDS) Recommended Standard for File Delivery Protocol (reference [1]), and it contains material that will be helpful in understanding the primary document. This Report will assist decision makers and implementers with evaluating the applicability of the protocol to mission needs, as well as with making implementation, option selection, and configuration decisions related to the protocol.

1.2 SCOPE

This Report provides supporting descriptive and tutorial material**. This document is not part of the Recommended Standard.** In the event of conflicts between this Report and the Recommended Standard, the Recommended Standard shall prevail.

1.3 ORGANIZATION OF THIS REPORT

This Report is divided into two parts. The first part (this document) provides an introduction to the concepts, features, and characteristics of the CCSDS File Delivery Protocol (CFDP). It is intended for an audience of persons unfamiliar with the CFDP or related protocols. This Report contains three sections and two annexes, as follows:

a) section 1, Introduction;

b) section 2, Summary and Overview (of the CFDP);

c) section 3, Example Configurations (i.e., possible configurations using the protocol);

d) annex A, Abbreviations and Acronyms;

e) annex B, Major Revisions to Version 3 Blue Book.

The second part of this Report (reference [3]) is an implementer’s guide. It provides information to assist implementers in understanding the details of the protocol and in the selection of appropriate options, and contains suggestions and recommendations about implementation-specific subjects. The second part also contains implementation reports from various member Agencies, and the requirements upon which the CFDP is based.

1.4 CONVENTIONS AND DEFINITIONS

1.4.1 BIT NUMBERING CONVENTION AND NOMENCLATURE

In this document, the following convention is used to identify each bit in an N-bit field. The first bit in the field to be transmitted (i.e., the most left-justified when drawing a figure) is defined to be ‘Bit 0’; the following bit is defined to be ‘Bit 1’ and so on, up to ‘Bit N-1’. When the field is used to express a binary value (such as a counter), the Most Significant Bit (MSB) shall be the first transmitted bit of the field, that is, ‘Bit 0’, as shown in figure 1-1.

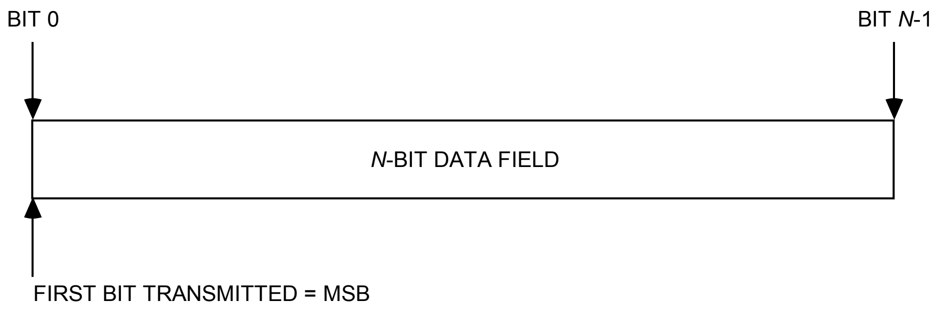

Figure 1-1: Bit Numbering Convention

In accordance with modern data communications practice, spacecraft data fields are often grouped into 8-bit ‘words’ that conform to the above convention. Throughout this Report, the nomenclature shown in figure 1-2 is used to describe this grouping.

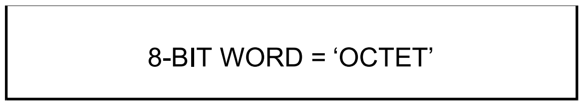

Figure 1-2: Octet Convention

By CCSDS convention, all ‘spare’ bits shall be permanently set to value ‘zero’.

1.4.2 DEFINITIONS

Within the context of this document, the following definitions apply:

A *file* is a bounded or unbounded named string of octets that resides on a storage medium.

A *filestore* is a system used to store files; CFDP defines a standard *virtual filestore* interface through which CFDP accesses a filestore and its contents.

A *CFDP protocol entity* (or *CFDP entity*) is a functioning instance of an implementation of the CFDP protocol, roughly analogous to an Internet protocol ‘host’. Each CFDP entity has access to exactly one filestore, which may comprise any number of allocations of storage space in any number of physical devices. Each entity also maintains a *Management Information Base* (MIB), which contains such information as default values for user communications requirements, for example, for address mapping, and for communication timer settings.

The functional concatenation of a file and related *metadata* is termed a *File Delivery Unit* (FDU); in this context, the term ‘metadata’ is used to refer to any data exchanged between CFDP protocol entities in addition to file content, typically either additional application data (such as a ‘message to user’) or data that aid the recipient entity in effectively utilizing the file (such as file name). It should be noted that an FDU may consist of metadata only. It should also be noted that the term ‘file’ is frequently used in this document as an abbreviation for ‘file delivery unit’; only when the context clearly indicates that only actual files are being discussed should the term ‘file’ not be read as ‘file delivery unit’.

The individual, bounded, self-identifying items of CFDP data transmitted between CFDP entities are termed *CFDP Protocol Data Units* or *CFDP PDU*s. Unless otherwise noted, in this document the term ‘PDU’ always means ‘CFDP PDU’. CFDP PDUs are of two general types: *File Data PDUs* convey the contents of the files being delivered, while *File Directive PDUs* convey only metadata and other non-file information that advances the operation of the protocol.

A *transaction* is the end-to-end transmission of a single FDU between two CFDP entities. A single transaction normally entails the transmission and reception of multiple PDUs.

Any single end-to-end file transmission task has two associated entities: the entity that has the file at the beginning of the task (the *source*), and the entity that has a copy of the file when the task is completed (the *destination*).

Each end-to-end file transmission task includes a point-to-point file copy operation. Any single file copy operation has two associated entities: the entity that has a copy of the file at the beginning of the operation (the *sender*) and the entity that has a copy of the file when the operation is completed (the *receiver*).

The term *CFDP user* is used to refer to the software task that causes the local entity to initiate a transaction or the software task that is notified by the local entity of the progress or completion of a transaction. The CFDP user local to the source entity is referred to as the *source CFDP user*. The CFDP user local to the destination entity is referred to as the *destination CFDP user*. The CFDP user may be operated by a human or by another software process. Unless otherwise noted, the term *user* always refers to the CFDP user.

A *message to user* (or *user message*) allows information related to a transaction to be delivered to the destination user, in synchronization with the transaction.

A *filestore request* is a request to the remote filestore for service (such as creating a directory or deleting a file) at the successful completion of a transaction.

Service primitives form the software interface between the CFDP user and its local entity. The user issues *request* service primitives to the local entity to request protocol services, and the local entity issues *indication* service primitives to the user to notify it of the occurrence of significant protocol events.

1.5 REFERENCES

The following documents are referenced in the text of this Report. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Report are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS Recommended Standards.

[1] *CCSDS File Delivery Protocol (CFDP)*. Issue 5. Recommendation for Space Data System Standards (Blue Book), CCSDS 727.0-B-5. Washington, D.C.: CCSDS, July 2020.

[2] *Organization and Processes for the Consultative Committee for Space Data Systems*. Issue 4. CCSDS Record (Yellow Book), CCSDS A02.1-Y-4. Washington, D.C.: CCSDS, April 2014.

[3] *CCSDS File Delivery Protocol (CFDP)—Part 2: Implementers Guide*. Issue 4. Report Concerning Space Data System Standards (Green Book), CCSDS 720.2-G-4. Washington, D.C.: CCSDS, May 2021.

2 SUMMARY AND OVERVIEW

2.1 BACKGROUND

In recent years, CCSDS has concentrated on providing flexible and efficient transfer protocols for various data over space links.

The basic CCSDS suite solves the data transfer problems for current missions in which the manipulation of onboard storage tends to be handled either manually or by ad hoc protocols developed privately. While this is an acceptable way of managing a limited amount of memory, with the rapid development and take-up of solid state mass memory, this is no longer the case.

The availability of gigabytes of solid state memory leads to a new era of spacecraft operation in which much of the routine traffic to and from the spacecraft will be in the form of files. Furthermore, because of the random-access nature of the onboard storage medium, it becomes possible to repeat transmission of data lost on the link and thus guarantee delivery of critical information.

Drivers Toward CFDP

– Spacecraft now use mass memory with very large data files.

– For cost reasons, the trend is toward more autonomous operation whereby the spacecraft ‘decides’ (for example) when it should download stored data and when it should upload new operational plans.

– Interoperability within and among Agencies, and between space-ground networks (e.g., toward interoperability with the ground-based Internet) is becoming increasingly important as economic considerations require consolidation of networks.

– Some of the new deep space missions do not have direct line of sight between Earth and final destination; rather, data must be relayed between a series of spacecraft, each providing a store-and-forward capability, until the final destination is reached.

– Spacecraft constellations (e.g., fixed or formation-flying) require efficient and reliable data file transfer.

– The increasing onboard use of real-time operating systems (such as VxWorks and RTEMS), which assume the presence of a ‘file system’, make onboard data handling increasingly file-oriented.

While the onboard storage medium has rapidly evolved, the essential constraints of space missions remain.

Mission Constraints

- Systems resources that may be restricted in one or both of the entities involved in an end-to-end data transfer may include computational power and memory capacities, driven by the need for expensive parts qualification, as well as the need to limit power, weight, and volume in the remote end system.
- Environmental restrictions may include noisy, bandwidth-limited, asymmetrical, and interrupted communications links with very long propagation delays.
- User needs often include a requirement for early access to transferred data regardless of its quality, as well as a method of providing data of progressively increased quality.

In response to these factors, the CFDP has been developed to complement the existing CCSDS space packet and encapsulation packet protocol standards.

What is CFDP?

- CFDP provides the capability to transfer ‘files’ to and from a spacecraft mass memory.
- The content of the files may be anything from a conventional timeline update to an unbounded SAR image.
- Files can be transferred reliably, when it is guaranteed that all data will be delivered without error, or unreliably, when a ‘best effort’ delivery capability is provided.
- Files can be transmitted with a unidirectional link, a half-duplex link, or a full-duplex link, with near-Earth and deep-space delays.
- File transfer can be triggered automatically or manually.

NOTE – CFDP was designed to support the transfer of true files stored in a true file system. However, because CFDP is based on the concept of an abstract ‘virtual filestore’ (which in practice might be implemented in ways that are wholly unlike conventional ‘file systems’), and because the CFDP specification does not define exactly what a ‘file’ is, the protocol can in practice be used to convey blocks of data between repositories that may not look like file systems. Such use, however, is a private matter within the using organization and should be defined by a local technical note or agreement among the using parties.

The CFDP has many unique characteristics compared to terrestrial file transfer protocols.

Distinctive Features of CFDP Compared to Terrestrial File Transfer Protocols

- Efficient operation over simplex, half-duplex, and full-duplex links.
- Transfers that can span ground station contacts (time disjoint connectivity).
- Transfers that can span multiple ground stations.
- Effectiveness over highly unbalanced link bandwidths.
- Minimization of link traffic.
- Data availability to the user as the file is received.
- Minimization of onboard memory requirements through buffer sharing.
- Automatic file store-and-forward operation.
- Effectiveness spanning low Earth orbit and deep space.

2.2 OPERATIONAL CONTEXT

The CFDP enables the moving of a file from one filestore to another, in which the two filestores are in general resident in separate data systems and often with an intervening space link. In addition to the purely file-delivery-related functions, the protocol also includes file-management services to allow control over the storage medium.

The protocol is independent of the technology used to implement data storage and requires only a few fundamental filestore capabilities in order to operate. It assumes that there are two filestores, typically one within the spacecraft and one on the ground, and operates by copying data between the two filestores.

The protocol makes no assumptions about the information being transferred and can be utilized for a wide range of applications involving the loading, dumping, and control of spacecraft storage.

The protocol has been specifically designed to minimize the resources required for operation. It is also scalable, so that only those elements required to fulfill the selected options need be implemented.

The protocol can operate over a wide range of underlying communication services, specifically including CCSDS packet services (see figure 2-1). It may be noted that a promising alternative is to run CFDP over Delay-Tolerant Networking protocols, either over Bundle Protocol, with transmission reliability supported by the underlying Licklider Transmission Protocol (LTP), or directly over LTP

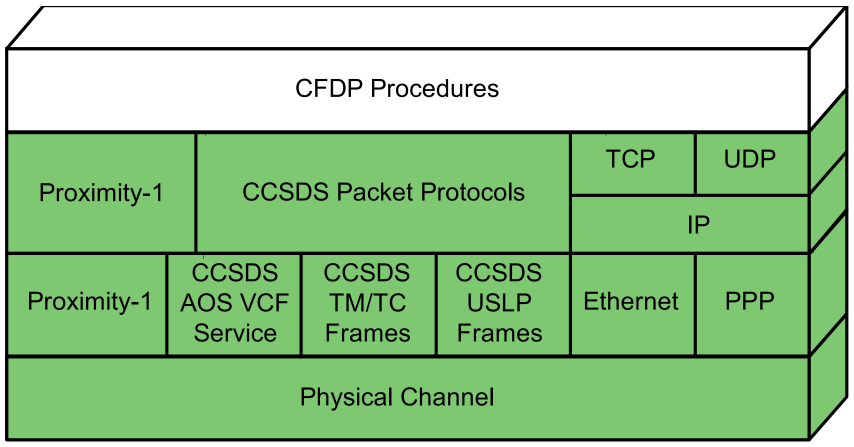

Figure 2-1: The CFDP Operates over a Wide Range of Underlying Protocols

2.3 DESIGN CONCEPT

As depicted in figure 2-2, the protocol consists of Copy File procedures. The Copy File procedures constitute the interaction between two protocol entities with a direct network path between them. The sending entity is the entity from which the file is copied in a file copy operation. The receiving entity is the entity to which the file is copied in a file copy operation.

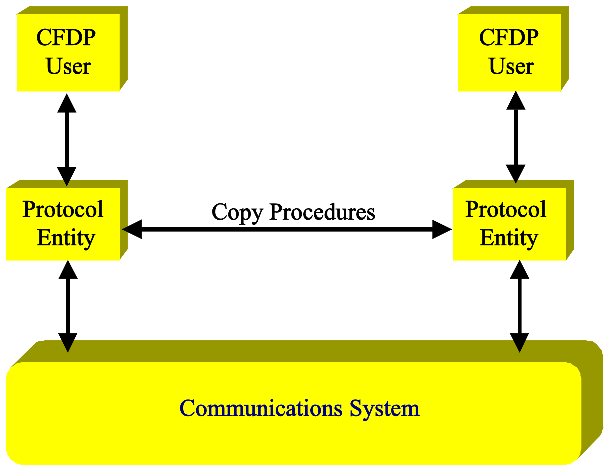

Figure 2-2: CFDP Procedures

When direct network connectivity between the source and destination is impossible, a *Store-and-Forward Overlay* (SFO) system may be added to user applications as a standard user operation. The user application at each relay point examines each incoming file and, if the accompanying metadata indicates that the file's final destination is elsewhere, initiates another point-to-point file transmission either to the final destination or to another relay point that is farther along the route.

It is expected that in deep-space use, a pair of CFDP entities that have files to exchange may at any given moment be unable to communicate; for example, a spacecraft orbiting Mars may be on the far side of the planet, unable to transmit to Earth. For this reason, CFDP, when using store-and-forward overlay, is built entirely on a *store and forward* communication model. If transmission of a file from Earth to a Mars-orbiting spacecraft is interrupted when the spacecraft passes behind the planet, the CFDP entities at both ends of the transmission simply store their outbound Protocol Data Units (PDUs) (possibly in non-volatile memory, to assure continued service even in the event of an unplanned system reset) until the spacecraft re-emerges and transmission can resume. A collateral benefit of this model is that it largely insulates user applications from the state of the communication system: an instrument can record an observation in a file and 'transmit' it (that is, submit it to CFDP for transmission) to Earth immediately without considering whether or not physical transmission is currently possible. By sequestering outbound data management and transmission planning functions within CFDP, this *deferred transmission* can simplify flight and ground software and thereby reduce mission costs.

Using powerful forward error correction coding can minimize data loss in communication across deep space but cannot eliminate it altogether. Consequently, CFDP supports optional 'acknowledged' modes of operation in which data loss is automatically detected and retransmission of the lost data is automatically requested. However, the large signal propagation delays that characterize interplanetary transmission limit the usefulness of the retransmission strategies commonly used in terrestrial protocols. For example, delaying the transmission of PDU *N* until an acknowledgment that PDU *N - 1* (or even PDU *N - 100*) has been received would significantly retard data flow if the round-trip time on the link exceeded the time required to radiate the PDU(s) for which acknowledgment is required. For this reason, CFDP's retransmission model is one of *concurrent transmission*: data PDUs for multiple files may be transmitted as rapidly as possible, one after another, without waiting for acknowledgment, and requests for retransmission are handled asynchronously as they are received. As a result, portions of multiple files may be in transit concurrently.

2.4 ARCHITECTURE ELEMENTS

2.4.1 GENERAL

The architectural elements involved in the file delivery protocol are depicted in figure 2-3 and described in subsections 2.4.2 through 2.4.6.

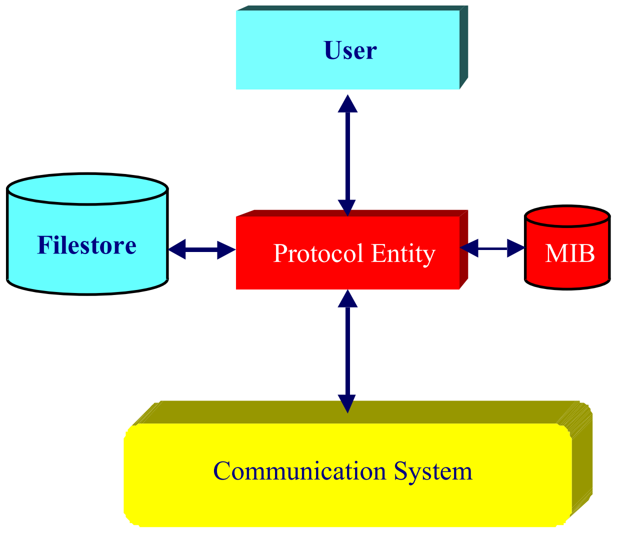

Figure 2-3: Architectural Elements of the File Delivery Protocol

As the figure indicates, each protocol entity has access to exactly one filestore and is accessed by exactly one user.

2.4.2 USER

The protocol operates at the request of the CFDP user. The user interacts with the protocol using the service primitives. A CFDP user is always a software task, which may or may not be operated by a human. Each CFDP protocol entity has one user.

2.4.3 PROTOCOL ENTITY

The protocol entity consists of implementations of the *Copy File procedures*, which allow immediate file delivery and manipulation over a single network hop.

2.4.4 FILESTORE

The protocol operates by copying files from storage medium to storage medium, and it is therefore assumed that all CFDP entities have access to a local storage capability. As the ways in which the storage capability is provided will vary, the protocol is built on the premise that any file or organized set of files (i.e., a filestore) can be described in terms of a single standard representation. This representation, called a '*Virtual Filestore*', is assigned a standard set of attributes with which the protocol manages the file delivery process. This approach allows complete independence from the technology used to implement the filestore.

In an implementation, the virtual filestore must be mapped to and from the actual hardware and software that constitute the real filestore. The virtual filestore has been defined to be as universal as possible while still resembling the interface provided to most real filestores. Therefore, the convergence function required to adapt real filestores to this model is easily realizable. Figure 2-4 shows an example of such a possible implementation structure.

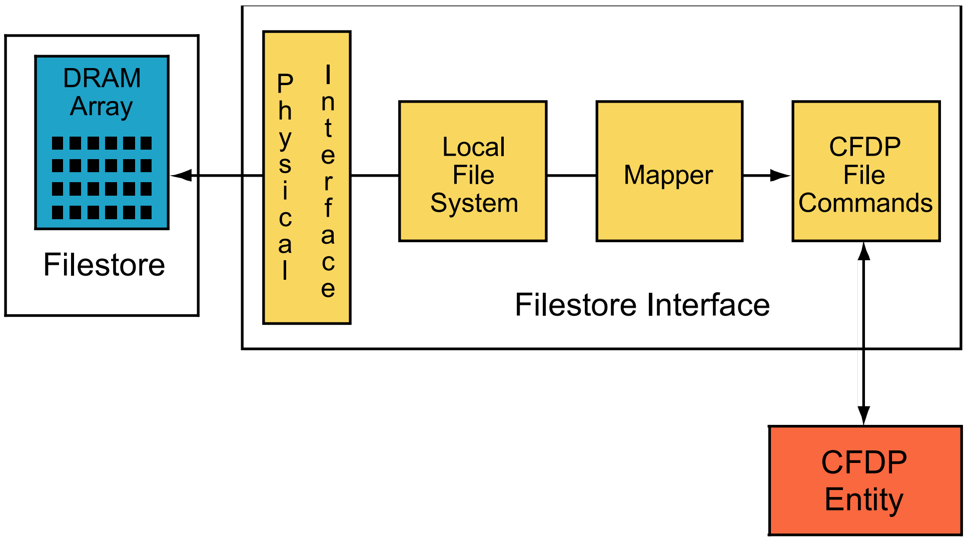

Figure 2-4: Possible Virtual Filestore Structure

To enable interoperability, the protocol assumes a minimum set of capabilities from the Virtual Filestore, but also provides an extensibility mechanism to support use of additional capabilities. The minimum required filestore capabilities are:

- create file;
- delete file;
- deny file;
- rename file;
- append file;
- replace file;
- create directory;
- remove directory;
- deny directory;
- list directory.

In some circumstances, it is advantageous for the CFDP protocol to be able to recognize record boundaries within the file. If this option is to be used, the filestore must have the capability to make the distinction between such files and those that are to be treated as a stream of octets.

2.4.5 MANAGEMENT INFORMATION BASE

To perform a file delivery, a significant amount of information must be passed by the local user to its local CFDP entity, and by the local CFDP entity to the remote CFDP entity. Typically, this data is static and is maintained by the CFDP entities as system tables, referred to as the Management Information Base (MIB). The MIB contains such information as default values for user communications requirements, for example, for address mapping, and for communication timer settings. The MIB is defined as part of the protocol specification.

2.4.6 UNDERLYING COMMUNICATION SYSTEM

The protocol assumes the availability of an underlying communication system to which all CFDP entities in a given CFDP addressing domain have access. In order that the protocol may operate over a wide range of implementations including CCSDS, Internet, and Delay-Tolerant Networking (DTN) networks, the services required by the protocol have intentionally been kept as simple as possible. This underlying conceptual communication system is referred to as the Unitdata Transfer (UT) layer. Since only minimal network capabilities are assumed, services sometimes provided by the UT layer (such as transaction multiplexing, sequence auditing, and error detection) are provided by CFDP.

It should be noted that although the CFDP protocol can operate over a service in which data errors, data loss, and out-of-sequence delivery occur, it is not intended to compensate for networks in which these effects are prevalent. Severe performance reductions will result if such an approach is taken.

2.5 PROTOCOL OPERATIONS

2.5.1 SERVICE INTERFACE

2.5.1.1 Service Primitives

The software interface between the CFDP user and its local entity consists of two types of service primitives: ‘request’ primitives and ‘indication’ primitives. The user issues ‘request’ service primitives to the local entity to request protocol services, and the local entity issues ‘indication’ service primitives to the user to notify it of the occurrence of significant protocol events. Each primitive has parameters that convey related information.

Table 2-1 lists the five ‘request’ primitives and eleven ‘indication’ primitives that make up the CFDP user interface.

Table 2-1: CFDP Service Primitives

Request Primitives	Indication Primitives
Put	Transaction
Cancel	Metadata-Recv
Suspend	File-Segment-Recv
Resume	Suspended
Report	Resumed
	EOF-Sent
	Transaction-Finished
	Report
	Fault
	Abandoned
	EOF-Recv

The protocol specification contained in reference [1] formally defines the service primitives in detail; the remainder of this subsection 2.5.1 gives a brief overview of the operation of the request primitives and their relation to the indication primitives that the protocol entities generate as a result.

2.5.1.2 The CFDP Transaction

The *transaction* is the fundamental operation that the protocol performs to transfer user data between CFDP entities. Each transaction is the end-to-end transmission of a single FDU from a single *source* CFDP entity to a single *destination* CFDP entity. The entities communicate with each other using the PDU messages and procedures defined in the protocol specification.

Each CFDP entity is identified by a unique *entity ID*. Each transaction is identified by a unique *transaction ID* that consists of the ID of the transaction's source entity and a transaction sequence number that the source entity assigns when it initiates the transaction.

A transaction may transfer a file from the local filestore, one or more user messages, and/or one or more filestore requests to the destination entity.

User messages allow delivery of information related to a transaction in synchronization with the transaction; they are not intended for general-purpose messaging. CFDP defines a small number of reserved user messages to implement user operations transactions. Except for these reserved messages, the contents and meanings of user messages are not defined or constrained by the CFDP.

There is an inactivity timer that, if there is a cessation of PDU reception for a given Transaction for the specified time period, causes the issuance of a Fault.indication to the local user. It takes no other action. That is, any further resulting action is not specified by the protocol, but is taken by the user in an implementation-specific manner. The user might try to restart or otherwise salvage the Transaction, abandon it, or take some other action appropriate to the implementation and operational configuration. The basic purpose of the timer is to handle situations that are outside of the protocol itself. Examples might be the crashing of the operating system in the other entity party to the ongoing Transaction or the extended failure of a communication link.

2.5.1.3 Put Request

The 'Put' request is issued by the source entity to initiate a CFDP transaction; in fact, every transaction is the result of a Put request. The parameters of the request may contain all the information needed to specify the transaction, including destination entity ID, source and destination file names, messages to user and filestore requests to accompany the file transfer operation, and protocol options. If optional parameters are omitted, the entity supplies default values from the MIB.

As a pure file delivery request, 'Put' only allows the source user to send a file from its local filestore to a remote filestore. However, the ability to include user messages and filestore requests in a Put request enables the requesting user to initiate more complex operations, such as getting a file from the destination entity and then deleting it from the remote filestore. These capabilities are described in more detail in subsection 2.5.2.

When the source user issues a 'Put' request, the local entity uses the request's parameters to build a *metadata* PDU that describes the transaction, and it assigns a unique transaction ID to be used in later service requests and indications related to the transaction. Since concurrent transactions may be active, the entity issues a 'Transaction' indication to pass the ID back to the user. It then initiates transmission procedures to the destination entity.

2.5.1.4 Put Operations

In CFDP procedures, the source entity sends the metadata (which contains any user messages and filestore requests) followed by any file data to the destination entity. Upon receipt of the metadata PDU, the destination entity creates and initializes the data structures it will use to track and control the transaction, retains any filestore requests for later use, and issues the Metadata-Recv indication to its user. The user then retrieves any user messages contained in the metadata, including Proxy and List Directory messages, acts on any of these two message types, and passes on any others in an implementation-specific manner.

Upon receipt of each PDU containing file data, the destination entity optionally issues the File-Segment-Recv indication to its user. (When the source entity sends the EOF PDU for the file, it may optionally notify its user via an EOF-Sent indication. Likewise, when the destination entity receives the EOF PDU for the file, it may optionally notify its user via an EOF-Recv indication). If transfer of the entire FDU completes successfully, the destination entity then executes any filestore requests it originally saved from the metadata.

Upon successful completion of the FDU transfer or, if there were any filestore requests, at the completion of any filestore requests, the destination entity sends a Finished PDU to the source entity and may optionally issue a ‘Transaction-Finished’ indication to its user. Upon receipt of the Finished PDU, the source entity issues a ‘Transaction-Finished’ indication to its user. In both cases, the ‘Transaction-Finished’ indication contains a condition code indicating completion status:

– successful transfer of the complete FDU;

– cancellation by the source or destination CFDP user;

– fault, including protocol error, filestore error, or inactivity.

The ‘Transaction-Finished’ indication also contains a completion status for each of the transaction’s filestore requests, if any.

2.5.1.5 Cancel Request

A ‘cancel’ request may be issued at any time by either the source or destination entity of an ongoing transaction. The request causes the immediate and unconditional cessation of all activities involved in the designated transaction, eliminating it as an activity. The source and destination entities notify their users of the cancellation by issuing the ‘Transaction-Finished’ indication with a condition code indicating cancellation.

2.5.1.6 Suspend Request

A ‘suspend’ request may be issued at any time by either the source or destination entity of an ongoing transaction. The suspend originating entity, if it is the FDU source entity, notifies its user of the suspension by issuing the ‘Suspended’ indication.

2.5.1.7 Resume Request

A ‘resume’ request may be issued at any time by the entity that suspended the transaction. The resume originating entity, if it is the FDU source entity, notifies its user of the resumption by issuing the ‘Resumed’ indication. When the resume responding entity is the FDU destination entity, it may optionally notify its user of the resumption by issuing the ‘Resumed’ indication.

2.5.1.8 Report Request

A ‘report’ request about an ongoing transaction may be issued at any time by either the source or destination entity user; it causes the local CFDP entity to return a status report about the designated transaction in a ‘report’ indication.

2.5.2 USER OPERATIONS

2.5.2.1 Definition

The term 'user operations' refers to the use of the CFDP services offered by the local CFDP entity to cause the CFDP user of a remote CFDP entity to initiate additional CFDP transactions. User operations are implemented using the 'Message to User' capability of the protocol to forward an 'order' to the remote CFDP user, which will in turn initiate a transaction with its local CFDP entity.

Six standard user operations are defined:

a) proxy operations;

b) remote status report operations;

c) remote suspend operations;

d) remote resume operations;

e) directory operations;

f) store-and-forward overlay operations.

2.5.2.2 Proxy Operations

Proxy operations are used to initiate the delivery of a file from a remote CFDP entity to some other user, either to the local user itself (in which case the proxy operation functions as a 'Get') or to the user of some third CFDP entity. The FDU transmitted in a proxy operation normally contains a file but may contain only metadata, such as filestore directives or a Message to User containing an order to another remote CFDP user.

2.5.2.3 Remote Status Report Operations

Remote status report operations are used to request a report of the status of a specified CFDP transaction at the remote entity.

2.5.2.4 Remote Suspend Operations

Remote suspend operations are used to request the suspension of a specified transaction at the remote entity.

2.5.2.5 Remote Resume Operations

Remote resume operations are used to request the resumption of a specified transaction at the remote entity.

2.5.2.6 Directory Operations

Directory operations are used to request a listing of the contents of a specified directory in the remote user's local filestore.

2.5.2.7 Store-and-Forward Overlay Operations

Store-and-forward operations provide a mechanism for transmitting files between users of CFDP entities that may never be in direct communication. Each transmitted file is received, stored, and forwarded in a hop-by-hop manner by intermediate *waypoint users* until it finally reaches a user termed the *agent*, whose CFDP entity can directly communicate with that of the *destination* user.

2.6 PROTOCOL RELIABILITY OPTIONS

2.6.1 GENERAL

The quality of service offered by the protocol is selectable, according to mission requirements and transmission capability, and ranges from an unacknowledged option, whereby a file is transmitted with no attempt at completeness should errors occur (errors will be detected and data discarded), to a fully acknowledged option providing error recovery through retransmission. For the acknowledged mode of operation, several sub-options may be selected by the receiver. These sub-options relate to release time of any Negative Acknowledgments (NAKs) and range from immediate release to deferred release (whereby any NAKs are stored until the assumed end of the transmission). The unacknowledged option is appropriate when two-way communication is not possible, when incomplete transmission is acceptable, or when the underlying communication mechanism already ensures reliable data transfer; in the latter case, the sending entity may request that the receiver send it a notice of 'closure', that is, an acknowledgment of successful file delivery. The acknowledged sub-options share a common acknowledgment mechanism but use different strategies in making retransmission requests to optimize for different scenarios.

This subsection contains sequence diagrams that illustrate how the protocol uses data transmission and protocol control PDUs to implement the various options available in the unreliable and reliable services. Table 2-2 defines the abbreviations used in the figures for various PDU types; the protocol definition in reference [1] defines the meaning and format of these PDUs in detail. In each file delivery operation, the sequence of events between the file *sender* and the file *receiver* is as shown.

Table 2-2: Abbreviations

Abbr.	PDU Type
M	metadata
FD(n)	file data segment
NAK	retransmission request
EOF	end of file (sender to receiver)
FIN	finished (receiver to sender)
ACK	acknowledgment
PRMPT	prompt

2.6.2 UNRELIABLE SERVICE

When unreliable service has been selected, the receiving entity makes no attempt to improve the quality (completeness) of the received file by transmitting information about missing data to the sending entity. That is, the file is sent in what amounts to a simplex mode in that there is only one-way communication, from the sender to the receiver. Figure 2-5 illustrates this mode. A file completion map identifying any missing portions of the file can optionally be delivered to the receiving user. If a CFDP implementation includes this option, it will provide the file completion map as part of the status message returned with the Transaction-Finished indication.

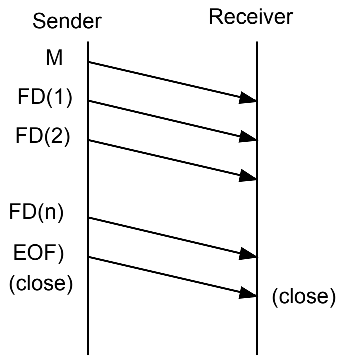

Figure 2-5: Unreliable Service Mode

2.6.3 RELIABLE SERVICE

2.6.3.1 Negative Acknowledgments and Acknowledgments

When reliable service has been selected, the CFDP uses both NAKs and Acknowledgments (ACKs). NAKs are used to request retransmission of lost data. ACKs are used to ensure the receipt of EOF and Finished PDUs.

Since lost data may still be outstanding after the EOF sequence, a Finished PDU is sent by the receiving entity when all file data has been successfully assembled. Delivery is ensured by requiring an ACK for the Finished PDU.

NAK procedures are utilized throughout the transmission. There are four user-selectable options associated with the issuance of NAKs:

– Deferred;

– Immediate;

– Prompted;

– Asynchronous.

In the Deferred NAK mode, the receiving entity saves all information about missing data until the EOF is received. It then issues a NAK to request the missing data. An example is shown in figure 2-6. The deferred NAK mode may be appropriate when communicating entities are very loosely coupled, such as when interplanetary distances introduce very long light time delays.

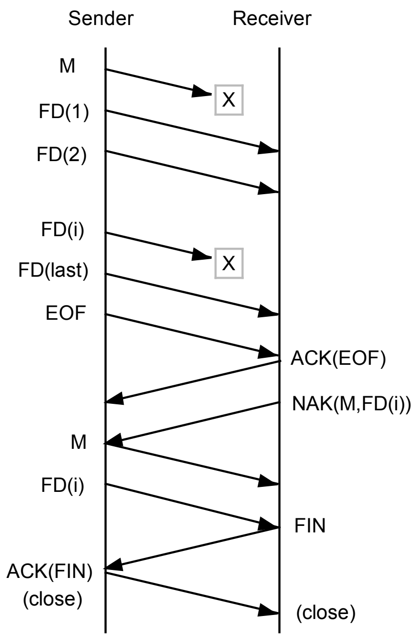

Figure 2-6: Deferred NAK Mode

In the Immediate NAK mode, each discontinuity in the data detected at the receiving entity results in the immediate transmission of a NAK to the sending entity. Examples of this mode are shown in figure 2-7. The Immediate NAK mode is useful, for example, when the communicating entities are tightly coupled; it makes no attempt to control the number of NAK messages it uses in return for maximizing completeness of the received portion of a file as the transfer progresses.

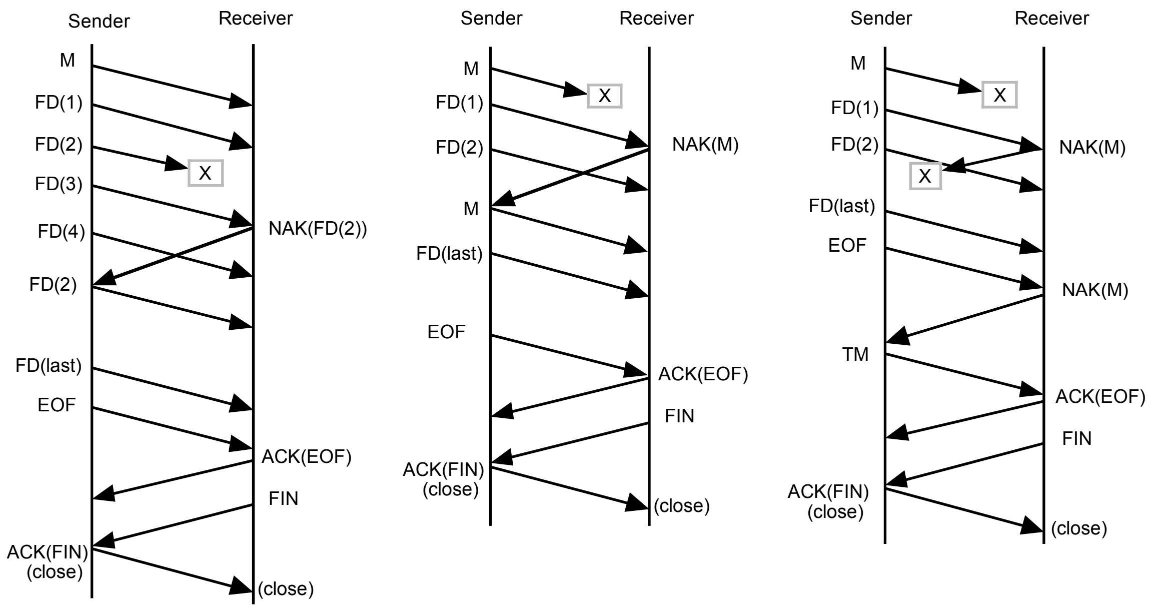

Figure 2-7: Immediate NAK Mode

In the Prompted NAK mode, the sending entity transmits a Prompt (NAK) message to the receiving entity telling it to send its NAK. When the receiving entity receives the Prompt (NAK), it sends any outstanding NAK. In response to a Prompt (NAK) when no data is missing, a CFDP NAK may be empty (that is, request the retransmission of no data). The EOF is treated as an inherent prompt and results in the receiving entity's sending a NAK if any data is missing. This mode is illustrated in figure 2-8. The Prompted NAK mode allows the sending entity to control the frequency of NAK messages, which may be useful, for example, when the return link for sending NAKs is only occasionally available, or is very bandwidth-limited.

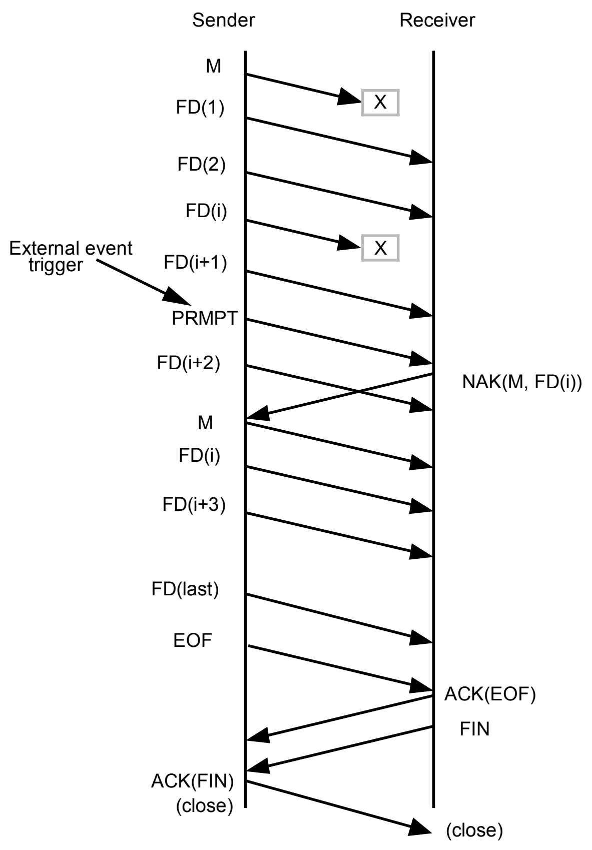

Figure 2-8: Prompted NAK Mode

In the Asynchronous NAK mode, the receiving entity issues a NAK (if any data is missing) in response to some outside event; that is, the receiving entity is triggered by something outside of the CFDP to send any necessary NAK. Such an external event might for instance be the impending loss of the space-to-ground link. An example of this mode is shown in figure 2-9. Like the Prompted NAK mode, the Asynchronous NAK mode also limits the frequency of NAK messages, but in this case, the receiving entity has control.

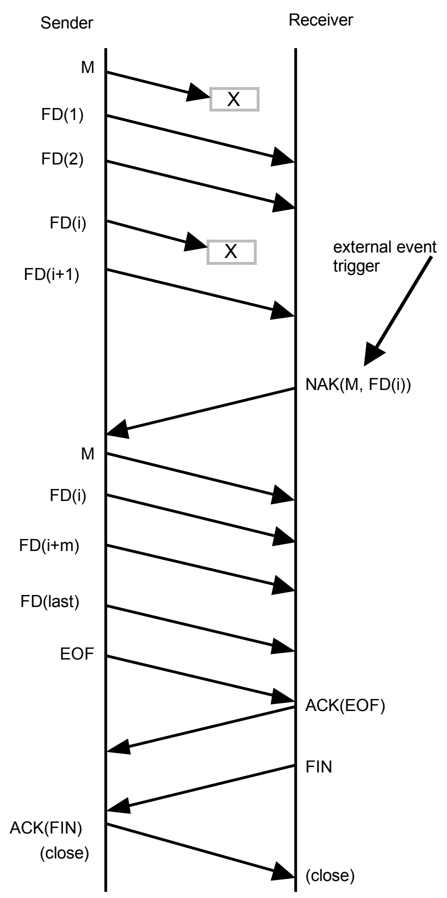

Figure 2-9: Asynchronous NAK Mode

2.6.3.2 Timers

Several timers are used in the reliable service processes. In each case in which a time-out capability is required, a timer is started upon issuance of the item. Upon receipt of the required response, the timer is disabled. If the required response is not received before the timer expires, the item is reissued. A count of the number of retransmissions is kept. If the preset limit of retransmissions is exceeded, a fault is declared.

In reliable service, within the file copying process timers are invoked for the EOF, the EOF-triggered NAK, and Finished (FIN) transmissions.

The operation of the NAK time-out is illustrated in figure 2-10. A NAK timer is started upon the issuance of the EOF-triggered NAK (which requests [re]transmission of all file data not yet received). It should be noted that previous individual NAKs are not acknowledged. When the timer expires, the receiving entity again determines whether or not any of the transaction's file data or metadata have yet to be received. If any file data gaps or missing metadata remain, normally, a NAK is issued, and the timer is reset.

The operation of the end-of-file and finished time-outs is shown in figure 2-11, parts (a) and (b), respectively.

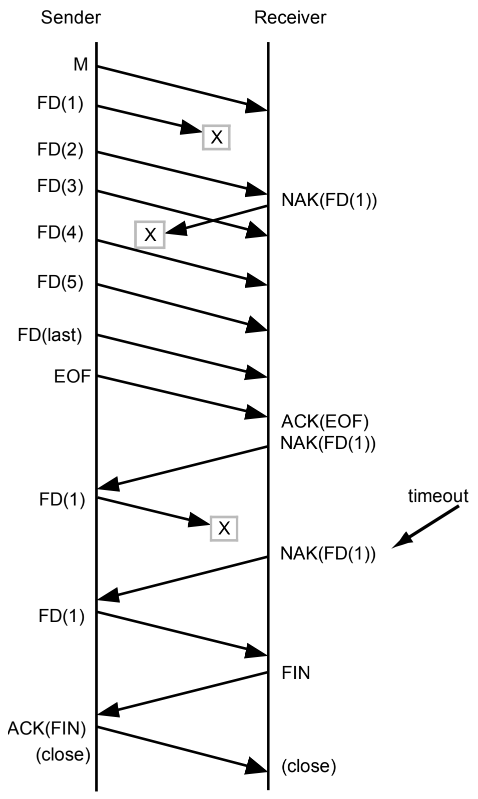

Figure 2-10: Time-out Triggered NAK Retransmission

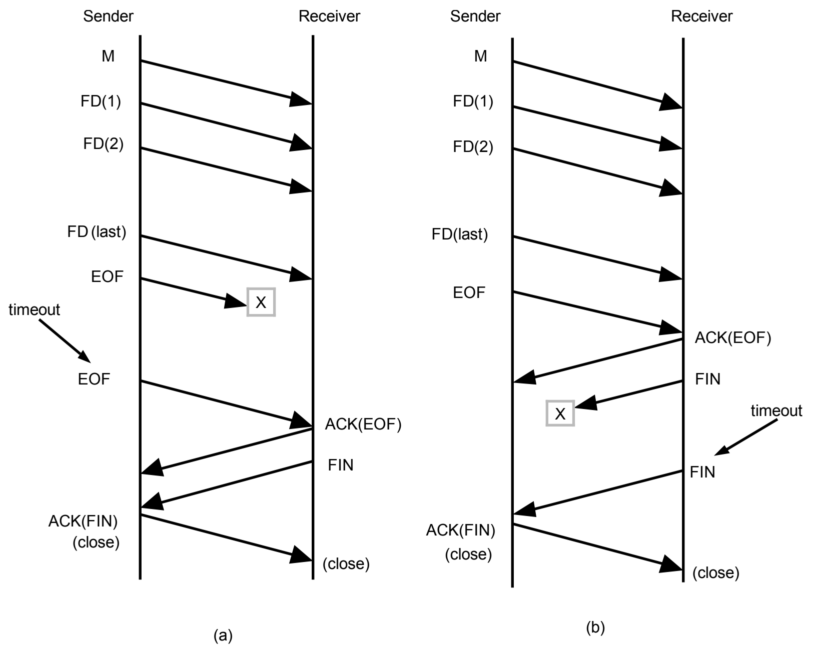

Figure 2-11: Time-out Triggered EOF and Finished Retransmissions

In addition to the time-outs just described, there is an overall inactivity timer. If there is a cessation of PDU reception for a given Transaction for the specified time period, this timer causes the issuance of a Fault indication, with a condition code identifying the condition ‘Inactivity’ to the local user. It takes no other action. That is, any further resulting action is not specified by the protocol, but is taken by the user in an implementation-specific manner. The user might try to restart or otherwise salvage the Transaction, abandon it, or take some other action appropriate to the implementation and operational configuration. The basic purpose of the timer is to handle situations that are outside the protocol itself. Examples might be the crashing of the operating system in the other entity party to the ongoing Transaction or the extended failure of a communication link. This timer is mandatory, except that it is not used at the Source end of an unacknowledged mode transfer.

2.7 PRIMITIVES, PDUS, AND PIPES

The ‘spawning’ relationships between Request Primitives and PDUs, and between PDUs and Indication Primitives in the operational process from initiation through termination, are shown in figure 2-12. Figure 2-13 is a ‘pipe’ diagram of the CFDP showing several of the possible lower layers. The examples of lower layers shown in the figure are just that, examples, and are not intended to be all-inclusive.

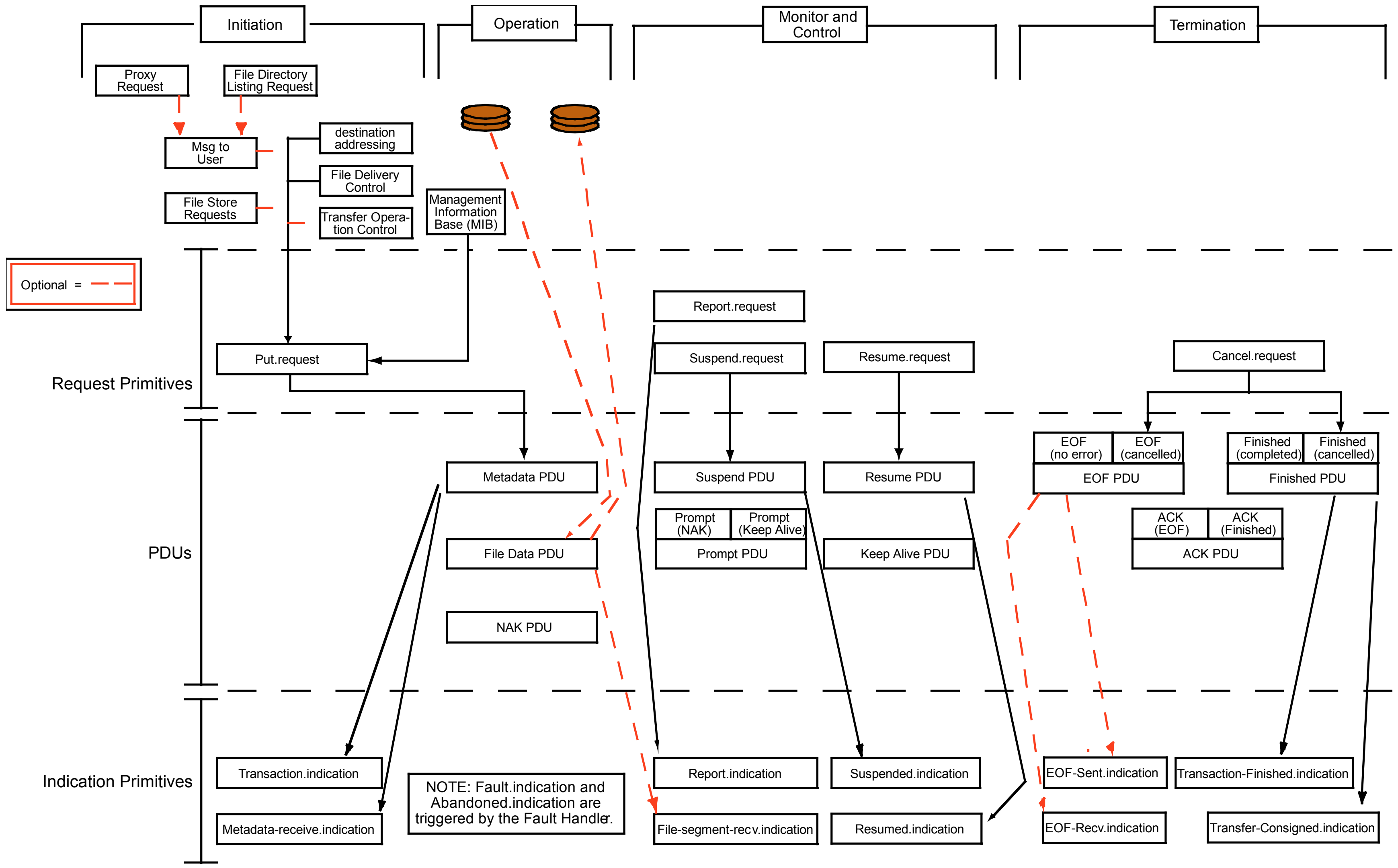

Figure 2-12: Request Primitives, PDUs, and Indication Primitives—Operational View

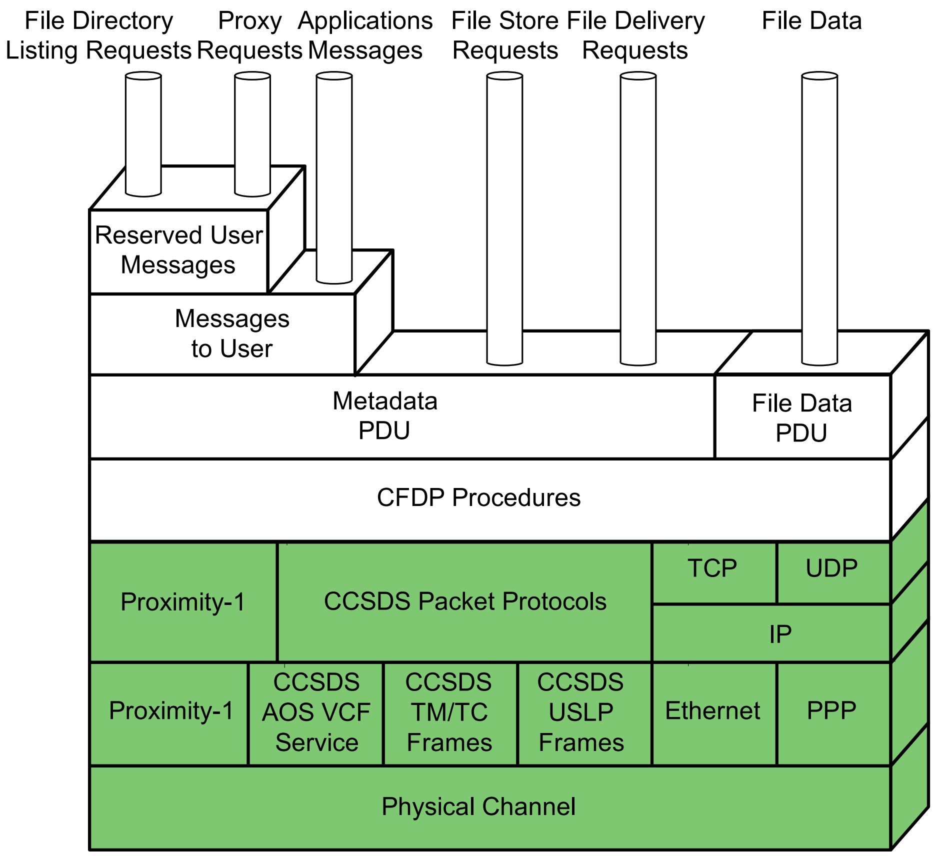

Figure 2-13: CFDP Pipe Diagram

3 EXAMPLE CONFIGURATIONS

3.1 OVERVIEW

This section provides examples that illustrate some of the widely varying network topologies over which the CFDP will operate, and the use of some of the CFDP options.

The entire purpose of the CFDP is to perform a file transfer between two end points. These end points may be located in Earth-orbiting spacecraft, mission control centers, or interplanetary spacecraft.

The following example mission configurations are included:

- file deliveries involving only two parties;
- file deliveries using three party proxy.

Ground station handover management is required any time a file transfer cannot be completed within a single pass and must be resumed later over the same or a different ground station. CFDP contains basic concepts (transaction ID) and services (Suspend/Resume and Link Lost/Link Acquired) to meet handover requirements in many situations. Although handover management is not a part of the protocol, and no specific handover management signaling takes place between CFDP entities, CFDP is well adapted to the automation of this process and therefore strongly supports station operations automation. System functions outside the CFDP may be used to implement fully autonomous handover operations.

3.2 FILE DELIVERIES

3.2.1 EXAMPLE CONFIGURATION 1: FILE DELIVERIES INVOLVING ONLY TWO PARTIES

3.2.1.1 General

In Example Configuration 1, a file transfer takes place between two CFDP entities, a spacecraft and an NCC. One or more ground stations handle frames and route data between the spacecraft and NCC. The ground stations do not contain CFDP entities and therefore are relays, and there is a (functional) point-to-point connection between the two CFDP entities. Examples are given for unreliable space-to-ground, reliable space-to-ground, reliable ground-to-space, and two-party proxy (Get). Three party proxy file deliveries are shown in Example Configuration 2.

When only one ground station is used (figure 3-1) and the file being delivered is large, multiple space-to-ground contacts (passes) with the ground station may be required in order to complete the delivery of the subject file.

If more than one ground station is used (figure 3-2) there may be redundant data (if multiple space-to-ground contacts are required, the ground station contacts may overlap in time). The

CFDP service at each endpoint deletes any duplicate data that it received because of overlapping contacts.

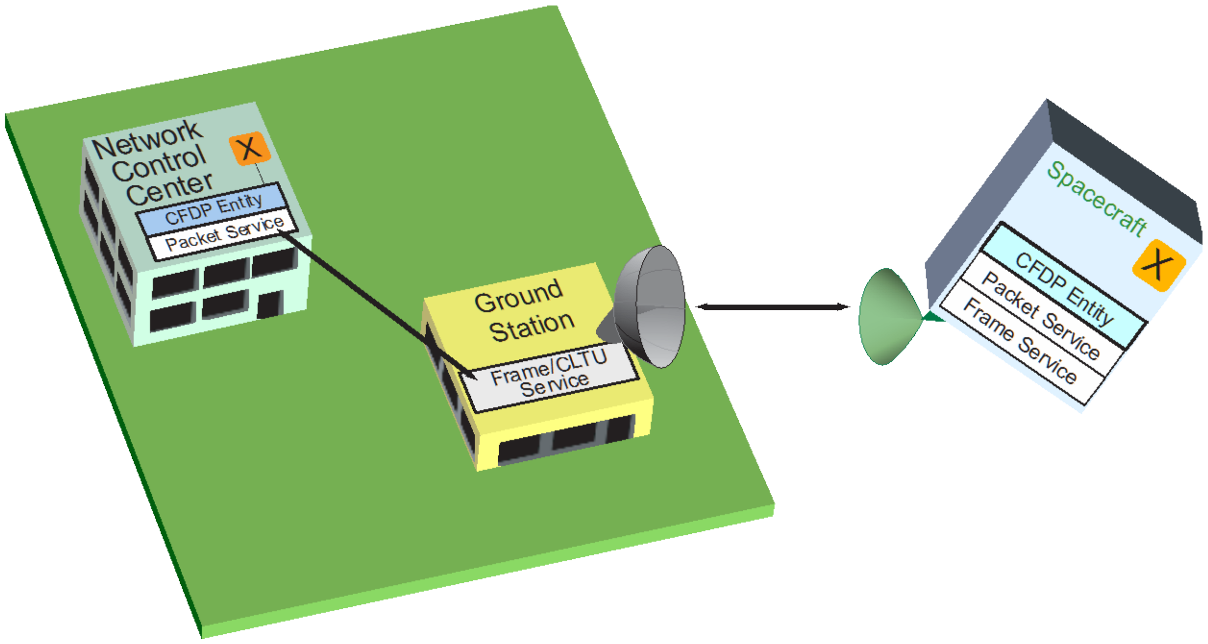

Figure 3-1: Example Configuration 1

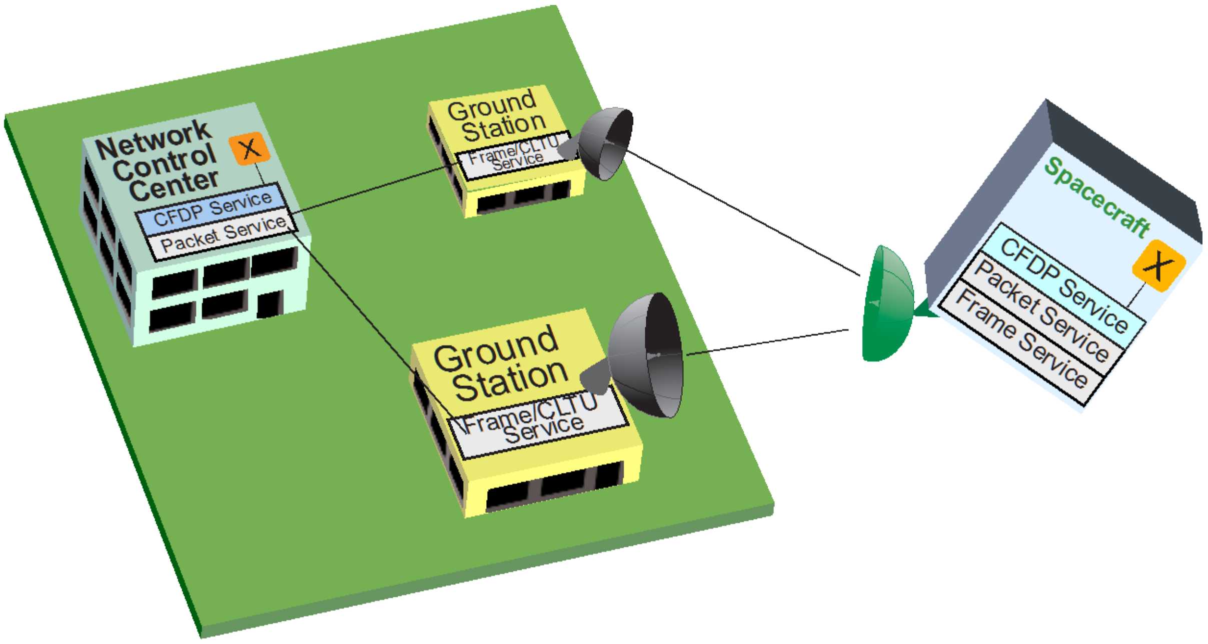

Figure 3-2: Example Configuration 1a

3.2.1.2 Unreliable Download

Example 1 of Configuration 1 is of an unreliable file delivery from a spacecraft to an NCC. The network configuration is directly from the spacecraft through ground stations to the NCC, as shown in figures 3-1 and 3-2. The file size may be large enough to require more than one ground/spacecraft contact period. The ground station contacts may overlap in time, or may be time disjoint.

A user (human or automated) on the spacecraft initiates the transaction by causing a Put request to be sent to the local spacecraft CFDP entity.

Upon receipt of the Put request, the spacecraft CFDP entity initiates the transaction. It configures the protocol options (e.g., quality of service) according to the information contained in the MIB, unless overridden by information in the Put request. The CFDP entity places the required information in the file metadata and begins the file delivery operation. Each item that the CFDP entity wishes transmitted is placed in a PDU, file data segment metadata may optionally be added, and the item is passed to the lower layer network. In this example, the interface is to CCSDS Path Service. The Path Service places the PDUs within CCSDS packets, virtual channels, and frames and transmits them to the ground station(s). The ground station(s) synchronizes on the frames, optionally performs error correction, and routes them to the NCC. At the NCC, the NCC CCSDS packet service extracts the packets from the frames, the PDUs from the packets, and passes the PDUs to the NCC CFDP service for action and file assembly. (Alternatively, the packet extraction process could be accomplished by CCSDS services at the ground stations, with the extracted packets being sent to the NCC CFDP entity via CCSDS Path Service.) Any duplicate PDUs caused by overlapping ground station contacts are removed by the CFDP entity. Because unreliable service has been selected, there is no CFDP traffic from the NCC to the spacecraft. Details of the CFDP PDUs, their contents and formats, and the manner in which they operate are detailed in reference [1].

The sequence of events is shown in figure 3-3. It should be noted that since this example is for unreliable quality of service, the loss of File Data PDU ‘N+1’ is irrecoverable. One should also note that an optional File Completion Map, which shows the size and exact location within the file of the missing data, is available to the NCC User. (In all the sequence diagrams, optional items are shown in red.) This mode may be particularly useful for the transmission of image data, etc., when file sizes tend to be large and absolute completeness may not be necessary.

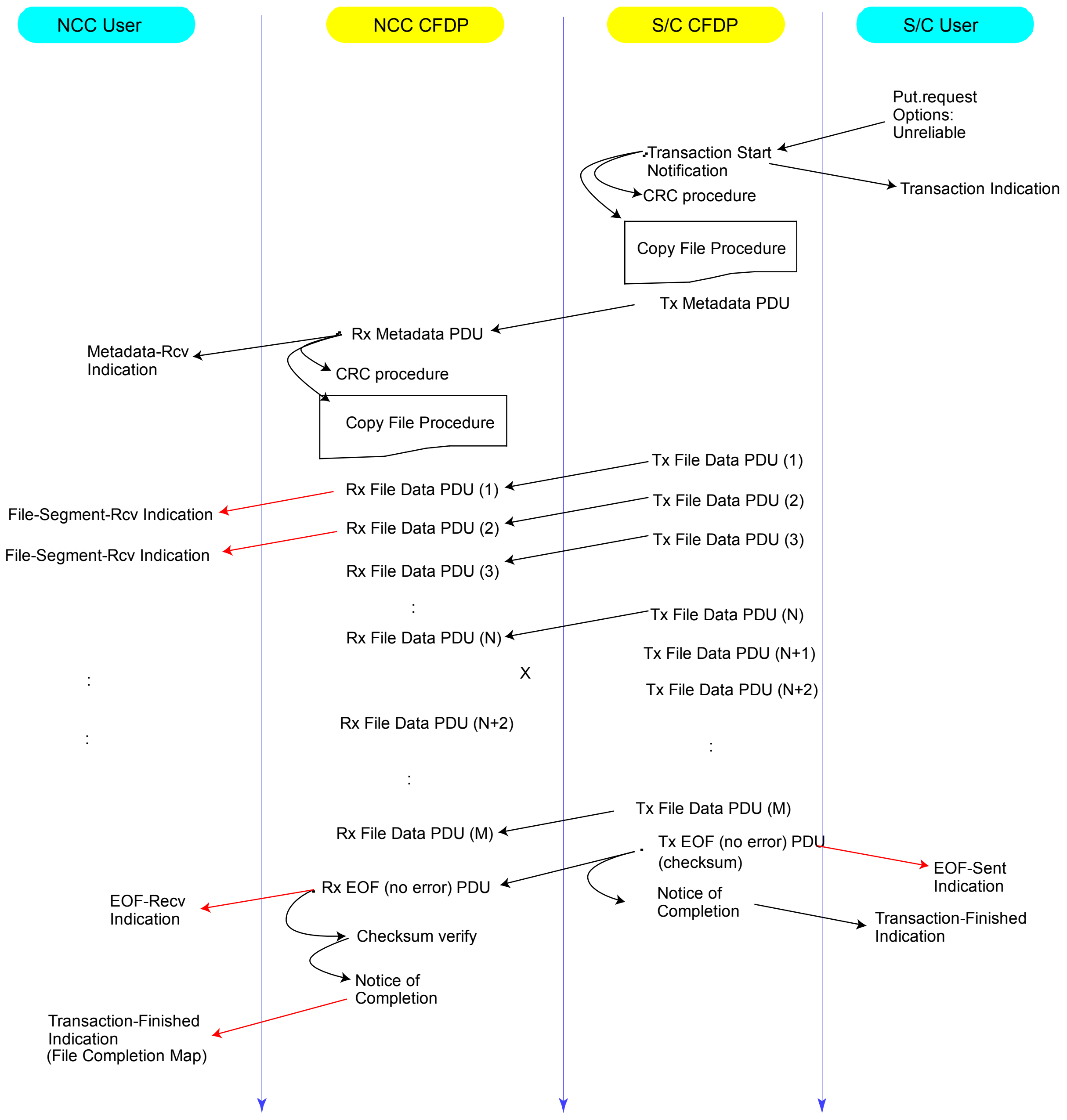

Figure 3-3: Unreliable Download

3.2.1.3 Unreliable Upload with Closure Requested

The second example of Configuration 1 is of an NCC-initiated reliable delivery, from the control center to a spacecraft, of a command sequence file. The network configuration is directly from the NCC through ground stations to the spacecraft as shown in figures 3-1 and 3-2. The spacecraft is in cruise configuration in an interplanetary transfer trajectory, and because of power restrictions in cruise mode, the downlink transmitter will not be turned on until late in the contact period. The file size is small enough to require only a single ground/spacecraft contact period.

Unacknowledged CFDP service is selected because the UT-layer service is itself reliable (e.g., the UT-layer service might be the Delay-Tolerant Networking stack). Upon receipt of the file, the receiving entity will use a Finish PDU to notify the sender that the file has arrived.

A user (human or automated) in the NCC initiates the transaction by causing a Put request to be sent to the local NCC CFDP entity.

Upon receipt of the Put request, the NCC CFDP entity initiates the transaction. It configures the protocol options according to the information contained in the MIB, unless overridden by information in the Put request. It then begins the file delivery operation. Each item that the CFDP entity wishes transmitted is placed in a PDU and passed to the (reliable) lower layer network.

The sequence of events is shown in figure 3-4. It should be noted that since this example is for a reliable UT service, the lost File Data PDU 'N+1' is recovered by the UT-layer protocol.

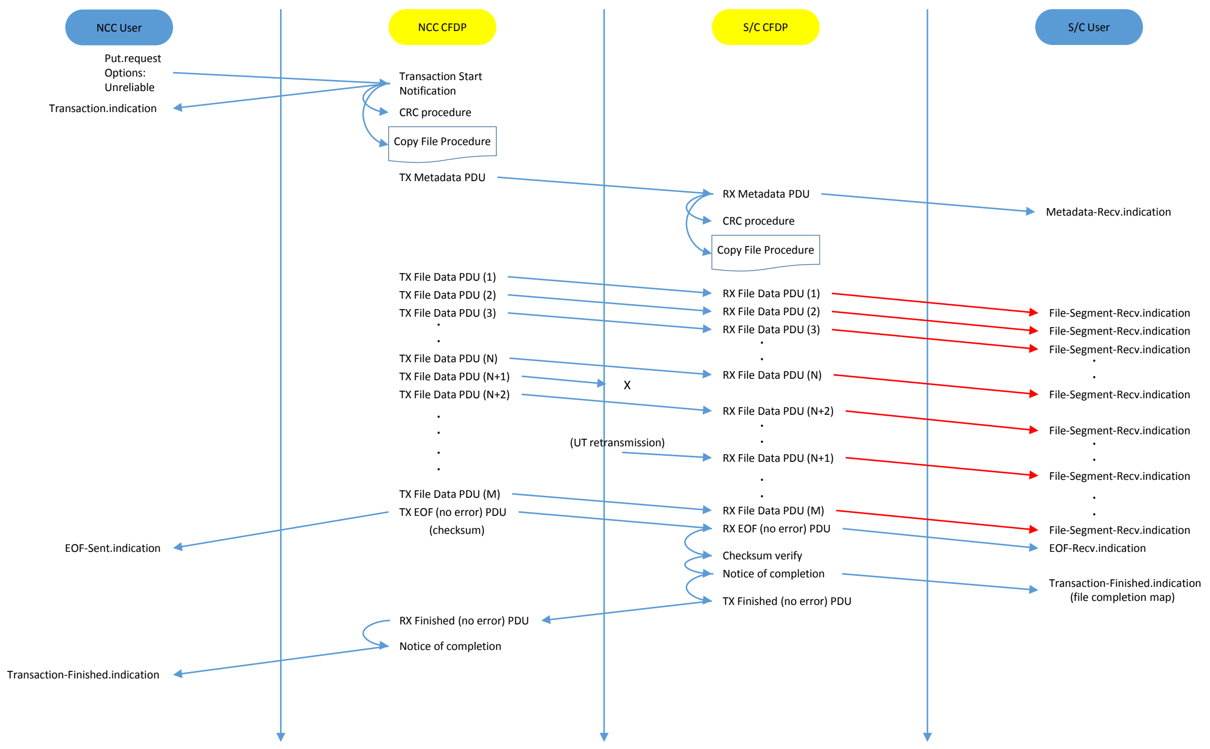

Figure 3-4: Unreliable Upload with Closure Requested

3.2.1.4 Reliable Download

Example 3 of Configuration 1 is of a reliable delivery of a file from a spacecraft to an NCC. The network configuration is directly from the spacecraft through ground stations to the NCC, as shown in figures 3-1 and 3-2. The spacecraft is in low Earth orbit, and the contact will have simultaneous uplink and downlink. The file size may be large enough to require more than one ground/spacecraft contact period. The ground station contacts may overlap in time or may be time disjoint. The following options could be selected to meet the requirements:

– quality of service—reliable;

– NAK mode—immediate.

A user (human or automated) on the spacecraft initiates the transaction by causing a Put request to be sent to the local spacecraft CFDP entity.

Upon receipt of the Put request, the spacecraft CFDP entity initiates the transaction. It configures the protocol options (e.g., quality of service, NAK mode) according to the information contained in the MIB, unless overridden by information in the Put request. It places the required information in the file metadata and begins the file delivery operation. Each item that the spacecraft entity wishes transmitted is placed in a PDU and passed to the lower-layer network. In this case, the interface is to CCSDS Path Service. The operation through the Path Service has been described previously (3.2.1.2).

The sequence of events is shown in figure 3-5. It should be noted that since this example is for reliable quality of service, the lost File Data PDU 'N+1' is recovered. One should also note that as the NAK mode is Immediate, when File Data PDU 'N+2' is received, and it is therefore known that N+1 was missed, the NAK requesting retransmission of N+1 is sent immediately. Finally, it should be noted that an optional File Completion Map, which shows the size and exact location within the file of any missing data, is still available to the NCC User, even though, since this is a reliable transfer, there should be no missing data. This is because in the event of a protocol error that causes cancellation of the remaining file delivery operation, it may be desirable to retain that portion of the file that has been received. In that situation, the file completion map can be important. If a CFDP implementation includes this option, it will provide the file completion map as part of the status message returned with the Transaction-Finished indication.

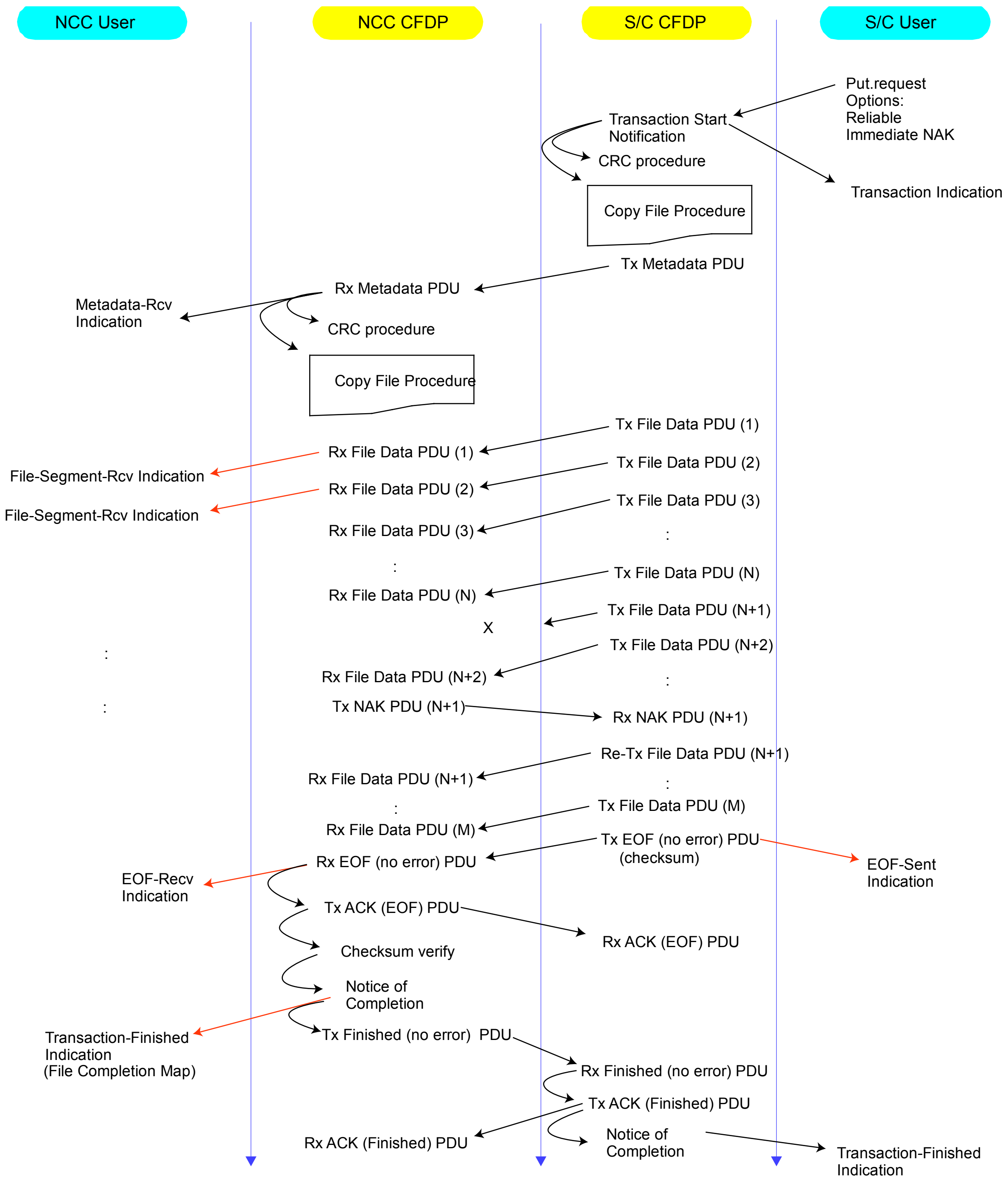

Figure 3-5: Reliable Download

3.2.1.5 Reliable Upload

The fourth example of Configuration 1 is of an NCC-initiated reliable delivery, from the control center to a spacecraft, of a file constituting a new star catalogue for use by star trackers of the onboard navigation system. The network configuration is directly from the NCC through ground stations to the spacecraft as shown in figures 3-1 and 3-2. The spacecraft is in cruise configuration in an interplanetary transfer trajectory and, because of power restrictions in cruise mode, the downlink transmitter will not be turned on until late in the contact period. The file size is small enough to require only a single ground/spacecraft contact period. It is desired to include a message to the onboard navigation system to switch from the old to the new star catalogue during the first slew maneuver, after successful receipt of the new star catalogue.

NOTE – The main purpose of the CFDP's inclusion of the ability to carry a 'message to user' in the metadata of the file being transferred is to synchronize utilization of that message with delivery of the file. In this example, if the command to switch to the new star catalogue was sent in a separate transmission, it might arrive either before or long after the map itself, depending on packet loss and retransmission effects. However, if the message is attached to the file itself, then synchronized receipt of the message is implicit. Also, the CFDP has the capability of carrying file store directives in specially marked versions of message to users, thus assuring that the file store directives will be synchronized with the file to which they are related, if so desired.

The following options could be selected to meet the requirements:

– message to user—transmitted in the Put request;

– quality of service—reliable;

– NAK mode—asynchronous.

The asynchronous NAK mode is selected, since a downlink will not be available until late in the contact period. By using the asynchronous mode, the bulk of the file can be delivered before the downlink is available. Then when the downlink transmitter is turned on, only the transmission of any necessary NAK and the resulting data retransmissions, as well as the transaction close-out actions, remain to be accomplished. (The deferred NAK mode might be a better solution, but this example is used to illustrate the function of the Asynchronous NAK mode.)

A user (human or automated) in the NCC initiates the transaction by causing a Put request to be sent to the local NCC CFDP entity. The message to user to be sent with the file is contained within the request.

Upon receipt of the Put request, the NCC CFDP entity initiates the transaction. It configures the protocol options (e.g., quality of service, NAK mode) according to the information contained in the MIB, unless overridden by information in the Put request. It places the message to user described above in the file metadata and begins the file delivery operation.

Each item that the CFDP entity wishes transmitted is placed in a PDU and passed to the lower layer network. In this case, the interface is to CCSDS Path Service. The operation through the Path Service has been described previously (3.2.1.2).

The sequence of events is shown in figure 3-6. It should be noted that since this example is for reliable quality of service, the lost File Data PDU ‘N+1’ is recovered. One should also note that as the NAK mode is Asynchronous, when File Data PDU ‘N+2’ is received, and it is therefore known that N+1 was missed, the NAK requesting retransmission of N+1 is not sent immediately, but is held until an external event (establishment of downlink connectivity) triggers its transmission. Finally, it should be noted that an optional File Completion Map, which shows the size and exact location within the file of any missing data, is available to the Spacecraft User even though, since this is a reliable transfer, there should be no missing data. This is because in the event of a protocol error that causes cancellation of the remaining file delivery operation, it may be desirable to retain that portion of the file that has been received. In that situation, the file completion map can be important. If a CFDP implementation includes this option, it will provide the file completion map as part of the status message returned with the Transaction-Finished indication.

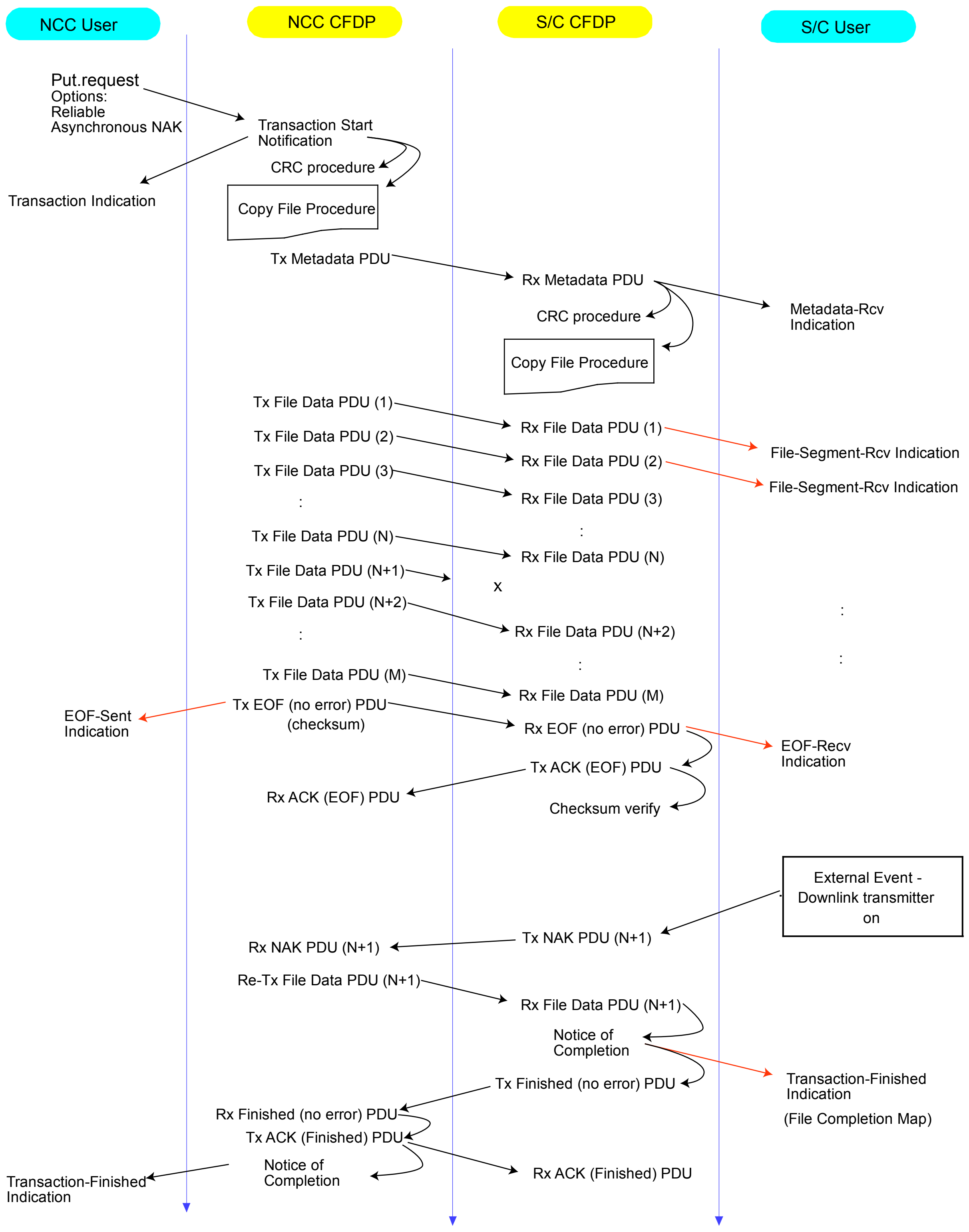

Figure 3-6: Reliable Upload

3.2.1.6 Two Party Proxy (Get)

The CFDP provides a functional 'Get' capability through the use of a proxy operation. It effects this through the use of a 'Proxy Put Request'. This example is that of an NCC initiating a reliable delivery of a file from a spacecraft to the NCC (a functional 'Get'). The network configuration is directly from the spacecraft through ground stations to the NCC, as shown in figures 3-1 and 3-2. The spacecraft is in low Earth orbit, and the contact will have simultaneous uplink and downlink. The following options could be selected to meet the requirements:

– transaction type—Proxy Put;

– quality of service—reliable;

– NAK mode—immediate.

The sequence of events is shown in figure 3-7. The user at the NCC initiates the transaction by inputting a Put.request primitive to the local NCC CFDP entity. The Put.request primitive contains a Proxy Put Request message, and the options selection shown above. The resulting Put.request PDU is sent to the spacecraft CFDP entity as a single transaction, called Transaction 'X' in figure 3-7. This transaction contains only metadata (containing the specially marked message to user that in turn contains the Proxy Put Request), and no file data. Upon receipt of the Proxy Put Request, the spacecraft CFDP entity user initiates a Put file delivery transaction from itself to the NCC (Transaction 'Y'), using the parameters sent by the NCC. The spacecraft/NCC transaction is a normal Put transaction, except that when the Transaction-Finished indication for Transaction 'Y' is received by the spacecraft user, it causes a completion notification to be sent (as Transaction 'Z') back to the NCC CFDP entity, informing it that the proxy file delivery it requested in Transaction 'X' has been completed.

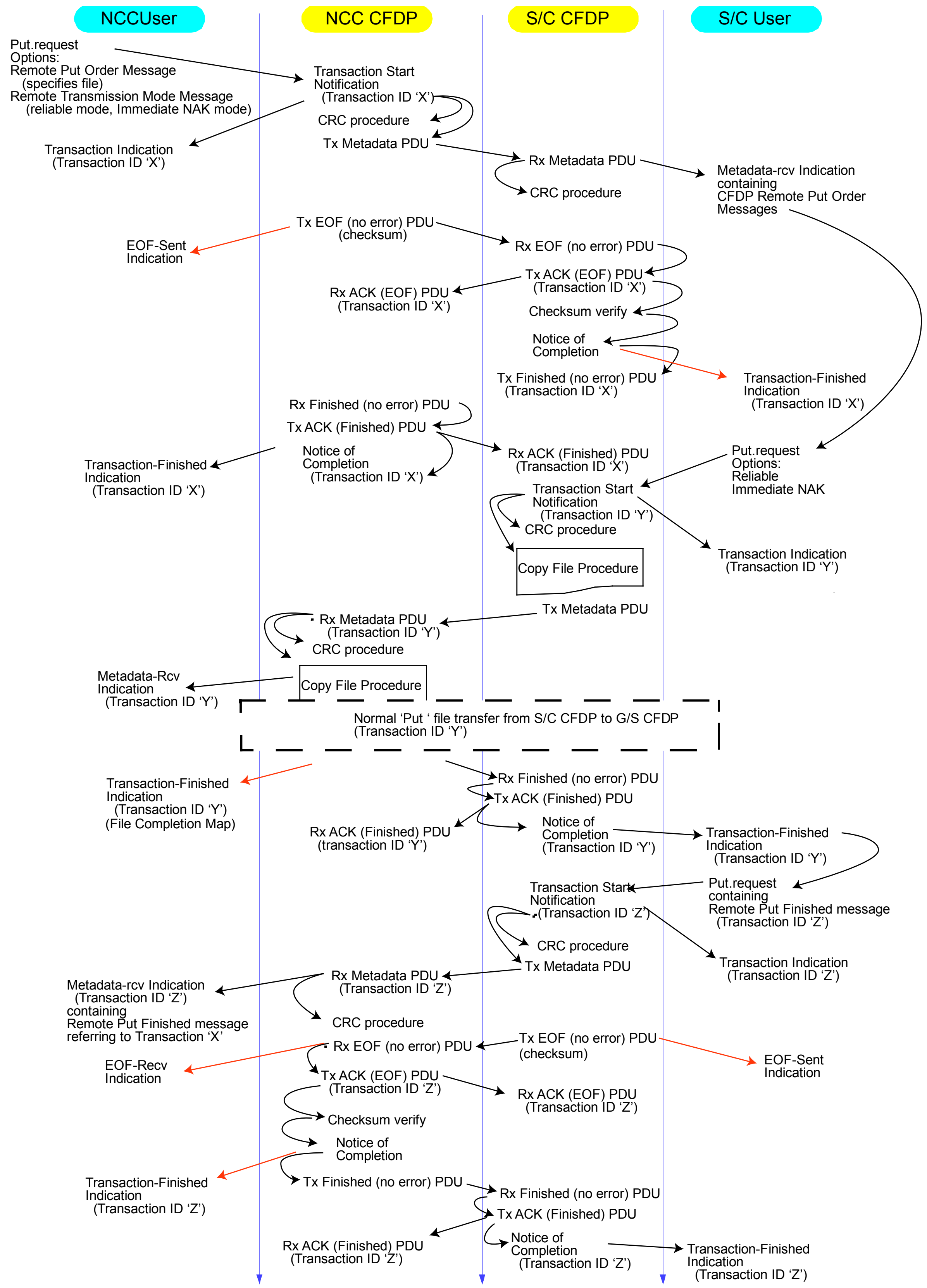
NCCUser
NCC CFDP
S/C CFDP
S/C User
Put.request Options: Remote Put Order Message (specifies file) Remote Transmission Mode Message (reliable mode, Immediate NAK mode)
Transaction Start Notification (Transaction ID 'X')
CRC procedure
Tx Metadata PDU
Transaction Indication (Transaction ID 'X')
Rx Metadata PDU
Metadata-rcv Indication containing CFDP Remote Put Order Messages
CRC procedure
Tx EOF (no error) PDU (checksum)
EOF-Sent Indication
Rx EOF (no error) PDU
Tx ACK (EOF) PDU (Transaction ID 'X')
Rx ACK (EOF) PDU (Transaction ID 'X')
Checksum verify
Notice of Completion
Tx Finished (no error) PDU (Transaction ID 'X')
Transaction-Finished Indication (Transaction ID 'X')
Rx Finished (no error) PDU
Tx ACK (Finished) PDU
Notice of Completion (Transaction ID 'X')
Transaction-Finished Indication (Transaction ID 'X')
Rx ACK (Finished) PDU (Transaction ID 'X')
Put.request Options: Reliable Immediate NAK
Transaction Start Notification (Transaction ID 'Y')
CRC procedure
Transaction Indication (Transaction ID 'Y')
Copy File Procedure
Tx Metadata PDU
Rx Metadata PDU (Transaction ID 'Y')
CRC procedure
Metadata-Rcv Indication (Transaction ID 'Y')
Copy File Procedure
Normal 'Put ' file transfer from S/C CFDP to G/S CFDP (Transaction ID 'Y')
Transaction-Finished Indication (Transaction ID 'Y') (File Completion Map)
Rx Finished (no error) PDU
Tx ACK (Finished) PDU
Rx ACK (Finished) PDU (transaction ID 'Y')
Notice of Completion (Transaction ID 'Y')
Transaction-Finished Indication (Transaction ID 'Y')
Transaction Start Notification (Transaction ID 'Z')
Put.request containing Remote Put Finished message (Transaction ID 'Z')
CRC procedure
Tx Metadata PDU
Transaction Indication (Transaction ID 'Z')
Metadata-rcv Indication (Transaction ID 'Z') containing Remote Put Finished message referring to Transaction 'X'
Rx Metadata PDU (Transaction ID 'Z')
CRC procedure
Rx EOF (no error) PDU
Tx EOF (no error) PDU (checksum)
EOF-Sent Indication
EOF-Recv Indication
Tx ACK (EOF) PDU (Transaction ID 'Z')
Rx ACK (EOF) PDU (Transaction ID 'Z')
Checksum verify
Notice of Completion
Transaction-Finished Indication (Transaction ID 'Z')
Tx Finished (no error) PDU
Rx Finished (no error) PDU
Tx ACK (Finished) PDU
Rx ACK (Finished) PDU (Transaction ID 'Z')
Notice of Completion (Transaction ID 'Z')
Transaction-Finished Indication (Transaction ID 'Z')

Figure 3-7: Two Party Proxy (Get)

3.2.2 EXAMPLE CONFIGURATION 2: THREE PARTY PROXY

The next example is that of a user (perhaps notified by telephone of a spacecraft emergency) requesting an NCC to initiate reliable delivery of an instrument contingency operations file *resident at the NCC* to a spacecraft. The user is remote from his/her home facility and uses a laptop computer, which contains a CFDP entity. The network configuration is from the user's laptop (via the Internet), to the NCC, to a ground station, to the spacecraft, as shown in figure 3-8. The spacecraft is in low Earth orbit, and the NCC/spacecraft contact will have simultaneous uplink and downlink. The file size is small enough that it can be completely delivered within one ground/spacecraft contact period.

The following options could be selected to meet the requirements:

– transaction type—Proxy Put;

– quality of service—reliable;

– NAK mode—immediate.

The sequence of events is shown in figure 3-9. The remote user initiates the transaction by inputting a Put.request primitive to his/her local laptop CFDP entity. The Put.request primitive contains the address of the NCC CFDP entity, a Proxy Put Request message, and the options selection shown above. The resulting Put.request PDU is sent to the NCC CFDP entity, via the Internet, as a single transaction, called Transaction 'X' in figure 3-9. This transaction contains only metadata (containing the specially marked message to user that in turn contains the Proxy Put Request), and no file data. Upon receipt of the Proxy Put Request, the NCC CFDP entity user initiates a Put file delivery transaction from itself to the spacecraft (Transaction 'Y'), using the parameters sent by the remote User. The NCC/spacecraft transaction is a normal Put transaction, except that when the Transaction-Finished indication for Transaction 'Y' is received by the NCC user, it causes a completion notification to be sent (as Transaction 'Z') back to the Remote User's CFDP entity, informing it that the proxy file delivery it requested in Transaction 'X' has been completed.

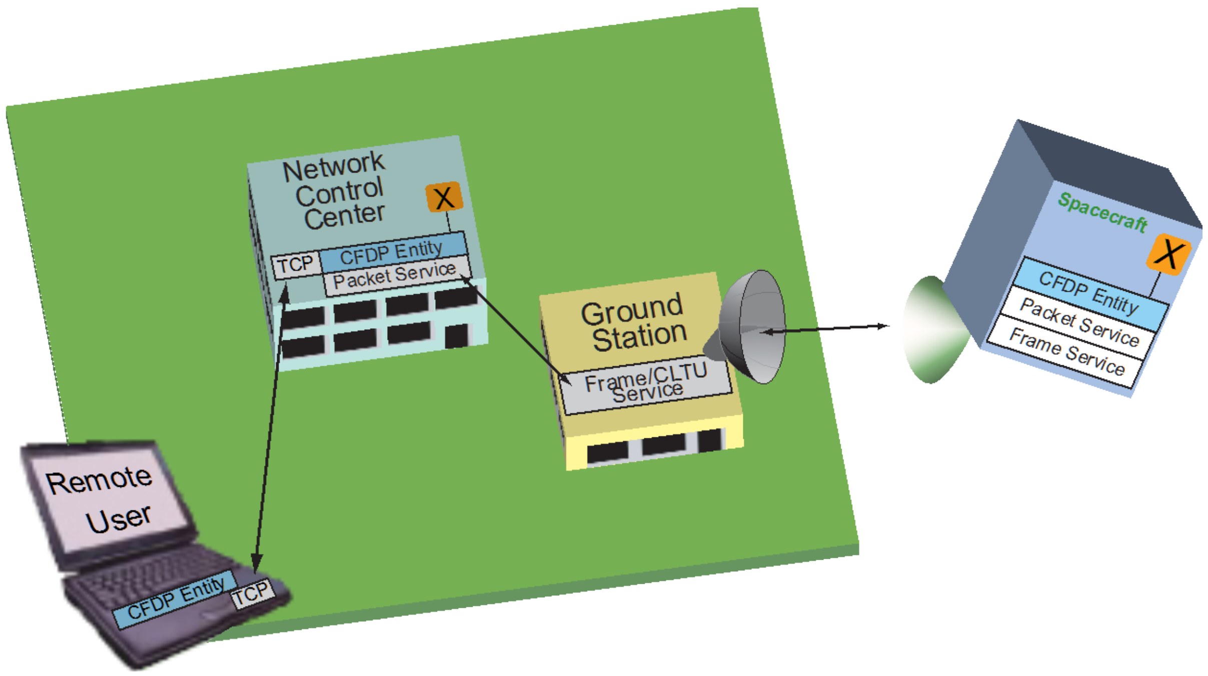

Figure 3-8: Example Mission Configuration 2 (Three Party Proxy)

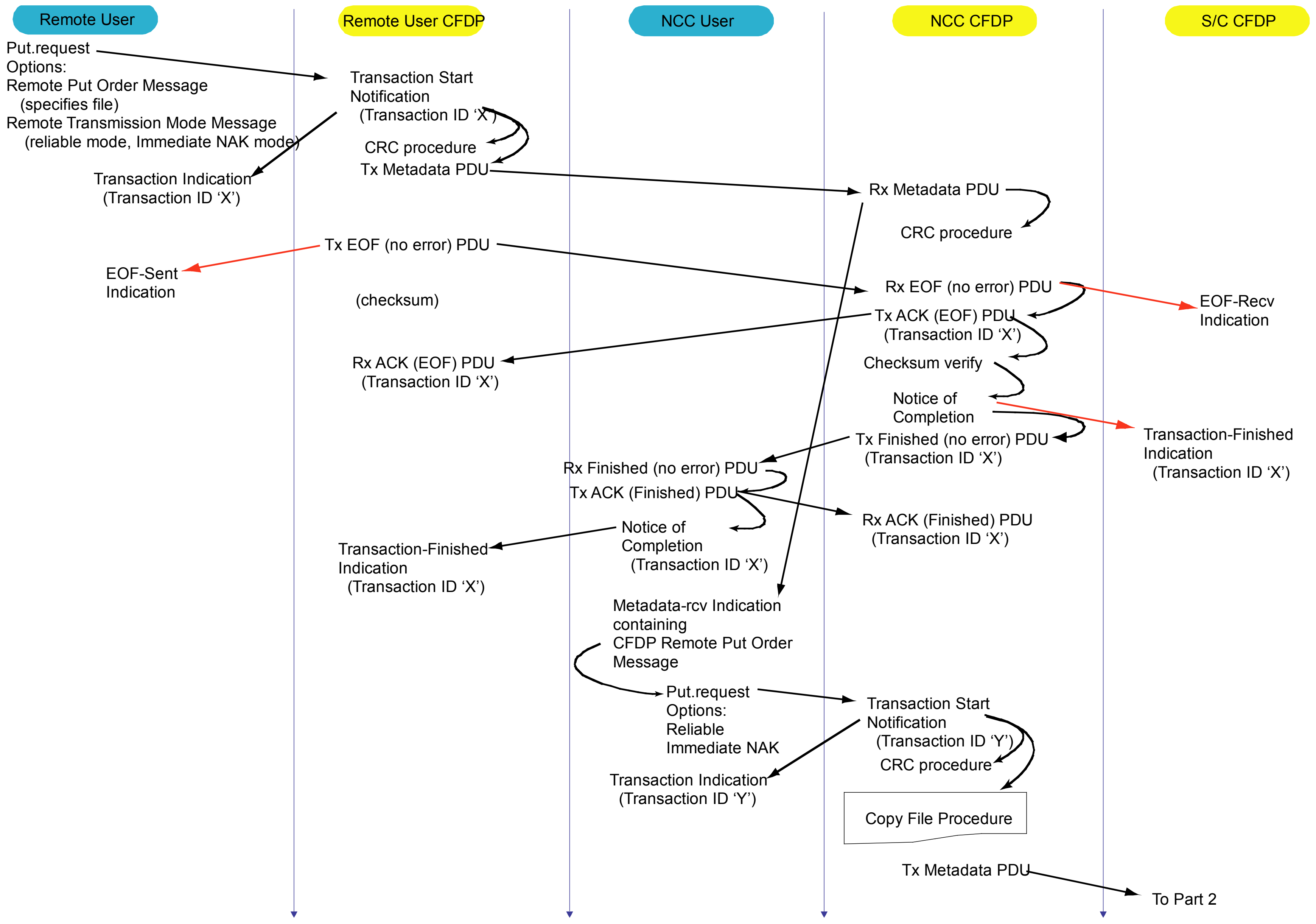

Figure 3-9: Three Party Proxy (Part 1)

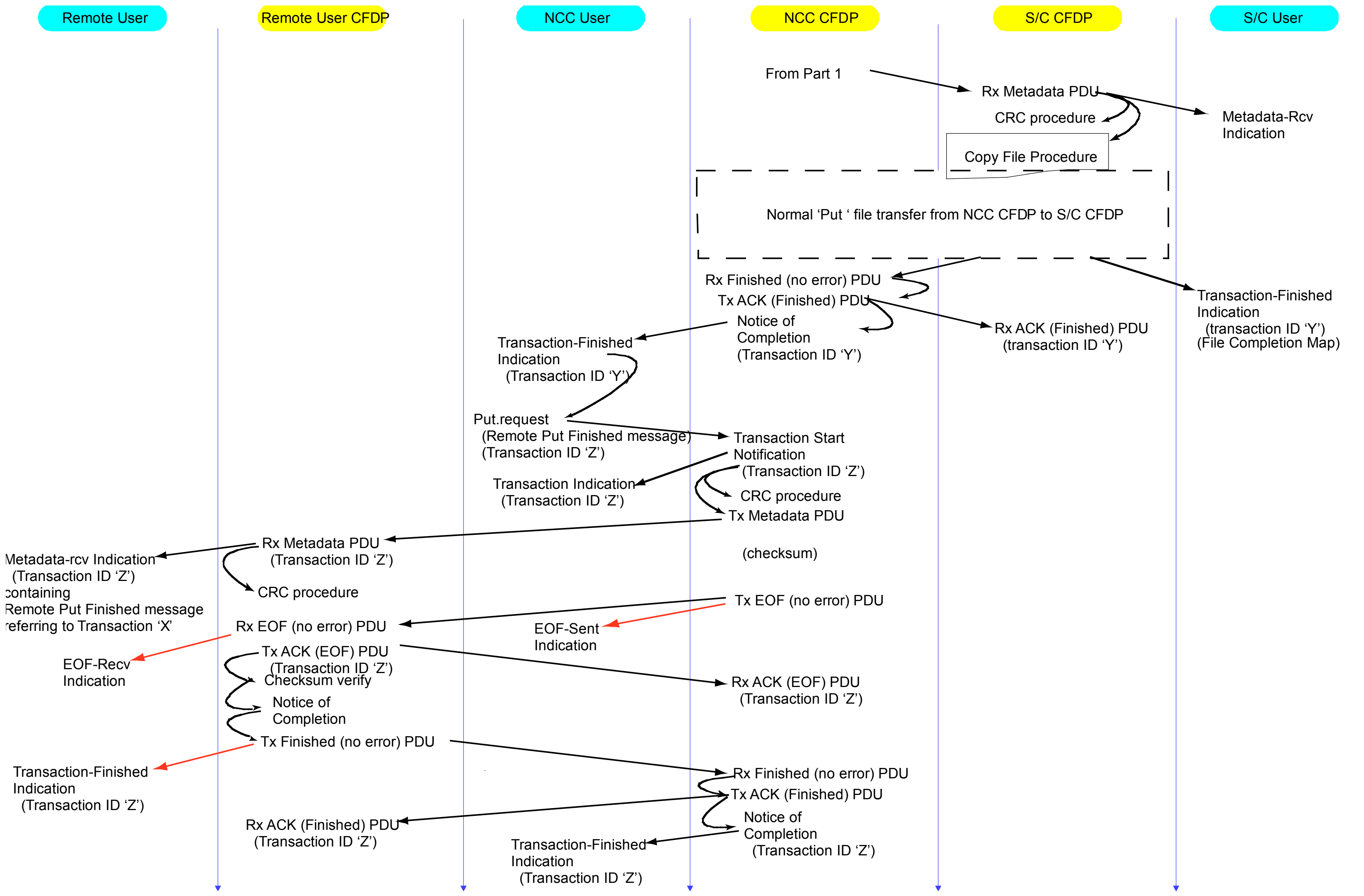
Remote User
Remote User CFDP
NCC User
NCC CFDP
S/C CFDP
S/C User
From Part 1
Rx Metadata PDU
CRC procedure
Metadata-Rcv Indication
Copy File Procedure
Normal 'Put ' file transfer from NCC CFDP to S/C CFDP
Rx Finished (no error) PDU
Tx ACK (Finished) PDU
Notice of Completion (Transaction ID 'Y')
Rx ACK (Finished) PDU (transaction ID 'Y')
Transaction-Finished Indication (transaction ID 'Y') (File Completion Map)
Transaction-Finished Indication (Transaction ID 'Y')
Put.request (Remote Put Finished message) (Transaction ID 'Z')
Transaction Start Notification (Transaction ID 'Z')
Transaction Indication (Transaction ID 'Z')
CRC procedure
Tx Metadata PDU
(checksum)
Rx Metadata PDU (Transaction ID 'Z')
Metadata-rcv Indication (Transaction ID 'Z') containing Remote Put Finished message referring to Transaction 'X'
CRC procedure
Tx EOF (no error) PDU
EOF-Sent Indication
Rx EOF (no error) PDU
EOF-Recv Indication
Tx ACK (EOF) PDU (Transaction ID 'Z')
Checksum verify
Notice of Completion
Rx ACK (EOF) PDU (Transaction ID 'Z')
Tx Finished (no error) PDU
Transaction-Finished Indication (Transaction ID 'Z')
Rx Finished (no error) PDU
Tx ACK (Finished) PDU
Notice of Completion (Transaction ID 'Z')
Rx ACK (Finished) PDU (Transaction ID 'Z')
Transaction-Finished Indication (Transaction ID 'Z')

Figure 3-8: Three Party Proxy (Part 2)

ANNEX A

ABBREVIATIONS AND ACRONYMS

Term	Meaning
ACK	(positive) acknowledgment
AOS	Advanced Orbiting Systems
CCSDS	Consultative Committee for Space Data Systems
CFDP	CCSDS File Delivery Protocol
EOF	end of file
FD(n)	file data segment
FDU	file delivery unit
FIN	finished (receiver to sender)
LTP	Licklider Transmission Protocol
M	metadata
MIB	Management Information Base
MSB	most significant bit
NAK	negative acknowledgment
NCC	Network Control Center
OSI	Open Systems Interconnection
PDU	protocol data unit
PPP	point-to-point protocol
PRMPT	prompt
Rx	reception
SAR	synthetic aperture radar
TC	telecommand
TCP	Transmission Control Protocol
TM	telemetry
Tx	transmission
UDP	User Datagram Protocol
USLP	Unified Space Data Link Protocol
VCF	virtual channel frame

ANNEX B

MAJOR REVISIONS TO VERSION 5 BLUE BOOK

This annex enumerates the revisions made to the CFDP Blue Book version 5 in the five-year refresh cycle of 2020.

– Moves Store and Forward Overlay Operations from section 6 to a new normative annex (annex B).

– Removes Extended Procedures.

– Adds support for large files, multi-segment records, and per-segment metadata.

– Adds support for choice of checksum algorithm from a new SANA Checksum Types registry.

– Updates, clarifies some text based on interoperability testing.