Experiences Using a Miniature Vehicular Network Testbed

Agoston Petz, Chien-Liang Fok, and Christine Julien
The University of Texas at Austin

{agoston, liangfok, c.julien}@mail.utexas.edu

TR-ARiSE-2012-006

ARiSE
Advanced Research in Software Engineering

Experiences Using a Miniature Vehicular Network Testbed *

Agoston Petz, Chien-Liang Fok, and Christine Julien
The University of Texas at Austin
{agoston, liangfok, c.julien}@mail.utexas.edu

ABSTRACT

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 present 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.

Categories and Subject Descriptors

C.2.0 [**Computer Systems Organization**]: Computer Communication Networks

Keywords

Vehicular Networks, Testbeds, Mobility Frameworks, Performance

1. INTRODUCTION

While research in mobile and opportunistic vehicular networks is becoming common, the effects of real-world mobility on network and application performance are not well-understood. Potential deployments of opportunistic vehicular networks include urban networks in which wireless encounters between cars and buses are used to transfer messages from disconnected portions of the network to areas with Internet access [12]—such disruption-tolerant (or delay-tolerant) networks make use of *data ferries*, i.e., autonomous roving vehicles that provide Internet connectivity [5, 22]. When vehicles travel together on a highway, they communicate to form convoys and exchange information about future hazards to increase both efficiency and safety [7]. As autonomous vehicles increase in numbers, stop signs and traffic lights can be replaced with new intersection management schemes in which approaching vehicles communicate to traverse the intersection while reducing delays [10]. In all of these examples, connections unpredictably appear and disappear though in ways that vary between scenarios and significantly impact communication protocol performance.

Vehicular mobility affects communication protocol performance often in ways that are difficult to understand and quantify. Evaluations are usually performed in simulators or testbeds that restrict experiments to a small number of mobility models and generally decouple measuring mobility properties from network properties. We believe that given the significant interdependencies between movement and network performance, evaluations must consider both mobility and the network stack to understand the nature and magnitude of their interactions.

Evaluations of vehicular networks often rely on simulations that often inaccurately model real network stacks [1, 20]; however simulations can scale to very large systems that would be difficult to implement for real, and many provide increasingly realistic network behavior, e.g., in terms of radio propagation models and network protocols. What is sorely lacking, however, is the ability to acquire a concrete and empirical understanding of the impact of vehicular mobility on real-world network protocol performance. This makes it challenging to understand how a new protocol or application for mobile vehicular networks will be affected by different mobility scenarios; simulation cannot model every facet of a real-world system.

While simulations are important, they do not enable complete understanding of opportunistic vehicular network behavior. For this reason, we developed a mobile autonomous vehicular network testbed called Pharos for small to medium scale network evaluations. Although our nodes are not full-sized vehicles, our testbed serves as a bridge between simulation and full-scale vehicular network deployments; we provide the ability to run experiments using real network hardware and software on a system that provides "push-button" repeatability, something that is difficult, if not im-

*This material is based upon work supported by the National Science Foundation under Grant No. 844850 and the Department of Defense. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of sponsoring agencies.

VANET'12, June 25, 2012, Low Wood Bay, Lake District, UK.

possible, to do with real vehicles that are available today. Pharos improves upon the information provided by typical testbed-based experiments by explicitly tying network protocol performance in the testbed to network statistics gathered through simulation. This allows us to reason about how testbed results will scale beyond the physical limitations of the testbed.

In this paper we introduce a pair of frameworks (one for simulation, and one for real networks using Pharos) that jointly evaluate mobility and network performance. This means one can associate statistics that *quantify* the mobility observed in the experiment and quantitatively relate mobility to network performance. Each framework can enact numerous mobility models or mobility traces and enable simultaneous collection of statistics about node movement and network protocols. Our frameworks are unified; we can collect the same network performance statistics and execute the same mobility patterns in simulation as we do in the real testbed and vice versa. The first of our pair of frameworks performs this task within a well-known and widely used network simulator, OMNeT++ [25]. The second framework transitions these mobility models to Pharos [24], a real-world testbed based on miniature autonomous vehicles, allowing the same kind of mobility driven validation on real vehicular nodes. We describe OMNeT++ and Pharos, justify our use of them, and provide details of our frameworks' implementations in Section 4. We use these frameworks to evaluate the differences between simulated and real-world network performance and mobility of a small vehicular network in Section 5. The paper ends with lessons learned in Section 6.

2. RELATED WORK

Before describing our frameworks in detail, we first consider the state of the art in modeling, simulating, and controlling mobility.

The seminal work characterizing mobility models' effect on network performance compared several random models with more realistic mobility patterns [6]. However, statistical analysis was not applied to newer more realistic mobility models. Baumann *et al.* developed a Generic Mobility Simulation Framework (GMSF) [3] for comparing mobility models for vehicular communication networks. GMSF can export simulation traces to formats accepted by many network simulators and computes several statistics that can be used to reason about the mobility models. However, GMSF omits a number of statistics that are important to opportunistic networks, e.g., network partition sizes and memberships, and lacks a realistic radio model forcing researchers to export traces to other simulators to use the latest radio models—this prevents evaluation of scenarios in which mobility and communication influence each other [5].

Our desire to collect comprehensive statistics to characterize the differences between mobility models is not unique. BonnMotion [9] implements several popular mobility models and provides a comprehensive statistical analysis of each. However, similar to GMSF, BonnMotion is not a simulator but rather a movement trace generator and mobility analysis tool—it can therefore not capture the complex relationship between mobility and communication. Other frameworks provide statistics on network performance given mobility models as input [17, 16] but none of these tie real-world results to simulated results for the same mobility input.

The challenge is somewhat different in testbed evaluations. Here, real mobility is required as the network nodes operate in physical space. Numerous autonomous multi-vehicle testbeds exist. For example, Kolodko and Vlacic used golf-cart-like Imara vehicles [19] and focus on evaluating an autonomous intersection; we focus on analyzing the differences between multi-vehicle simulations and real-world behavior. RAVEN [13] is an indoor testbed for UAV control systems, and, like many other autonomous robotic testbeds, the focus is on control systems and not on the validation of simulated communication protocols and the analysis of its performance. Other testbeds for mobile networks in general enable mobile devices to follow traces of mobility provided by the experimenter [4, 8, 15, 21]. To our knowledge, there is no existing work on providing a framework that directly incorporates mobility modeling into the testbed, allowing a single testbed to execute different mobility models representing different targeted scenarios while evaluating both the network performance and statistics of the mobility pattern.

3. CHARACTERIZING PHAROS

The Pharos vehicular testbed consists of numerous autonomous vehicles called *Proteus*. Designed for modularity and economy, Proteus uses commercial-off-the-shelf (COTS) equipment to maximize robustness, flexibility, and cost-effectiveness. Below we discuss the design in four major functional sections: software support, mobility, behavior, and interaction.

Software support. To reduce component coupling, we use well established APIs whenever possible. The Player API [2] allows movement definition (e.g., drive forward) to be fully differentiated from the hardware implementing the behavior. 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 [2] for movement or OMNeT++ [25] for networking). Once validated individually, they can be quickly integrated with the nodes.

Physical Mobility. Physical mobility is provided by one of three options: an iRobot Create®, a Segway® RMP50, or a customized Traxxas Stampede, a high-performance remote controlled truck 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, lightweight, outdoor compatibility, and range of speeds makes it a desirable option. Details on the hardware, assembly, and software are all available publicly [24].

Behavior and Communications. A low-power VIA EPIA® x86 Linux-based computer coupled with a Freescale™ 9S12 micro-controller (MCU) constitutes the platform for Proteus node used in this paper. This dual architecture offloads low level tasks to the micro-controller while allowing the x86 to focus on higher level aspects. The MCU opens a wide range of I/O options for connecting sensors and other peripherals. Basic communication is provided by an on-board 802.11 b/g wireless interface controller with a 5.5 dBi antenna.

Environmental Interaction. The Proteus's fourth functional area is sensing and actuating. We support various range-finding sensors, a digital compass, global positioning system (GPS), and cameras, as well as ambient sensing devices including MEMSIC® motes.

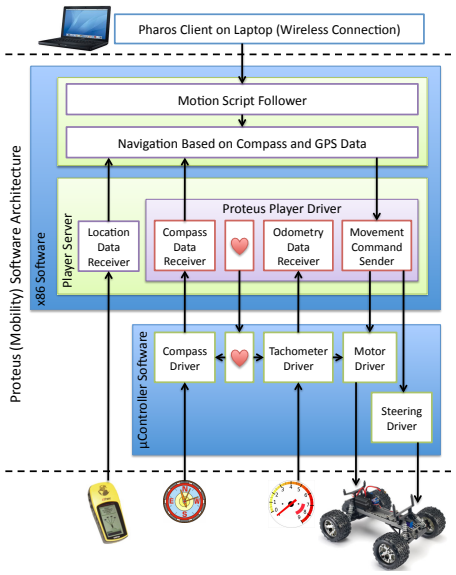

Figure 1: The Proteus Mobility Architecture

3.1 Software Architecture

The Pharos testbed's software architecture is shown in Figure 1. At a high-level, it consists of three main components: a Pharos client residing on a laptop that wirelessly communicates with one or more Proteus nodes, the Pharos Server 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 assigns motion scripts to Proteus nodes and initiates the execution of the motion script. Upon receiving the motion scripts and experiment configuration, which contains the node specifications and motion script assignments, the Pharos client wirelessly connects to the Pharos servers on each node, configures them, 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 log files, organizing them by experiment identifier and node ID.

Pharos Server. The Pharos server 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; upon arrival it pauses for the specified amount of time and repeats the process. The software that implements the network protocol being evaluated runs in parallel with the Motion Script Follower and can influence the sequence of waypoints that a node visits and the speed at which it travels. The Navigation component requires compass and GPS data, using both to adjust the steering angle and speed. The Navigation component obtains the sensor data and issues the movement commands through a Player server that also runs on the x86. Hardware actuation and feedback is accomplished through a combination of the well-known Player Server robot API and custom micro-controller drivers.

4. FRAMEWORK TO SUPPORT NETWORK EXPERIMENTS

We developed Pharos because it is insufficient to rely *only* on simulation to evaluate vehicular networks, and we wanted a real-world mobile testbed upon which to test such systems. To establish correlations between simulated and real-world performance, we provide a pair of unified frameworks in which to evaluate and quantify the impacts of mobility on protocol performance in opportunistic vehicular networks. The framework has two components, one that runs in the OMNeT++ simulator, and one that runs on the Proteus nodes. To unify simulation and testbed results, it is important to execute the same code in both places. This increases the validity of the comparison and decreases the overhead of moving experiments between simulation and the real world. For the simulation-side, we chose to use the OMNeT++ simulator and for the Pharos testbed framework we use the Click modular router [18]—the two have similar architectures easing the burden of porting protocol implementations.

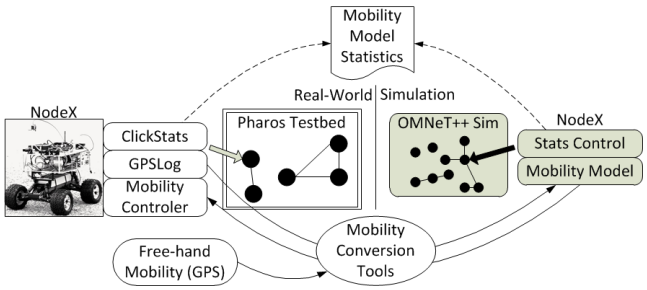

Figure 2: Unified Simulation and Testbed Analysis Framework

Figure 2 shows the general architecture of our frameworks; it consists of three parts, (*i*) *StatisticsControl*: the OMNeT++ mobility model analysis framework, (*ii*) *ClickStats*: the Click-based testbed implementation of the same, and (*iii*) tools to transfer mobility traces from simulation to the real-world, and *vice versa*.

4.1 OMNeT++ Statistics

We previously designed a *StatisticsControl* package for OMNeT++ to help us reason about how characteristics of node mobility affect routing performance and other higher layer protocols [23]. These statistics enable us to draw conclusions about the similarities and differences between mobility models. For example, it is interesting to discover that two models that appear to be quite different on the surface result in similar values for those metrics that govern the performance of a particular routing protocol. *StatisticsControl* works within the INET framework for OMNeT++ versions 3.3 and 3.4 and collects the following statistics for every node in a network:

Core Statistics: The module samples each node on a user-defined interval. It collects the following basic statistics for each node in the network and records their time-varying values and their averages: node position, number of neighbors, and number of unique neighbors (which indicates how many other unique nodes the node encountered during the course of the simulation).

Connection and Partition Tracking: The module records possibilities for connections, or "contacts," every time two nodes come within ideal radio range. It records the number of these contacts over the lifetime of the simulation and their average duration per node. It also tracks the number of partitions in the network, their sizes, their

memberships, and the number of disconnected nodes at any given time.

Relative Mobility: The module also samples the position of all nodes on a separate user-defined timer and records the relative velocities of all nodes with respect to each other. This metric is used to calculate the network's total relative mobility using the algorithm given in [14]. Relative mobility is useful for describing the level of node movement in a network.

Message Delivery Possibilities: We have also defined a notion of potential message delivery opportunity. A given experiment can be split up into any number and duration of "epochs," during which every node starts with a unique message and attempts to deliver it to every other node. For the heuristic, any recorded contact between two nodes results in a *hypothetical* exchange of every unique message they currently hold, and each node records the time at which it first receives any message. The "routing algorithm" is thus an oracle that shows how perfect routing could perform if every contact was fully utilized. Although unrealistic, this establishes a best case delivery latency for every message, and one can also easily see which (if any) messages remain undelivered at the end of an experiment. Since the user can define any number of epochs, it is easy to observe how the delivery potential changes over time due to the nature of the mobility model.

4.2 Click-based Real-World Statistics

For this paper we developed a real-world implementation of the statistics package that we previously built for OMNeT++. The real-world implementation uses the Click Modular Router [18] and runs on the Linux-based robots of the Pharos Testbed (see Section 3). The Click-based statistics package uses the 802.11 radios of the Proteus robots to send and receive UDP beacons at user-defined intervals to directly measure the **core statistics** described above. The other statistics **connection and partition tracking**, **relative mobility**, and **message delivery probabilities** are *global* statistics in that they cannot be sensed or estimated without global information. They are instead computed offline using GPS data to estimate relative mobility and using the connectivity information to estimate both the connections and partitions and message delivery probabilities.

4.3 Conversion Tools

Another key element of our dual mobility model analysis frameworks is the ability to transfer mobility traces from simulation to real-world nodes, and *vice versa*. To accomplish this, we built a set of tools to automate the conversion of OMNeT++-style mobility traces into GPS waypoints that can be followed by the Proteus robots. These conversion tools coupled with our *TraceMobility* module allow us to recreate the movements of *any* real-world GPS-based node traces in OMNeT++. Finally, it is also possible to draw freehand mobility models using a GPS-based mapping tool such as GPSVisualizer [11] and to run these on both the Pharos testbed and in simulation.

5. EARLY PHAROS TESTBED RESULTS

In this section we present several sets of early results using Pharos. The first uses our framework to evaluate differences between real-world connectivity and simulated connectivity and demonstrates our methodology for applying simulated mobility patterns to real (small scale) vehicular mobile autonomous nodes. The next two quantify other aspects of the testbed.

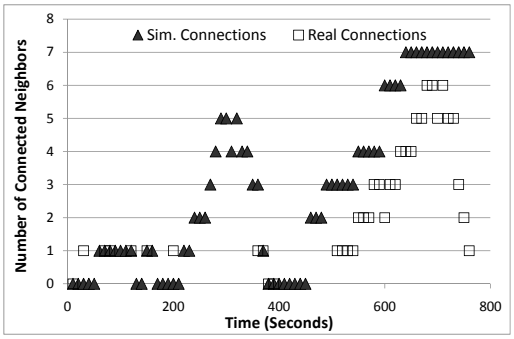

Figure 3: Real-world vs. simulated connections (using 50m simulated radio range)

5.1 Comparing Simulation to the Real-World

Our framework allows us to analyze mobility models using Pharos with traces generated in OMNeT++ and to compare the results. We also sought to characterize the difference between simulated connectivity between mobile nodes and the real-world connectivity of the vehicles in Pharos. To begin to understand the relationships between the real-world connectivity in Pharos 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 Pharos' middleware (at user-configurable intervals) containing a sequence number to allow us to track which beacons were lost.

We sent nine Proteus vehicles along a lollipop-shaped mobility script in Figure 6, starting each robot 30 seconds apart. Temporally separating the nodes in this manner reduces the likelihood of robot 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 using OMNeT++, we converted the resulting traces of the nine robots and ran them in simulation using our *TraceMobility* model.

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}.

Neighbor Connectivity Differences: To illustrate the difference between real-world connectivity and simulated connectivity, we plot the simulated connections and real connections seen by each node during every 10 second interval for various simulated radio ranges. Figure 3 shows the connectivity we measured on one node alongside a simulation of the same mobility and communication trace using a simulated radio range of 50m. 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

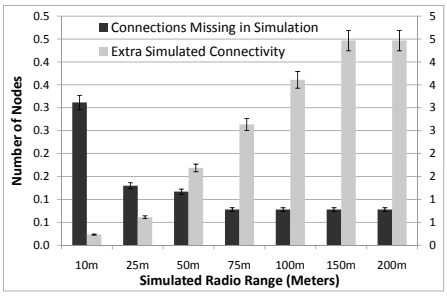

Figure 4: Neighbors missing from simulation (left axis) and extra neighbors in simulation (right axis) vs. simulated radio range

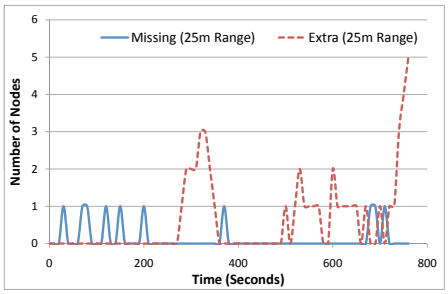

Figure 5: Neighbors vs. time for simulation (25m simulated radio range) and the real-world

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.

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 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 4 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 a number of reasons such as interference 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.

5.2 Experiment Repeatability with Pharos

To analyze our claim of Pharos' "push-button repeatability," we analyze geographical *path divergence* between different runs of the same experiment.To ensure comprehensive evaluation of the vehicle's motion characteristics, the motion script contains segments of varying length and turns of different directions and angles. The actual paths that one vehicle took for these seven trips are shown in Figure 6. While the vehicle did not travel the exact same path each time, it did follow the same general path. Figure 7 shows Lonestar's average absolute divergence over the seven runs, i.e., the distance between the node's route and the ideal route. The error bars indicate 95% confidence intervals. Note that the average divergences are 0-5m and the confidence intervals are 0-1.29m, which are small relative to 45m, the average edge length. The small divergence values is evidence that Pharos achieves movement repeatability across experiment rounds.

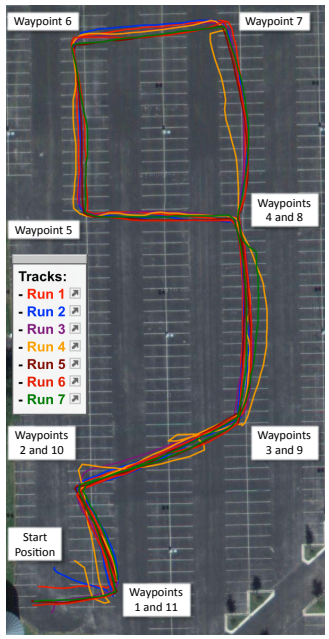

Figure 6: Seven executions of the lollipop-shaped motion script

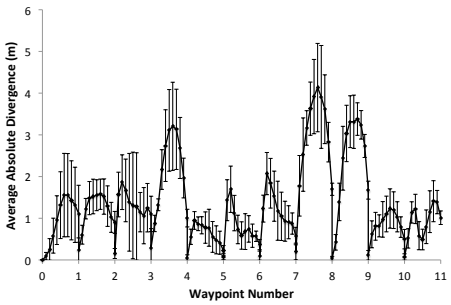

Figure 7: Absolute path divergence

6. LESSONS LEARNED & CONCLUSIONS

Simulations of vehicular networks heavily depend on the choice of mobility model as the varying connectivity opportunities presented by each greatly impact network performance; the connectivity heuristics of different mobility models and real mobility traces present different challenges to network protocols. These differences in mobility models are not extensively categorized, and the relationships between mobility and the ultimate aim of "good network

performance" are unexplored. Our frameworks provide a mechanism to not only reason about the differences between mobility models, but to also examine their *best case* routing performance—in both simulation and the real-world.

In examining differences between real-world and simulated results of the same mobility pattern, we also learned that real-world GPS-based autonomous vehicles do not follow the "perfect" trajectories of their simulated analogues. Although this was expected given the unavoidable error in digital compasses, GPS receivers, motor controllers, and navigation algorithms, we did not expect the skew between real-world mobility and simulated mobility to be as large as it was—the difference between expected node placement and real-world node placement affected communication opportunities even during relatively short tests. In fact it is due to the magnitude of this difference that we chose to implement our *TraceMobility* module for OMNeT++, which allows us to replay exact GPS traces taken by the robots in simulation to better model real-world movements.

In conclusion, we developed a pair of novel frameworks for evaluating network performance simultaneously in simulation and the real-world, using the OMNeT++ simulator and the autonomous vehicles in the Pharos testbed. We provide an empirical study categorizing the similarities and differences of real-world connectivity versus simulated radio ranges and show several other early results obtained using our vehicular testbed. Additionally, we categorize our testbed's ability to repeat the same mobility trace providing "push-button" repeatability of network experiments. Our frameworks are generic and reusable, and can be deployed on any Linux-based vehicular or robotics platform.

7. REFERENCES

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

[2] M. Anderson, L. Thaete, and N. Wiegand. Player/Stage: A unifying paradigm to improve robotics education delivery. In *Proc. of the Wkshp. on Res. in Robots for Education*, 2007.

[3] R. Baumann, F. Legendre, and P. Sommer. Generic mobility simulation framework (GMSF). In *Proc. of MobilityModels*, pages 49–56, 2008.

[4] 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.

[5] B. Burns, O. Brock, and B. Levine. Mora routing and capacity building in disruption-tolerant networks. *Ad Hoc Networks*, 6(4):600–620, 2008.

[6] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Comm. and Mobile Comp.*, 2(5):483–502, 2002.

[7] J. Davis, A. Animashaun, E. Schoenherr, and K. McDowell. Evaluation of semi-autonomous convoy driving. *J. Field Robot.*, 25(11-12):880–897, Nov. 2008.

[8] 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.

[9] C. de Waal and M. Gerharz. BonnMotion: A mobility scenario generation and analysis tool. `http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewall/BonnMotion`, 2003.

[10] K. Dresner and P. Stone. A multiagent approach to autonomous intersection management. *J. of Artificial Intel. Research*, 31:591–656, March 2008.

[11] `http://www.gpsvisualizer.com/`.

[12] K. Harras, K. Almeroth, and E. Belding-Royer. Delay tolerant mobile networks dtmns: Controlled flooding in sparse mobile networks. In *Proc. of NETWORKING*, pages 1180–1192, May 2005.

[13] J. How, B. Bethke, A. Frank, D. Dale, and J. Vian. Real-time indoor autonomous vehicle test environment. *Control Systems, IEEE*, 28(2):51 – 64, April 2008.

[14] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. In *Proc.of MobiCom*, pages 195–206, 1999.

[15] 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.

[16] A. Khelil, P. J. MarrÃşn, R. Dietrich, and K. Rothermel. Evaluation of partitionaware manet protocols and applications with ns-2. In *Proc. of SPECTS*, 2005.

[17] A. Khelil, P. J. MarrÃşn, and K. Rothermel. Contact-based mobility metric for delay-tolerant ad-hoc networking. In *Proc. of MASCOTS*, pages 435–444, 2005.

[18] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek. The Click modular router. *ACM Trans. on Comp. Sys.*, 18(3):263–297, 2000.

[19] J. Kolodko and L. Vlacic. Cooperative autonomous driving at the intelligent control systems laboratory. *Intelligent Systems, IEEE*, 18(4):8 – 11, jul-aug 2003.

[20] 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.

[21] 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.

[22] P. Pathirana, N. Bulusu, A. Savkin, and S. Jha. Node localization using mobile robots in delay-tolerant sensor networks. *IEEE Trans. on Mobile Comp.*, 4(3):285–296, May–June 2005.

[23] A. Petz, J. Enderle, and C. Julien. A framework for evaluating dtn mobility models. In *Proc. of SCENES*, March 2009.

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

[25] A. Varga et al. The omnet++ discrete event simulation system. In *Proc. of ESM*, 2001.