A Management Suite for a Disruption-Tolerant Wireless Network Testbed

Kerry Veenstra*, Katia Obraczka*, Wade Gobel†⋆, Daniel Olivares‡⋆ and Valdislav Petkov§
*University of California, Santa Cruz, CA 95060, USA. Email: {veenstra, katia}@soe.ucsc.edu
†Microsoft Corporation, One Microsoft Way, Redmond, WA 98052, USA. Email: wgobel@sbcglobal.net
‡Humboldt State University, Arcata, CA 95521, USA. Email: dolivares@humboldt.edu
§Apple Inc., 1 Infinite Loop, Cupertino, CA 95014, USA. Email: vladi@soe.ucsc.edu

***Abstract*—This paper describes a management suite that we developed for our heterogeneous disruption-tolerant wireless network testbed. Unlike other testbeds whose administrative backchannels have continuous connection, the backchannel of our network is connected to nodes only sporadically, and so our management suite must be designed to accommodate this unusual nature. Our testbed includes shuttles that are part of the UC Santa Cruz campus transportation network. To reduce deployment costs, we include features in our testbed that make it beneficial to a wider user community, thereby encouraging other campus services to enter into a partnership with us to maintain the testbed.**

I. Introduction

This paper describes our experience developing a management suite for the SCORPION wireless network testbed. SCORPION, or Santa Cruz mObile Radio Platform for Indoor and Outdoor Networks, is a heterogeneous wireless network testbed we have built which includes a variety of nodes including different types of ground– and autonomous aerial vehicles. One of SCORPION's main goals is to serve as a real-world testbed which researchers and protocol developers can use to deploy, test, and evaluate network protocols under real-world conditions [1]. The SCORPION management suite introduced here is thus a critical part of the testbed as it facilitates deploying, running, and managing experiments, as well as collecting and retrieving experimental data.

Because of some of its unique features, SCORPION poses a number of significant challenges from a network management– and control point of view. SCORPION emphasizes node heterogeneity by trying to accommodate a variety of nodes with different processing, storage, communication, mobility, and connectivity characteristics. Another interesting aspect of SCORPION is that it incorporates campus transportation vehicles. By having the buses as part of SCORPION, we are able to track their location and route in real time.[1]

Another unique feature of SCORPION is that it is prone to intermittent connectivity. These types of networks are often referred to as "challenged", "disruption-tolerant" (DTN), or "episodically-connected" [3]. So there is no guarantee that an end-to-end path exists between every pair of nodes at all times. Indeed, rather than being exceptions, partitions can be part of the normal operation of the testbed. This, of course, poses interesting challenges to SCORPION's management tool as will be discussed later in the paper.

A. Background and Related Work

The use of testbeds is prevalent in the networking research community. Initiatives like PlanetLab [7] and GENI [8] are evidence that it is now widely recognized that it is critical to deploy and test network protocols in controlled environments that still closely mimic real deployment operation. Such deployments are even more important in the case of wireless networks where the physical world plays a definitive role and is much harder to capture using mathematical abstractions and models.

Differences among wireless testbeds determine their experimental– and management capabilities. Focusing on management, immobile wireless nodes may allow continuous control since a fixed node can remain connected to a nearby wireless access point or to a wired backchannel. This is the case of MoteLab [9] and ORBIT [10]. To achieve the same level of management with *mobile* wireless nodes—particularly those that test disruption-tolerant networks (DTNs)—testbeds require either a dense network of WiFi access points or nodes with secondary 3G links, as are provided with DieselNet [11] [12]. However, including such communications infrastructure in a testbed adds to both deployment costs and maintenance costs. When inclusion of backchannels does not fit into researchers' budgets or plans, DTN testbeds must be designed to operate properly when they are managed through only sporadic connections.

A more direct effect on the budgets of mobile wireless testbeds is apparent when one considers ongoing costs associated with the equipment's housing and operation, such as rent and power. The costs of outdoor mobile wireless testbeds can be greatest since some form of vehicles are required to provide the necessary node mobility. Researchers can reduce costs by partnering with transit authorities to deploy nodes of an outdoor testbed on existing vehicles. Based on our own experience, which is consistent with that of DOME reported in [13], we believe that such a partnership is more likely to be successful when the researchers and the transportation authorities find the resulting system jointly beneficial.

⋆Work supported by the National Science Foundation under Grant No. 1156606.

[1]SCORPION's bus tracking sub-system is described in [2].

B. Paper Outline

The remainder of the paper is organized as follows. Section II summarizes the SCORPION testbed. The SCORPION management suite's use and implementation are described in Sections III and IV. In sections V and VI we discuss our current experience with the testbed management system and our ideas for future work. We conclude our paper in section VII.

II. SCORPION

The SCORPION testbed comprises mobile nodes, gateways, and base stations. Mobile nodes run the protocols that are under test and report their locations; gateways configure and control the nodes; and base stations provide continuous, real-time location tracking. We summarize interactions among these components below. For more details on the testbed, the reader is referred to [1].

There are five different kinds of mobile nodes, three of whose mobility is autonomous. The autonomous nodes include remotely controlled model airplanes with Paparazzi autopilots [14], self-stabilizing remotely controlled model helicopters, and iRobot Create ground robots. Among the non-autonomous mobile nodes, some are permanently installed in campus transportation shuttles, and the remaining are available in briefcase form to be carried by people.

The desire to configure the different kinds of nodes using similar programs suggests that all nodes should have identical compute and communication features. Supporting this ideal, all nodes run Linux and have GPS receivers. However differences in the nodes' mobility capabilities force us to limit the compute and communications features of aerial nodes due to weight limitations: while each terrestrial node features a battery-powered mini-ITX [15] computer and two 802.11a/b/g radios[2], each aerial node uses a smaller Gumstix [16] microcomputer and a single radio. In addition, to support the needs of managers of the campus's transportation-shuttle service, each node that is permanently installed in a shuttle adds a 900-MHz radio for continuous, real-time location tracking. A set of five base stations receive location and status information from the nodes that are installed in campus transportation shuttles. These base stations store their information into a common MySQL database for further analysis.

Network researchers manage the testbed nodes through gateways, each of which is a Linux-based computer with a 802.11a/b/g radio. The management software that runs on the gateways is detailed in Section III. Figure 1 illustrates the SCORPION testbed with its different nodes.

III. Management Suite Overview

The testbed's two independent groups of users (network researchers and managers of the campus transportation system) use separate control interfaces.

Our design of the interface for network researchers needed to confront SCORPION's lack of a continuous administrative backchannel. Although we contemplated using the testbed's DTN network itself for administrative communication, we decided that it is prudent not to require perfect operation of an experimental network in order to retrieve the data generated by the networks experiments. Consequently, we created a simple but robust set of commands that do not rely on the DTN working correctly.

Network researchers can reprogram nodes and can control them using the commands described in Sections III-A and **??**. Since all nodes are deployed from a location near a gateway at the beginning of a test and then collected later at the test's end, initial configuration and final data collection tasks assume that nodes are one hop away from a gateway. This single-hop assumption relieves the nodes of the additional task of routing data logs to a sink, since we do not want to require any routing protocol under test to operate correctly in order to diagnose its operation! Additionally, we do not want collection of experimental logs to interfere with data transmission carried out by the protocols being tested.

Tests using bus nodes require special planning because bus nodes are configured by a gateway near the transportation service's fuel pumps. Network researchers must allocate sufficient time for all bus nodes to be configured before the start time of a test, and they also must wait for a sufficient time after a test for stored data logs to be collected from all buses.

The user interface used by transportation-system managers is presented in Section III-B.

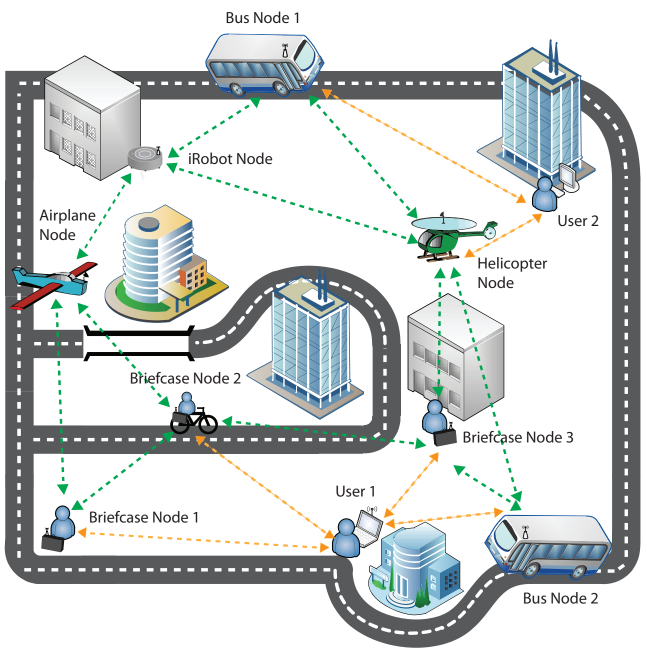

Fig. 1. Diagram of a SCORPION deployment with all five kinds of nodes Also appearing in the diagram are researchers (User 1 and User 2) managing the nodes using the testbed-management suite. A typical experiment uses a subset of the testbed.

[2]The reason for the having two radios was to address the needs of researchers that need to test multi-radio protocols.

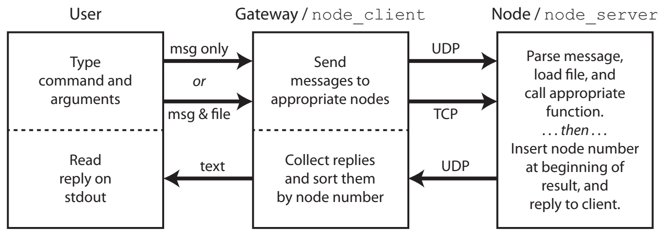
User
Gateway / node_client
Node / node_server
Type command and arguments
msg only
or
msg & file
Send messages to appropriate nodes
UDP
TCP
Parse message, load file, and call appropriate function.
. . . then . . .
Insert node number at beginning of result, and reply to client.
Read reply on stdout
text
Collect replies and sort them by node number
UDP

Fig. 2. The client-server model used by the SCORPION testbed management suite.

A. Test-run Management and Data Collection

A researcher configures and runs tests on the testbed's nodes wirelessly by running a set of five commands on a gateway. Usually the gateway is a laptop computer with a wireless port that has been configured to communicate with the nodes. The utilities are simple enough for interactive use, but they can be scripted as well.

- `nodels` reports which nodes are near the gateway.
- `nodediff` compares a file on the gateway with corresponding files on nearby nodes.
- `noderun` runs programs on nearby nodes.
- `nodecron` schedules experiments by updating cron files of nearby nodes.
- `nodegs` retrieves logged data by fetching syslog files of nearby nodes.

B. Vehicle-Tracking Management Suite

Locations of bus nodes are displayed as colored markers on a web page using the Google Maps JavaScript API. Different marker colors indicate specific bus routes. Marker positions on the map are updated every 2.5 seconds. Anyone with access to a web browser can display the bus-location map, but only transportation-system managers may associate specific buses with specific routes. All received bus location data is stored into a database to support offline analysis.

IV. Implementation of Testbed Management Suite

A. Client-Server Interaction

Our implementation follows the client-server model, where each gateway is a client running `node_client` as commanded, and each node is a server running a background process `node_server`. The `node_client` program is itself run by the commands described in Section III.

When `node_client` is run on a gateway, it communicates with nodes using either TCP or UDP, depending on whether the command that runs it transfers a file. (See Figure 2.)

Since `nodels`, `nodediff`, and `noderun -c` are short commands that receive a short text result from each node, the commanded request can be transmitted in a single UDP packet, as can each of the nodes' results. And so `node_client` uses UDP for these cases. However `noderun -s`, `nodecron`, and `nodegs` all transfer files between the client and the node, making UDP inadequate because data of the transferred file may be split into multiple packets, some of which may not arrive error-free or at all. In this case `node_client` uses TCP, which ensures that all packets of a transmitted file will arrive error-free and in-order. The extra time required to establish a TCP connection is a small cost for data integrity.

B. Client-Server Communication Protocols

`node_client` and `node_server` communicate using a simple request/response protocol. When using UDP, messages are sent as text strings of whitespace-delimited tokens representing a command followed by server arguments. For example, when the gateway executes `nodels -t 5`, which has the *client* argument `-t 5`, it sends to the node just the message `nodels` (but not the `-t` parameter because `-t` controls the client's retransmission behavior). As a second example, executing the gateway command `noderun -c 'uname -a'` sends the message `noderun -c uname -a` to the node.

When `node_client` and `node_server` communicate using TCP (i.e., when a file is transmitted), the TCP message consists of both length fields and data fields: the first field is one byte long and represents the length of the command string, the next field is three bytes long and encodes the length of the file that is being transmitted, the third field is the command string, and the fourth field is the contents of the file.

Except in the case of `nodegs` (whose "reply" is a syslog file that is obtained using `scp`), a reply from `node_server` is transmitted as a UDP packet that consists of two space-separated fields: the first field is the node's number encoded as a text string, next is the space-character field separator, and the second field is the first line of the result that is returned by the command that was executed.

V. Discussion

A. SCORPION and DOME

When surveying existing wireless network testbeds, DOME [12] is the one that comes the closest to SCORPION. Although the two systems have similar goals, their very selection of different technologies for administrative backchannels necessitates major differences in system management.

DOME's 3G network provides sufficient bandwidth to distribute configuration images and provides continuous connectivity for node management. Consequently, updates to nodes can be performed at any time, as long as the node is powered up. SCORPION has a much slower 900-MHz monitoring link for real-time node localization. This link is too slow for image distribution, and so image updates to each SCORPION node must be performed via the node's WiFi link. Although the campus has only a few WiFi access points in buildings close to transit routes, we are fortunate that transit buses must be refueled periodically. Therefore, we located a WiFi access point in a building near the campus filling station to provide necessary connectivity for updates. A feature of the Linux computers used on the bus nodes causes them to remain powered for a few minutes after the bus's engine is turned off for fueling, thereby allowing time for image transfers.

Another consequence of managing the testbed over a disconnected network is that researchers must adapt their tasks to the nodes presently nearby. `nodels` helps identify nearby nodes, `nodediff` provides a low-bandwidth means of confirming the contents of files on nodes, `nodecron` lets one schedule experiments quickly by uploading cron files, and `noderun` provides a means of executing Linux commands and scripts to prepare the testbed's nodes for experiments.

Clearly a backchannel network that maintains continuous connectivity would reduce and possibly eliminate the need for management tools as we have implemented them, but using such networks—e.g., 3G—tends to incur large recurring costs. For some research groups these costs are prohibitively high compared to their benefits, and for these groups, a management system more like ours is necessary.

B. Bus Node

As we mentioned earlier, the jointly beneficial nature of our collaboration with TAPS (our campus's transportation authority) provides motivation for both parties to maintain the bus nodes of the SCORPION system. That motivation is apparent as we work together to develop the next generation nodes for the campus's upgraded bus fleet.

Since the fleet upgrade requires removing node hardware from the campus's original buses, we have decided to upgrade the nodes before installing them in vehicles of the new fleet. The original use of a single Linux computer for two network purposes—that is, for controlling a WiFi research network and for controlling a separate 900-MHz localization network—reduced the cost of deploying the original bus nodes in more than twenty vehicles. However, a consequence of this cost reduction was that the localization code needed to be linked into each node's Linux image to preserve the node's bus-tracking feature during experiments. Since the processor requirements for the localization network are minimal, we are moving the localization functions of each node to a separate microcontroller board that will be installed alongside the Linux computer on each bus. The Linux computer and the microcontroller will communicate over a simple serial-port link: the Linux computer receiving localization information for inclusion in logs, and the microcontroller receiving computer status for transmission over the 900-MHz network.

VI. Future Work

To obtain reliable administrative communication, we provide simple, low-level access to testbed nodes. However, researchers still can benefit from using the research network itself—when it is operating properly—to aid in retrieving experimental results.

One can imagine a researcher using the current testbed-management suite to deploy an experiment that ends with the routing of the experiment's data to one or more fixed collection points. Since such a strategy is not provided directly by the tools of the management suite, the final retrieval phase must be appended at the end of the experiment by the researcher. The advantage of such a strategy is ease of operation since final data retrieval is automatic, but regardless, the low-level access methods should remain available in case the final automatic retrieval phase of such an experiment fails.

VII. Conclusion

In this paper we describe the SCORPION testbed management suite. Because of SCORPION's disruption-tolerant nature, we have designed its management suite to operate properly with only sporadic connectivity to its administrative backchannel. We provide a set of simple utilities that let researchers directly monitor and control the testbed's nodes before, during, and after their deployment. Our inclusion of features in our testbed that are valued by the wider campus community has led to a vital partnership with the campus's transportation authority to maintain the testbed's vehicle-based resources. Such a partnership allows us to share ongoing costs of the testbed, reducing the cost to any individual department while benefiting all.

References

[1] S. Bromage, C. Engstrom, J. Koshimoto, M. Bromage, S. Dabideen, M. Hu, R. Menchaca-Mendez, D. Nguyen, B. Nunes, V. Petkov, D. Sampath, H. Taylor, M. Veyseh, J.J. Garcia-Luna-Aceves, K. Obraczka, H. Sadjadpour, and B. Smith. 2009. *SCORPION: a heterogeneous wireless networking testbed*. SIGMOBILE Mob. Comput. Commun. Rev. 13, 1 (June 2009), 65-68.

[2] J. Koshimoto, M. Bromage, V. Petkov, and K. Obraczka. 2009. *Slug-Transit: a location-based public transportation management system*. In Proceedings of the 6th International Conference on Mobile Technology, Application & Systems (Mobility '09). ACM, New York, NY, USA.

[3] S. Farrell and V. Cahill. 2006. *Delay- and Disruption-Tolerant Networking*. Artech House. Boston.

[4] SCALABLE Network Technologies, http://www.scalable-networks.com

[5] The Network Simulator ns-2, http://www.isi.edu/nsnam/ns

[6] ns-3, http://www.nsnam.org

[7] PlanetLab Consortium, https://www.planet-lab.org

[8] T. Anderson and M.K. Reiter: *GENI: Global Environment for Network Innovations Distributed Services Working Group* (2006)

[9] G. Werner-Allen, P. Swieskowski, and M. Welsh. 2005. *MoteLab: a wireless sensor network testbed*. In Proceedings of the 4th international symposium on Information processing in sensor networks (IPSN '05). IEEE Press. Piscataway, NJ, USA.

[10] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu and M. Singh. 2005. *Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols*. WCNC'05. March 2005.

[11] J. Burgess, B. Gallagher, D. Jensen, and B.N. Levine. 2006. *MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks*. In Proceedings of the 25th IEEE International Conference on Computer Communications, INFOCOM 2006.

[12] H. Soroush, N. Banerjee, A. Balasubramanian, M.D. Corner, B.N. Levine, B. Lynn. 2009. *DOME: a diverse outdoor mobile testbed*. In Proceedings of the 1st ACM International Workshop on Hot Topics of Planet-Scale Mobility Measurements (HotPlanet '09). ACM. New York, NY, USA.

[13] H. Soroush, N. Banerjee, M.D. Corner, B.N. Levine, B. Lynn. 2009. *DOME: a diverse outdoor mobile testbed*. Dept. of Computer Science Technical Report UM-CS-2009-23, Univ. of Massachusetts Amherst.

[14] P. Brisset and A. Drouin. *PaparaDzIY: do-it-yourself UAV*. In Journees Micro Drones, Toulouse, France, September 2004.

[15] mini-ITX Platform, http://www.via.com.tw/en/index.jsp

[16] Gumstix, Gumstix way small computing, http://www.gumstix.com

[17] NextBus, http://news.nextbus.com/how-nextbus-works-2