Distributed Operations for the Mars Exploration Rover Mission with the Science Activity Planner

Justin V. Wick, John L. Callas, Jeffrey S. Norris, Mark W. Powell, Marsette A. Vona, III
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA

Abstract—Due to the length of the Mars Exploration Rover Mission, most scientists were unable to stay at the central operations facility at the Jet Propulsion Laboratory. This created a need for distributed operations software, in the form of the Distributed Science Activity Planner. The distributed architecture saved a considerable amount of money and increased the number of individuals who could be actively involved in the mission, contributing to its success.

TABLE OF CONTENTS

1. INTRODUCTION

The Mars Exploration Rover Mission has been an unqualified success. At the time of writing, both Spirit and Opportunity have exceeded their operational lifetimes by a factor of three. Both rovers continue to roam Mars, returning a wealth of valuable new information to Earth.\

The unprecedented length of the Mars Exploration Rovers mission created many challenges for mission planners. Although the original architecture of the mission planning system was intended to be distributed in nature[1], budget constraints did not allow for the development of this capability. As a result, the planning software was designed in such a way that it was heavily reliant upon internal computing resources at JPL, making it unusable at remote sites.

In March 2004, as the primary mission for both rovers drew to a close, it became evident that both rovers were likely to continue operating long past their original 90 sol lifetimes. Faced with the reality that mission scientists would shortly begin departing from JPL to their respective research institutions, the decision was made to change the system to accommodate the participation of scientists at remote sites. Because the Science Activity Planner was the primary tool used by scientists in the high level planning process, an effort was begun to adapt SAP to use outside of JPL and provide a collaborative framework for scientists to operate it in.

2. CENTRALIZED OPERATIONS

During the centralized phase of the MER mission, scientists were gathered in a single building at JPL, and used the planning and analysis software in a specially configured computing environment, which was called the Flight Operations System. The Flight Operations System was the platform for the Ground Data System (GDS) software that drove the data processing and planning processes for the mission.

The Science Activity Planner (SAP) is a GDS tool that is used to perform dual roles facilitating manual science and engineering-level data analysis, and planning the daily actions of each rover in a coarse, high level fashion. This tactical decision making process is similar to that practiced in the FIDO field tests [2].

During each day the tactical process starts with a science meeting during which scientists are briefed about the current situation. After this, the scientists break up into "theme groups" such as "atmospheric" or "soils" or "long term planning" and work in parallel, using SAP to construct sequences of instructions for the rover that reflect their scientific goals. During this several hour period, the data is analyzed within SAP, points in space are designated as targets for the rover's actions, and the potential plans are put together. After this time, the final plan is debated and assembled in the Science Operations Working Group (SOWG) meeting. Possible scientific observations from each group are ranked according to importance. After a structured debate, the accepted observations are arranged together in SAP to meet the daily energy, time, and bandwidth budgets that have been established by the engineering team. The final merged plan is then delivered for further refinement and processing downstream to be converted to the actual sequence of instructions sent to the rover.

Collaboration within the system was facilitated by a homogeneous computing environment consisting of custom-built workstations running the Linux operating system.

There was a central Network File System server (the OSS) for each rover (MER-A and MER-B) and also a central SQL database server for each. Because all downlinked data and planning information were kept in these two central repositories, collaboration was simple. When a scientist saved a plan file, it was immediately available to all others to be analyzed and merged. Target designation, critical to the planning process, was also synchronized with low latency via the central SQL server. All science workstations were guaranteed to have access to the exact same set of data.

3. Moving to Distributed Operations

Due to budget and lifestyle constraints, the mission was shifted to a new, more distributed mission architecture. The cost of keeping relevant scientists on location in Pasadena was prohibitive, and many of the scientists and engineers had family elsewhere in the world for which they were responsible. Because of this, the decision was made to create an environment in which scientists could tactically plan with SAP at remote sites. This software environment would have to:

- Make planning-relevant downlink data available to the remote scientists in a timely fashion.
- Allow scientists to interactively share targets designations.
- Facilitate the sharing of plan files that contain the scientific observations for the day.
- Dynamically create indexing metadata of available data products to make it available in SAP.
- Maintain operational security through use of encryption, authentication, and firewalls.

It was decided that the best possible action was to closely replicate the JPL software environment, rather than change SAP itself. SAP expects a highly structured filesystem database containing images, range data, three dimensional meshes, spectral data records, coordinate frame information, planning constraints, and plan files. Because no available network file system server was fast enough to be used by SAP interactively, the relevant data sets would have to be mirrored locally. This also meant that the indexing of that data (which is how SAP knows what information is available on the filesystem) would also have to be done locally. Also, the sharing of targets and plans presented a challenge, as the servers hosting the plans and targets were not accessible outside of JPL.

Data Synchronization

The first matter was to arrange for the data to be delivered to the remote SAP workstations. SAP expects data to exist in a highly structured, hierarchical system of folders, numbering well over a million for each rover. This filesystem database is known as the Operational Software System, or OSS. The folders separate data by sol (martian day), instrument, and data type. The job of the data synchronization subsystem was to replicate the internal filesystem database of downlinked data on client workstations around the world.

The first solution to this problem that was developed utilized an open source program known as RSYNC, which can synchronize files and directories recursively between machines, through a secure ssh tunnel. A daemon was created that repeatedly synchronized the directories for recent sols with a central server. The central server itself was to be filled with the SAP-relevant data from the operational NFS servers.

The problem with this approach was that because it relies heavily on polling, and tens of thousands of files and directories had to be recursively compared. It was decided that it would place too much load on the server to have an acceptably low latency for data delivery. Worse, overloading issues were already a severe problem on the operational NFS server, and it was decided that this solution would most likely exacerbate the situation.

It was then decided to create a second data synchronization solution, utilizing the JPL Multi-mission Image Processing Lab's (MIPL) File Exchange Interface (FEI). FEI is a system that MIPL uses to automatically push out data to remote sites, such as research institutions or museums. While it supports polling and client-initiated downloading, it also has an event-driven server-push mode that relies on the "subscriptions" of a client to a set of file types. Because this system has a very low latency (on the order of 2 seconds within the JPL network) and is very well load balanced, this was chosen.

The main problem associated with this approach was that FEI does not keep track of the path in the filesystem to the directory where a particular file came from – this data would have to be reconstructed. In addition to this, FEI contains a large number of files that cannot be used by SAP and are not relevant for tactical planning. Because of the low bandwidth at many remote sites, the files would have to be filtered for relevance prior to downloading. The system also had to allow for the gathering of archived files from specific sols of interest – a feature not natively supported by FEI.

The final solution was to have two methods of getting files – an automatic subscription program, and a manual archived file retrieval program. Both programs used a filter to determine whether or not a given file was desired based on its relevance (and in the case of archival data, whether or not it fell in a specified range of sols). Also, a script was assembled that could sort the files into their final locations based solely on the file names. This was made possible by the fact that the file names systematically encode the data type, instrument name, time acquired, and from which rover the data was obtained.

Each workstation established a connection with the FEI server, and signed up to be "notified" when files in a relevant "filetype" were made available. This notification was pushed from the server to the client, at which time the client decided, based on the filename, if the file was desirable. If the file was wanted, it was retrieved from the server and then sorted into the filesystem. This system has latencies on the order of minutes or less, and has nearly idea bandwidth use (the server/client messages are very short).

Obtaining access to archival data was somewhat less straightforward. That program, given a rover designation (A or B, for Spirit or Opportunity) and a desired range of sols, downloads an entire roster of all available files in relevant filetypes. It then filters the names of files to find those which fall into the specified range of sols, and also do not currently exist on the local filesystem. This roster listing process is very inefficient and takes several minutes, however downloading the data can take hours, so the overhead is acceptable.

The final step in the data synchronization is the Data State Manager Daemon – a daemon process that scans available downlinked data products and creates a comprehensive index of what data is available. Every thirty seconds the most recent sols are scanned (and occasionally older sols, according to a probabilistic algorithm) to see if new data has been made available. When new data is discovered, it is processed and incorporated into the index, making it available for SAP. A nearly identical process is run at JPL, where the cost of all open SAP instances scanning each sol would have been prohibitive.

Target Synchronization

Target synchronization was another vital component of the distributed SAP system. At JPL, targets were synchronized between machines by storing them in a central SQL server. The various SAP instances would poll the server every two seconds, checking timestamps in the database to see if new targets had been created, or if old ones had been modified. There were no security issues because the database was not accessible from the outside world, and all individuals using computers that could access the database were cleared to designate targets.

In a distributed setup, however, everything changed. It was not going to be possible to make the central JPL target server available to machines outside of JPL for security reasons, however each remote site had to be able to see the same targets as users at JPL with minimal latency. Moreover, there had to be a method to take targets from outside JPL and import them to the internal JPL server. This entire process was required to be as low-latency and automatic as possible, while maintaining operational security.

The solution that we arrived at was that there should be a secondary, "external" SQL server that would be accessible to authorized machines outside of JPL. A script at JPL forwarded changes and new additions to the JPL internal target database out to the external server every few seconds. Because of the nature of the database, it was acceptable for targets to exist in the external database but not in the internal database without causing any problems. Plan files, however, reference targets (to decide where to drive, or aim a camera, etc). If a plan were brought into JPL that referenced an external target, that target would have to be manually imported by a script at JPL. That script would have to then extract a static copy of the target from the plan file text. This process was considered secure because it required a human in the loop to verify that the target was valid. Also, the external server was protected by a strict firewall that only allowed access from a set of secured university computers that were certified as part of the planning process. The data from JPL was encrypted using an SSH tunnel, with public key authentication.

A final consideration for target sharing was the complication that was caused by SAP's use of MySQL database polling – the newly changed entries in the JPL target database had to have a timestamp in the remote database that would cause the remote SAP clients to notice the change. Due to various internal details of the SAP client and MySQL servers, these timestamps had to be adjusted into the future before being sent to the external targets server.

Plan Sharing

The issues associated with plan sharing were similar to that of target synchronization in that the central server (in this case, the internal NFS server at JPL) was not accessible to the outside world. Also, there were similar security concerns – plan files coming out of JPL automatically were not considered to be a security issue, however no one from outside JPL could be able to insert a plan file into the normal planning directories inside JPL.

The solution that was decided upon was that there should be two repositories for plan files, one inside JPL (the NFS server) and one outside JPL. These two repositories would automatically synchronize, however no user outside JPL could be allowed to write a file that would propagate to a normal planning directory inside JPL. Instead, users outside JPL would have to place plans into special "external" directories. The planning directories inside JPL for each sol had names such as "apxs" or "soil", etc, broken down by group, and each containing an additional named "working" directory. The planning directories were modified by adding another directory named "external" in each subgroup directory. A user outside JPL could submit their plan to the central external plan server, but only if it resided inside an "external" directory.

The majority of synchronization was automatic. JPL's NFS server was considered the canonical source for "internal" plans. Every 30 seconds the next 5 sols worth of internal plans were sent to the external server. Every 30 seconds or so, those same sols were synchronized from the external server to the SAP workstations at each institution. However, because there was no single canonical source for plans created at an institution, it was decided that submission of an "external" plan to the central server would be a manual process. Once an "external" plan was submitted to the central server, within 30 seconds it would be copied to the same directory on the JPL NFS server, to be seen by those at JPL. This is how planned observations that were created outside of JPL could become part of the final plan at the SOWG meeting held at JPL.

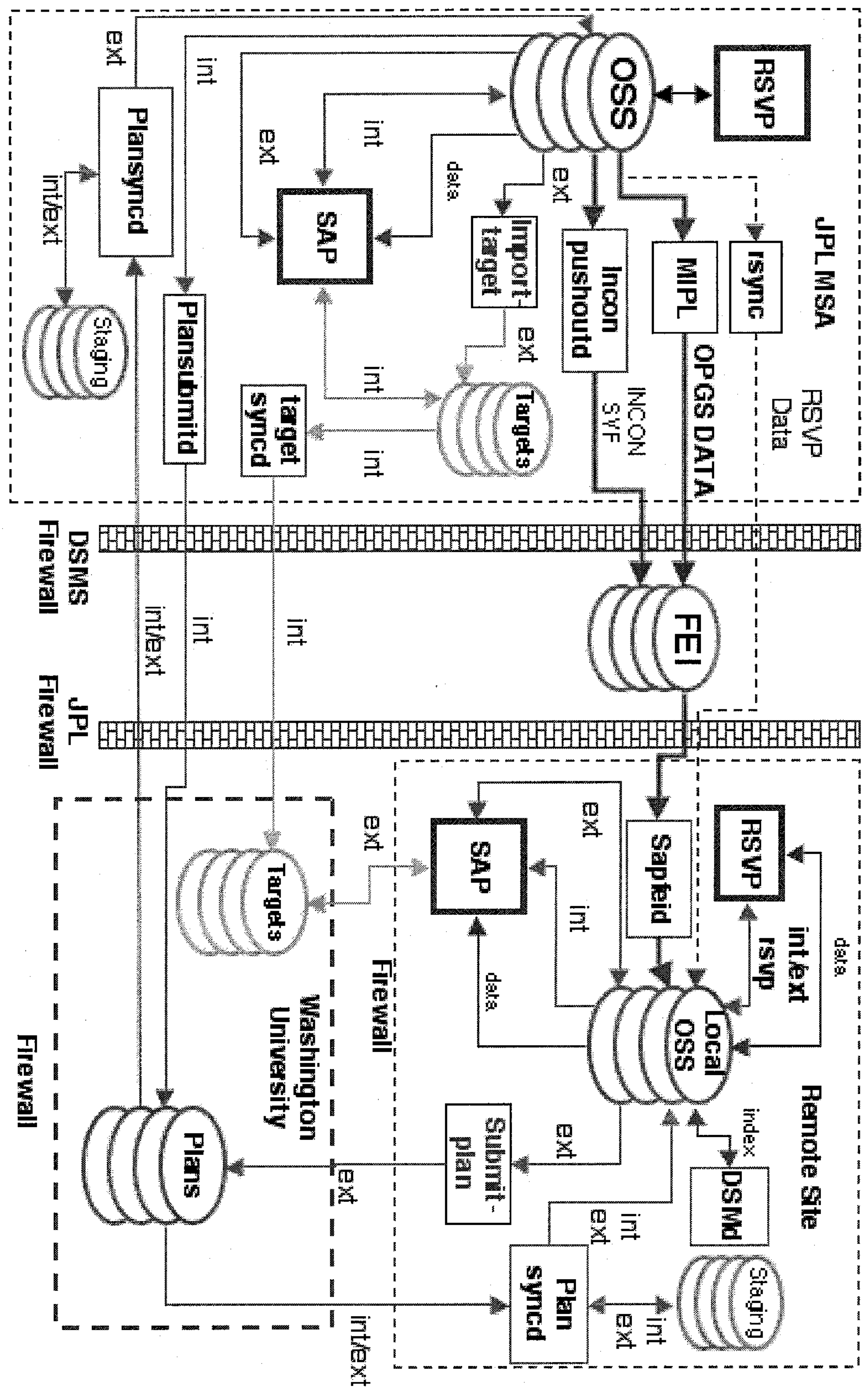
JPL MSA
RSVP
OSS
rsync
RSVP Data
MIPL
OPGS DATA
Incon pushoutd
INCON SYF
Import-target
SAP
Targets
target syncd
Plansubmitd
Plansyncd
Staging
int
ext
data
int/ext
DSMS Firewall
FEI
JPL Firewall
Remote Site
RSVP
int/ext rsvp
Sapfeid
SAP
Local OSS
index
DSMd
Staging
Submit-plan
Plan syncd
Targets
Plans
Washington University
Firewall
Firewall

Figure 1

4. ARCHITECTURE AND IMPLEMENTATION

Figure 1 illustrates the overall architecture of the system. The left side of the diagram represents the portion of the system running at JPL. The lower right corner of the diagram shows the servers at Washington University of St. Louis. Finally, the upper right represents each individual Distributed SAP workstation. The dataflow is illustrated by colored arrows: blue for downlink data, green for plans, and yellow for targets. The cylindrical shapes represent servers, and the named rectangles signify a process or collection of processes that are logically grouped together. A name in red signifies that the process requires a human intervention. Whether or not a target or plan being transferred was created inside or outside JPL is indicated by an "int" or "ext" label on the associated arrow. Names ending in "d" refer to "daemon" processes that run constantly in the background. RSVP is an engineering level planning program that is used by some scientists remotely, and uses much of the same data as SAP

Downlinked Data

To understand how the system works, one should first examine the downlink data flow (blue arrows). The Multi-mission Image Processing Laboratory (MIPL) is the source of all processed imagery used in this system. That, along with the "Inconpushoutd" – a daemon that pushes out initial conditions of the rover for a sol, the planning constraints, and coordinate frame information – supply the FEI server with the files that are needed for use of SAP outside of JPL.

After the files are sent to the FEI server, the clients are notified of the newly available files through the "Sapfeid", a daemon that handles all of the processes that wait for new files. If the files are deemed relevant, they are downloaded from the server into a temporary directory, and then put away into the local OSS (the local set of folders that hold the data for each sol). The new data is noticed by the DSMd (Data State Manager daemon), which then indexes it. After that the data can be accessed by SAP.

Target Data

The target dataflow is more symmetric – a target can originate either at JPL or at an external workstation. SAP instances at JPL create targets in the internal database. A JPL computer running the "targetsyncd" – the target synchronization daemon – takes newly generated targets and sends them outwards to the centralized target server at Washington University. Every time an external SAP client opens a plan from a given sol, it fetches the targets associated with that sol from the central server. The SAP client also maintains a polling thread that keeps looking for new targets being made on that sol.

A computer external to JPL can create a target in the external database, making it available to all other external SAP instances. If the target needs to be used inside JPL, an external plan file is saved and then submitted to the plan server; a copy arrives at the JPL OSS. A person, either at JPL, or logged in remotely, then runs the import-target script, giving it the plan file, and the name of the target to be imported. The import-target program reads the target data from the plan file, and then enters it into the local JPL database. Any open internal SAP instances can then see the new target, and it can be used in the final, official plan.

Planning Data

The final component of the system is the shared planning dataflow (green arrows). Just like shared targets, there are two separate places where plans can be generated – internal to JPL by SAP, or external to JPL by SAP. Inside JPL, they are kept in special directories on the OSS. An automated process, Plansubmitd, polls the OSS every 30 seconds to check for new plans, or newly modified plans, and uploads them to the external planning server at Washington University. A similar process, Plansyncd, polls the server for new or newly modified external plans to be imported. Plansyncd imports all changed plans to a staging area, but only copies external plans to the actual OSS for security reasons. This prevents anything submitted to the external server from affecting the internal plans without intervention from a human at JPL.

The right side to this dataflow concerns the remote sites. If a plan is created or modified at a remote site, and the user wants to share it with the rest of the distributed SAP users, the "submit-plan" script is run. This sends the file to the server (overwriting any older version of that file if it previously existed). Also, a slightly different version of the Plansyncd is running in the background. It is identical the JPL version, except that it copies both internal and external plans to the local OSS.

Programming Languages Used

All of the daemon programs were written in Perl 5, and utilized utility shell scripts. The import-target program is a combination of a Perl frontend and a Java backend. Perl was used because the system is tied heavily to the underlying OS, and it made invocation of Unix commands and file manipulation particularly easy. Also a large amount of the work done by these programs involved text parsing.

5. TECHNICAL CHALLENGES

The technical challenges in this project were many and varied. Most of the challenges involved reliable communication between all of the parts of the system, atomicity of transactions, and server load. Also, out of necessity, many parts of the system used software in ways that were not originally intended.

The most common technical challenge of the entire project was the large set of problems created by repeated polling of filesystems and servers. Because the MER GDS has no centralized, common event-driven architecture, most of the components of the distributed system use some form of polling to handle propagated changes. Polling itself is not a significant challenge in software development, however the efficiency of the polling was a severe limiting factor in what design choices that were available, and it forced us to use nondeterministic algorithms for some of the less important parts of the system.

Our data indexing process, the Data State Manager (DSM), needed to poll tens of thousands of subdirectories of the filesystem every thirty seconds. This grew to the point where it was untenable, so a compromise was made in the system's design. Instead of scanning all sol data directories every 30 seconds, it would scan only the three most recent for new data constantly. The older directories would have a probability of being scanned each 30 second sweep such that about 95% of all sols would be scanned in a given 24 hour period. The use of nondeterministic algorithms was considered safe because older sols tended not to change often, and their changes tended not to be important.

Another example where polling was a bottleneck was the Plan Synchronization Daemon. The Plan Synchronization Daemon (plansync) relied heavily on polling of a central server. Plansync used the RSYNC client tunneled through SSH, and rsync only permits one directory to be recursively synchronized per connection. Because of this, and the fact that the first five upcoming sols had to be synchronized every thirty seconds, each requiring a separate connection, the ssh authentication server on the central planning server became intolerably slow. While plans still propagated, it was at a reduced rate, and often connections to the server were rejected due to the overload. As of this writing, we plan to replace this polling process with a manual process due to the incredible load it places on the server.

A different issue encountered was reliable communications through a highly heterogeneous network environment. There were a lot of very complicated firewalls involved – two levels at JPL, at least two at Washington University of Saint Louis, and usually between one and two firewalls at other institutions. SSH tunneling made communications through these firewalls possible, however this required authentication keys to be distributed. Network failures were not entirely uncommon, and temporary workarounds had to be set up in the event that a server was not reachable. Server load and reliability was often the deciding factor for the success of the Distributed SAP system.

One of the biggest causes of bugs was the relative heterogeneity of systems running SAP outside of JPL. Inside JPL the software was run exclusively on Red Hat Linux 7.3 boxes, all of which contained identical processors and graphics cards. Outside, Red Hat Linux 7.3, 8, 9, Red Hat Enterprise Linux 3, and Fedora Core 1 were in use. This was a problem because it required different systems to use different versions of the FEI client, which was not fully tested on Fedora Core 1. Also, newer Linux distributions shipped version 5.8 of Perl, which has subtly different semantics for a few very important operations, such as regular expression matching. This lead to a few bugs involving data delivery, which were very difficult to track down.

Last but not least was the fact that some software components of the system were being used in ways that their creators had not intended; this sometimes put the system into odd states requiring manual intervention. The FEI server system was not designed, for instance, to notify clients if a file that already existed on the server was modified, only when new files were added. So, when certain important configuration files had to be pushed out, they had to be removed and then added to FEI. Also, FEI had no method of filtering files based on the sol they belong to, or specific details of the file type; this had to be implemented in one of the more complicated Perl programs that we created. The same goes for the lack of filesystem metadata preservation in FEI – the files had to be sorted by a Perl program, based solely on the name of the file – a fact that precluded the sorting of certain types of files accurately.

6. Mission Impact

The impact of the Distributed Science Activity Planner on the MER mission was very significant. By allowing scientists to analyze data and collaboratively plan at remote institutions, Distributed SAP was a primary enabling factor in the feasibility of the distributed operations architecture.

Transitioning to distributed operations has saved a considerable amount of money during the extended mission. Travel costs were significantly lower, there was a much reduced demand for temporary housing, and most scientists returned to using normal work areas at their home institution, freeing resources at JPL. The funding reduction itself is important because it is unlikely that many individuals could be actively involved with the planning process if all operations were conducted at JPL; the participating team would have to be very small, which would seriously reduce the science return of the mission.

This new distributed architecture has had negative impact on the mission as well – communications are much harder when people are not in the same room. Also, a significant amount of time was spent emailing screen shots back and forth, due to the fact that many mission computer programs were not designed to be collaborative over a distance. Much of the communications difficulties were mitigated by the use of teleconferencing equipment, web cameras, and Virtual Network Computing, and SSH. There are still aspects to the system that need improved, however, the net impact of moving to a distributed architecture is overwhelmingly positive.

7. CONCLUSION

The Distributed Science Activity Planner has contributed to the success of distributed operation for the Mars Exploration Rover mission. Scientists were able to analyze data and plan from their home institution, and collaborate with other scientists around the world. The distributed operations architecture has enabled a large science team to operate Spirit and Opportunity well beyond the original mission lifetime as they continue to return valuable scientific information to earth. Distributed MER operations will serve as a model for missions into the future.

8. ACKNOWLEDGMENT

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

REFERENCES

[1] Robert Steinke, Paul G. Backes, and Jeffrey S. Norris. Distributed mission operations with the multi-mission encrypted communication system. In *Proceedings IEEE Aerospace Conference*, Big Sky, Montana, March 9-16 2002.

[2] Paul G. Backes , Jeffrey S. Norris , Mark W. Powell , Marsette A. Vona , Robert Steinke , and Justin Wick. The Science Activity Planner for the Mars Exploration Rover Mission: FIDO Field Test Results. In *Proceedings IEEE Aerospace Conference*, Big Sky, Montana, March 8-15 2003.