Collin Engstrom

Professor Katia Obraczka

SURF-IT, Summer 2008

September 3, 2008

SCORPION: A Heterogeneous Network Testbed

Introduction:

Wireless networks are becoming a ubiquitous technology in our society. Inherent in this technology is intermittent connectivity. Nodes, or devices on these wireless networks, may not be connected all the time. To that end, with this novel approach to networking comes the need for newer paradigms and protocols. Traditionally, simulations have been used to assess protocol efficiency, but unfortunately these simulations do not account for all of the intricacies of actual networks. To provide more accurate results of protocol testing, it often becomes necessary to use actual network equipment in a realistic environment. Oftentimes one may exploit network testbeds, or physical networks used for simulation, as a means of attaining this accuracy. This summer I worked in the i-NRG lab on building one of these testbeds that we later named SCORPION, Santa Cruz mObile Radio Platform for Indoor and Outdoor Networks. I was also responsible for designing the management software that communicated with nodes on this testbed, allowing us to monitor and control them.

The Testbed:

Prior to SCORPION, other groups had researched network testbeds. For example, Rutgers University's ORBIT has been used to collect highly-replicable data, while the University of Utah's Emulab focuses on mobile robots as network nodes, thereby incorporating a dynamic topology. Our project was to design a testbed that was an amalgamation of these, and other, qualities.

Initially, our group envisaged our testbed with certain functionality. Firstly, a central node management console would display all nodes along with their status metrics (e.g. IP address(es), kernel information, and what version of code was running). These nodes would be shutdown, rebooted, and otherwise controlled from the console as well. To allow for the testing of protocols on the network, we determined that the console should also include some mechanism for disseminating code to all nodes on the network testbed. In order to facilitate timesharing between different people using the same testbed nodes, we also wanted to incorporate a schedule for dividing testbed usage into time blocks.

Changes:

As time progressed, certain parts of this vision changed, while others remained the same. To simplify the process of pushing out software and protocols onto the nodes, our group decided to use the existing SVN (a code management system) structure in the i-NRG lab. All protocols and other software were checked into the lab's SVN repository. This, in turn, was automatically copied to each of the nodes through a boot-up script. Scheduling was also done via scripting.

At the console end, changes were to be made as well. The node console was to display not only a brief summary of each node and its status, but also more detailed information by 'zooming in' on a particular node. Additionally, a web interface was introduced to monitor the status of the nodes from virtually anywhere. This interface used an output file generated by the console to determine which nodes were up and whether or not they were up-to-date.

Programming:

The management suite for the testbed was designed to display all nodes on the network and also allow a person to remotely control these nodes. Two pieces of software work together to achieve this: the management console (gateway side software) and the agent (node side software). Owing to my familiarity with C++, I elected to design the management console in this language. The GUI portion of the application was written with GTK, a graphical programming tool. In an attempt to facilitate coding in GTK, I first used a tool called Glade that generated the GTK code for use with C++. In the end, however, this turned out to be counterproductive, so I coded by hand, using example code where necessary. This GUI facet of the application was designed to run on top of the command line portion of the program, which is also responsible for communication.

All network connectivity between the nodes and the console is done over a UDP socket. The management suite is programmed to send out a status request every ten seconds. The nodes within range update their status information and transmit it to the gateway over this socket. When the gateway receives this information, it displays it in a tree in the GUI. Similarly, if a

person issues a mass shutdown/reboot command from the management console, the communication module will transmit it to all nodes within range.

In order to make communication with nodes possible, each of them must run an agent at startup. Like the management suite, this agent software uses a UDP socket connection to handle communication. This agent runs in the background at startup and waits for commands from the management suite. If it receives a packet over the socket, it decodes the contents, compares them against a list of known commands, and executes the necessary instructions for that command. For example, if the node receives a status request, it first compares the request to a list of known commands. After matching the packet contents to a status request, it runs those instructions necessary for a status request. It begins by collecting information pertaining to the kernel, uptime, and code revision. It then transmits this data back to the gateway over the same socket. Other commands are interpreted and handled in a similar fashion, allowing for great flexibility in adding new commands.

Insight:

By participating in this research project, I have gained valuable insight pertinent to both the domain of academic research and work in general. Firstly, I believe that as a result of our group endeavors with this project, I have improved on my ability to communicate with others working with me. I believe that I am now more capable of being an effective team member. For example, the interdependence between my management suite and a fellow team member's web interface forced us to work closely together to produce the end product. I believe that these team skills are portable and will be useful in any setting, academic or otherwise.

Conclusion:

The network testbed that our group designed this summer embodied existing ideas, while still including several features of its own. Initially, my own task was to design a piece of software that would be used to manage the nodes for our group's network testbed. We believed that a central console would ultimately possess the ability to monitor nodes, remotely control them, and even deploy and schedule software on them. Throughout the course of the summer, we deemed it better to use existing software to accomplish the latter of these goals. Implementing the first two aspects of the project required work in C++, socket programming, and GTK. While challenges like Glade necessitated a certain amount of backtracking, they were a vital part of the research process. After completing this summer project, I believe that I am now a more adept researcher and will carry these new skills with me wherever I go.