# Lunar Rovers

## Introduction

NASA is sending a lander to the far side of the moon to compete with the Chinese rover(s). They are planning a more extensive survey involving an initial landing craft and 10 rovers that will be deployed from the landing craft. Each rover will have an internal network of sensors (such as Ground Penetrating Radar, temperature, cameras, LIDAR, etc.). The rovers have a very limit transmission range, and therefore are required to always be in range of at least one other rover, with at least one rover always being in range of the lander, which has powerful transmitter capable of transmitting on another frequency back to earth.

As the rovers wander about collecting data, they must send the data through the network of rovers to the main landing craft who will forward the data to earth. Therefore, each rover must transmit its data to the next closest rover, until it reaches the lander.

## Simulation

Since the rovers have yet to be built, and commandeering a lot of computers is impractical, we will have to make some concessions in the protocols and layouts to run all the software for all the rovers.

The radio network will be simulated by a multicast IP. The communication port(s) will be determined ahead of time (and in a high number range) for ease of use and avoidance of collisions.

Since MAC Addresses cannot be used in simulation, we will assign each rover a unique 8-bit number that may be included in all communications.

## Part 1 / Project 1

### Introduction

In this project, you will implement a distance-vector routing protocol called RIPv2. Each rover on the network executes RIP to exchange routing information with its neighbors, and based on this information, the rover computes the shortest paths from itself to all the other rovers and the lander, which for this experiment may be considered just another rover that does not move. You will implement this routing protocol in Java. We will assume in this part of the project that the may move around (appearing and disappearing to the other rovers).

### The RIP Routing Protocol

You should first read the RIP specifications (RFC 1058 for RIP version 1 and RFC 2453 for RIP 2) VERY carefully and understand them completely. The related RFCs can be found here: http://www.ietf.org/rfc.html . Be aware that you do not have to implement all the features defined in RFC 2453. Most of what we need can be found in Chapters 3 and 4. If there are any subtle issues about the specification, contact or e-mail the instructor.

Your implementation at least should support 1) active RIP as a router, 2) handling incoming route messages, 3) CIDR, and 4) route message broadcasts. We assume that only RIPv2 is used. Also, set the route update time to 5 seconds, not 30 seconds as defined in the RFC, to reduce the convergence time, and 10 seconds for a rover to be determined as offline.

If a rover fails to respond, it should be assumed the rover is out of range and RIP should respond to recover routes if alternate routes are available. To see this effect clearly, you can implement rover movement by terminating the rover’s process or by selective use of firewall blocking. The neighbors of a missing rover should detect that the it is indeed unreachable and set the appropriate distance to infinity. This information will be propagated to the entire network through the RIP routing protocol. As you probably know, triggered updates need to be invoked for fast recovery, and split horizon with poisoned reverse needs to run to prevent the count-to-infinity problem.

## Running Your Program

Your program should take as a command-line parameters the multi-cast IP, the port number and the rover ID.  Each running process should be assumed to be a separate rover.

Each time the routing table changes internally, the rover should print its routing table. An example output may be:

```
Address          Next Hop          Cost
=========================================
10.0.1.0/24      1                 0
10.0.2.0/24      2                 1
10.0.6.0/24      2                 3
10.0.3.0/24      3                 1
10.0.4.0/24      4                 1
```

Note: Example only, your results may be displayed in a different order.

## Submission

Due date: Sunday, March 20, 2022

Zip all source files, project files, makefiles, and anything else required to build a rover jar file on a CS Ubuntu lab machine. Include a README.TXT file giving your name, RIT ID, and any special build/run instructions. Submit the zip file to the MyCourses Project 2 folder for this class.