

Autonomous Data Transfer Operations for Missions

**Max Repaci, Paul Baker, and Fred Brosi
Global Science and Technology, Inc.
Greenbelt, MD, USA**

Automating the data transfer operation can significantly reduce the cost of moving data from a spacecraft to a location on Earth. Automated data transfer methods have been developed for the terrestrial Internet. However, they often do not apply to the space environment, since in general they are based on assumptions about connectivity that are true on the Internet but not on space links. Automated file transfer protocols have been developed for use over space links that transfer data via store-and-forward of files or segments of files. This paper investigates some of the operational concepts made possible by these protocols.

The current trend in space missions is away from a few large expensive missions and toward many small low-cost missions. If the overall budget is not increased, this trend necessarily implies a lower cost per mission. We argue that a portion of the necessary cost reductions can be realized through increased standardization and automation. Automation of operations and standardization of communications protocols and operational methods can allow significant savings in both the development and operational phases of a mission. Standardization of communications protocols and operations can allow the development, maintenance, and even operational costs of infrastructure and software to be shared among many missions. Automation can eliminate a significant component of the operations budget.

In the past, the mission budgets were larger and there were fewer missions. Those mission components whose costs would be significantly reduced by standardization and automation were a relatively smaller fraction of the total mission budget, and there were fewer missions over which to share costs. Therefore missions made heavy use of custom protocols and software, and manual operations were common. Now that the cost of the other mission components has been reduced, for example, by flying smaller spacecraft, missions have a strong incentive to reduce costs by these methods.

Standardized protocols and operational methods can significantly reduce development and operations costs for a mission. These reductions come about by the reuse of program code and procedures, by reducing operator and developer training costs, and by reducing the time and effort required for design of new missions. Standards also permit interoperability between missions, which allow not only sharing of development costs, but also operational costs, as infrastructure and personnel can be shared among missions.

Certainly automating the data transfer operation, if it is possible, will reduce operations costs. We argue that these reductions will be maximized by the use of a reliable transfer protocol to transfer data. A reliable transfer protocol is one that guarantees that data will be delivered in the same order in which it was sent and without errors. Due to the availability of random-access storage on the spacecraft, it is now possible to automatically retransmit data lost on the link, thus guaranteeing the delivery of transmitted data in a complete and uncorrupted form. This is done by breaking the data up into small blocks and retransmitting those blocks that were corrupted in transit. While this practice admittedly increases the overhead for data transfer, it can still result in overall savings. Compare it, for example, to a method commonly used in the past. With this method of data transfer from a spacecraft, the downlink error rates were monitored until the link quality exceeded a specified level, and then the data was downloaded. If the data contained more than the acceptable number of errors, then the entire transfer had to be repeated. With a reliable protocol, the transfer can proceed automatically as soon as the connection is established, even if the error rate is relatively high, and there are guaranteed to be no errors in the transferred data. Thus, data can be transferred under link conditions that would not have been possible if a reliable protocol were not being used. Also, allowing uncorrected errors to remain in the transferred data can impede the automation of the analysis of the data, and may require potentially expensive corrective measures to be taken by the end user. Therefore, even without automation, the

use of a reliable transfer protocol can in itself provide cost savings by allowing more efficient use of the link time.

We will investigate operational scenarios possible with an automated data transfer method. We will consider operations to support routine data transfers, since those most readily lend themselves to automation. Examples of these transfers include that of command loads to the spacecraft or of stored telemetry data from the spacecraft. Automating routine transfers can reduce operations costs; development costs can be reduced as well by standardizing the software, interfaces, and protocols used for data transfer. We will discuss how standardized data transfer protocols based on automated file transfers can be used to operate a mission, and present examples of two such protocols.

I. Mission operations via file transfers

We will now describe a general procedure for mission operations via file transfers. The concept of mission operations can be abstracted to that of transfer of data to and from the spacecraft. For example, the data may represent telemetry data transferred from the spacecraft or a command load transferred to the spacecraft. In any case, data is stored on a local file store at the source node. The local file transfer application is directed to transfer the file to the remote node. This transfer can be managed in one of two ways: as a "push" operation or as a "pull" operation. In the push mode, the file is sent directly to the remote node from the sending node, without any prior prompting from the receiving node. In the pull mode, the receiving application asks the application on the sending node to send the file. A client/server application transfers data with a pull mode. Which mode is better to use depends on the application. For example, consider an automated transfer process where the sending node always knows where to send the data. In this case, using a push transfer might eliminate any polling that might be necessary if a pull mode were used instead. On the other hand, one would use a pull mode transfer if the destination of the data were not a managed parameter.

The process of getting the data to the local file store is not managed by the data transfer protocols, and must be arranged by the mission designers. In the case of the transfer of instrument data, for example, a method must be defined for transferring the data from the instrument to a mass storage accessible to the file transfer protocol application. There are other limitations and requirements implicit in operating missions using file transfer operations. Naturally, the spacecraft must have at least a rudimentary file system on board. Of course, the "file system" could be as simple as some random-access memory. In addition, if a mission is required to transfer data that cannot be efficiently transferred as files, there must be another method for transferring the data. For example, heavily interactive operations or a real-time data stream cannot be supported efficiently due to overhead considerations. Most spacecraft have health and safety data that is transferred as a real-time (low latency) stream. However, any operation that transfers data that is stored on the spacecraft is probably suitable for transferring as files. For example, downloading of stored instrument or other telemetry data, and uploading of memory or command loads.

It may not be possible to establish a reliable connection all the way from one end of the data transfer path to the other. The contact with a spacecraft is very often intermittent, and communications may be established to the spacecraft via different groundstations on different contacts. Therefore, the data transfer methods considered here also support a store and forward method. The store and forward method is a concept similar to that of sending email on the Internet. It allows automated routine data transfers to proceed without concern for the contact schedule of the spacecraft.

The two file transfer protocols described here are the CCSDS File Delivery Protocol (CFDP) and the Simple Automatic File Exchange (SAFE) protocol¹. CFDP is being standardized by the CCSDS (Consultative Committee for Space Data Systems), an international standards organization for space communications protocols. More information can be found on CFDP in the relevant CCSDS documents, referenced at the end of this paper². The development of SAFE was supported by NASA at Goddard Space Flight Center's Advanced Architectures and Automation branch. These protocols both

¹ More information about SAFE can be found on the SAFE homepage: <<http://safe.gst.com>>.

² Also see the CCSDS homepage <<http://www.ccsds.org>>.

make use of existing underlying protocols; SAFE assumes the existence of a transport layer and CFDP assumes the existence of a link layer. We will discuss the ramifications of these assumptions of the underlying communications structure and discuss how the general file transfer methods outlined above can be implemented with these protocols.

II. Operations with the SAFE protocol

The SAFE protocol is a file transfer protocol that is designed to transfer data automatically to or from a host that can only be contacted intermittently. The file transfer operation is performed via messages exchanged between SAFE entities at the source and destination of the transfer operation. If store and forward operation is required, there would also be entities at the store and forward nodes, but these are transparent to the endpoints. The messages can contain file data (identifier, offset, length), or requests for file data. Since the messages containing the file data are self-identifying, the protocol will support data transfer both in push and pull modes. In the pull mode, the transfer fits the client/server model: there is a data server running on the node containing the data source and a client on the node to which the data is to be sent. The transfer operation is performed via requests for data that are sent from the client. These requests contain an identifier for the file, the offset of the start of the request in the file, and the length of data requested. When the server receives a request, it responds with the data requested, or with an error condition if the data is not available. When the client receives the data, it makes it available to user applications. If the data does not arrive in a reasonable time or is not complete, the client requests the missing data again. The push mode transfer works in the reverse fashion; the sender sends data, the receiver acknowledges the data when it is received, and if the sender does not receive an acknowledgement in a reasonable time then it resends the data. Thus, reliability is maintained through a data transfer loop closed at the application layer.

It is assumed that there is an underlying transport protocol that will deliver the messages between the SAFE entities. The transport layer is assumed only to provide a stream service, i.e., ordering of the data, and notification of lost data. SAFE does not require that the transport layer provide reliable delivery in order to achieve end to end reliability, since reliability is managed at the application layer. The benefits and drawbacks of protocol layering will be discussed below.

SAFE can also provide a reliable store and forward stream service using a standard extension called the Recyclable File System (RFS). An application on the node with the data source maintains the RFS, which is a series of rotating buffers into which data can be written. The data is then automatically transferred to an equivalent set of buffers on the destination node. The application also keeps track of which data has been transferred, so that applications can access the data as soon as it arrives, even if it hasn't yet been contiguously transferred. An application that will use the data can read it from the remote RFS, either as a stream or by random access. The reliable stream service could be used, for example, to upload command loads to the spacecraft.

III. Operations with CFDP

CFDP is a protocol designed for transferring files over space links and maintaining a remote file system on a spacecraft. Like SAFE, CFDP operates via the exchange of messages between entities residing on the source and destination nodes of the data transfer. The messages can contain file data, directives for remote management of the file store, user message data, or any combination thereof. The CFDP entities handle the processing of file management directives and writing of the file passed with the message, if there was one. File management directives include such things as file deletion, file or directory creation, or renaming files. The behavior the entities and the format of the messages are specified by the protocol. Each entity is controlled by a user application that directs it to send messages. The process of sending a message in the CFDP nomenclature is called a 'transaction'. The user application can direct a CFDP entity to start a transaction (the Put directive), cancel, suspend, or resume a transaction, or report on the status of a transaction in process. For example, to transfer a file from the spacecraft to the ground, the user on the spacecraft would start a 'Put' transaction, that is, send a message. The message would include the file to be transferred. The remote entity would receive the message and automatically write the file to the local file system. Transactions may also include

user messages. The user messages can be used to add functionality to the core procedures of CFDP. The format of the user messages is not specified, except for that of two reserved messages. These reserved user messages represent standard extensions to CFDP. One of them allows a remote user to get directory listings of the local file store and the other initiates proxy transactions. Proxy transactions allow a user to direct a remote CFDP entity to initiate a transaction, for example, to send a file. Without the proxy user message, CFDP would not have any standard way to get a file from a remote entity.

CFDP can be easily configured to perform automated data transfer. When an instrument writes a data file on the source node file system, the local user application directs the resident CFDP entity to perform a Put operation to transfer the file to the ground entity. When the transaction is finished, the entity at the destination notifies the spacecraft entity that the file has been delivered. The spacecraft entity then notifies its user, which can then delete the file to free resources. The reverse procedure could be used to transfer command loads to the spacecraft. The user applications on the spacecraft and on the ground node are not part of the CFDP standard. They would be developed by the implementers of this automated file transfer operation. The user application on the spacecraft directs its local CFDP entity to transfer any new files and handles the delivery of any user messages. It may also delete the data after it has been transferred successfully, or its peer application on the ground could issue the delete remotely. The user application on the ground uploads commands and other data, handles user messages, and provides an interface for remote management of the data store on the spacecraft.

IV. Discussion

Certainly the reliable, automatic transfer of data occurs all the time on the terrestrial Internet. However, solutions developed for the Internet may not be easy to apply to the space environment. The Internet protocols are optimized for use under the conditions typically found on the terrestrial Internet. The space environment has many characteristics that can cause the Internet protocols to operate in a non-optimal way or to fail outright. For example, the Internet file transfer protocol, FTP, is not bandwidth-efficient and requires an underlying end-to-end reliable transport layer protocol. Much effort has been made to adapt the most widely used Internet protocols to the space environment, including extensions and modifications to the Internet protocols standardized by the CCSDS³. For example, the CCSDS equivalent of FTP, SCPS-FP, has improved bandwidth efficiency. However, it is still not as efficient as an application designed for automated transfers in the space environment can be, and it still requires an end-to-end connection to operate. It is therefore not optimally suited for routine automated operations in the space environment.

The Internet Protocol suite is an example of a layered protocol. Protocol layering refers to the process of dividing the protocols into modular sections, or layers, that handle certain functionality. Standard interfaces are provided between the layers, so that implementations of certain layers that are optimized for a particular environment or application can be substituted while still reusing the other parts. The use of a layered communications protocol also allows code reuse. So layering can reduce development costs, at the expense of additional overhead in the data transfer.

A solution to the problem of designing an automated data transfer application for space operations is to design an application that is built on the lower layers of the Internet protocols or their space qualified variants. SAFE is one such application. In fact, work is currently underway to layer SAFE further to separate out the code that handles the store and forward delivery of messages. Such a "messaging layer" can be then be used by other applications that transfer data by store and forward, but don't need to write files.

CFDP is designed to support file transfers over space links, and is well suited to automated operations. This focus should allow it to perform well under its targeted operations scenarios. However, CFDP does not have a layered design. Therefore, even if a transport protocol suitable for space data transfers is available, the implementers of CFDP for the mission cannot use it. Conversely, if the transfer of data

³ More information can be found on the SCPS homepage at <<http://www.scps.org>>.

as files is not adequate for the needs of the mission, one cannot write a new application that uses the transport layer of CFDP⁴. Therefore, if substantial additional functionality beyond file transfer were required, operations using the Internet protocols would probably cost less.

V. References

CCSDS File Delivery Protocol (CFDP). Draft Recommendation for Space Data System Standards, CCSDS 720.1-G-0.5. Green Book. Issue 0.5. Washington, D.C.: CCSDS, July 1999.

CCSDS File Delivery Protocol (CFDP). Draft Recommendation for Space Data System Standards, CCSDS 727.0-R-3. Red Book. Issue 3. Washington, D.C.: CCSDS, July 1999.

⁴ CFDP can be operated over the Internet's unreliable transport layer, UDP/IP, but at the cost of additional overhead and without gaining the benefit of using the reliable transport layer. Also, one could write an application which used the messaging service of CFDP, but it is somewhat limited and there is no standard for the interface or message format.