Platforms and Testbeds for Experimental Evaluation of Cognitive Ad Hoc Networks

Kaushik R. Chowdhury, Northeastern University
Tommaso Melodia, State University of New York Buffalo

ABSTRACT

This article reviews, discusses, and classifies the existing experimental platforms and testbeds for experimental evaluation of CR ad hoc networks. The article first describes the internal organization of typical existing software defined radio platforms. Then the existing software frameworks are discussed and classified. Consequently, the challenges involved in integration of these platforms to form multihop networks are discussed, and the major efforts for large-scale testbed implementations described. Finally, limitations of existing platforms and future research challenges toward the design of accurate open testbeds that will enable sound experimental evaluation of CR networks are discussed.

INTRODUCTION

Cognitive radio (CR) networks [1] are emerging as a promising technology to improve the utilization efficiency of the existing radio spectrum. In CR ad hoc networks, the dynamic nature of the radio spectrum and the distributed nature of operations require the development of new spectrum-aware routing and resource allocation algorithms. Since spectrum occupancy is location-dependent, in a multihop path the available spectrum bands may be different at each relay node. Hence, controlling the interaction between routing and spectrum management functionalities and addressing the concerns of end-to-end reliable communication are of fundamental importance.

In this context, this article is intended to be a resource for researchers interested in advancing the state of the art in experimental research on CR ad hoc networks.

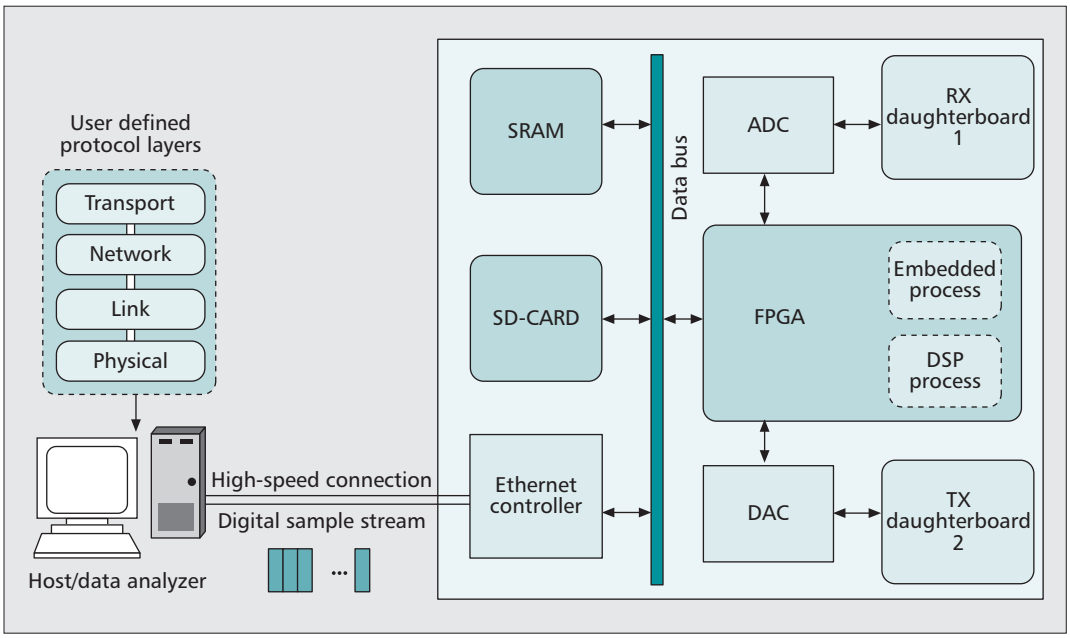
Experiments in wireless networking in general, and with software defined/reconfigurable radios in particular, are inherently complex, typically time-consuming to set up and execute, and hard to repeat by other researchers. For these reasons, simulation has been the methodology of choice for researchers in the wireless networking domain, and CR is no exception. However, there is growing awareness of the fact that current simulators make several simplifying assumptions to model many essential characteristics of real systems, such as substituting irregular transmission regions by unit disk graphs, capturing only the statistical effect of multipath propagation instead of actual ray-tracing, and assuming basic on-off models for licensed user activity, among others. It has also been argued that due to an apparent degradation in scientific standards in the conduct of simulation studies, simulation results are often questionable and of limited credibility [2]. This gap between simulated and experimental results may lead to significant differences between the behavior of the simulated system and that of the real system. It is therefore of fundamental importance to advance theoretical design and analysis of networking protocols and algorithms in parallel with sound experimental validation. For such experiments to be commonplace, the cost to set up and maintain an experimental testbed must be decreased, and a consensus has to be reached among researchers as to what should be the common requirements of experimental platforms, in terms of means for programming the devices, tools for collection and statistical analysis of experimental data, and techniques to ensure that motion is performed accurately when necessary.

For the reasons discussed above, in this article we review, discuss, and classify the existing experimental platforms and testbeds for CR ad hoc networks to shed light on how these platforms address the challenges of experimental evaluation. In addition, we discuss their limitations and highlight future research challenges.

The remainder of this article is organized as follows. In the next section we discuss the general architecture of a single CR device. The software tools and operating systems running on these devices are described in the following section, while the underlying hardware platforms are given next. We then describe the integration of these devices in experimental testbeds. The key challenges in this area and future research directions are then summarized, and in the final section we conclude the article.

The work of Tommaso Melodia was supported in part by the U.S. Air Force Research Laboratory under Grant FA8750-08-1-0063.

0163-6804/10/$25.00 © 2010 IEEE



The host contains implementation of the communication protocol stack, which handles data at the packet level, and of the physical layer functionalities. The output of the processing on the host is a digital sample stream that is transferred to the CR device for further processing.

Figure 1. *Internal organization of a CR node.*

Architecture of a CR Node

In this section we describe the architecture of a CR device, discussing the key component block and the factors that determine their design choices. As illustrated in Fig. 1, typical existing platforms are based on a split architecture, where a reconfigurable generic radio device is connected to a host, typically a desktop or laptop computer. The key resource allocation and digital signal processing (DSP) operations are undertaken on the host, which is connected to the CR device through a high-speed connection (e.g., USB or Gigabit Ethernet). The host contains implementation of the communication protocol stack, which handles data at the packet level, and of the physical layer (PHY) functionalities. The output of the processing on the host is a digital sample stream that is transferred to the CR device for further processing.

Reconfigurable Board

A typical CR device consists of a motherboard with plug-in transceivers that permit incoming radio frequency (RF) signals to be digitized and generates outgoing RF signals from the digital sample stream sent by the host computer. The platform typically includes an embedded field-programmable gate array (FPGA), allowing processing in hardware of the samples on the ingress and egress paths. Interchangeable daughterboards can cover different frequency ranges. Digital/analog conversion is realized either on the FPGA or on dedicated hardware. For example, the Universal Software Radio Peripheral 2 (USRP2), described in detail later, mounts a specialized 400 Msamples/s 14-bit digital-to-analog converter (DAC) and a 100 Msamples/s 16-bit analog-to-digital converter (ADC).

Clearly, DACs and ADCs may become performance bottlenecks for the radio. Specifically, the speed of the ADC determines how large a bandwidth can be sampled instantaneously (for spectrum sensing or decoding purposes), while the performance of the DAC limits the analog bandwidth of the transmitted signal. In addition to this, the bus connecting DACs/ADCs to the processing unit may also limit the effective data rate and sensing capabilities of the radio. This, along with frequency and capabilities of the processing unit, may limit the complexity of processing and resource allocation algorithms that can be executed in real time to control the behavior of the radio.

A soft-core processor may be implemented on the FPGA, shown in the diagram as an *embedded processor*, to handle internal board operations (e.g., controlling arbitration on the data bus). For example, in USRP2 a 32-bit soft-core processor (AeMB) is implemented on the FPGA and handles most of the internal USRP2 operation. Note that in the basic USRP2 configuration, the AeMB does not perform any DSP functions; these are done purely in software in the DSP RX and TX pipelines on the host. Other products, such as the DM6446 DSP-Virtex-4 FPGA-based software radio developed by Lyrtech, have an 8–48 kHz stereo audio codec implemented on the board itself. Thus, a simple DSP co-processor may also be implemented in software in the FPGA to aid various physical layer tasks. Additionally, a CR platform will embed fast access memory (on the order of a few Mbytes), which can be implemented on the FPGA itself; and larger low-cost memory for long-term storage (e.g., through an external SD card).

Software Platforms

User-friendly software platforms are critical for easy and rapid device programming. As CR research is highly interdisciplinary, drawing from signal processing, networking, machine learning, and other engineering and computer science disciplines, the development of intuitive

	Software	Hardware only	Hybrid
Academic and Military	MIRAI-SF [4] GNU-Radio [3] SCA/OSSIE [5]		BEE2 [7, 8] WINC2R [9] WARP [10]
Industry		USRP and USRP2 [6]	KNOWS [11] SORA [12]

Table 1. *Existing software, hardware-only, and hybrid device-level research.*

design and development tools for a diverse set of users is a challenge. In CR the key functionalities at all layers of the protocol stack, including physical layer functions such as modulation/demodulation, detection, coding, and filtering, are implemented in software. The main time-sensitive signal processing and communication functionalities are written in a compiled language, which allows faster runtime performance, while a higher level language may provide tools for interconnecting basic blocks in a user-friendly fashion, or setting the configuration parameters, and scenario information. Specifically, for CR operation, the software must also provide a rich set of signal processing libraries, interaction with the hardware components embedded in the hardware platform, and easy extension and upgradeability. Higher-layer functionalities, including medium access control, routing, and transport protocols, are implemented separately from the physical layer functionalities, and often run on a separate host connected to the radio, as discussed in further detail in the previous section.

In the following we describe the main existing software frameworks for CR (Table 1).

GNU Radio

GNU Radio is an open source project that provides a free software toolkit for developing a software defined radio running on Linux on standard computers or over Cygwin for the Windows platform [3].

The programming environment is organized around the principle of constructing a signal processing graph that describes the data flow in the software defined radio. This graph is executed in an integrated runtime system. The vertices of the graph represent signal processing blocks and the edges are the data flow between them. Signal processing blocks operate on data streams flowing from a number of input ports to a number of output ports specified per block.

GNU Radio frameworks are built by a combination of Python and C++. The Python code provides the signal processing graphs that outline the order in which the main component functions of the radio are executed, the data flow, a graphical user interface, and other non-performance-critical support to the user. The main signal processing blocks are written in C++, and glued together to the Python code using a tool called SWIG. This task can be simplified by using the GNU Radio Companion or GRC, which provides a graphical user interface similar to that of Simulink. The complexity of these blocks can be increased by hierarchically adding existing primitives or integrating other hierarchical blocks. This modular expansion facilitates easy prototyping, aided by an extensive library of signal processing routines (optimized filters, fast file transfers [FFTs], equalizers), modulation schemes (Gaussian minimum shift keying [GMSK], phase shift keying [PSK], quadrature amplitude modulation [QAM], orthogonal frequency division multiplex [OFDM]), task scheduling, error-correcting codes (Reed-Solomon, Viterbi, turbo codes), and timer maintenance, among others. User-initiated development is further supported by allowing the testing of the blocks in a simulated data set, while maintaining the same program that would need real RF hardware for its operation. Thus, the overall aim of this effort is to give the user complete flexibility and reconfigurable support by moving the operation of a radio into the software domain and as close as possible to the antenna.

Software Communications Architecture

The Software Communications Architecture (SCA) is an open-source implementation-independent framework for the development of software defined radios initiated by the U.S. Military. Similar to GNU Radio, the components of a software defined radio are modeled as interconnected blocks in a data flow diagram with standard interfaces that allow easy linking with user devices. SCA includes real-time embedded operating system functions, and provides multithreaded support for all software executing on the system. The framework is written in C++/Python, and the different message passing functions between blocks are undertaken by Common Object Request Broker Architecture (CORBA), with support for the Linux operating system. The component-based structure in SCA allows easy reuse of these building blocks across different radio platforms. One such SCA-based commercially available platform developed by Lyrtech allows for a powerful but small form factor radio supporting 0.2–2.0 GHz RF operation with integrated WiMAX. The general operating environment for the SCA defines a subset of the portable operating system interface (POSIX) interfaces that ensure software interoperability between different systems. The use of CORBA hides internal architectural details, such as the number and type of processors, the operating system (OS), and other aspects of the radio configuration. In the context of SCA, the term *waveform* is used to define a radio, that is, a series of transformations on the signal from the point of its generation to its eventual transmission. The framework controls each waveform, allowing these transformations to be drawn from several different sources, and yet managed and deployed under a common set of rules. SCA uses unified modeling language (UML), an industry-wide standard, to graphically represent the different software blocks and their interconnections. Ideally, SCA is designed to run on any underlying SCA compliant operating environ-

Component name	USRP	USRP2
Analog-digital converters (ADCs)	Number: 4 Speed: 64 Msamples/s, 12 bit	Number: 2 Speed: 100 Msamples/s, 14-bit
Digital-analog converters (DACs)	Number: 4 Speed: 128 Msamples/s, 14 bit	Number: 2 Speed: 400 Msamples/s, 16-bit
RF bandwidth	16 MHz	100 MHz
Host connection	USB2.0 Speed: 480 Mb/s	Gigabit Ethernet Speed: 1 Gb/s

Table 2. *Hardware differences between USRP and USRP2 [6].*

ments with a minimal amount of changes to the original code. Currently, the OSSIE software (SCA-Based Open Source Software Defined Radio) provides an implementation of a software defined radio built over the SCA framework assuming the presence of an underlying USRP board. OSSIE provides software procedures for narrowband and wideband sensing [5]. A major contribution of this project is a Graphical User Interace (GUI) tool that allows a designer to create new waveforms using standard templates, and also generates the C++/Python code that automatically handles the interactions within the SCA and the links with the CORBA middleware. Further signal processing and modulation schemes, such as QAM and PSK, or new test algorithms can be added by the user.

MIRAI CR
Execution Framework

While GNU Radio and SCA are focused on lower-layer functionalities, the MIRAI CR Execution Framework is designed to provide a full CR emulation environment, including implementation of functionalities at all layers of the communication protocol stack. The framework is designed for large-scale implementations, and provides a default implementation of TCP at the transport layer and wireless LAN backoff implementation at the link layer, thus making it suitable for multihop experiments [4]. Moreover, its multi-threaded structure allows handling of packets and events in parallel, unlike the ns-2 simulator. A key feature of MIRAI is that it allows the construction of a mixed simulated/emulated environment, where a few nodes are mere software entities (from the physical to higher layers), while others may be actual devices. The various functional components in the different layers of the protocol stack are implemented as *plug-in*, which leads to scalability. Moreover, as the simulator has a UNIX-compatible communication stack, using an actual hardware component at the physical layer simply involves replacing the plug-in with the device to be tested as long as it uses one of the supported standards. The CR-specific physical layer has software commands for changing spectrum band, modulation, frequency, bandwidth, and standards such as IEEE 802.11 a/b/g/e and IEEE 802.16. The program logic for affecting these changes is supplied by a user through a separate plug-in.

Existing CR Platforms
Universal Software Radio Platform

Several recently proposed devices and testbeds for CR networks rely on the USRP open source hardware platforms [6], which use the GNU Radio software package. Developed and distributed by Ettus Research LLC, USRP is one of the most popular commercial solutions used in the design and implementation of software defined radio systems. The URSP product family consists of the motherboards (USRP and USRP2), which contain an FPGA and interchangeable daughterboards. Together, they bridge a controller host computer with one or more antennas.

As shown in Table 2, the initial design of the USRP has been significantly improved in the USRP2 boards in terms of higher communication speeds (using Gigabit Ethernet as opposed to 480 Mb/s over USB 2.0 for host connection), better processing ability (larger FPGA including implementation of an AeMB processor core), and increased flexibility of prototyping. The USRP platform family can be interfaced with different daughterboards, which permits incoming RF signals from a possible range of DC to 5.85 GHz to be digitized, and generates outgoing RF signals from the digital sample stream sent by the host computer. The platform includes an embedded Xilinx Sparta 3-2000 FPGA, allowing processing in hardware on the samples on the ingress and egress paths. The USRP2 is endowed with 14-bit digital-to-analog converters (DACs), and 16-bit analog-to-digital converters (ADCs), both significant advances over the earlier USRP. The current configuration allows sampling portions of the spectrum of approximately 25 MHz, as opposed to 8 MHz for the previous platform. Similarly, the Gigabit Ethernet interface to the host computer allows handling signals of adequate bandwidth in real time.

The USRP2 holds a single transceiver daughterboard, and multiple USRP2s can be connected together to form a wide multiple-input multiple-output (MIMO) system (up to 8 × 8). Different daughterboards provide the USRP and USRP2 with the appropriate RF front-end for different frequency bands. In addition, a daughterboard can be a single transmitter, a single receiver, or a full transceiver. In Table 3 the frequency bands of operation of the main daughterboards provided by Ettus Research LLC are

The URSP product family consists of the motherboards (the USRP and the USRP2), which contain an FPGA, and interchangeable daughterboards. Together, they bridge a controller host computer with one or more antennas.

Component name	Function	Frequency band
BasicTX	Transmitter	1–250 MHz
BasicRX	Receiver	1–250 MHz
LFTX	Transmitter	DC–30 MHz
LFRX	Receiver	DC–30 MHz
TVRX	Receiver	50–860 MHz
DBSRX	Receiver	800 MHz–2.4 GHz
RFX400	Transceiver	400–500 MHz
RFX900	Transceiver	800 MHz–1 GHz
RFX1200	Transceiver	1.15 MHz–1.4 GHz
RFX1800	Transceiver	1.5–2.1 GHz
RFX2400	Transceiver	2.25–2.9 GHz
XCVR2450	Transceiver	800 MHz–2.4 GHz

Table 3. *Available transmitters and receivers for the USRP family [6].*

summarized. Moreover, different custom-made boards can be designed to match the specifications of the USRP and USRP2 boards.

Hybrid Platforms

In this section we describe existing devices that have both a hardware implementation and an associated software framework specific to the hardware. In all of these approaches, the basic architecture uses an FPGA board, external ADC/DAC modules, an implementation of a processor core, and configurable RF front-ends of varying bandwidths and center frequencies. Additionally, the software component generally supports the operation of the higher-layer protocol operations, such as MAC protocol implementations and enhanced support for networking among multiple devices.

Kognitiv Networking Over White Spaces — The KNOWS project is a collaborative industry effort toward building a prototype CR device [11]. The device itself has three components: a host computer, a spectrum scanner, and a frequency translator between the 2.4 GHz industrial, scientific, and medical (ISM) and 512–698 MHz UHF bands. The host maintains the control plane, and contains the implementations of the MAC and higher layer protocols. It is equipped with a standard WiFi card, and the translator downconverts the output signal to the UHF band and vice versa for the incoming signals. As each channel is defined as 6 MHz wide, the channel bandwidth of the output signal is limited to 5 MHz so that they can be contained within the standard UHF TV channels 21–51. However, these individual channels can be aggregated to form 5, 10, 15, and 20 MHz channels when contiguous channel vacancies are detected in the UHF band. The spectrum scanner is implemented in a USRP board [6] having a 50–8600 MHz TVRX receiver-only daughterboard. The scanning is set to intervals of 30 s as the TV transmissions are unlikely to exhibit sudden fluctuations. Apart from these main components, the KNOWs platform also has a GPS module in the event that a database of the locations of TV stations and transmission towers is available. The platform also includes an x86 embedded processor that controls the radios, obtains MAC layer control packets from the host, and parses them into instructions for configuring the RF hardware, and conversely, aggregates the raw received signals into packets that can be operated on by the host.

Software — The software implements algorithms for scanning the spectrum, detecting beacon signals/data packets sent by other CR users, and detecting the presence of incumbents; these tasks are complicated by multiple channel bandwidths. The spectrum usage history is built over time allowing improved decisions, and new time-domain techniques for detecting the start-stop time of data packets are proposed. The latter are especially needed as the channel bandwidth used by the neighboring CR users may not be known in advance, rendering the classical FFT-based techniques infeasible. A MAC layer is developed in [11] that allows the formation of a WiFi-like network architecture in the UHF band, with multiple CR users linked to a single access point.

Winlab Network Centric CR — The WiNC2R hardware is composed of a baseband processing module and an RF front-end. The physical layer processing is performed by a set of hardware accelerators, and the less computationally intensive functions at the MAC and higher layers are performed by an array of data processors. A unique feature of this platform is that the physical layer parameters and the MAC functionality can be changed on a per-packet basis, thereby allowing rapid response to sudden changes in the condition of the network. The incoming signals are digitized at a rate of 125 Msamples/s before further processing by a Xilinx 4FX12 FPGA that resides on the board. The settings of the transmission parameters, including frequency selection, and alternating between transmission and receiving functions are controlled by the instructions residing in the FPGA. The RF front-end is highly configurable with a baseband of 0–500 MHz, and up-conversion is supported in the range 30 MHz to 6 GHz. Moreover, multiple (virtual) OFDM-based radio standards are supported that can be used up to speeds of 10 Mb/s. The architecture in WiNC2R follows a virtual flow pipeline concept, in which a wide range of functions can be inserted in the data flow, thereby allowing the actual selection of their sequence to be undertaken at runtime. Unlike fixed sequence operations that must adhere strictly to a predecided order of function calls and timing requirements, the proposed approach allows greater flexibility by changing the nature of the operations on a per-flow basis. Additional synchronization mechanisms ensure that the functional units are shared without overlapping between different flows.

Software — The WiNC2R software uses GNU radio [3] as an underlying base for developing the physical and MAC layers. The prototyping of the higher layer functions is based on the CogNet software package [9], which provides the control and management planes, support for collaborative physical layer functions, dynamic spectrum coordination, adaptive MAC protocols, clustering and group formation, and cross-layer design. Most important the processing flow is driven by events rather than by the program counter as in the stored program paradigm. The events driving the flow can be time- or data-based, and both can be generated by the environment or as the result of internal processing. CogNet also provides for a global control channel that could be used for bootstrapping, device discovery, addressing, and data path setup, among other functions in a CR ad hoc network.

Berkeley Emulation Engine 2 — The BEE2 is a generic multipurpose emulation platform that supports up to 500 giga-operations per second by distributing the load among its component FPGAs. The BEE2 can be coupled with external hosts, and independently designed radio modems via high-speed optical connects. The BEE2 hardware architecture has a computational module and a configurable RF frontend [7, 8]. The computational module consists of five Vertex-IIPro 70 FPGAs, four of which are used for processing; the fifth provides the control plane with intra-FPGA connection links supporting up to 20 Gb/s. A full IP protocol stack that allows easy interfacing with external computers is provided on the control FPGA. Thus, the platform emulates a virtual FPGA of five times the capacity of a single one. Each FPGA has access to 4 Gigabyte of memory and embeds a PowerPC 405 core for facilitating fast reconfiguration of the communication modules. These FPGAs can connect to the external world using serial multigigabit interfaces. For CR experiments, a RF front end, fully programmable over a range of 80 MHz, in steps of 20 MHz channels, in the 2.4 GHz ISM band, is developed that can be connected to the BEE2. An additional Vertex-IIPro20 FPGA on the radio front end is used to implement control logic for the radio, maintaining accuracy of the analog components by periodic calibration and real-time access to registers onboard the radio. The use of optical cables connecting the processing components on the FPGA to the RF frontend allows greater physical separation between the two and, consequently, better isolation from self-induced interference.

Software — The BEE2 software uses the BORPH OS, which extends a standard Linux kernel to view the different FPGAs as a single unified computational resource. This user process on BORPH could be either a program that is running on a dedicated processor or a set of instructions that are executed through hardware configuration on the FPGA. The resources utilized by this OS to spawn hardware processes can range from an entire FPGA to a predecided region within the FPGA. Furthermore, the UNIX file system abstraction allows users to interact with the FPGA at runtime without manually constructing software interfaces. Further support in implementing key blocks is provided by allowing their design to be undertaken in Simulink and, finally, compilation and mapping to the FPGA on the BEE2 through a tool developed using Xilinx.

SORA — The SORA platform from Microsoft Research allows the implementation of higher-layer wireless protocol stacks that require highly precise timing, not possible through USB or Gigabit Ethernet interfaces linking the wireless module to the host computer [12]. The hardware is composed of a radio control board (RCB) that bridges an RF front-end with the host over the high-speed and low-latency PCIe bus. With this bus standard, the RCB can accommodate transfer speeds up to 16.7 Gb/s, and the specialized support for multicore processors residing in the host allows the digital signal processing to be undertaken completely at the host. The IEEE 802.11a/b/g PHY and MAC in the 2.4 and 5 GHz bands have been tested using this platform. The cost of the RCB is comparable to the existing USRP2 platform (RF frontends can be either WARP radio boards or USRP daughterboards), although additional host requirements, such as workstation-based PCs with at least a PCIe x8 or x16 slot, are specified.

Software — SORA has two key features that greatly expedite software processing tasks: First, the physical layer adaptation (e.g., choice of modulation, and spreading code among others) is undertaken via precomputed loop tables instead of deriving the optimal value through calculations. This allows the operation to be fast, and leverages the low-latency caches present in the host processors. It also makes full use of the instruction sets present in existing processors, such as single instruction multiple data (SIMD) to further accelerate the processing. Second, SORA provides a new kernel service called as *core dedication* that allocates host processor cores exclusively for real-time tasks, which guarantees the availability of computational resources. SORA requires Windows XP for its use.

WARP — The wireless open access research platform (WARP) board, developed at Rice University, has a Xilinx Virtex-4 FPGA, with the provision of supporting up to four daughterboards with RF front-ends [10]. Multiple WARP boards may be connected to the host PC via an Ethernet switch. An interesting deviation from the USRP-based architecture is the ability to construct the baseband samples in MATLAB and then store them in buffers on the FPGA on the transmitter node before beginning the transfer. A trigger signal from the host can then begin the transmission of the samples to the receiver side over the wireless channel. The WARP board supports 40 MHz of bandwidth independent of the carrier frequency. The two embedded PowerPC processors provide sufficient onboard computational power, while the 328 18-kbit block RAMs allows for fast access of data from within the FPGA. The onboard processing ability of the WARP platforms allows some time-critical tasks

The BEE2 is a generic multipurpose emulation platform that supports up to 500 giga-operations per second by distributing the load among its component FPGAs. The BEE2 can be coupled with external hosts and independently designed radio modems via high-speed optical connects.

Testbed name	Location	Access	Number of nodes	Hardware	Frequency range	Protocol support
Emulab	Univ. of Utah	Open	25 10	USRP USRP2	800–1000 MHz 2.4 & 5 GHz	CSMA/CA IP-addressable
VT-CORNET	Virginia Tech	Open	48	USRP2	100 MHz–4 GHz	In-house cognitive/policy engine
ORBIT	Rutgers Univ. Winlab	Open	11 12	USRP USRP2	50–2.2 GHz 2.4 & 5 GHz	COGNET global MAC

Table 4. *A comparison of the currently implemented multihop CR testbeds.*

to be completed within the board itself, facilitating time-sensitive operations discussed later.

Software — The key benefit of using WARP is ease of prototyping enabled by the WARPlab framework. This framework provides the software that allows controlling and programming the individual nodes from within the MATLAB workspace running on the host computer. The WARPlab framework itself has three components: The *platform studio* generates the implementations of the network protocols (assumed to be input in C/C++). The *system generator* takes the MATLAB-specified physical layer algorithms, and converts them to a hardware model for the FPGA implementation. Finally, it has a *low-level HDL and ASIP* development module that exposes the internal hardware components to the higher-layer MATLAB routines. Extensive software support is also available for WARP toward advanced networking functionalities, including carrier sense multiple access (CSMA) based protocols, spectrum management, MIMO, cooperative communication, power control, and energy-efficient transmission through published works and downloadable code.

TESTBEDS

There exists a high level of heterogeneity in the design of practical CR networks. We characterize testbeds in the following way based on the devices used, communication system, and level of heterogeneity:

- **Accessibility**: The overhead, in terms of both the material and manpower required for a thorough experimental evaluation, is often a detrimental factor in building testbeds. Several institutions provide external interfaces, where the individual devices of the testbed can be programmed by a user remotely, often over the Internet. We refer to this as open access, and strongly advocate this as the best strategy for undertaking reproducible results. Moreover, this approach extends support to the research community, who can make use of an existing investment in infrastructure.
- **Device hardware**: There are commercially available device types, as well as custom designs that may support varying number of transceivers, processors clocked at different instructions/per second, frequency ranges, and bandwidth that are supported by these transceivers and antenna types, among others. Ideally, the testbed should be highly reconfigurable in terms of dynamic spectrum selection and capable of wideband sensing over large ranges.
- **Scale**: Depending on the resource constraints, testbeds vary in the number of nodes. Our survey points to a large variation, ranging from around a dozen to nearly 50 nodes. Specifically, large testbeds allow for realistic testing of the effect of the interference caused by intra-CR network transmissions on licensed user detection, and the performance of spectrum sharing in the detected vacant bands.
- **Heterogeneity**: Most of the existing testbeds are composed of homogenous devices, and each node has similar communication and processing capabilities. However, real-world networks are likely to be highly diversified in these respects. Hence, devising a compatibility plane is needed, where individual nodes offer additional and often distinct capability over a minimum acceptable feature set. Such testbeds may also be used to test the coexistence of CR networks managed by different entities in a common spectrum pool.
- **Protocol support**: Given the high degree of reconfigurability in CR devices, developing a user-modifiable protocol stack that can run on the devices is a challenge. When research efforts are undertaken at each layer, basic implementations of the other protocols layers, not in the scope of the work, must be present. As an example, TCP implementations over CR may rely on spectrum sensing information from the physical layer. Moreover, the basic structure should be easily modifiable, as there has been an increasing trend toward developing cross-layer techniques that integrate and utilize the complete spectrum as well as network information.

In this section we describe the key features of existing testbeds with respect to the above characteristics, and point out the general research challenges that exist.

EXISTING TESTBEDS AND IMPLEMENTATIONS

In the following we describe some of the main experimental deployments of CR testbeds to date (Table 4).

Virginia Tech CR Network (VT-CORNET) — This highly reconfigurable testbed allows the evaluation of independently developed CR engines, sensing techniques, applications, protocols, performance metrics, and algorithms. The current and planned testbed capabilities include 48 USRP2-based nodes, spread over four floors of a building and equipped with a custom-made daughterboard spanning the frequency range 100 MHz to 4 GHz. Key differentiating aspects of this testbed are the use of the Software Communications Architecture (SCA) [5] software framework that assumes USRP nodes at the physical

layer, and provides the support for generating and visualizing a range of radio configurations called as waveforms.

Open Access Research Testbed for Next-Generation Wireless Networks — The ORBIT testbed has a combination of USRP (11) and USRP2 (12) that are externally addressable and placed in a grid-like arrangement, leading to a variety of standard topologies. For the USRP boards, two daughterboards are interfaced, one with a sensing-only range of 50–2.2 GHz, and another capable of sending and receiving signals in the 2.4 and 5 GHz ISM bands. Only the USRP2 has the ISM band support, and powerful quad-core host machines that fully use the increased communication speeds enabled by the platform. Apart from the USRP there are a few other FPGA-only platforms but they require the external user to have Matlab/Simulink and a Xilinx ISE license to work with these platforms.

The ORBIT testbed allows both controlled testing on an emulator as well as field experiments, and the execution code must be uploaded on the ORBIT server. Users then make a description of the properties of the networking protocol to be tested, such as packet size, transmission rate, and a list of the applications that must be installed on the nodes. The next stage involves assigning certain nodes protocol-specific roles, and addressing the handling of the dynamically changing properties based on different network conditions observed during execution of the experiment. Moreover, the ORBIT Measurement Framework (OML) provides a library for logging results in a relational database for offline analysis.

Emulab — Emulab is an open access testbed that has its center at the University of Utah and several branches spread over the world. It is widely used as an educational emulation tool as well as for research. Currently, CR experiments can be run on the USRP boards, with 13 nodes each having two 2.4 GHz transceivers, and 12 additional boards operational over the 800–1000 MHz range. Newer expansions involve 10 USRP2 boards using the 2.4 and 5 GHz bands, with a projected total number of USRP2 nodes of 30 in the near future. Emulab has several facilities to automate network actions, such as rebooting, providing routes between a given source-destination pair, creating batch experiments, and using supported OSs as well as custom OS images, among others. For experiments with a very large number of nodes, typically seen in CR ad hoc networks, a multiplexed virtual node implementation is provided that allows the use of an increase of 10 over the number of physical machines.

Research Challenges and Architectural Considerations

From an architectural point of view, the distribution of functionalities across the processing units significantly impacts the performance, flexibility, and ease of reprogramming of the CR. Moreover, as the burden of processing is moved to software, which is typically slower than hardware-only operation, the design must also be able to successfully accommodate time-critical functions of the protocols. We describe some of these challenges and solutions using the widely used USRP family of boards as an illustrative example.

Need for Moving Time-Critical Tasks to FPGA Hardware

To achieve a high level of flexibility and reprogramming, the USRP places the majority of processing (i.e., modulation) on the host CPU, where the functionality is easy to modify through a high-level language (referred as host-PHY in [13]). However, increased performance can be achieved by implementing low-level processing in the radio hardware on the FPGA. For example, this architecture is the base of the WARP platform, which places the PHY and MAC layers on the radio hardware [10]. This approach is also referred to as NIC-PHY [13].

Similarly, MAC functionalities can be implemented either near the radio hardware for performance or near the host for flexibility. Experiments on USRP radio report that roundtrip times between the device driver on the host and the FPGA is about 300 ms for 4 kbytes of data, with relatively modest jitter [13–15]. The roundtrip from GNU Radio is about double, but with significantly more jitter [13]. As a result, a host-based MAC protocol will not be able to precisely control packet timing or implement small, precise interframe spacings, which will hurt the performance of basically each and every existing MAC protocol. Hence, time-critical radio or MAC functions should not be placed on the host CPU. No latency measurements are currently available for the USRP2. While the USB connection has been replaced by a GbE link, our preliminary measurement experiments reveal a latency between the host and radio transmission on the order of milliseconds. In particular, we measured a carrier sense delay of 2.1 ms with variance $7 \cdot 10^{-7}$ over 1000 measurement experiments. This is still certainly inappropriate for accurate MAC timing, which requires a precision in the order of microseconds. In addition, the host-to-host latency on a point-to-point link is approximately 5.2 ms with variance $0.4 \cdot 10^{-6}$.

Split MAC Design with Host and FPGA Implementations

While approaches based on implementing core MAC functionalities on FPGAss are possible, at the expense of cost and flexibility, different approaches may be possible. Key time-critical MAC functionalities should be implemented on the radio hardware, particularly precise scheduling in time, carrier sense, and backoff, while trying to maintain the flexibility offered by a high-level programming language. In fact, while it would seem natural to implement these on FPGAs, the available space on an FPGA is limited, and FPGA programming requires a steep learning curve and skills that may not be available to networking researchers. An alternative is

Increased performance can be achieved by implementing low-level processing in the radio hardware on the FPGA. For example, this architecture is the base of the WARP platform, which places the PHY and MAC layers on the radio hardware. This approach is also referred to as NIC-PHY.

Experimental evaluation is undoubtedly complex, especially in the highly dynamic CR ad hoc network scenarios. However, there is a visible need of testbeds for successfully and convincingly demonstrating new ideas in this nascent area of research.

to implement time-sensitive functionalities on soft-core processors on the FPGA. A soft processor is a CPU developed using logic synthesis that can be implemented in a semiconductor device containing programmable logic such as an FPGA. In this way the FPGA emulates, using logic circuitry, a processor that can be used conventionally. Currently, there are a number of soft cores available on the market, such as MicroBlaze by Xilinx, Niosr II by Alterar, LEON3 by Aeroflex Gaisler, and OpenRISC 1200 and AeMB from OpenCores. AeMB is a 32-bit reduced instruction set computer (RISC) architecture soft processor core with 32 general-purpose registers, an arithmetic logic unit (ALU), and an instruction set which is very similar to the RISC-based DLX architecture. The MicroBlaze soft processor has been implemented with an IEEE-754-compatible single-precision floating-point unit (FPU), which connects directly to the MicroBlaze instruction execution pipeline and is clocked at 50 MHz, thus enabling precise control with a granularity of 20 ns.

The core time-sensitive MAC functionalities can then be implemented in a high-level language. In this way, one can keep the advantage of high-level language constructs, while obtaining the performance gains given by implementation of time-critical functionalities close to the radio. Split MAC design can also be implemented by using hard CPU cores available in some platforms. This approach is, for example, used in WARP. Alternatively, the host CPU may be connected over high-speed serial buses (as is the case for SORA and WINC2R).

Conclusions

We have presented a thorough classification and survey of existing software and hardware tools for experimental evaluation of CR ad hoc networks. We believe that support for wide spectrum sensing/transmission abilities at the RF frontend, embedded and reconfigurable processing ability through flexible FPGA programming, prototyping support by using graphical tools, and scalability are critical considerations for device design. Moreover, the challenges involved in integration of these platforms to form multihop networks are discussed, and the major efforts for large-scale testbed implementations described. Experimental evaluation is undoubtedly complex, especially in highly dynamic CR ad hoc network scenarios. However, there is a visible need for testbeds to successfully and convincingly demonstrate new ideas in this nascent area of research.

References

[1] I. F. Akyildiz, W.-Y. Lee, and K. Chowdhury, "CRAHNs: CR Ad Hoc Networks," *Ad Hoc Net. J.*, vol. 7, no. 5, July 2009, pp. 810–36.

[2] S. Kurkowski, T. Camp, and M. Colagrosso, "MANET Simulation Studies: The Incredibles," *ACM Mobile Comp. Commun. Rev.*, vol. 9, no. 4, Oct. 2005, pp. 50–61.

[3] GNU Radio Project; http://www.gnuradio.org

[4] K. Ishizu *et al.*, "Adaptive Wireless-Network Testbed for CR Technology," *Proc. ACM WiNTECH*, Los Angeles, CA, Sept. 2006.

[5] C. R. A. Gonzalez *et al.*, "Open-Source SCA-Based Core Framework and Rapid Development Tools Enable Software-Defined Radio Education and Research," *IEEE Commun. Mag.*, vol. 47, no. 10, Oct. 2009, pp. 48–55.

[6] Ettus Research LLC; http://www.ettus.com/

[7] C. Chang, J. Wawrzynek, and R. Brodersen, "BEE2: A High-End Reconfigurable Computing System," *Proc. IEEE Design Test Comp.*, vol. 22, no. 2, Mar.–Apr. 2005, pp. 114–25.

[8] S. Mellers *et al.*, "Radio Testbeds using BEE2," *Proc. Asilomar Conf. Signals, Sys., Comp.*, Pacific Grove, CA, Nov. 2007.

[9] D. Raychaudhuri *et al.*, "Cognet: An Architectural Foundation for Experimental Cognitive Radio Networks within the Future Internet," *Proc. IEEE/ACM MobiArch*, San Francisco, CA, Dec. 2006.

[10] WARP, "Wireless Open-Access Research Platform," Rice University; http://warp.rice.edu/index.php

[11] P. Bahl *et al.*, "White Space Networking with Wi-Fi like Connectivity," *Proc. ACM SIGCOMM*, Barcelona, Spain, Aug. 2009.

[12] K. Tan *et al.*, "Sora: High Performance Software Radio Using General Purpose Multi-Core Processors," *Proc. Usenix NSDI*, Boston, MA, Apr. 2009.

[13] G. Nychis *et al.*, "Enabling MAC Protocol Implementations on Software-Defined Radios," *Usenix NSDI*, Apr. 2009.

[14] T. Schmid, O. Sekkat, and M. B. Srivastava, "An Experimental Study of Network Performance Impact of Increased Latency in Software Defined Radios," *Proc. ACM WiNTECH '07*, New York, NY, 2007, pp. 59–66.

[15] S. Valentin, H. von Malm, and H. Karl, "Evaluating the GNU Software Radio Platform for Wireless Testbeds," Tech. Rep. TR-RI-06-273, Feb. 2006.

Biographies

KAUSHIK R. CHOWDHURY [M'09] (krc@ece.neu.edu) is Assistant Professor in the Electrical and Computer Engineering Department at Northeastern University, Boston, MA. He graduated with B.E. in Electronics Engineering with distinction from VJTI, Mumbai University, India, in 2003. He received his M.S. in Computer Science from the University of Cincinnati, OH, in 2006, and Ph.D. from the Georgia Institute of Technology, Atlanta, GA in 2009. His M.S. thesis was given the outstanding thesis award jointly by the ECE and CS departments at the University of Cincinnati. He has also won the BWN researcher of the year award during his Ph.D. in 2007, and the best paper award in the Ad Hoc and Sensor Networks symposium at the IEEE ICC conference in 2009. His expertise and research interests lie in wireless cognitive radio ad hoc networks, energy harvesting, and multimedia communication over sensors networks.

TOMMASO MELODIA [M'07] (tmelodia@buffalo.edu) is an Assistant Professor with the Department of Electrical Engineering at the University at Buffalo, The State University of New York (SUNY), where he directs the Wireless Networks and Embedded Systems Laboratory. He received his Ph.D. in Electrical and Computer Engineering from the Georgia Institute of Technology in June 2007. He had previously received his "Laurea" (integrated B.S. and M.S.) and Doctorate degrees in Telecommunications Engineering from the University of Rome "La Sapienza," Rome, Italy, in 2001 and 2005, respectively. He is the recipient of the BWN-Lab Researcher of the Year award for 2004.%, and he is the author of about 40 publications in leading conferences and journals on wireless networking. He coauthored a paper that was was recognized as the *Fast Breaking Paper in the field of Computer Science* for February 2009 by Thomson ISI Essential Science Indicators. He is an Associate Editor for the Computer Networks (Elsevier) Journal, Transactions on Mobile Computing and Applications (ICST) and for the Journal of Sensors (Hindawi). He has served in the technical program committees of several leading conferences in wireless communications and networking, including IEEE Infocom, ACM Mobicom, and ACM Mobihoc. He was the technical co-chair of the Ad Hoc and Sensor Networks Symposium for IEEE ICC 2009. His current research interests are in modeling and optimization of multi-hop wireless networks, cross-layer design and optimization, cognitive radio networks, multimedia sensor networks, underwater acoustic networks.