REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
03-05-2018	Final Report	17-Jul-2013 - 16-Jul-2015

4. TITLE AND SUBTITLE
Final Report: Pharos: A Mobile Testbed for the Development, Validation, and Deployment of Networking Protocols

5a. CONTRACT NUMBER
W911NF-13-1-0269

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
611102

6. AUTHORS
Sriram Vishwanath

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
University of Texas at Austin
101 East 27th Street
Suite 5.300
Austin, TX 78712 -1532

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
63483-NS-RIP.2

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Sriram Vishwanath
UU	UU	UU	UU		19b. TELEPHONE NUMBER 512-471-1190

RPPR Final Report
as of 14-May-2018

Agency Code:

Proposal Number: 63483NSRIP **Agreement Number: W911NF-13-1-0269**
INVESTIGATOR(S):

Name: Sriram Vishwanath
Email: sriram@ece.utexas.edu
Phone Number: 5124711190
Principal: Y

Organization: **University of Texas at Austin**
Address: 101 East 27th Street, Austin, TX 787121532
Country: USA
DUNS Number: 170230239 EIN: 746000203
Report Date: 16-Oct-2015 Date Received: 03-May-2018
Final Report for Period Beginning 17-Jul-2013 and Ending 16-Jul-2015
Title: Pharos: A Mobile Testbed for the Development, Validation, and Deployment of Networking Protocols
Begin Performance Period: 17-Jul-2013 **End Performance Period:** 16-Jul-2015
Report Term: 0-Other
Submitted By: Sriram Vishwanath Email: sriram@ece.utexas.edu
Phone: (512) 471-1190
Distribution Statement: 1-Approved for public release; distribution is unlimited.

STEM Degrees: 3 **STEM Participants:** 3

Major Goals: The goal of this project is to develop, from the ground up, a mobile testbed for real world mobile testing for communications, control and distributed computing. We worked on Pharos, a testbed that enabled us to understand mobility patterns in ad hoc networks, delay tolerant networks, autonomous agents and decentralized control systems. This DURIP was highly successful in instrumenting research in these and related domains such as robotics and computer vision at the University of Texas, Austin and collaborative institutions.

Accomplishments: We accomplished a considerable depth and breadth of research during the course of this project:

1. We fully functionally developed Pharos into a mobile testbed: . Our current methods and tools for validation and evaluation allow examination of components of the overall system in isolation; however our repertoire of tools lacks the ability to gain a complete system understanding of complex deployments with many unpredictable moving pieces. In our work, we develop the Pharos testbed, a networked system of autonomously mobile devices that can coordinate with each other and with networks of embedded sensors and actuators. Using Pharos, we identify and quantify many of the unique challenges associated with mobile cyber-physical systems. Through a series of experiments using the testbed, we demonstrate one of its most important aspects: push-button repeatability, or the ability to easily recreate the same experiment multiple times. In our research, we draw parallels between experiments in live mobile cyber-physical systems and simulation of them to demonstrate the complexities that are not fully captured in the latter and to posit mechanisms by which results of live experiments can be use to drive future mobile cyber-physical system simulations. These results, coupled with demonstrations of how the Pharos testbed can be used for a variety of experiments for mobile cyber-physical systems and how new experiments can be seamlessly executed on the testbed, demonstrate both the critical importance of

real-world validation of mobile cyber-physical systems and
the capabilities of Pharos in supporting such validation.

2. Autonomous Intersection Planning: Autonomous intersection management (AIM) is a new intersection control protocol that exploits the capabilities of autonomous vehicles to control traffic at intersections in a way better than traffic signals and stop signs. A key assumption of this protocol is that vehicles can always follow their trajectories. But mechanical failures can occur in real life, causing vehicles to deviate from their trajectories. A previous approach for handling mechanical failure was to prevent vehicles from entering the intersection after the failure. However, this approach cannot prevent collisions among vehicles already in the intersection or too close to stop because (1) the lack of coordination among vehicles can cause collisions during the execution of evasive actions; and (2) the intersection may not have enough room for evasive actions. In our work, we propose a preemptive approach that pre-computes evasion plans for several common types of mechanical failures before vehicles enter an intersection. This preemptive approach is necessary because there are situations in which vehicles cannot evade without pre-allocation of space for evasion. We present a modified AIM protocol and demonstrate the effectiveness of evasion plan execution on a miniature autonomous intersection testbed.

3. Miniaturizing Vehicular Testbeds: Despite increasingly realistic vehicular network simulations, the effects of real-world mobility on network and application performance in vehicular networks are still not well understood. We use Pharos, a small-scale vehicular network testbed with “push-button”experiment repeatability and develop a framework for analyzing network performance of vehicular networks simultaneously in simulation and in the real world. We empirically study the differences between real-world and simulated connectivity. Early experiment results using our vehicular testbed show significant differences between simulated and actual movements resulting in differences in wireless connectivity. Because of this, we implement a trace mobility model that allows the OMNeT++ simulator replay actual GPS-based movement traces collected by the testbed and scale to larger networks.

4. Multi-Robot Patrols: Multi-robot patrol is a fundamental application of multi-robot systems. While much theoretical work exists providing an understanding of the optimal patrol strategy for teams of coordinated homogeneous robots, little work exists on building and evaluating the performance of such systems for real. In our work, we evaluate the performance of multirobot patrol in a practical outdoor distributed robotic system, and evaluate the effect of different coordination schemes on the performance of the robotic team. The multi-robot patrol algorithms evaluated vary in the level of robot coordination: no coordination, loose coordination, and tight coordination. In addition, we evaluate versions of these algorithms that distribute state information—either individual state, or entire team state (global-view state). Our experiments show that while tight coordination is theoretically optimal, it is not practical in practice. Instead, uncoordinated patrol performs best in terms of average waypoint visitation frequency, though loosely coordinated patrol that shares only individual state performed

best in terms of worst-case frequency. Both are significantly better than a loosely coordinated algorithm based on sharing global-view state. We respond to this discrepancy between theory and practice, caused primarily by robot heterogeneity, by extending the theory to account for such heterogeneity, and find that the new theory accounts for the empirical results.

Training Opportunities: Nothing to Report

Results Dissemination: Our work on Pharos was presented at multiple forums, including at ICRA, ICC and INFOCOM. We worked with colleagues at other universities, as well as collaborators at UT in disseminating our research results. Our collaborators include: Prof. Sanjay Shakkottai, Prof. Christine Julien, Prof. Peter Stone (CS) and Prof. Luis Sentis (ME). Each of the collaborators gave multiple talks on the topic, including within the University, at A&M, Rice and the University of Houston.

Honors and Awards: Nothing to Report

Protocol Activity Status:

Technology Transfer: The Pharos lab became a hotbed of innovation, with two startup companies emerging from the research conducted at the lab:

1. Lynx Laboratories Inc. (now part of Occipital Inc.): Lynx Labs developed the world's first 3D imaging camera, which was subsequently acquired by Occipital Inc.

2. M87 Inc. : M87 building mobile multihop communications systems, and is the pioneer in client-centric multihop communication systems, WiFi tunneling and client SON.

PARTICIPANTS:

Participant Type: PD/PI
Participant: Sriram Vishwanath
Person Months Worked: 1.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

Participant Type: Faculty
Participant: Sanjay Shakkottai
Person Months Worked: 1.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

Participant Type: Faculty
Participant: Christine Julien
Person Months Worked: 1.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

RPPR Final Report
as of 14-May-2018

Participant Type: Postdoctoral (scholar, fellow or other postdoctoral position)
Participant: Liang Fok
Person Months Worked: 3.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

Participant Type: Graduate Student (research assistant)
Participant: Agoston Petz
Person Months Worked: 3.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

Participant Type: Graduate Student (research assistant)
Participant: Nicholas Paine
Person Months Worked: 3.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

ARTICLES:

Publication Type: Journal Article Peer Reviewed: Y **Publication Status:** 1-Published
Journal: Cyber Physical Systems
Publication Identifier Type: Publication Identifier:
Volume: Issue: First Page #:
Date Submitted: 5/3/18 12:00AM Date Published:
Publication Location:
Article Title: Pharos: A Testbed for Mobile Cyber-Physical Systems
Authors: Chien-Liang Fok, Agoston Petz, Drew Stovall, Nicholas Paine, Christine Julien, and Sriram Vishwanath
Keywords: Cyber Physical Systems
Abstract: As mobile cyber-physical systems (MCPS) begin to pervade our everyday environments, tools and techniques for robustly and reliably evaluating system solutions are an essential but as yet missing component for the development process. Our current methods and tools for validation and evaluation allow examination of components of the overall system in isolation; however our repertoire of tools lacks the ability to gain a complete system understanding of complex deployments with many unpredictable moving pieces. In this paper, we present the Pharos testbed, a networked system of autonomously mobile devices that can coordinate with each other and with networks of embedded sensors and actuators.
Distribution Statement: 1-Approved for public release; distribution is unlimited.
Acknowledged Federal Support: **Y**

RPPR Final Report
as of 14-May-2018

Publication Type: Journal Article Peer Reviewed: Y **Publication Status:** 1-Published
Journal: IEEE/RSJ International Conference on Intelligent Robots and Systems
Publication Identifier Type: Publication Identifier:
Volume: Issue: First Page #:
Date Submitted: 5/3/18 12:00AM Date Published:
Publication Location:
Article Title: Evasion Planning for Autonomous Vehicles at Intersections
Authors: Tsz-Chiu Au, Chien-Liang Fok, Sriram Vishwanath, Christine Julien, and Peter Stone
Keywords: Autonomous Control
Abstract: — Autonomous intersection management (AIM) is a new intersection control protocol that exploits the capabilities of autonomous vehicles to control traffic at intersections in a way better than traffic signals and stop signs. A key assumption of this protocol is that vehicles can always follow their trajectories. But mechanical failures can occur in real life, causing vehicles to deviate from their trajectories. A previous approach for handling mechanical failure was to prevent vehicles from entering the intersection after the failure. However, this approach cannot prevent collisions among vehicles already in the intersection or too close to stop because (1) the lack of coordination among vehicles can cause collisions during the execution of evasive actions; and (2) the intersection may not have enough room for evasive actions. In this paper, we propose a preemptive approach that pre-computes evasion plans for several common types of mechanical failures before vehicles enter an intersec
Distribution Statement: 1-Approved for public release; distribution is unlimited.
Acknowledged Federal Support: **Y**

Pharos: A testbed for mobile cyber-physical systems

6 authors, including:

Nicholas Paine
University of Texas at Austin
19 PUBLICATIONS **297** CITATIONS

SEE PROFILE

Christine Julien
University of Texas at Austin
150 PUBLICATIONS **1,391** CITATIONS

SEE PROFILE

Pharos: A Testbed for Mobile Cyber-Physical Systems

Chien-Liang Fok, Agoston Petz, Drew Stovall, Nicholas Paine,
Christine Julien, and Sriram Vishwanath

The Mobile and Pervasive Computing Group,
The Center for Advanced Research in Software Engineering,
The University of Texas at Austin

TR-ARiSE-2011-001

ARiSE
Advanced Research in Software Engineering

Pharos: A Testbed for Mobile Cyber-Physical Systems

Chien-Liang Fok, Agoston Petz, Drew Stovall, Nicholas Paine,
Christine Julien, and Sriram Vishwanath
The University of Texas at Austin
{liangfok, agoston, dstovall, npaine, c.julien, theory}@mail.utexas.edu

ABSTRACT

As mobile cyber-physical systems (MCPS) begin to pervade our everyday environments, tools and techniques for robustly and reliably evaluating system solutions are an essential but as yet missing component for the development process. Our current methods and tools for validation and evaluation allow examination of components of the overall system in isolation; however our repertoire of tools lacks the ability to gain a complete *system* understanding of complex deployments with many unpredictable moving pieces. In this paper, we present the Pharos testbed, a networked system of autonomously mobile devices that can coordinate with each other and with networks of embedded sensors and actuators. Using Pharos, we identify and quantify many of the unique challenges associated with mobile cyber-physical systems. Through a series of experiments using the testbed, we demonstrate one of its most important aspects: push-button repeatability, or the ability to easily recreate the same experiment multiple times. Finally, we draw parallels between experiments in live mobile cyber-physical systems and simulation of them to demonstrate the complexities that are not fully captured in the latter and to posit mechanisms by which results of live experiments can be use to drive future mobile cyber-physical system simulations. These results, coupled with demonstrations of how the Pharos testbed can be used for a variety of experiments for mobile cyber-physical systems and how new experiments can be seamlessly executed on the testbed, demonstrate both the critical importance of real-world validation of mobile cyber-physical systems and the capabilities of Pharos in supporting such validation.

1. INTRODUCTION

Mobile cyber-physical systems (MCPS) are gaining importance as key enablers of emerging applications; this necessitates reliable, robust, and rapid validation and evaluation mechanisms for integrated communication, coordination, and control solutions. Theoretical research in specific sub-domains of mobile networking, distributed control, sensing, and ubiquitous computing have generated many algorithms and techniques to solve problems that are supportive of mobile cyber-physical systems. Individually, these problems are technically complicated: they have many pieces but are ultimately made predictable in terms of their elicited behavior. Together in the form of a complete mobile cyber-physical system, the integration of these pieces is extremely complex: its behavior from the perspective of any component can be highly unpredictable, and disparate system components can often lead to unexpected behaviors. Although individual pieces of the solutions are commonly evaluated rigorously through mathematical arguments or statistically through software simulation, these evaluations are performed in isolation, thus oversimplifying many of the complexities of a real mobile cyber-physical system. Indeed, such oversimplifications are necessitated by the need for analytical/simulation tractability, but these can lead to myopic solutions, which when brought together can result in outcomes inconsistent with the original simplifying assumptions.

In this paper, we introduce the Pharos testbed, an autonomous mobile testbed for extensive validation and evaluation of mobile cyber-physical systems. This is motivated by the fact that current evaluations supported by mathematical models or simulation are largely invalidated [2]; to correct for this, results from simulation should be coupled with a validation of the complete mobile system, from the application and its operating scenario down to the mobility platforms and physical radios [14]. Some effort has been devoted to checking the accuracy of simulations using live networks [11, 16], showing that there is a significant difference between simulations and real systems [1, 14]. However, these live network results have served primarily to discredit simulation results, remaining difficult to replicate and offering little to no guidance on how mobile cyber-physical solutions may perform in the real world. Testbeds have measured the impact of radio propagation on real deployments [4, 8, 7, 9, 17, 20, 25], and some have connected those measurements with simulation [15]. We present a more detailed overview of this related work in Section 7. As a general rule, these testbeds do not explicitly account for mobility, though a few simulate the movement of devices by dynamically attenuating radio signals [26, 27, 29]. A few testbeds support ac-

tual mobility but often lack reproducibility because the mobility is itself ad hoc and not repeatable [21] or tied to a specific infrastructure [6].

Pharos supports mobile cyber-physical system evaluation at all levels (from hardware through the network stack to tailored application functionality) in live networks. Our fundamental building block, the Proteus [1], is an autonomous mobile system with highly modular software and hardware. Pharos[2] contains a number of the Proteus nodes and enables execution of live mobile cyber-physical systems deployments. This paper presents three contributions related to Pharos: *i*) we introduce the Pharos testbed as a comprehensive mobile cyber-physical system testbed (Section 3); *ii*) we characterize the *repeatability* of Pharos experiments (Section 4); and *iii*) we quantify the divergence of Pharos experiments from simulations (Section 5).

In identifying the design requirements of mobile cyber-physical systems, we identify many unique characteristics and challenges they pose. Unlike traditional cyber-physical systems, mobile environments have constantly changing topologies with nodes that opportunistically communicate via a wireless ad hoc network. Thus, the communication "graph" between nodes is rapidly evolving due to mobility and the vagaries of the wireless medium, and it is difficult if not impossible to reliably recreate it in simulation. Pharos gives us the unique ability to deploy tens of autonomous mobile nodes that can simultaneously and continuously measure position, state of the communication channel, and other system attributes and compare them with simulation. Pharos also offers us push-button repeatability of these experiments, providing insights into diverse behaviors that a mobile cyber-physical system can exhibit across runs and in different environments in spite of using the same hardware and software. Using the Pharos testbed, we pinpoint cyber and physical variations across runs, empirically establishing relationships between physical properties and cyber properties. We demonstrate the repeatability of our system despite unpredictable and thus uncontrollable dynamics inherent to a mobile cyber-physical system. Finally, we correlate Pharos experiments with simulations, demonstrating that existing simulators fail to completely capture the complexity of even small-scale mobile cyber-physical system deployments.

Pharos explores fundamental issues related to the reproducibility and accuracy of mobile cyber-physical experiments. The testbed and its constituent components are easily changeable from both a hardware and a software perspective, resulting in an expressive, malleable, and extensible platform for evaluating sophisticated mobile cyber-physical system solutions.

[1]Proteus: a person or thing that readily changes appearance, principles, etc., or, from mythology, a sea god noted for his ability to assume different forms.

[2]Pharos: any lighthouse or beacon to direct sailors, or, from mythology, the mythical home of the sea god Proteus.

2. PROBLEM DEFINITION

In the "real-world," user perception of performance is measured through application responsiveness in the context of a complete system. Ultimately, the goal of mobile computing research is to improve this performance. Unfortunately, testing new components in a complex mobile cyber-physical system is excessively difficult due to the unpredictable and uncontrollable external factors. We create a testbed that reduces these hurdles associated with validating and evaluating complete mobile cyber-physical systems.

We must rely on commodity hardware to best ensure that results from the testbed reflect how an application or system would function in a real deployment. Therefore, in addition to providing support for autonomous movement, our platform must also enable heterogeneous *system* evaluation in the presence of real mobility and real wireless communication. Using human volunteers to accomplish this proves difficult, as it is difficult to ensure experiment repeatability and scalability (finding enough volunteers for each test may be non-trivial). Thus, our testbed must be robotic, while incorporating realism, repeatability, and scalability in its design. The complete vision is a monumental undertaking; we apply pragmatic constraints that allow us to make a significant contribution toward supporting the validation of complete mobile cyber-physical systems. Specifically, we support moderate speeds of mobility; we focus on movement patterns in which devices move at no more than 10.5 meters per second, or 25 miles per hour.

This paper lays the foundation for Pharos, intended to serve ultimately as a benchmark for the design and analysis of mobile cyber-physical systems. To this end, we undertake the following specific challenges:

- Design steps for the Pharos Testbed to support heterogeneity and extensibility in both hardware and software to enable a wide variety of experiments with mobile cyber-physical systems;
- Creation of a supporting software infrastructure that enables push-button repeatability, including repeatability of mobility patterns and communication capabilities to the extent possible; and
- Understanding of and quantifying the similarities and differences between experimental results and simulated ones with the purpose of replicating experiments and building on them in simulation.

3. THE PHAROS TESTBED

In this section, we first describe the Proteus platform, which provides the hardware capabilities that drive the

Pharos testbed. We then discuss the testbed's overarching software architecture, a key component in enabling the push-button repeatability we describe in the next section. This constitutes the first contribution of this paper: the design and development of an expressive and extensible testbed for mobile cyber-physical systems.

3.1 The Proteus Platform

The centerpiece of Pharos is the Proteus mobile node [24], which is the individual participant in the environment. The Proteus design focuses on componentization and reuse of commercial-off -the-shelf (COTS) equipment to maximize both robustness and flexibility. Each node is inexpensive, easy to manufacture, and simple to work with. Below we discuss the details of the design in three major functional sections, shown in Figure 1: mobility, behavior, and interaction.

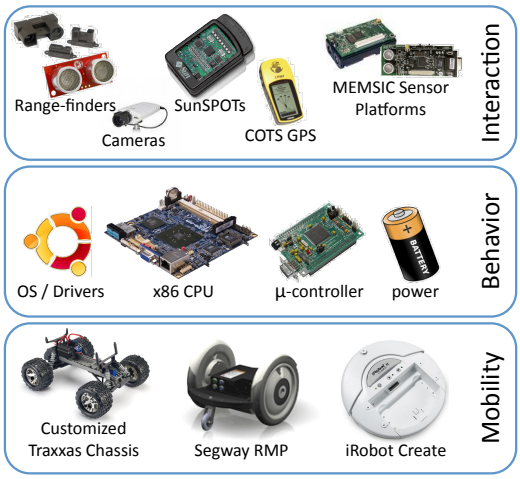


Figure 1: The Proteus Node Hardware

To reduce component coupling, we use well established application programming interfaces (APIs) whenever possible. For example, the Player/Stage API [3] allows movement definition (e.g., drive forward) to be fully differentiated from the hardware implementing the behavior (e.g., the Segway® or the Traxxas). This approach also extends to the interface between the Proteus node and experimental behavior. Conceptually, individual pieces of an experiment can be developed and validated independently using off-the-shelf simulators tailored for a particular purpose (e.g., Stage [3] for movement or OMNeT++ [28] for networking). Once validated individually, these behaviors can be straightforwardly adapted to Proteus nodes, and the complete cyber-physical system assembled quickly. Thus Proteus's architecture enables a rapid transition from individual pieces to an integrated real-world experiment.

Physical Mobility. The physical mobility of a Proteus is provided by one of three options: an iRobot Create®, a Segway® RMP50, or a customized Traxxas Stampede. The Create® is a low cost, low speed, differentially steered robot with a simple serial control interface. The RMP50 is based on Segway®'s popular self-balancing products and is controllable over a CAM bus or USB port. It is more expensive than the Create® but offers higher speeds, higher payload capacity, and long-range outdoor use. The third mobility option is a customized Traxxas Stampede. The Stampede is a high-performance remote controlled car with Ackerman steering and 4-wheel-independent suspension. Each platform provides its own power to reduce dependencies on other components. While the Traxxas is not a COTS component, the low cost, light weight, outdoor compatibility, and range of speeds makes it a desirable option. The Traxxas platform is controlled by the on-board micro-controller described in the next section. Details on the hardware, assembly, and software are all available to external groups wishing to reproduce them[23, 24]. Figure 2 shows a photograph of a Proteus node using the Traxxas mobility platform.

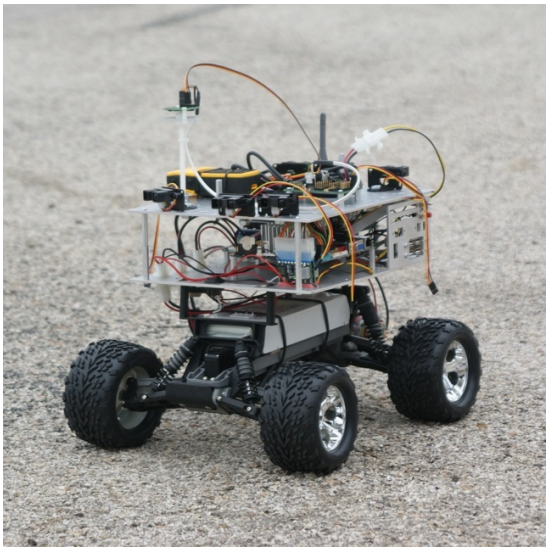


Figure 2: The Proteus Mobile Node

Behavior and Communications. A low-power VIA EPIA® x86 Linux-based computer coupled with a Freescale™ 9S12 micro-controller provides the platform for Proteus node behaviors. This dual architecture offloads many real-time tasks to the micro-controller while allowing the x86 system to focus on higher level aspects. The two-level approach also opens a wide range of I/O options for connecting sensors and other peripherals. Basic communications are provided by an on-board 802.11 b/g wireless network interface controller with a 5.5 dBi antenna. Other network communication technologies (e.g., Bluetooth, MEMSIC® Motes) can be easily added to any Proteus as required by a particular experiment by attaching an additional plane to the platform; we have experimented with multiple

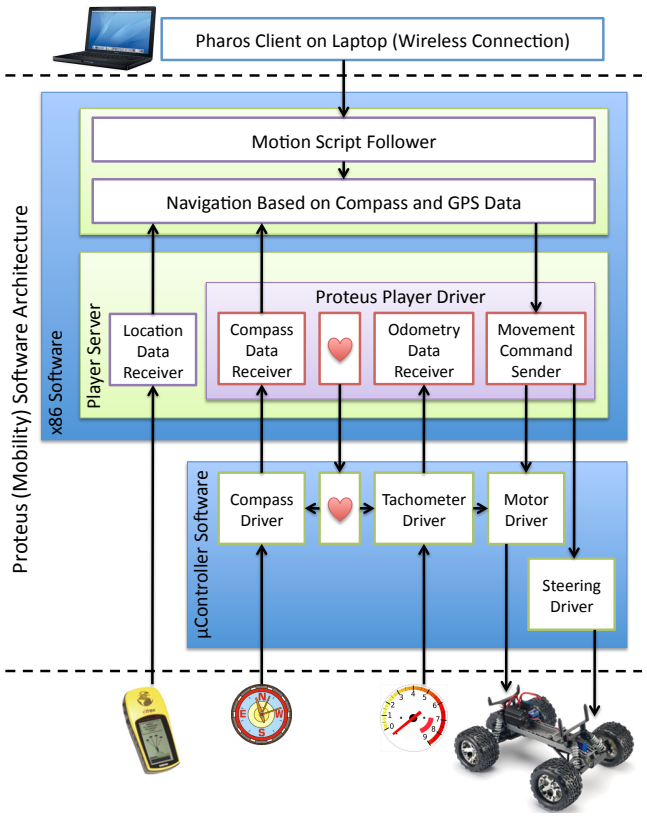


Figure 3: The Proteus (Mobility) Software Architecture

planes built from a variety of sensors. Typically, mobility commands for a Proteus node are executed by user-level applications through the Player/Stage API. Depending on the platform, these commands are then sent to the mobility platform via serial interface, USB, or the micro-controller. Sensor data is collected in much the same way via serial, USB, or the micro-controller.

Environmental Interaction. The third functional area of the Proteus is sensing and actuating. We currently support various range-finding sensors, digital compass, global positioning system (GPS), and cameras, as well as ambient sensing devices including MEMSIC® motes. Many of these sensors are supported by third-party drivers for the Player API. The remaining sensors have matching interfaces in the Player API and only require us to implement a device-specific driver that typically resides on the micro-controller and is exposed to the x86 through the serial connection. The drivers enable sensed data to be used to influence the node's mobility in addition to being incorporated into applications running on the x86 computer.

3.2 Software Architecture

The software architecture used by the Pharos testbed is shown in Figure 3. At a high-level, it consists of three components: a Pharos client residing on a laptop that wirelessly communicates with one or more Proteus nodes, Pharos and Player servers running on each Proteus's x86 computer, and sensor/actuator drivers that reside within Proteus's micro-controller.

Pharos Client. The Pharos client is written in Java™ and serves as the experiment coordinator. It is responsible for assigning motion scripts to Proteus nodes and initiating the execution of the motion script. The motion script is a text file containing tuples of the form <LATITUDE, LONGITUDE, SPEED, PAUSE TIME>, indicating a waypoint, a desired steady state speed, and a pause time that the node should wait at the waypoint before moving on. Upon receiving the motion scripts and experiment configuration, which contains the node specifications and motion script assignments, from the user, the Pharos Client wirelessly connects to the Pharos Servers on each node, transfers the appropriate motion script, and coordinates the start of the experiment. At this point, the nodes may move out of range of the Pharos Client, and it has no further role until the experiment is over when it collects the log files, organizing them by experiment identifier and node ID.

Pharos Server. The Pharos Server is also written in Java™ and consists of a Motion Script Follower and a Navigation component. The Motion Script Follower informs the Navigation component of the next waypoint and desired speed and waits for it to finish steering the node to the specified waypoint. Upon arrival, the Motion Script Follower pauses the specified amount of time before repeating the process. The Navigation component requires compass and GPS data, using both to adjust the steering angle and speed at a frequency of 5Hz. The Navigation component obtains the sensor data and issues the movement commands through a Player Server that also runs on the x86.

Player Server. The Player Server is implemented in C/C++ and provides a popular middleware abstraction for obtaining GPS and compass data and issuing movement commands to control movement, to which we've added functionality to allow it communicate with the custom hardware on the Proteus robots. It connects to the Garmin® eTrex GPS device through serial connections, and sends both movement commands and a 1Hz heartbeat to a micro-controller which implements the low level robotic hardware routines.

μController. The micro-controller implements low-level drivers in a mix of C and assembly. It has a Compass Driver that reads the compass heading via an I2C bus; a Tachometer Driver that computes the current velocity based on dual frequency-modulated signals from the tachometer; a Motor Driver that uses a PWM signal to control the power delivered to the rear wheel motor, and a Steering Driver that controls the angle of the front wheels using a PWM signal. It automatically sends compass and odometer data to the x86

upon receiving the heartbeats. This push-based data delivery model reduces communication overhead allowing the micro-controller to focus on time-critical tasks.

4. CHARACTERIZING REPEATABILITY

In this paper, we characterize various facets of Pharos using the Traxxas mobility plane and a sensor plane consisting of a CMPS03 digital compass and Garmin eTrex GPS. We use this deployment because it is suitable for an outdoor environment at the scale of a parking lot.[3] Because the evaluation location contained obstacles like lamp posts, and because our nodes were not configured with our obstacle detection sensors and avoidance algorithms, we created a motion script to purposely avoid obstacles. For consistency and to enable comparisons, we use the same motion script throughout our evaluations; this particular script consists of eleven waypoints as shown in Figure 4 (we call it the "lollipop motion script" because of its shape). To ensure comprehensive evaluation, the lollipop motion script contains segments of varying length and turns of different directions and angles. In this section, we investigate various aspects of push-button repeatability in Pharos.

4.1 Definition of Motion Divergence

Before quantifying motion repeatability, we first define what it means for two executions' mobility patterns to diverge. In Pharos, divergence in mobility is the difference between the node's actual and ideal trajectories. We can calculate this divergence by measuring the distance between the node's actual location and its ideal location at certain points in time as it follows its motion script. There are many ways in which ideal location can be defined. In this paper, we use four different definitions, illustrated in Figure 5. In the figure "start" is a waypoint in the mobility trace, while "destination" is the subsequent waypoint. Since our evaluation environment has no obstacles, in an ideal situation, the node would travel a perfectly straight line from the start to the destination (indicated by a dashed line in the figures). However, the node usually diverges from this ideal trajectory due to myriad factors such as sensing inaccuracies, the node's non-zero turning radius, misalignment of the front wheels, imperfect tuning of the suspension or lubrication of the wheels, etc. A sample actual path is depicted in Figure 5 by the wavy solid line from the start to the destination; black circles represent points at which the on-board GPS can be sampled and divergence can be measured.[4]

Absolute Divergence. Our first definition of divergence is *absolute*, illustrated in Figure 5(a). It this case, the node's movement error is always calculated in reference to the original ideal trajectory. At any moment, the node's ideal location is the point on the ideal trajectory that is shortest distance to the node's actual location, i.e., the point on the line perpendicular to the ideal trajectory from the current location. Absolute divergence defines how far off the node's actual trajectory is relative to the theoretical ideal of the motion script, which matches the expectations of the user of the testbed, i.e., that the nodes will move along perfectly linear lines from one waypoint to the next.

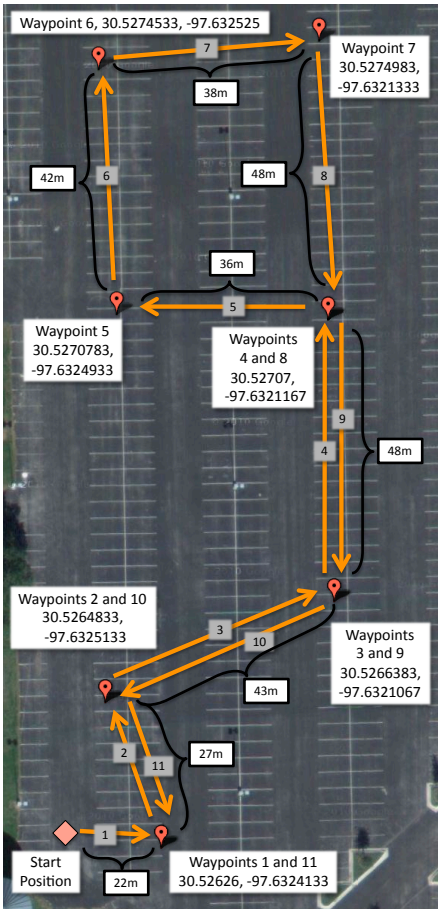
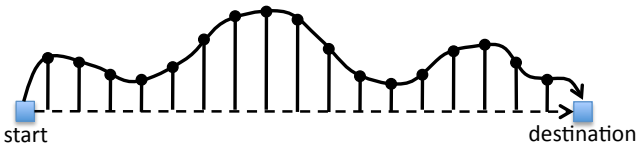


Figure 4: The lollipop motion script used throughout our evaluations

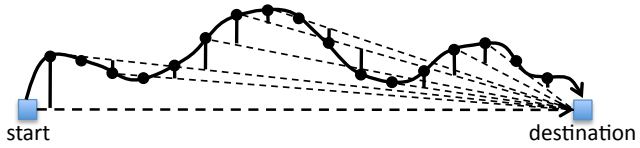
Relative Divergence. While absolute divergence is useful, it is not "fair" in that it does not use the same definition of the ideal path as that used by the waypoint navigation algorithm implemented on the Proteus. The waypoint navigation algorithm does not consider the ideal path to be the straight line connecting the segment's start position to its destination position. Instead, it assumes that the ideal path is the linear line from the node's *current location* to the destination (i.e., the node's objective is to reach the final destina-

[3]For these experiments, the Pharos testbed was deployed in the West parking lot of the Dell Diamond in Round Rock, TX (GPS coordinate: 30.527345,-97.632118).

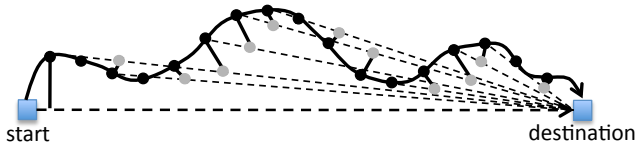
[4]In this study, we assume GPS provides ground-truth location. On average, its indicated accuracy was 3m (10ft).



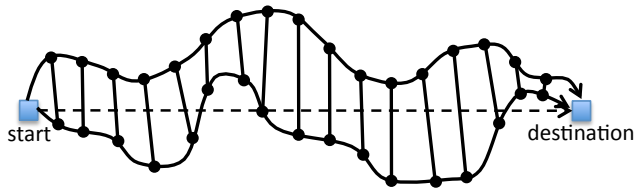
(a) Absolute Divergence



(b) Relative Divergence



(c) Relative Speed Divergence



(d) Reflective Divergence

Figure 5: The four definitions of path divergence used to evaluate motion characteristics

tion rather than to follow the original ideal trajectory). Thus, the ideal path along which the node should travel changes every time the node moves, as indicated by the additional dashed lines in Figure 5(b) from the node's current location to the destination. To match the objective of the navigation algorithm, we introduce *relative divergence* as the shortest distance between the node's current location and the recalculated ideal path based on the node's previous location. Relative divergence is useful because it quantifies the minimum amount of node positioning error that accumulates during the inter-arrival time of GPS location measurements.

Relative-Speed Divergence. Relative divergence captures the semantics of the ideal path as defined by the Proteus's waypoint navigation algorithm but does not consider node speed. We introduce *relative-speed divergence*, shown in Figure 5(c). Relative-speed divergence is the distance between the node's current location and the the location it should be at if it had traveled at the same speed along the recalculated ideal path, whereas relative divergence is simply the shortest distance between the node's current location and *any* point along the recalculated ideal path. The gray circles along the recalculated ideal paths represent the ideal position of the node assuming it traveled at the same speed along the ideal path. Note that relative-speed divergence is by definition greater than or equal to relative divergence. Relative-speed divergence quantifies the actual positioning error that accumulates between GPS location measurements relative to the navigation algorithm's theoretical ideal.

Reflective Divergence. The last form of divergence is *reflective* and is distinguished in that it compares one execution of a motion script with another execution of the same motion script. Specifically, the motion script is segmented into percentages of completion. At every ten percent increment, the location of the node during one execution is compared with the position of the node in another execution of the same motion script. Any difference is the reflective divergence. Reflective divergence is important because it enables direct comparison of the repeatability across experimental runs. It does not compare to a theoretical ideal but rather to actual location during previous executions of the same motion script; reflective divergence allows us to measure the preciseness of mobility in Pharos.

4.2 Repeatability of a Single Node

To evaluate movement repeatability, we programmed a single node, named Lonestar, to follow the lollipop motion script seven times. For all runs, the nodes traveled at 1.5m/s. The actual paths Lonestar took are shown in Figure 6. While Lonestar did not travel the exact same path each time, perhaps due to variations in GPS/compass accuracy, it did follow the same general path. In this section, we delve deeper into the details of the motion repeatability for these trips by Lonestar. Note that this is simply a single example of such repeatability; we restrict our analysis to only Lonestar's movement since factors that differ across nodes may impact the results (see Section 4.3).

Figure 7(a) shows the absolute divergence of Lonestar over the seven runs.[5] The error bars indicate 95% confidence intervals. The graph is divided into eleven segments, one for each segment in the motion script. The two most significant digits of the x-axis denote the segment being traversed, which is also the destination waypoint, while the least two significant digits indicate the percentage of the segment traversed. Because the segment lengths are not identical and GPS location measurements do not arrive in fixed time intervals (their inter-arrival times vary between 1-2 seconds), the absolute divergence shown is the result of normalizing the timestamps of the raw data to be a percentage of segment traversal and then linearly interpolating the node's position at fixed percentages of segment traver-

[5]Due to space restrictions, only a limited number of motion divergence graphs are shown. For a full list, see: http://pharos.ece.utexas.edu/wiki/index.php?title=Evaluating_Motion_Divergence

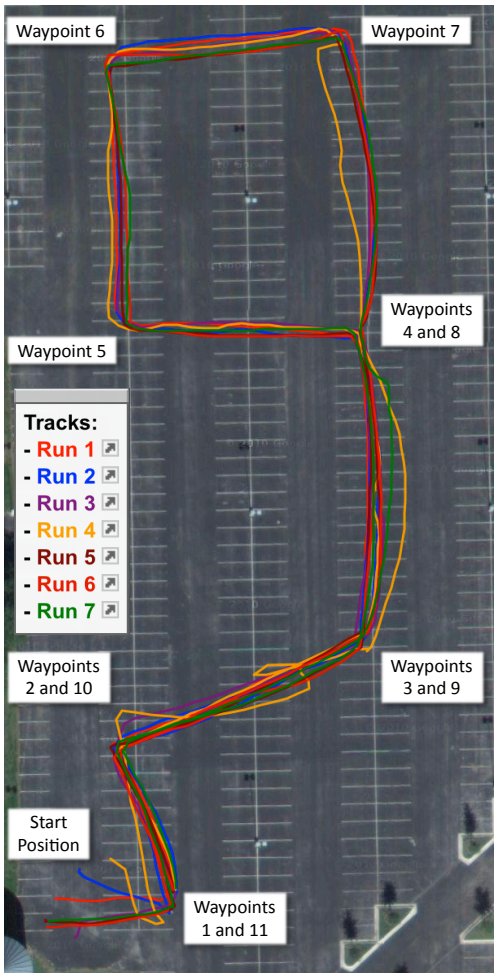


Figure 6: The paths traveled by Lonestar over seven executions of the lollipop motion script

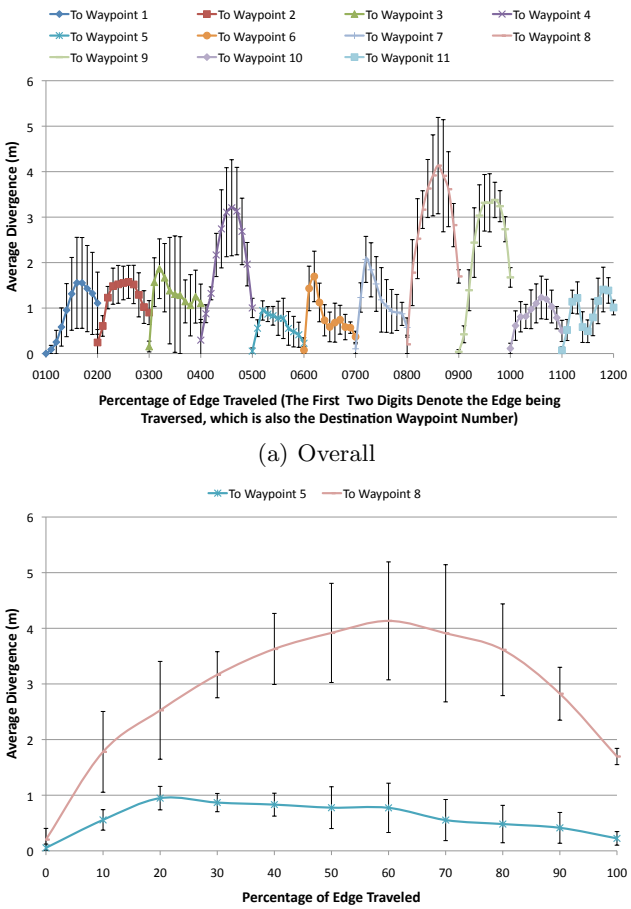


Figure 7: Absolute divergence of Lonestar

sal. In this case, we interpolated the location of the node at every ten percent of segment traversal.

While the average absolute divergence was relatively low, three of the segments show poorer performance. Figure 7(b) shows two samples that represent divergence extremes: one for the trip from waypoint 4 to 5 that had the least divergence, and one from waypoint 7 to 8 that had the worst. The average absolute divergence over all segment traversals was 1.34 ± 0.08m. Considering just the "bad" segment traversals (i.e., segments 4, 8, and 9), the average absolute divergence was 2.39 ± 0.18m. Removing these segments from the others, the average divergence was only 0.95 ± 0.06m. In examining the nature of the environment where the mobility trace occurred, a possible explanation for why the trip from waypoint 7 to 8 exhibited increased error may be the incline of the parking lot and the alignment of the front wheels on Lonestar. These divergences are relatively small considering that the length of the entire path is two orders of magnitude higher at 304m. Furthermore, these are the absolute divergences that are relative to the straight line from the segment's starting location to final destination, meaning they unfairly overestimate the perceived error (as described when defining absolute divergence). Despite this, the small absolute divergence is evidence that the Pharos Testbed achieves movement repeatability for the same node across experimental runs.

Figure 8 shows the relative divergence for Lonestar over the same seven executions of the lollipop motion script. As expected, the divergence is much smaller since the ideal path is continuously recalculated. Averaged over all segments, the relative divergence is only 0.31 ± 0.02m. This is approximately three times less than the absolute divergence, indicating that while the node may not be able to follow the ideal line from the segment's start location to the destination, it is able to remain relatively close to the adjusted ideal line. One interesting phenomena is the spike at the beginning of each segment traversal. This is caused by the non-zero turning radius of the node, which forces the node to significantly diverge from the ideal route as it orients itself towards the next waypoint. Because of this non-zero

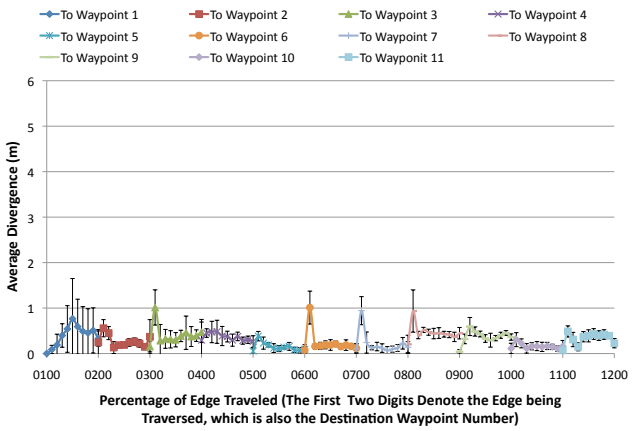


Figure 8: Relative divergence of Lonestar

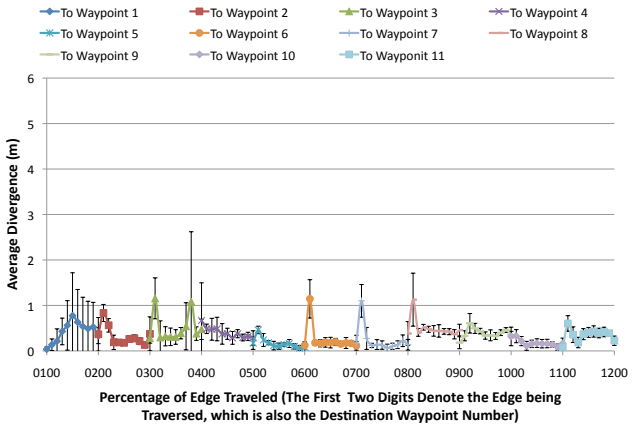


Figure 9: Relative-speed divergence of Lonestar

radius, the node inherently drifts away from the absolute ideal path at the very beginning of each segment, which is another reason why the relative divergence is so much lower than the absolute divergence.

Figure 9 shows the relative speed divergence for Lonestar over the same seven runs. Averaged over all segments of all executions, the relative divergence was 0.34 ± 0.03m which is slightly higher than relative divergence as expected. An interesting feature is the spike about 80% of the way into traversing the third segment. Looking at the top-down view of the path shown in Figure 6, it is apparent that the spike is due to the node temporarily getting lost and running in a loop during one of the motion script executions. This spike only appears in the relative-speed divergence because it considers the speed of the node, which is actually negative when the node mistakenly makes a loop.

So far, this divergence analysis is relative to an ideal path defined in terms of the original motion script (absolute divergence) or the previous location of the node during the same execution of the motion script (relative and relative-speed divergence). These metrics characterize the *accuracy* of the mobility execution, i.e., the degree to which the experimental results reflect the target underlying condition. To better understand how the motion of the same node varies across executions of the same motion script, i.e., the precision of executing the mobility script, we measured the reflective divergence exhibited by Lonestar as it executed the lollipop motion script seven times. The results are shown in Figure 10. The figure shows the average reflective divergence across every pairwise combination of the seven executions. As before, the error bars indicate 95% confidence intervals. One immediately apparent phenomenon is the node takes significantly different paths while going towards the first waypoint. This is a result of our control of experiment initiation; we neglected to start the node at the same location during each execution of the script. Another feature is that as the node traverses each segment, its divergence is initially high but drops throughout the segment. This is due to the node ending up with a different orientation at the end of the previous segment, which is due to cumulative but varying errors throughout the previous segment's traversal, resulting in the node needing to turn by different amounts as it begins to traverse the next segment. Overall, discounting the divergence going towards the first waypoint, the average reflective divergence was only 1.81 ± 0.07m. This is relatively small compared to the size of the experiment, which covers a region over 40m wide and 100m long.

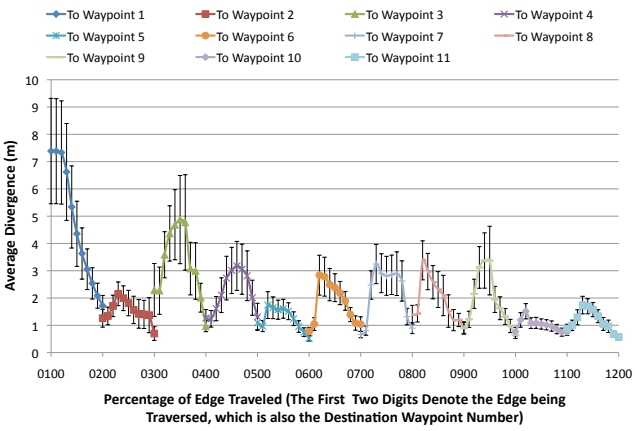


Figure 10: Reflective divergence of Lonestar

4.3 Repeatability Across Multiple Nodes

To understand motion repeatability across nodes, we used the results from executions of two additional nodes, Wynkoop and Shiner. Figure 11 shows an overhead view of their actual motion traces. The figure clearly indicates that each node exhibits its own unique "mobility personality." For example, Shiner sometimes makes very wide right turns relative to the other two nodes.

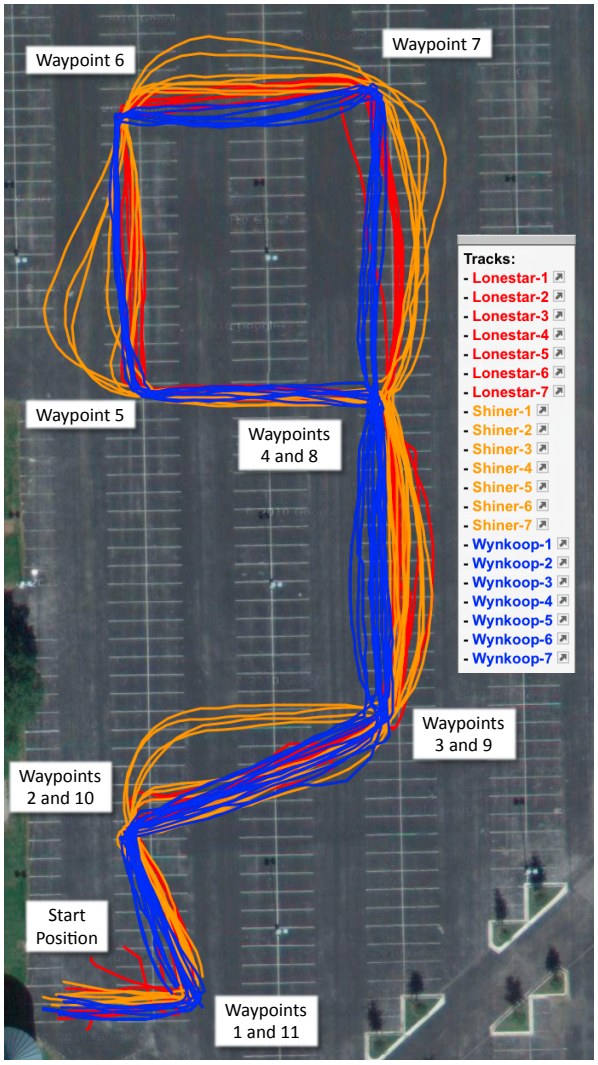


Figure 11: The paths Lonestar, Shiner, and Wynkoop traveled over seven executions of the lollipop-shaped motion script

This could be indicative of differences in the alignment of the front wheels or calibration of the steering arms. Among the three nodes, Wynkoop seems to be the most accurate. These observations are supported by the differences in their absolute divergences. Specifically, the average absolute divergence and 95% confidence intervals of nodes Lonestar, Shiner, and Wynkoop across all seven executions are 1.34 ± 0.08m, 2.49 ± 0.18m, and 1.20 ± 0.06m, respectively. Clearly, despite the fact that each node exhibits its own unique motion profile, which is sometimes inaccurate, the 95% confidence intervals in the absolute divergence remain relatively small. This characterizes the Pharos testbed's ability to provide motion repeatability across multiple nodes.

4.4 Instant-Simulation Replay of Experiments

The log files generated by the Pharos Middleware while executing a motion script can be directly fed into a simulator for achieving instant-replays of the experiment. This is useful to visualize what occurred and aid in debugging the system. To accomplish this, we built a tool and an OMNeT++ [28] module that converts a node's locations as recorded in the log files into a specialized motion script that is used by OMNeT++ to simulate the movements of the nodes. Every nodes' GPS coordinates are logged as they arrive from the onboard GPS device along with the corresponding GPS_time timestamp. We built a perl script which translates these GPS coordinates into absolute {x,y} locations (in meters) with the southwestern-most point of the experiment space bounding-box as the {0,0} location. To simulate the motion, we built a mobility module for OMNeT++ called **TraceMobility** that follows these {x,y,timestamp} points. Since GPS readings are not instantaneously available at all times (due to the sampling rate of the GPS device), **TraceMobility** interpolates the position of each node along a straight line between sampled locations using a uniform speed that would take the node from a point A to a point B in the amount of time that separates the two samples. This interpolation may not be ideal since it does not account for the direction the node was heading or turn radius constraints. However, since the GPS data inter-arrival time is every 1-2 seconds, the effect of this limitation is small and well within the absolute error limitation of the GPS devices themselves. Using these tools, we can automatically replay real-world experiments in OMNeT++.

5. DIVERGENCE FROM SIMULATION

The repeatability of Pharos experiments characterized in the previous section looks at measures of the Pharos testbed that are explicitly under our control, namely different facets of the Proteus node's mobility. Communication, on the other hand, is known to be unrepeatable in the real world, even in experiments that control for other factors. Therefore we sought to characterize the difference between simulated connectivity among mobile nodes and the real-world connectivity of the nodes in the Pharos testbed.

Real-world connectivity between wireless nodes often varies, sometimes to a large degree, from simulated connectivity. A great deal of research has improved simulator radio models to achieve better accuracy, however there is still a marked difference between the state-of-the-art wireless simulators and the real-world [1, 14]; ultimately, nothing beats evaluating a mobile computing solution *in situ*. This variance in communication characteristics is one of the most compelling reasons to evaluate mobile cyber-physical system solutions using real-world experiments in addition to simulations. To begin to understand the relationships between the real-

world connectivity in the Pharos testbed and simulation results, we measured connectivity by sending wireless beacons between the nodes in all of our experiments and recorded when any node saw another nodes' beacon. The beacons themselves were UDP packets sent to the subnet broadcast address by the Pharos middleware (at user-configurable intervals) containing a sequence number to allow us to track which beacons were lost.

5.1 Experimental Setup

To compare the Pharos testbed to simulation, we use the OMNeT++ network simulator [28]. For the simulated radio within OMNeT++, we used the unit disk radio model which considers a nodes' wireless range to be a perfect circle; if two nodes are in range of each other they are considered to be connected. Although this is a naïve radio model, it is popular and often used for simulation. It is also straightforward to replace this radio model with a more sophisticated one in OMNeT++; we use it to get an initial quantification of differences between the Pharos testbed and simulation.

We sent nine nodes along the lollipop mobility script in Figure 4, starting each node 30 seconds apart. Temporally separating the nodes in this manner reduces the likelihood of collisions at the start of the experiment. The nodes were configured to broadcast beacons on a random interval between five to ten seconds apart. To precisely simulate the same experiment, instead of using the input waypoint set for simulation (Figure 4), we converted the actual GPS traces recorded by the nodes (containing the node's coordinates at every GPS sample point) into an OMNeT++ mobility file for the simulations as described in Section 4.4. This ensured that the simulated nodes moved along the same trajectories as the real nodes including any discrepancies between the intended path and the actual path taken.

In the OMNeT++ simulation, we recorded the number of neighbors each node had at each sample point and compared it to the recorded beacons showing which nodes were actually connected at that same sample point in the Pharos testbed experiment. These sample points were defined based on the beacon interval in the Pharos testbed; to ensure fairness for the simulator, the finest grained sampling rate we could use was 10 seconds. We compared one real-world run of this experiment with simulations using simulated radio ranges of {10m, 25m, 50m, 75m, 100m, 150m, and 200m}.

5.2 Neighbor Connectivity Differences

To illustrate the difference between real-world connectivity (based on received beacons) and simulated connectivity (based on an ideal radio model), we plot the simulated connections and real connections seen by each node during every 10 second interval for various simulated radio ranges. Figure 12 shows the connectivity we measured on the first node (Manny) to start the experiment (chosen because it had the longest running time) plotted alongside a simulation of the same mobility and communication trace using a simulated radio range of 50m.[6] The figure shows a number of possible connections (around t=300s) present in simulation but missed by the Proteus node and also illustrates that the simulated connectivity (although close to the real-world data) consistently reported one to two extra neighbors near the end of the simulation. Naturally, we observed that the chosen simulated radio range had a significant effect on the number of extra neighbors. However with an appropriately chosen simulated radio range, the connectivity trends appeared to be the same between the real-world and simulation for most of the graphs.

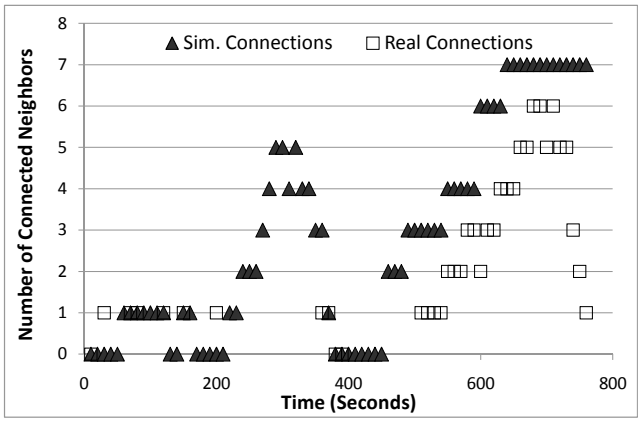


Figure 12: Real-world connections vs. simulated connections (using 50m simulated radio range)

We do not average multiple real-world runs together into one connectivity data set since the unpredictability of communication across runs (not to mention across different nodes even within the same run) is one of the more interesting aspects of running real-world experiments and one of the main motivations for building the testbed in the first place. In the future, Pharos could be used to collect data to build an empirical radio model, but that is out of the scope of this paper.

5.3 Comparing Effective Radio Ranges

From our empirical results, we are also able to estimate the effective radio range of a given experiment. To reason about the real-world range experienced by the 802.11 radios on the Proteus nodes, we compared the number of recorded neighbors a node saw in a ten second interval to the "expected" number of neighbors from simulation. We accounted separately for neighbors that were *missing* in the simulation but present in

[6]Due to space constraints, all of the results cannot be included. For the complete set of graphs for every node, please visit: `http://pharos.ece.utexas.edu/wiki/index.php/Evaluating_Connectivity`

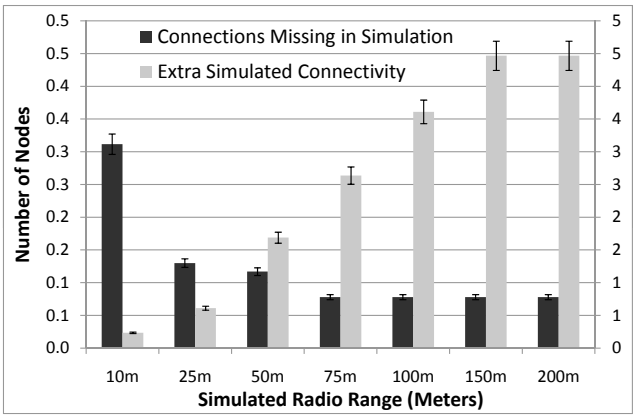


Figure 13: Neighbors missing from simulation (left axis) and extra neighbors in simulation (right axis) vs. simulated radio range

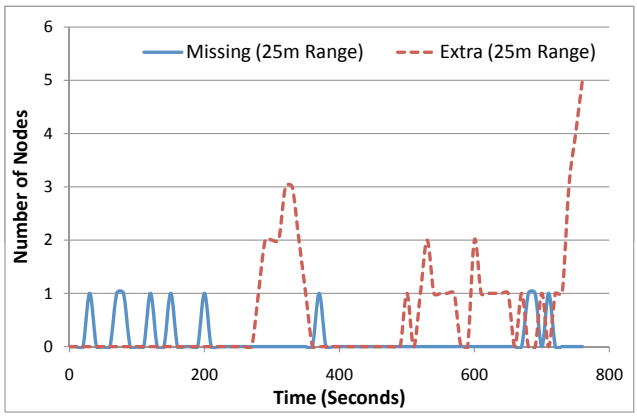


Figure 14: Neighbors vs. time for simulation (25m radio range) and the real-world

the real-world (which was rare) and neighbors that were *extra* in the simulation but unaccounted for in the real-world (which was, as expected, common). This data is presented in Figure 13 with 95% confidence intervals. It is interesting to note that any simulated radio range greater than 50 meters, on average, results in more than two extra neighbors that the real-world node "should" have been able to see but did not. This could be due to any number of reasons such as interference (although there was only one detectable wireless network in range of the experiment, with no connected users) or wireless propagation effects, given that our radios were approximately one foot off the concrete parking lot. Additional experiments resulted in similar real-world ranges.

From these results, we determined that the optimal simulated radio range is surprisingly low: very close to 25m. This is surprising in that it is very short given the large unobstructed open space we used for the experiment. Figure 14 uses this 25m radio range and plots the neighbors missed by the simulator and extras that were not seen in the real world; this represents the best-case of all of the radio ranges we tried in simulation.

6. LIMITATIONS & LESSONS LEARNED

In this section, we explore limitations of the Pharos Testbed, some of which were overcome in performing this work while others still remain. The examination of these limitations provide insights for researchers with similar efforts and drive our future work with Pharos.

Limitations in the Architecture. The Pharos testbed currently contains several limitations that must be overcome before it can attain widespread use. One limitation is due to node complexity and the unpredictable ways in which the cyber and physical worlds interact, resulting in frequent device failures that limit the scale of mobile cyber-physical system experiments. The testbed contains over 30 Proteus nodes, but we were frequently limited to experiments involving only ten nodes due to various hardware failures. Overcoming this limitation will require novel diagnostic tools to detect failures, automated procedures for fixing them, and evolving our hardware and software in ways that render them inherently more robust against failure. Our current manual approach to detecting and resolving hardware and software problems on a case-by-case basis is not sustainable as the number of nodes increases.

Limitations in Pharos Software Architecture. Another limitation resides in the flexibility of the software architecture. The current software only supports one form of motion script based on GPS waypoints. Expressing acceleration or complex patterns like weaving around obstacles is cumbersome using the current motion script syntax. Conditional and probabilistic movements are not supported. In addition, there is no coordination between motion and wireless communication, so a node cannot change its mobility based on communications received. This is acceptable for the evaluations in this paper because we focused on the repeatability of motion and the properties of wireless connectivity, which can be done using beacons emitted independent of physical position. Future experiments will demand a Pharos client that can more flexibly and seamlessly allow deployment and initiation of arbitrary software on the Proteus. We envision creating a semantically rich mobile cyber-physical experiment middleware that can capture even the smallest nuances of motion, communication, and their interaction. Like a musical score sheet, it will enable the musician (a mobile cyber-physical system researcher) to orchestrate (control) numerous independent instruments (mobile nodes) simultaneously in a rhythmic (temporally-accurate) manner.

Limitations in Hardware and Device Drivers. Additional challenges relate to the underlying hardware and its interactions with the physical world. The micro-

controller is computationally weak and highly sensitive to interrupt latencies. Even the act of toggling the LEDs too rapidly will interfere with other interrupts and result in non-deterministic behavior. To achieve reliability, we had to carefully design the micro-controller software components to maximize efficiency by fully exploiting all available hardware. For example, a separate hardware counter was used for every periodic task, and only a minimal amount of computation is done within an interrupt context. This was achieved by implementing a dual-layer thread model based on a task queue into which interrupt handlers can post long-running computations that execute in a foreground context.

Another lesson learned is related to the Proteus' Li-polymer batteries. When we first started large experiments, the nodes would occasionally stop moving. This was traced to a safety mechanism that shuts down the battery when the current exceeds 6A, which can occur when the node accelerates too quickly or encounters excessive resistance. We installed an inrush-limiting power thermistor and modified the software to limit acceleration and cut the motor power when a discrepancy occurs between the tachometer and the power being sent to the motor. This, however, does induce limitations in the achievable mobility patterns.

The default drivers for the Atheros wireless chip sets are unreliable: when the nodes are configured to beacon at 1-$\frac{1}{2}$Hz, they frequently fail. We had to decrease the beaconing period to 5-10s before we could achieve acceptable reliability. In the future, we plan to use the MadWiFi drivers, which we expect to be more stable.

With respect to monitoring location and heading, we often encounter challenges in sensing. The compass is highly-sensitive to voltage fluctuations, which frequently occur when the node is moving. This is an on-going problem that we are still trying to resolve. The GPS device sometimes has difficulty locking on the satellites; we suspect that the electrical noise emitted by the node's electronics and the metal surface of the node may be interfering with the sensors. We rely on additional software to filter apparently "bad" GPS readings and delay node movement. This is a challenge for the Pharos testbed, but it is also reflective of the real world, where commodity GPS devices often have similar issues.

Limitations of Experiments to Date. The experimental results described in this paper indicate that we need to better control controllable factors in our experiments. For example, one is the starting location and orientation of each node. We failed to start the nodes at the exact same location and with the exact same orientation with each execution of the motion script. This resulted in widely varying motion behaviors as the nodes traveled towards the first way point. In the future, more care should be taken to minimize variations across experiments. While we have experimented with various mobility patterns, the in-depth results in this paper rely on a single motion script. We have not evaluated how the motion script itself may influence repeatability.

Our evaluations do not yet evaluate how speed affects motion repeatability or wireless connectivity. The evaluations presented in this paper involve nodes traveling at 1.5m/s. We have tested the nodes at other speeds ranging from 0.5m/s to 3m/s, but we have not yet determined the limits of reliability and repeatability.

In summary, the Pharos testbed is inherently extremely complex. As we scale to more nodes, we will create new hardware and software mechanisms that further automate the use of the testbed, from problem diagnosis and resolution to the deployment of arbitrary MCPS experiments. We are heartened by the results in this paper, which indicate that despite vagaries in node behavior, Pharos achieves relatively consistent behavior.

7. RELATED WORK

We are motivated by the significant gap between theoretical and experimental research for mobile cyber-physical systems (MCPS). Researchers generally evaluate mobile system components using mathematical models and software simulation. For tractability, these approaches make significant assumptions about the environment and the network; these assumptions misrepresent real networks [1, 2, 14]. Instead of relying on software simulation, examining, validating, and evaluating in a real deployment is ideal. Testbeds for evaluating networked systems, and, more specifically, mobile and pervasive computing environments, have become common [9, 16, 19, 22]. We do not provide an exhaustive survey here but instead focus on a few approaches that are related to or motivate our proposed work.

Judd et al. have developed a wireless emulator that focuses on repeatability at a very fine level of granularity at the physical layer [12]. The emulator uses real application input to drive physical layer measurements. While this approach allows researchers to elicit application performance characteristics, the emulator does not modularize arbitrary mobile cyber-physical systems, execute complete *mobile* experiments, and connect these results to results from simulation. Hydra uses a software defined radio to allow fine-grained measurements of MAC and physical layer protocols [17]. Hydra has motivated our use of a modular software architecture, but it does not address mobility concerns or the high-level applications that drive MCPS deployments.

MiNT is a miniaturized multi-hop wireless network testbed that connects live emulations with running simulations in real time [7]. MiNT uses radio attenuation for devices in the testbed, allowing it to operate in a small space. Bonsai validates that attenuated radio links do not correctly mimic actual links in all cases [20]. Similar to MiNT's connection to ns-2, TWINE com-

bines emulated links with simulated ones [30]. ORBIT also uses wired links and significant mathematical computations to emulate wireless links [25]. Pharos, in comparison to these approaches, promotes the use of real wireless links with real applications and real mobility.

Many existing approaches focus on the wireless nature of the networks but ignore mobility aspects. EWANT [26] and Meshtest [27, 29] address mobility and maintain the small testbed environment; they simulate movement by changing attenuation properties of emulated nodes. APE was one of the first to incorporate real mobile nodes [21]; however APE nodes are not autonomous. For reproducibility and common usage, we require a testbed that can run repeatedly on its own.

MiNT-m extends MiNT to make nodes mobile while still connecting live experiments with simulation [6]. MiNT-m is limited by its choice of technologies (i.e., it can only operate indoors) and its use of attenuated links. MiNT-2 redesigns MiNT-m to reduce some of the limitations but is still limited to indoor operation with attenuated radios [18]. Mobile Emulab is an indoor testbed to execute real applications in a radio-attenuated environment [10] and connects the nodes to a situated network of sensing devices, allowing for expressive MCPS situations. Like MiNT-m, the testbed is tied to a specific room, and the links may not represent real-world links due to attenuation. Brown et al. have built a wireless testbed of unmanned aerial vehicles (UAVs), with purpose-constructed hardware [5]. The testbed was specifically designed to monitor network quality and therefore has reduced ability to support arbitrary updates to both hardware and software.

With respect to connecting testbed experiments to simulation, Liu et al. have devised a testbed that directly executes traces of simulation output [15]. In comparison to our motivation, we aim to directly execute mobile cyber-physical solutions in the Pharos testbed and to automatically generate simulatable versions of them for large-scale tests. This has the benefit of directly evaluating the solution (and all of the performance metrics enabled by that direct evaluation) instead of a trace that merely represents the behavior.

The EXC Toolkit focuses on the software components of a wireless multihop network, distancing these software concerns from any specific hardware [13]. This work is a springboard for our work in managing the large number of software artifacts required in Pharos, but the distancing from the hardware is disadvantageous since the EXC user now must control the hardware deployment and management of mobility and other physical aspects by hand. Pharos motivates conjoining these concerns with respect to a given mobile cyber-physical system scenario to get an integrated understanding of the interplay of hardware and software.

We focus on end-to-end evaluation for cyber-physical applications on real hardware for sensing, computation, communication, and actuation and use real software modules deployed in those mobile cyber-physical systems. Significant differences between our approach and existing efforts include the combined use of real links and real, autonomous, and repeatable mobility. We also aim to eventually support the direct translation of software from a testbed to software simulation. Finally, we require both hardware and software modularity and the ability to evaluate metrics at all levels of abstraction. We focus on reproducibility (i.e., the ability to repeat experiments within the testbed reliably and to reproduce the results of experiments in simulation) and on the ability to easily and swiftly compare solutions that tackle similar problems. In addition, because the testbed builds directly on devices that would be used in an actual deployment, Pharos is capable of completely and directly emulating target mobile cyber-physical systems.

8. CONCLUSIONS

In this paper, we identify and quantify the many unique challenges posed by mobile cyber-physical systems (MCPS). To this end, we present the Pharos testbed, a modular, flexible, COTS-based and thus inexpensive and scalable testbed for MCPS. Pharos offers push-button repeatability giving insight into the diversity of behaviors that a cyber-physical system exhibits across runs, despite using the same hardware, software, and physical environment. Using Pharos, we pinpoint cyber and physical variations across runs and empirically establish the relationship between physical-properties and cyber-properties. We also demonstrated that our nodes exhibit repeatable motion patterns with narrow confidence intervals of just fractions of a meter despite the unpredictable-and-thus-uncontrollable dynamics inherently present in a MCPS. Finally, we demonstrate that existing simulators cannot capture the complexity of even small MCPS consisting of only tens of mobile nodes, but provide a mechanism by which one such simulator, OMNeT++, can be tuned to more accurately match real-world MCPS behavior.

9. REFERENCES

[1] G. Anastasi, E. Borgia, M. Conti, E. Gregori, and A. Passarella. Understanding the real behavior of mote and 802.11 ad hoc networks: An experimental approach. *Pervasive and Mobile Computing*, 1(2):237–256, 2005.

[2] T. R. Andel and A. Yasinsac. On the credibility of manet simulations. *IEEE Computer*, 39(7):48–54, July 2006.

[3] M. Anderson, L. Thaete, and N. Wiegand. Player/Stage: A unifying paradigm to improve robotics education delivery. In *Proceedings of the*

Workshop on Research in Robots for Education, 2007.

[4] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *Proc. of MobiCom*, pages 31–42, August-September 2005.

[5] T. X. Brown, S. Doshi, S. Jadhav, and J. Himmelstein. Test bed for a wireless network on small UAVs. In *Proc. of the AIAA 3rd "Unmanned Unlimited" Tech. Conf.*, pages 1–8, September 2004.

[6] P. De, A. Raniwala, R. Krishnan, K. Tatavarthi, J. Modi, N. A. Syed, S. Sharma, and T.-C. Chiueh. MiNT-m: an autonomous mobile wireless experimentation platform. In *Proc. of MobiSys*, pages 124–137, June 2006.

[7] P. De, A. Raniwala, S. Sharma, and T. Chiueh. Mint: a miniaturized network testbed for mobile wireless research. In *Proc. of Infocom*, pages 2731–2742, March 2005.

[8] P. De, A. Raniwala, S. Sharma, and T. C. Chiueh. Design considerations for a multihop wireless network testbed. *IEEE Comm. Magazine*, 43(10):102–109, October 2005.

[9] M. Dyer, J. Beutel, T. Kalt, P. Oehen, L. Thiele, K. Martin, and P. Blum. Deployment support network: A toolkit for the development of WSNs. In *Proc. of EWSN*, pages 195–211, January 2007.

[10] D. Johnson, T. Stack, R. Fish, D. M. Flickinger, L. Stoller, R. Ricci, and J. Lepreau. Mobile emulab: A robotic wireless and sensor network testbed. In *Proc. of Infocom*, April 2006.

[11] D. B. Johnson, D. A. Maltz, and J. Broch. DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. *Ad Hoc Networking*, 1:139–172, 2001.

[12] G. Judd and P. Steenkiste. Using emulation to understand and improve wireless networks and applications. In *Proc. of NSDI*, May 2005.

[13] W. Kiess, T. Ogilvie, and M. Mauve. The EXC toolkit for real-world experiments with wireless multihop networks. In *Proc. of WoWMoM*, pages 1–6, June 2008.

[14] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. In *Proc. of MSWiM*, pages 78–82, 2004.

[15] J. Liu, Y. Yuan, D. M. Nicol, R. S. Gray, C. C. Newport, D. Kotz, and L. F. Perrone. Simulation validation using direct execution of wireless ad-hoc routing protocols. In *Proc. of PADS*, pages 7–16, May 2004.

[16] D. A. Maltz, J. Broch, and D. B. Johnson. Quantitative lessons from a full-scale multi-hop wireless ad hoc network testbed. In *Proc. of WCNC*, pages 992–997, 2000.

[17] K. Mandke, S. Choi, G. Kim, R. Grant, R. C. Daniels, W. Kim, R. W. Heath, and S. Nettles. Early results on Hydra: A flexible MAC/PHY multihop testbed. In *Proc. of VTC*, pages 1896–1900, April 2007.

[18] C. Mitchell, V. P. Munishwar, S. Singh, X. Want, K. Gopalan, and N. B. Abu-Ghazaleh. Testbed design and localization in MiNT-2: A miniaturized robotic platform for wireless protocol development and emulation. In *Proc. of COMSNETS*, January 2009.

[19] T. Miyachi, K.-I. Chinen, and Y. Shinoda. StarBED and SpringOS: Large-scale general purpose network testbed and supporting software. In *Proc. of ValueTools*, 2006.

[20] V. Naik, E. Ertin, H. Zhang, and A. Arora. Wireless testbed bonsai. In *Proc. of WiOpt*, pages 1–9, April 2006.

[21] E. Nordstrom, P. Gunningberg, and H. Lundgren. A testbed and methodology for experimental evaluation of wireless mobile ad hoc networks. In *Proc. of TridentCom*, pages 100–109, February 2005.

[22] L. Peterson, A. Bavier, M. E. Fiuczynski, and S. Muir. Experiences building PlanetLab. In *Proc. of OSDI*, 2006.

[23] `http://pharos.ece.utexas.edu`.

[24] `http://proteus.ece.utexas.edu`.

[25] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols. In *Proc. of WCNC*, pages 1664–1669, March 2005.

[26] S. Sanghani, T. X. Brown, S. Bhandare, and S. Doshi. EWANT: The emulated wireless ad hoc network testbed. In *Proc. of WCNC*, pages 1844–1849, March 2003.

[27] M. Seligman, B. D. Walker, and T. C. Clancy. Delay-tolerant network experiments on the meshtest wireless testbed. In *Proc. of CHANTS*, pages 49–56, September 2008.

[28] A. Varga. The OMNeT++ discrete event simulation system. In *Proc. of ESM*, pages 319–324, June 2001.

[29] B. D. Walker, I. D. Vo, M. Beecher, and M. Seligman. A demonstration of the meshtest wireless testbed for delay-tolerant network research. In *Proc. of CHANTS*, pages 105–108, September 2008.

[30] J. Zhou, Z. Ji, and R. Bagrodia. TWINE: A hyrbid emulation testbed for wireless networks and applications. In *Proc. of Infocom*, pages 23–29, April 2006.