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 wiill 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 2

Introduction

Now that we have a router for our rovers, we need a reliable transfer protocol to ensure our packets arrive back on Earth safely. Your goal is to produce and demonstrate a protocol that can guarantee the delivery of the packets. The packets produced must also move easily through the regular Internet. The resulting protocol must, in theory, perform better than TCP with smaller packets and hopefully fewer packets sent.

You must submit a one- to two-page paper describing your protocol, and a reference implementation to demonstrate the new protocol in Java. In your description, include the packet headers that are required and used, along with reasons for your choices.

The reference implementation must work on RIT's CS Ubuntu machines and demonstrate sending data, an image file from one machine to another, along with a couple simple command packets.

Use Case

Typical use of the protocol will be for a rover sending large sensor data files and images to the NASA, and for NASA to send small, single packet commands back to the rover. It is not important that the packets arrive in a timely fashion.

Hints

These are hints. You should feel free to use them or not at your discretion.

You should understand how TCP operates and make improvements to the general design based on the situation. For example: TCP does not attempt to correct the bits of missed or corrupted packets at the receiver but retransmits those packets when the sender detects that something goes wrong with those packets. Would a different checksum help in automatically correcting the errors (like 2D Parity)? Would that cause too much computational overhead to be worthwhile? Or miss too many other errors?

The TCP sender infers those errors based on acknowledgments from the receiver and timeout (TO). Are acknowledgments better than negative acknowledgments? Or some combination?

TCP packets have a sequence number that can be used to reorder arrived packets at the receiver. Is that needed?

UDP may be a reasonable protocol to build on top of. Could one define the header format of your own protocol and add this header to each datagram generated from UDP?

TCP supports a congestion control mechanism, but the rovers don't have to worry about too much congestion. Is there a lighter-weight congestion control mechanism? Or can you justify not having a congestion control mechanism?

Perhaps there should be a timeout as well?

How do the command packets fit into this? How about the larger image files?

Other existing ideas: RFC 1151?

Test and Output

For testing, the sending program should send a 5 MB file to a receiving program on another computer using your new protocol. A few sample command packets should also be sent. The packets sent/received should be captured using a packet capturing tool (Wireshark on Windows, TCPDump on Linux, etc.).

Turning in the Project

Due date: Sunday, April 10, 11:59pm

Upload all source files, make files, readme.txt, and other documents in a zip file to the MyCourses Project 3 folder.

If you desire, a time can be scheduled for you to demonstrate your project.