FLYING CFDP ON MESSENGER

Christopher J. Krupiarz and Brian K. Heggestad
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Road, Laurel, MD USA 20723-6099
E-mail: Christopher.Krupiarz@jhuapl.edu, Brian.Heggestad@jhuapl.edu

Richard D. Carper
Consultant, Space Data Systems
3934 N.W. Sitka Place, Corvallis, OR USA 97330-3344
E-mail: richardcarper@comcast.net

ABSTRACT

The MESSENGER mission to Mercury will downlink data files via a protocol defined by the Consultative Committee for Space Data Systems (CCSDS) called the CCSDS File Delivery Protocol (CFDP). A reduced implementation of the protocol was developed for the spacecraft due to various system constraints and operational requirements. The software operates in conjunction with the playback features of the MESSENGER flight software allowing for the autonomous downlinking of files as well as providing for the management of the file system by the mission operations team. This paper presents the software implementation, metrics, and the lessons learned.

KEYWORDS

CCSDS, CFDP, MESSENGER, File Transfer

INTRODUCTION

MErcury, Surface, Space ENvironment, GEochemistry, and Ranging (MESSENGER) is a NASA Discovery mission to study the planet Mercury. MESSENGER was launched on August 3, 2004, and will include one flyby of Earth, two flybys of Venus, and three flybys of Mercury, followed by Mercury orbit insertion in March 2011 for a one-year science-gathering mission. Through all phases of the mission, MESSENGER will collect data from seven instruments on key characteristics of the planet to further understand Mercury and the formation of the inner solar system [1]. As part of the system software developed by The Johns Hopkins University Applied Physics Laboratory (JHU/APL), MESSENGER will reliably downlink files of instrument and spacecraft housekeeping data via a protocol defined by the Consultative Committee for Space Data Systems (CCSDS) called the CCSDS File Delivery Protocol (CFDP) [2].

A typical spacecraft computer for a science mission at JHU/APL consists of one or more microprocessors performing various operations for commanding and controlling the spacecraft along with a device for storing mission data. Examples of such systems are shown in Table 1, which provides an overview of past JHU/APL Command & Data Handling (C&DH) subsystems and the amount of storage available on their Solid State Recorders (SSRs). These SSRs consist of static random access memory (SRAM) or other types of volatile or nonvolatile memory to store science and spacecraft housekeeping data.

Table 1: Command and Data Handling Spacecraft Computers for JHU/APL Missions

Mission	CPU	Speed (MHz)	SSR (Gbits)
NEAR (1996)	RTX2010	6	1 & .5
ACE (1997)	RTX2010	6	1
TIMED (2001)	Mongoose	12	2.5
CONTOUR (2002)	Mongoose	12	5
MESSENGER (2004)[1]	RAD6000	25	8

[1]Processor performs both C&DH and G&C

In JHU/APL missions prior to MESSENGER, the SSR was treated as a raw device on which data were serially stored and then retrieved for transmissions to the ground. For example, for CONTOUR, all instrument data were formatted into CCSDS telemetry packets of one particular size and placed on the recorder. Upon playback, pointers were set to indicate from where on the SSR to begin reading and to end reading. The packets were then inserted four at a time into CCSDS transfer frames and telemetered to the ground. Upon receipt of the data, ground software reassembled the data, at which time it was determined whether data needed to be resent. With the MESSENGER mission, operational and system requirements are such that an alternate storage method using an on-board file system is necessary. This is due to two primary factors [3].

The first factor is that MESSENGER has severe constraints on its downlink capability as a result of the available downlink rates, the amount of downlink time, and the long round-trip light time. As such, information stored on the spacecraft needs to be prioritized by importance with some data, such as contingency engineering telemetry and even some science results, not being downlinked in the event the bandwidth is not available. In order to accomplish this using a raw device, a complex and proprietary partitioning system would have been required to allow the flight software or the mission operations team to select what data should be telemetered. With a file system, this process is simplified as the data can be stored by priority and only those high priority data that fit within the bandwidth constraints will be transmitted. The file system also provides simplified management of this information by deleting lower priority data which are no longer needed, thus allowing for space to be made available on the SSR.

The second factor is that most of the science data to be stored on the SSR are from the Mercury Dual Imaging System (MDIS). These are images taken by the instrument that the flight software may compress and then downlink as individual files. Again, a file system reduces the complexity by providing the ability to delineate individual variable length images and to also allow for prototyping of the image compression software on a desktop computer.

Besides enabling the ability to achieve mission requirements, the shift to a file system also allows operations personnel to clearly see what data is available on the flight system as well as providing a streamlined method for comparing what has been downlinked. With this new addition to the standard JHU/APL spacecraft software architecture, a method needed to be developed to downlink the files to the ground. This led to the inclusion of CFDP into the MESSENGER software architecture.

THE CCSDS FILE DELIVERY PROTOCOL

As part of the system engineering process, the needs for MESSENGER's file downlinking ability were examined to determine what would be expected from a file transfer protocol. Various requirements were determined such as the need to autonomously find errors in the downlinked data as well as the need to handle a round-trip light time of several minutes. Included in this review of methodologies was CFDP. However, upon the initial evaluation, the use of CFDP was rejected by the mission software engineers as it was determined to be still in its infancy and thus too risky to include on MESSENGER. Work was then performed by team members to develop a protocol to meet the mission needs. As this work developed and through interaction with members of other mission teams, a second look was given to CFDP. As it was readily apparent that the direction being taken by the "in-house" protocol was beginning to match in many ways with the now solidifying CFDP standard, the protocol was adopted for use on MESSENGER.

At its core, CFDP can be described as a File Transfer Protocol (FTP)-like mechanism for transferring files between two entities. However, unlike typical Internet file transfers from one place on the globe to another, this protocol enables transfer of files across interplanetary distances. It is specifically engineered for spacecraft operating outside of Earth orbit with long link delays > 10 seconds and asymmetric bandwidth. CFDP can be configured to transfer files from the ground to the spacecraft and from the spacecraft to the ground, as well as among configurations of a network of platforms such as a spacecraft constellation or a series of planetary landers. CFDP also enables the control of a distant filestore through the use of typical file operation commands such as delete, move, and copy.

Although a full description of the protocol is beyond the scope of this paper, it is useful to observe an example file transfer as outlined in Figure 1. This typical transaction has two CFDP entities: (1) a spacecraft and (2) a ground system. The file being transferred in this diagram is from the spacecraft to the ground with the transaction being initiated either remotely by the ground or locally by the spacecraft flight software. The lines between the two entities represent transfers of CFDP Protocol Data Units (PDUs) from MESSENGER to the Mission Control Center (MOC). PDUs can range from information about the file being transferred to actual pieces of the file itself. The diagram shows what occurs during nominal transmissions as well as anomalous operation, typically due to lost data. In this example, a method of negative acknowledgement (NAK) called Deferred NAKs is used. This means the ground will wait until the flight software has completed sending all pieces of the file before transmitting to the spacecraft what pieces were not received. Other NAK methods are available in CFDP as well as an Unacknowledged Mode where no NAKs are sent.

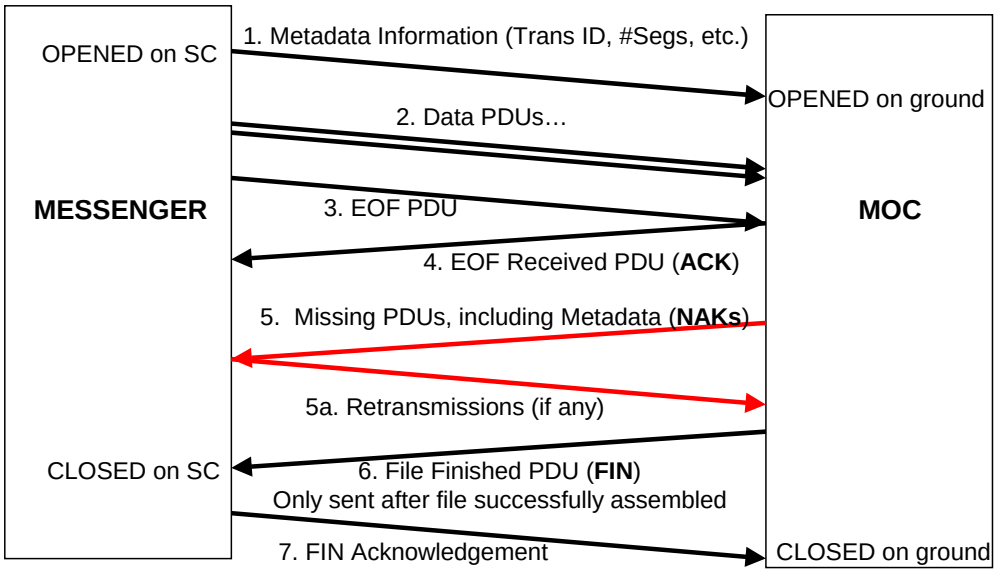

Figure 1: CFDP Transaction Example

The first step in the transmission of the file is the creation of a Metadata PDU by the flight software. This PDU contains information such as the file name and file size. Also included is CFDP housekeeping information such as the unique transaction ID for this file. This value is used to distinguish the various ongoing transactions from one another. Next, the file is segmented into File Data Units (FDUs) and transmitted to the ground. The receiving software then reassembles the file as these are stored. Once the file has been completely transmitted, an End of File (EOF) PDU is sent to the ground indicating that all data have been sent. Upon receipt of this information, the receiving entity acknowledges its receipt by sending an EOF Acknowledgement (ACK) and then determines if any pieces of the file are missing. If so, the ground sends a NAK to the spacecraft listing what segments of the file need to be resent. The flight software then retransmits these pieces of the file to the ground. Once the ground has determined the file to have been completely received, it notifies the flight software of this state through the transmission of a Finished Indicator (FIN). The flight software responds with a FIN ACK and then considers the transmission complete.

Also defined in the CFDP protocol are timers to ensure that the receiving or sending entities do not wait forever for data that may have been lost. In this example, there are four timers. Both the ground and the flight software have overall transaction timers in which the transmission of the file must complete. The flight software also has a timer that starts after transmission of the EOF. If an EOF ACK is not received before the timer runs out, the EOF is retransmitted with the number of retransmissions is configurable. On the ground side, there is a similar timer for the NAKs as well as the FIN. If the requested NAK data are not received or a FIN ACK is not seen, the NAK or the FIN is resent.

MESSENGER FILE SYSTEM STORAGE

Files on MESSENGER consist of spacecraft housekeeping and science data. The flight software allows for the routing of all telemetry packets into files. It is in this way that housekeeping data and science telemetry packets are stored. The other types of files are images from the MDIS instrument. These are stored via a Direct Memory Access (DMA) transfer between an interface card and the SSR.

When a file is completed and ready for downlinking, it is placed in one of ten priority directories. These are shown in Figure 2. Operations personnel have the option of downlinking these files individually by command or by using MESSENGER's autonomous playback capability. This process involves the flight software periodically scanning these directories and determining if a file is available. If so, it starts the CFDP process by initiating a transaction to downlink the file.

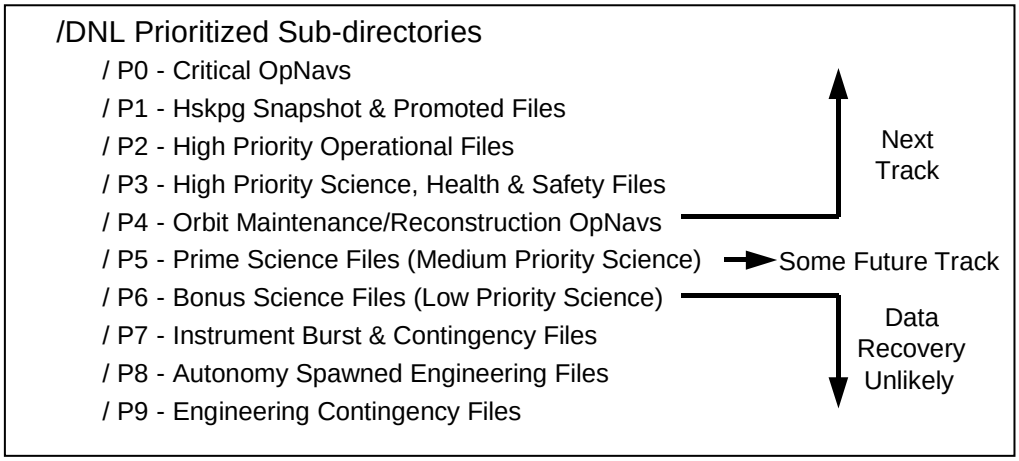

Figure 2: MESSENGER Directory Structure

CFDP ON MESSENGER

The first step in defining the use of CFDP on MESSENGER was to determine what aspects of the protocol were needed. The CFDP protocol is highly configurable depending upon mission needs, and selections such as the Deferred NAK mode were selected as a result. Also for MESSENGER, files are transferred only from the spacecraft to the ground, as heritage code is used for sending data to the spacecraft.

The initial approach for software on the flight side was to use an implementation provided by the NASA Jet Propulsion Laboratory (NASA/JPL). This implementation has many benefits including that the same code runs on both the flight and ground systems (evidenced by its incorporation into the MESSENGER ground system [4]), it is highly configurable, and it includes the full CFDP implementation. It also provides a test environment for linking entities operating on various platforms through the User Datagram Protocol (UDP), thus permitting early testing before the underlying CFDP communication layer is developed. Various tests were performed on this system, and it was found to function well on both the flight and ground hardware. However, MESSENGER

has only 8 Megabytes of RAM and 4 Megabytes of EEPROM for storing the flight image and a processor running at 25 MHz that must perform not only C&DH but Guidance and Control (G&C) as well as image compression. It was decided that in order to ensure the meeting of these tight margins, the CFDP code needed to be completely under the control of JHU/APL developers with a design created with these constraints in mind. A home-grown version of the CFDP protocol was thus designed and implemented.

This "CFDP-lite" version of the protocol implements only the aspects of CFDP that will be used on MESSENGER. For instance, as stated earlier, files are to be sent only from the spacecraft to the ground, so no capability to rebuild files on the flight side were implemented. Also, memory usage was kept to a minimum by limiting file information stored onboard for each transaction. Developing the code in-house also allows for a tighter coupling with the playback capability of MESSENGER. File system operations were placed outside of the protocol, which again added to simplifying the flight code [5].

RESULTS AND OBSERVATIONS

Initial results of the flight CFDP implementation have been developed through the build testing of the software. These results indicate that the implementation will meet the mission memory needs with a 66-kilobyte code footprint and 33 kilobytes of RAM memory usage. A study of the throughput was also performed to examine the maximum rate of PDU downlinking. It was also determined that although compatible with future JHU/APL missions, a more generic design would have allowed for easier reuse among organizations.

Once it was determined that full functionality was met, the focus on the evaluation of the CFDP implementation turned to throughput. The flight computer was loaded with approximately 100 105-kilobyte image files which were then downlinked. Processor loading was approximately 55%. A timer was set as each transfer frame was starting to be built and stopped when it was ready to be downlinked. These times to create a transfer frame ranged from < 1 millisecond to 256 milliseconds, with most transfer frames built between 1 and 2 milliseconds. With each transfer frame being 8920 bits, the "average" theoretical throughput would be between 8.9 Mbps and 4.5 Mbps. Although useful for evaluation purposes, it's important to remember this test doesn't cover how long it takes to write the transfer frame into the downlink buffer (although not a great amount of time), nor does it account for all the transfer frames that took much longer than 1 to 2 milliseconds to be built.

Further study was made on the transfer frames that fell outside of the 1 to 2 millisecond range. Initial observations indicated that some of the larger creation times were due to impacts from high-priority tasks. A second test was run with the playback task running at a higher priority. This greatly diminished the number of timing overruns, but some still remained. More analysis indicated these were due to both the opening and closing of files as well as filling transfer frames with multiple PDUs. The majority of transfer frames contain parts of two FDUs. However, at the end of a file, there may not be enough data to fill the bulk of a frame. At this time, the current file is closed and a new file is started. Thus that transfer frame could potentially consist of a partial FDU, a complete FDU, an EOF, a Metadata PDU, and a partial FDU from the next file. As a result, four calls are made to the CFDP library to retrieve a PDU for that transfer frame where it is on average

just one, and two of those calls are high-overhead file system calls involving the opening and closing of files. A lesson learned from this is that it would be advantageous to queue the transfer frames, although that leads to questions about when to start the various timers.

A final observation is this implementation of CFDP is not very portable as it is tightly coupled with the playback task. From the standpoint of JHU/APL, this is acceptable since missions use similar architectures and this would be adaptable to future spacecraft. However, it was evident that the design could have been improved when a request from outside JHU/APL was made to share this implementation. Unlike the NASA/JPL implementation which was readily available for use outside NASA/JPL, another layer would need to be added to allow this to happen with the JHU/APL implementation. Much has been learned about the protocol since the initial design that would have enabled a more adaptable version.

SUMMARY

Due to mission and operational requirements, the MESSENGER mission brought a change to the standard method of storing science and housekeeping data on JHU/APL spacecraft. Unlike past missions where a raw storage model was used, MESSENGER will be using a file system. This new technology insertion presented an issue on how to downlink efficiently the data stored in these files. It was found that CFDP would meet these needs, and through the reuse of a NASA/JPL implementation on the ground and a JHU/APL "CFDP-lite" implementation on the flight side, this protocol was included in the MESSENGER software architecture.

ACKNOWLEDGEMENTS

Scott Burleigh of NASA/JPL was extremely helpful and responsive to the needs of the JHU/APL flight software developers in interpreting the nuances of CFDP. Thanks are also due to William Stratton, Constantine Frangos, Joseph Harrison, Kevin Lyons, and Doug Holland, who implemented the ground side of the CFDP equation at JHU/APL as well as Michael Paul for his role in testing the flight software.

REFERENCES

[1] Solomon, S. C., R. L. McNutt, Jr., R. E. Gold, M. H. Acuña, D. N. Baker, W. V. Boynton, C. R. Chapman, A. F. Cheng, G. Gloeckler, J. W. Head, III, S. M. Krimigis, W. E. McClintock, S. L. Murchie, S. J. Peale, R. J. Phillips, M. S. Robinson, J. A. Slavin, D. E. Smith, R. G. Strom, J. I. Trombka, and M. T. Zuber, The MESSENGER mission to Mercury: Scientific objectives and implementation, *Planet. Space Sci., 49,* 1445-1465, 2001.

[2] *CCSDS File Delivery Protocol (CFDP).* Recommendation for Space Data System Standards, CCSDS 727.0-B-1, Blue Book, Issue 1, CCSDS, Washington, D.C., January 2002.

[3] Krupiarz, C. J., D. A. Artis, A. B. Calloway, C. M. Frangos, B. K. Heggestad, D. B. Holland, and W. C. Stratton, File-based data processing on MESSENGER, *Proc. 5th IAA International Conference on Low-Cost Planetary Missions, Acta Astronautica, Journal of the International Academy of Astronautics,* SP-542, 435-441, Noordwijk, The Netherlands, September 24-25, 2003 (published January 2004).

[4] Stratton, W. C., C. M. Frangos, J. J. Harrison, and D. B. Holland, Reuse of the JPL CFDP software in the APL Common Ground System, *Proceedings of the 5th International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations (RCSGSO), Paper 103-A0036,* 8 pp., Pasadena, CA, July 8-12, 2003.

[5] Krupiarz, C. J., S. C. Burleigh, C. M. Frangos, B. K. Heggestad, D. B. Holland, K. M. Lyons, and W. C. Stratton, The use of the CCSDS file delivery protocol on MESSENGER, *Space Operations 2002 Conference, American Institute of Aeronautics and Astronautics (AIAA),* abstract T5-35, Houston, TX, October 9, 2002.