Aardvark:
An Autonomous System for Scientific Experimentation on Mars

Dan Ports, Amanda Smith, Sarah Lieberman

May 6, 2004

Executive Summary

1 Overview

This report presents a design for an autonomously-controlled system for managing rovers on Mars. It addresses the problems of dealing with lossy communication using a acknowledgement-based protocol, deals with sensor faults by repeating experiments, and schedules experiments and assigns them to rovers using a greedy algorithm that allows for batching of multiple missions, cooperative work with multiple rovers on the same mission, and variable prioritizing of individual missions.

The Mars Rover project can most easily be split into three components: communication protocol, mission execution, and mission distribution. The problems associated with communication protocol are relatively straightforward and include message authentication and error detection and correction. Mission execution covers issues relating to the actual completion of missions by rovers, and includes such issues as sensor failure and data relay between rovers. Mission distribution, possibly the most difficult problem in the project, contains the algorithm which the command center uses to distribute missions to rovers, as well as the mechanisms it uses to detect and deal with rover failure, distribute amendments, and coordinate data uploads.

2 Communications

The basic communication protocol is an at-least-once delivery system, with possible duplicate messages, using a standard packet structure, and only broadcasting one packet at a time to prevent misordering of packets. The packet structure contains the name of the sending device, a message nonce, a message type, a message (optional), a signature, and a checksum. The name is simply an integer coded into each rover and the command center, and the nonce is simply a large integer used as a message identifier. The type is an integer code such as "01" for an acknowledgement, "06" for mission data, and "10" for a request to resend a message. The message is any additional content, such as mission assignments or nonces in acknowledgements. Finally, the devices use a shared-secret key and a SHA1 hash as a checksum to provide authentication and guard against transmission errors, respectively. Encryption and other protections against malicious devices are not needed, because the only other devices on Mars are non-malicious Poodle rovers. To detect the presence of another device, a device sends out queries until it recieves an acknowledgement. The device then opens communication, which stops both devices from moving if it is acknowledged. The device will send out messages and packets of data one at a time, waiting for an acknowledgement (or a resend request) before sending another, thus ensuring at-least-once delivery; duplicate messages are fine since

the only communications are data transmission rather than action requests. When the devices end communication, they send out end messages and then resume other operations. If one device does not recieve a message from another in a certain amount of time, it will try resending its last packet or sending a query; if this fails repeatedly, it will assume a dead device and end communication.

3 Mission Execution

Each rover is capable of autonomously executing missions according to parameters received from the control center. It does so by maintaining a queue of instructions. There are three types of instructions:

- travel to a certain location
- perform an experiment (e.g. taking a photo)
- return to the control center

When a rover performs a experiment, there is a non-zero probability of sensor failure. However, to ensure that a correct result is returned with 95% probability, the rover will repeat the experiment the required number of times. The rover will have a preprogrammed number of trials to perform with each sensor to ensure a 95% chance of at least one accurate result, based on the failure rate of the sensor.

When two rovers are sent together to perform a mission, or a rover travels within communication range of another rover, they will attempt to transmit data to each other. A rover will reject any data which it does not have room to store in favor of keeping its own mission data. Each data copy will have a version number so that if the command center recives multiple copies of data from the same mission, it will know which is the most complete and up-to-date. This will reduce the probability of data loss due to rover failure.

4 Mission Distribution

The mission distributor in the control center is the module responsible for receiving missions and amendments from Earth, allocating them to rovers, and transmitting new assignments to rovers. It maintains tables to keep track of the system state. At all times, it must be aware of which missions are available, which missions have been assigned to which rovers, and the current state of each rover. Rovers can be in four states:

- *available*, meaning it is present at the control center and waiting to be given an assignment
- *busy*, meaning it is currently executing an assignment
- *lame*, meaning it is currently executing an assignment that has become moot due to a mission amendment
- *dead*, meaning that the control center has not heard from the rover and has presumed it to have failed

For rovers that are busy or lame, the control center maintains an expected return time based on the sum of the required travel time for the assignment and the predicted times required to complete each experiment.

The control center assigns missions to available rovers by periodically checking which rovers are in the available state. It then performs mission scheduling using a greedy algorithm. Our algorithm is designed to meet the principal design objective of *maximizing useful work*. We assume that rovers

will fail using a memoryless failure model having a mean time to failure of 100 days, and this failure is independent of whether the rovers are active or idle. Hence, in order to complete the maximum amount of work before all rovers fail, it is optimal to have all the rovers doing as much work as often as possible; idle rovers are "wasted resources."

We introduce the concept of a *priority value* for each mission, which captures the notion of the relative importance of missions. Assignment is performed by a greedy algorithm that takes a list of assignments, ranks them in terms of the quotient $\left(\frac{\text{priority value}}{\text{required time}}\right)$, and assigns the highest-ranked to any available rovers.

If beneficial, multiple missions may be *batched* and executed sequentially by the same rover. The mission distributor uses a metric that takes into account the benefit of the time saved by traveling from the first mission's endpoint directly to the second, rather than through the control center, and weighs it against the expected work that would be lost if the rover failed in the extra time it spends performing the second mission. It then joins any missions for which batching is beneficial (recursively, if appropriate), and assigns these greedily.

If more rovers are available than missions, the mission distributor may assign multiple rovers to an assignment that needs to be performed multiple times. This allows missions to be performed more rapidly and provides redundancy: if one rover fails while performing the experiment, its previous results will have been replicated onto the other rovers, minimizing the amount of result data lost. This process is only performed when there are no unassigned missions, but there are "extra" available rovers.

The control center is periodically queried by NASA on Earth to see if there are any mission results available. If so, it transmits all available mission results. Once the results are acknowledged, they are deleted from the control center's queue. The control center then receives new missions and amendments from Earth. New missions are added to the queue, and assigned to any available rovers as appropriate. Amendments are treated similarly; the old mission in the tables is revised, and the revision ID is incremented. The changed parts of the newly amended mission can then be assigned to rovers.

The control center also periodically checks whether there are any rovers that have been busy for much longer than their assignment's expected time requirement. If so, it assumes that the rovers have failed, and reassigns their missions.

5 Conclusion

This design should be able to ensure that resources are efficiently allocated in order to ensure that as much valid scientific data as possible gets sent back to the scientists at NASA. In order to achieve this goal, we took into consider the various complications that could hinder the process. These complications include mission amendments, the failure of sensors and rovers, and poor communications. The mission distribution algorithm is made flexible by the variable amount of rovers that can get sent out to perform a mission. This flexibility is an asset because it allows the system to efficiently allocate resources under a wide range of work loads. Also, the priority values that can be assigned to missions make the overall system fairly customizable. With simple changes in the procedure for value assignment, the overall system can be customized to suit a variety of needs. Some of the design choices we have made depend on assumptions that would be valid in the real world, whereas other assumptions might only be valid given this design projects model of the world. However, we believe that we have kept these assumptions to a reasonable level and that within the world of this design project the design would soundly deal with the issues with which it is presented.