Early Results on Hydra: A Flexible MAC/PHY Multihop Testbed

Ketan Mandke, Soon-Hyeok Choi, Gibeom Kim, Robert Grant,
Robert C. Daniels, Wonsoo Kim, Robert W. Heath, Jr., and Scott M. Nettles

***Abstract*— Hydra is a flexible wireless network testbed being developed at UT Austin. Our focus is networks that support multiple wireless hops and where the network, especially the MAC, takes advantage of sophisticated PHY techniques, such as OFDM and MIMO. We argue that for this domain simulation alone is not adequate and that working prototypes are needed to validate algorithms and protocols. Hydra nodes consist of a flexible RF front-end and a general purpose machine with a software based MAC and PHY. Using the frameworks of the Click modular router and GNU Radio and coding in C++ makes it relatively easy to implement working prototypes of cross-layer designs that require custom MACs and PHYs. We present the architecture and implementation of Hydra, as well as a preliminary cross-layer design experiment for a rate-adaptive MAC. These early results show Hydra is a capable prototyping tool for wireless network research.**

***Keywords*– wireless testbed, prototype, cross-layer design**

The impact of wireless communication and networking is obviously significant and growing. Today most communications take a single wireless hop from a node to a cell tower or access point. In future systems though, it is likely that architectures in which data may take multiple wireless hops will be widely adopted. For example, in next generation cellular systems two-hop relay architectures may be used to overcome coverage problems. Other prominent examples include mesh networks, ad hoc networks, many varieties of sensor networks, etc. In general, networks with multiple wireless hops couple networking issues more closely to those of wireless transmission, in part simply because of the implied need to make routing decisions over variable wireless channels. They also present opportunities for cross-layer optimizations. For example, the media access control (MAC) layer may benefit from using special physical (PHY) layer techniques to avoid interference among nodes on a multihop path. Our research is broadly focused on such multihop networks where close coordination between the PHY and the network layers, especially the MAC, is important. Our ultimate goal is to harness the many increasingly sophisticated PHY techniques as well as a general understanding of wireless communication to create wireless networks that are better along a number of dimensions, including capacity, bandwidth, latency, reliability, and cost.

This material is based in part upon work supported by the National Science Foundation under grants EIA-0322957, CNS-0435307, and CNS-0626797, the Air Force Research Lab under grant numbers FA8750-06-1-0091, and FA8750-05-1-0246, the Office of Naval Research under grant number N00014-05-1-0169, and the DARPA IT-MANET program, Grant W911NF-07-1-0028.

Most wireless networking research uses simulation as its major validation technique, but there is significant evidence that this is problematic [1]. As we argue in Section I, this is particularly true when sophisticated PHYs and MACs are used to communicate over real wireless channels. Our claim is that to achieve a high degree of realism, we must build working prototypes of the systems of interest and evaluate them using realistic channels, either by actually transmitting packets over the air, or at the least by using sophisticated channel emulation.

The Hydra testbed is a concrete realization of this approach. The most important goal of Hydra is to facilitate experimentation with state-of-the-art PHYs, MACs, and other network protocols and their interactions, and to do so on working hardware over realistic channels. As such, Hydra places a premium on the flexibility of implementation of the PHY, MAC, and other network software. Each Hydra node is composed of an open-source reconfigurable radio frequency (RF) front-end connected to a general purpose computer in which the PHY, MAC, and other network functionality are implemented in software. In particular, we use the Universal Software Radio Peripheral (USRP) board developed by Ettus Research [2] to implement the RF front-end of Hydra, our PHY is implemented in C++ using the GNU Radio framework [3], and our MAC is implemented in C++ using the Click modular router framework [4]. Click also provides wireless routing and a connection to the Linux TCP/IP stack, which facilitates end-to-end testing.

The use of C++ and several open-source frameworks promotes a further goal of making Hydra accessible to researchers who may not wish or be able to implement high performance hardware. The low cost of our software-defined approach means that it is feasible to have a significant number of Hydra nodes; and we believe the low cost and open-source nature of the Hydra platform will facilitate duplication of Hydra outside of UT Austin. Our hope is that the end result will be a system that we and other researchers can use effectively to validate a wide variety of cross-layer algorithms and protocols.

Section I further motivates our approach and discusses other prototypes and testbeds, thus placing Hydra in context with other efforts. We describe our system architecture and implementation in Section II. As an example of using Hydra for cross-layer design, Section III describes results from a rate-adaptive MAC protocol implemented using Hydra. Finally, Section IV considers Hydra's future and concludes.

I. Simulation, Emulation, Testbeds, and Hydra

Most wireless network protocols cannot be fully investigated purely analytically. In practice, protocols are implemented as part of their design and verification. Typically implementation is done in a network level simulator, for example, NS-2 [5] or OPNET [6]. Unfortunately, Kurkowski showed that a majority of such simulations reported in one high quality conference had serious methodological problems [1]. Even with careful methodology, it is very difficult for network simulators to have both acceptable performance and accurate implementations or models of radios with new PHY techniques, such as OFDM or MIMO. Modelling complex channels with shadowing, small scale fading, frequency selectivity, etc. is also difficult. Another limitation is that most network level researchers lack expertise in the issues of wireless communication. At the same time, physical layer researchers may have high quality simulations of the PHY and channel, but these simulations are too costly to be used as part of a network simulation. Further, even these simulations are likely to make assumptions that may not be met by real hardware, for example, that various aspects of the system are perfectly synchronized.

Another approach is to use actual hardware radios (perhaps driven by a network simulator), but to emulate the channel. Like pure simulation, a significant advantage of this approach is reproducibility and control of the channel. Further, since such an approach uses real radios, the accuracy of this aspect of the system is not in doubt. In [7] the authors emulate the wireless channel between a small number of nodes. Because of the performance requirements to do such real-time emulation, the authors used FPGAs and DSPs to implement their emulation system. In [8] the authors describe TWINE, a scalable heterogeneous emulation system. This system simulates the wireless channel as well as the MAC and PHY protocol layers on a general purpose processor.

Testbeds with real world deployment of standard off-the-shelf wireless nodes are increasingly important. These wireless network testbeds (such as [9]–[13] and references therein) are used to investigate questions about network interactions and performance. Typical applications include mesh networks, multihop routing, link quality measurements, and mobility experiments. A chief advantage here is that the system is "real". This does limit the level of reproducibility that can be achieved. For our purposes, the chief disadvantage is that although limited MAC customizability may be possible, these systems generally have a fixed PHY, which limits their utility in investigating many of the problems of interest to us.

There are also PHY prototypes/testbeds (such as [14]–[18] and references therein), which are typically used to investigate the PHY itself as well as point-to-point link performance. Applications include MIMO channel measurements, implementing/prototyping new standards, and real-time implementation of space-time codes and MIMO algorithms. These prototypes are generally designed around high-performance hardware such as FPGAs, ASICs, DSPs, and high-speed DA/AD converters. The obvious advantage is that state-of-the-art PHY techniques are employed, although the hardware used creates a significant barrier to implementation and a high per-node cost. Most PHY prototypes are also not coupled to experimental MAC and network software.

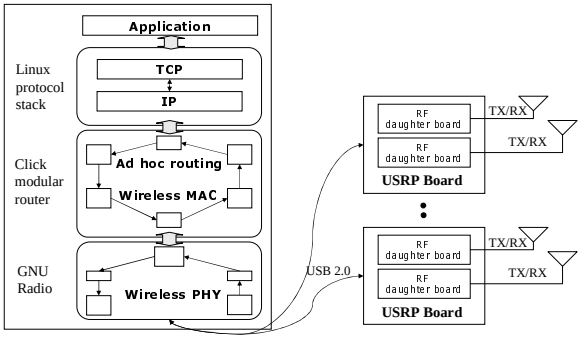

Fig. 1. Block diagram of a Hydra node.

All of the approaches above have certain tradeoffs. By choosing a design that lies in the intersection of the network and PHY testbeds, we hope that Hydra can be used to answer questions that none of the current approaches can address. Our goals are to investigate PHY, MAC, and network issues in a multihop configuration. Because most of the system consists of software, implementation barriers are significantly reduced, and we have a great deal of flexibility to change any or all parts of the system. Additionally, the use of low-cost USRP hardware means that it is cost effective to have a significant number of nodes. These factors give us hope that other researchers can replicate Hydra. The chief disadvantage is that end-to-end link performance is limited, not only by the computation needed by PHY algorithms, but also by the limited bandwidth of the USB 2.0 interface which connects the general purpose computer to the USRP front-end. Hydra is also compatible with the emulation approach. Hydra nodes can be connected to hardware emulators as they become available. In addition, we have implemented a software channel emulator that applies channel models to the complex baseband samples generated by the PHY, normally sent to the RF front-end. Finally, we hope experience with a working hardware prototype will make it possible to improve the quality of the simulations that will remain an important part of any protocol validation.

II. System Architecture and Implementation

This section describes the hardware and software used to implement Hydra. Figure 1 shows the block diagram of a Hydra node. Each node is composed of a general purpose PC (GPP) and a programmable RF front-end [2]. The RF front-end performs some basic filtering as well as upconversion and downconversion and connects to the GPP using USB 2.0. All other aspects of Hydra are implemented in software on the GPP. Although an FPGA or ASIC implementation would provide better performance, using a high-level programming language (C++) on a GPP makes our system easy to change, flexible, and programmable by researchers with little or no 'hardware' background. Additionally, the cost for each Hydra node is only $1250 USD plus the cost of a GPP.

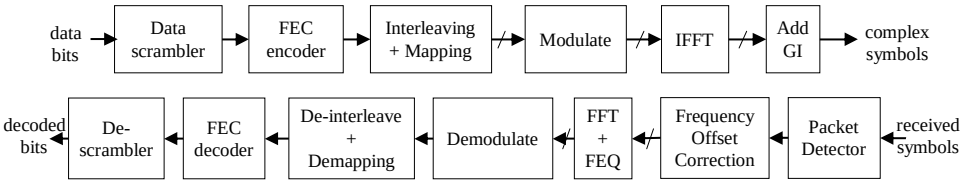

Fig. 2. Block diagram of transmit and receive chain for a single-carrier OFDM physical layer based on IEEE 802.11a [19].

A. The RF Front-End

Hydra's RF front-end is the USRP board developed by Ettus Research for use with GNU Radio [2]. The USRP is comprised of a baseband board and up to two RF daughter cards. The baseband board is composed of a USB 2.0 controller, a million gate FPGA, and multiple high-speed DA and AD converters. The FPGA provides control functionality, FIFO data queueing, as well as decimation/interpolation filtering. The baseband board's four high-speed DA and AD converters operate at 128 and 64 mega-samples-per-second respectively. The DA/AD converters also have programmable gain amplifiers controlled by the FPGA. All of the control and data manipulation functionality of the baseband board can be controlled in software running on the GPP; and all data and control signals are communicated to the front-end over USB.

GNU Radio defines a flexible API for the front-end that also supports reconfiguration of the RF daughter cards. The daughter cards are frequency agile radio transceivers with programmable gain control. They can be swapped to allow transceiver operation in various frequency bands (e.g. 400-500 MHz, ISM band, etc.). Additionally, the front-end features a fully synchronous design that allows multiple daughter cards to be synchronized providing support for research in multiple antenna systems (i.e. MIMO systems).

B. The Physical Layer

Hydra uses the GNU Radio framework [3] for PHY implementation. This open-source software framework handles all hardware interfacing, multithreading, and portability issues, thus freeing the implementor to focus on the signal processing aspects of the software radio design. To compose a protocol in GNU Radio, users first build signal processing blocks in C++ and then connect these blocks together using the Python language to form a flow graph. In addition to the relative ease of programming in C++, this approach makes it easy to transition blocks that are part of a PHY simulator into GNU Radio. In fact, the blocks that make up Hydra were initially developed and verified as part of a LabVIEW™ simulation. GNU Radio is also an attractive development platform because of its growing community of users, who range from amateur radio enthusiasts to university researchers.

To investigate advanced physical layer concepts (such as feedback in MIMO systems, OFDMA, and coding), Hydra's PHY supports OFDM and channel coding, and support for multiple antennaes is being added. The PHY also has a very flexible interface to the MAC, implemented over UDP through a local IP connection. This low latency connection provides an efficient interface for cross-layer communication, and facilitates experiments in cross-layer design.

For our current experiments, a single-antenna OFDM physical layer (shown in Figure 2) was used. This PHY is based on the IEEE 802.11a physical layer [19]. The prototype supports physical layer data rates of up to 5.4 Mbps. This rate is limited by the bandwidth of USB 2.0, but higher bandwidth interfaces are being investigated for future revisions of the front-end [20].

C. The MAC Layer

Hydra uses the Click framework [4] for its MAC implementation. This software framework, developed at MIT, runs on a GPP and was originally created for building flexible and high performance routers. Similar to GNU Radio, packet processing elements are coded in C++ and connected together using Click's own glue language. Elements can be flexibly configured to perform tasks for packet processing such as packet classification or scheduling, and then connected together in a flow graph to compose a protocol. Click not only handles memory management and scheduling for flow graph elements, but also allows users to select and modify various scheduling algorithms. As with GNU Radio, Click is also an attractive open-source development platform because of its growing community of users, which have used Click for a wide variety of applications including modular router design, ad hoc routing, and network coding.

We have implemented several random access MAC protocols for Hydra in Click, in particular CSMA/CA and the distributed coordination function (DCF) mode of IEEE 802.11. From this experience, it is clear that implementing a slotted MAC protocol for Hydra would be straight forward. As a proof of concept for Hydra's cross-layer design potential, we have also extended the current DCF MAC design to create a rate-adaptive MAC protocol based on RBAR [21], described in detail in Section III. This cross-layer protocol will be used to investigate rate-adaptation in multihop networks.

D. Other Protocol Layers

In Hydra, the wireless ad hoc networking protocols used to manage network connectivity are also implemented in Click. The code for these network protocols was contributed to the Click codebase by the creators of Grid [22], as well as other networking researchers. Since the MAC and networking protocols are both implemented in Click, they run together in their own address space, separate from GNU Radio and the TCP/IP protocol stack (running in the Linux user and kernel address space respectively). This parallelism may provide a performance improvement in multiprocessor environments.

Click features a simple tunnelling mechanism that allows protocols to interface with the standard Linux TCP/IP stack. Thus, any application built on TCP/IP can be used with Hydra.

TABLE I

KEY PARAMETERS FOR RBAR EXPERIMENT.

System Bandwidth	2 MHz
Center Frequency	420 MHz
Maximum TX Power	12 mW
Carrier Sense Threshold	1.4 μW
Modulation	BPSK, QPSK, 16-QAM, 64-QAM
Coding Rates	$\frac{1}{2}$, $\frac{2}{3}$, and $\frac{3}{4}$
Data Rates	0.6, 0.9, 1.2, 1.8, 2.4, 3.6, 4.8, 5.4 Mbps
MAC Timing	SIFS (1 msec), DIFS (30 msec), slot time (30 msec)
MPDU	MAC/PHY: 2312/4096 bytes

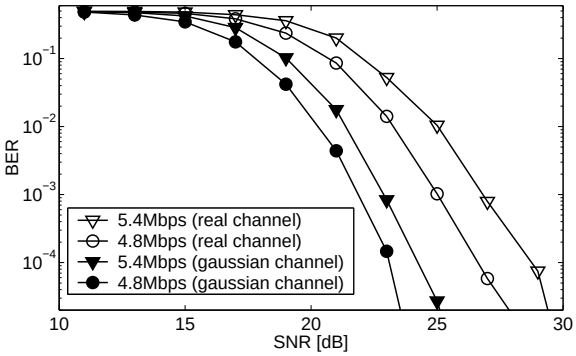

Fig. 3. Calibration for the 4.8 Mbps and 5.4 Mbps data rates.

This allows researchers using Hydra to easily test new wireless protocols with end-to-end application level experiments. Applications such as ping, ftp, and web sessions are used regularly to verify and debug the operation of Hydra.

III. A CROSS-LAYER DESIGN DEMONSTRATION

To investigate how Hydra can facilitate exploration of cross-layer design, we implemented a rate-adaptive MAC protocol based on the Receiver Based Auto-Rate (RBAR) protocol [21] and performed some preliminary experiments. We first discuss the protocol and then the details of the experiments.

A. The Rate-Adaptive MAC Protocol

RBAR is a MAC protocol that uses the control messages of the DCF mode of IEEE 802.11 to perform opportunistic link level rate-adaptation. The goal is to use a higher rate when the wireless channel permits. The 802.11 DCF MAC consists of a four way handshake. Request-To-Send (*RTS*) and Clear-To-Send (*CTS*) control messages reserve the "floor" around the sender and receiver prior to data transmission. After successfully decoding a data (*DATA*) packet, the receiver responds with an Acknowledgment (*ACK*) message. In RBAR, the PHY at the receiver uses the RTS message to estimate link quality between the sender and receiver. Then, the MAC extracts this link quality information from the PHY and determines the maximum rate that can be supported by the link. The receiver then piggybacks this rate information in the CTS response to the sender, which uses that rate to send the data. Cross-layer interactions are essential here, since the MAC has no direct way to determine link quality, while the PHY has no way to directly communicate with the sender.

In Hydra, the PHY estimates the signal-to-noise ratio (SNR) for each packet and then piggybacks this information to the MAC layer on the decoded packet. The flexible interface between the MAC and PHY facilitates this cross-layer communication and can be extended to accommodate other types of MAC/PHY interaction needed, for example, in opportunistic sub-channel allocation in OFDMA or interference mitigation.

B. Experimental Setup

In this simple experiment, our goal was simply to observe the rate adaptation process between two Hydra nodes. Hydra was operated in an indoor office environment. The two nodes were separated by a fixed distance. To change the wireless link quality, the transmit power of the sender node was varied over time. The MAC protocol implemented in Hydra responds to these changes in link quality by adapting the rate of the data transmission. The traffic pattern for this experiment consists of a stream of 1000 byte packets transmitted using UDP. Table I describes some of the key parameters used in the experiment.

The MAC protocol selects the data rate for the transmission from the possible rates using a threshold policy. We ran a careful calibration procedure to select the proper SNR thresholds for this policy by measuring the bit-error rate (BER) performance of the system as it varied with SNR for each data rate. Figure 3 shows the results of this calibration procedure (a plot of SNR in dB versus average BER) for the data rates of 5.4 Mbps and 4.8 Mbps over both an emulated additive gaussian noise channel and an actual wireless channel. We set SNR thresholds for our MAC protocol in order to achieve an average BER below approximately 10^{-5}. For example, the thresholds for 4.8 Mbps and 5.4 Mbps were 28 db and 30 dB respectively.

C. Experimental Results

Figure 4 shows a trace of the results of this experiment. The X-axis is the packet sequence number and the Y-axis on the left is for the received SNR in dB and the Y-axis on the right is for the transmit power in dBm. Each packet is plotted at the SNR threshold for the rate at which it was transmitted. Open circles indicate that a packet was successfully received, while a closed circle indicates the packet was dropped.

In this trace, the transmit power was decreased and then increased in steps. We see that in general when changes in power cause the received SNR to cross a threshold, our MAC protocol adapts the data rate of the transmission as expected. We also see instances of packets that are transmitted at higher or lower data rates than most packets at the same power level. These are examples of how the algorithm can adapt to short term fluctuations in the channel. Finally, upon investigating the cause of the dropped packets, we found they occurred when the channel worsened between the transmission of the RTS and the data, resulting in a packet sent at rate unsuitable for the channel. Thus even with this simple trace we see the complexities of studying such a protocol in the real world.

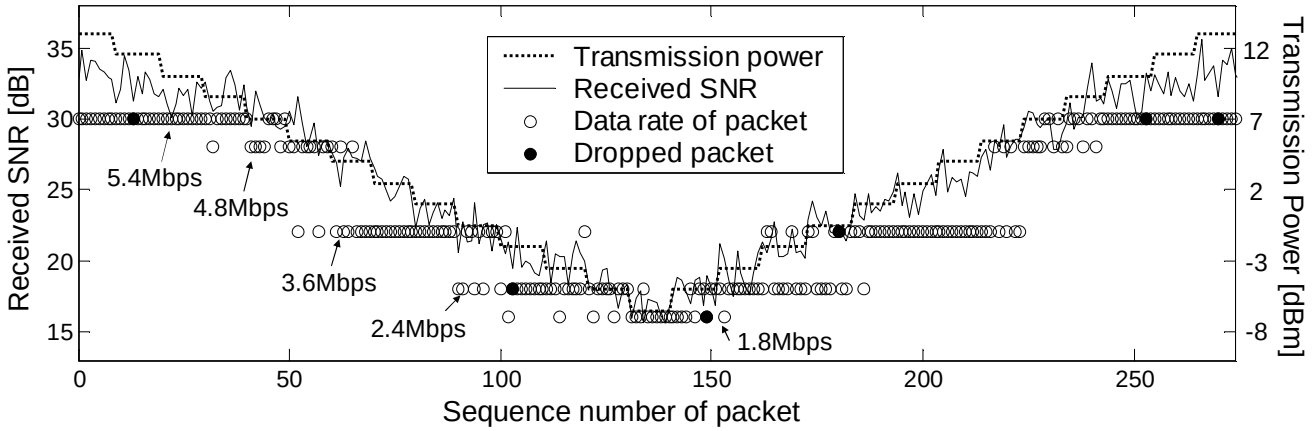

Fig. 4. Trace for experiment with rate-adaptive MAC protocol.

IV. Conclusion

We have presented early results about the Hydra testbed. In the near term we expect to make several advances. On the PHY front the most important is incorporating multiple antenna (i.e. MIMO) techniques. The blocks for this are currently being debugged in simulation and will be incorporated into Hydra's GNU Radio framework. We are also undertaking a detailed experimental evaluation of our rate adaptive MAC. Longer term we are developing protocols that incorporate a variety of channel feedback schemes and we are looking at using OFDMA to allow the MAC to schedule concurrent transmissions of potentially interfering nodes on orthogonal subcarriers. There are a wealth of other possibilities.

Hydra has been introduced as a flexible, low cost testbed that has moderate data rates and facilitates low development time and the ability to make interrelated changes in both the PHY and network software. The successful implementation of an experimental rate-adaptive MAC protocol has shown that Hydra is a capable prototyping platform filling an unusual niche for validating cross-layer designs. As wireless networks that employ multihop and MIMO technology continue to garner research interest, testbeds that are capable of supporting a wide array of research, such as Hydra, will become more important. This testbed's low cost, flexibility, and accessibility make it well suited to enable rapid prototyping in the wireless research community.

References

[1] S. Kurkowski, T. Camp, and M. Colagrosso, "MANET simulation studies: the incredibles," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 9, no. 4, pp. 50–61, October 2005. [Online]. Available: http://dx.doi.org/10.1145/1096166.1096174

[2] E. Blossom, "GNU Radio: Universal Software Radio Peripheral." [Online]. Available: http://www.comsec.com/wiki?UniversalSoftwareRadioPeripheral

[3] E. Blossom, "GNU Software Radio." [Online]. Available: http://www.gnu.org/software/gnuradio/

[4] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, 2000.

[5] "The Network Simulator - NS-2," The VINT Project. [Online]. Available: http://www.isi.edu/nsnam/ns/

[6] OPNET Technologies, Inc., "OPNET university program." [Online]. Available: http://www.opnet.com/services/university/

[7] G. Judd and P. Steenkiste, "Using emulation to understand and improve wireless networks and applications," in *Proceedings of NSDI*, Boston, MA, 2005.

[8] J. Zhou, Z. Ji, and R. Bagrodia, "TWINE: A hybrid emulation testbed for wireless networks and applications," in *Proceedings of IEEE INFOCOM*, Barcelona, Spain, 2006.

[9] B. A. Chambers, "The Grid Roofnet: A rooftop ad-hoc wireless network," Master's thesis, Massachusetts Institute of Technology, 2002.

[10] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Wireless Networks*, vol. 11, no. 4, pp. 419–434, 2005.

[11] 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 *IEEE Wireless Communications and Networking Conference*, vol. 3, 2005, pp. 1664–1669.

[12] P. De, A. Raniwala, S. Sharma, and T. Chiueh, "Design considerations for a multihop wireless network testbed," *IEEE Communications Magazine*, vol. 43, no. 10, pp. 102–109, 2005.

[13] E. Nordstrom, P. Gunningberg, and H. Lundgren, "A testbed and methodology for experimental evaluation of wireless mobile ad hoc networks," in *First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, Tridentcom*, 2005, pp. 100–109.

[14] R. Rao *et al.*, "Multi-antenna testbeds for research and education in wireless communications," *IEEE Communications Magazine*, vol. 42, no. 12, pp. 72–81, 2004.

[15] S. Caban, C. Mehlfuhrer, R. Langweiser, A. L. Scholtz, and M. Rupp, "Vienna MIMO testbed," *EURASIP Journal on Applied Signal Processing*, vol. 2006, pp. 1–13, 2006.

[16] M. Takai *et al.*, "Scalable testbed for next-generation wireless networking technologies," in *First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, Tridentcom*, 2005, pp. 162–171.

[17] P. Murphy, F. Lou, A. Sabharwal, and J. Frantz, "An FPGA based rapid prototyping platform for MIMO systems," in *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, vol. 1, 2003, pp. 900–904.

[18] R. Mostafa *et al.*, "Design and implementation of a DSP-based MIMO system prototype for real-time demonstration and indoor channel measurements," *EURASIP Journal on Applied Signal Processing*, vol. 2005, no. 16, pp. 2673–2685, 2005.

[19] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: High-Speed Physical Layer in the 5 GHz Band*, Part 11 standard ed., IEEE 802.11 Working Group, September 1999.

[20] M. Ettus, Personal communication.

[21] G. Holland, N. Vaidya, and P. Bahl, "A rate-adaptive MAC protocol for multi-hop wireless networks," in *Proceedings of the 7th annual international conference on mobile computing and networking*. New York, NY, USA: ACM Press, 2001, pp. 236–251.

[22] "The Grid Ad-Hoc Networking Project," MIT PDOS group. [Online]. Available: http://pdos.csail.mit.edu/grid/