Testbed Design and Localization in MiNT-2: A Miniaturized Robotic Platform for Wireless Protocol Development and Emulation

Christopher Mitchell[†], Vikram P. Munishwar, Shailendra Singh, Xiaoshuang Wang,
Kartik Gopalan, Nael B. Abu-Ghazaleh
Computer Science, Binghamton University (State University of New York)
[†]S*ProCom[2], The Cooper Union for the Advancement of Science and Art
Contact: {kartik,nael}@cs.binghamton.edu

***Abstract*—A significant challenge in the development of robust wireless networking protocols is often the need to prototype and test these protocols in a small-scale setting before they can be widely deployed. Two contrasting prototyping and testing methods are currently used, requiring a choice between convenience and accuracy. The first involves simulating a wireless network solely in software. Although convenient, this option fails to accurately account for real-world factors such as realistic radio propagation models and their interaction with node mobility and obstacles. The second relies on setting up a large-scale physical testbed that, although accurate, represents a single design point and tends to be expensive to reconfigure and manage. The MiNT project at Stony Brook University was one of the first to propose an accurate and inexpensive small-scale physical testbed built using commercially-available robots coupled with a version of NS2 built to work cooperatively on multiple nodes. MiNT combines the best features of the two popular performance evaluation methods, achieving network accuracy comparable to that of large-scale physical testbeds without abandoning the convenience and flexibility of software simulation. In this paper, we describe our initial experiences in developing *MiNT-2*, the next generation of MiNT. *MiNT-2* represents a fresh redesign of MiNT that at once simplifies and improves the original design, and extends it with a range of new features. The paper describes a number of these improvements including a new, simplified, node design, an improved node localization using RFIDs, node position calibration, and automated layout configuration. We also demonstrate the accuracy of the new localization approach and outline planned testbed improvements.**

I. Introduction

A major challenge currently facing wireless networking researchers and developers is to prototype and test their wireless protocols in a small scale setting before deployment in real-world conditions. The two predominant options for research and development in multi-hop wireless networks are *software-only simulation* and experimentation using *large-scale testbeds*. Network simulators [1]–[3] are particularly popular due to the convenience of evaluating a variety of networking configurations in a controlled and fully observable environment without setting up a physical testbed. However, network simulator are also known to provide insufficient fidelity [4]–[6] in capturing radio propagation effects such as non-uniform path loss, multi-radio interference, and multi-path fading. In fact, most published papers simply rely on these tools without double-checking their validity, presumably due to lack of alternatives.

Recognizing these limitations, an increasing number of wireless researchers choose to validate their protocols by performing tests on large-scale custom-built wireless network testbeds [7]–[9]. Although empirical results collected from these testbeds are certainly more credible, initial experiences with them reveal several fundamental limitations. These include a lack of sufficient flexibility in reconfiguring the initial topology, mobility patterns, radio settings, significant manual maintenance effort required to program and recharge nodes, lack of autonomous 24x7 operation, and most importantly, large physical space requirements due to relatively long-range radio coverage. These issues make testbeds limited in terms of their ability to provide a range of experimental scenarios for evaluation, and extremely expensive from the standpoints of experiment setup and routine maintenance.

The *MiNT* [10], [11] project at the Stony Brook University was one of the first multi-hop wireless testbeds that "miniaturized" the physical space requirements of the testbed. Two key architectural features in the *MiNT* testbed were (1) the use of mobile robots to transport wireless network nodes, and (2) the use of radio signal attenuators to shrink the physical space requirements of a multi-hop wireless testbed. Each *MiNT* node was built from a mobile robot and an embedded computer equipped with multiple IEEE 802.11a/b/g WLAN interfaces. The robots used in the *MiNT* testbed are inexpensive Roomba [12] robots from the iRobot Corporation, which can be programmatically controlled via a well-defined serial API. Node mobility via robots permits protocol developers to automate the process of reconfiguring the network topology and to specify the node mobility pattern during an experiment. This drastically decreases the setup, fine-tuning, and management efforts required to tailor a testbed to the needs of a protocol study. *MiNT* performs node position localization through computer vision-based robot positioning and navigation techniques. A self-charging mechanism allows each robot to detect a nearly-depleted battery and find a docking station to recharge.

A. MiNT-2: A Fresh Re-Design

Our group at Binghamton University has recently been developing *MiNT-2* – the second generation of an improved miniaturized robotic platform for wireless protocol development and testing. The goal of the *MiNT-2* project is two-fold. First is to reproduce the *MiNT* testbed's functionality independently to carry out wireless networking research within our group. Second is to improve the original testbed design and operation along the way through use of more effective algorithms and technologies. In this paper, we document our implementation experiences in improving several aspects of the original *MiNT* testbed, but in particular testbed design and node localization techniques. Nodes in *MiNT-2* are designed for low cost and maximal functionality using the latest generation of iRobot Create robots. The Create offers a mobility platform with well-documented x86-compatible libraries and a bidirectional serial protocol for movement control and feedback, battery charge level, proximity/cliff sensor data, etc. A low-power x86-based embedded controller board interfaces with the robot and an RFID reader, provides multiple wireless interfaces necessary for multi-channel protocols, and runs a distributed "hybrid" version of the NS2 software package. Compared to the original *MiNT* testbed that used camera-based localization, our *MiNT-2* prototype uses a simpler and more effective three-stage system of motor commands, distance sensor feedback, and RFID tags on the floor of the testbed that provide authoritative position information. Power management is also improved by directly running each node's on-board electronics from its robot's battery, which is automatically charged by the auto-docking procedure triggered from a low battery condition detected via the robot's API. Although cost reduction was not our primary objective, we did extensively use commodity hardware to reduce per-node testbed costs to around $1000, which is significantly cheaper than the nodes in the original *MiNT* testbed.

B. Localization

The goal of localization is to allow a node to accurately determine its own absolute position and orientation as well as that of other robots in the testbed. The Roomba robots used in the original *MiNT* prototype could not sense their own positions. As a result, they relied exclusively on the central controller which used a vision-based position/orientation tracking system consisting six ceiling-mounted webcams with overlapping image planes. This system was used to track node location and to command the robots on which direction to move at any instant. To account for growing discrepancy over time between the central controller's per-node position information and the actual node positions, mobile robots were manually brought to fixed locations to re-synchronize their logical and physical node positions, periodically. Each node was identified using unique color patches mounted on the mobile nodes. Since this system relied on visual identification, it tended to develop inaccuracies over time if any of the six cameras moved slightly, or color patterns on the nodes faded, or lighting varied.

We redesigned the localization mechanism from scratch in the *MiNT-2* testbed to overcome the limitations of its predecessor. Our new localization mechanism uses inexpensive RFID technology coupled with enhanced mobility sensors within the Create robot to reduce maintenance overheads, which helps achieve high levels of accuracy. Since the RFID tags are distributed within the testbed area, a node can localize itself whenever it crosses an RFID tag, by assigning the position of the tag to itself.

In the remainder of the paper, we first give an overview of the current and planned features of *MiNT-2*, present the details of improved node design, and localization, and finally present a preliminary performance evaluation of our current testbed.

II. Overview of *MiNT-2* Features

We now provide an overview of the key features of the *MiNT-2* testbed as well as ongoing developments. This will be followed by a detailed description of two aspects of the system design we explore in this paper, namely: *MiNT-2* node design and node localization.

[Miniaturization] *MiNT-2* significantly reduces the physical space requirement of a multi-hop wireless network testbed, just as in the original *MiNT* testbed does, by attenuating the radio signals in a controlled fashion. This space reduction drastically decreases the setup, fine-tuning, and management efforts required to tailor a testbed to the needs of a protocol study. A *MiNT-2* node is an embedded computer mounted on a Create robot [13] – a low-cost (less than $250) programmable robot from iRobot. Create robot supports a basic set of externally controllable movements, is able to carry a large payload (up to 30 pounds), and comes with an effective auto-recharging capability. Mounted on each Create is a Soekris net5501 embedded board, which is a low-power small-form-factor computer with a 433MHz processor and runs on power from Create's internal battery. Net5501 has a PCI extension board (RB-52), which allows us to put 4 Atheros-based 802.11 a/b/g mini-PCI cards to support multi-radio experiments [14]. A radio signal attenuator [15] is inserted between each wireless LAN interface and its antenna to reduce the transmitted and received signal strength and thus the physical space requirement.

[Autonomic reconfigurability and management] The use of programmable Create robots in the *MiNT-2* testbed allows a user to configure an arbitrary initial network topology and to set up an arbitrary node mobility pattern during a simulation run. *MiNT-2* automates the support for such flexibility by employing physical-level radio signal measurements and RFID-based robot positioning and navigation techniques. A key requirement for the *MiNT-2* infrastructure is to be an autonomic testbed that is remotely accessible for 24x7 operation without human intervention. Usually, battery charging is a manual process requiring the administrator to take discharged nodes to charging sockets [16]. *MiNT-2* features a self-charging mechanism that allows Create robots to recharge their battery when they run low, thus completely eliminates manual charging efforts. This mechanism, together with automated

robot positioning and navigation, allows *MiNT-2* to sustain 24x7 operation without human intervention.

[Support for protocol development, testing, and debugging] *MiNT-2* is aimed primarily as a platform for wireless protocol implementation, testing, and debugging. We are currently enhancing *MiNT-2* to support the following capabilities: (1) a network fault injection and result analysis tool that will allow protocol developers to declaratively specify high-level network fault patterns (e.g., three consecutive packet drops after receiving the second ACK) and the desired responses to these faults, inject the specified faults, and automatically analyze the responses of the protocol under study; (2) a distributed debugger that will allow a protocol developer to pause/resume, single-step, breakpoint, and roll-back a simulation run, and (3) a visualization interface that will provide real-time view of the testbed configuration, traffic load, node/link liveliness, and evolution of protocol-specific states, and allow users to steer a simulation run by modifying protocol/network configuration and input load parameters on-the-fly. We are also developing an application programming interface (API) for users to implement and test wireless protocols and applications directly on *MiNT-2* without detouring through NS2. This API will incorporate two facilities for wireless protocol/application development: access to low-level details of wireless LAN interface and user-level TCP/IP stack. To this end, we are developing a procedural interface to the Linux Wireless Extension (WE) [17] package that exposes network interface configuration and run-time statistics common to wireless cards.

[Running existing simulation code on *MiNT-2*] Many existing wireless protocols are written as NS2 simulation models. Just as with original *MiNT*, *MiNT-2* provides the ability to directly execute existing NS2 scripts and models on the testbed. *MiNT-2* includes a distributed execution version of the NS2 engine where (1) the simulations of physical and MAC layer protocols are replaced by direct execution on the WLAN interface hardware, and (2) higher-layer protocol simulation of each simulated node runs on a separate *MiNT-2* testbed node. To run NS2 on a *MiNT-2* testbed, each link-layer frame is wrapped and sent to its destination as a UDP packet. On the receiving node, the packet is extracted from the UDP payload and a new packet receive event is generated and scheduled. This way, the physical and MAC layers of a simulation run are based on direct execution rather than software simulation. Compared with software-only NS2, *MiNT-2* thus provides the same user interface to start and monitor NS2 simulation runs, but produces *higher-quality* simulation results *faster* because it does not simulate MAC and physical layers in software.

[Control server] A central control server presents a single front-end interface for the end user to interact with the *MiNT-2* prototype. For example, to run an NS2 simulation on *MiNT-2*, the user simply starts the NS2 script on the central controller, which then takes care of parsing, rewriting, and propagating the script, synchronizes the system clocks, and starts the distributed simulation run. In addition, it also coordinates the operation and physical movement of the wireless nodes by sending commands and receiving responses using a special control protocol over a a dedicated 802.11g channel. All control traffic is transported on an IEEE 802.11g channel, which does not interfere with IEEE 802.11a channels, which are used in actual experiments. The central controller receives inputs from robots over the control channel about their current position/orientation and sends movement commands to mobile nodes as per execution requirements.

iRobot Create with battery and charger	$250
Soekris net5501 x86 embedded board	$260
PCI to 4x miniPCI adapter card	$65
4x R52 802.11a/b/g cards	$200
4x antennae	$60
3x attenuators	$100
ID-12 RFID reader	$30
Small miscellaneous components	$35
Total	$1000

TABLE I
PER-NODE COST IN *MiNT-2*-2

[Limitations of *MiNT-2*] Although *MiNT-2* represents a significant advance in the emulation technology for mobile wireless networking research, it has its share of limitations. First, *MiNT-2* does not physically miniaturize the real-world around the testbed. Rather it attenuates radio signals. Therefore, it cannot reproduce exactly the same signal propagation characteristics as in a real-world wireless network due to the presence of physical objects, such as walls, doors, tables, and humans. To be sure, this appears to be a fundamental limitation of *any* reconfigurable wireless testbed, because it is difficult to duplicate all the factors that affect the radio signal propagation characteristics of a given real-world wireless network. Second, radio signal attenuators only attenuate the signals transmitted from sources under *MiNT-2*'s control, but not from other radio sources such as nearby microwave ovens or production-mode WLAN interfaces. This means that when a *MiNT-2* testbed is deployed in an environment where there are other radio sources, its signal to noise characteristics may not be the same as those of the non-attenuated version it emulates. For example, experiences with earlier *MiNT* prototype indicate that the ratio between sensing range and hearing range between a pair of wireless nodes is increased as a result of miniaturization. This problem is less serious if the physical location hosting the *MiNT* testbed is relatively free from external radio sources.

III. NODE DESIGN

The design of our *MiNT-2* prototype node has been improved over several iterations before we finalized a setup to minimize per-node complexity, permit all necessary MiNT functionality, and provide expandability for future uses of the testbed. In this section, we describe the details of the *MiNT-2* node and testbed design which is shown in Figures 1, 2 and 3. Table I shows the approximate cost breakup for a *MiNT-2* node with total per-node cost around $1000, substantially less than the per-node cost of a functionally comparable node in the original *MiNT* testbed.

[Mobility Using The iRobot Create] We used the iRobot Create platform to achieve node mobility, since it offers the

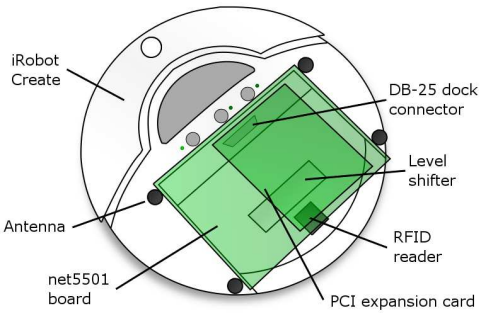

Fig. 1. Physical layout of a MiNT-2 node.

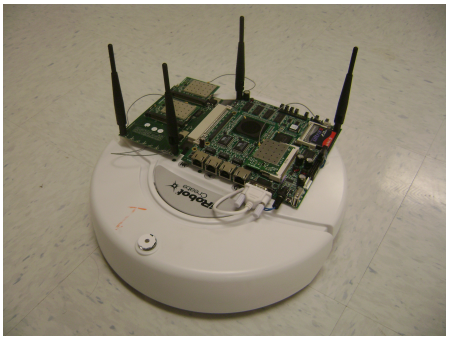

Fig. 2. Picture of a MiNT-2 node.

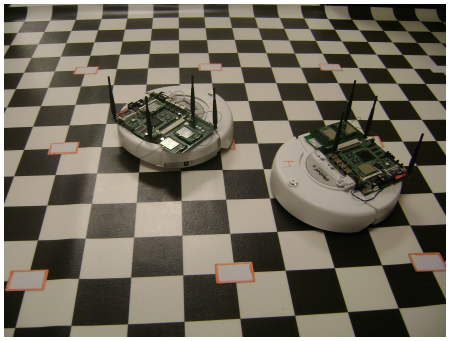

Fig. 3. Picture of the MiNT-2 testbed. RFID cards are marked red.

full range of motion we require – forward, backwards, in an arc, or in-place rotation – and includes an easily-interfaced serial control protocol. The Create is a developer-friendly robot, which differs from the commercial Roomba robotic vacuuming robot used in the prior *MiNT* testbed in that it provides a much more complete interface for controlling the robot. One of our goals was to eliminate the external battery necessary in previous wireless testbeds to power the embedded controller board, so we looked only at boards capable of running on the 15V, 1.5A unregulated voltage supplied by the robot. The embedded board used direct serial control to issue commands to and read sensor data from the robot. The robot has full freedom of 2-dimensional movement. Each Create can move forward and backwards at velocities between 1 and 500 mm/s in increments of 1mm/s, turn in place, and move in forward and backward arcs from 1 to 2000mm in radius in steps of 1mm. The robot, besides functioning as the mobility component of each node, provides three classes of connections from a DB-25 connector on the inside of its cargo bay: a 57,600-baud bidirectional control interface, an unregulated 15 volt (1.5 amps) power for the embedded board, and 5 volts (100 milliamps) to power an external RFID reader chip. We describe these in detail later.

[Embedded Control Board] From the beginning of the design process, we assumed several minimum requirements for the embedded board to be mounted on each node. First and foremost, it needed to have sufficient computing power to run full-fledged unmodified networking applications and simulation packages, such as NS2, and needed to support a modern distribution of Linux. Secondly, it needed an RS232 serial port to interface with the robot, and later, a second serial port to receive data from the RFID reader in each node. It also needed to consume relatively little power, and accept a variable-voltage power supply. Thus, a board with a built-in voltage regulator was preferable. In addition, we wanted to eliminate the hard drive used in the original *MiNT* testbed and rely solely on flash-based memory such as Compact Flash (CF) or Secure Digital (SD) cards [18]. Finally, to perform multiple-channel simulations and provide future expandability, we wanted at least four wireless interfaces per node.

The Soekris net5501 board met our specifications, provided additional features, and supported a functional Linux operating system. The net5501 is built around an x86 compatible AMD Geode processor running at 433MHz with 256MB of RAM, a Compact Flash slot, a single miniPCI slot, and a PCI riser slot. By adding a PCI expansion card, we were able to connect four additional miniPCI wireless cards for a total of five wireless interfaces. The Voyage Linux distribution worked out of the box, and with some basic setup, the Madwifi drivers built into Voyage for Atheros wireless cards recognized and initialized all five wireless interfaces. The Soekris board can accept 5 to 25V DC, and draws well under the 27-watt maximum continuous power that the iRobot Create can provide.

[Automatic Node Recharging] Automated recharging adds to the robustness of the testbed, as it allows the testbed to be used remotely for extended periods of time. It minimizes the need for local administration, specifically, manually bringing each node to a recharging station or plugging a power supply into each robot. Create robots are designed to use a docking station to self-charge their batteries. The docking station emits IR beacons that enable a Create robot with depleted batteries to home in to the station automatically and recharge. *MiNT-2* nodes use the robot's built-in docking algorithm and check the battery charge level using the Create's API to determine when battery failure is imminent. In our design, the net5501 embedded board is powered using the Create's own battery, as opposed to the earlier *MiNT* prototype which required a separate universal laptop battery to power the onboard computer. We reverse engineered the level-shifting circuitry of Create so that the net5501 board can be powered using Create's internal power source. Unlike the Roomba robot used in earlier *MiNT* prototype, the Create robot in *MiNT-2* provides an interface to probe the residual charge in its battery. We use a residual battery capacity estimation algorithm to track the battery status and to determine when to recharge which nodes. Section V provides details on both the charge estimation algorithm and battery performance.

[Wireless Setup] Each node needs at least two communications interfaces: one for node control and one or more for data exchange. The control interface is used to communicate between the controller node and each individual node, while the data interfaces are used to communicate with other nodes

during experimentation. Bluetooth, 802.11a, and 802.11b/g were considered for the control and data interfaces. We determined that 802.11a would be the best choice for data channel due to higher bandwidth and less interference with the 2.4 GHz spectrum, leaving 802.11b/g free for control packets communication. The R52 is a low-power miniPCI wireless card supporting 802.11a/b/g based on the Atheros AR5414 chipset. It requires Madwifi drivers, that are included by default in Voyage Linux. While the four wireless interfaces in each node are intended primarily to experiment with multiple channel protocols and MIMO capabilities, other possible uses include using each physical node as up to four virtual nodes. Virtualizing nodes would permit four times as many wireless clients to be simulated in the testbed, with the caveat that each of the virtual clients contained in a physical node would be in the same physical location at all times.

[Miniaturization of Radio Channel] The maximum transmit power of each R52 card used in our testbed is 65mW. Two methods are paired to reduce this power and thus limit the range of the card: electrically attenuating the antenna connection, and reducing the transmission power in software. The iwconfig utility built into Voyage Linux allows a reduction in transmission power down to 1mW, but as observed in the earlier *MiNT* prototype [19], this does not sufficiently limit each node's range. The physical space requirement of the *MiNT-2* testbed is further reduced by inserting a radio signal attenuator [15] between the WLAN interface and its antenna on each mobile node. By combining hardware and software attenuation control, we are able to adjust the range of each wireless interface at a fine granularity. Each mini-PCI WLAN card on the net5501 board is connected to a 2 dBi external antenna through a 20 dB attenuator. In addition to the fixed attenuation, the transmit power on the mini-PCI cards can be altered by another 17 dBm to provide additional flexibility in tuning inter-node signal to noise ratio. Since a radio signal attenuator reduces the strength of the input signal and relays the result to its output completely in the analog domain, it does not incur any observable performance overhead.

IV. Node Localization

Our localization infrastructure has three components. First, the robot's wheel motors can be controlled in terms of distance per second, and each wheel's speed can be varied in integer steps between -500mm/s and 500mm/s. This system by itself may accumulate error over time from mechanical variation, and also fails to take into account mechanical acceleration and deceleration after start and stop commands are issued. To detect and correct for these inaccuracies, at the second level, each of the robot's two driving wheels includes an axle-mounted rotation sensor that measures the actual distance each wheel has traveled. The sensors can be queried via the serial interface, and return both distance traveled by the robot since the last query (in mm) and angle the robot has rotated through since the last query (in degrees). Since the data read from wheel-rotation sensors is used to estimate both change in orientation as well as the change in distance, any inaccuracies present in angle measurement can cause errors in distance estimates as well. Over time, even the accumulated sensor measurements may grow inaccurate due to rounding errors, wheel slippage, and encoder inaccuracy. In addition, a node may be manually picked up and moved to a new location. Thus, a third absolute positioning system is needed to periodically re-calibrate the node's position and orientation in the testbed space. We considered a wide assortment of localization technologies before settling on RFID as the most cost-effective and sufficiently accurate method.

[RFID-based Localization Algorithm] An array of fixed RFID tags on the floor of the testbed allow each robot to determine its absolute location with an uncertainty equal to the maximum tag sensing radius of the RFID reader ($2.25cm$). We mounted ID-12 RFID reader at the base of each node's cargo bay, soldered it to a breakout board, and wired it via a DB-10 connector to the internal serial connection of the net5501 embedded board. The RFID reader, which is used to determine the node's absolute (x, y) position within the testbed by matching the IDs of the RFID tags with a static translation table of ID to coordinates, corrects for errors in the sensors and compensates for extraneous interactions such as hitting an obstacle or researchers lifting and manually relocating the nodes. The localization system is currently implemented as a set of functions detailed below, with preliminary accuracy measurements presented later.

The heart of the localization algorithm is a section of code running periodically every 50ms that is in charge of acquisition and processing of data from movement sensors and the RFID reader. Figure 4 presents a more structured view of the localization algorithm. Every time the localization tick runs, it reads the delta change in distance and orientation since the last sensor access, adds the changes to the last known position and orientation respectively, and clears the sensors. In addition, more precise position and orientation calibration is performed when passing over RFID tags to remove accumulated error from the Create's sensors feedback. RFID data is considered more authoritative than sensor data, and has priority whenever information from the sensor and the RFID sources disagree. Once at least two tags have been read, the node can determine its orientation from the coordinates of each tag, its orientation at the first tag, and the amount of node rotation between the first and second tags.

A node may move from one tag to the next in a variety of ways. If the node moved in a straight line (no change in orientation over time with forward motion) or arc of constant radius (constant rotation over time with forward motion), the RFID data is used to calculate the node's heading at the second tag. The orientation at the second tag can be calculated using the current (x_2, y_2) position read from the current tag, the position (x_1, y_1) at the previous tag, and the node's rotation (change in orientation) between the two tags as measured by its sensors, $\Delta\theta_{RFID}$:

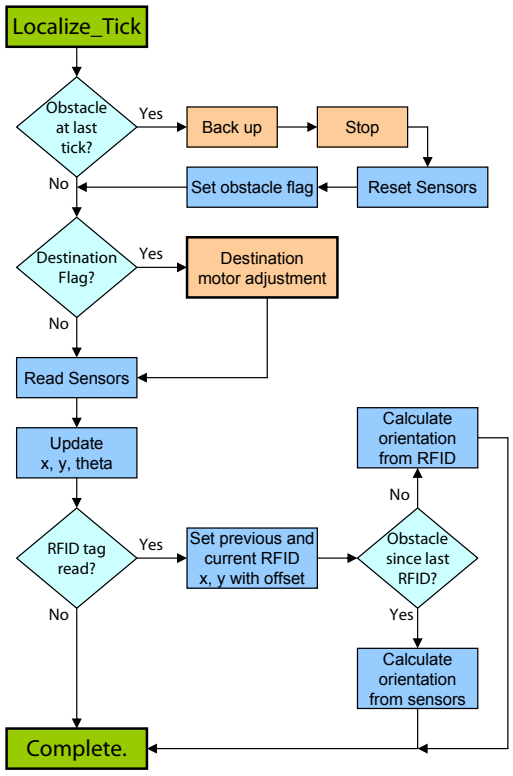

Fig. 4. Scheduled localization tick algorithm.

$$\theta = (tan^{-1}[(y_2 - y_1)/(x_2 - x_1)] + \Delta\theta_{RFID}/2) \mod 360 \quad (1)$$

If the node travels in a straight line, then $\Delta\theta_{RFID}$ is 0 and the equation reduces to the trivial case. If it traveled in a constant-radius arc, its deviation from the straight line path between the two tags is equal in magnitude at both tags, but opposite in sign. In other words, since the node traveled in an arc of constant radius, it would have been oriented as far to one side of the straight line path between the tags at the first tag as it was to the other side of the path at the second tag. Thus its rotation between the two tags is twice the individual magnitude of deviation from a straight- line path at each tag, and current orientation can be determined solely from the $\Delta\theta_{RFID}$ and two pairs of position coordinates while discarding all orientation and position data up to the first of the two tags. Figure 5 elaborates more the robot's orientation calculation for the constant-radius arc movement. As only the change in rotation between the two tags is retained, accumulated inaccuracy from the on-board rotation sensors is discarded each time this calculation is performed.

If instead the robot travels from one tag to the next in an arc of non constant radius or encounters any obstacles between the two sequential tags, the RFID data is only used to determine position, and node orientation is calculated using the incremental change in rotational sensor reading since the last localization tick. Also, the structure of a MiNT

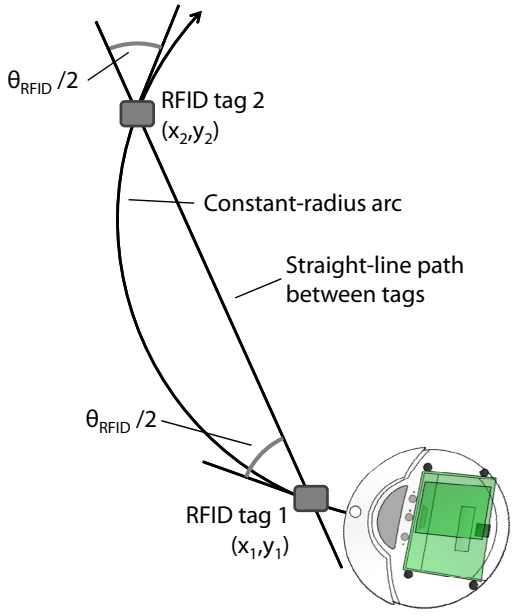

Fig. 5. Orientation calculation for constant-radius arc movement.

node itself slightly complicates localization because the RFID reader cannot be placed exactly at the center of the node. The internal sensors return data most accurate at the point halfway between the robot's two axles, but the RFID reader is located 147mm behind this point. To compensate for this offset, the sensor-derived position information is combined with the (x, y) coordinate of the RFID tag to calculate the actual position of the center of the robot. Future modifications to the node design are accommodated by maintaining two constants, $RFID_{OnAxis}$ and $RFID_{OffAxis}$, containing the distance in mm from the center of the robot to the center of the RFID reader parallel and perpendicular to the direction of forward motion, respectively.

[Initial Node Calibration] Our goal in *MiNT-2* is to *completely automate* the steps of calibration and initial positioning and of the nodes. Accurate calibration means that the testbed operator can drop each robot anywhere in the testbed and the robot automatically positions itself at a nearest reference point with known orientation. Following calibration, the node then automatically moves to an initial position/orientation specified in the experimental scenario. The calibration step of each node's position and orientation is performed by traversing the testbed in a series of straight lines. If a node reaches an obstacle, such as a wall or another robot, it rotates and attempts another straight line movement. Calibration is only considered successful if the node reads two RFID tags over a straight line movement. Correct calibration of position is further complicated by the off-center RFID reader, but the offset is handled by the localization tick as detailed in the previous section. Orientation can be calculated from two collinear RFID tags regardless of the node's offset, but accurate position determination taking the RFID reader offset into account requires the authoritative heading of the node

to be predetermined. Thus, after passing the second tag, the node's orientation is calculated from the tabulated positions of the two RFID tags as given in equation 1, while its position is determined from the tabulated position of the second RFID tag corrected with an offset calculated from the now-initialized orientation. Once initialization of both position and orientation is complete, the node stops and is ready to receive an initial position for a simulation scenario. Given that the first RFID tag crossed is at (x_1, y_1) and the second is at (x_2, y_2), the position and orientation at the second tag are:

$$\theta = \arctan[(y_2 - y_1)/(x_2 - x_1)] \quad (2)$$

$$x = x_2 + RFID_{OnAxis}\cos(\theta) + RFID_{OffAxis}\sin(\theta) \quad (3)$$

$$y = y_2 + RFID_{OnAxis}\sin(\theta) + RFID_{OffAxis}\cos(\theta) \quad (4)$$

For our nodes' construction, $RFID_{OnAxis}$, was 147mm and $RFID_{OffAxis}$ was 0mm.

[Network Layout Initialization] Before a scenario execution, the user can specify the topology as well as the node locations of the target network. After calibration, each node then travels to the user-specified initial position and orients itself in a specific direction before an experimental scenario begins. User can also modify the initial location of a node and have the controller instruct the corresponding robot to move accordingly. During this process, every testbed node is constantly measuring the radio signal strength between itself and each of its neighbors within hearing range, and relaying the information to the central controller, which then feeds it back to the user. Such interactive initial placement and feedback greatly simplifies the network setup effort because the administrator no longer needs to manually move the robots.

V. Performance

A. Battery Performance

Figure 6 shows the results of battery tests and reveals that node longevity matches the worst-case measurements for other testbeds. Note that only data for the standard battery has been presented because node life did not vary significantly between the standard and extended batteries from iRobot, presumably due to the presence of embedded control board. Nodes are able to move continuously while running the Soekris board for 2.25 hours before the battery discharges completely. Tests performed with static nodes transmitting constantly achieve 3.6 hours of battery life. Board clock speed and node movement speed remain constant to the limits of the applicable sensors regardless of current battery level. The distinctive shape of the discharge curve provides a strong indication of impending battery failure. As expected, motor operation claims the largest impact on longevity, while a single active wireless interface only removes twenty minutes of battery life. Multiple simultaneously-active interfaces are expected to incur a linear toll on the battery.

B. Localization

As discussed earlier, our localization mechanism uses a combination of readings obtained from distance and angle sensors mounted on the robot and pre-deployed RFID tags. Specifically, we use RFID tags to recalibrate current position and orientation of the robot. However, since the RFID tag can be detected by the RFID reader, immediately when it comes in the range with the tag, (which is a circle of 2.25 cm radius), the RFID tags based error calibration has a maximum error bound of 2.25 cm. In this section, we first describe our experiments to measure accuracy of distance and angle sensors, and then present the results representing error estimates for the localization algorithm.

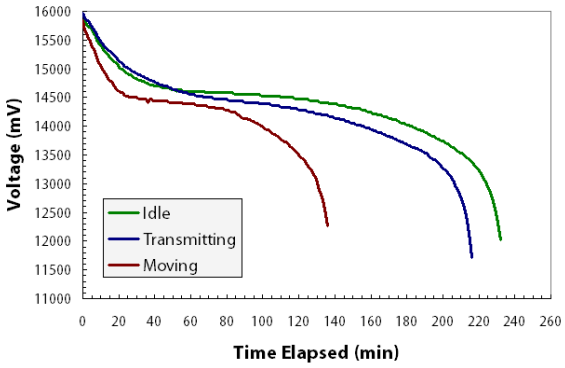

Fig. 6. Standard battery life on an idle node, a continuously-transmitting node, and a continuously-moving node.

1) Sensor Accuracy: In order to calculate error bound for the distance sensor, we created multiple scenarios of RFID tags by keeping them in straight line and varying distance between a pair of tags for different scenarios. We programmed the node to move in a straight line such that it will encounter the RFID tags placed along the way. The distance sensor value was calibrated at each encountered RFID tag and the error was noted, where the error is the difference between the (x,y) position represented by the distance sensor and that of the detected RFID tag. In Figure 7, values along x-axis represent the distances between a pair of RFID tags for different scenarios, whereas y-axis represents the error in the distance sensor readings.

We can observe that the error in position estimates is less than 2 cm. However, for all the scenarios used in our setup, we did not observe an explicit trend in the estimates of errors with respect to the increase in distance between two tags. The difference in the error estimates for different scenarios is due to the mismatch between the timings of reading an RFID tag and the distance sensor (which is read every 50 milliseconds by the localization algorithm).

To calculate error bound for the angle sensor, we kept RFID tags in a pyramid like structure and moved the robot from the tip to the base of the pyramid in a straight line. The height of the pyramid was the length of our testbed area, and closely-spaced sets of RFID tags were kept in a gap of fixed distance along the height of the pyramid. Thus, based on the RFID tags that the robot has encountered along its straight line path, we calculated the difference between the angle obtained using RFID tags information and the angle obtained from the angle-

sensor. Figure 8 shows the change in the robot's angle when it is moved in a straight line path, across multiple trials. For this experiment, the robot's movement was kept perpendicular to the x-axis of the coordinate space of our testbed. The figure shows that when the robot is moved in a straight line, it sometimes changes its angle by 0.5 degrees, as it can be observed for trial numbers 3 and 5.

2) Position Accuracy: For this experiment, we arranged the RFID tags in a uniform grid and moved the robot in a random fashion over the deployed area of RFID tags. We kept track of three readings for calculating (x, y) coordinates of the robot when it encountered an RFID tag: (1) *sensors-based readings:* obtained from the angle and distance sensors, (2) *estimated readings:* readings obtained by combining information from RFID-tags with the sensors readings, and (3) *RFID-based readings:* actual coordinates of the RFID tags.

We conducted three experiments to estimate position accuracy of the robot. The error in sensor-based readings and RFID+sensors based (estimated) readings is obtained by calculating their Eucledian distance with the actual RFID-tag position. Figure 10 presents a graph comparing errors for sensor-based and RFID+sensors based readings. It can be seen that the error for RFID+sensors based approach is much lower than just the sensors based readings. Additionally, the sensors based readings show an increasing trend as the duration of the robot's movement increases. Similarly for Figure 11, and Figure 12, we changed the interval between successive position updates, and velocity of the robot respectively. It can be noted that the average localization error for just sensors-based readings is higher than the RFID+sensors based readings in both the graphs. Overall, for larger value of position-update-interval (e.g. 100 milliseconds), and larger value of velocity (e.g. 400 mm/s), localization error for both the approaches is on a higher side, and thus such values of these parameters should be generally avoided.

C. Wireless Range and Attenuation

When operating without attenuation and with only the control interface active, range was found to exceed 200 meters of open space. Figure 9 shows variations in the RSSI values observed at the receiver with respect to the distance between the sender node and the receiver node, with and without attenuation. In our experiment, we used a fixed signal attenuator of 20 dB at the sender, and moved the receiver with the granularity of 1 ft. The transmission power at the sender was kept as 1 dBm. It can be noticed that the variation in RSSI values follows a similar trend with and without attenuation. In addition to the RSSI values, we also measured the reception throughput at the receiver. For the communication without attenuation, the throughput observed was consistently above 95% within the distance of 1 ft. to 7 ft. However, with 20 dB attenuation, throughput above 80% was observed for the distance up to 2 ft., beyond which the throughput degraded gradually, resulting in 0.8% of throughput at the distance of 7 ft.

VI. Related Work

[Network Simulators] Network simulators [1]–[3] have been popular among wireless networking researchers, primarily due to the convenience they afford in evaluating wireless protocols while eliminating the need to set up a large physical networking infrastructure. However, in spite of their popularity, it has been shown that most simulation tools do not accurately emulate the majority of real-world radio signal propagation effects such as non-uniform path loss, multi-radio interference, and multi-path fading [5], [6]. Often researchers rely on these simulation tools to generate results without cross-checking the accuracy of the underlying models. Concerns about simulation accuracy are particularly timely when the wireless networking community is gravitating toward research on cross-layer protocol optimizations, such as hop-by-hop error control, MAC-layer anycast, and signal strength-aware routing, where accurate physical-level radio channel models are essential.

[Large Testbeds] Recognizing these limitations, an increasing number of wireless researchers choose to validate their protocols by running their protocol implementations on larger custom-built wireless network testbeds [7]–[9]. Although empirical results collected from these testbeds are certainly more credible, initial experiences with them reveal several fundamental limitations. First, users of these testbeds do not have the flexibility to reconfigure the network connectivity by specifying an initial topology, moving the nodes around, or setting pairwise signal-to-noise ratios. Second, those testbeds that do support node mobility require significant manual maintenance efforts for node charging and positioning, and thus cannot work as an autonomic research infrastructure that supports 24x7 operation. Finally, physical distance between adjacent wireless nodes is critical because it affects the quality of connectivity between them. Most current wireless research targets IEEE 802.11 WLANs or cellular networks, both of which have relatively long-range radio coverage. Therefore testbeds designed to support meaningful protocol studies on these large testbed need to span a large geographical area so that they can contain a reasonable number of wireless nodes which do not fall within the same collision domain.

[Small-Scale Testbeds] Among other existing wireless network testbeds, Orbit [20], [21], Emulab [16], [22], and CMU Wireless Emulator [23] are closest to *MiNT-2*. In contrast to *MiNT-2*, which performs radio signal attenuation, Orbit uses explicit noise injection to control the signal-to-noise ratios between selected pairs of nodes. This approach cannot scale to a large number of pairs because it may be impossible to generate an external noise profile that can simultaneously satisfy the signal-to-noise ratio requirements of all active pairs. Emulab did not attempt to solve the problem of setting up a multi-collision-domain network within a small physical space. CMU Wireless Emulator uses an FPGA-based DSP engine to emulate the signal propagation environment between nodes. Compared to simulations, the emulator uses real MAC and PHY layers, and supports real applications. However the

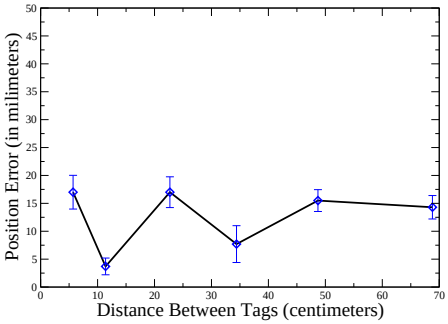

Fig. 7. Error estimate for distance sensor

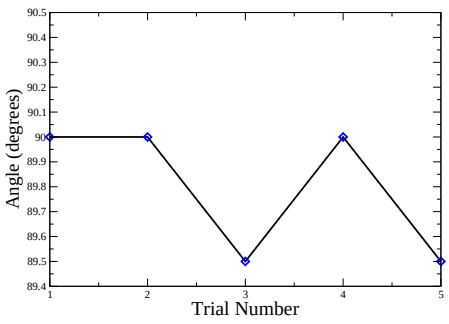

Fig. 8. Error estimate for angle sensor

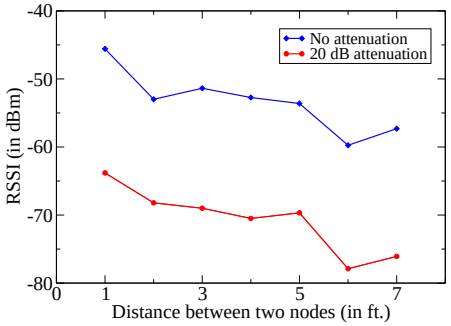

Fig. 9. Distance vs. RSSI

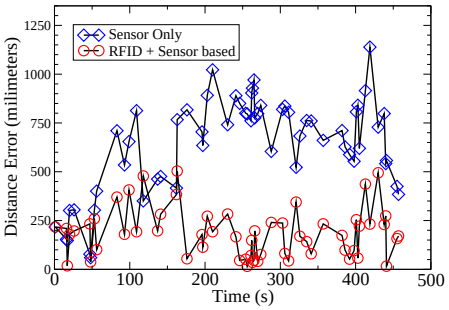

Fig. 10. Errors in sensor-only vs. RFID+sensor-based localization with time.

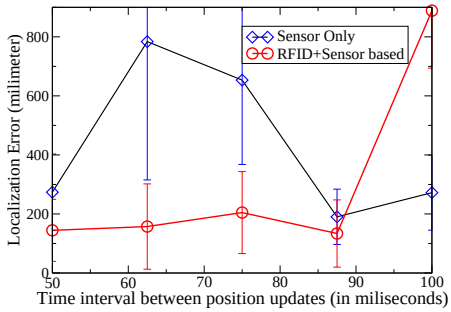

Fig. 11. Errors in sensor-only vs. RFID+sensor-based localization with update frequency.

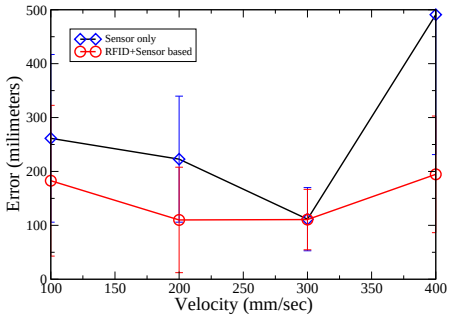

Fig. 12. Errors in sensor-only vs. RFID+sensor-based localization with node velocity.

FPGA-based emulation of RF signal propagation can only be as accurate as the accuracy of the models that it is programmed with. *MiNT-2*, besides using real MAC and PHY layers, preserves the time-varying nature of the RF propagation characteristics observed in real-world. *MiNT-2* also provides true node mobility and autonomic 24x7 testbed operation. On the other hand, Orbit "simulates" node mobility by migrating the simulation process for the mobile node from one physical node to another. Because Orbit's nodes are organized as a fixed grid, the node mobility pattern that Orbit can support is less flexible than *MiNT-2*. Mobile Emulab [16] also uses mobile robots to support node mobility but requires frequent manual recharging.

Furthermore, all the above testbeds lack the mechanism to inject controlled distributed faults into a wireless environment to determine the effect of malfunctions and failures. Moreover, the other testbeds' robotic platforms must be manually taken to their charging stations every 2-3 hours. The iRobot Create comes with a high capacity battery and an automatic charging dock, as well as a serial level-shifter that can be easily modified for our purposes, making the Create a particularly cost-effective solution.

[Localization] There is a large body of work [24] on localization in the context of ubiquitous computing, location-aware services and robotics research. Our focus in the *MiNT-2* testbed has been on developing a simple, inexpensive, and practical localization technique for robots used in wireless protocol development and testing context. We considered active and passive infrared triangulation as well as an infrared version of the Stony Brook vision system that would use patterns of infrared LEDs instead of colors to identify each robot and provide its location and heading. MIT's Cricket ultrasonic localization [25] was also considered. However the cost of these systems combined with the significant associated computation burden rendered it unsuitable for our application. A pressure-based system called SmartFloor [26] would not have been able to differentiate between the nodes, although it is well-suited to determine the precise location of each node. Mobile Emulab [22] also uses a vision based image processing technique for localization, using two color pattern on top of each robot. In addition to sharing the same drawbacks as the original *MiNT* testbed, the two color patterns cannot help identify the robots individually. *MiNT-2* nodes do not run into these problems because an inexpensive RFID reader is mounted on each robot and thus the node itself can detect/compute its location without relying on an external entity for identification. RFID based localization techniques have also proven effective in other contexts for both large-scale localization [27] and more limited table-sized applications [28].

VII. Discussion and Future Work

As described earlier, *MiNT-2* primarily aims at improving accuracy of simulations, and reducing complexities associated with conducting experiments on large scale wireless testbeds. In that sense, *MiNT-2* can be used as a stepping stone for protocol development and evaluation before actual deployment and testing of a protocol in a larger testbed. However, *MiNT-2* has its share of limitations. For instance, the current *MiNT-2* prototype primarily supports 802.11 family of protocols and does not support the cellular communication technology. Another limitation of it is that while *MiNT-2* miniaturizes

the radio propagation, it obviously does not miniaturize the physical space or the node movement patterns.

Improvements to the *MiNT-2* testbed are ongoing. As discussed in the Section II, the original *MiNT*, as well as *MiNT-2* provide the ability to directly execute NS2 simulations on the testbed. We plan to extend *MiNT-2* further to support GloMoSim [2] and the more recent NS3 [29]. Moreover, we plan to improve the accuracy of the RFID-based localization mechanism by experimenting with different RFID tag densities, as well as by fine tuning the robot orientation calculation mechanism.

VIII. Conclusion

In this paper, we described the improved testbed design and localization techniques in *MiNT-2* – a ground-up re-design of the predecessor *MiNT* project from Stony Brook. Our vision is to simplify the architecture for designing a fully functional miniaturized wireless protocol development and testing platform, so that any networking research group in the world can easily set up such a testbed using commodity hardware and open-source software. Each node in *MiNT-2* is comprised of an iRobot Create providing mobility to the node, an ID-12 RFID reader performing localization, a Soekris net5501 x86-compatible embedded board directing the robot, and five wireless network interfaces with attenuators and antennae for control and communication. Node localization system is implemented using robots internal sensor and external RFID data, including the functionality to automatically calibrate each node's position and initialize initial network setup. The testbed took slightly more than two months to design and construct from scratch into a working form (including waiting for shipment of parts), and per-node cost has been kept to around $1000, even though cost reduction was not the primary objective.

Acknowledgment

We'd like to thank members of the Stony Brook's *MiNT* group, in particular Rupa Krishnan, Jui-hao Chiang, and Tzicker Chiueh, for their active support in helping construct *MiNT-2* from scratch. This work is supported in part by the National Science Foundation through grants CNS-0751161 and CCF-0649252.

References

[1] Information Sciences Institute, "The Network Simulator – NS-2 http://www.isi.edu/nsnam/ns/."

[2] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks," in *Workshop on Parallel and Distributed Simulation*, May 1998.

[3] Scalable Network Technologies Inc., "QualNet WiFi simulator," in *http://www.scalable-networks.com/products/qualnet_wifi.php*, 2004.

[4] J. Heidemann, N. Bulusu, and J. Elson, "Effects of Detail in Wireless Network Simulation," in *Proceedings of the SCS Multiconference on Distributed Simulation*, January 2001.

[5] M. Takai, J. Martin, and R. Bagrodia, "Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks," in *Proceedings of MobiHoc*, Oct 2001.

[6] J. Heidemann, N. Bulusu, and J. Elson, "Effects of detail in wireless network simulation," in *SCS Multiconference on Distributed Simulation*, January 2001.

[7] D. Maltz, J. Broch, and D. Johnson, "Experiences Designing and Building a Multi-Hop Wireless Ad-Hoc Network Testbed," in *Technical Report 99-116, School of Computer Science, CMU*, Mar 1999.

[8] H. Lunndgren, D. Lundberg, J. Nielsen, E. Nordstrom, and C. Tscudin, "A Large-scale Testbed for Reproducible Ad Hoc Protocol Evaluations," in *Proceedings of Wireless Communications and Networking (WCNC)*, 2002.

[9] Daniel Aguayo et al., "MIT Roofnet Implementation http://www.pdos.lcs.mit.edu/roofnet/design/," August 2003.

[10] P. De, A. Raniwala, S. Sharma, and T. cker Chiueh, "Mint: A miniaturized network testbed for mobile wireless research," *In Proceedings of IEEE Infocom*, 2005.

[11] P. De, "Mint: A reconfigurable mobile muli-hop wireless network testbed," Ph.D. dissertation, Stony Brook University, 2007.

[12] J. Luck and J. Ioannidis, "Roomba Internals," in *http://www.tla.org/roomba/1disass.html*, January 2003.

[13] iRobot Corporation, "iRobot create programmable robot, http://www.irobot.com/sp.cfm?pageid=305."

[14] A. Raniwala, K. Gopalan, and T. cker Chiueh, "Centralized Channel Assignment and Routing Algorithms for Multi-Channel Wireless Mesh Networks," in *ACM SIGMOBILE Mobile Computing and Communications Review*, April 2004.

[15] JFW Industries, Inc., "Fixed Attenuators http://www.jfwindustries.com/Cat2000/FixedAttenuators-Terminations.pdf."

[16] D. Johnson, T. Stack, R. Fish, D. Flickinger, L. Stoller, R. Ricci, and J. Lepreau, "Mobile Emulab: A Robotic Wireless and Sensor Network Testbed," in *Proceedings of IEEE INFOCOM 2006*, April 2006.

[17] J. Tourrilhes, "Wireless Tools for Linux http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html," May 2004.

[18] P. De, A. Raniwala, S. Sharma, and T. cker Chiueh, "Design considerations for a multi-hop wireless network testbed," *IEEE Communication Magazine*, Oct 2005.

[19] P. De, R. Krishnan, A. Raniwala, K. Tatavarthi, N. A. Syed, J. Modi, and T. cker Chiueh, "Mint-m: An autonomous mobile wireless experimentation platform," *In Proceedings of Mobisys*, 2006.

[20] D. Raychaudhuri, "ORBIT: Open-Access Research Testbed for Next-Generation Wireless Networks," in *As proposal submitted to NSF Network Research Testbeds Program*, May 2003.

[21] 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*, Mar 2005.

[22] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in *Proc of OSDI'02, Boston, MA*, Dec. 2002, pp. 255–270.

[23] G. Judd and P. Steenkiste, "A software architecture for physical layer wireless network emulation," in *Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization*. ACM New York, NY, USA, 2006, pp. 2–9.

[24] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *Computer*, vol. 34, no. 8, pp. 57–66, Aug 2001.

[25] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2000, pp. 32–43.

[26] R. J. Orr, "SmartFloor, http://www.cc.gatech.edu/fce/smartfloor/."

[27] J. Bohn, "Prototypical implementation of location-aware services based on a middleware architecture for super-distributed rfid tag infrastructures," *Personal Ubiquitous Comput.*, vol. 12, no. 2, pp. 155–166, 2008.

[28] S. Hinske and M. Langheinrich, "An RFID-based Infrastructure for Automatically Determining the Position and Orientation of Game Objects in Tabletop Games."

[29] T. R. Henderson, S. Roy, S. Floyd, and G. F. Riley, "The NS-3 Project http://www.nsnam.org/."