MDP: Reliable File Transfer for Space Missions[1]

James Rash
NASA/Goddard Space Flight Center
Greenbelt, MD 20771
Ed Criscuolo, Keith Hogie, Ron Parise
Computer Sciences Corp
7700 Hubble Dr.
Lanham-Seabrook, MD 20706

Abstract—**This paper presents work being done at NASA/GSFC by the Operating Missions as Nodes on the Internet (OMNI) project to demonstrate the application of the Multicast Dissemination Protocol (MDP) to space missions to reliably transfer files. This work builds on previous work by the OMNI project to apply Internet communication technologies to space communication.**

The goal of this effort is to provide an inexpensive, reliable, standard, and interoperable mechanism for transferring files in the space communication environment. Limited bandwidth, noise, delay, intermittent connectivity, link asymmetry, and one-way links are all possible issues for space missions. Although these are link-layer issues, they can have a profound effect on the performance of transport and application level protocols. MDP, a UDP-based reliable file transfer protocol, was designed for multicast environments which have to address these same issues, and it has done so successfully. Developed by the Naval Research Lab in the mid 1990's, MDP is now in daily use by both the US Post Office and the DoD.

This paper describes the use of MDP to provide automated end-to-end data flow for space missions. It examines the results of a parametric study of MDP in a simulated space link environment and discusses the results in terms of their implications for space missions. Lessons learned are addressed, which suggest minor enhancements to the MDP user interface to add specific features for space mission requirements, such as dynamic control of data rate, and a checkpoint/resume capability. These are features that are provided for in the protocol, but are not implemented in the sample MDP application that was provided. A brief look is also taken at the status of standardization. A version of MDP known as NORM (Nack Oriented Reliable Multicast) is in the process of becoming an IETF standard.

TABLE OF CONTENTS

1. INTRODUCTION

Our reference architecture for MDP-based reliable space data transfer utilizes the Internet suite of protocols. These are based on the OSI seven-layer model of networking, but with some differences. In the OSI model, there are seven distinct layers. Starting from the lowest, they are:

1. Physical - Raw bits, coding (wire, fiber, RF)
2. Link - Frames (HDLC, FDDI, ATM, ethernet)
3. Network - end-to-end addressed datagrams (IP)
4. Transport - multiplexed packets (TCP, UDP)
5. Session - login, authentication
6. Presentation - formating, translation
7. Application - user data

In the Internet implementation, layers 5 - 7 tend to be compressed into a single application layer. For example, the Internet file transfer application "FTP" incorporates elements of the session layer (user login), presentation layer (translation of ASCII files), and application layer (transfer user files).

MDP is an application layer protocol. It operates over a User Datagram Protocol (UDP) transport layer. The simple nature of UDP packets renders them largely insensitive to link layer issues, such as delay and asymmetry, but requires reliability to be implemented at the application layer.

2. OVERVIEW OF BULK DATA TRANSFERS IN SPACE

In order to make effective use of MDP for bulk data transfers from space missions, mission designers must change the way they think about data transfers.

Old Paradigm: Recorder Playback

Current missions have a legacy architecture that derives from a time when bulk data storage was implemented with a sequential tape or wire recorder. Data would be recorded while the spacecraft was out of contact, often preformatting the downlink frames directly onto the recorder. Later, the tape would be rewound and the data played back during a ground contact. Reliability was achieved by using sufficient amounts of forward-error-correction (FEC) codes, such as Reed-Solomon. If the data quality was still unacceptable, and there was sufficient contact time, portions of the data would be played back a second time. Ground software would then combine the two playbacks and attempt to fill in any dropouts.

With the advent of space-qualified solid-state memory, newer missions replaced mechanical tapes with solid state recorders (SSRs). But these SSRs continued to emulate the functionality of sequential tape recorders, burdening the flight operations team (FOT) with the increasingly complex tasks of managing the recorder's storage and assuring downlink data quality.

New Paradigm: File Transfer

New missions are beginning to move to a different architecture for data storage and playback. This new paradigm features the use of an onboard operating system (OS) that supports files. Bulk, solid-state memory is organized to support implementing a file system on top of it. This incurs two immediate benefits: automatic storage management and random access playback. With files, the low-level storage management is taken care of by the OS and the file system, instead of by the FOT. These commercial-off-the-shelf (COTS) packages have thousands of hours of testing and optimization, unlike manual procedures or mission-specific application code. The random-access nature of files makes them well suited to support prioritized or non-sequential playback and deletion. This is an increasingly important requirement, particularly for high-data-volume earth-science missions such as Landsat. By organizing the data into files, each file can be downlinked using a generic file transfer application, such as MDP, that assures data quality by automatically performing error correction and/or retransmission as needed.

3. Space Related Issues

There are a number of apparent issues for space-based usage of Internet protocols. Although these are link-layer issues[1], they can have a profound effect on the performance of transport and application level protocols[2]. Any practical design for using Internet protocols for reliably transporting files across a space-to-ground link must take these link-layer issues into account.

Bandwidth

Space missions always operate in a constrained bandwidth environment. Often, the constraints are determined by factors that are independent of the science data requirements, such as electrical power budget and launch vehicle payload capacity. Regardless of the absolute numbers, a practical protocol must make reasonably efficient use of its bandwidth, and must either share the link fairly with other protocols, or be able to be throttled to a fixed portion of the available bandwidth.

Noise

Frequently, it is pointed out that most packet losses on the Internet are due to congestion, whereas most losses on a space-to-ground link are due to noise. TCP, the most well known of the Internet protocols, has no mechanism for distinguishing packet loss due to noise from packet loss due to congestion, so it always assumes congestion and responds to noise by slowing down. This feature of TCP is often used to imply that *all* Internet protocols operate sluggishly or fail outright in the presence of noise. *This is not true for UDP based protocols.* UDP does not perform flow control and never attempts to throttle the data at the transport layer. So an IP based file transfer solution should either be based on UDP, or run over a link with sufficient FEC to reduce the bit error rate (BER) to a level that will allow TCP to operate efficiently.

Delay

Often it is stated that space missions *must* be carried out with "Round trip delays much greater than ground systems"[3], and that "...long propagation times cause terrestrial protocols to operate sluggishly or fail outright"[3]. For low earth orbit (LEO) missions, which represent the *large majority* of space missions, this is simply not true. A LEO spacecraft is only 200-400 miles away when it passes overhead. Since radio waves travels at the speed of light, this translates into only a 4 mS round trip time! Even at the horizon, which for a spacecraft in a 400 mile high orbit is approximately 3000 miles away, this is about a 32 mS round trip time. Compare this with typical Internet ping times from Baltimore to Los Angeles of 100 mS and the LEO spacecraft should actually run *better* than coast-to-coast terrestrial links. Even out to geosynchronous orbit, the round trip delay time is only 240 mS. Experiments have been performed at the NASA Glenn Research Center[4] using the ACTS satellite, which have operated TCP/IP at over 400 megabits/second at this distance. Laboratory experiments have suggested that lunar distance, with its 1666 mS round trip time, would require some care in setting up the connection, and represents the practical limit for TCP based applications. Beyond this distance, deep space missions, such as Mars, should look to using a delay-insensitive UDP based protocol.

Intermittent Connectivity

Spacecraft that are not in a geosynchronous orbit cannot maintain continuous direct contact with the ground. Contacts are limited to a brief time when the spacecraft passes within line-of-sight of the ground station. For a low earth orbit, this "pass" is typically no more than 8 to 15 minutes long, a few times a day. If more contacts are needed, more ground stations must be used, complicating the routing of data to and from the spacecraft[1].

Link Asymmetry / Unidirectional Links

Most spacecraft have a much greater downlink bandwidth than uplink bandwidth. This asymmetry is often incorrectly attributed to the fact that spacecraft are limited by their power and weight budgets, and cannot generally support large steerable high-gain antennas. While this fact is true, it is not what limits the uplink data rate. Up to a point, any shortcomings of the spacecraft antenna or receiver can be compensated for by more power and bigger antennas on the ground. The real limitation is driven in part by physics, but mostly by convention and legacy equipment. In any event, a practical IP based file transfer solution should be able to operate with large link asymmetry, and should be able to take advantage of any unidirectional "return only" downlinks.

4. General Description of MDP

MDP, a UDP-based reliable file transfer protocol, was designed for multicast environments which have to address these same issues, and it has done so successfully.

Developed by the Naval Research Lab in the mid 1990's, MDP is now in daily use by both the U S Post Office and the DoD. The code is freely available from NRL's MDP website[2], and runs on multiple platforms.

MDP is implemented in two major pieces: A protocol library and a file transfer application. The library embodies all of the MDP protocol logic, and is accessed through a well-defined application program interface (API)[5]. The application program provides the interfaces to the user, the file system, and the protocol library. This modular approach allows the possibility of tailoring the application for specific requirements while maintaining interoperability with all other MDP implementations.

Although developed specifically for reliable file transfer in a multicast environment, MDP has had to address and solve the issues of channel utilization, delay tolerance, noise tolerance, asymmetry, unidirectional links, and link intermittency.

Bandwidth Utilization

The MDP protocol is implemented using UDP packets. UDP is a connectionless transport protocol designed to operate over IP. Its primary functions are error detection and multiplexing. UDP does not guarantee the delivery or order of packets, but guarantees that if a packet is ever delivered with errors, such errors will be detected. Because the UDP format is simple, it has a low overhead. This results in a very efficient protocol. MDP's bandwidth utilization, calculated as file-size/total-bits-transferred, can be as high as 90%. Because this number is calculated by using the *total* bits transferred over the link, it includes *all* overhead from *all* sources, not just MDP's overhead. In addition, the MDP file transfer application provides the ability to throttle transfers to a specified average bitrate.

Delay Tolerance

TCP is a delay sensitive protocol, due to its need to establish a "connection" with a three-way handshake, and to acknowledge every two packets sent UDP, on the other hand, is a "send and forget" protocol. This makes it completely delay insensitive. By using UDP, and maintaining it's own internal timers, MDP has been designed to operate with large round-trip-time delays, on the order of hours or days.

Noise Tolerance

On an IP based space link, noise manifests itself as dropped packets, usually due to cyclic-redundancy-check (CRC) failures. MDP has two mechanisms for handling this: retransmissions and application-level reed-solomon FEC. When the MDP client on the receiving side of a transfer detects that it has missed one or more packets, it sends an aggregated Negative Acknowledgement (NACK) back to the sender, who will automatically retransmit the lost packets. In addition, in a highly errored environment, MDP has the option of proactively adding *additional* reed-solomon FEC symbols to the transfer *at the application layer*. These can be used to reconstruct damaged or lost packets without requesting retransmission. The amount of FEC added is selectable, and should be based on a study of the trade-offs between the overhead of retransmissions vs the overhead of additional FEC, at a particular error rate.

High Link Asymmetry

Because MDP is NACK based rather than ACK based, it is extremely conservative of the uplink channel, maintaining at least a 1000:1 downlink/uplink ratio, even in the presence of a 10E-5 BER. Ratios of 10,000:1 and beyond are common.

Unidirectional Link Capability

By design, the MDP protocol has loose coupling between the downlink of data and the uplink of NACKs. This means that the sender does not wait for NACKs while downlinking a file. And, thanks to MDP's use of a connectionless UDP transport, the NACKs can even be segregated into a different contact! This means that an MDP server onboard a spacecraft can make use of the much more readily available "downlink only" contacts to get the bulk of the data downlinked, and make use of a later "bidirectional" contact to uplink any pending NACKs and downlink the retransmissions. MDP also has an "Emissions Control" (EMCOM) mode where the client *never* requests retransmission, and simply makes a best-effort attempt to receive the file.

Intermittent Links

Again, because of the loose coupling and delay insensitivity, MDP can begin the transfer of a file during one contact, and complete it on subsequent contacts.

5. End-to-End Data Flow Architectures

The MDP application is a single program that can be operated as either a client or a server. The MDP server is designed to utilize a "hot directory" concept, where new files arriving in the "hot directory" are automatically queued for transfer. The MDP client passively receives files that are "pushed" to it by the server, and has the capability to hand-off the received file to another arbitrary application upon completion. This section will briefly examine two possible MDP end-to-end data flow architectures, but a detailed discussion is beyond the scope of this paper. A more complete discussion of end-to-end data flow scenarios can be found in the paper "Internet Data Delivery for Future Space Missions"[6].

Direct to User

In this configuration, a single MDP server runs on the spacecraft, and the MDP client runs in the end user facility, such as the Mission Operations Center (MOC) or the Science Operations Center (SOC). See Figure 1. In addition, multiple simultaneous clients are possible, utilizing MDP's multicast capability. Using multicast to ship the data to multiple clients is desirable because any needed packet replication is taken care of by the *routers on the ground network*, never by MDP. In a multicast configuration, the clients can be a mix of EMCON and non-EMCOM, where only the "primary" non-EMCON clients are expected to send back a "file received" acknowledgement.

[2] http://pf.itd.nrl.navy.mil/projects/mdp/

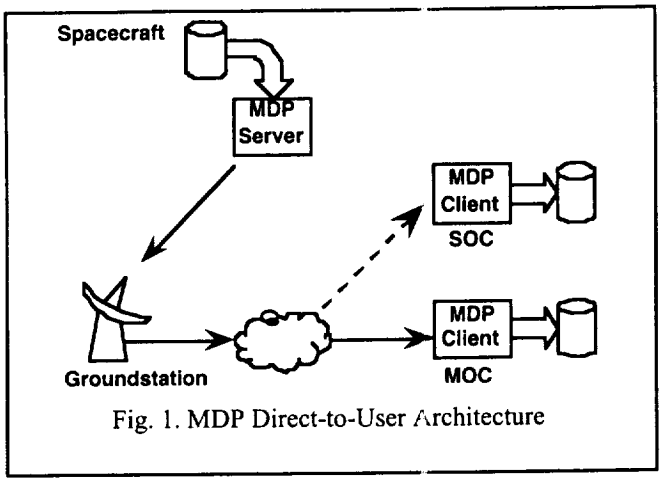


Fig. 1. MDP Direct-to-User Architecture

Store and Forward

In this configuration, shown in figure 2, a single MDP server runs on the spacecraft, sending to a single MDP client at the groundstation. Files are stored at the groundstation for subsequent transfer to end users, possibly at a different time and at a different data rate from the downlink. Again, a mix of "primary" and "best effort" clients are possible.

6. MDP Parametric Study

In 2001, we performed a parametric study of MDP in a simulated space link environment. The purpose was to characterize MDP's performance under a wide range of conditions, including ones that are typical for many space missions.

Independent Variables

During the test, four independent variables were varied, one at a time, for two different test series[3] For series 1, these were:

1. Data bitrates: 128K, 256K, 512K, 1M, 2M
2. File sizes: 1MB, 2MB, 4MB
3. Bit Error Rates: 0, 1E-8, 1E-7, 1E-6, 1E-5
4. Round trip delay: 0mS, 10ms, 100mS, 500mS

For test series 2, these were:

1. Data bitrates: 1M, 2M
2. File sizes: 5MB, 50MB
3. Bit Error Rates: 0, 1E-8, 1E-7, 1E-6, 1E-5
4. Round trip delay: 0mS, 10ms, 100mS, 500mS

Dependent Variables

After the tests, two dependent variables were plotted. These were:

1. Bandwidth Utilization
2. Link Asummetry

Test Setup

An automated test environment, shown in fig 3, was developed to perform the file transfers and collect packet level statistics at the server, router and client. A programmable Adtech channel simulator was inserted in the serial communications link to insert delay and noise. The test control software was resident on the same machine as the MDP client, but used separate communications paths for control so as not to introduce error into the measurements.

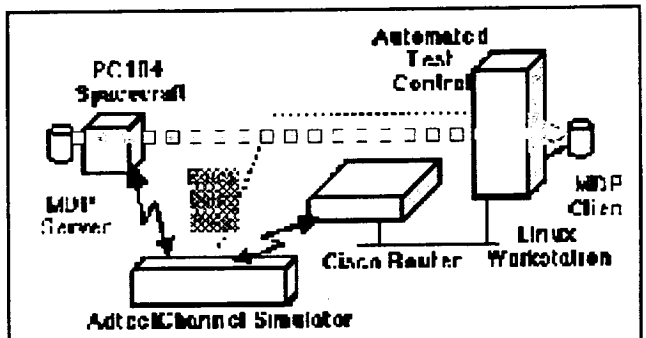


Fig. 3. Test Setup

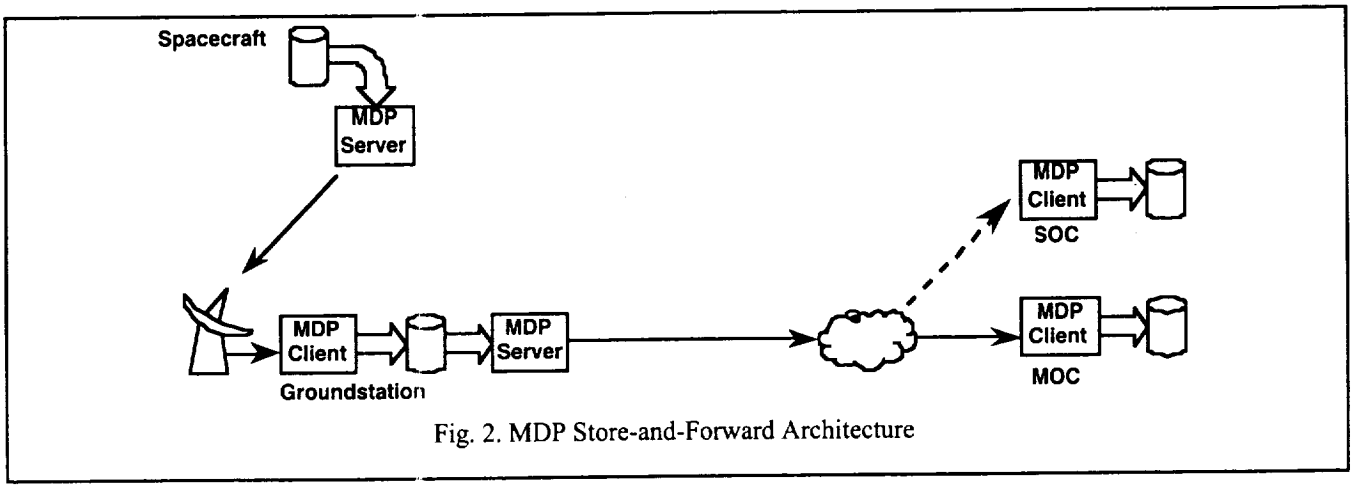


Fig. 2. MDP Store-and-Forward Architecture

[3] For both test series, the uplink was constrained to 2 Kbits/sec.

Test Results

A summary of the test results appears in figures 4 and 5. A set of spreadsheets containing the complete results of the test is available for download[4].

One of the most notable findings is that bandwidth utilization and link asymmetry are essentially independent of round trip time (delay). This behavior for bandwidth utilization is in marked contrast to TCP-based protocols, such as FTP, which run into a performance "wall" once the delay-bandwidth product exceeds the size of their window buffer.

The large values for link asymmetry mean that even a mission with 10E-5 BER and a 2 Kbit/sec uplink can support downlinks of 1 Mbit//sec. And at a more typical[5] BER of 10E-7 (after FEC), downlinks of 10 Mbits/sec are possible. Although this is important for near-term missions which must accommodate a legacy 2 Kbit/sec uplink from existing groundstations, it is less of a concern for TDRSS based missions, which can support a symmetrical uplink/downlink if desired.

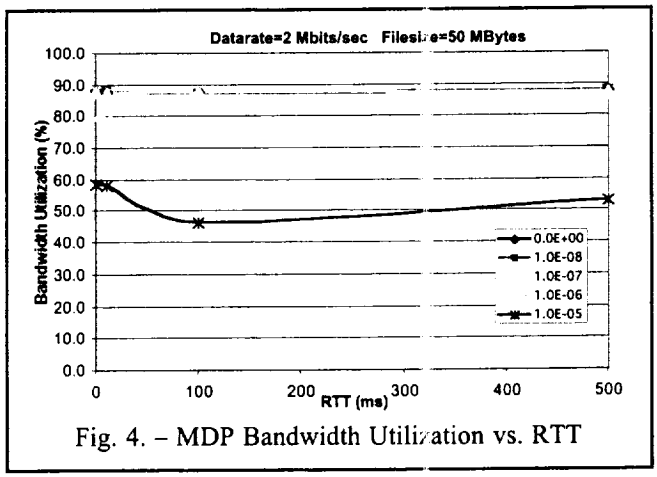


Fig. 4. – MDP Bandwidth Utilization vs. RTT

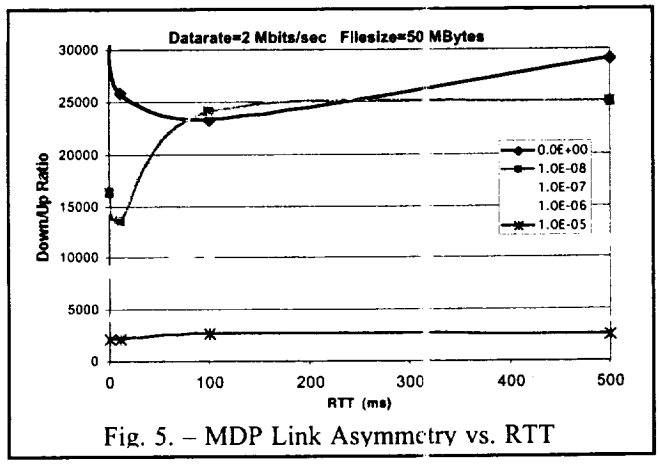


Fig. 5. – MDP Link Asymmetry vs. RTT

[4] http://ipinspace.gsfc.nasa.gov/documents/

[5] Based on measurements of actual Wind/Polar mission data.

7. MDP Flight Tests

MDP will be flight tested in the summer of 2002 on the Communication and Navigation Demonstration on Shuttle (CANDOS) mission. This HitchHiker mission is part of a 16 day shuttle flight, and has its own independent transceiver which will be used to directly contact either gorundstations or TDRSS, independent of the shuttle comm system. CANDOS will demonstrate basic IP connectivity on the space link, mobile-IP routing, and reliable file transfer using MDP. File transfers will be conducted under realistic conditions, including intermittent and "downlink only" contacts. The CANDOS mission is discussed in more detail in the paper "Space Communications Demonstration Using Internet Technology"[7].

8. Enhancements to the MDP Application

In the course of our investigations, we identified several potential enhancements to the MDP application that would improve its ease of use in a spacecraft environment. These enhancements are primarily associated with improving the ease of automatically handling intermittent contacts.

Runtime Control Interface

Currently, the MDP application can only set its runtime parameters via commandline switches set at its initial execution. This requires stopping and restarting the application each time a change is needed to one of these parameters. Almost all of them are settable through a call to the MDP protocol library using the documented API. It would be a straightforward addition to the MDP application to have it open a "commanding" socket, and accept runtime commands to alter these parameters on the fly.

Runtime Datarate Throttle Control

MDP has the ability to throttle it's maximum bitrate. This is a necessary feature for applications built on top of UDP, as UDP does not incorporate any flow control. Giving the MDP application a command to *dynamically* change its bitrate would allow it to adapt to changing spacecraft modes without having to stop and restart the server.

Checkpoint / Restore

By adding a pair of commands to save all of MDP's internal state into a file, and restore it later, we would gain the ability to resume an incomplete transfer that was interrupted by a reboot of the processor, such as when the spacecraft goes into a "safehold" mode.

Pause / Resume

This pair of functions would be used to manage the MDP server and client by pausing it when the spacecraft was out of contact with the ground. In its "paused" state, all MDP's timers would be frozen, but its other state information would be preserved. This would prevent the server and client from uselessly sending either data packets or NACKS while out of contact. This functionality can currently be provided by using the host operating system to suspend the MDP application, but this approach requires external scripts to make it happen.

React to Transceiver State

This capability may just be a refinement of the Pause/Resume commands. It would allow notifying MDP of the status of the transmitter and the receiver separately. MDP's response to XmitOff and RecvOff commands would be different, depending on whether it was running as a server or as a client. For example, a client with its transmitter off but its receiver on would continue accepting data packets sent by the server, but would accumulate and defer any NACKS needed until such time as the transmitter was on.

9. Standardization Activities

RMT Working Group

The Internet Engineering Task Force (IETF) has established the Reliable Multicast Transport (RMT) working group. The purpose of the RMT is to standardize reliable multicast transport. Its efforts focus on one-to-many transport of large amounts of data. This working group expects to initially standardize three protocol instantiations, one each from the following three families:

1. A NACK-based protocol
2. A Tree-based ACK protocol
3. An "Asynchronous Layered Coding" protocol that uses Forward Error Correction

MDP falls into the first class. The authors of MDP are active in the RMT and have submitted MDP as the basis for their standard NACK Oriented Reliable Multicast (NORM) protocol.

NORM

The NACK Oriented Reliable Multicast protocol is currently defined in a set of Internet-Drafts dated November 2001 and March 2002. It is essentially based on MDP, with some additional generalization to support arbitrary types of FEC. MDP is specific in its use of Reed-Solomon for application-level FEC, whereas NORM allows the use of standardized FEC "building blocks". These functional "building blocks" are at the core of the RMT working group's efforts, because many of the functions (such as FEC) have applicability across all three classes of Reliable Multicast Transports. Work in the NORM area of the RMT is active and ongoing.

10. Future Work

More MDP Flight Experience

MDP is currently being considered for use on several upcoming space missions, including the Global Precipitation Mission (GPM), and the Solar Dynamics Observatory (SDO). The OMNI project is actively working with these, and other missions, to provide systems engineering support for the preliminary design of their end-to-end IP infrastructure.

Implement and Fly Enhancements

Work is currently underway to implement some of the enhancements to the MDP application that were proposed in section 8. These enhancements will be incorporated into a flight-ready MDP package before the year's end.

Hardware Assisted High-Rate Transfers

Later this year, preliminary work will begin on designing an approach for providing hardware-assisted high-rate data transfers. The plan is to identify that portion of the MDP protocol that can be incorporated into hardware without compromising the layered approach of an IP architecture.

Status Updates

Information on the results of future MDP activities will be posted on the OMNI project web site at http://ipinspace.gsfc.nasa.gov/.

11. Conclusions

MDP is well suited to provide an inexpensive, reliable, standard, and interoperable mechanism for transferring files in the space communication environment. It successfully addresses the issues of limited bandwidth, noise, delay, intermittent connectivity, link asymmetry, and one-way links. MDP's high link asymmetry tolerance makes it particularly well suited to Earth-Science missions with high downlink requirements and limited uplink capabilities.

12. Acknowledgements

The research described in this paper was carried out by personnel from Computer Sciences Corporation working for NASA's Goddard Space Flight Center under contract GS-35F-4381G S-43981-G, with additional efforts and support contributed by individuals from various GSFC organizations. The work was funded by NASA's Earth Science Technology Office (ESTO). The authors would like to thank Joe Macker and Brian Adamson of NRL for the development of MDP, Dave Israel and GSFC code 450 for their pioneering work on the CANDOS project, and ITT Industries for the development of the Low Power Transceiver (LPT) used on CANDOS.

REFERENCES

[1] K Hogie, E Criscuolo, R Parise, "Link and Routing Issues for Internet Protocols in Space", 2001 IEEE Aerospace Conference, Big Sky MT, March 2001

[2] E Criscuolo, K Hogie, R Parise, "Transport Protocols and Applications for Internet Use in Space", 2001 IEEE Aerospace Conference, Big Sky MT, March 2001

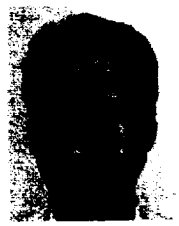
[3] Consultative Committee for Space Data Systems, "Space Communications Protocol Specification (SCPS) - RATIONALE, REQUIREMENTS, AND APPLICATION NOTES", CCSDS 710.0-G-0.3, April 1997

[4] A Welch, D Brooks, D Beering, D Hoder, M Zernic, "Experimental results of running TCP/IP over ATM on NASA ACTS HDR", NASA/GRC, 1997, http://acts.grc.nasa.gov/library/docs/gsn/welchpaper.pdf

[5] J Macker, R B Adamson, "The Multicast Dissemination Protocol (MDP) Toolkit", IEEE, 1999, http://manimac.itd.nrl.navy.mil/MDP/MdpToolkitOverview.ps.gz

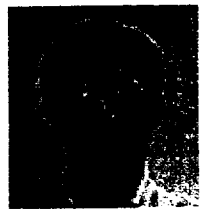
[6] J Rash, R Cassanta, K Hogie, "Internet Data Delivery for Future Space Missions", NASA Earth Science Technology Conference, Pasadena CA, June 2002

[7] D. Israel, K Hogie, R Parise, E Criscuolo, "Space Communications Demonstration Using Internet Technology", International Telemetering Conference ITC/USA 2002, San Diego CA, October 2002

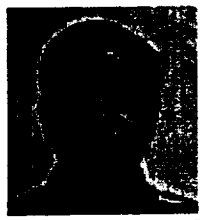


James Rash *– Goddard Space Flight Center – Mr. Rash currently manages the Operating Missions as Nodes on the Internet (OMNI) project in the Advanced Architectures and Automation Branch at NASA's Goddard Space Flight Center. He also leads development of formal methods capabilities with respect to agent-based systems. Previous assignments have included development of systems applying artificial intelligence and evolutionary programming techniques.*

Edward Criscuolo Jr. *joined Computer Sciences Corp. in 1991 as a Senior Computer Scientist working for the Goddard Space Flight Center. In that time, he has been the task lead for a number of spacecraft ground system projects that span many aspects of Goddard space missions, including Planning & Scheduling systems, spacecraft command management, and level-0 processing of telemetry and science data. In 1999, Mr. Criscuolo joined the OMNI project as a senior project member, where his duties include systems analysis, systems engineering, top-level design, prototype development, and backup technical lead.*



Keith Hogie *- Computer Sciences Corporation - Mr. Hogie has an extensive background in designing and building satellite data processing systems, control centers, and networks at GSFC. He has developed ground data processing systems and control centers for over 14 spacecraft over the last 25 years at NASA/GSFC, and led the development of the NASA Internetworking Laboratory Environment in 1990. He is the technical leader of the OMNI project at GSFC where he is applying his networking and satellite background to develop and demonstrate new communication technologies for future space missions.*



Dr. Ron Parise *- Computer Sciences Corporation - In 1984, Dr. Parise was selected as a payload specialist astronaut and was involved in mission planning, simulator development, integration and test activities, flight procedure development, and scientific data analysis. He has logged 615 hours in space as a member of the STS-35 and STS-67 crews. In 1996 Dr. Parise assumed a communications engineering support role for Mir, International Space Station (ISS), and the X-38 project. In 1997 Dr. Parise also began working with the OMNI project as a scientific liaison and systems architect.*