

RAPID: Collaboration Results from Three NASA Centers in Commanding/Monitoring Lunar Assets

R. Jay Torres
Jet Propulsion Laboratory,
California Institute of
Technology
4800 Oak Grove Drive
Pasadena, CA 91109
818-393-0037
Jay.Torres@jpl.nasa.gov

Mark Allan
NASA Ames Research
Center
Mail Stop 204-2
Moffett Field CA, 94035
650-604-0537
Mark.B.Allan@nasa.gov

Robert Hirsh
NASA Johnson Space
Center
2101 NASA Parkway
Houston, Texas 77058
765-532-8922
Robert.L.Hirsh@nasa.gov

Michael N. Wallick
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
818-393-0037
Michael.N.Wallick@jpl.nasa.gov

Abstract—Three NASA centers are working together to address the challenge of operating robotic assets in support of human exploration of the Moon^{1,2}. This paper describes the combined work to date of the Ames Research Center (ARC), Jet Propulsion Laboratory (JPL) and Johnson Space Center (JSC) on a common support framework to control and monitor lunar robotic assets. We discuss how we have addressed specific challenges including time-delayed operations, and geographically distributed collaborative monitoring and control, to build an effective architecture for integrating a heterogeneous collection of robotic assets into a common work. We describe the design of the Robot Application Programming Interface Delegate (RAPID) architecture that effectively addresses the problem of interfacing a family of robots including the JSC Chariot, ARC K-10 and JPL ATHLETE rovers. We report on lessons learned from the June 2008 field test in which RAPID was used to monitor and control all of these assets. We conclude by discussing some future directions to extend the RAPID architecture to add further support for NASA’s lunar exploration program.

complicated, as they will all have a different communication structure and interaction interface. We address these issues through our Robot Application Programming Interface Delegate, or RAPID Project. This paper discusses the development of the RAPID Protocol, a standard "language" by which assets communicate; and the RAPID components: 1) RAPID Workbench, 2) RAPID Bridge, 3) RAPID Middleware Components.

One of the greatest challenges is to be able to command and monitor lunar robots developed at different NASA centers via a single software tool. NASA’s Human Robotic Systems project is integrating and testing several robotic assets developed by different NASA centers. Each of these assets was developed according to differing specifications. This proposes a problem since each robot has its own: 1) commands and messages 2) communication pipeline 3) system architecture 4) commanding and monitoring tool. RAPID was developed to address these issues. The RAPID Workbench is a software tool that gives the ground operator the ability to command and/or monitor lunar assets.

The goal of this paper is to present how obstacles were overcome in order to implement and utilize the RAPID Workbench. It describes the development, testing, and field trial environments as far as getting the RAPID interface implemented. Through the development of the RAPID protocol each lunar asset can be monitored and commanded via the RAPID workbench. The protocol has been designed such that any tool integrating into the RAPID API can be used to command and/or monitor any lunar asset.

In June 2008 teams from several NASA centers converged at Moses Lake, on an off-road vehicle site in central Washington State for a two-week field test to characterize the performance of several candidate lunar surface robots. The test team included robots from the Jet Propulsion Laboratory (JPL) in Pasadena, California; Ames Research Center (ARC) in Mountain View, California; and Johnson Space Center (JSC) in Houston, Texas. Lessons learned from the Moses Lake field test will be incorporated into the evolving design of the lunar operations system, and will be tested at subsequent field trials [2].

TABLE OF CONTENTS

1. INTRODUCTION.....	1
2. RAPID PROTOCOL	2
3. RAPID IMPLEMENTATION	3
4. MOSES LAKE FIELD TRIAL.....	8
5. CONCLUSIONS AND FUTURE WORK	9
REFERENCES	10
BIOGRAPHY	10

1. INTRODUCTION

NASA’s initiative to return to the Moon has brought about many interesting challenges to overcome. Multiple NASA centers have been working on different robotic assets to aid in this goal. Each center has different design strategies and techniques. As a result, learning to manage all of the different assets, or managing them at once can be quite

¹ 978-1-4244-2622-5/09/\$25.00 ©2009 IEEE

² IEEEAC paper #1307, Version 8, Updated 2008:11:02



Figure 1-1. Assets Family Portrait, Moses Lake, WA

There were many barriers that had to be overcome in order to get a successful field trial. Even after those barriers were broken, many lessons were learned from the experience such that the issues should return in subsequent tests. The barriers include but not limited to communication issues between centers and coordination of developers between all centers.

During the field trial, lunar robots and space suits (assets) were commanded and monitored through RAPID. Figure 1-1 displays the entire set of lunar assets at Moses Lake, WA. The following is the list of lunar assets that, unless otherwise noted, were commanded and monitored during the field trial: 1) 2 Astronaut Suits; monitored only (JSC) 2) The ATHLETE robot (JPL) 3) The CHARIOT lunar vehicle (JSC) 4) The K10 robots (ARC)

2. RAPID PROTOCOL

Motivation—The requirements of commanding and monitoring lunar assets motivated the development of the RAPID protocol.

The RAPID protocol arose from a previous iteration called Astronaut Interface Device (AID). The AID interface was used as a way for Astronauts to command and monitor assets[5]. Figure 2-1 shows a screenshot of the AID interface program. It was used in limited forms in previous field trials. The evolution to RAPID was such that the interface was more general and it also allowed for a more feature rich capabilities.



Figure 2-1. AID Graphical User Interface

Requirements—The requirements for RAPID are as follows:

- 1) Multi-center communication
- 2) Asset control language agnostic
- 3) Real time remote communication

Multi-center communication—The system must be able to monitor and command regardless of center/asset ownership, for example a JSC asset should be controllable by a JPL operator located either at the field trial or JPL.

Asset control language agnostic—The system must be able to handle assets developed in different control languages. The JPL assets are developed in JAVA, while the JSC and Ames assets utilize C/C++.

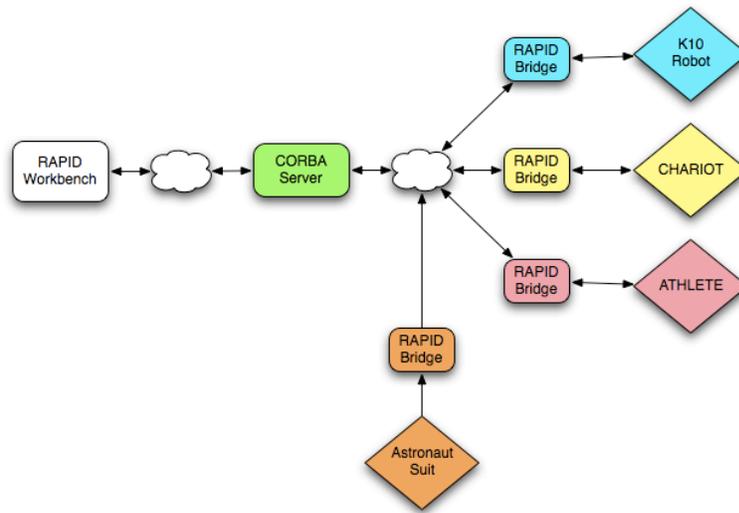


Figure 3-1 RAPID Connection Diagram

Ames K10 – The K10 robot’s native communication is through the use of CORBA. This allowed for a seamless integration with the RAPID middleware. Since it was also in CORBA, the RAPID message types became a part of the native language understood by the robots. The underlying code was written in both JAVA and C++.

JSC Chariot – The Chariot vehicle utilizes sockets as its main communication pipeline. Both the flight software and the existing ground software was written in C++. The design was to use the JAVA components of RAPID and have it coexist with current CHARIOT software.

JSC Suits – The astronaut suits only sent telemetry status information. To accomplish this, a RAPID bridge was created that connected the suits to the RAPID pipeline.

JPL ATHLETE – The existing commanding and monitoring tool used for ATHLETE is the ATHLETE Workbench [1]. The ATHLETE Workbench communicates in the ATHLETE native language through the JMS and a direct socket connection [2].

RAPID takes all these different types of communication configuration and unifies them through the middleware and RAPID command dictionary.

Real Time Remote Communication—This requirement includes having a robust system with the following specifications: 1) commands cannot be dropped 2) telemetry from the asset can be dropped 3) time delay must be taken into account.

An operator console called the Exploration Technology Development Program Multi-Center Cockpit was developed to perform remote operations [3]. The console is able to send commands to the assets as well as monitor remote telemetry. A monitoring station at JSC was also used to

display all asset real-time telemetry data simultaneously. The station is discussed in Section 4 of this paper.

3. RAPID IMPLEMENTATION

The RAPID Workbench software, developed at JPL, is meant to provide operators with a ground operations tool in order to command and monitor multiple assets through RAPID. The RAPID interface is collaboratively developed across the three NASA centers.

The RAPID Workbench is based on an existing tool called the ATHLETE Workbench. The ATHLETE Workbench is used to command and monitor the ATHLETE robot. There are two main differences between these software tools. The first is that the ATHLETE Workbench contains many ATHLETE specific features that are not required for generic robot software. The second difference concerns the messaging protocol used. The ATHLETE workbench utilizes the Java Messaging System (JMS) since all of the components communicating with the workbench are in the Java programming language [2]. The RAPID workbench, on the other hand, had to be designed to be language independent. This gives the freedom for existing components, written in languages other than Java, to be able to be compatible with the RAPID workbench. The Middleware Component section describes how CORBA was used in the RAPID workbench to allow messaging between different software. Figure 3-1 shows the diagram of the RAPID Workbench and communication between the different assets.

Message Set – The following the basic set of message types that are implemented in the RAPID protocol.

- 1) **Status** – reports the power reserve of the asset as well as information about the current task.

- 2) **Position Data** – provides the location vector, orientation, heading and velocity of the asset.
- 3) **Joint Data** – gives detailed information on the robot's current joint configuration if applicable to the asset.
- 4) **Picture Data** – returns an image acquired by the asset.

Command Set – The following is the set of commands that all RAPID assets must implement, if appropriate for that asset.

- 1) **Get Status** – returns one or more of the messages above to indicate the current status of the asset.
- 2) **Robot Move Commands** – are the basic commands to navigate the robot in a particular direction, and stop when necessary. These commands do not make sense for all assets, such as the suits, and therefore do not need to be implemented.

into the RAPID infrastructure. Figure 3-2 shows a diagram of the CORBA Notification Service.

CORBA allows the RAPID Workbench and the RAPID bridges to perform two types of communication: 1) push model 2) pull model. Both depend on the Notification Server holding a CORBA Event Channel. Any client of the CORBA middleware registers with the Notification Server. That client also registers with the Event Channel in the Notification Server.

Push Model – The RAPID Bridge publishes messages through the CORBA notification server. Any client that desires to listen to the messages, can register with the notification service and receive published messages.

The asset, through the RAPID Bridge, registers with the Event Channel. The asset is considered a push supplier because it supplies the Event Channel with information. The asset publishes telemetry type messages in this channel at a given rate, i.e. updated position information is published at 1Hz; status information can be published at 4Hz. Each asset will register with its own Event Channel, publishing RAPID

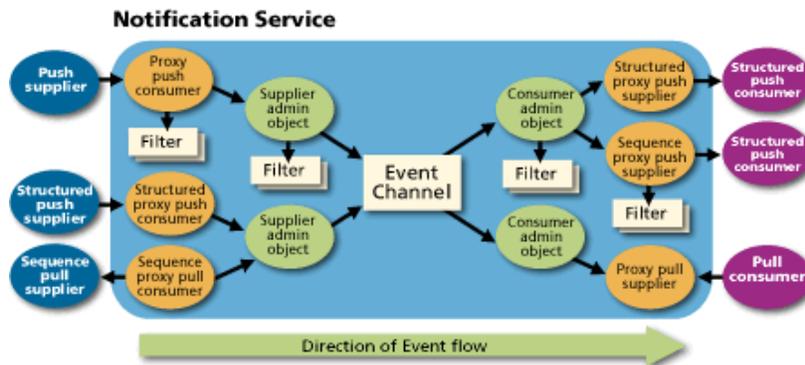


Figure 3-2. CORBA Middleware Notification Service [4]

- 3) **Robot Specific Commands** – are commands that are specific to a particular asset, such as placing a robot's joints into a particular configuration.
- 4) **Camera Commands** – instructs the camera to point in a given direction, take a picture, etc.; and send the acquired image back through picture message above. These commands only need to be implemented when the asset contains a camera.

Middleware Layer— The middleware component used for RAPID for the purpose of the June 2008 Field Trial was CORBA. The reasoning for using it was it was inherited from the AID interface used in previous field trials. Many existing designs using CORBA and AID were modified to fit

messages to any subscriber. The map view utilizes the information published to display position information for all assets in a single display.

The RAPID Workbench is also registered with all Event Channels for all assets. Relevant information is displayed based on which asset the operator is currently controlling; and messages are filtered as such. In this model, the RAPID workbench has no control over when messages are being published by the asset. If a command is sent via the Event Channel, it is the asset's responsibility to ensure that the command is meant for that asset. Any client listening to the Event Channel will be able to obtain the response message published by the asset.

Pull Model – The RAPID Workbench can communicate to the robot by connecting to the asset itself, via the RAPID

Bridge, and passes messages directly between components. This is different from the Push Model in that the RAPID Workbench can send commands and immediately receive a response from the asset. The capability was used to obtain immediate status information instead of waiting for the asset to publish telemetry information. The messages passed are not broadcast to other clients connecting to the Notification Service. The RAPID Workbench connected to the asset through the pull model, sends commands and the response from is not broadcast to the Event Channel. The usefulness of this is that only one RAPID Workbench can control the asset at one time. The asset sends the response of the command directly to the connected RAPID Workbench. Simultaneously, the asset continues to publish messages on the Event Channel providing status and

position information to any client.

Field Trial Topology— To simplify the connections to the middleware component, a RAPID federation using CORBA was implemented.

The federation configuration allows multiple CORBA event services to forward UDP packets to a central server. This allows clients the simplicity of only having knowledge of one CORBA server instead of having to connect to multiple servers. Commanding was also done through the federation. Clients send commands to the central server and it handles the connections to the assets. The assets then publish telemetry data to the server and any client listening will get

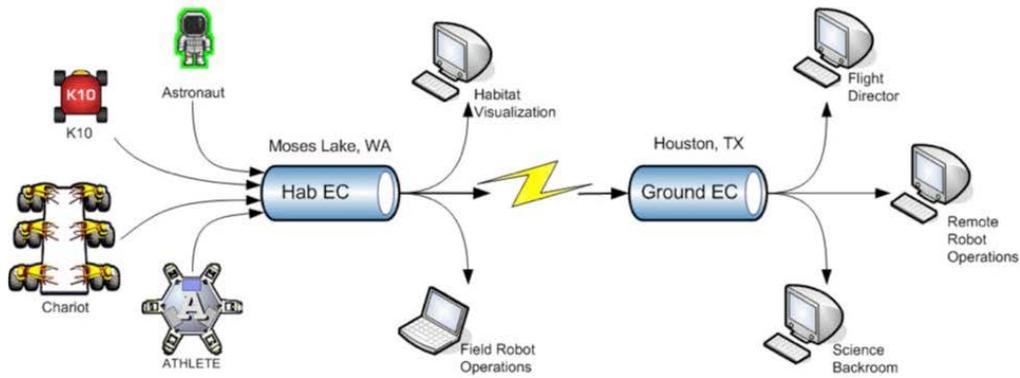


Figure 3-3. Field Connection Topology

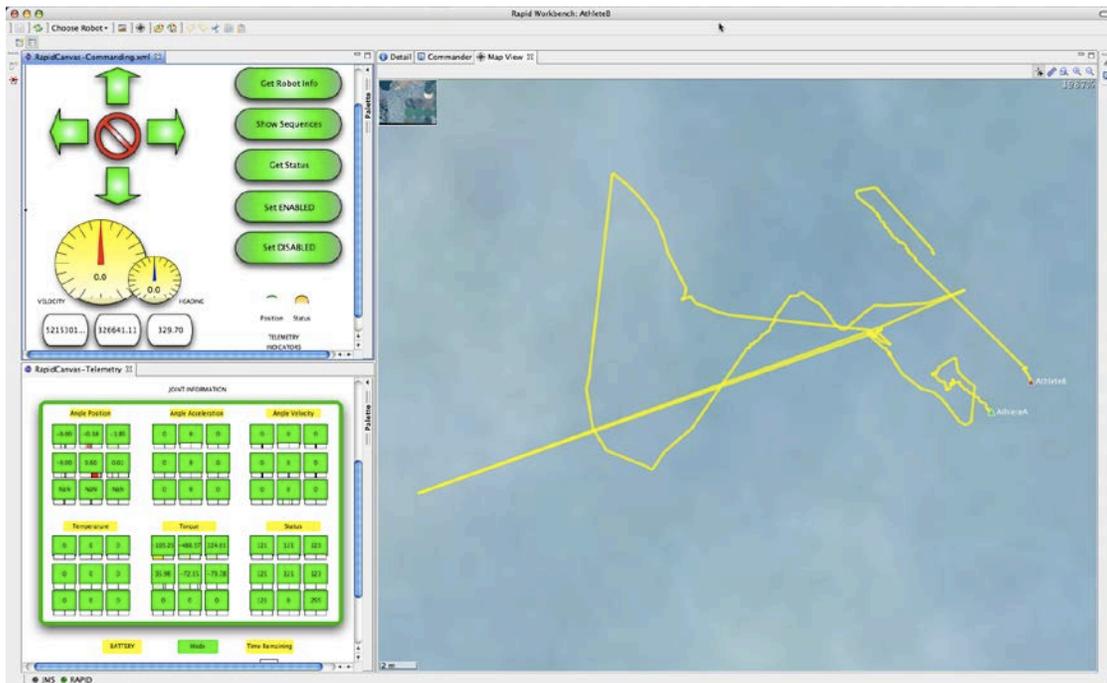


Figure 3-4. RAPID Workbench

notified of the message.

Figure 3-3 shows the central server at Moses Lake, WA. The Hab EC is the command habitat where clients and assets connect. The figure also shows the Ground EC or the remote operations federation at JSC in Houston, TX. The Ground EC component is another federation connecting to the Moses Lake site. Its purpose was to handle the remote traffic while allowing local client computers to have a central server to connect to.

The RAPID Workbench— The workbench, shown in Figure 3-4, has the following features that can be used by all robots: 1) Mapping features 2) Commanding graphical user interface 3) Ability to receive telemetry information and display them to the operator 4) ability to specify on a single workbench, which robot the operator is interested in viewing data from.

Mapping Features— The mapping capabilities of the RAPID workbench provide the operator with visual cues as to where the assets are located. The benefit of having this is that the operator can have a general sense of the proximities of the robots. This is important since looking at raw values sent from the assets as to its position data is difficult to comprehend without a visual tool.



Figure 3-5. Mapping Visualization in the RAPID Workbench

The Map View displays a high-resolution photo of the site where all the assets will be located. Moses Lake, Washington was the site of the June 2008 field trial. Therefore, the image displayed was of the Moses Lake testing site.

The Map View takes the RAPID messages pertaining to position information, from all assets received by the RAPID Workbench and overlays a glyph representing the asset on the map. The history of the traversal is also displayed by overlaying a line representing connecting two consecutive RAPID position data.

Commanding User Interface - Commanding the asset is done through the telemetry canvas. The telemetry canvas is composed of clickable widgets that provide the operator an interface to send RAPID commands. The purpose of this is so that the operator does not have to type in the RAPID command and all the required parameters.

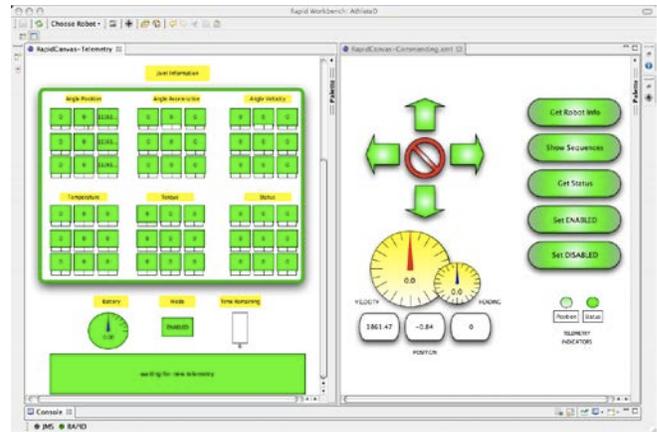


Figure 3-6. Commanding and Monitoring Telemetry Canvas in the RAPID Workbench

If the operator is comfortable with entering commands through a command line, the RAPID Workbench provides that capability. The operator will open up a Commander View and the command prompt will be available.

The smooth operation of a complex vehicle in a dynamic environment requires that the operator have an easy way to quickly ascertain the state of the vehicle and respond appropriately. The Telemetry Canvas (Figure 3-6) graphically and intuitively provides that feedback to the operator and facilitates commonly used commands and error recovery techniques [2].

Telemetry Data Representation— The RAPID messages received are presented to the operator in the form of Graphical User Interface widgets. These widgets provide information based on the telemetry received from the asset. There are indicators notifying the operator when a particular value has exceeded its threshold. Other widgets show the position information as well as its heading and velocity through dials. There is also a widget that displays what the asset is doing at that particular time.

These widgets are populated by the RAPID messages received from the asset. The RAPID Workbench determines what type of message is has received and represents that information accordingly.

Multi-robot Workbench – The polymorphic capabilities of the workbench allows the operator to command and/or see telemetry information from any specified asset. This eliminates any confusion the operator may have as to which asset they are dealing with. The operator specifies during

runtime, what asset they would like to communicate with. Once specified, the only RAPID messages displayed or sent will be from the specific asset. However, this does not affect the map display. As stated above, the map RAPID messages pertaining to the asset's position and displays it.

Native Robot Commands in RAPID – Certain commands cannot be universally created since not all assets will know how to handle them. RAPID provides a way to send native commands through the RAPID workbench. When the native command is seen in RAPID, it passes it along to the asset without having to do any interpretation. Currently, there is no error checking, therefore, the command is sent as is.

RAPID Bridge—The RAPID Bridge is the link between the asset and ground software. The sole purpose of the bridge is to translate asset specific messages to RAPID messages. A message in this case can either be a command from the ground software or information the asset would like to send to the ground software.

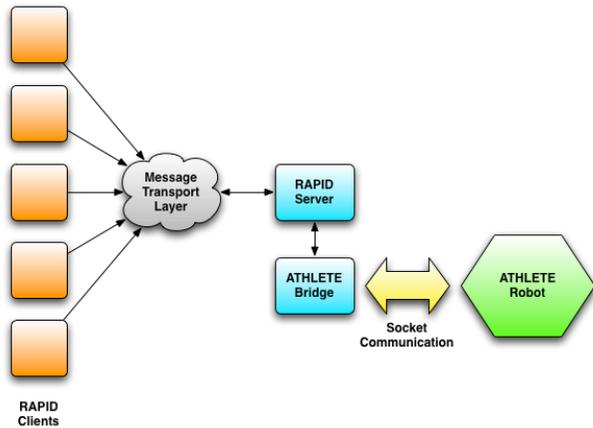


Figure 3-7. RAPID Connection Diagram for ATHLETE

Figure 3-7, shows the connection between the RAPID clients and its relation to an asset, the ATHLETE Robot in this case. Each RAPID client is separate ground software implementing the RAPID. For the purpose of the paper, they are considered separate instances of the RAPID Workbench.

Components - The diagram also shows the two components that make up the RAPID Bridge specific to the ATHLETE robot and they are the RAPID Server and the ATHLETE Bridge.

The RAPID Server takes the messages from the RAPID Workbench and translates them to the ATHLETE robot specific command. The Server then sends that command to the ATHLETE Bridge.

The ATHLETE Bridge is in charge of directly communicating with the ATHLETE robot via sockets. It sends the translated RAPID command to the robot.

When the robot has messages that should be passed back to the ground software, it sends the telemetry information to the ATHLETE Bridge. The Bridge sends that information to the RAPID Server. The server determines what type of information the telemetry data is for a set of RAPID types: 1) position information 2) status information 3) image data. Once it determines the RAPID equivalent, it converts the telemetry data into a RAPID message. This message is then sent through the middleware and received by the RAPID Workbench.

Each robot will also publish information regarding its current health, state and the status of a command. The robot sends messages that are uniquely identified by the type of information they contains. Clients can filter message traffic by choosing to listen to a specific type of message while ignoring other message types [2].

The main component that allows for multiple asset communication is the RAPID bridge component. The RAPID Bridge is the layer between the actual RAPID message and communication to the asset. The Bridge translates any message sent in the RAPID command set to a message the asset can understand. Conversely, any telemetry data the asset publishes is translated into an equivalent RAPID message so that any software listening via RAPID will be able to understand the information.

Robot Specific Implementation—Each center has an implementation of the RAPID Bridge specific to their asset.

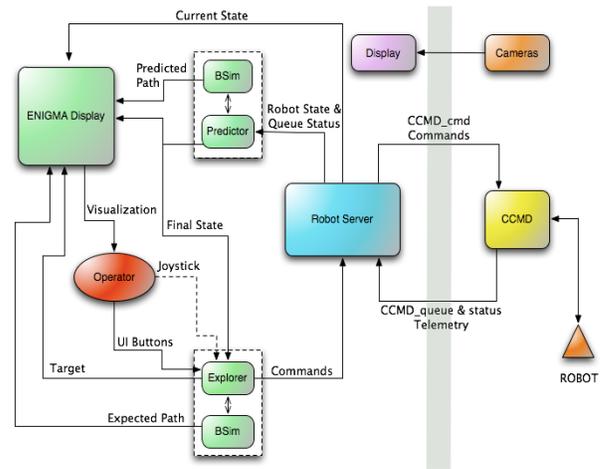


Figure 3-8. Connection Diagram for CHARIOT

Figure 3-8, shows the implementation for the CHARIOT robot. The Robot Server acts as the Rapid Server and the CCMD component is the CHARIOT Bridge.

4. MOSES LAKE FIELD TRIAL

The RAPID Workbench played a big role here because the operators at JSC used it to obtain RAPID information from the assets used it. Specifically, the lessons learned could be categorized into four types: 1) operator experience 2) Mapping and tracking all assets 3) Commanding. 4) RAPID middleware pipeline

Operator Experience – The operator experience was designed prior to the field trial. Still, it proposed a difficult scenario since the total number of assets tracked was more than doubled in the field trial compared to the dry runs. There was much data to monitor and each required space to be adequately visualized. The number of assets that required monitoring included five robots and two space suits.

The other issue was that how it became clear that monitoring and commanding multiple assets proved to be difficult to do for a single operator, no matter what tool was presented for use.

Mapping and Tracking Assets – This proved to be one of the significant features of the RAPID Workbench that operators found useful.

The Map view allowed the operator to view field trial site and see all assets position and heading in real time. It also allowed the operator to view this history of each of the tracked objects to determine possible missed target areas.

The Map view was instrumental in allowing the operator to plan a traverse.

Commanding – Although the commanding through the RAPID Workbench was limited, using it at the field trial provided feedback on how to improve the usage.

The person commanding must also do monitoring, but the RAPID telemetry messages should provide more useful information.

Commanding more than one robot at one time was very complicated and impossible. The RAPID Workbench



Figure 4-1. RAPID Workbench Actively Monitoring All Assets

The solution to monitoring all assets was to have the RAPID Workbench shown on an elevated platform displayed on multiple monitors (Figure 4-2). The operator in charge of monitoring the assets sat near the console and viewed the telemetry streaming from the assets.

The commanding operator was able to stay on their console and only deal with the asset they were interested in. The operator was still able to view the map view from the RAPID Workbench when required.

should provide improved indicators to notify the operator when constraints of any kind are violated.

The command dictionary must also be richer than the rudimentary commands. The field trial provided a proof of concept that will enable the development of more RAPID commands.

RAPID Middleware Pipeline – Setting up the RAPID pipeline also proved to be somewhat difficult. Initial issues crossing the firewall provided difficulty in getting RAPID to work. Once all the firewall issues and middleware

initialization issues were resolved, the pipeline proved solid and RAPID messages were successfully passed between the RAPID Workbench and the assets.

The experience gained here showed that more middleware diagnostic tools were required. It also showed the pipeline needed to be extremely robust to be able to handle clients and servers going offline due to technical issues.

5. CONCLUSIONS AND FUTURE WORK



Figure 4-2. Display Configuration of RAPID Workbench at JSC

The Workbench was used to remotely monitor asset activities at Moses Lake, WA.

The June 2008 field trial in Moses Lake, Washington provided the avenue for continual development on RAPID. The feature rich RAPID Workbench provided the tools the operators required to remotely command and/or monitor the assets. An operator was able to send and receive messages remotely in a timely manner.

The collaborative work between three NASA centers allowed for developing useful capabilities in the RAPID Workbench. Many of the capabilities, including the mapping features, were found to be vital to commanding the assets remotely. The lessons learned from the experience provided a path on how to make the intra-center asset communication better through the RAPID Workbench. We will continue to develop features that will enhance the operator's ability to perform desired tasks. This includes generating a larger RAPID command dictionary allowing the operator to command more complex tasks.

The mapping feature of the RAPID Workbench will also be enhanced to allow for point and click mobility commanding.

This will allow the operator to create waypoints on the map and the RAPID Workbench will generate the commands to the asset. Mapping will also be expanded to display indicators as to where cameras are pointing. The operator will point the camera on the asset to the target and the RAPID Workbench will send the appropriate commands.

As far as the RAPID protocol is concerned, there is room from improvement. The field trial did prove that RAPID

messages can be passed along assets and clients developed in different NASA centers. Utilizing the RAPID Workbench enabled an operator to remotely perform required tasks. The middleware configuration allowed for seamless communication between the Moses Lake site and the remote site at Houston.

Although, commanding assets through RAPID was successful, certain deficiencies were discovered. The command dictionary was lacking in many complex commands. It was clear that a richer command dictionary would have allowed operators to control the assets through intricate operations scenarios. Many native commands were sent to perform these tasks, but with a larger set of RAPID commands, it would not have been necessary to send native commands.

Native commands cannot be eliminated since RAPID commands cannot perform all asset specific tasks due to different asset configurations. However, if the RAPID

workbench understood the native commands, it could provide the operator help in generating the required inputs to the commands. Future iterations of the RAPID workbench software should be aware of some, if not all, native commands of assets such that command completion can be performed. This will allow the operator to concentrate on the task at hand and not deal with ancillary parameters.

The middleware performance showed there were aspects that could be improved upon. It should be noted that when CORBA connections were made, they were highly efficient and provided the best service for RAPID. Unfortunately, getting the services up and running and consistently connected took too much resources (including time and personnel) that a conclusion could be made that CORBA was more complex than what we required. The main issue was the level of CORBA expertise amongst the different teams as well was not equal. Other issues showed that there were not enough adequate debugging tools available to pinpoint the server issues (in our case firewall issues). We are currently looking into alternatives to the CORBA middleware. But, one of the designs that did perform well was the federation configuration. Having clients and assets only be aware of a single server eased the implementation for both. The new middleware should be able to perform in a similar fashion.

The components each center was responsible for successfully completed their goals. The RAPID servers along with the asset specific bridges allowed RAPID commands and messages to flow between clients and assets. The asset bridge will stay center specific, but the RAPID server can be universal. Future designs will allow for more uniform components, specifically, utilizing the same source code and software in all three centers.

REFERENCES

- [1] Wilcox, B. H., Litwin, T., Biesiadecki, J., Matthews, J., Heverly, M., Morrison, J., Townsend, J., Ahmad, N., Sirota, A., and Cooper, B., "ATHLETE: A Cargo Handling and Manipulation Robot for the Moon." *Journal of Field Robotics* [online journal], Vol. 24, No. 5, URL: <http://www3.interscience.wiley.com/journal/114211098/issue>, Wiley Periodicals, May 2007, pp. 421–434
- [2] Mittman, D. S., Norris, J. S., Powell, M. W., Torres, R. J., and McQuinn C., "Lessons Learned from All-Terrain Hex-Limbed Extra-Terrestrial Explorer Robot Field Test Operations at Moses Lake Sand Dunes, Washington," *AIAA Space 2008* [submitted for publication], AIAA, Washington, DC, Sep. 2008.
- [3] Mittman, D. S., Norris, J. S., Torres, R. J., Hambuchen, K. A., Hirsh, R. L., Allan, M. B., Utz, H. H., Burrige, R. R., and Seibert, M. A., "The Exploration Technology Development Program Multi-Center Cockpit," *AIAA Space 2008* [submitted for publication], AIAA, Washington, DC, Sep. 2008.
- [4] Trythall, S., "JMS and CORBA Notification Interworking", http://www.onjava.com/pub/a/onjava/2001/12/12/jms_not.html.
- [5] Hirsh, R. L., Simon, C. L., Tyree, K. S., Ngo, T., Mittman, D. S., Utz, H., Allan, M. B., and Burrige, R. R., "Astronaut Interface Device (AID)," *AIAA Space 2008* [submitted for publication], AIAA, Washington, DC, Sep. 2008.

BIOGRAPHY



R. Jay Torres is a staff member of the Planning Software Systems Group at the Jet Propulsion Laboratory in Pasadena, CA. He leads the Maestro Planning Software tool for the Mars Exploration Rover (MER) Project allowing MER scientists around the world, to plan the day-to-day science tasks for both rovers. He also works on a science planning software that allows orbital missions to visually plan their science observations. The tool is called the Science Opportunity Analyzer and is currently being used on several missions, notably CASSINI and DAWN. He holds a B.S. degree from the California Polytechnic University, Pomona. In his free time he spends time with his wife, Cora, and kids, Sara and Zachary.



Mark B. Allan is a Senior Software Engineer with the Intelligent Robotics Group at NASA Ames Research

Center. Mark has been a contractor in the Intelligent Systems Division for over 10 years, and is currently employed by Stinger Ghaffarian Technologies, Inc. He specializes in data visualization and has worked in the areas of ground control systems for remote exploration, novel human/computer interfaces, massively parallel data flow architectures, and flight simulators. Mark holds a M.S. in Information Systems from Santa Clara University and a B.S. in Biology from U.C. Santa Barbara. Current topics of interest include the use of virtual worlds to effectively explore remote worlds, the application of technology to enhance individual and team effectiveness, and architectures that enable efficient human-robotic coordination.



Robert Hirsh is an Aerospace Engineer in the Intelligent Systems Branch of the Automation Robotics and Simulation Division at the Johnson Space Center. He first joined NASA during college as a Co-Op student in 1995, and has been working full time at JSC since 2001. During his 13 years at JSC he has worked in the design and development of multiple planetary robots and exploration vehicles. Robert has participated in 9 analog field tests with prototype surface robots in various moon/Mars analog locations around the United States. He is currently working on software development and human interface systems. Robert received a B.S. in Electrical (1997) and a M.S. in Computer Engineering (1999) from Purdue University.



Michael Wallick received his BS in Computer Science (with Honors) from the University of Central Florida in 2001. He was awarded a Masters and Ph.D. in Computer Science from the University of Wisconsin-Madison in 2003 and 2007 respectively, where he worked on software for automatic video editing and automatic organization of large collections of digital photographs. In 2004 he was named as a Microsoft Research Graduate Fellow. He joined the Operations Planning Software Group (and Ensemble Project) at the NASA Jet Propulsion Laboratory in 2007, where he is leading the development of a mission data search interface for the Mars Science Laboratory rover, and future missions.